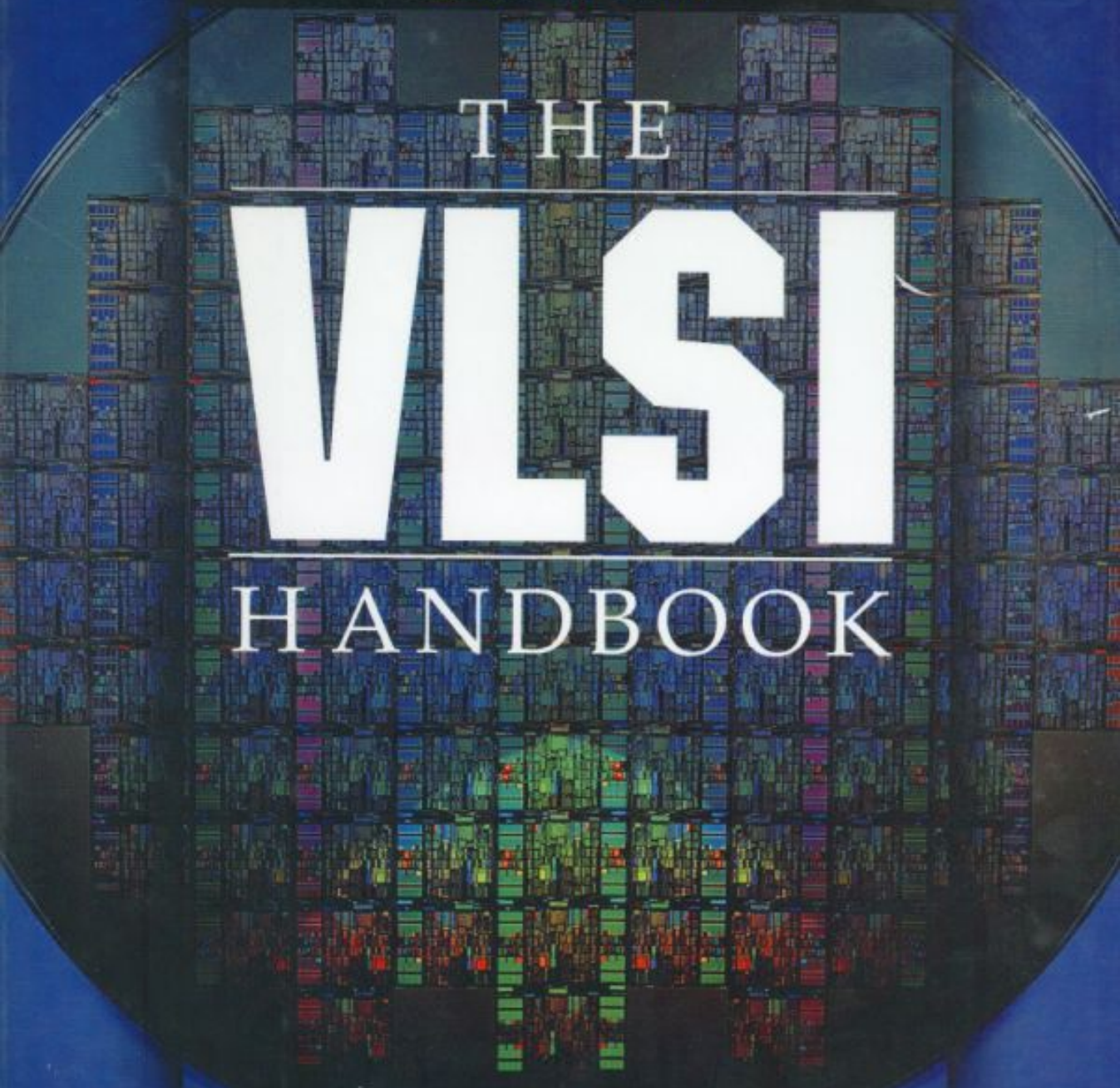
WAI-KAI CHEN
Editor-in-Chief

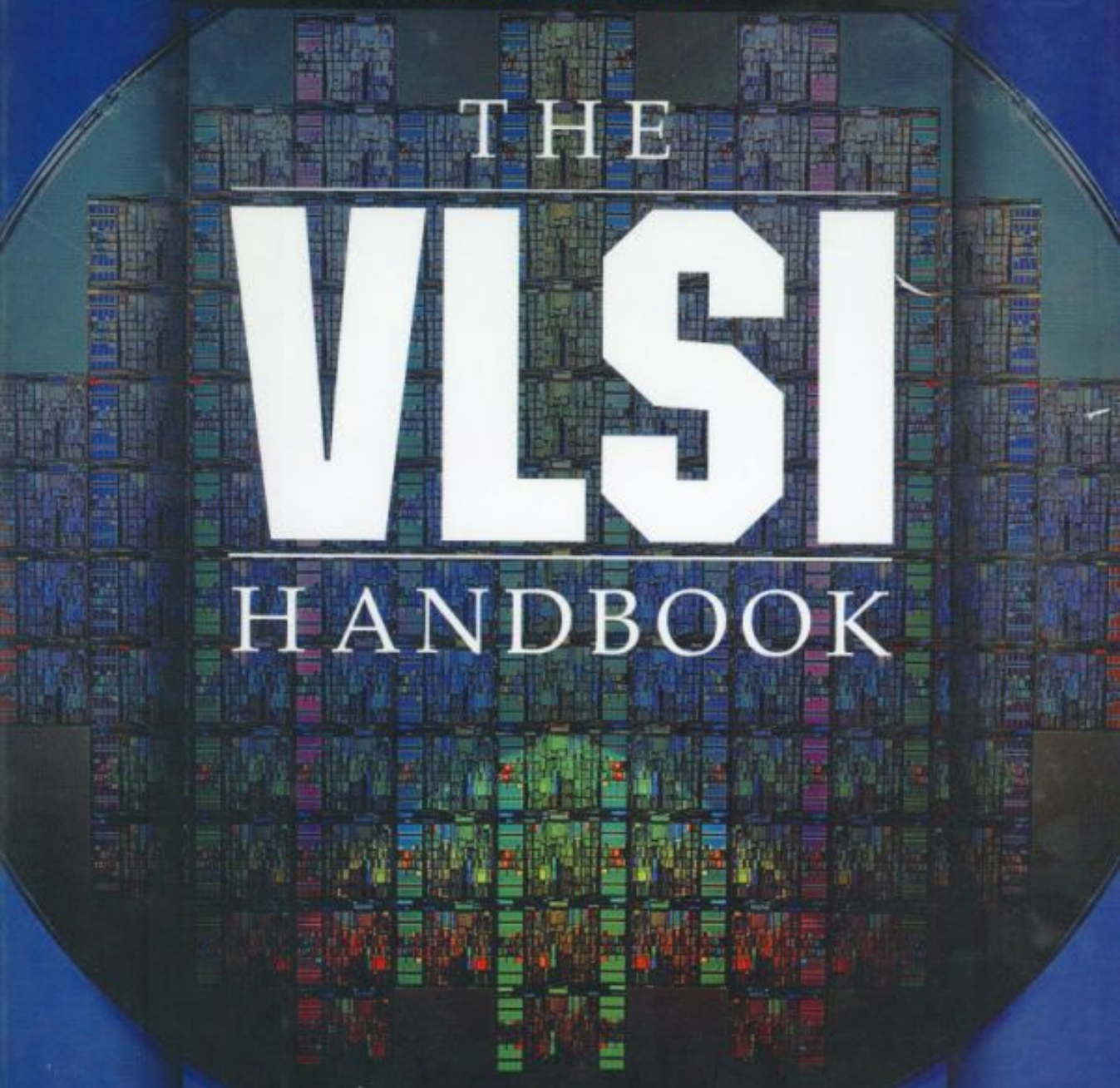
THE VLSI HANDBOOK



CRC PRESS
IEEE PRESS
A CRC Handbook Published in Cooperation with IEEE Press

WAI-KAI CHEN
Editor-in-Chief

THE VLSI HANDBOOK



CRC PRESS
IEEE PRESS
A CRC Handbook Published in Cooperation with IEEE Press

Library of Congress Cataloging-in-Publication Data

The VLSI handbook / edited by Wai-Kai Chen.
p. cm.
Includes bibliographical references and index.
ISBN 0-8493-8593-8 (alk. paper)
1. Integrated circuits--Very large scale integration. I. Chen,Wai-Kai, 1936–

TK7874.75.V573 1999
621.39'5—dc21 99-047682
CIP

No claim to original U.S. Government works
International Standard Book Number 0-8493-8593-8
Library of Congress Card Number 99-047682
Printed in the United States of America 1 2 3 4 5 6 7 8 9 0
Printed on acid-free paper

Preface

Purpose

The purpose of *The VLSI Handbook* is to provide in a single volume a comprehensive reference work covering the broad spectrum of VLSI technology. It is written and developed for the practicing electrical and computer engineers in industry, government, and academia. The goal is to provide the most up-to-date information in IC technology, devices and their models, circuit simulations, amplifiers, logic design, memory, registers and system timing, microprocessor and ASIC, test and testability, design automation, and design languages. The handbook is not an all-encompassing digest of everything taught within an electrical and computer engineering curriculum on VLSI technology. Rather, it is the engineer's first choice in looking for a solution. Therefore, full references to other sources of contributions are provided. The ideal reader is a B.S.-level engineer with a need for a one-source reference to keep abreast of new techniques and procedures as well as review standard practices.

Background

The handbook stresses fundamental theory behind professional applications. In order to do so, it is reinforced with frequent examples. Extensive development of theory and details of proofs have been omitted. The reader is assumed to have a certain degree of sophistication and experience. However, brief reviews of theories, principles and mathematics of some subject areas are given. These reviews have been done concisely with perception. The handbook is not a textbook replacement, but rather a reinforcement and reminder of material learned as a student. Therefore, important advancement and traditional as well as innovative practices are included.

Since the majority of professional electrical engineers graduated before powerful personal computers were widely available, many computational and design methods may be new to them. Therefore, computers and software use are thoroughly covered. Not only does the handbook use traditional references to cite sources for the contributions, it also contains all *relevant* sources of information and tools that would assist the engineer in performing his/her job. This may include sources of software, databases, standards, seminars, conferences, etc.

Organization

Over the years, the fundamentals of VLSI technology have evolved to include a wide range of topics and a broad range of practice. To encompass such a wide range of knowledge, the handbook focuses on the key concepts, models, and equations that enable the electrical or computer engineer to analyze, design and predict the behavior of very large-scale integrated circuits. While design formulas and tables are listed, emphasis is placed on the key concepts and theories underlying the applications.

The information is organized into 13 major sections, which encompass the field of VLSI technology. Each section is divided into chapters, each of which was written by a leading expert in the field to enlighten and refresh knowledge of the mature engineer, and to educate the novice. Each chapter contains introductory material, leading to the appropriate applications, and references. The *references* provide a list of useful books and articles for following reading.

Locating Your Topic

Numerous avenues of access to information contained in the handbook are provided. A complete table of contents is presented at the front of the book. In addition, an individual table of contents precedes each of the thirteen sections. Finally, each chapter begins with its own table of contents. The reader is urged to look over these tables of contents to become familiar with the structure, organization, and content of the book. For example, see Section VIII: Microprocessor and ASIC, then Chapter 61: Microprocessor Design Verification, and then Section 61.8: Emulation. This tree-like structure enables the reader to move up the tree to locate information on the topic of interest.

A combined subject and author index has been compiled to provide means of accessing information. It can also be used to locate definitions; the page on which the definition appears for each key defining term is given in this index.

The VLSI Handbook is designed to provide answers to most inquiries and direct inquirers to further sources and references. We trust that it will meet your needs.

Acknowledgments

The compilation of this book would not have been possible without the dedication and efforts of the Editorial Board of Advisors, the section editors, the publishers, and most of all the contributing authors. I wish to thank them all and also my wife, Shiao-Ling, for her patience and understanding.

Wai-Kai Chen
Editor-in-Chief

Editor-in-Chief

Wai-Kai Chen, Professor and Head of the Department of Electrical Engineering and Computer Science at the University of Illinois at Chicago, teaches graduate and undergraduate courses in electrical engineering in the fields of circuits and systems. He received his B.S. and M.S. in electrical engineering at Ohio University where he was later recognized as a Distinguished Professor. He earned his Ph.D. in electrical engineering at the University of Illinois at Urbana-Champaign.

Professor Chen has extensive experience in education and industry and is very active professionally in the fields of circuits and systems. He has served as a visiting professor at Purdue University and the University of Hawaii at Manoa. He was Editor of the *IEEE Transactions on Circuits and Systems,* both *Series I and II* and President of the IEEE Circuits and Systems Society. Currently, he is Editor-in-Chief of the *Journal of Circuits, Systems and Computers* and Editor of the *Advanced Series in Electrical and Computer Engineering*, Imperial College Press. He received the *Lester R. Ford Award* from the Mathematical Association of America, the *Alexander von Humboldt Award* from Germany, the *Ohio University Alumni Medal of Merit for Distinguished Achievement in Engineering Education*, the *Senior University Scholar Award* from University of Illinois at Chicago, the *Distinguished Alumnus Award* from the University Illinois at Urbana-Champaign, and the *Society Meritorious Service Award* and the *Education Award* from IEEE Circuits and Systems Society. He also received more than a dozen honorary professor awards from major institutions in China.

A Fellow of the Institute of Electrical and Electronics Engineers and the American Association for the Advancement of Science, Professor Chen is widely known in the profession for his *Applied Graph Theory* (North-Holland*), Theory and Design of Broadband Matching Networks* (Pergamon Press*), Active Network and Feedback Amplifier Theory* (McGraw-Hill), *Linear Networks and Systems* (Brooks/Cole*), Passive and Active Filters: Theory and Implementations* (John Wiley), *Theory of Nets* (John Wiley), and *The Circuits and Filters Handbook* (Editor-in-Chief, CRC Press).

Advisory Board

Contributors

Ramachandra Achar
Carleton University
Ottawa, Ontario, Canada

James H. Aylor
University of Virginia
Charlottesville, Virginia

R. Jakob Baker
University of Idaho
Boise, Idaho

David L. Barton
Intermetrics, Inc.
Vienna, Virginia

Andrea Baschirotto
Università di Pavia
Pavia, Italy

Charles R. Baugh
C. R. Baugh and Associates
Bellevue, Washington

J. Bhasker
Cadence Design Systems
Allentown, Pennsylvania

David Blaauw
Motorola, Inc.
Austin, Texas

Marc Borremans
Katholieke Universiteit Leuven
Leuven-Heverlee, Belgium

Victor Boyadzhyan
Jet Propulsion Laboratory
Pasadena, California

Charles E. Chang
Conexant Systems, Inc.
Newbury Park, California

Wai-Kai Chen
University of Illinois
Chicago, Illinois

Kuo-Hsing Cheng
Tamkang University
Tamsui, Taipei Hsien, Taiwan

John Choma, Jr.
University of Southern California
Los Angeles, California

Amy Hsiu-Fen Chou
National Tsing-Hua University
Hsin-Chu, Taiwan

Moon Jung Chung
Michigan State University
East Lansing, Michigan

David J. Comer
Brigham Young University
Provo, Utah

Donald T. Comer
Brigham Young University
Provo, Utah

Daniel A. Connors
University of Illinois
Urbana, Illinois

Donald Cottrell
Silicon Integration Initiative, Inc. (Si2, Inc.)
Austin, Texas

John D. Cressler
Auburn University
Auburn, Alabama

Sorin Cristoloveanu
Institut National Polytechnique de Grenoble
Grenoble, France

Bram De Muer
Katholieke Universiteit Leaven
Leuven-Heverlee, Belgium

Geert A. De Veirman
Silicon Systems, Inc.
Tustin, California

Maria del Mar Hershenson
Stanford University
Stanford, California

Allen M. Dewey
Duke University
Durham, North Carolina

Abhijit Dharchoudhury
Motorola, Inc.
Austin, Texas

Donald B. Estreich
Hewlett-Parkard Company
Santa Rosa, California

John W. Fattaruso
Texas Instruments, Incorporated
Dallas, Texas

Eby G. Friedman
University of Rochester
Rochester, New York

Thad Gabara
Lucent Technologies
Murray Hill, New Jersey

Stantanu Ganguly
Intel Corp.
Austin, Texas

Yosef Gavriel
Virginia Polytechnic Institute and State University
Blacksburg, Virginia

Jan V. Grahn
Royal Institute of Technology
Kista-Stockholm, Sweden

Rajesh K. Gupta
University of California
Irvine, California

Sumit Gupta
University of California
Irvine, California

Peter J. Hesketh
The Georgia Institute of Technology
Atlanta, Georgia

Karl Hess
University of Illinois
Urbana, Illinois

Charles Ching-Hsiang Hsu
National Tsing-Hua University
Hsinchu, Taiwan

Jen-Sheng Hwang
National Science Council
Hsinchu, Taiwan

Wen-mei Hwu
University of Illinois
Urbana, Illinois

Kazumi Inoh
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Hidemi Ishiuchi
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Mohammed Ismail
The Ohio State University
Columbus, Ohio

Hiroshi Iwai
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Vikram Iyengar
University of Illinois
Urbana, Illinois

Johan Janssens
Katholieke Universiteit Leuven
Leuven-Heverlee, Belgium

Dimitri Kagaris
Southern Illinois University
Carbondale, Illinois

Steve M. Kang
University of Illinois
Urbana, Illinois

Nick Kanopoulos
Atmel, Multimedia and Communications
Morrisville, North Carolina

Tanay Karnik
Intel Corporation
Hillsboro, Oregon

Yasuhiro Katsumata
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Pankaj Khandelwal
University of Illinois
Chicago, Illinois

John M. Khoury
Lucent Technologies
Murray Hill, New Jersey

Heechul Kim
Hankuk University of Foreign Studies
Yongin, Kyung Ki-Do, Korea

Hideki Kimijima
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Isik C. Kizilyalli
Lucent Bell Laboratories
Orlando, Florida

Robert H. Klenke
Virginia Commonwealth University
Richmond, Virginia

Ivan S. Kourtev
University of Pittsburgh
Pittsburgh, Pennsylvania

Thomas H. Lee
Stanford University
Stanford, California

Harry W. Li
University of Idaho
Moscow, Idaho

Chi-Hung Lin
The Ohio State University
Columbus, Ohio

Frank Ruei-Ling Lin
National Tsing-Hua University
Hsin-Chu, Taiwan

John W. Lockwood
Washington University
St. Louis, Missouri

Stephen I. Long
University of California
Santa Barbara, California

Flavio Lorenzelli
ST Microelectronics, Inc.
San Diego, California

Ashraf Lotfi
Lucent Technologies
Murray Hill, New Jersey

Joseph W. Lyding
University of Illinois
Urbana, Illinois

Martin Margala
University of Alberta
Edmonton, Alberta, Canada

Samuel S. Martin
Lucent Technologies
Murray Hill, New Jersey

Erik A. McShane
University of Illinois
Chicago, Illinois

Shin-ichi Minato
NTT Network Innovation Laboratories
Yokosuka-shi, Japan

Sunderarajan S. Mohan
Stanford University
Stanford, California

Hisayo S. Momose
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Eiji Morifuji
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Toyota Morimoto
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Saburo Muroga
University of Illinois
Urbana, Illinois

Akio Nakagawa
Toshiba Corporation
Saiwai-ku, Kawasaki, Japan

Yuichi Nakamura
NEC Corporation
Miyamae-ku, Kawasaki, Japan

Michel S. Nakhla
Carleton University
Ottawa, Ontario, Canada

Zainalabedin Navabi
Northeastern University
Boston, Massachusetts

Philip G. Neudeck
NASA Glenn Research Center
Cleveland, Ohio

Kwok Ng
Lucent Technologies
Murray Hill, New Jersey

Hideaki Nii
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Tatsuya Ohguro
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Mikael Östling
Royal Institute of Technology
Kista-Stockholm, Sweden

Alice C. Parker
University of Southern California
Los Angeles, California

Alison Payne
Imperial College
University of London
London, England

Massoud Pedram
University of Southern California
Los Angeles, California

J. Gregory Rollins
Antrim Design Systems
Scotts Valley, California

Elizabeth M. Rudnick
University of Illinois
Urbana, Illinois

Kirad Samavati
Stanford University
Stanford, California

Rolf Schaumann
Portland State University
Portland, Oregon

Rick Shih-Jye Shen
National Tsing-Hua University
Hsinchu, Taiwan

Krishna Shenai
University of Illinois
Chicago, Illinois

Bing J. Sheu
Avant! Corporation
Los Angeles, California

Bang-Sup Song
University of California
La Jolla, California

Michiel Steyaert
Katholieke Universiteit Leuven
Leuven-Heverlee, Belgium

Haruyuki Tago
Toshiba Semiconductor Company
Saiwai-ku, Kawasaki-shi, Japan

Naofumi Takagi
Nagoya University
Nagoya, Japan

Donald C. Thelen
Analog Interfaces
Bozeman, Montana

Chris Toumazou
Imperial College
University of London
London, England

Spyros Tragoudas
Southern Illinois University
Carbondale, Illinois

Yuh-Kuang Tseng
Industrial Research and Technology Institute
Chutung, Hsinchu, Taiwan

Meera Venkataraman
Troika Networks, Inc.
Calabasas Hills, California

Suhrid A. Wadekar
IBM Corp.
Hopewell Junction, New York

Chorng-kuang Wang
National Taiwan University
Taipei, Taiwan

R. F. Wassenaar
University of Twente
Enschede, The Netherlands

Louis A. Williams III
Texas Instruments, Incorporated
Dallas, Texas

Wayne Wolf
Princeton University
Princeton, New Jersey

Chung-Yu Wu
National Chiao Tung University
Hsinchu, Taiwan

Evans Ching-Song Yang
National Tsing-Hua University
Hsinchu, Taiwan

Kazuo Yano
Hitachi Ltd.
Kokubunji, Tokyo, Japan

Kung Yao
University of California
Los Angeles, California

Ko Yoshikawa
NEC Corporation
Fuchu, Tokyo, Japan

Kuniyoshi Yoshikawa
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Takashi Yoshitomi
Toshiba Corporation
Isogo-ku, Yokohama, Japan

Min-shueh Yuan
National Taiwan University
Taipei, Taiwan

C. Patrick Yue
Stanford University
Stanford, California

Contents

SECTION II Devices and Their Models

SECTION III Circuit Simulations

SECTION V Logic Design

SECTION VII Analog Circuits

SECTION VIII Microprocessor and ASIC

SECTION X Compound Semiconductor Digital Integrated Circuit Technology

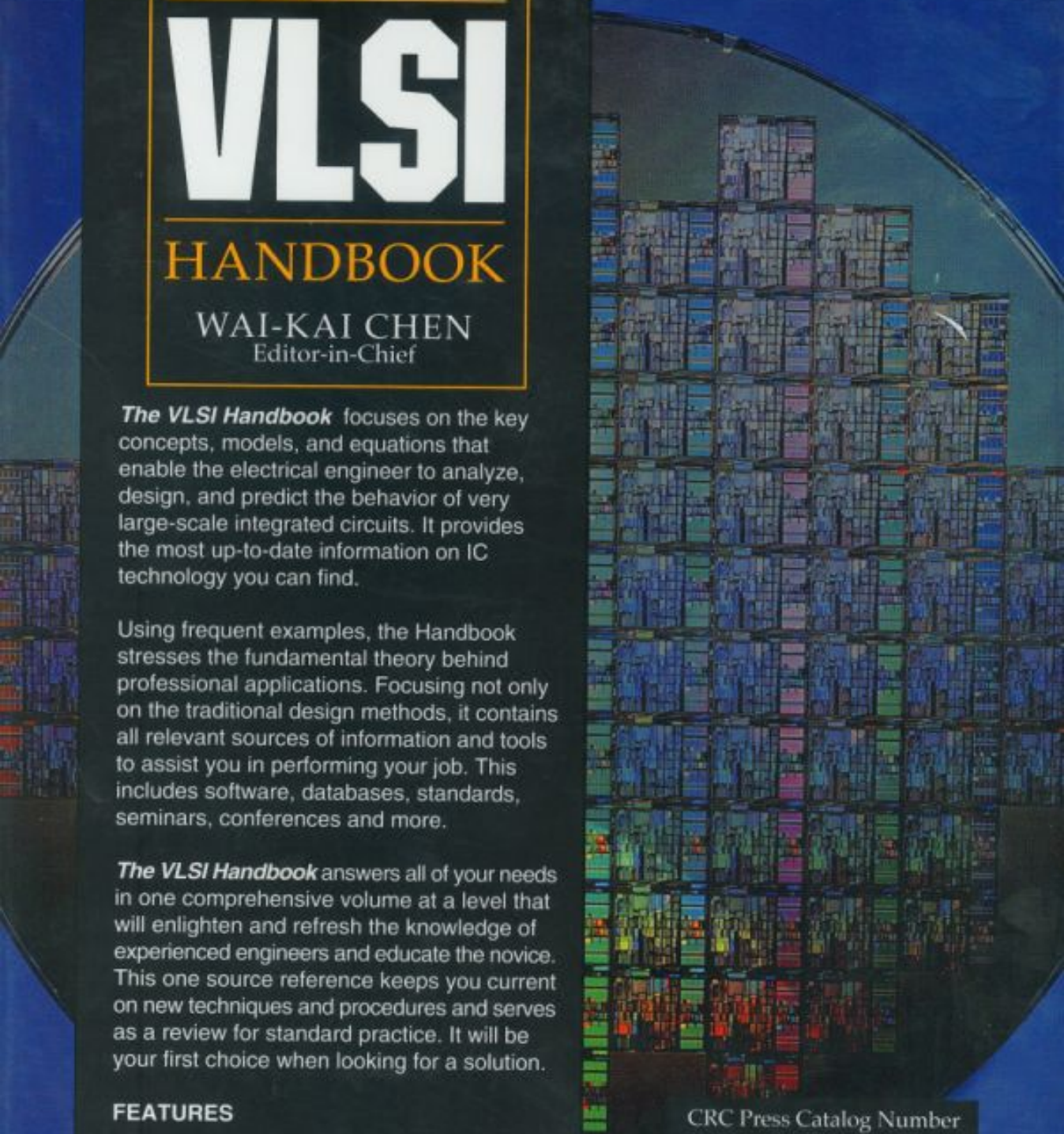
SECTION XI Design Automation

SECTION XIII Design Languages

Electrical/Computer Engineering

THE VLSI HANDBOOK

WAI-KAI CHEN
Editor-in-Chief



The VLSI Handbook focuses on the key concepts, models, and equations that enable the electrical engineer to analyze, design, and predict the behavior of very large-scale integrated circuits. It provides the most up-to-date information on IC technology you can find.

Using frequent examples, the Handbook stresses the fundamental theory behind professional applications. Focusing not only on the traditional design methods, it contains all relevant sources of information and tools to assist you in performing your job. This includes software, databases, standards, seminars, conferences and more.

The VLSI Handbook answers all of your needs in one comprehensive volume at a level that will enlighten and refresh the knowledge of experienced engineers and educate the novice. This one source reference keeps you current on new techniques and procedures and serves as a review for standard practice. It will be your first choice when looking for a solution.

FEATURES

- Covers the latest technologies in VLSI circuit design, fabrication, and testing
- Provides articles on research and applied technology
- Focuses on key issues facing the integrated circuit industry including the scaling of circuits, fabrication process, materials, and CAD simulation tools for design and testing
- Discusses automating the design process

CRC Press Catalog Number 8593

IEEE Order Number: PC5856

ISBN 0-8493-8593-8
90000
9 780849 385933

Shenai, K., et al. “VLSI Technology: A System Perspective”
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

1

VLSI Technology: A System Perspective

Krishna Shenai
Erik A. McShane
University of Illinois at Chicago

1.1 Introduction

The development of VLSI systems has historically progressed hand-in-hand with technology innovations. Often, fresh achievements in lithography, or semiconductor devices, or metallization have led to the introduction of new products. Conversely, market demand for particular products or specifications has greatly influenced focused research into the technology capabilities necessary to deliver the product. Many conventional VLSI systems as a result have engendered highly specialized technologies for their support.

In contrast, a characteristic of emerging VLSI products is the integration of diverse systems, each of which previously required a unique technology, into a single technology platform. The driving force behind this trend is the demand in consumer and noncommercial sectors for compact, portable, wireless electronics products — the nascent "system-on-a-chip" era.[1–4] Figure 1.1 illustrates some of the system components playing a role in this development.

Most of the achievements in dense systems integration have derived from scaling in silicon VLSI processes.[5] As manufacturing has improved, it has become more cost-effective in many applications to replace a chip set with a monolithic IC: packaging costs are decreased, interconnect paths shrink, and power loss in I/O drivers is reduced. Further scaling to deep submicron dimensions will continue to widen the applications of VLSI system integration, but also will lead to additional complexities in reliability, interconnect, and lithography.[6] This evolution is raising questions over the optimal level of integration: package level or chip level. Each has distinct advantages and some critical deficiencies for cost, reliability, and performance.

Board-level interconnection of chip sets, although a mainstay of low-cost, high-volume manufacturing, cannot provide a suitably dense integration of high-performance, core VLSI systems. Package- and chip-level integration are more practical contenders for VLSI systems implementation because of their compact dimensions and short signal interconnects. They also offer a tradeoff between dense monolithic integration and application-specific technology optimization. It is unclear at this time of the pace in the further

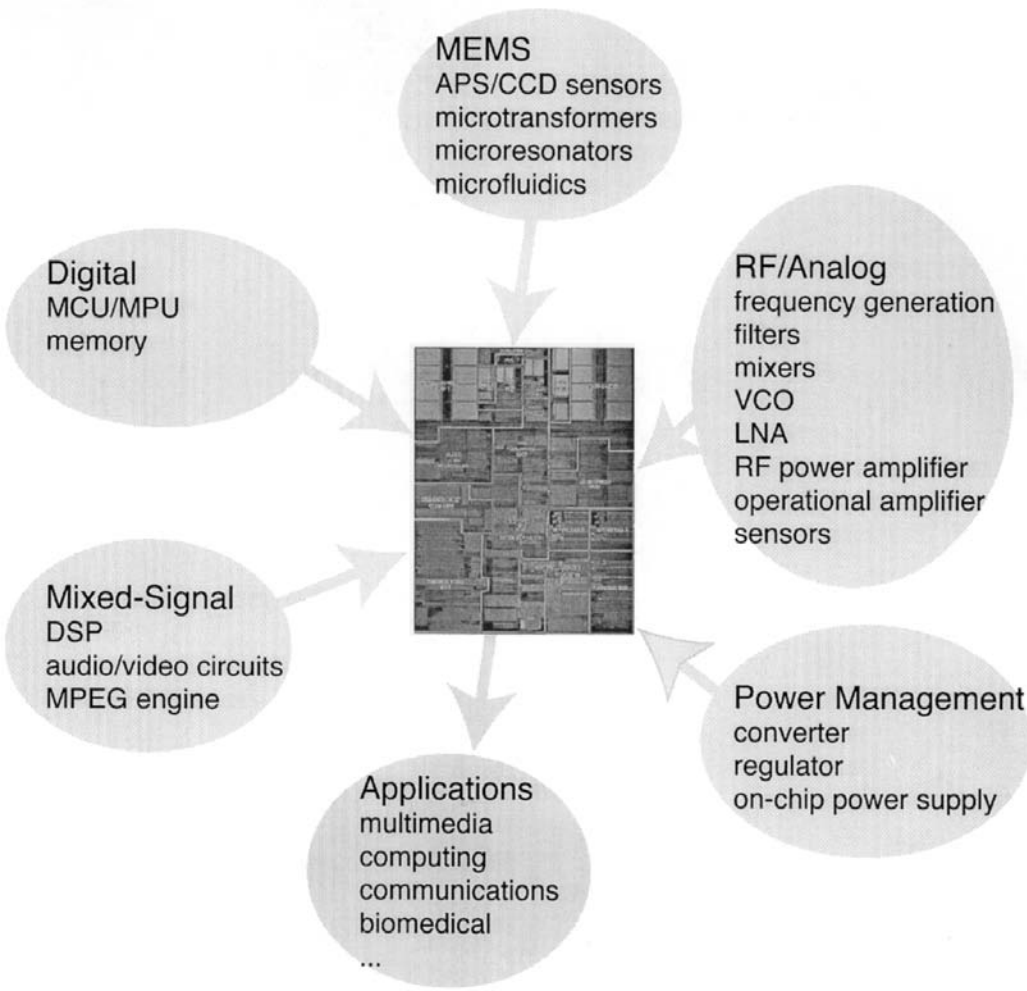


FIGURE 1.1 These system components are representative of the essential building blocks in VLSI "systems-on-a-chip."

evolution of VLSI systems, although systems integration will continue to influence and be influenced by technology development.

The remainder of this chapter will trace the inter-relationship of technology and systems to date and then outline emerging and future VLSI systems and their technology requisites. Alternative technologies will also be introduced with a presentation of their potential impact on VLSI systems. Focused discussion of the specific VLSI technologies introduced will follow in later chapters.

Given the level of systems integration afforded by available technology and the diverse signal-processing capabilities and applications supported, in this chapter a "VLSI system" is loosely defined as any complex system, primarily electronic in nature, based on semiconductor manufacturing with an extremely dense integration of minimal processing elements (e.g., transistors) and packaged as a single- or multi-chip module.

1.2 Contemporary VLSI Systems

VLSI systems can be crudely categorized by the nature of the signal processing they perform: analog, digital, or power. Included in analog are high-frequency systems, but they can be distinguished both by

design methodology and their sensitivity to frequency-dependent characteristics in biasing and operation. Digital systems consist of logic circuits and memory, although it should be noted that most "digital" systems now also contain significant analog subsystems for data conversion and signal integrity. Power semiconductor devices have previously afforded only very low levels of integration considering their extreme current- and voltage-handling requirements (up to 1000 A and 10 kV) and resulting high temperatures. However, with the advent of hybrid technologies (integrating different materials on a single silicon substrate), partial insulating substrates (with dielectrically isolated regions for power semiconductor devices), and MCM packaging, integrated "smart" power electronics are appearing for medium power (up to 1 kW) applications. A relative newcomer to the VLSI arena is microelectromechanical systems (MEMS). As the name states, MEMS is not purely electronic in nature and is now frequently extended to also label systems that are based on optoelectronics, biochemistry, and electromagnetics.

Digital Systems

Introduction

The digital systems category comprises microprocessors, microcontrollers, specialized digital signal processors, and solid-state memory. As mentioned previously, these systems may also contain analog, power, RF, and MEMS subsystems; but in this section, discussion is restricted to digital electronics.

Beginning with the introduction in 1971 of the first true microprocessor — the Intel 4004 — digital logic ICs have offered increasing functionality afforded by a number of technology factors. Transistor miniaturization from the 10-micron dimensions common 30 years ago to state-of-the-art 0.25-micron lithography has boosted IC device counts to over 10 million transistors. To support subsystem interconnection, multilevel metallization stacks have evolved. And, to reduce static and switching power losses, low-power/low-voltage technologies have become standard. The following discussion of VLSI technology pertains to the key metrics in digital systems: power dissipation, signal delay, signal integrity, and memory integration.

Power Dissipation

The premier technology today for digital systems is CMOS, owing to its inherent low-power attributes and excellent scaling to deep submicron dimensions. Total power dissipation is expressed as

$$\begin{aligned} P &= P_{dynamic} + P_{static} \\ &= (P_{switching} + P_{short\text{-}circuit} + P_{leakage}) \\ &= V_{DD}^2 f \sum_n a_n c_n + V_{DD} \sum_n i_{sc_n} + V_{DD} \sum_n (1 - a_n) i_{leak_n} \end{aligned} \tag{1.1}$$

where V_{DD} is the operating supply; f is the clock frequency; and per node a_n is the switching activity, c_n is the switching capacitance, i_{scn} is the short-circuit current, and i_{leakn} is the leakage current (subthreshold conduction and junction leakage). From this expression it is apparent that the most significant reduction in power dissipation can be accomplished by scaling the operating supply. However, as V_{DD} is reduced to 1 V, the contribution of leakage current to overall power dissipation increases if transistor V_T is scaled proportionally to V_{DD}. Subthreshold current in bulk CMOS, neglecting junction leakage and body effects, can be expressed as[7]

$$I_{sub} = \frac{W}{L} I_0 e^{\frac{V_{GS} - V_T}{n\phi_t}} \left(1 - e^{\frac{-V_{DS}}{\phi_t}}\right) \tag{1.2}$$

where

$$I_0 = k'(n-1)\phi_t^2 \tag{1.3}$$

$$n = 1 + \frac{\gamma}{2\sqrt{\phi_t}} \tag{1.4}$$

$$k' = \mu C'_{ox} \tag{1.5}$$

$$\gamma = \frac{\sqrt{2q\varepsilon_S N_B}}{C'_{ox}} \tag{1.6}$$

$$C'_{ox} = \frac{\varepsilon_{ox}}{t_{ox}} \tag{1.7}$$

W and L are channel width and length, respectively; ϕ_t is thermal voltage (approximately 0.259 V at 300 K); μ is carrier mobility in the channel; ε_{ox} is gate dielectric permittivity (3.45×10^{-13} F/cm for SiO_2); ε_S is semiconductor permittivity (1.04×10^{-12} F/cm for Si); N_B is bulk doping; and t_{ox} is gate dielectric thickness. This trend is exacerbated if minimal-switching circuit techniques are employed or if sleep modes place the logic into idle states for long periods. Device scaling thus must consider the architecture and performance requirements.

Figures 1.2 and 1.3 show the inverse normalized energy-delay product (EDP) contours for a hypothetical 0.25-micron device.[8] The energy required per operation is

$$E = \frac{P}{f} \tag{1.8}$$

Normalization is performed relative to the best obtained EDP for this technology. Fig. 1.2 shows data for an ideal device and Fig. 1.3 adds non-idealities by considering velocity saturation effects and uncertainty in V_{DD}, V_T, and temperature T. In the ideal device, the dashed lines indicate vectors of normalized constant performance relative to the performance obtained at the optimal EDP point. The switching frequency can be approximated by

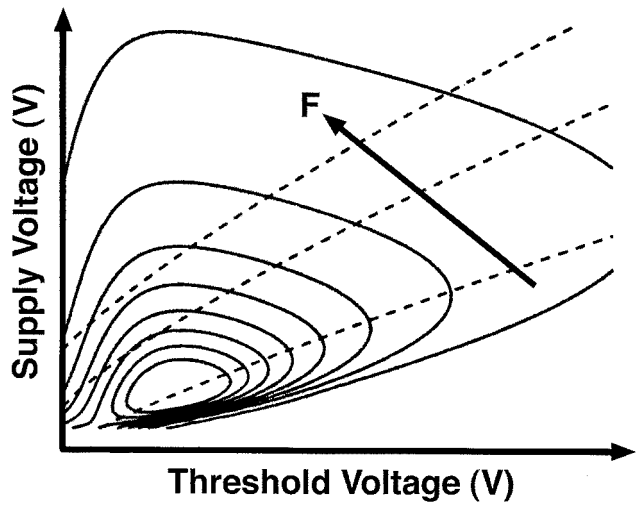


FIGURE 1.2 Inverse normalized EDP contours for an ideal device (after Ref. 8). Dashed lines indicate vectors of constant performance. Arrow *F* shows direction of increasing performance.

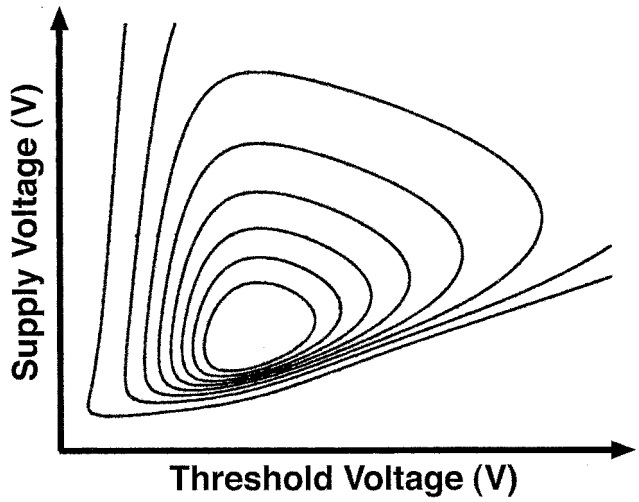


FIGURE 1.3 Inverse normalized EDP contours for a non-ideal device considering velocity saturation and uncertainty in V_{DD}, V_T, and temperature T (after Ref. 8).

$$f = \frac{1}{t_{rise}} = \frac{1}{t_{fall}} = \frac{I_{Dsat}}{CV_{DD}} \tag{1.9}$$

and the performance, F, when considered as proportional to f, can be expressed as

$$f = \alpha \frac{(V_{DD} - V_T)^2}{V_{DD}} \tag{1.10}$$

where scaling factor α is applied to normalize performance. These plots illustrate the tradeoffs in optimizing system performance for low-power requirements and highest performance. Frequency can also be scaled to reduce power dissipation, but this is not considered here as it generally also degrades performance.

Considering purely dynamic power losses (CV^2f), scaling the operating supply again yields the most significant reduction; but this scaling also affects the subthreshold leakage since V_T must be scaled similarly to maintain comparable performance levels (see Eq. 1.10). In this respect, fully depleted SOI CMOS offers improved low-voltage, low-power characteristics as it has a steeper subthreshold slope than bulk CMOS. Subthreshold slope, S, is defined as

$$S = \frac{dV_G}{d(\log I_D)} \tag{1.11}$$

This can be expressed (from Ref. 9) for bulk CMOS as

$$S = \frac{\hat{k}T}{q} \ln(10) \left(1 + \frac{C_D}{C_{ox}}\right) \tag{1.12}$$

and for fully depleted SOI CMOS (assuming negligible interface states and buried-oxide capacitance) as

$$S = \frac{\hat{k}T}{q}\ln(10) \tag{1.13}$$

where $\hat{k}$ is Boltzmann's constant (1.38×10^{-23} V·C/K), C_D is depletion capacitance, and C_{ox} is gate dielectric capacitance. Hence, for the same weak inversion gate bias, SOI CMOS can yield a leakage current several orders of magnitude less than in bulk CMOS.

Additional power dissipation occurs in the extrinsic parasitics of the active devices and the interconnect. This contribution can be minimized by salicide (self-aligned silicide) processes that deposit a low sheet resistance layer on the source, drain, and gate surfaces.

Switching Frequency and Signal Integrity

After power dissipation, the signal delay (or maximum switching frequency) of a system is the most important figure-of-merit. This characteristic, as mentioned previously, provides a first-order approximation of system performance. It also affects the short-circuit contribution to power loss since a dc path between the supply rails exists during a switching event. Also, signal delay and slope determine the deviation of a logic signal pulse from an ideal step transition.

Digital systems based on silicon bipolar and BiCMOS technologies still appear for high-speed applications, exploiting the higher small-signal gain and greater current drive of bipolar transistors over MOSFETs; but given the stringent power requirements of portable electronics, non-CMOS implementations are impractical. Emerging technologies such as silicon heterojunction bipolar and field-effect transistors (HBTs and HFETs) hold some promise of fast switching with reduced power dissipation, but the technology is too immature to be evaluated as yet.

The switching rate of a capacitively loaded node in a logic circuit can be approximated by the time required for the capacitor to be fully charged or discharged, assuming that a constant current is available for charge transport (see Eq. 1.9). Neglecting channel-length modulation effects on saturation current, the switching frequency can be written as

$$\begin{aligned} f &= \frac{I_{Dsat}}{CV_{DD}} \\ &= \frac{\frac{k'}{2}\frac{W}{L}(V_{GS} - V_T)^2}{CV_{DD}} \\ &\propto \frac{(V_{DD} - V_T)^2}{CV_{DD}} \end{aligned} \tag{1.14}$$

Voltage scaling and its effects on power dissipation have already been discussed. Considering the capacitive contribution, a linear improvement to switching speed can be obtained by scaling node capacitance. Referring to Fig. 1.4 and neglecting interconnect capacitance, the node capacitance, of a MOSFET can be expressed as

$$C_{OUT} = C_{GD} + \kappa(V_{OL}, V_{OH})(C_{db}) \tag{1.15}$$

$$C_{GD} = x_{jl}C_{ox}W + \frac{1}{2}C_{ox}WL_{eff} \tag{1.16}$$

$$C_{db} = C_{j0}WY + 2C_{jsw}(W + Y) \tag{1.17}$$

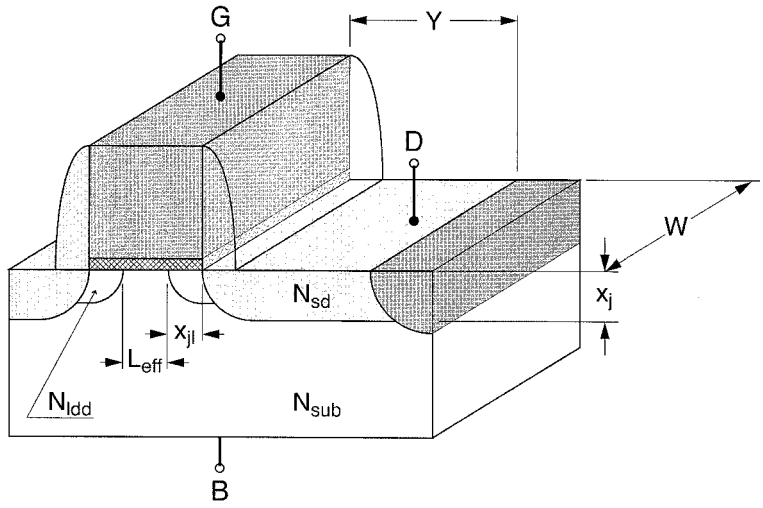


FIGURE 1.4 A MOSFET isometric cross-sectional view with critical dimensions identified.

$$C_{j0} = \sqrt{\frac{q\varepsilon_{Si}}{2\left(\frac{1}{N_{sd}} + \frac{1}{N_{sub}}\right)\phi}} \tag{1.18}$$

$$C_{jsw} \approx C_{j0} x_j \tag{1.19}$$

The drain-to-body junction capacitance C_{db} is bias dependent, and the scaling factor κ is included to determine an average value of output voltage level. Source/drain diffusion capacitance has two components: the bottom areal capacitance C_{j0} and the sidewall perimeter capacitance C_{jsw}. Although C_{jsw} is a complex function of doping profile and should account for high-concentration channel-stop implants, an approximation is made to equate C_{jsw} and C_{j0}. From Fig. 1.5, it is clear that SOI CMOS has greatly reduced device capacitances compared to bulk CMOS by the elimination junction areal and perimeter capacitances. Another technique in SOI CMOS for improving switching delay involves dynamic threshold voltage control (DTMOS) by taking advantage of the parasitic lateral bipolar transistor inherent in the device structure.[10]

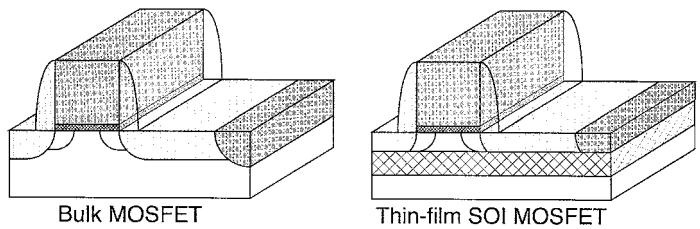


FIGURE 1.5 Cross-sectional views of a MOSFET: bulk and thin-film SOI.

To reduce interconnect resistance, copper interconnect has been introduced to replace traditional aluminum wires.[11] Table 1.1 compares two critical parameters. The higher melting point of copper also reduces long-term interconnect degradation from electromigration, in which energetic carriers dislodge

metal atoms creating voids or nonuniformities. Interconnect capacitance relative to the substrate is determined by the dielectric constant, ε_r, and the signal velocity can be defined as

$$v = \frac{c}{\sqrt{\varepsilon_r}} \tag{1.20}$$

Low-ε_r interlevel dielectrics are appearing to reduce this parasitic effect.

TABLE 1.1 Comparison of Interconnect Characteristics for Al and Cu

Material	Specific Resistance (μΩ-cm)	Melting Point (°C)
Al	2.66	660
Cu	1.68	1073

Memory Scaling

The two most critical factors determining the commercial viability of RAM products are the total power dissipation and the chip area. For implementations in battery-operated portable electronics, the goal is a 0.9-V operating supply — the minimum voltage of a NiCd cell. RAM designs are addressing these objectives architecturally and technologically. SRAMs and DRAMs share many architectural features, including memory array partitioning, reduced voltage internal logic, and dynamic threshold voltage control. DRAM, with its higher memory density, is more attractive for embedded memory applications despite its higher power dissipation.

Figure 1.6 shows a RAM block diagram that identifies the sources of power dissipation. The power equation as given by Itoh et al.[12] is

$$P = I_{DD}V_{DD} \tag{1.21}$$

$$I_{DD} = mi_{act} + m(n-1)i_{hld} + (n+m)C_{DE}V_{INT}f + C_{PT}V_{INT}f + I_{DCP} \tag{1.22}$$

where i_{act} is the effective current in active cells, i_{hld} is the holding current in inactive cells, C_{DE} is the decoder output capacitance, C_{PT} is the peripheral circuit capacitance, V_{INT} is the internal voltage level, I_{DCP} is the static current in the peripheral circuits, and n and m define the memory array dimensions.

In present DRAMs, power loss is dominated by i_{act}, the charging current of an active subarray; but as V_T is scaled along with the operating voltage, the subthreshold current begins to dominate. The trend in DRAM ICs (see Fig. 1.7) shows that the dc current will begin to dominate the total active current at about the 1-Gb range. Limiting this and other short-channel effects is necessary then to improve power efficiency.

Figure 1.8 shows trends in device parameters. A substrate doping of over 10^{18} cm^{-3} is necessary to reduce SCE, but this has the disadvantage of also increasing junction leakage currents. To achieve reduced SCE at lower substrate dopings, shallow junctions (as thin as 15 nm) are formed.[13]

Bit storage capacitors must also be scaled to match device miniaturization but still retain adequate noise tolerance. Alpha-particle irradiation becomes less significant as devices are scaled, due to the reduced depletion region; but leakage currents still place a minimum requirement on bit charge. Figure 1.9 shows that required signal charge, Q_S, has reduced only slightly with increased memory capacity, but cell areas have shrunk considerably. High-permittivity (high-ε_r) dielectrics such as Ta_2O_5 and BST ($Ba_xSr_{1-x}TiO_3$) are required to provide these greater areal capacitances at reduced dimensions.[14] Table 1.2 lists material properties for some of the common and emerging dielectrics. In addition to scaling the cell area, the capacitor aspect ratio also affects manufacturing: larger aspect ratios result in non-planar interlevel dielectric and large step height variation between memory arrays and peripheral circuitry.

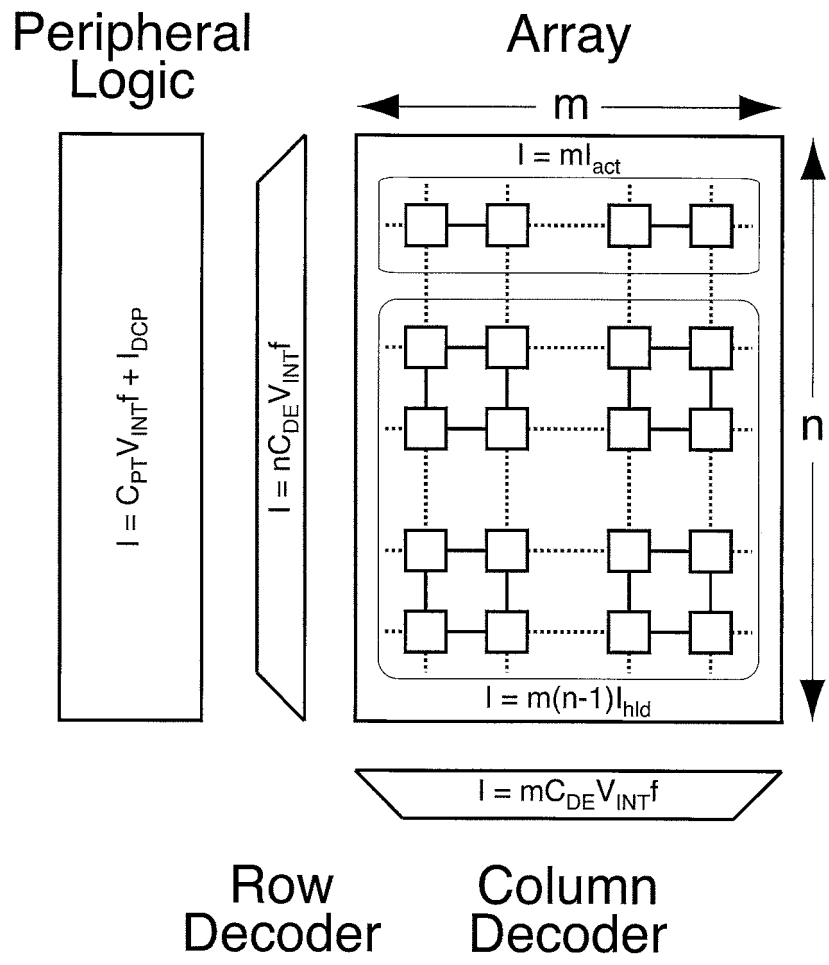


FIGURE 1.6 RAM block diagram indicating effective currents within each subsystem.

Analog Systems

Introduction

An analog system is any system that processes a signal's magnitude and phase information by a linearized response of an active device to a small-signal input. Unlike digital signals, which exhibit a large output signal swing, analog systems rely on a sufficiently small signal gain that the linear approximation holds true across the entire spectrum of expected input signal frequencies. Errors in the linear model are introduced by random process variation, intrinsic device noise, ambient noise, and non-idealities in active and passive electronics. Minimizing the cumulative effects of these "noise" contributions is the fundamental objective of analog and RF design.

Reflecting the multitude of permutations in input/output specifications and operating conditions, analog/RF design is supported by numerous VLSI technologies. Key among these are silicon MOST, BJT, and BiCMOS for low-frequency applications; silicon BJT for high-frequency, low-noise applications; and GaAs MESFET for high-frequency, high-efficiency amplifiers. Newcomers to the field include GaAs and SiGe heterojunction bipolar junction transistors (HBTs). The bandgap engineering employed in their fabrication results in devices with significantly higher f_T and f_{max} than in conventional devices, often at lower voltages.[15–17] Finally, MEMS resonators and mechanical switches offer an alternative to active device implementations.

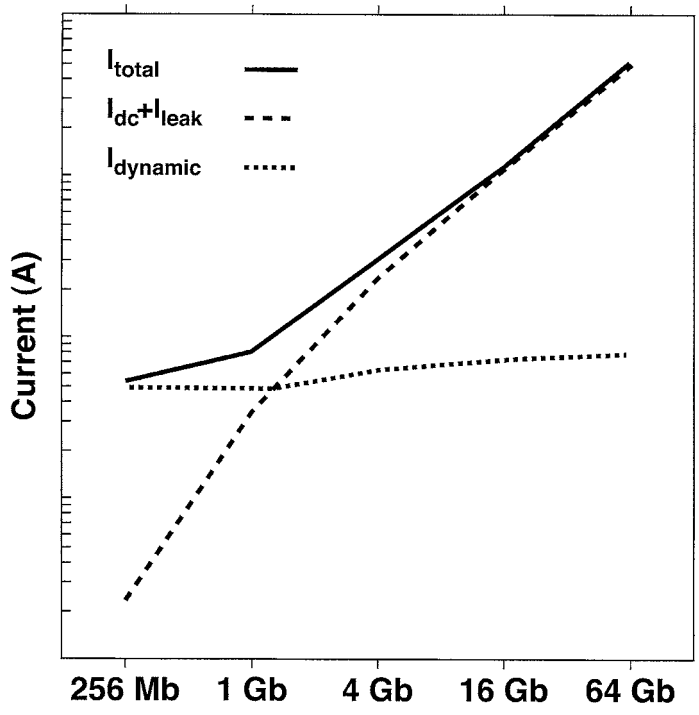


FIGURE 1.7 Contributions to total current in DRAMs (after Ref. 12).

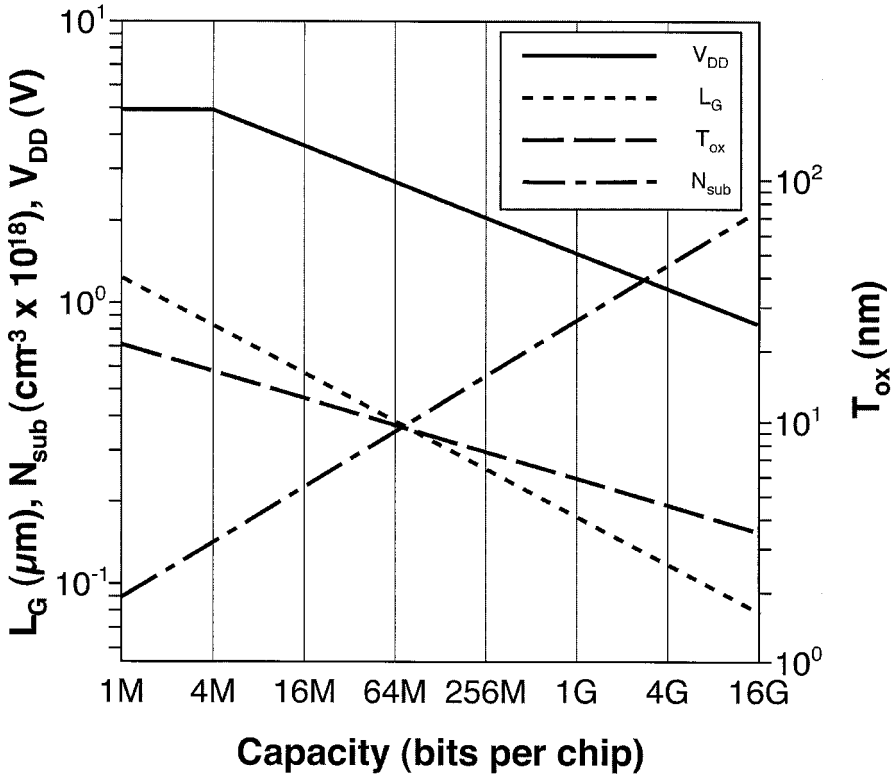


FIGURE 1.8 Trends in DRAM device technology (after Ref. 13).

TABLE 1.2 Comparison of High-Permittivity Constant Materials for DRAM Cell Capacitors

Material	Dielectric Constant	Minimum Equivalent Oxide Thickness (nm)
NO	7	3.5 to 4
Ta_2O_5	20–25	2 to 3
BST	200–400	?

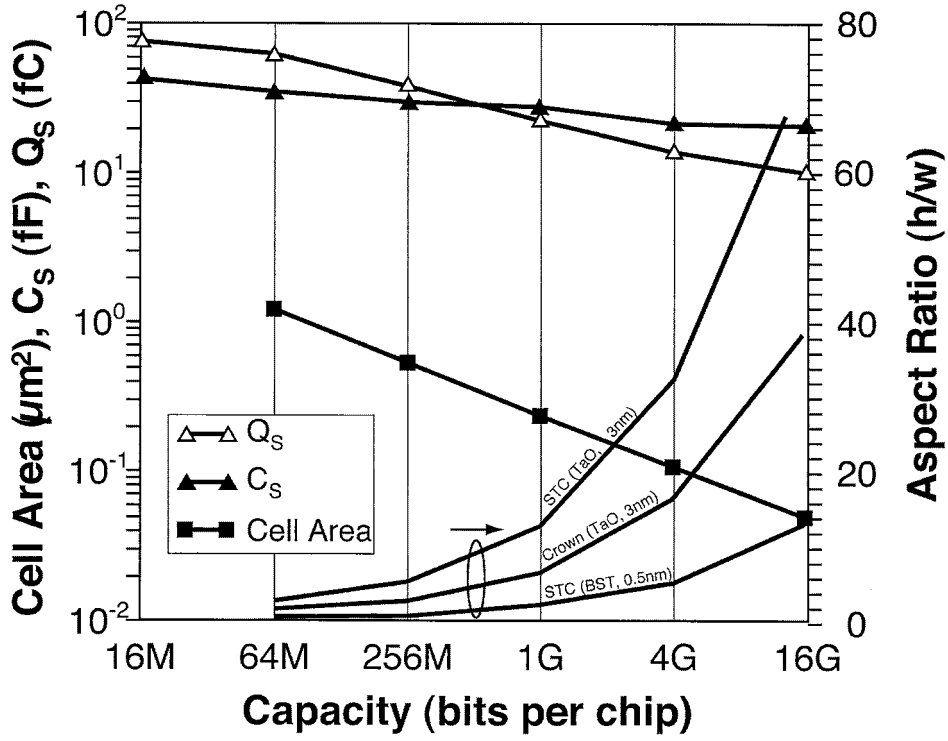


FIGURE 1.9 Trends in DRAM cell characteristics (after Ref. 13).

The most familiar application of a high-frequency system is in wireless communications, in which a translation is performed between the high-frequency modulated carrier (RF signal) used for broadcasting and the low-frequency demodulated signal (baseband) suitable for audio or machine interpretation. Wireless ICs long relied on package-level integration and scaling to deliver compact size and improved efficiency. Also, low-cost commercial IC technologies previously could not deliver the necessary frequency range and noise characteristics. This capability is now changing with several candidate technologies at hand for monolithic IC integration. CMOS has the attractive advantage of being optimal for integration of low-power baseband processing.

Amplifiers

Amplifiers boost the amplitude or power of an analog signal to suppress noise or overcome losses and enable further processing. Typical characteristics include a low noise figure (NF), large (selectable) gain (G), good linearity, and high power-added efficiency (PAE). To accommodate the variety of signal frequencies and performance requirements, several amplifier categories have evolved. These include conventional single-ended, differential, and operational amplifiers at lower frequencies and, at higher frequencies, low-noise and RF power amplifiers.

A challenge in technology scaling is providing a suitable signal-to-noise ratio and adequate biasing at a reduced operating supply. For a fixed gain, reducing the operating supply implies a similar scaling of the input signal level, ultimately approaching the noise floor of the system and leading to greater susceptibility to internal and external noise sources. Large-signal amplifiers (e.g., RF power amplifiers) that exhibit a wide output swing face similar problems with linearity at a lower operating supply.

A low-noise amplifier (LNA) is the first active circuit in a receiver. A common-source configuration of a MOSFET LNA is shown in Fig. 1.10. The input network is typically matched for lowest NF, and the output network is matched for maximum power transfer. Input impedance is matched to the source resistance, R_s, when[18]

$$\omega_0^2(L_1 + L_2)C_{gs} = 1 \tag{1.23}$$

$$\frac{g_m L_1}{C_{gs}} = R_s \tag{1.24}$$

The gain from the input matching network to the transistor gate-source voltage is equal to Q, the quality factor

$$Q = \frac{1}{g_m \omega_0 L_1} \tag{1.25}$$

where ω_0 is the RF frequency. If only the device current noise is considered, then the LNA noise figure can be expressed as

$$NF = 1 + \frac{2}{3}\frac{\omega_0 L_1}{QR_s} \tag{1.26}$$

It is observed that a larger quality factor yields a lower noise figure, but current industry practice selects an LNA Q of 2 to 3 since increasing Q also increases the sensitivity of the LNA gain to

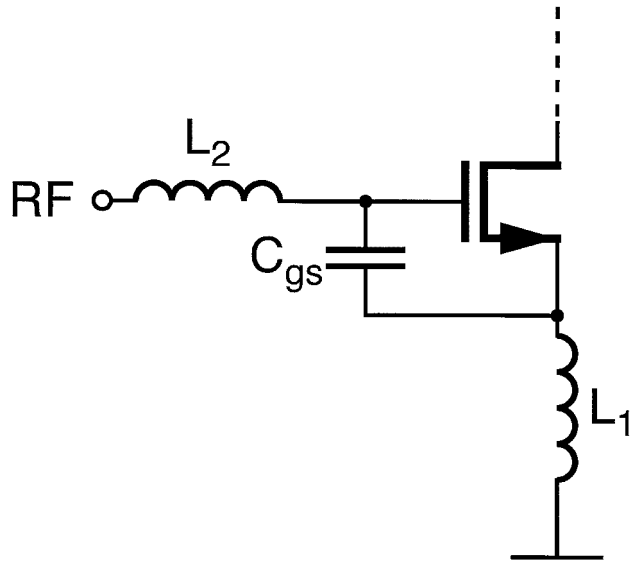


FIGURE 1.10 Common-source LNA circuit schematic.

tolerances in the passive components. By combining Eqs. 24 and 25, the device input capacitance C_{gs} can be defined

$$C_{gs} = \frac{1}{R_S Q \omega_0} \tag{1.27}$$

Assuming that the transistor is in the saturation region and that Miller feedback gain is –1, the contributions to the input capacitance are

$$C = \left(\frac{2}{3} C'_{ox} L_{eff} + C_{gso} + 2 C_{gdo}\right) W \tag{1.28}$$

where C_{gso} and C_{gdo} are, respectively, the gate-source and gate-drain overlap capacitances. The bias current (assuming a reasonable value for g_m) can then be obtained from

$$I_{Dsat} = \left(\frac{g_m^2}{2W}\right)\frac{L_{eff}}{k'} \tag{1.29}$$

As device dimensions are reduced, the required biasing current drops. Since cutoff frequency, f_T, also improves with smaller device dimensions, MOSFET performance in RF applications will continue to improve.

Power amplifiers, the last active circuit in a transmitter, have less stringent noise figure requirements than an LNA since the input signal is generated locally in the transmitter chain. Instead, linearity and PAE are more critical, particularly for variable-envelope communications protocols. RF amplifiers typically operate in class AB mode to compromise between efficiency and linearity.[19] Power-added efficiency is defined as

$$PAE = \frac{\eta}{1 - \frac{1}{G}} \tag{1.30}$$

where η is the drain (collector) efficiency (usually about 40 to 75%) and G is the amplifier power gain. This balance is highly sensitive to the precision of matching networks. Technologies such as GaAs, with its high-resistivity substrate, and SOI, with its insulating buried oxide, are best suited for integrated RF power amplifiers since they permit fabrication of low-loss, on-chip matching networks.

Interconnects and Passive Components

Passive components in analog and RF design have the essential role of providing biasing, energy storage, and signal level translation. As device technology has permitted a greater monolithic integration of active devices, a similar trend has appeared in passive components. The quality of on-chip passives, however, has lagged behind that of high-precision discrete components. Two characteristics are required of VLSI interconnects for RFICs: low-loss and integration of high-quality factor passives (capacitors and inductors). As discussed previously, resistive losses increase the overall noise figure, lead to decreased efficiency, and degrade the performance of on-chip passive components. Interconnect and device resistance are minimized by saliciding the gate and source/drain surfaces and appropriately scaling the metallization dimensions. Substrate coupling losses, which also degrade quality factors of integrated passives and can introduce substrate noise, are controlled by selecting a high-resistivity substrate such as GaAs or shielding the substrate with an insulating layer such as in SOI.

In forming capacitors on-chip, two structures are available, using either interconnect layers or the MOS gate capacitance. Metal-insulator-metal (MIM) and dual-poly capacitors both derive a capacitance from a thin interlevel dielectric (ILD) layer deposited between the conducting plates. MIM capacitors

offer a higher Q than dual-poly capacitors since, even with silicidation, resistance of poly layers is higher than in metal. Both types can suffer from imprecision caused by non-planarity in the ILD thickness caused by process non-uniformity across the wafer.

MOS gate capacitance is less subject to variation caused by dielectric non-uniformity since the gate oxide formation is tightly controlled and occurs before any back-end processing. MOS capacitors, however, are usually dismissed for high-Q applications out of concern for the highly resistive well forming the bottom plate electrode. Recent work, however, has shown that salicided MOS capacitors biased into strong inversion will achieve a Q of over 100 for applications in the range 900 MHz to 2 GHz.[20]

Inductors are essential elements of RFICs for biasing and matching, and on-chip integration translates to lower system cost and reduced effects from package parasitics. However, inductors also require a large die area and exhibit significant coupling losses with the substrate. In addition to degrading the inductor Q, substrate coupling results in the inductor becoming a source of substrate noise. The Q of an inductor can be defined[21] as

$$Q = \frac{\omega L_s}{R_s}. \text{ Substate loss factor} \cdot \text{Self-resonance factor} \tag{1.31}$$

where L_s is the nominal inductance and R_s is the series resistance. The substrate loss factor approaches unity as the substrate resistance goes to either zero or infinity. This implies that the Q factor is improved if the substrate is either short- or open-circuited. Suspended inductors achieve an open-circuited substrate by etching the bulk silicon from under the inductor structure.[22] Another approach has been to short-circuit the substrate by inserting grounding planes (ground shields).[21]

Power Systems

Introduction

Power processing systems are those devoted to the conditioning, regulation, conversion, and distribution of electrical power. Voltage and current are considered the inputs, and the system transforms the input characteristics to the form required by the load. The distinguishing feature of these systems is the specialized active device structure (e.g., rectifier, thyristor, power bipolar or MOS transistor, IGBT) required to withstand the electrothermal stresses imposed by the system. The label power integrated circuits (PICs) refers to the monolithic fabrication of a power semiconductor device along with standard VLSI electronics. At the system level, this integration has been made possible by digital control techniques and the development of mixed-signal ICs comprising analog sensing and digital logic. On the technology side, development of the power MOSFET and IGBT led to greatly simplified drive circuits and decreased complexity in the on-chip electronics.

Three types of PIC are identified: smart power, high-voltage ICs (HVICs), and discrete modules. Discrete modules are those in which individual ICs for power devices and control are packaged in a single carrier. Integration is at the package level rather than at the IC. Smart power adds a monolithic integration of analog protection circuitry to a standard power semiconductor device. The level of integration is quite low, but the power semiconductor device ratings are not disturbed by the other electronics.

HVICs are different in that they begin from a standard VLSI process and accommodate the power semiconductor device by manufacturing changes. HVICs are singled out for further discussion as they are the most suitable for VLSI integration. Although the power semiconductor device ratings cannot achieve the levels of a discrete device, HVICs are available for ratings with currents of 50 to 100 A and voltages up to 1000 V.

Two critical technical issues faced in developing HVICs are the electrical isolation of high-power and low-voltage electronics, and the development of high-Q passive components (e.g., capacitors, inductors, transformers). In the following discussion, the characteristics of power semiconductor devices will not be considered, only the issues relating to HVIC integration.

Electrical Isolation

Three types of electrical isolation are available as illustrated in Fig. 1.11.[23] In junction isolation, a p^+ implant is added to form protective diodes with the n^- epitaxial regions. The diodes are reverse-biased by applying a large negative voltage (~ –1000 V) to the substrate. Problems with this isolation include temperature-dependent diode leakage currents and the possibility of a dynamic turn-on of the diode. Additional stress to the isolation regions and interlevel dielectric is introduced when high-voltage interconnect crosses the isolation implants. The applied electric field in this situation can result in premature failure of the device.

A self-isolation technique can be chosen if all the devices are MOSFETs. When all devices are placed in individual wells (a twin-tub process), all channel regions are naturally isolated since current flow is near the oxide–semiconductor interface. The power semiconductor device and signal transistors are

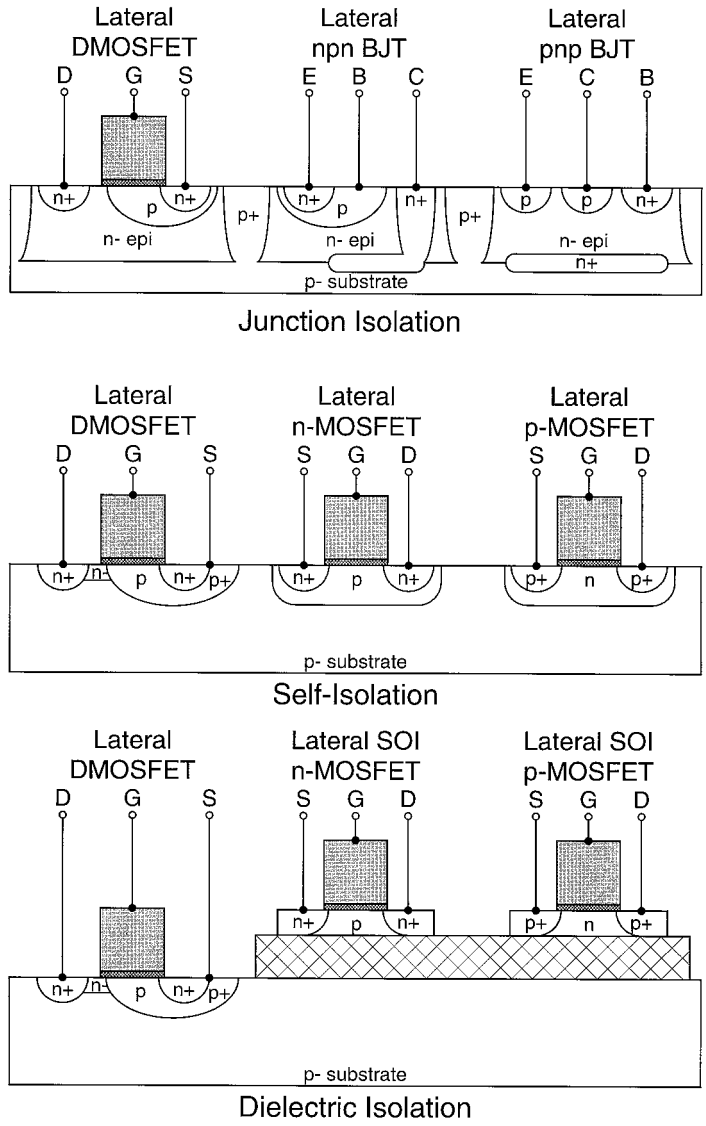


FIGURE 1.11 High-voltage IC isolation technologies: junction, self-, and dielectric-isolation.

fabricated simultaneously in junction and self-isolation, resulting in a compromise in performance.[24] In practice, bulk isolation techniques are a combination of junction and self-isolation since many HVICs exhibit dynamic surges in substrate carriers, corresponding to power device switching, that may result in latchup of low-voltage devices.[25]

Dielectric isolation decouples the fabrication of power and signal devices by reserving the bulk semiconductor for high-voltage transistors and introducing an epitaxial semiconductor layer for low-voltage devices on a dielectric surface. Formation of the buried oxide can be accomplished either by partially etching an SOI wafer to yield an intermittent SOI substrate or by selectively growing oxide on regions intended for low-voltage devices, followed by epitaxial deposition of a silicon film. In addition to providing improved isolation of power semiconductor devices, the buried oxide enhances the performance of signal transistors by reducing parasitic capacitances and chip area.[26]

Interconnects and Passive Components

A key objective in applying VLSI technology to power electronics is the reduction in system mass and volume. Current device technologies adequately provide the monolithic cofabrication of low-voltage and high-voltage electronics, and capacitors and inductors of sufficiently high Q are available to integrate substantial peripheral circuitry. As switching frequencies are increased in switching converters, passive component values are reduced, further improving the integration of power electronics. These integration capabilities have resulted from similar requirements in digital and analog/RF electronics. Conventional VLSI technologies, however, have yet to reproduce the same integration of magnetic materials needed for transformers.

Magnetics integration is hindered by the incompatibility of magnetic materials with standard VLSI processes and concerns over contamination of devices. A transformer cross-section is shown in Fig. 1.12 where the primary and secondary are wound as lateral coils with connections between the upper and lower conductors provided by vias.[27]

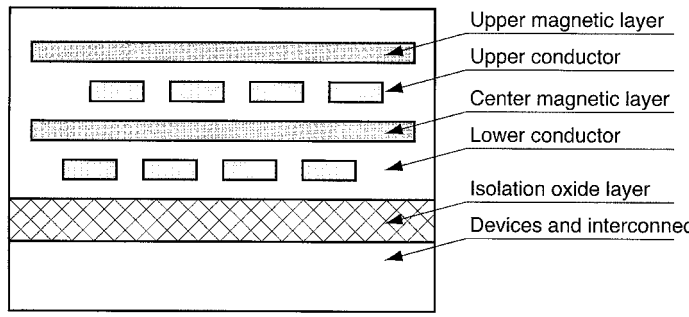


FIGURE 1.12 Cross-sectional view of a VLSI process showing integration of magnetic layers for coil transformers.

1.3 Emerging VLSI Systems

Building on the successes in integrating comprehensive systems from the individual signal processing domains, many next-generation hybrid systems are appearing that integrate systems in a cross-platform manner. The feasibility of these systems depends on the ability of technology to either combine monolithically the unique elements of each system or to introduce a single technology standard that can support broad systems.

Embedded Memory

Computer processors today are bandwidth limited with the memory interface between high-capacity external storage and the processing units unable to meet access rates. Multimedia applications such as

3-D graphics rendering and broadcast rate video will demand bandwidths from 1 to 10 GB/s. A conventional 4-Mb DRAM with 4096 sense amplifiers, a 150-ns cycle time, and a configuration of 1-M × 4-b achieves an internal bandwidth of 3.4 GB/s; however, as this data can be accessed at the I/O pins only in 4-bit segments, external bandwidth is reduced to 0.1% of available bandwidth.

Although complex cache (SRAM) hierarchies have been devised to mask this latency, each additional cache level introduces additional complexity and has an asymptotic performance limit. Since the bandwidth bottleneck is introduced by off-chip (multiplexed) routing of signals, embedded memories are appearing to provide full-bandwidth memory accesses by eliminating I/O multiplexing.

Integration of DRAM and logic is non-trivial as the technology optimization of the former favors minimal bit area, a compact capacitor structure, and low leakage, but in the latter favors device performance. Embedded DRAM therefore permits two implementations: logic fabricated in a DRAM process or DRAM fabricated as a macro in an ASIC process.

Logic in DRAM Process

DRAM technologies do not require multilevel interconnect since their regular array structure translates to very uniform routing. Introducing logic to a DRAM process has been accomplished by merging DRAM front-end fabrication (to capture optimized capacitor and well structures) with an ASIC back-end process (to introduce triple- or quad-level metallization).

Example systems include a 576-Kb DRAM and 8-way hypercube processor[28] and a 1-Mb DRAM and 4-element pixel processor.[29]

DRAM Macro in ASIC Process

DRAM macros currently developed for instantiation in an ASIC process implement a substrate-plate-trench-capacitor (SPT) by modifying deep trench isolation (DTI). Access transistor isolation is accomplished with a triple-well process to extract stray carriers injected as substrate noise.

Fabrication of the stacked-capacitor (STC) structures typical of high-capacity DRAMs cannot be performed in standard ASIC processes without significant modification; this limitation forces additional tradeoffs in the optimal balance of embedded memory capacity and fabrication cost. Example systems include an 8-Mb DRAM and 70-K gate sea-of-gates IC.[30]

Monolithic RFICs

Technologies for monolithic RFICs are proceeding in two directions: all-CMOS and silicon bipolar. All-CMOS has the attractive advantages of ready integration of baseband analog and digital signal processing, compatibility with standard analog/RF CMOS processes, and better characteristics for low-power/low-voltage operation.[31] CMOS is predicted to continue to offer suitable RF performance at operating supplies below 1 V as long as the threshold voltage is scaled accordingly. Silicon bipolar implementations, however, outperform CMOS in several key areas, particularly that of the receiver LNA.

A frequency-hopped spread-spectrum transceiver has been designed in a 1-micron CMOS process with applications to low-power microcell communications.[32,33] A microcell application was chosen since the maximum output power of 20 mW limits the transmission range.

RFICs in bipolar and BiCMOS typically focus on the specific receiver components most improved by non-CMOS implementations. A common RFIC architecture is an LNA front-end followed by a down-mixer. Interstage filtering and matching, if required, are provided off-chip. In CMOS, a 1-GHz RFIC achieved a conversion gain of 20 dB, an NF of 3.2 dB, an IP3 of 8 dBm, with a current drain of 9 mA from a 3-V supply.[34] A similar architecture in BiCMOS at 1 GHz achieved a conversion gain of 16 dB, an NF of 2.2 dB, an IP3 of –10 dBm, with a current drain of 13 mA from a 5-V supply.[35]

Single-Chip Sensors and Detectors

One of the most active areas of system development is in the field of "camera-on-a-chip" in which VLSI-compatible photodetector arrays are monolithically fabricated with digital and analog peripheral circuitry.

The older charge-coupled device (CCD) technology is being replaced by active-pixel sensing (APS) technologies in new systems since CCD has a higher per-cell capacitance.[36] APS also offers a monolithic integration of analog-to-digital converters and digital control logic. On-chip digitization of the sensor outputs eliminates the additional power loss and noise contributions incurred in buffering an analog signal for off-chip processing.

Conventional CMOS technology is expected to permit co-fabrication of APS sensors down to feature dimensions as small as 0.25 microns,[37] although leakage currents will become a concern for low-power operation. Other low-power techniques that degrade APS performance such as silicidation and thin-film devices can be controlled by slight process modifications. Silicide blocks eliminate the opaque, low-resistance layer; and thin-film devices, such as those found in fully depleted SOI, can be avoided by opening windows through the SOI-buried oxide to the bulk silicon.

MEMS

Electrothermal properties of MEMS suspended or cantilevered layers are finding applications in a variety of electromechanical systems. For example, suspended layers have been developed for a number of purposes, including microphones,[38] accelerometers, and pressure sensors. By sealing a fluid within a MEMS cavity, pressure sensors can also serve as infrared detectors and temperature sensors. Analog feedback electronics monitor the deflection of a MEMS layer caused by the influence of external stresses. The applied control voltage provides an analog readout of the relative magnitude of the external stresses.

MEMS micropumps have been developed which have potential applications in medical products (e.g., drug delivery) and automotive systems (e.g., fuel injection).[39] Flow rates of about 50 μl/min at 1-Hz cycling have been achieved, but the overall area required is still large (approximately 1 cm^2), presently limiting integration. Beyond macro applications, microfluidics has been proposed as a technique for integrated cooling of high-power and high-temperature ICs in which coolant is circulated within the substrate mass.

1.4 Alternative Technologies

The discussion so far has assumed a VLSI system to be comprised primarily of transistors, with the exception of non-electronic MEMS. Although this implementation has evolved unchallenged for nearly 50 years, several innovative technologies have progressed to the state that they are attracting serious attention as future competitors. Some, such as quantum computing, are still fundamentally electronic in nature. Others, like biological, DNA, and molecular computing, use living cells as elemental functional units.

The chief advantage of these technologies is the extreme power efficiency of a computation. Quantum computing achieves ultra-low-power operation by reducing logic operations to the change in an electron spin or an atomic ionization state. Biological and DNA computing exploit the energy efficiency of living cells, the product of a billion years of evolution. A second attribute is extremely fast computation owing to greatly reduced signal path lengths (to a molecular or atomic scale) and massively parallel (MP) simultaneous operations.

Quantum Computing

Quantum computing (QC) is concerned with the probabilistic nature of quantum states, by which a single atom can be used to "store" and "compare" multiple values simultaneously. QC has two distinct implementations: an electronic one based on the wave-function interaction of fixed adjacent atoms[40] and a biochemical one based on mobile molecular interactions within a fluid medium.[41] In fixed systems, atom placement and stimulation are accomplished with atomic force microscopy (AFM) and nuclear magnetic resonance (NMR), but performance as a system also requires the ability to

individually select and operate on an atom.[42] Molecular systems avoid this issue by using a fluid medium as a method of introducing initial conditions and isolating the computation from the measurement. Computational redundancy then statistically removes measurement error and incorrect results generated at the fluid boundaries.

Recent work in algorithms has demonstrated that QC can solve two categories of problems more efficiently than with a classical computer science method by taking advantage of wave function indeterminate states. In search and factorization problems (involving a random search of N items), a classical solution requires $O(N)$ steps, but a QC algorithm requires only $O(\sqrt{N})$; and binary parity computations can be improved from $O(N)$ to $O(N/2)$ in a QC algorithm.[43]

Practical implementation of QC algorithms to very large data sets is currently limited by instrumentation. For example, a biochemical QC system using NMR to change spin polarity has signal frequencies of less than 1 kHz. Despite this low frequency, the available parallelism is expected to factor (with the $O(\sqrt{N})$ algorithm) a 400-digit number in one year: greater than 3×10^{186} MOPS (mega-operations per second).[44]

DNA Computing

In DNA computing, a problem set is encoded into DNA strands which then, by nucleotide matching properties, perform MP search and comparison operations. Most comparison techniques to date rely on conformal mapping, but some reports appearing in the literature indicate that more powerful DNA algorithms are possible with non-conformal mapping and secondary protein interaction.[45] The challenge is in developing a formal language to describe DNA computing compounded by accounting for these secondary and tertiary effects, including protein structure and amino acid chemical properties.[46] Initial work in formal language theory has shown that DNA computers can be made equivalent to a Turing machine.[47]

DNA can also provide extremely dense data storage, requiring about a trillionth of the volume required for an equivalent electronic memory: 10^{12} DNA strands, each 1000 units long, is equal to 1000 T bits.[48] DNA computations employ up to 10^{20} DNA strands (over 11 million TB), well beyond the capacity of any conventional data storage.

Molecular Computing

Molecular computers are a mixed-signal system for performing logic functions (with possible subpicosecond switching) and signal detection with an ability to evolve and adapt to new conditions. Possible implementations include modulation of electron, proton, or photon mobility; electronic-conformation interactions; and tissue membrane interactions.[49] Table 1.3 lists some of the architectures and applications.

"Digital" cell interactions, such as found in quantum or DNA computing, provide the logic implementation. Analog processing is introduced by the nonlinear characteristics of the cell interaction with respect to light, electricity, magnetism, chemistry, or other external stimulus. Molecular systems are also thought to emulate a neural network that can be capable of signal enhancement and noise removal.[50]

TABLE 1.3 Summary of Some Architectures and Applications Possible from a Molecular Computing System

Mechanisms and Architectures	Applications
Light-energy transducing proteins	Biosensors
Light-energy transducing proteins (with controlled switching)	Organic memory storage
Optoelectronic transducing	Pattern recognition and processing
Evolutionary structures	Adaptive control

References

1. McShane, E., Trivedi, M., Xu, Y., Khandelwal, P., Mulay, A., and Shenai, K., "Low-Power Systems on a Chip (SOC)," *IEEE Circuits and Devices Magazine*, vol. 14, no. 5, pp. 35-42, June 1998.
2. Laes, E., "Submicron CMOS Technology — The Enabling Tool for System-on-a-Chip Integration," *Alcatel Telecommunications Review*, no. 2, pp. 130-137, 1996.
3. Ackland, B., "The Role of VLSI in Multimedia," *IEEE J. of Solid-State Circuits*, vol. 29, no. 4, pp. 381-388, Apr. 1994.
4. Kuroda, I. and Nishitani, T., "Multimedia Processors," *Proc. of the IEEE*, vol. 86, no. 6, pp. 1203-1221, Jun. 1998.
5. Clemens, J. T, "Silicon Microelectronics Technology," *Bell Labs Technical Journal*, vol. 2, no. 4, pp. 76-102, Fall 1997.
6. Asai, S. and Wada, Y., "Technology Challenges for Integration Near and Below 0.1 Micron [Review]," *Proc. of the IEEE*, vol. 85, no. 4, pp. 505-520, Apr. 1997.
7. Tsividis, Y., *Mixed Analog-Digital VLSI Devices and Technology: An Introduction*, McGraw-Hill, New York, 1996.
8. Gonzalez, R., Gordon, B.M., and Horowitz, M. A., "Supply and Threshold Voltage Scaling for Low Power CMOS," *IEEE J. of Solid-State Circuits*, vol. 32, no. 8, pp. 1210-1216, Aug. 1997.
9. Colinge, J.-P., *Silicon-on-Insulator Technology: Materials to VLSI*, 2nd edition, Kluwer Academic Publishers, Boston, 1997.
10. Assaderaghi, F., Sinitsky, D., Parke, S. A., Bokor, J., Ko, P.K., and Hu, C. M., "Dynamic Threshold-Voltage MOSFET (DTMOS) for Ultra-Low Voltage VLSI," *IEEE Trans. on Electron Devices*, vol. 44, no. 3, pp. 414-422, Mar. 1997.
11. Licata, T.J., Colgan, E.G., Harper, J. M. E., and Luce, S. E., "Interconnect Fabrication Processes and the Development of Low-Cost Wiring for CMOS Products," *IBM J. of Research & Development*, vol. 39, no. 4, pp. 419-435, Jul. 1995.
12. Itoh, K., Sasaki, K., and Nakagome, Y., "Trends In Low-Power RAM Circuit Technologies," *Proc. of the IEEE*, vol. 83, no. 4, pp. 524-543, Apr. 1995.
13. Itoh, K., Nakagome, Y., Kimura, S., and Watanabe, T., "Limitations and Challenges of Multigigabit DRAM Chip Design," *IEEE J. of Solid-State Circuits*, vol. 32, no. 5, pp. 624-634, May 1997.
14. Kim, K., Hwang, C.-G., and Lee, J. G., "DRAM Technology Perspective for Gigabit Era," *IEEE Trans. on Electron Devices*, vol. 45, no. 3, pp. 598-608, Mar. 1998.
15. Cressler, J. D., "SiGe HBT Technology: A New Contender for Si-Based RF and Microwave Circuit Applications," *IEEE Trans. on Microwave Theory & Techniques*, vol. 46, no. 5 Part 2, pp. 572-589, May 1998.
16. Hafizi, M., "New Submicron HBT IC Technology Demonstrates Ultra-Fast, Low-Power Integrated Circuits," *IEEE Trans. on Electron Devices*, vol. 45, no. 9, pp. 1862-1868, Sept. 1998.
17. Wang, N. L. L., "Transistor Technologies for RFICs in Wireless Applications," *Microwave Journal*, vol. 41, no. 2, pp. 98-110, Feb. 1998.
18. Huang, Q. T., Piazza, F., Orsatti, P., and Ohguro, T.,"The Impact of Scaling Down to Deep Submicron on CMOS RF Circuits," *IEEE J. of Solid-State Circuits*, vol. 33, no. 7, pp. 1023-1036, Jul. 1998.
19. Larson, L. E., "Integrated Circuit Technology Options for RFICs — Present Status and Future Directions," *IEEE J. of Solid-State Circuits*, vol. 33, no. 3, pp. 387-399, Mar. 1998.
20. Hung, C.-M., Ho, Y. C., Wu, I.-C., and K. O, "High-Q Capacitors Implemented in a CMOS Process for Low-Power Wireless Applications," *IEEE Trans. on Microwave Theory & Techniques*, vol. 46, no. 5 Part 1, pp. 505-511, May 1998.
21. Yue, C. P. and Wong, S. S., "On-Chip Spiral Inductors with Patterned Ground Shields for Si-Based RF IC's," *IEEE J. of Solid-State Circuits*, vol. 33, no. 5, pp. 743-752, May 1998.
22. Chang, J. Y.-C., Abidi, A. A., and Gaitan, M.,"Large Suspended Inductors on Silicon and Their Use in a 2-mm CMOS RF Amplifier," *IEEE Electron Device Lett.*, vol. 14, pp. 246-248, May 1993.

23. Mohan, N., Undeland, T. M., and Robbins, W. P., *Power Electronics: Converters, Applications, and Design*, 2nd edition, John Wiley & Sons, New York, 1996.
24. Tsui, P. G. Y., Gilbert, P. V., and Sun, S. W., "A Versatile Half-Micron Complementary BiCMOS Technology for Microprocessor-Based Smart Power Applications," *IEEE Trans. on Electron Devices*, vol. 42, no. 3, pp. 564-570, Mar. 1995.
25. Chan, W. W. T., Sin, J. K. O., and Wong, S. S., "A Novel Crosstalk Isolation Structure for Bulk CMOS Power ICs," *IEEE Trans. on Electron Devices*, vol. 45, no. 7, pp. 1580-1586, Jul. 1998.
26. Baliga, J., "Power Semiconductor Devices for Variable-Frequency Drives," *Proc. of the IEEE*, vol. 82, no. 8, pp. 1112-1122, Aug. 1994.
27. Mino, M., Yachi, T., Tago, A., Yanagisawa, K., and Sakakibara, K.,"Planar Microtransformer With Monolithically-Integrated Rectifier Diodes For Micro-Switching Converters," *IEEE Trans. on Magnetics*, vol. 32, no. 2, pp. 291-296, Mar. 1996.
28. Sunaga, T., Miyatake, H., Kitamura, K., Kogge, P. M., and Retter, E., "A Parallel Processing Chip with Embedded DRAM Macros," *IEEE J. of Solid-State Circuits*, vol. 31, no. 10, pp. 1556-1559, Oct. 1996.
29. Watanabe, T., Fujita, R., Yanagisawa, K., Tanaka, H., Ayukawa, K., Soga, M., Tanaka, Y., Sugie, Y., and Nakagome, Y.,"A Modular Architecture for a 6.4-Gbyte/S, 8-Mb DRAM-Integrated Media Chip," *IEEE J. of Solid-State Circuits*, vol. 32, no. 5, pp. 635-641, May 1997.
30. Miyano, S., Numata, K., Sato,K., Yabe, T., Wada, M., Haga, R., Enkaku, M., Shiochi, M., Kawashima, Y., Iwase, M., Ohgata, M., Kumagai, J., Yoshida, T., Sakurai, M., Kaki, S., Yanagiya, N., Shinya, H., Furuyama, T., Hansen, P., Hannah, M., Nagy, M., Nagarajan, A., and Rungsea, M., "A 1.6 Gbyte/sec Data Transfer Rate 8 Mb Embedded DRAM," *IEEE J. of Solid-State Circuits*, vol. 30, no. 11, pp. 1281-1285, Nov. 1995.
31. Bang, S. H., Choi, J., Sheu, B. J., and Chang, R. C., "A Compact Low-Power VLSI Transceiver for Wireless Communication," *IEEE Trans. on Circuits & Systems I—Fundamental Theory & Applications*, vol. 42, no. 11, pp. 933-945, Nov. 1995.
32. Rofougaran, A., Chang, J.G., Rael, J., Chang, J. Y.-C., Rofougaran, M., Chang, P. J., Djafari, M., Ku, M. K., Roth, E. W., Abidi, A. A., and Samueli, H., "A Single-Chip 900-MHz Spread-Spectrum Wireless Transceiver in 1-micron CMOS — Part I: Architecture and Transmitter Design," *IEEE J. of Solid-State Circuits*, vol. 33, no. 4, pp. 515-534, Apr. 1998.
33. Rofougaran, A., Chang, G., Rael, J. J., Chang, J. Y.-C., Rofougaran, M., Chang, P. J., Djafari, M., Min, J., Roth, E. W., Abidi, A. A., and Samueli, H., "A Single-Chip 900-MHz Spread-Spectrum Wireless Transceiver in 1-micron CMOS — Part II: Receiver Design," *IEEE J. of Solid-State Circuits*, vol. 33, no. 4, pp. 535-547, Apr. 1998.
34. Rofougaran, A., Chang, J. Y. C., Rofougaran, M., and Abidi, A. A., "A 1 GHz CMOS RF Front-End IC for a Direct-Conversion Wireless Receiver," *IEEE J. of Solid-State Circuits*, vol. 31, no. 7, pp. 880-889, Jul. 1996.
35. Meyer, R. G. and Mack, W. D. , "A 1-GHz BiCMOS RF Front-End IC," *IEEE J. of Solid-State Circuits*, vol. 29, no. 3, pp. 350-355, Mar. 1994.
36. Mendis, S., Kemeny, S. E., and Fossum, E. R. , "CMOS Active Pixel Image Sensor," *IEEE Trans. on Electron Devices*, vol. 41, no. 3, pp. 452-453, Mar. 1994.
37. Fossum, E. R.,"CMOS Image Sensors — Electronic Camera-On-a-Chip," *IEEE Trans. on Electron Devices*, vol. 44, no. 10, pp. 1689-1698, Oct. 1997.
38. Pederson, M., Olthuis, W., and Bergveld, P., "High-Performance Condenser Microphone with Fully Integrated CMOS Amplifier and DC-DC Voltage Converter," *IEEE J. of Microelectromechanical Systems*, vol. 7, no. 4, pp. 387-394, Dec. 1998.
39. Benard, W. L., Kahn, H., Heuer, A. H., and Huff, M. A. , "Thin-Film Shape-Memory Alloy Actuated Micropumps," *IEEE J. of Microelectromechanical Systems*, vol. 7, no. 2, pp. 245-251, Jun. 1998.
40. Brassard, G., Chuang, I., Lloyd, S., and Monroe, C., "Quantum Computing," *Proc. of the National Academy of Sciences of the USA*, vol. 95, no. 19, pp. 11032-11033, Sept. 15, 1998.

41. Wallace, R., Price, H., and Breitbeil, F., "Toward a Charge-Transfer Model of Neuromolecular Computing," *Int'l J. of Quantum Chemistry*, vol. 69, no. 1, pp. 3-10, Jul. 1998.
42. Scarani, V., "Quantum Computing," *American J. of Physics*, vol. 66, no. 11, pp. 956-960, Nov. 1998.
43. Grover, L. K., "Quantum Computing — Beyond Factorization And Search," *Science*, vol. 281, no. 5378, pp. 792-794, Aug. 1998.
44. Gershenfeld, N. and Chuang, I. L., "Quantum Computing with Molecules," *Scientific American*, vol. 278, no. 6, pp. 66-71, Jun. 1998.
45. Conrad, M. and Zauner, K. P., "DNA as a Vehicle for the Self-Assembly Model of Computing," *Biosystems*, vol. 45, no. 1, pp. 59-66, Jan. 1998.
46. Rocha, A. F., Rebello, M. P., and Miura, K., "Toward a Theory of Molecular Computing," *J. of Information Sciences*, vol. 106, pp. 123-157, 1998.
47. Kari, L., Paun, G., Rozenberg, G., Salomaa, A., Yu, S., "DNA Computing, Sticker Systems, and Universality," *Acta Informatica*, vol. 35, no. 5, pp. 401-420, May 1998.
48. Forbes, N. A. and Lipton, R. J.,"DNA Computing — A Possible Efficiency Boost for Specialized Problems," *Computers in Physics*, vol. 12, no. 4, pp. 304-306, Jul.-Aug. 1998.
49. Kampfner, R. R.,"Integrating Molecular and Digital Computing — An Information Systems Design Perspective," *Biosystems*, vol. 35, no. 2-3, pp. 229-232, 1995.
50. Rambidi, N. G., "Practical Approach to Implementation of Neural Nets at the Molecular Level," *Biosystems*, vol. 35, no. 2-3, pp. 195-198, 1995.

Katsumata, Y. “CMOS/BiCMOS Technology”
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

2

CMOS/BiCMOS Technology

Yasuhiro Katsumata
Tatsuya Ohguro
Kazumi Inoh
Eiji Morifuji
Takashi Yoshitomi
Hideki Kimijima
Hideaki Nii
Toyota Morimoto
Hisayo S. Momose
Kuniyoshi Yoshikawa
Hidemi Ishiuchi
Toshiba Corporation

Hiroshi Iwai
Tokyo Institute of Technology

2.1 Introduction

Silicon LSIs (large-scale integrated circuits) have progressed remarkably in the past 25 years. In particular, complementary metal-oxide-semiconductor (CMOS) technology has played a great role in the progress of LSIs. By downsizing[2] MOS field-effect-transistors (FETs), the number of transistors in a chip increases, and the functionality of LSIs is improved. At the same time, the switching speed of MOSFETs and circuits increases and operation speed of LSIs is improved.

On the other hand, system-on-chip technology has come into widespread use and, as a result, the LSI system requires several functions, such as logic, memory, and analog functions. Moreover, the LSI system sometimes needs an ultra-high-speed logic or an ultra-high-frequency analog function. In some cases, bipolar-CMOS (BiCMOS) technology is very useful.

The first part of this chapter focuses on CMOS technology as the major LSI process technology, including embedded memory technology. The second part, describes BiCMOS technology; and finally, future process technology is introduced.

2.2 CMOS Technology

Device Structure and Basic Fabrication Process Steps

Complementary MOS (CMOS) was first proposed by Wanlass and Sah in 1963.[1] Although the CMOS process is more complex than the NMOS process, it provides both n-channel (NMOS) and p-channel (PMOS) transistors on the same chip, and CMOS circuits can achieve lower power consumption. Consequently, the CMOS process has been widely used as an LSI fabrication process.

Figure 2.1 shows the structure of a CMOS device. Each FET consists of a gate electrode, source, drain, and channel, and gate bias controls carrier flow from source to drain through the channel layer.

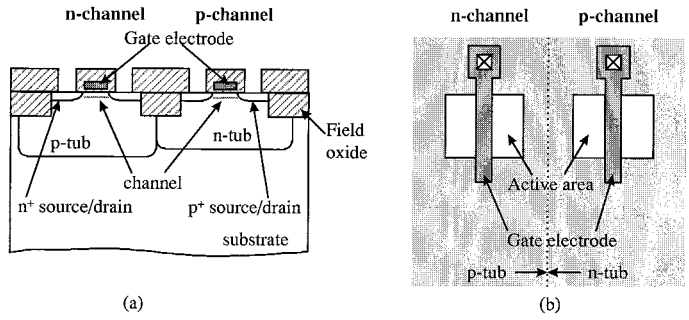


FIGURE 2.1 Structure of CMOS device: (a) cross-sectional view of CMOS, (b) plain view of CMOS.

Figure 2.2 shows the basic fabrication process flow. The first process step is the formation of p tub and n tub (twin tub) in silicon substrate. Because CMOS has two types of FETs, NMOS is formed in p tub and PMOS in n tub.

The isolation process is the formation of field oxide in order to separate each MOSFET active area in the same tub. After that, impurity is doped into channel region in order to adjust the threshold voltage, V_{th}, for each type of FET. The gate insulator layer, usually silicon dioxide (SiO_2), is grown by thermal oxidation, because the interstate density between SiO_2 and silicon substrate is small. Polysilicon is deposited as gate electrode material and gate electrode is patterned by reactive ion etching (RIE).

The gate length, L_g, is the critical dimension because L_g determines the performance of MOSFETs and it should be small in order to improve device performance. Impurity is doped in the source and drain regions of MOSFETs by ion implantation. In this process step, gate electrodes act as a self-aligned mask to cover channel layers. After that, thermal annealing is carried out in order to activate the impurity of diffused layers.

In the case of high-speed LSI, the self-aligned silicide (salicide) process is applied for the gate electrode and source and drain diffused layers in order to reduce parasitic resistance. Finally, the metallization process is carried out in order to form interconnect layers.

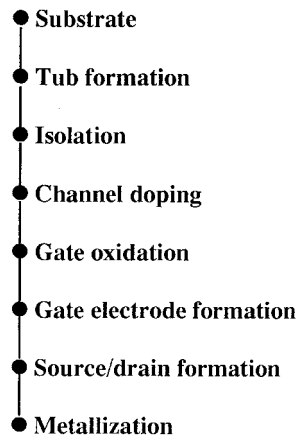


FIGURE 2.2 Basic process flow of CMOS.

Key Process Steps in Device Fabrication

Starting Material

Almost all silicon crystals for LSI applications are prepared by the Czochralski crystal growth method,[2] because it is advantageous for the formation of large wafers. (100) orientation wafers are usually used

for MOS devices because their interstate trap density is smaller than those of (111) and (110) orientations.[3] The light doping in the substrate is convenient for the diffusion of tub and reduces the parasitic capacitance between silicon substrate and tub region. As a starting material, lightly doped (~10^{15} atoms/cm^3) p-type substrate is generally used.

Tub Formation

Figure 2.3 shows the tub structures, which are classified into 6 types: p tub, n tub, twin tub,[4] triple tub, twin tub with buried p^+ and n^+ layers, and twin tub on p-epi/p^+ substrate. In the case of the p tub process, NMOS is formed in p diffusion (p tub) in the n substrate, as shown in Fig. 2.3(a). The p tub is formed by implantation and diffusion into the n substrate at a concentration that is high enough to overcompensate the n substrate.

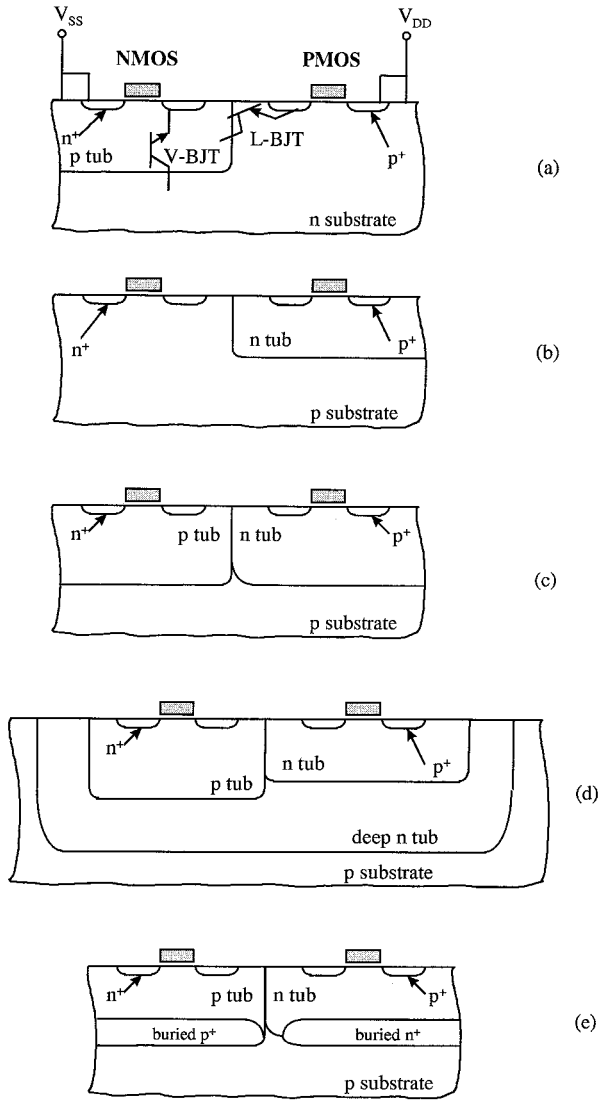


FIGURE 2.3 Tub structures of CMOS: (a) p tub; (b) n tup; (c) twin tub; (d) triple tub; (e) twin tub with buried p^+ and n^+ layers; and (f) twin tub on p-epi/p^+ substrate.

The other approach is to use an n tub.[5] As shown in Fig. 2.3(b), NMOS is formed in the p substrate.

Figure 2.3(c) shows the twin-tub structure,[4] that uses two separate tubs implanted into silicon substrate. In this case, doping profiles in each tub region can be controlled independently, and thus neither type of device suffers from excess doping effect.

In some cases, such as mixed signal LSIs, a deep n tub layer is sometimes formed optionally, as shown in Fig. 2.3(d), in order to prevent the crosstalk noise between digital and analog circuits. In this structure, both n and p tubs are electrically isolated from the substrate or other tubs on the substrate.

In order to realize high packing density, the tub design rule should be shrunk; however, an undesirable mechanism, the well-known latch-up, might occur.

Latch-up (i.e., the flow of high current between V_{DD} and V_{SS}) is caused by parasitic lateral pnp bipolar (L-BJT) and vertical npn bipolar (V-BJT) transistor actions[6] as shown in Fig. 2.3(a), and it sometimes destroys the functions of LSIs. The collectors of each of these bipolar transistors feed each others' bases and together make up a pnpn thyristor structure. In order to prevent latch-up, it is important to reduce the current gain, h_{FE}, of these parasitic bipolar transistors, and the doping concentration of the tub region should be higher. As a result, device performance might be suppressed because of large junction capacitances.

In order to solve this problem, several techniques have been proposed, such as p^+ or n^+ buried layer under p tub[7] as shown in Fig. 2.3(e), the use of high-dose, high-energy boron p tub implants,[8,9] and the shunt resistance for emitter-base junctions of parasitic bipolar transistors.[7,10,11] It is also effective to provide many well contacts to stabilize the well potential and hence to suppress the latch-up. Recently, substrate with p epitaxial silicon on p^+ substrate, can also be used to stabilize the potential for high-speed logic LSIs.[12]

Isolation

Local oxidation of silicon (LOCOS)[13] is a widely used isolation process, because this technique can allow channel-stop layers to be formed self-aligned to the active transistor area. It also has the advantage of recessing about half of the field oxide below the silicon surface, which makes the surface more planar.

Figure 2.4 shows the LOCOS isolation process. First, silicon nitride and pad oxide are etched for the definition of active transistor area. After channel implantation as shown in Fig. 2.4(a), the field oxide is selectively grown, typically to a thickness of several hundred nanometers.

A disadvantage of LOCOS is that involvement of nitrogen in the masking of silicon nitride layer sometimes causes the formation of a very thin nitride layer in the active region, and this often impedes the subsequent growth of gate oxide, thereby causing low gate breakdown voltage of the oxides. In order to prevent this problem, after stripping the masking silicon nitride, a sacrificial oxide is grown and then removed before the gate oxidation process.[14,15]

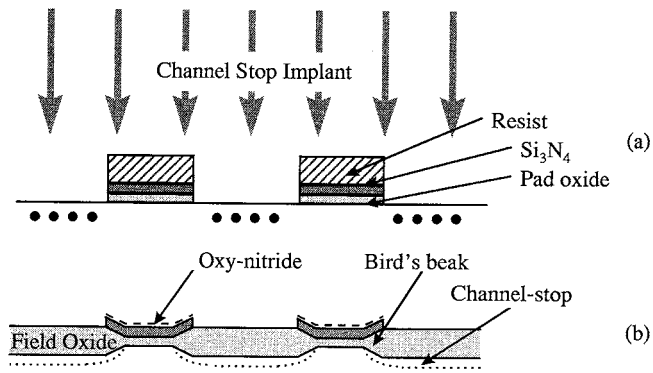


FIGURE 2.4 Process for local oxidation of silicon: (a) after silicon nitride/pad oxide etch and channel-stop implant; (b) after field oxidation, which produces an oxynitride film on nitride.

In addition, the lateral spread of field oxide (bird's beak)[14] poses a problem regarding reduction of the distance between active transistor areas in order to realize high packing density. This lateral spread is suppressed by increasing the thickness of silicon nitride and/or decreasing the thickness of pad oxide. However, there is a tradeoff with the generation of dislocation of silicon.

Recently, shallow trench isolation (STI)[16] has become a major isolation process for advanced CMOS devices. Figure 2.5 shows the process flow of STI. After digging the trench into the substrate by RIE as shown in Fig. 2.5(a), the trench is filled with insulator such as silicon dioxide as shown in Fig. 2.5(b). Finally, by planarization with chemical mechanical polishing (CMP),[17] filling material on the active transistor area is removed, as shown in Fig. 2.5(c).

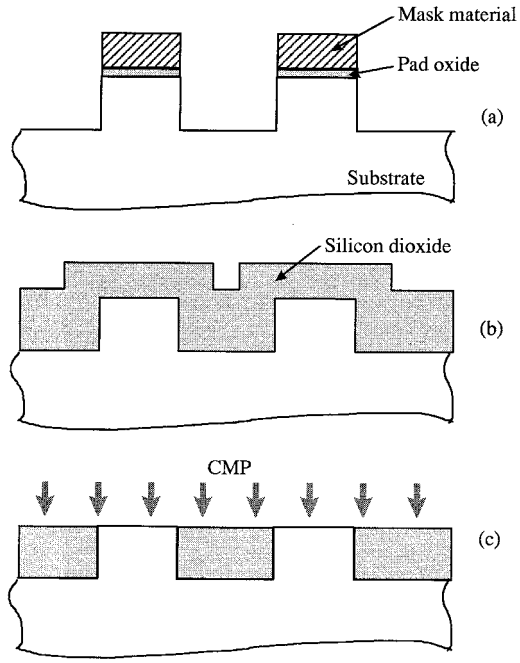


FIGURE 2.5 Process flow of STI: (a) trenches are formed by RIE; (b) filling by deposition of SiO_2; and (c) planarization by CMP.

STI is a useful technique for downsizing not only the distance between active areas, but also the active region itself. However, a mechanical stress problem[18] still remains, and several methods have been proposed[19] to deal with it.

Channel Doping

In order to adjust the threshold voltage of MOSFETs, V_{th}, to that required by a circuit design, the channel doping process is usually required. The doping is carried out by ion implantation, usually through a thin dummy oxide film (10 to 30 nm) thermally grown on the substrate in order to protect the surface from contamination, as shown in Fig. 2.6. This dummy oxide film is removed prior to the gate oxidation. Figure 2.7 shows a typical CMOS structure with channel doping. In this case, n^+ polysilicon gate electrodes are used for both n- and p-MOSFETs and, thus, this type of CMOS is called single-gate CMOS. The role of the channel doping is to enhance or raise the threshold voltage of n-MOSFETs. It is desirable to keep the concentration of p tub lower in order to reduce the junction capacitance of source and drain. Thus, channel doping of p-type impurity — boron — is required. Drain-to-source leakage current in short-channel MOSFETs flows in a deeper path, as shown in Fig. 2.8; this is called the short-channel effect. Thus, heavy doping of the deeper region is effective in suppressing the short-channel effect. This doping is called deep ion implantation.

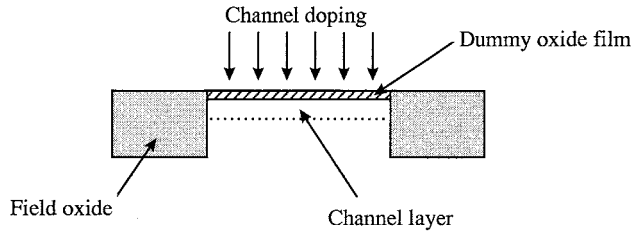


FIGURE 2.6 Channel doping process step.

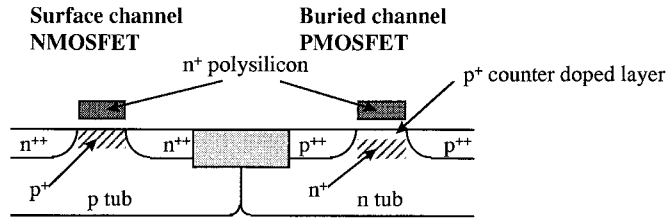


FIGURE 2.7 Schematic cross-section of single-gate CMOS structure.

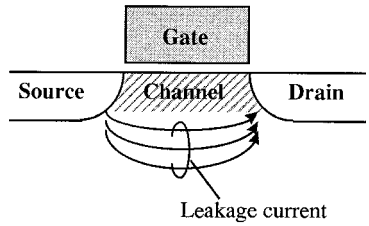


FIGURE 2.8 Leakage current flow in short-channel MOSFET.

In the case of p-MOSFET with an n^+ polysilicon gate electrode, the threshold voltage becomes too high in the negative direction if there is no channel doping. In order to adjust the threshold voltage, an ultra-shallow p-doped region is formed by the channel implantation of boron. This p-doped layer is often called a counter-doped layer or buried-channel layer, and p-MOSFETs with this structure are called buried-channel MOSFETs. (On the other hand, MOSFETs without a buried-channel layer are called surface-channel MOSFETs. n-MOSFETs in this case are the surface-channel MOSFETs.) In the buried-channel case, the short-channel effect is more severe, and, thus, deep implantation of an n-type impurity such as arsenic or phosphorus is necessary to suppress them.

In deep submicron gate length CMOS, it is difficult to suppress the short-channel effect,[20] and thus, a p^+-polysilicon electrode is used for p-MOSFETs, as shown in Fig. 2.9. For n-MOSFETs, an n^+-polysilicon electrode is used. Thus, this type of CMOS is called dual-gate CMOS. In the case of p^+-polysilicon p-MOSFET, the threshold voltage becomes close to 0 V because of the difference in work function between n- and p-polysilicon gate electrode,[21–23] and thus, buried layer is not required. Instead, n-type impurity channel doping such as arsenic is required to raise the threshold voltage slightly in the negative direction.

Impurity redistribution during high-temperature LSI manufacturing processes sometimes makes channel profile broader, which causes the short-channel effect. In order to suppress the redistribution, a dopant with a lower diffusion constant, such as indium, is used instead of boron.

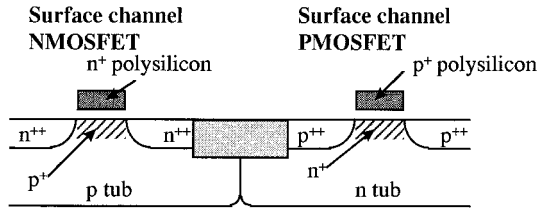


FIGURE 2.9 Schematic cross-section of dual-gate CMOS structure.

For the purpose of realizing a high-performance transistor, it is important to reduce junction capacitance. In order to realize lower junction capacitance, a localized diffused channel structure.[24,25] as shown in Fig. 2.10, is proposed. Since the channel layer exists only around the gate electrode, the junction capacitance of source and drain is reduced significantly.

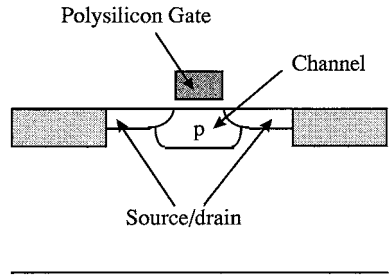


FIGURE 2.10 Localized channel structure.

Gate Insulator

The gate dielectric determines several important properties of MOSFETs and thus uniformity in its thickness, low defect density of the film, low fixed charge and interface state density at the dielectric and silicon interface, small roughness at the interface, high reliability of time-dependent dielectric breakdown (TDDB) and hot-carrier induced degradation, and high resistivity to boron penetration (explained in this section) are required. As a consequence of downsizing of MOSFET, the thickness of the gate dielectric has become thinner. Generally, the thickness of the gate oxide is 7 to 8 nm for 0.4-μm gate length MOSFETs, and 5 to 6 nm for 0.25-μm gate length MOSFETs.

Silicon dioxide (SiO_2) is commonly used for gate dielectrics, and can be formed by several methods, such as dry O_2 oxidation,[26] and wet or steam (H_2O) oxidation.[26] The steam is produced by the reaction of H_2 and O_2 ambient in the furnace. Recently, H_2O oxidation has been widely used for gate oxidation because of good controllability of oxide thickness and high reliability.

In the case of the dual-gate CMOS structure shown in Fig. 2.9, boron penetration from the p^+ gate electrode to the channel region through the gate silicon dioxide, which is described in the following section, is a problem. In order to prevent this problem, oxynitride has been used as the gate dielectric material.[27,28] In general, the oxynitride gate dielectric is formed by the annealing process in NH_3, NO (or N_2O) after silicon oxidation, or by direct oxynitridation of silicon in NO (or N_2O) ambient. Figure 2.11 shows the typical nitrogen profile of the oxynitride gate dielectric. Recently, remote plasma nitridation[29,30] has been much studied, and it is reported that the oxynitride gate dielectric grown by the remote plasma method showed better quality and reliability than that grown by the silicon nitridation method.

In the regime of a sub-quarter-micron CMOS device, gate oxide thickness is close to the limitation of tunneling current flow, around 3 nm thickness. In order to prevent tunneling current, high κ materials,

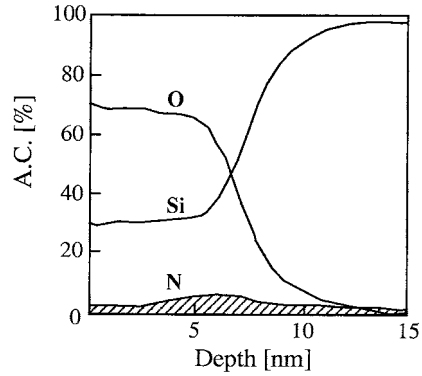


FIGURE 2.11 Oxygen, nitrogen, and silicon concentration profile of oxynitride gate dielectrics measued by AES.

such as Si_3N_4[31] and Ta_2O_5,[32] are proposed instead of silicon dioxide. In these cases, the thickness of the gate insulator can be kept relatively thick because high κ insulator realizes high gate capacitance, and thus better driving capability.

Gate Electrode

Heavily doped polysilicon has been widely used for gate electrodes because of its resistance to high-temperature LSI fabrication processing. In order to reduce the resistance of the gate electrode, which contributes significantly to RC delay time, silicides of refractory metals have been put on the polysilicon electrode.[33,34] Polycide,[34] the technique of combining a refractory metal silicide on top of doped polysilicon, has the advantage of preserving good electrical and physical properties at the interface between polysilicon and the gate oxide while, at the same time, the sheet resistance of gate electrode is reduced significantly.

For doping the gate polysilicon, ion implantation is usually employed. In the case of heavy doping, dopant penetration from boron-doped polysilicon to the Si substrate channel region through the gate oxide occurs in the high-temperature LSI fabrication process, as shown in Fig. 2.12. (On the other hand, usually, penetration of an n-type dopant [such as phosphorus or arsenic] does not occur.) When the doping of impurities in the polysilicon is not sufficient, the depletion of the gate electrode occurs as shown in Fig. 2.13, resulting in a significant decrease of the drive capability of the transistor, as shown in Fig. 2.14.[35] There is a tradeoff between the boron penetration and the gate electrode depletion, and so thermal process optimization is required.[36]

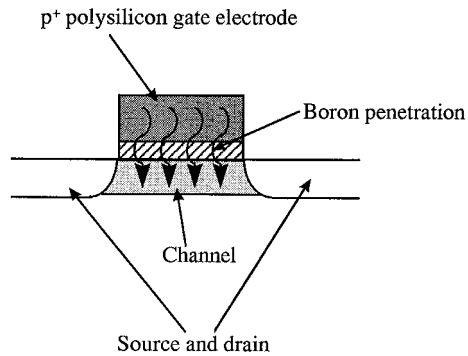


FIGURE 2.12 Dopant penetration from boron-doped polysilicon to silicon substrate channel region.

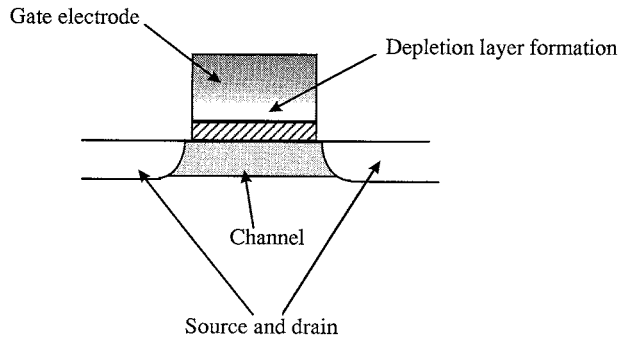


FIGURE 2.13 Depletion of gate electrode in the case that the doping of impurities in the gate electrode is not sufficient.

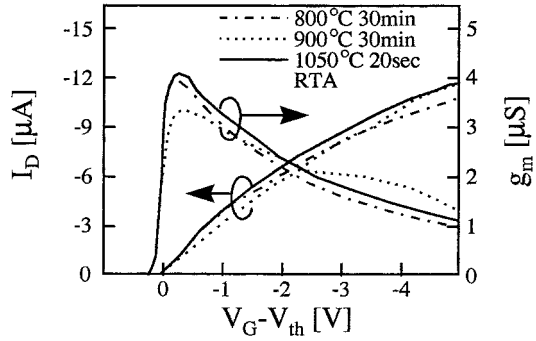


FIGURE 2.14 I_D, $g_m - V_G$ characteristics for various thermal conditions. In the case of 800°C/30 min, a significant decrease in drive capability of transistor occurs because of the depletion of the gate electrode.

Gate length is one of most important dimensions defining MOSFET performance; thus, the lithography process for gate electrode patterning requires high-resolution technology.

In the case of a light-wave source, the g-line (wavelength 436 nm) and the i-line (365 nm) of a mercury lamp were popular methods. Recently, a higher-resolution process, excimer laser lithography, has been used. In the excimer laser process, KrF (248 nm)[37] and ArF (193 nm)[38] have been proposed and developed. For a 0.25-μm gate length electrode, the KrF excimer laser process is widely used in the production of devices. In addition, electron-beam[39–41] and X-ray[42] lithography techniques are being studied for sub-0.1 μm gate electrodes.

For the etching of gate polysilicon, a high-selectivity RIE process is required for selecting polysilicon from SiO_2 because a gate dielectric beneath polysilicon is a very thin film in the case of recent devices.

Source/Drain Formation

Source and drain diffused layers are formed by the ion implantation process. As a consequence of transistor downsizing, at the drain edge (interface of channel region and drain) where reverse biased pn junctions exist, a higher electrical field has been observed. As a result, carriers across these junctions are suddenly accelerated and become hot carriers, which creates a serious reliability problem for MOSFET.[43]

In order to prevent the hot carrier problem, the lightly doped drain (LDD) structure is proposed.[44] The LDD process flow is shown in Fig. 2.15. After gate electrode formation, ion implantation is carried out to make extension layers, and the gate electrode plays the role of self-aligned mask that covers the channel layer, as shown in Fig. 2.15(b). In general, arsenic is doped for n-type extension of NMOS, and BF_2 for p-type extension of PMOS. To prevent the short-channel effect, the impurity profile of extension layers must

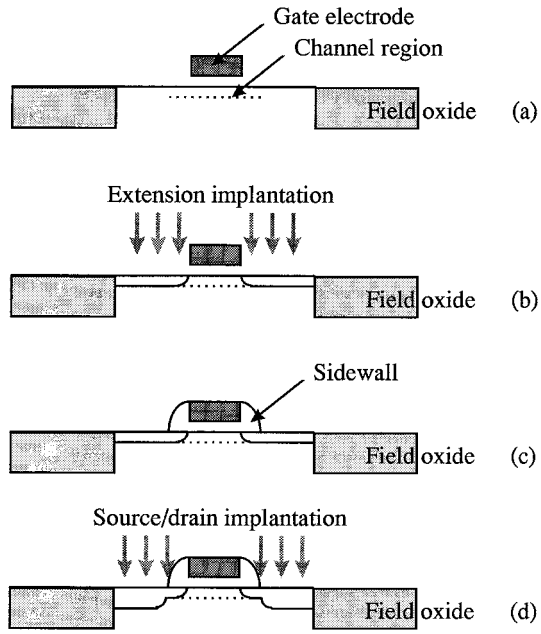


FIGURE 2.15 Process flow of LDD structure: (a) after gate electrode patterning; (b) extension implantation; (c) sidewall spacer formation; and (d) source/drain implantation.

be very shallow. Although shallow extension can be realized by ion implantation with low dose, the resistivity of extension layers becomes higher and, thus, MOSFET characteristics degrade. Hence, it is very difficult to meet these two requirements. Also, impurities diffusion in this extension affects the short-channel effect significantly. Thus, it is necessary to minimize the thermal process after forming the extension.

Insulating film, such as Si_3N_4 or SiO_2, is deposited by a chemical vapor deposition method. Then, etching back RIE treatment is performed on the whole wafer; as a result, the insulating film remains only at the gate electrode side, as shown in Fig. 2.15(c). This remaining film is called a sidewall spacer. This spacer works as a self-aligned mask for deep source/drain n^+ and p^+ doping, as shown in Fig. 2.15(d). In general, arsenic is doped for deep source/drain of n-MOSFET, and BF_2 for p-MOSFET. In the dual-gate CMOS process, gate polysilicon is also doped in this process step to prevent gate electrode depletion.

After that, in order to make doped impurities activate electrically and recover from implantation damage, an annealing process, such as rapid thermal annealing (RTA), is carried out.

According to the MOSFET scaling law, when gate length and other dimensions are shrunk by factor k, the diffusion depth also needs to be shrunk by $1/k$. Hence, the diffusion depth of the extension part is required to be especially shallow.

Several methods have been proposed for forming an ultra-shallow junction. For example, very low accelerating voltage implantation, the plasma doping method,[45] and implantation of heavy molecules, such as $B_{10}H_{14}$ for p-type extension,[46] are being studied.

Salicide Technique

As the vertical dimension of transistors is reduced with device downscaling, an increase is seen in sheet resistance — both of the diffused layers, such as source and drain, and the polysilicon films, such as the gate electrode. This is becoming a serious problem in the high-speed operation of integrated circuits.

Figure 2.16 shows the dependence of the propagation delay (t_{pd}) of CMOS inverters on the scaling factor, k, or gate length.[47] These results were obtained by simulations in which two cases were considered. First is the case in which source and drain contacts with the metal line were made at the edge of the diffused layers, as illustrated in the figure inset. In an actual LSI layout, it often happens that the metal

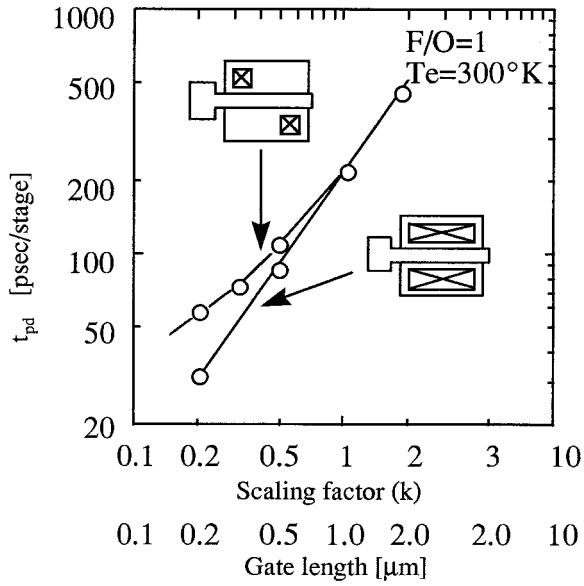


FIGURE 2.16 Dependence of the propogation delay (t_{pd}) of CMOS inverters on the scaling factor, k, or gate length.

contact to the source or drain can be made only to a portion of the diffused layers, since many other signal or power lines cross the diffused layers. The other case is that in which the source and drain contacts cover the entire area of the source and drain layers, thus reducing diffused line resistance. It is clear that without a technique to reduce the diffused line resistance, t_{pd} values cannot keep falling as transistor size is reduced; they will saturate at gate lengths of around a 0.25 microns.

In order to solve this problem — the high resistance of shallow diffused layers and thin polysilicon films — self-aligned silicide (salicide) structures for the source, drain, and gate have been proposed, as shown in Fig. 2.17.[48–50]

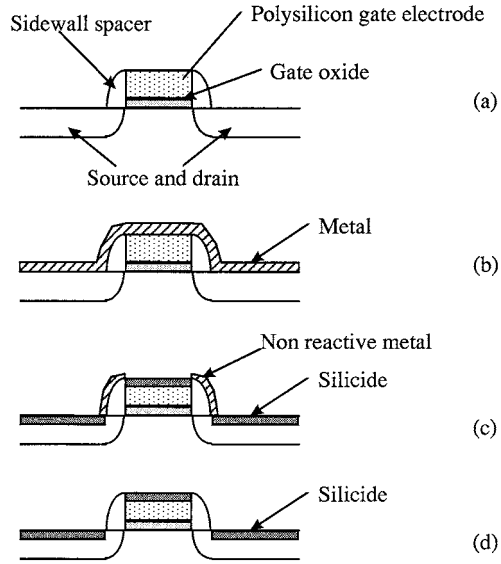


FIGURE 2.17 A typical process flow and schematic cross-section of salicide process: (a) MOSFET formation; (b) metal deposition; (c) silicidation by thermal annealing; and (d) removal of non-reactive metal.

First, a metal film such as Ti or Co is deposited on the surface of the MOSFET after formation of the polysilicon gate electrode, gate sidewall, and source and drain diffused layers, as shown in Fig. 2.17(b). The film is then annealed by rapid thermal annealing (RTA) in an inert ambient. During the annealing process, the areas of metal film in direct contact with the silicon layer — that is, the source, drain, and gate electrodes — are selectively converted to the silicide, and other areas remain metal, as shown in Fig. 2.17(c). The remaining metal can be etched off with an acid solution such as $H_2O_2 + H_2SO_4$, leaving the silicide self-aligned with the source, drain, and gate electrode, as shown in Fig. 2.17(d).

When the salicide process first came into use, furnace annealing was the most popular heat-treatment process[48–50]; however, RTA[51–53] replaced furnace annealing early on, because it is difficult to prevent small amounts of oxidant from entering through the furnace opening, and these degrade the silicide film significantly since silicide metals are easily oxidized. On the other hand, RTA reduces this oxidation problem significantly, resulting in reduced deterioration of the film and consequently of its resistance.

At present, $TiSi_2$[51–53] is widely used as a silicide in LSI applications. However, in the case of ultra-small geometry MOSFETs for VLSIs, use of $TiSi_2$ is subject to several problems. When the $TiSi_2$ is made thick, a large amount of silicon is consumed during silicidation, and this results in problems of junction leakage at the source or drain. On the contrary, if a thin layer of $TiSi_2$ is chosen, agglomeration of the film occurs[54] at higher silicidation temperatures.

On the other hand, $CoSi_2$[55] has a large silicidation temperature window for low sheet resistance; hence, it is expected to be widely used as silicidation material for advanced VLSI applications.[47]

Interconnect and Metallization

Aluminum is widely used as a wiring metal. However, in the case of downsized CMOS, electromigration (EM)[56] and stress migration (SM)[57] become serious problems. In order to prevent these problems, Al-Cu (typically ~0.5 wt % Cu)[58] is a useful wiring material. In addition, ultra-shallow junction for downsized CMOS sometimes needs barrier metal,[58] such as TiN, between the metal and silicon, in order to prevent junction leakage current.

Figure 2.18 shows a cross-sectional view of a multi-layer metallization structure. As a consequence of CMOS downscaling, contact or via aspect ratio becomes larger; and, as a result, filling of contact or via is not sufficient. Hence, new filling techniques, such as W-plug,[59,60] are widely used.

In addition, considering both reliability and low resistivity, Cu is a useful wiring material.[61] In the case of Cu is used, metal thickness can be reduced in order to realize the same interconnect resistance. The reduction of the metal thickness is useful for reducing the capacitance between the dense interconnect wires, resulting in the high-speed operation of the circuit. In order to reduce RC delay of wire

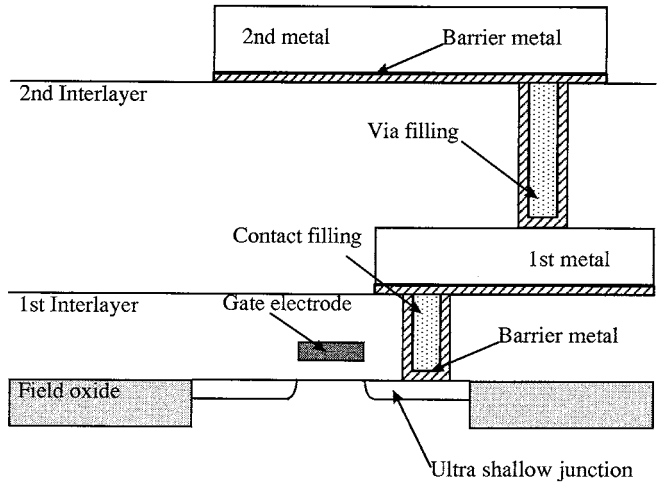


FIGURE 2.18 Cross-sectional view of multi-layer metallization.

in CMOS LSI, not only wiring material but also interlayer material is important. In particular, low-κ material[62] is widely studied.

In the case of Cu wiring, the dual damascene process[63] is being widely studied because it is difficult to realize fine Cu pattern by reactive ion etching. Figure 2.19 shows the process flow of Cu dual

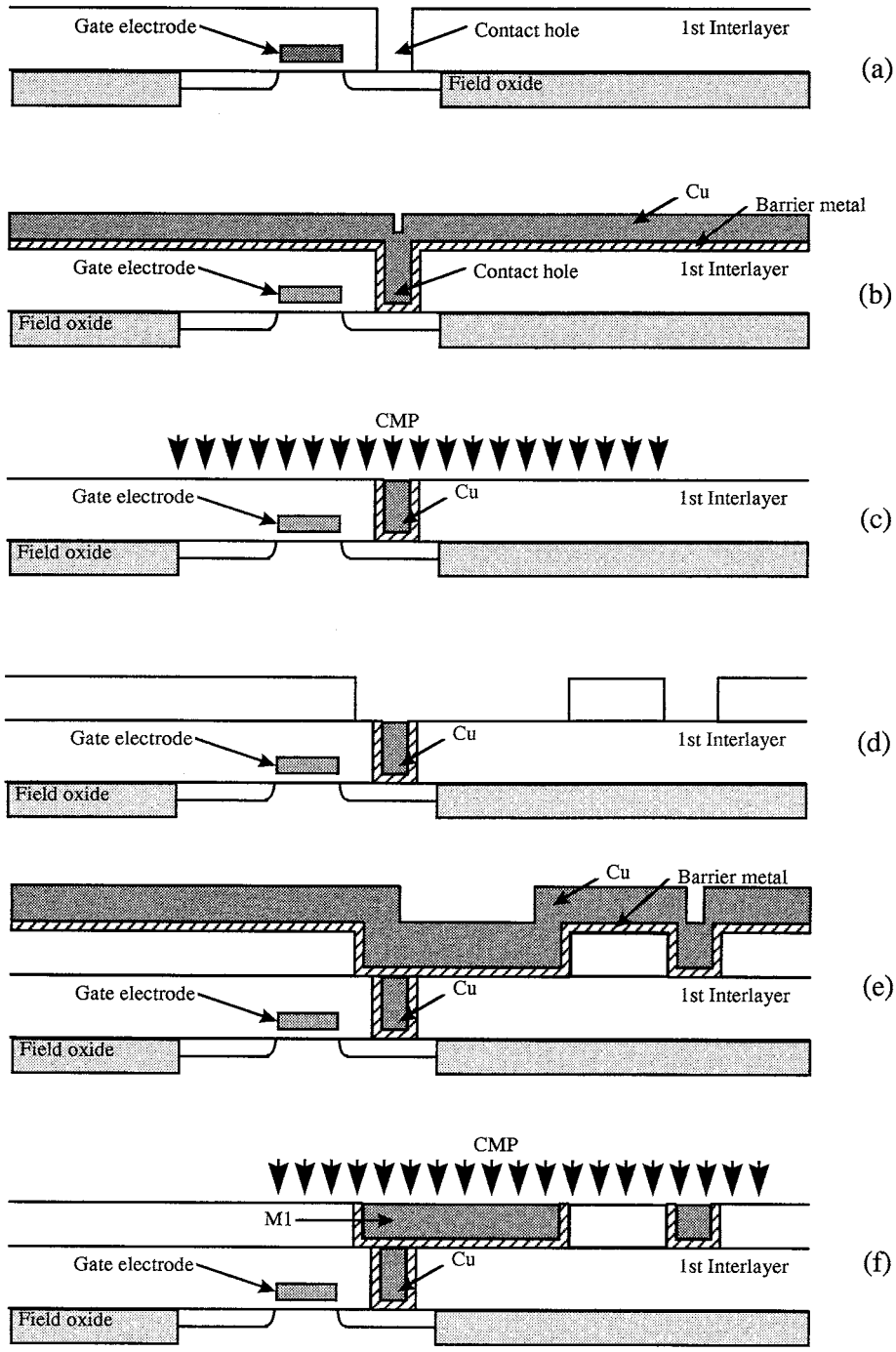


FIGURE 2.19 Typical process flow of Cu dual damascene.

damascene metallization. After formation of transistors and contact holes as shown in Fig. 2.19(a), barrier metal, such as TiN, and Cu are deposited as shown in Fig. 2.19(b). By using the CMP planarization process, Cu and barrier metal remains in the contact holes, as shown in Fig. 2.19(c). Insulator, such as silicon dioxide, is deposited and the grooves for first metal wires are formed by reactive ion etching, as shown in Fig. 2.19(d). After the deposition of barrier metal and Cu as shown in Fig. 2.19(e), Cu and barrier metal remain only in the wiring grooves due to use of a planarization process such as CMP, as shown in Fig. 2.19(f).

Passive Device for Analog Operation

System-on-chip technology has come into widespread use; and as a result, an LSI system sometimes requires analog functions. In this case, analog passive devices should be integrated,[64] as shown in Fig. 2.20.

Resistors and capacitors already have good performance, even for high-frequency applications. On the other hand, it is difficult to realize a high-quality inductor on a silicon chip because of inductance loss in Si substrate, in which the resistivity is lower than that in the compound semiconductor, such as GaAs, substrate. The relatively higher sheet resistance of aluminum wire used for high-density LSI is another problem. Recently, quality of inductor has been improved by using thicker Al or Cu wire[65] and by optimizing the substrate structure.[66]

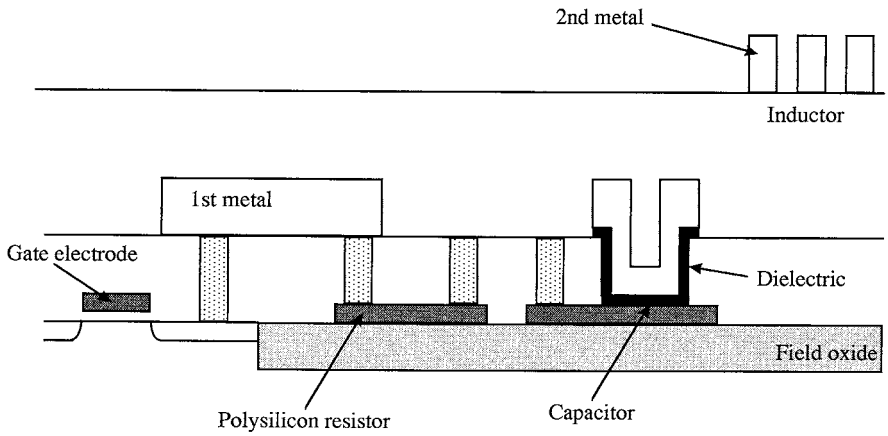


FIGURE 2.20 Various passive devices for analog application.

Embedded Memory Technology

Embedded DRAM

There has been strong motivation to merge DRAM cell arrays and logic circuits into a single silicon chip. This approach makes it possible to realize high bandwidth between memory and logic, low power consumption, and small footprint of the chip.[67] In order to merge logic and DRAM into a single chip, it is necessary to establish process integration for the embedded DRAM. Figure 2.21 shows a typical structure of embedded DRAM. However, the logic process and the DRAM process are not compatible with each other. There are many variations and options in constructing a consistent process integration for the embedded DRAM.

Trench Capacitor Cell versus Stacked Capacitor Cell

There are two types of DRAM cell structure: stacked capacitor cell[68–73] and trench capacitor cell.[74,75]

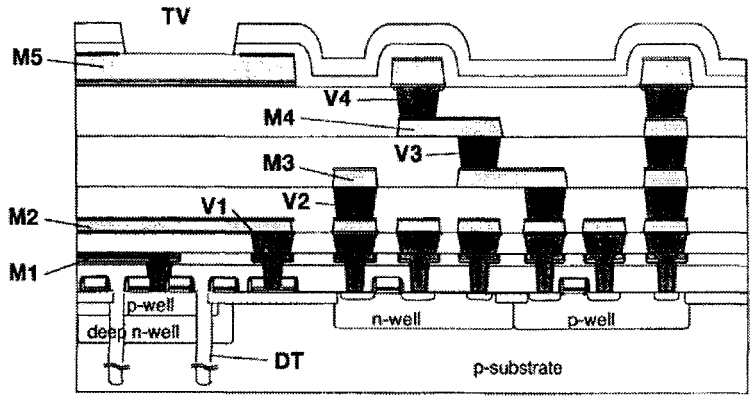


FIGURE 2.21 Schematic cross-section of the embedded DRAM, including DRAM cells and logic MOSFETs.

In trench cell technology, the cell capacitor process is completed before gate oxidation. Therefore, there is no thermal process due to cell capacitor formation after the MOSFET formation. Another advantage of the trench cell is that there is little height difference between the cell array region and the peripheral circuit region.[76–79]

In the stacked capacitor cell, the height difference high aspect ratio contact holes and difficulty in the planarization process after cell formation. The MOSFET formation steps are followed by the stacked capacitor formation steps, which include high-temperature process steps such as storage node insulator (SiO_2/Si_3N_4) formation, and Si_3N_4 deposition for the self-aligned contact formation. The salicide process for the source and drain of the MOSFETs should be carefully designed to endure the high-temperature process steps. Recently, high-permittivity film for capacitor insulators, such as Ta_2O_5 and BST, has been developed for commodity DRAM and embedded DRAM. The process temperature for Ta_2O_5 and BST is lower than that for SiO_2/Si_3N_4; this means the process compatibility is better with such high-permittivity film.[80–82]

MOSFET Structure

The MOSFET structure in DRAMs is different from that in logic ULSIs. In recent DRAMs, the gate is covered with Si_3N_4 for self-aligned contact process steps in the bit-line contact formation. It is very difficult to apply the salicide process to the gate, source, and drain at the same time. A solution to the problem is to apply the salicide process to the source and drain only. A comparison of the MOSFET structures is shown in Fig. 2.22. Tsukamoto et al.[68] proposed another approach, namely the use of W-bit line layer as the local interconnect in the logic portion.

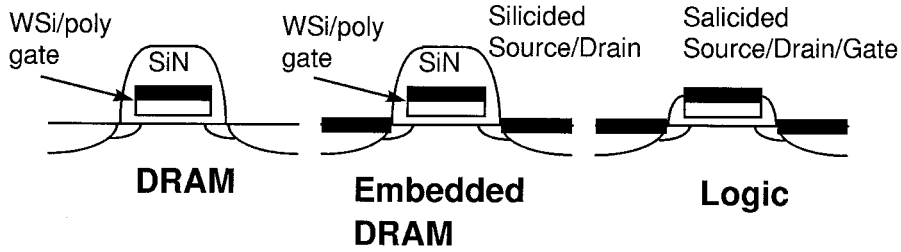


FIGURE 2.22 Typical MOSFET structures for DRAM, embedded DRAM, and logic.

Gate Oxide Thickness

Generally, DRAM gate oxide thickness is greater than that of logic ULSIs. This is because the maximum voltage of the transfer gate in the DRAM cells is higher than V_{CC}, the power supply voltage. In the logic ULSI, the maximum gate voltage is equal to V_{CC} in most cases. To keep up with the MOSFET performance in logic ULSIs, the oxide thickness of the embedded DRAMs needs to be scaled down further than in the DRAM case. To do so, a highly reliable gate oxide and/or new circuit scheme in the word line biasing, such as applying negative voltage to the cell transfer gate, is required.

Another approach is to use thick gate oxide in the DRAM cell and thin gate oxide in the logic.[83]

Fabrication Cost per Wafer

The conventional logic ULSIs do not need the process steps for DRAM cell formation. On the other hand, most of DRAMs use only two layers of aluminum. This raises wafer cost of the embedded DRAMs. Embedded DRAM chips are used only if the market can absorb the additional wafer cost for some reasons: high bandwidth, lower power consumption, small footprint, flexible memory configuration, lower chip assembly cost, etc.

Next-Generation Embedded DRAM

Process technology for the embedded DRAM with 0.18-μm or 0.15-μm design rules will include state-of-the-art DRAM cell array and high-performance MOSFETs in the logic circuit. The embedded DRAM could be a technology driver because the embedded DRAM contains most of the key process steps for DRAM and logic ULSIs.

Embedded Flash Memory Technology[84]

Recently, the importance of embedded flash technology has been increasing and logic chips with non-volatile functions have become indispensable for meeting various market requirements.

Key issues in the selection of an embedded flash cell[85] are (1) tunnel-oxide reliability (damage-less program/erase(P/E) mechanism), (2) process and transistor compatibility with CMOS logic, (3) fast read with low *Vcc*, (4) low power (especially in P/E), (5) simple control circuits, (6) fast program speed, and (7) cell size. This ordering greatly depends on target device specification and memory density, and, in general, is different from that of high-density stand-alone memories. NOR-type flash is essential and EEPROM functionality is also required on the same chip. Figure 2.23 shows the typical device structure of a NOR-type flash memory with logic device.[86]

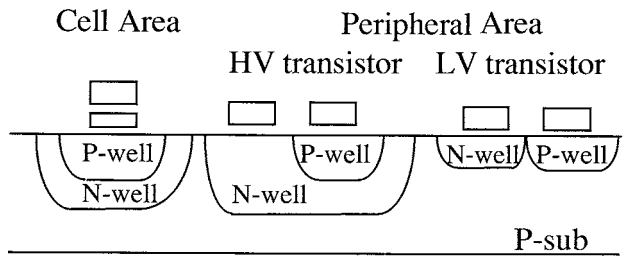


FIGURE 2.23 Device structure schematic view of the NOR flash memories with dual-gate Ti-salicide.

Process Technology[87]

To realize high-performance embedded flash chips, at least three kinds of gate insulators are required beyond the 0.25-μm regime in order to form flash tunnel oxide, CMOS gate oxide, high voltage transistor gate oxide, and I/O transistor gate oxide. Flash cells are usually made by a stacked gate process. Therefore, it is difficult to achieve less than 150% of the cost of pure logic devices.

The two different approaches to realize embedded flash chips are memory-based and logic-based, as shown in Fig. 2.24.

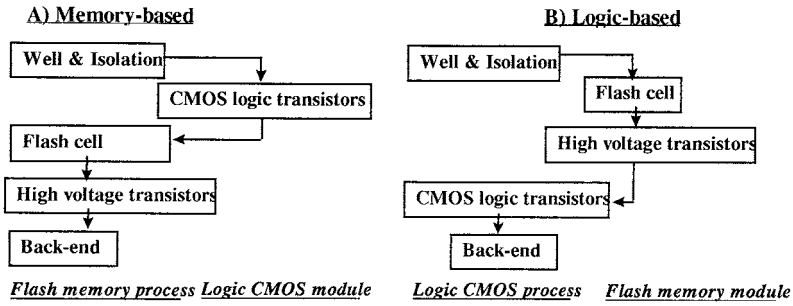


FIGURE 2.24 Process modules.

Memory-based approach is advantageous in that it exploits established flash reliability and yield guaranteed by memory mass production lines, but is disadvantageous for realizing high-performance CMOS transistors due to the additional flash process thermal budget. On the contrary, logic-based approach can use fully CMOS-compatible transistors as they are; but, due to the lack of dedicated mass production lines, great effort is required in order to establish flash cell reliability and performance. Historically, memory-based embedded flash chips have been adopted, but the logic-based chips have become more important recently. In general, the number of additional masks required to embed a flash cell into logic chips ranges from 4 to 9.

For high-density embedded flash chips, one transistor stack gate cell using channel hot electron programming and channel FN tunneling erasing will be mainstream. For medium- or low-density, high-speed embedded flash chips, two transistors will be important in the case of using the low power P/E method. From the reliability point of view, a p-channel cell using band-to-band tunneling-induced electron injection[88] and channel FN tunneling ejection are promising since page-programmable EEPROM can also be realized by this mechanism.[85]

2.3 BiCMOS Technology

The development of BiCMOS technology began in the early 1980s. In general, bipolar devices are attractive because of their high speed, better gain, better driving capability, and low wide-band noise properties that allow high-quality analog performance. CMOS is particularly attractive for digital applications because of its low power and high packing density. Thus, the combination would not only lead to the replacement and improvement of existing ICs, but would also provide access to completely new circuits.

Figure 2.25 shows a typical BiCMOS structure.[89] Generally, BiCMOS has a vertical npn bipolar transistor, a lateral pnp transistor, and CMOS on the same chip. Furthermore, if additional mask steps are

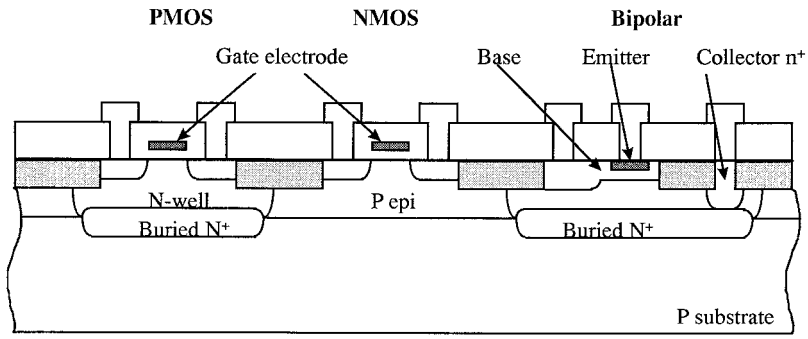


FIGURE 2.25 Cross-sectional view of BiCMOS structure.

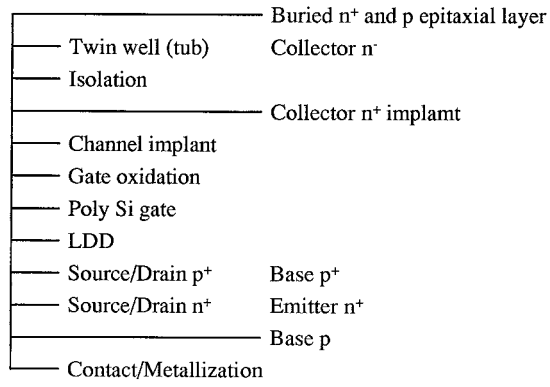


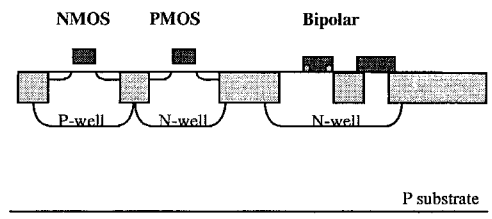
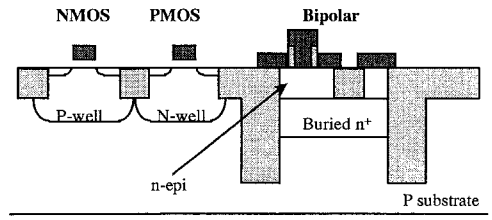
FIGURE 2.26 Typical process flow of BiCMOS device.

allowed, passive devices are integrated, as described in the previous section. The main feature of the BiCMOS structure is the existence of a buried layer because bipolar processes require an epitaxial layer grown on a heavily doped n^+ subcollector to reduce collector resistance.

Figure 2.26 shows typical process flow for BiCMOS. This is the simplest arrangement for incorporating bipolar devices and a kind of low-cost BiCMOS. Here, the BiCMOS process is completed with minimum additional process steps required to form the npn bipolar device, transforming the CMOS baseline process into a full BiCMOS technology. For this purpose, many processes are merged. The p tub of n-MOSFET shares an isolation of bipolar devices, the n tub of p-MOSFET device is used for the collector, the n^+ source and drain are used for the emitter regions and collector contacts, and also extrinsic base contacts have the p^+ source and drain of PMOS device for common use.

Recently, there have been two significant uses of BiCMOS technology. One is high-performance MPU[90] by using the high driving capability of bipolar transistor; the other is mixed signal products that utilize the excellent analog performance of the bipolar transistor, as shown in Table 2.1.

TABLE 2.1 Recent BiCMOS Structures

Type	Structure	Future
Digital BiCMOS		Simplified bipolar. Merged well or diffused layer. For digital application.
Mixed-signal BiCMOS		Double poly self-aligned bipolar, including Si or SiGe base. Non-merged process. For mixed-signal application.

For the high-performance MPU, merged processes were commonly used, and the mature version of the MPU product has been replaced by CMOS LSI. However, this application has become less popular now with reduction in the supply voltage. Mixed-signal BiCMOS requires high performance, especially with respect to f_T, f_{max}, and low noise figure. Hence, a double polysilicon structure with a silicon[91] or SiGe[92] base with trench isolation technology is used.

The fabrication cost of BiCMOS is a serious problem and, thus, a low-cost mixed-signal BiCMOS process[93] has also been proposed.

2.4 Future Technology

In this section, advanced technologies for realizing future downsized CMOS devices are introduced.

Ultra-Thin Gate Oxide MOSFET

From a performance point of view, ultra-thin gate oxide in a direct-tunneling regime is desirable for future LSIs.[94] In this section, the potential and possibility are discussed.

Figure 2.27 shows a TEM cross-section a 1.5-nm gate oxide. Figure 2.28 shows I_d-V_d characteristics for 1.5-nm gate oxide MOSFETs with various gate lengths. In the long-channel case, unusual electrical characteristics were observed because of the significant tunneling leakage current through the gate oxide. However, the characteristics become normal as the gate length is reduced because the gate leakage current decreases in proportion to the gate length and the drain current increases in inverse proportion to the gate length.[95,96] Recently, very high drive currents of 1.8 mA/mm and very high transconductances of more than 1.1 S/mm have been reported using a 1.3-nm gate oxide at a supply voltage of 1.5 V.[97] They also operate well at low power and high speed with a low supply voltage in the 0.5-V range.[98]

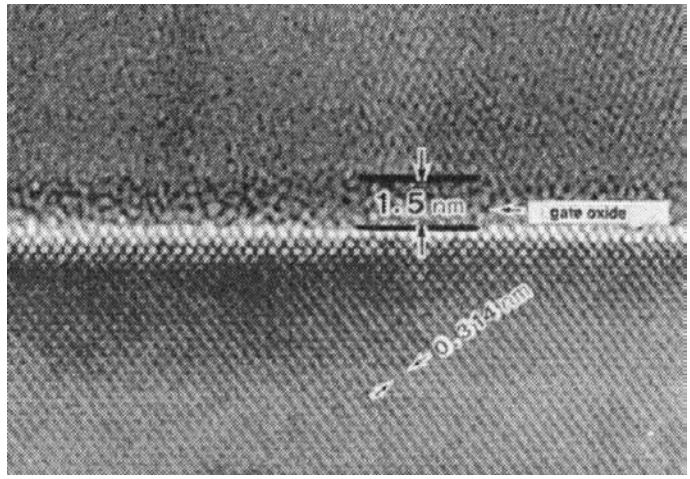


FIGURE 2.27 TEM cross-section of a 1.5-nm gate oxide film. Uniform oxide of 1.5-nm thickness is observed.

Figure 2.29 shows the dependence of cutoff frequency, f_T, of 1.5-nm gate oxide MOSFETs on gate length.[99] Very high cutoff frequencies of more than 150 GHz were obtained at gate lengths in the sub-0.1-μm regime due to the high transconductance. Further, it was confirmed that the high transconductance offers promise of a good noise figure.

Therefore, the MOSFETs with ultra-thin gate oxides beyond the direct-tunneling limit have the potential to enable extremely high-speed digital circuit operation as well as high RF performance in analog

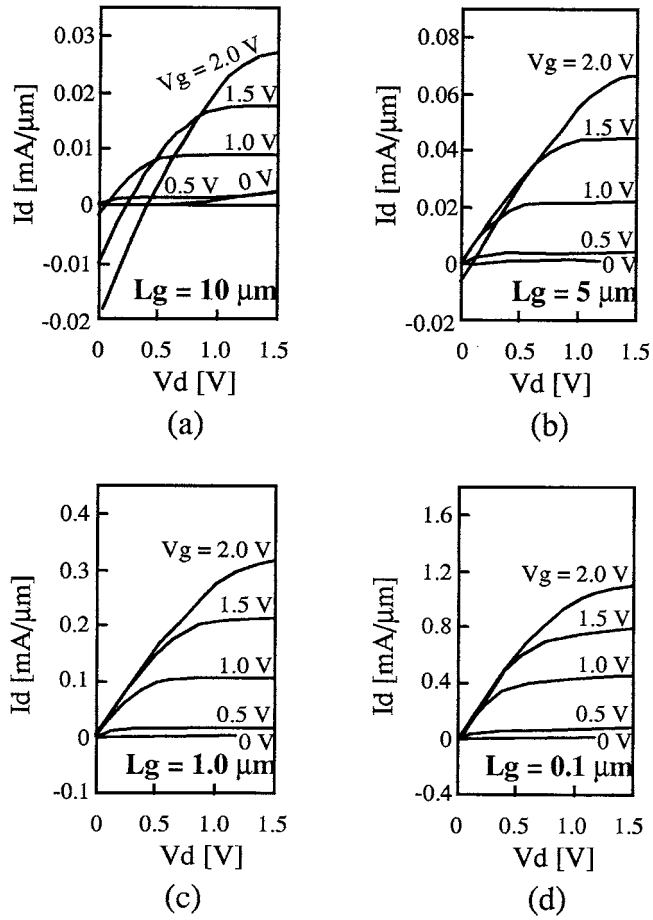


FIGURE 2.28 I_d-V_d characteristics of 1.5-nm gate oxide MOSFETs with several gate lengths: (a) Lg = 10 μm; (b) Lg = 5 μm; (c) Lg = 1.0 μm; and (d) Lg = 0.1 μm.

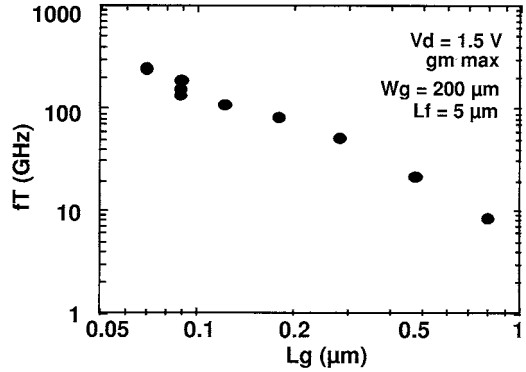


FIGURE 2.29 Dependence of cutoff frequency (f_T) on gate length (L_g) of 1.5-nm gate oxide MOSFETs.

applications. Fortunately, the hot-carrier and TDDB reliability of these ultra-thin gate oxides seems to be good.[95,96,100] Thus, ultra-thin gate oxides are likely to be used for such LSIs, for certain application.

In actual applications, even though the leakage current of a single transistor may be very small, the combined leakage of the huge number of transistors in a ULSI circuit poses problems, particularly for battery backup operation.[101,102] There is, however, the possibility of using these direct-tunneling gate oxide MOSFETs only for the smaller number of switches in the critical path determining operation speed. Also, use of a slightly thicker oxide of 2.0 or 2.5 nm would significantly reduce leakage current. The use of these direct-tunneling gate oxide MOSFETs in LSI devices with smaller integration is another possibility.

Epitaxial Channel MOSFET

As the design rule progresses, the supply voltage decreases. In order to obtain high drivability, lower V_{th} has been required under low supply voltage. However, substrate concentration must be higher in order to suppress the short-channel effects. The ideal channel profile for this requirement is that the channel surface concentration is lower to realize lower V_{th}, and the concentration around extension region is higher in order to suppress the short-channel effects. It is difficult to realize such a channel profile by using ion implantation because the profile is very broad. This requirement can be realized by non-doped epitaxial Si formation on doped Si substrate.[103] Figure 2.30 shows the process flow of MOSFETs with epitaxial Si channel, n channel, and p channel. Although the problem with this structure is the quality of epitaxial Si, degradation of the quality can be suppressed by wet treatment to clean the Si surface, and heating process before epitaxial growth. The zero V_{th} can be realized while suppressing the short-channel effect even when gate length is 0.1 μm. A 20% improvement of drivability can be realized.

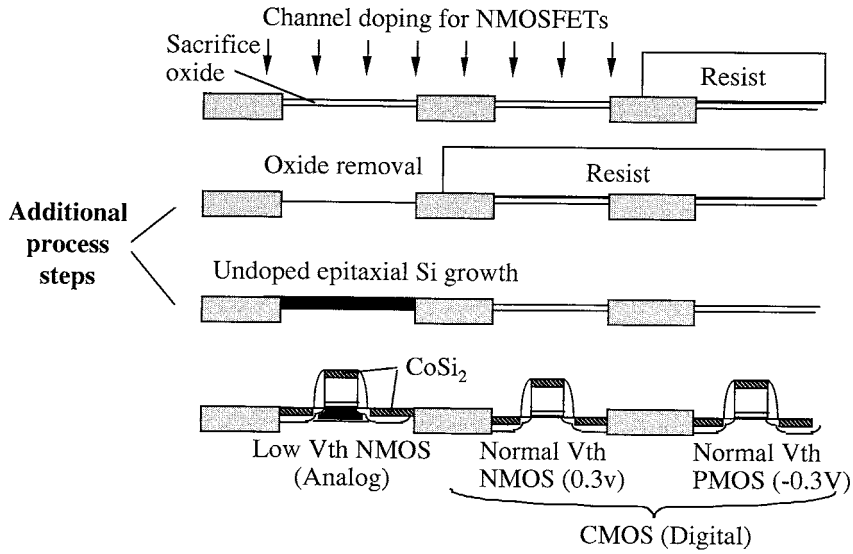


FIGURE 2.30 The process flow of MOSFETs with epitaxial Si channel, n channel, and p channel.

Raised Gate/Source/Drain Structure

In order to realize high drivability of MOSFETs, it is necessary to reduce the resistance under the gate sidewall. However, as the sidewall thickness becomes thinner, the stability of the short-channel effect degrades because the deeper source and drain become closer. If this junction depth becomes shallower, junction leakage degradation occurs because the distance between the bottom of the silicide and the junction becomes shorter. Raised gate/source/drain[104] has been proposed as one way to resolve these problems. The structure is shown in Fig. 2.31.

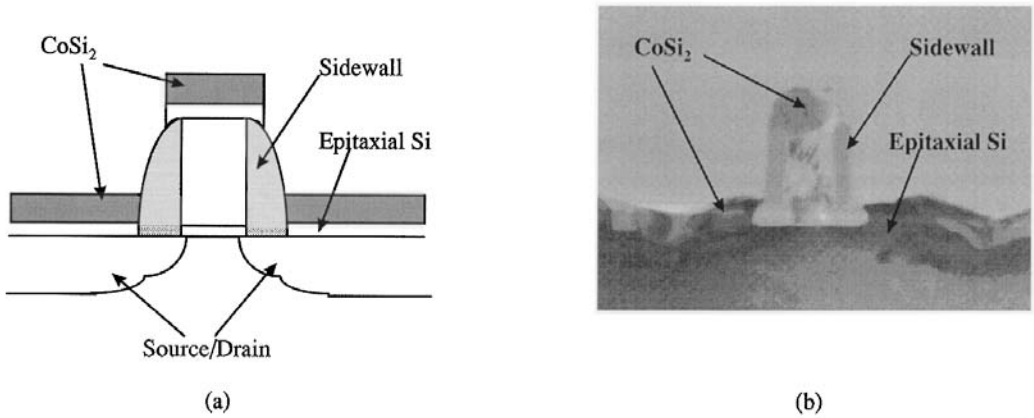


FIGURE 2.31 Structure of raised source/drain/gate FET: (a) schematic cross-section; (b) TEM photograph.

Even if the junction depth measured from the original Si substrate becomes shallower, the distance between the bottom of silicide and the junction becomes constant using this structure. Additionally, gate resistance becomes lower because top of the gate electrode is T-shaped. Thus, low gate, source, and drain resistance can be realized while the short-channel effects and junction leakage current degradation are suppressed.

2.5 Summary

This chapter has described CMOS and BiCMOS technology. CMOS is the most important device structure for realizing the future higher-performance devices required for multimedia and other demanding applications. However, certain problems are preventing the downsizing of device dimensions. The chapter described not only conventional technology but also advanced technology that has been proposed with a view to overcoming these problems.

BiCMOS technology is also important, especially for mixed-signal applications. However, CMOS device performance has already been demonstrated for RF applications and, thus, analog CMOS circuit technology will be very important for realizing the production of analog CMOS.

References

1. Wanlass , F. M. and Sah, C. T., "Nanowatt Logic Using Field Effect Metal-Oxide Semiconductor Triode," *IEEE Solid State Circuits Conf.*, p. 32, Philadelphia, 1963.
2. Rea, S. N., "Czochralski Silicon Pull Rate Limits," *Journal of Crystal Growth*, vol. 54, p. 267, 1981.
3. Sze, S. M., *Physics of Semiconductor Devices, 2nd ed.*, Wiley, New York, 1981.
4. Parrillo, L. C., Payne, R. S., Davis, R. E., Reutlinger, G. W., and Field, R. L., "Twin-Tub CMOS — A Technology for VLSI Circuits," *IEEE International Electron Device Meeting 1980*, p. 752, Washington D.C., 1980.
5. Ohzone, T., Shimura, H., Tsuji, K., and Hirano, "Silicon-Gate n-Well CMOS Process by Full Ion-Implantation Technology," *IEEE Trans. Electron Devices*, vol. ED-27, p. 1789, 1980.
6. Krambeck, R. H., Lee, C. M., and Law, H. F. S. , "High-Speed Compact Circuits with CMOS," *IEEE Journal of Solid State Circuits*, vol. SC-17, p. 614, 1982.
7. Ochoa, A., Dawes, W., and Estreich, "Latchup Control in CMOS Integrated Circuits," *IEEE Trans. Nuclear Science*, vol. NS-26(6), p. 5065, 1979.

8. Rung, R. D., Dell'Oca, C. J., and Walker, L. G., "A Retrograde p-Well for Higher Density CMOS," *IEEE Trans. Electron Devices*, vol. ED-28, p. 1115, 1981.
9. Combs, S. R.,"Scaleable Retrograde p-Well CMOS Technology," *IEEE International Electron Device Meeting 1981*, p. 346, Washington D.C., 1981.
10. Schroeder, J. E., Ochoa Jr., A., and Dressendrfer, P. V., "Latch-up Elimination in Bulk CMOS LSI Circuits," *IEEE Trans. Nuclear Science*, vol. NS-27, p. 1735, 1980.
11. Sakai, Y., Hayashida, T., Hashimoto, N., Mimato, O., Musuhara, T., Nagasawa, K., Yasui, T., and Tanimura, N., "Advanced Hi-CMOS Device Technology," *IEEE International Electron Device Meeting 1981*, p. 534, Washington D.C., 1981.
12. de Werdt, R., van Attekum, P., den Blanken, H., de Bruin, L., op den Buijsch, F., Burgmans, A., Doan, T., Godon, H., Grief, M., Jansen, W., Jonkers, A., Klaassen, F., Pitt, M., van der Plass, P., Stomeijer, A., Verhaar, R., and Weaver, J., "A 1M SRAM with Full CMOS Cells Fabricated in a 0.7 μm Technology," *IEEE International Electron Device Meeting 1987*, p. 532, Washington D.C., 1987.
13. Apples, J. A., Kooi, E., Paffen, M.M., Schlorje, J. J. H., and Verkuylen, W. H. C. G., "Local Oxidation of Silicon and its Application in Semiconductor Technology," *Philips Research Report*, vol. 25, p. 118, 1970.
14. Shankoff, T. A., Sheng, T. T., Haszko, S. E., Marcus, R. B., and Smith, T. E., "Bird's Beak Configuration and Elimination of Gate Oxide Thinning Produced During Selective Oxidation," *Journal of Electrochemical Society*, vol. 127, p. 216, 1980.
15. Nakajima, S., Kikuchi, K., Minegishi, K., Araki, T., Ikuta, K., and Oda, M., "1 μm 256K RAM Process Technology Using Molybdenum-Polysilicon Gate," *IEEE International Electron Device Meeting 1981*, p. 663, Washington D.C., 1981.
16. Fuse, G., Ogawa, H., Tateiwa, K., Nakano, I., Odanaka, S., Fukumoto, M., Iwasaki, H., and Ohzone, T., "A Practical Trench Isolation Technology with a Novel Planarization Process," *IEEE International Electron Device Meeting 1987*, p. 732, Washington D.C., 1987.
17. Perry, K. A. , "Chemical Mechanical Polishing: The Impact of a New Technology on an Industry," *1998 Symposium on VLSI Technology, Digest of Technical Papers*, p. 2, Honolulu, 1998.
18. Kuroi, T., Uchida, T., Horita, K., Sakai, M., Itoh, Y., Inoue, Y., and Nishimura, T., "Stress Analysis of Shallow Trench Isolation for 256M DRAM and Beyond," *IEEE International Electron Device Meeting 1998*, p. 141, San Francisco, CA, 1998.
19. Matsuda, S., Sato, T., Yoshimura, H., Sudo, A., Mizushima, I., Tsumashima, Y., and Toyoshima, Y., "Novel Corner Rounding Process for Shallow Trench Isolation utilizing MSTS (Micro-Structure Transformation of Silicon)," *IEEE International Electron Device Meeting 1998*, p. 137, San Francisco, CA, 1998.
20. Hu, G. J. and Bruce, R. H., "Design Trade-off between Surface and Buried-Channel FETs," *IEEE Trans. Electron Devices*, vol. 32, p. 584, 1985.
21. Cham, K. M., D. W. Wenocur, Lin, J., Lau, C. K., and Hu, H.-S., "Submicronmeter Thin Gate Oxide p-Channel Transistors with p^+ Poly-silicon Gates for VLSI Applications," *IEEE Electron Device Letters*, vol. EDL-7, p. 49, 1986.
22. Amm, D. T., Mingam, H., Delpech, P., and d'Ouville, T. T., "Surface Mobility in p^+ and n^+ Doped Polysilicon Gate PMOS Transistors," *IEEE Trans. Electron Devices*, vol. 36, p. 963, 1989.
23. Toriumi, A., Mizuno, T., Iwase, M., Takahashi, M., Niiyama, H., Fukumoto, M., Inaba, S., Mori, I., and Yoshimi, M., "High Speed 0.1 μm CMOS Devices Operating at Room Temperature," *Extended Abstract of 1992 International Conference on Solid State Devices and Materials*, p. 487, Tsukuba, Japan, 1992.
24. Oyamatsu, H., Kinugawa, M., and Kakumu, M., "Design Methodology of Deep Submicron CMOS Devices for 1V operation," *1993 Symposium on VLSI Technology, Digest of Technical Papers*, p. 89, Kyoto, Japan, 1993.
25. Takeuchi, K., Yamamoto, T., Tanabe, A., Matsuki, T., Kunio, T., Fukuma, M., Nakajima, K., Aizaki, H., Miyamoto, H., and Ikawa, E. , "0.15 μm CMOS with High Reliability and Performance," *IEEE International Electron Device Meeting 1993*, p. 883, Washington D.C., 1993.

26. Ligenza, J. R. and Spitzer, W. G., "The Mechanism for Silicon Oxidation in Steam and Oxygen," *Journal of Phys. Chem. Solids*, vol. 14, p. 131, 1960
27. Morimoto, T. , Momose, H. S., Ozawa, Y., Yamabe, K., and Iwai, H., "Effects of Boron Penetration and Resultant Limitations in Ultra Thin Pure-Oxide and Nitrided-Oxide," *IEEE International Electron Device Meeting 1990*, p. 429, Washington D.C., 1990.
28. Uchiyama, A., Fukuda, H., Hayashi, T. , Iwabuchi, T., and Ohno, S., "High Performance Dual-Gate Sub-Halfmicron CMOSFETs with 6nm-thick Nitrided SiO_2 Films in an N_2O Ambient," *IEEE International Electron Device Meeting 1990*, p. 425, Washington D.C., 1990.
29. Rodder, M., Chen, I.-C., Hattangaly, S., and Hu, J. C., "Scaling to a 1.0V-1.5V, sub 0.1μm Gate Length CMOS Technology: Perspective and Challenges," *Extended Abstract of 1998 International Conference on Solid State Devices and Materials*, p. 158, Hiroshima, Japan, 1998.
30. Rodder, M. , Hattangaly, S., Yu, N., Shiau, W., Nicollian, P. , Laaksonen, T. , Chao, C. P., Mehrotra, M., Lee, C., Murtaza, S. , and Aur, A., "A 1.2V, 0.1μm Gate Length CMOS Technology: Design and Process Issues," *IEEE International Electron Device Meeting 1998*, p. 623, San Francisco, 1998.
31. Khare, M. , Guo, X., Wang, X. W. , and Ma, T. P. , "Ultra-Thin Silicon Nitride Gate Dielectric for Deep-Sub-Micron CMOS Devices," *1997 Symposium on VLSI Technology, Digest of Technical Papers*, p. 51, Kyoto, Japan, 1997.
32. Yagishita, A., Saito, T., Nakajima, K., Inumiya, S., Akasaka, Y., Ozawa, Y., Minamihaba, G. , Yano, H., Hieda, K. , Suguro, K., Arikado, T., and Okumura, K., "High Performance Metal Gate MOSFETs Fabricated by CMP for 0.1 μm Regime," *IEEE International Electron Device Meeting 1998*, p. 785, San Francisco, CA, 1998.
33. Murarka, S. P., Fraser, D. B., Shinha, A. K., and Levinstein, H. J., "Refractory Silicides of Titanium and Tantalum for Low-Resistivity Gates and Interconnects," *IEEE Trans. Electron Devices*, vol. ED-27, p. 1409, 1980.
34. Geipel, H. J. , Jr., Hsieh, N., Ishaq, M. H., Koburger, C. W., and White, F. R., "Composite Silicide Gate Electrode — Interconnections for VLSI Device Technologies," *IEEE Trans. Electron Devices*, vol. ED-27, p. 1417, 1980.
35. Hayashida, H. , Toyoshima, Y. , Suizu, Y. , Mitsuhashi, K., Iwai, H., and Maeguchi, K. , "Dopant Redistribution in Dual Gate W-polycide CMOS and its Improvement by RTA," *1989 Symposium on VLSI Technology, Digest of Technical Papers*, p. 29, Kyoto, Japan, 1989.
36. Uwasawa, K., Mogami, T. , Kunio, T. , and Fukuma, M. , "Scaling Limitations of Gate Oxide in p^+ Polysilicon Gate MOS Structure for Sub-Quarter Micron CMOS Devices," *IEEE International Electron Device Meeting 1993*, p. 895, Washington D.C., 1993.
37. Ozaki, T., Azuma, T., Itoh, M., Kawamura, D., Tanaka, S., Ishibashi, Y., Shiratake, S., Kyoh, S., Kondoh, T., Inoue, S., Tsuchida, K., Kohyama, Y., and Onishi, Y., "A 0.15μm KrF Lithography for 1Gb DRAM Product Using High Printable Patterns and Thin Resist Process," *1998 Symposium on VLSI Technology, Digest of Technical Papers*, p. 84, Honolulu, 1998.
38. Hirukawa, S., Matsumoto, K. , and Takemasa, K. , "New Projection Optical System for Beyond 150 nm Patterning with KrF and ArF Sources," *Proceedings of 1998 International Symposium on Optical Science, Engineering, and Instrumentation, SPIE's 1998 Annual Meeting*, p. 414, 1998.
39. Triumi, A. and Iwase, M., "Lower Submicrometer MOSFETs Fabricated by Direct EB Lithography," *Extended Abstract of the 19th Conference on Solid State Devices and Materials*, p. 347, Tokyo, Japan, 1987.
40. Liddle, J. A. and Berger, S. D., "Choice of System Parameters for Projection Electron-Beam Lithography: Accelerating Voltage and Demagnification Factor," *Journal of Vacuum and Science Technology*, vol. B10(6), p. 2776, 1992.
41. Nakajima, K., Yamashita, H., Kojima, Y., Tamura, T., Yamada, Y., Tokunaga, K., Ema, T., Kondoh, K., Onoda, N., and Nozue, H., "Improved 0.12μm EB Direct Writing for Gbit DRAM Fabrication," *1998 Symposium on VLSI Technology, Digest of Technical Papers*, p. 34, Honolulu, 1998.
42. Deguchi, K., Miyoshi, K., Ban, H., Kyuragi, H., Konaka, S., and Matsuda, T. , "Application of X-ray Lithography with a Single-Layer Resist Process to Subquartermicron LSI Fabrication," *Journal of Vacuum and Science Technology*, vol. B10(6), p. 3145, 1992.

43. Matsuoka, F., Iwai, H., Hayashida, H., Hama, K., Toyoshima, Y., and Maeguchi, K., "Analysis of Hot Carrier Induced Degradation Mode on pMOSFETs," *IEEE Trans. Electron Devices*, vol. ED-37, p. 1487, 1990.
44. Ogura, S., Chang, P. J., Walker, W. W., Critchlow, D. L., and Shepard, J. F., "Design and Characteristics of the Lightly-Doped Drain-Source (LDD) Insulated Gate Field Effect Transistor," *IEEE Trans. Electron Devices*, vol. ED-27, p. 1359, 1980.
45. Ha, J. M., Park, J. W., Kim, W. S., Kim, S. P., Song, W. S., Kim, H. S., Song, H. J., Fujihara, K., Lee, M. Y., Felch, S., Jeong, U., Groeckner, M., Kim, K. H., Kim, H. J., Cho, H. T., Kim, Y. K., Ko, D. H., and Lee, G. C., "High Performance pMOSFET with BF_3 Plasma Doped Gate/Source/Drain and A/D Extension," *IEEE International Electron Device Meeting 1998*, p.639, San Francisco, 1998.
46. Goto, K., Matsuo, J., Sugii, T., Minakata, H., Yamada, I., and Hisatsugu, T., "Novel Shallow Junction Technology Using Decaborone ($B_{10}H_{14}$)," *IEEE International Electron Device Meeting 1996*, p. 435, San Francisco, 1996.
47. Ohguro, T., Nakamura, S., Saito, M., Ono, M., Harakawa, H., Morifuji, E., Yoshitomi, T., Morimoto, T., Momose, H. S., Katsumata, Y., and Iwai, H., "Ultra-shallow Junction and Salicide Technique for Advanced CMOS Devices," *Proceedings of the Sixth International Symposium on Ultralarge Scale Integration Science and Technology, Electrochemical Society*, p. 275, May 1997.
48. Osburn, C. M., Tsai, M. Y., and Zirinsky, S., "Self-Aligned Silicide Conductors in FET Integrated Circuits," *IBM Technical Disclosure Bulletin*, vol. 24, p. 1970, 1981.
49. Shibata, T., Hieda, K., Sato, M., Konaka, M., Dang, R. L. M., and Iizuka, H., "An Optimally Designed Process for Submicron MOSFETs," *IEEE International Electron Device Meeting 1981*, p. 647, Washington D.C., 1981.
50. Ting, C. Y., Iyer, S. S., Osburn, C. M., Hu, G. J., and Schweighart, A. M., "The Use of $TiSi_2$ in a Self-Aligned Silicide Technology," *Proceedings of 1st International Symposium on VLSI Science and Technology, Electrochemical Society Meeting*, vol. 82(7), p. 224, 1982.
51. Haken, R. A., "Application of the Self-Aligned Titanium Silicide Process to Very Large Scale Integrated N-Metal-Oxide-Semiconductor and Complementary Metal-Oxide-Semiconductor Technologies," *Journal of Vacuum Science and Technology*, vol. B3(6), p. 1657, 1985.
52. Kobayashi, N., Hashimoto, N., Ohyu, K., Kaga, T., and Iwata, S., "Comparison of $TiSi_2$ and WSi_2 for Sub-Micron CMOSs," *1986 Symposium on VLSI Technology, Digest of Technical Papers*, p. 49, 1986.
53. Ho, V. Q. and Poulin, D., "Formation of Self-Aligned $TiSi_2$ for VLSI Contacts and Interconnects," *Journal of Vacuum Science and Technology*, vol. A5, p. 1396, 1987.
54. Ting, C. H., d'Heurle, F. M., Iyer, S. S., and Fryer, P. M., "High Temperature Process Limitation on $TiSi_2$," *Journal of Electrochemical Society*, vol. 133(12), p. 2621, 1986.
55. Osburn, C. M., Tsai, M. Y., Roberts, S., Lucchese, C. J., and Ting, C. Y., "High Conductivity Diffusions and Gate Regions Using a Self-Aligned Silicide Technology," *Proceedings of 1st International Symposium on VLSI Science and Technology, Electrochemical Society*, vol. 82-1, p. 213, 1982.
56. Kwork, T., "Effect of Metal Line Geometry on Electromigration Lifetime in Al-Cu Submicron Interconnects," *26th Annual Proceedings of Reliability Physics 1988*, p. 185, 1988.
57. Owada, N., Hinode, K., Horiuchi, M., Nishida, T., Nakata, K., and Mukai, K., "Stress Induced Slit-Like Void Formation in a Fine-Pattern Al-Si Interconnect during Aging Test," *1985 Proceedings of the 2nd International IEEE VLSI Multilevel Interconnection Conference*, p. 173, 1985.
58. Kikkawa, T., Aoki, H., Ikawa, E., and Drynan, J. M., "A Quarter-Micrometer Interconnection Technology Using a TiN/Al-Si-Cu/Al-Si-Cu/TiN/Ti Multilayer Structure," *IEEE Trans. Electron Devices*, vol. ED-40, p. 296, 1993.
59. White, F., Hill, W., Eslinger, S., Payne, E., Cote, W., Chen, B., and Johnson, K., "Damascene Stud Local Interconnect in CMOS Technology," *IEEE International Electron Device Meeting 1992*, p. 301, San Francisco, 1992.
60. Kobayashi, N., Suzuki, M., and Saitou, M., "Tungsten Plug Technology: Substituting Tungsten for Silicon Using Tungsten Hexafluoride," *Extended Abstract of 1988 International Conference on Solid State Devices and Materials*, p. 85, 1988.

61. Cote, W., Costrini, G., Eldlstein, D., Osborn, C., Poindexter, D., Sardesai, V., and Bronner, G., "An Evaluation of Cu Wiring in a Production 64Mb DRAM," *1998 Symposium on VLSI Technology, Digest of Technical Papers*, p. 24 Honolulu, 1998.
62. Loke, A. L. S., Wetzel, J., Ryu, C., Lee, W.-J., and Wong, S. S., "Copper Drift in Low-κ Polymer Dielectrics for ULSI Metallization," *1998 Symposium on VLSI Technology, Digest of Technical Papers*, p. 26, Honolulu, 1998.
63. Wada, J., Oikawa, Y., Katata, T., Nakamura, N., and Anand, M. B., "Low Resistance Dual Damascene Process by AL Reflow Using Nb Liner," *1998 Symposium on VLSI Technology, Digest of Technical Papers*, p. 48, Honolulu, 1998.
64. Momose, H. S., Fujimoto, R., Ohtaka, S., Morifuji, E., Ohguro, T., Yoshitomi, T., Kimijima, H., Nakamura, S., Morimoto, T., Katsumata, Y., Tanimoto, H., and Iwai, H., "RF Noise in 1.5 nm Gate Oxide MOSFETs and the Evaluation of the NMOS LNA Circuit Integrated on a Chip," *1998 Symposium on VLSI Technology, Digest of Technical Papers*, p. 96, Honolulu, 1998.
65. Burghartz, J. N., "Progress in RF Inductors on Silicon — Understanding Substrate Loss," *IEEE International Electron Device Meeting 1998*, p. 523, San Francisco, 1998.
66. Yoshitomi, T., Sugawara, Y., Morifuji, E., Ohguro, T., Kimijima, H., Morimoto, T., Momose, H. S., Katsumata, Y., and Iwai, H., "On-Chip Inductors with Diffused Shield Using Channel-Stop Implant," *IEEE International Electron Device Meeting 1998*, p. 540, San Francisco, 1998.
67. Borel, J., "Technologies for Multimedia Systems on a Chip," *International Solid State Circuit Conference, Digest of Technical Papers*, p. 18, 1997.
68. Tsukamoto, M., Kuroda, H., and Okamoto, Y., "0.25mm W-polycide Dual Gate and Buried Metal on Diffusion Layer (BMD) Technology for DRAM-Embedded Logic Devices," *1997 Symposium on VLSI Technology, Digest of Technical Papers*, p. 23, 1997.
69. Itabashi, K., Tsuboi, S., Nakamura, H., Hashimoto, K., Futoh, W., Fukuda, K., Hanyu, I., Asai. S., Chijimatsu,T., Kawamura, E., Yao, T., Takagi, H., Ohta, Y., Karasawa, T., Iio, H., Onoda, M., Inoue, F., Nomura, H., Satoh, Y., Higashimoto, M., Matsumiya, M., Miyabo, T., Ikeda, T., Yamazaki, T., Miyajima, M., Watanabe, K., Kawamura, S., and Taguchi, M., "Fully Planarized Stacked Capacitor Cell with Deep and High Aspect Ratio Contact Hole for Giga-bit DRAM," *1997 Symposium on VLSI Technology, Digest of Technical Papers*, p. 21, 1997.
70. Kim, K. N., Lee, J. Y., Lee, K. H., Noh, B. H., Nam, S. W., Park, Y. S., Kim, Y. H., Kim, H. S., Kim, J. S., Park, J. K., Lee, K. P., Lee, K. Y., Moon, J. T., Choi, J. S., Park, J. W., and Lee, J. G., "Highly Manufacturable 1Gb SDRAM," *1997 Symposium on VLSI Technology, Digest of Technical Papers*, p. 10, 1997.
71. Kohyama, Y., Ozaki, T., Yoshida, S., Ishibashi, Y., Nitta, H., Inoue, S., Nakamura, K., Aoyama, T., Imai, K., and Hayasaka, N., "A Fully Printable, Self-Aligned and Planarized Stacked Capacitor DRAM Cell Technology for 1Gbit DRAM and Beyond," *1997 Symposium on VLSI Technology, Digest of Technical Papers*, p. 17, 1997.
72. Drynan, J. M., Nakajima, K., Akimoto, T. , Saito, K., Suzuki, M., Kamiyama, S., and Takaishi, Y., "Cylindrical Full Metal Capacitor Technology for High-Speed Gigabit DRAMs," *1997 Symposium on VLSI Technology, Digest of Technical Papers*, p. 151, 1997.
73. Takehiro, S., Yamauchi, S., Yoshimura, M., and Onoda, H., "The Simplest Stacked BST Capacitor for the Future DRAMs Using a Novel Low Temperature Growth Enhanced Crystallization," *1997 Symposium on VLSI Technology, Digest of Technical Papers*, p. 153, 1997.
74. Nesbit, L., Alsmeier, J., Chen, B., DeBrosse, J., Fahey, P., Gall, M., Gambino, J., Gerhard, S., Ishiuchi, H., Kleinhenz, R., Mandelman, J., Mii, T., Morikado, M., Nitayama, A., Parke, S., Wong, H., and Bronner, G., "A 0.6μm^2 256Mb Trench DRAM Cell with Self-Aligned BuriEd STrap (BEST)," *IEEE International Electron Device Meeting*, p. 627, Washington D.C., 1993.
75. Bronner, G., Aochi, H., Gall, M., Gambino, J., Gernhardt, S., Hammerl, E., Ho, H., Iba, J., Ishiuchi, H., Jaso, M., Kleinhenz, R., Mii, T., Narita, M., Nesbit, L., Neumueller, W., Nitayama, A., Ohiwa,T., Parke, S., Ryan, J., Sato, T., Takato, H., and Yoshikawa, S., "A Fully Planarized 0.25μm CMOS Technology for 256Mbit DRAM and Beyond," *1995 Symposium on VLSI Technology, Digest of Technical Papers*, p. 15, 1995.

76. Ishiuchi, H., Yoshida, Y., Takato, H., Tomioka, K., Matsuo, K. , Momose, H., Sawada, S., Yamazaki, K., and Maeguchi, K., "Embedded DRAM Technologies," *IEEE International Electron Device Meeting*, p. 33, Washington D.C., 1997.
77. Togo, M., Iwao, S., Nobusawa, H., Hamada, M., Yoshida, K., Yasuzato, N., and Tanigawa, T., "A Salicide-Bridged Trench Capacitor with a Double-Sacrificial-Si_3N_4-Sidewall (DSS) for High-Performance Logic-Embedded DRAMs," *IEEE International Electron Device Meeting*, p. 37, Washington D.C., 1997.
78. Crowder, S., Stiffler, S., Parries, P., Bronner, G., Nesbit, L., Wille, W., Powell, M., Ray, A., Chen, B., and Davari, B., "Trade-offs in the Integration of High Performance Devices with Trench Capacitor DRAM," *IEEE International Electron Device Meeting*, p. 45, Washington D.C., 1997.
79. Crowder, S., Hannon, R., Ho, H., Sinitsky, D., Wu, S., Winstel, K., Khan, B., Stiffler, S. R., and Iyer, S. S., "Integration of Trench DRAM into a High-Performance 0.18 μm Logic Technology with Copper BEOL," *IEEE International Electron Device Meeting*, p. 1017, San Francisco, 1998.
80. Yoshida, M., Kumauchi, T., Kawakita, K., Ohashi, N., Enomoto, H., Umezawa, T., Yamamoto, N., Asano, I., and Tadaki, Y. ,"Low Temperature Metal-based Cell Integration Technology for Gigabit and Embedded DRAMs," *IEEE International Electron Device Meeting*, p. 41, Washington D.C., 1997.
81. Nakamura, S., Kosugi, M., Shido, H., Kosemura, K., Satoh, A., Minakata, H., Tsunoda, H., Kobayashi, M., Kurahashi, T., Hatada, A., Suzuki, R. , Fukuda, M., Kimura, T., Nakabayashi, M., Kojima, M., Nara, Y., Fukano, T., and Sasaki, N., "Embedded DRAM Technology Compatible to the 0.13 μm High-Speed Logics by Using Ru Pillars in Cell Capacitors and Peripheral Vias," *IEEE International Electron Device Meeting*, p. 1029, San Francisco, 1998.
82. Drynan, J. M., Fukui, K., Hamada, M., Inoue, K., Ishigami, T., Kamiyama, S., Matsumoto, A., Nobusawa, H., Sugai, K., Takenaka, M., Yamaguchi, H., and Tanigawa, T., "Shared Tungsten Structures for FEOL/BEOL Compatibility in Logic-Friendly Merged DRAM," *IEEE International Electron Device Meeting*, p. 849, San Francisco, 1998.
83. Togo, M., Noda, K., and Tanigawa, T., "Multiple-Thickness Gate Oxide and Dual-Gate Technologies for High-Performance Logic-Embedded DRAMs," *IEEE International Electron Device Meeting*, p. 347, San Francisco, 1998.
84. Yoshikawa, K., "Embedded Flash Memories — Technology assessment and future —," *1999 International Syposium on VLSI Technology, System, and Applications*, p. 183, Taipei, 1999.
85. Yoshikawa, K. , "Guide-lines on Flash Memory Cell Selection," *Extended Abstract of 1998 International Conference on Solid State Devices and Materials*, p. 138, 1998.
86. Watanabe, H., Yamada, S., Tanimoto, M., Mitsui, M., Kitamura, S., Amemiya, K., Tanzawa, T., Sakagami, E., Kurata, M., Isobe, K., Takebuchi, M., Kanda, M., Mori, S.,and Watanabe, T., "Novel 0.44μm^2 Ti-Salicide STI Cell Technology for High-Density NOR Flash Memories and High Performance Embedded Application," *IEEE International Electron Device Meeting 1998*, p. 975, San Francisco, 1998.
87. Kuo, C., "Embedded Flash Memory Applications, Technology and Design," *1995 IEDM Short Course: NVRAM Technology and Application, IEEE International Electron Device Meeting*, Washington D.C., 1995.
88. Ohnakado, T., Mitsunaga, K., Nunoshita, M., Onoda, H., Sakakibara, K., Tsuji, N., Ajika, N., Hatanaka, M., and Miyoshi, H., " Novel Electron Injection Method Using Band-to-Band Tunneling Induced Hot Electron (BBHE) for Flash Memory with a p-channel Cell," *IEEE International Electron Device Meeting*, p. 279, Washington D.C., 1995.
89. Iwai, H., Sasaki, G., Unno, Y., Niitsu, Y., Norishima, M., Sugimoto, Y., and Kannzaki, K., "0.8μm Bi-CMOS Technology with High f_T Ion-Implanted Emitter Bipolar Transistor," *IEEE International Electron Device Meeting 1987*, p. 28, Washington D. C., 1987.
90. Clark, L. T. and Taylor, G. F., "High Fan-in Circuit Design," *1994 Bipolar/BiCMOS Circuits & Technology Meeting*, p. 27, Minneapolis, MN, 1994.

91. Nii, H., Yoshino, C., Inoh, K., Itoh, N., Nakajima, H., Sugaya, H., Naruse, H., Kataumata, Y., and Iwai, H., "0.3 μm BiCMOS Technology for Mixed Analog/Digital Application System," *1997 Bipolar/BiCMOS Circuits & Technology Meeting*, p. 68, Minneapolis, MN, 1997.
92. Johnson, R. A., Zierak, M. J., Outama, K. B., Bahn, T. C., Joseph, A. J.,Cordero, C. N., Malinowski, J., Bard, K. A., Weeks, T. W., Milliken, R. A., Medve, T. J., May, G. A., Chong, W., Walter, K. M., Tempest, S. L., Chau, B. B., Boenke, M., Nelson, M. W., and Harame, D. L.,"1.8 million Transistor CMOS ASIC Fabricated in a SiGe BiCMOS Technology," *IEEE International Electron Device Meeting 1998*, p. 217, San Francisco, 1998.
93. Chyan, Y.-F., Ivanov, T. G., Carroll, M. S., Nagy, W. J., Chen, A. S., and Lee, K. H., "A 50-GHz 0.25-μm High-Energy Implanted BiCMOS (HEIBiC) Technology for Low-Power High-Integration Wireless-Communication System," *1998 Symposium on VLSI Technology, Digest of Technical Papers*, p. 92, Honolulu, 1998.
94. "The National Technology Roadmap for Semiconductors," Semiconductor Industry Association, 1997.
95. Momose, H. S., Ono, M., Yoshitomi, T., Ohguro, T., Nakamura, S., Saito M., and Iwai, H., "Tunneling Gate Oxide Approach to Ultra-High Current Drive in Small-Geometry MOSFETs," *IEEE International Electron Device Meeting*, p. 593, San Francisco, 1994.
96. Momose, H. S., Ono, M., Yoshitomi, T., Ohguro, T., Nakamura, S., Saito, M., and Iwai, H., "1.5 nm Direct-Tunneling Gate Oxide Si MOSFETs," *IEEE Trans. Electron Devices*, vol. ED-43, p. 1233, 1996.
97. Timp, G., Agarwal, A. , Baumann, F. H., Boone, T., Buonanno, M., Cirelli, R., Donnelly, V., Foad, M., Grant, D., Green, M., Gossmann, H., Hillenius, S., Jackson, J., Jacobson, D., Kleiman, R., Kornblit, A., Klemens, F., Lee, J. T.-C., Mansfield, W., Moccio, S., Murrell, A., O'Malley, M., Rosamilia, J., Sapjeta, J., Silverman, P., Sorsch, T.,Tai, W. W., Tennant, D., Vuong, H., and Weir, B., "Low Leakage, Ultra-Thin Gate Oxides for Extremely High Performance sub-100 nm nMOSFETs," *IEEE International Electron Device Meeting*, p. 930, Washington D.C., 1997.
98. Momose, H. S., Ono, M., Yoshitomi, T., Ohguro, T., Nakamura, S., Saito, M., and Iwai, H., "Prospects for Low-Power, High-Speed MPUs Using 1.5 nm Direct-Tunneling Gate Oxide MOSFETs," *Journal of Solid-State Electronics*, vol. 41, p. 707, 1997.
99. Momose, H. S., Morifuji, E., Yoshitomi, T., Ohguro, T., Saito, M., Morimoto, T., Katsumata, Y., and Iwai, H., "High-Frequency AC Characteristics of 1.5 nm Gate Oxide MOSFETs," *IEEE International Electron Device Meeting*, p. 105, San Francisco, 1996.
100. Momose, H. S., Nakamura, S., Ohguro, T., Yoshitomi, T., Morifuji, E., Morimoto, T., Katsumata, Y., and Iwai, H., "Study of the Manufacturing Feasibility of 1.5 nm Direct-Tunneling Gate Oxide MOSFETs: Uniformity, Reliability, and Dopant Penetration of the Gate Oxide," *IEEE Trans. Electron Devices*, vol. ED-45, p. 691, 1998.
101. Lo, S.-H., Buchanan, D. A., Taur, Y., and Wang, W., "Quantum-Mechanical Modeling of Electron Tunneling Current from the Inversion Layer of Ultra-Thin-Oxide nMOSFETs," *IEEE Electron Devices Letters*, vol. EDL-18, p. 209, 1997.
102. Sorsch, T., Timp, W., Baumann, F. H., Bogart, K. H. A., Boone, T., Donnelly, V. M., Green, M., Evans-Lutterodt, K., Kim, C. Y., Moccio, S., Rosamilia, J., Sapjeta, J., Silverman, P., Weir B., and Timp, G., "Ultra-Thin, 1.0–3.0 nm, Gate Oxides for High Performance sub-100 nm Technology," *1998 Symposium on VLSI Technology, Digest of Technical Papers*, p. 222, 1998.
103. Ohguro, T., Naruse, N., Sugaya, H., Morifuji, E., Nakamura, S., Yoshitomi, T., Morimoto, T., Momose, H. S., Katsumata, Y., and Iwai, H., "0.18 μm Low Voltage/Low Power RF CMOS with Zero Vth Analog MOSFETs made by Undoped Epitaxial Channel Technique," *IEEE International Electron Device Meeting*, p. 837, Washington D.C., 1997.
104. Ohguro, T., Naruse, H., Sugaya, H., Kimijima, H., Morifuji, E., Yoshitomi, T., Morimoto, T., Momose, H. S., Katsumata, Y., and Iwai, H., "0.12 μm Raised Gate/Source/Drain Epitaxial Channel NMOS Technology," *IEEE International Electron Device Meeting 1998*, p. 927, San Francisco, 1998.

Grahn, J.V., Ostling, M.“Bipolar Technology”
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

3

Bipolar Technology

Jan V. Grahn
Mikael Östling
Royal Institute of Technology (KTH)

3.1 Introduction

The development of a bipolar technology for integrated circuits goes hand in hand with the steady improvement in semiconductor materials and discrete components during the 1950s and 1960s. Consequently, silicon bipolar technology formed the basis for the IC market during the 1970s. As circuit dimensions shrink, the MOSFET (or MOS) has gradually taken over as the major technological platform for silicon integrated circuits. The main reasons are the ease of miniaturization and high yield for MOS compared to bipolar technology. However, during the same period of MOS growth, much progress was simultaneously achieved in bipolar technology.[1,2] This is illustrated in Fig. 3.1 where the reported gate delay time for emitter-coupled logic (ECL) is plotted versus year. [2,3] In 1984, the 100 ps/gate limit was broken and, since then, the speed performance has been improved by a factor of ten. The high speed and large versatility of the silicon bipolar transistor still make it an attractive choice for a variety of digital and analog applications.[4]

Apart from high-speed performance, the bipolar transistor is recognized by its excellent analog properties. It features high linearity, superior low- and high-frequency noise behavior, and a very large transconductance.[5] Such properties are highly desirable for many RF applications, both for narrow-band as well as broad-band circuits.[6] The high current drive capability per unit silicon area makes the bipolar transistor suitable for input/output stages in many IC designs (e.g., in fast SRAMs). The disadvantage of bipolar technology is the low transistor density, combined with a large power dissipation. High-performance bipolar circuits are therefore normally fabricated at a modest integration level (MSI/LSI). By using BiCMOS design, the benefits of both MOS and bipolar technology are utilized.[7] One example is mixed analog/digital systems where a high-performance bipolar process is integrated with high-density CMOS. This technology forms a vital part in several system-on-a-chip designs (e.g., for telecommunication circuits).

In this chapter, a brief overview of bipolar technology is given with an emphasis on the integrated silicon bipolar transistor. The information presented here is based on the assumption that the reader is

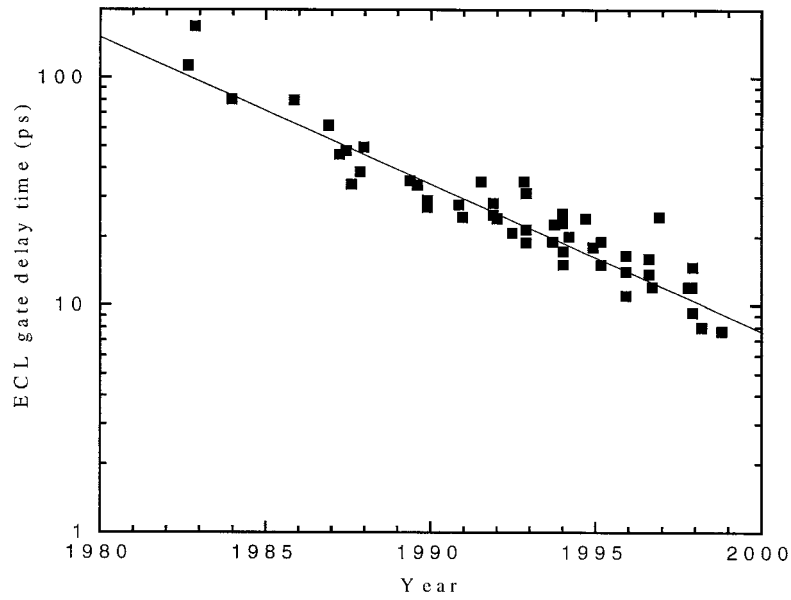


FIGURE 3.1 Reported gate delay time for bipolar ECL circuits vs. year.

familiar with bipolar device fundamentals and basic VLSI process technology. Bipolar transistors are treated in detail in the well-known textbooks by Ashburn[8] and Roulston.[9] The first part of this chapter will outline the general concepts in bipolar process design and optimization (Section 3.2). The second part will present the three generations of integrated devices representing state-of-the-art bipolar technologies for the 1970s, 1980s, and 1990s (Sections 3.3, 3.4, and 3.5, respectively). Finally, some future trends in bipolar technology are outlined.

3.2 Bipolar Process Design

The design of a bipolar process starts with the specification of the application target and its circuit technology (digital or analog). This leads to a number of requirements formulated in device parameters and associated figures-of-merit. These are mutually dependent and must therefore be traded off against each other, making the final bipolar process design a compromise between various conflicting device requirements.

Figures-of-Merit

In the digital bipolar process, the cutoff frequency (f_T) is a well-known figure-of-merit for speed. The f_T is defined for a common-emitter configuration with its output short-circuit when extrapolating the small signal current gain to unity. From a circuit perspective, a more adequate figure-of-merit is the gate delay time (τ_d) measured for a ring-oscillator circuit containing an odd number of inverters.[10] The τ_d can be expressed as a linear combination of the incoming time constants weighted by a factor determined by the circuit topology (e.g., ECL)[10,11] Alternative expressions for τ_d calculations have been proposed.[12] Besides speed, power dissipation can also be a critical issue in densely packed bipolar digital circuits, resulting in the power-delay product as a figure-of-merit.[4]

In the analog bipolar process, the dc properties of the transistor are of utmost importance. This involves minimum values on common-emitter current gain (β), Gummel plot linearity (β_{max}/β) breakdown voltage (BV_{CEO}), and Early voltage (V_A). The product $\beta \times V_A$ is often introduced as a figure-of-merit for the device

dc characteristics.[13] Rather than f_T, the maximum oscillation frequency, $f_{max} = \sqrt{f_T/(8\pi R_B C_{BC})}$ is preferred as a figure-of-merit in high-speed analog design, where R_B and C_{BC} denote the total base resistance and the base-collector capacitance, respectively.[14] Alternative figures-of-merit for speed have been proposed in the literature.[15,16] Analog bipolar circuits are often crucially dependent on a certain noise immunity, leading to the introduction of the corner frequency and noise figure as figures-of-merit for low-frequency and high-frequency noise properties, respectively.[17]

Process Optimization

The optimization of the bipolar process is divided between the intrinsic and extrinsic device design. This corresponds to the vertical impurity profile and the horizontal layout of the transistor, respectively.[10] See example in Fig. 3.2, where the device cross-section is also included. It is clear that the vertical profile and horizontal layout are primarily dictated by the given process and lithography constraints, respectively.

Figure 3.3 shows a simple flowchart of the bipolar design procedure. Starting from the specified dc parameters at a given operation point, the doping profiles can be derived. The horizontal layout must be adjusted for minimization of the parasitics. A (speed) figure-of-merit can then be calculated. An implicit relation is thus obtained between the figure-of-merit and the processing parameters.[11,18] In practice, several iterations must be performed in the optimization loop in order to find an acceptable compromise between the device parameters. This procedure is substantially alleviated by two-dimensional process simulations of the device fabrication[19] as well as device simulations of the bipolar transistor.[20,21] For optimization of a large number of device parameters, the strategy is based on screening out the unimportant factors, combined with a statistical approach (e.g., response surface methodology.)[22]

Vertical Structure

The engineering of the vertical structure involves the design of the collector, base, and emitter impurity profiles. In this respect, f_T is an adequate parameter to optimize. For a modern bipolar transistor with suppressed parasitics, the maximum f_T is usually determined by the forward transit time of minority carriers through the intrinsic component. The most important f_T tradeoff is against BV_{CEO}, as stated by the Johnson limit for silicon transistors:[23] the product $f_T \times BV_{CEO}$ cannot exceed 200 GHz-V (recently updated to 500 GHz-V).[24]

Collector Region

The vertical n-type collector of the bipolar device in Fig. 3.2 consists of two regions below the p-type base diffusion: a lowly or moderately doped n-type epitaxial (epi) layer, followed by a highly doped n^+-subcollector. The thickness and doping level of the subcollector are non-critical parameters; a high arsenic or antimony doping density between 10^{19} and 10^{20} cm^{-3} is representative, resulting in a sheet resistance of 20 to 40 Ω/sq. In contrast, the design of the epi-layer constitutes a fundamental topic in bipolar process optimization.

To first order, the collector doping in the epi-layer is determined by the operation point (more specifically, the collector current density) of the component (see Fig. 3.3). A normal condition is to have the operation point corresponding to maximum f_T, which typically means a collector current density on the order of $2–4 \times 10^4$ A/cm^2. As will be recognized later, bipolar scaling results in increased collector current densities. Above a certain current, there will be a rapid roll-off in current gain as well as cutoff frequency. This is due to high-current effects, primarily the base push-out or Kirk effect, leading to a steep increase in the forward transit time.[25] Since the critical current value is proportional to the collector doping,[26] a minimum impurity concentration for the epi-layer is required, thus avoiding f_T degradation (typically around 10^{17} cm^{-3} for a high-speed device). Usually, the epi-layer is doped only in the intrinsic structure by a selectively implanted collector (SIC) procedure.[27] An example of such a doping profile is seen in Fig. 3.4. Such a collector design permits an improved control over the base-collector junction; that is, shorter base width as well as suppressed Kirk effect. The high collector doping concentration,

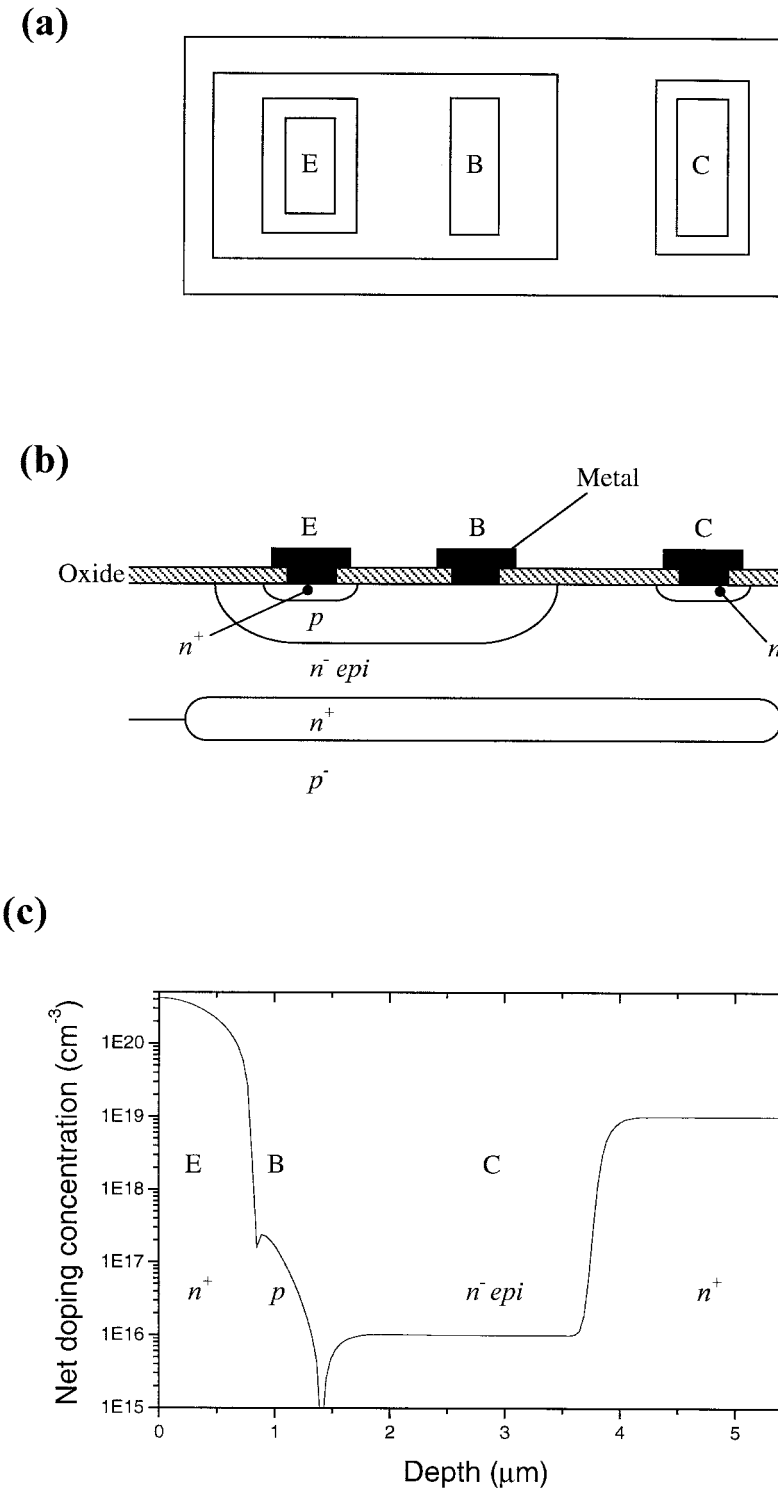


FIGURE 3.2 (a) Layout, (b) cross-section, and (c) simulated impurity profile through emitter window for an integrated bipolar transistor (E = emitter, B = base, C = collector).

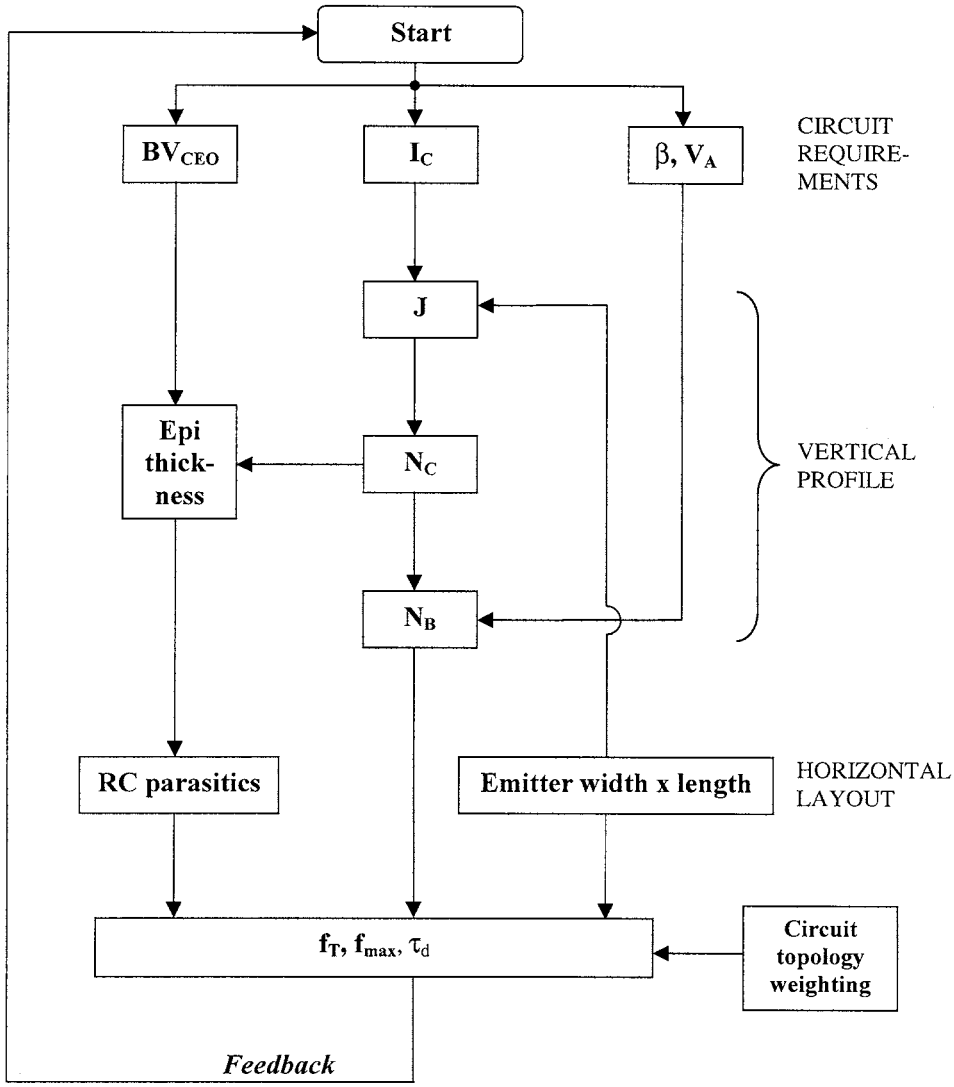


FIGURE 3.3 Generic bipolar device optimization flowchart.

however, may be a concern for both C_{BC} and BV_{CEO}. The latter value will therefore often set a higher limit on the collector doping value. One way to reduce the electrical field in the junction is to implement a lightly doped collector spacer layer between the heavily doped base and collector regions.[28,29] A retrograde collector profile with a low impurity concentration near the base-collector junction and then increasing toward the subcollector has also been reported to enhance f_T.[30,31]

The thickness of the epi-layer exhibits large variations between different device designs, extending several micrometers in depth for analog bipolar components, whereas a high-speed digital design typically has an epi-layer thickness around 1 μm or below, thus reducing total collector resistance. As a result, the transistor breakdown voltage is sometimes determined by reach-through breakdown (i.e., full depletion penetration of the epi-collector). The thickness of the collector layer can therefore be used as a parameter in determining BV_{CEO}, which in turn is traded off vs. f_T.[32]

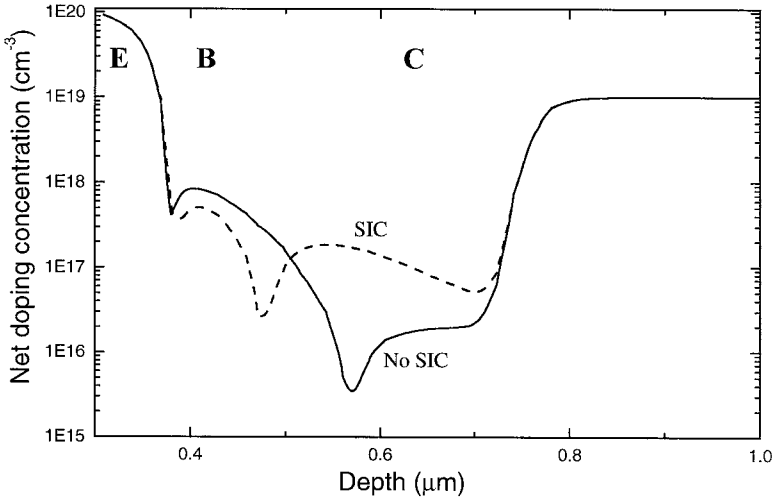


FIGURE 3.4 Simulated vertical impurity profile with and without SIC.

In cases where f_{max} is of interest, the collector design must be carefully taken into account. Compared to f_T, the optimum f_{max} is found for thicker and lower doped collector epi-layers.[33,34] The vertical collector design will therefore, to a large extent, determine the tradeoff between f_T and f_{max}.

Base Region

The width and peak concentration of the base profile are two of the most fundamental parameters in vertical profile design. The base width W_B is normally in the range 0.1 to 1 μm, whereas a typical base peak concentration lies between 10^{17} and 10^{18} cm^{-3}. The current gain of the transistor is determined by the ratio of the Gummel number in the emitter and base. Usually, a current gain of at least 100 is required for analog bipolar transistors, whereas in digital applications, β around 20 is often acceptable. A normal base sheet resistance (or pinch resistance) for conventional bipolar processes is of the order of 100 Ω/sq., whereas the number for high-speed devices typically is in the interval 1 to 10 kΩ/sq. This is due to the small $W_B < 0.1$ μm necessary for a short base transit time ($\propto W_B^2$). On the other hand, a too narrow base will have a negative impact on f_{max} because of its R_B dependence. As a result, f_{max} exhibits a maximum when plotted against W_B.[35]

The base impurity concentration must be kept high enough to avoid punch-through at low collector voltages; that is, the base-collector depletion layer penetrates across the neutral base. In other words, the base doping level is also dictated by the collector impurity concentration. Punch-through is the ultimate consequence of base width modulation or the Early effect manifested by a finite output resistance in the I_C-V_{CE} transistor characteristic.[36] The associated V_A or the product $\beta \times V_A$ serves as an indicator of the linear properties for the bipolar transistor. The V_A is typically at a relatively high level (>30 V) for analog applications, whereas digital designs often accept relatively low $V_A < 15$ V.

Figure 3.5 demonstrates simulations of current gain versus the base doping for various W_B.[37] It is clearly seen that the base doping interval permitting a high current gain while avoiding punch-through will be pushed to high impurity concentrations for narrow base widths. In addition, Fig. 3.5 points to another limiting factor for high base doping numbers above 5×10^{18} cm^{-3}, namely, the onset of forward-biased tunneling currents in the emitter-base junction[38] leading to non-ideal base current characteristics.[39] It is concluded that the allowable base doping interval will be very narrow for $W_B < 0.1$ μm.

The shape of the base profile has some influence over the device performance. The final base profile is the result of an implantation and diffusion process and, normally, only the peak base concentration is given along with the base width. Nonetheless, there will be an impurity grading along the base profile (see Figs. 3.2 and 3.4), creating a built-in electrical field and thereby adding a drift component for the

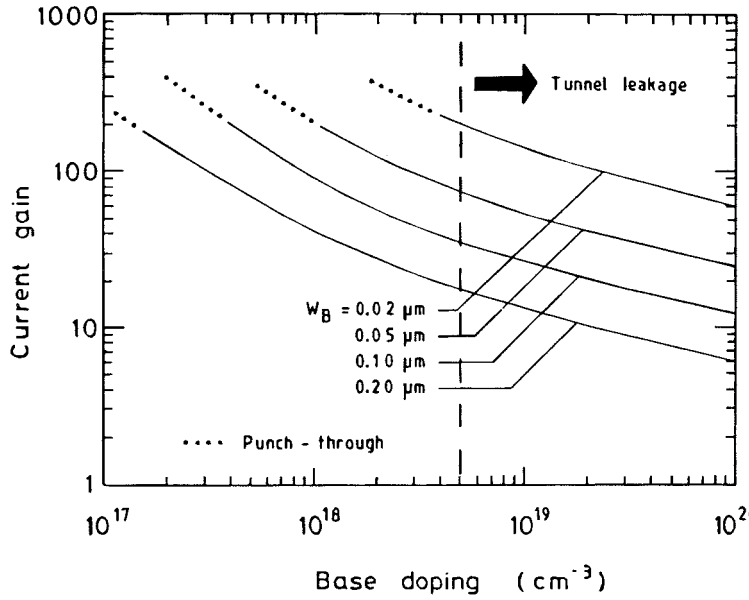


FIGURE 3.5 Simulated current gain vs. base doping density for different base widths ($N_C = 6 \times 10^{16}$ cm^{-3}, $N_E = 10^{20}$ cm^{-3}, and emitter depth 0.20 μm) (after Ref. 37, copyright© 1990, IEEE).

minority carrier transport.[40] Recent research has shown that for very narrow base transistors, the uniform doping profile is preferable when maximizing f_T.[41,42] This is also valid under high injection conditions in the base.[43] Uniformly doped base profiles are common in advanced bipolar processes using epitaxial techniques for growing the intrinsic base.

During recent years, base profile design has largely been devoted to implementation of narrow bandgap SiGe in the base. The resulting emitter-base heterojunction allows a considerable enhancement in current gain, which can be traded off against increased base doping, thus substantially alleviating the problem with elevated base sheet resistances typical of high-speed devices.[44] Excellent dc as well as high-frequency properties can be achieved. The position of the Ge profile with respect to the boron profile has been discussed extensively in the literature.[45–47] More details about SiGe heterojunction engineering are found in Chapter 5 by Cressler.

Emitter Region

The conventional metal-contacted emitter is characterized by an abrupt arsenic or phosphorus profile fabricated by direct diffusion or implantation into the base area (see Fig. 3.2).[48] In keeping emitter efficiency close to unity (and thus high current gain), the emitter junction cannot be made too shallow (~1 μm). The emitter doping level lies typically between 10^{20} and 10^{21} cm^{-3} close to the solid solubility limit at the silicon surface, hence providing a low emitter resistance as well as a large emitter Gummel number required for keeping current gain high. Bandgap narrowing, however, will be present in the emitter, causing a reduction in the efficient emitter doping.[49]

When scaling bipolar devices, the emitter junction must be made more shallow to ensure a low emitter-base capacitance. When the emitter depth becomes less than the minority carrier recombination length, the current gain will inevitably degrade. This precludes the use of conventional emitters in a high-performance bipolar technology. Instead, polycrystalline (poly) silicon emitter technology is utilized. By diffusing impurity species from the polysilicon contact into the monocrystalline (mono) silicon, a very shallow junction (< 0.2 μm) is formed; yet gain can be kept at a high level and even traded off against a higher base doping.[50] A gain enhancement factor between 3 and 30 for the polysilicon compared to the monosilicon emitter has been reported (see also Section 3.4).[51,52]

Scaling Rules

The principles for vertical design can be summarized in the bipolar scaling rules formulated by Solomon and Tang; [53,54] see Table 3.1. Since the bipolar transistor is scaled under constant voltage, the current density increases with reduced device dimensions. At medium or high current densities, the vertical structure determines the speed. At low current densities, performance is normally limited by device parasitics. Eventually, tunnel currents or contact resistances constitute a final limit to further speed improvement based on the scaling rules. A solution is to use SiGe bandgap engineering to further enhance device performance without scaling.

TABLE 3.1 Bipolar Scaling Rules (Scaling factor $\lambda < 1$)

Parameter	Scaling Factor
Voltage	1
Base width W_B	$\lambda^{0.8}$
Base doping N_B	W_B^{-2}
Current density J	λ^{-2}
Collecting doping	J
Depletion capacitances	λ
Delay	λ
Power	1
Power-delay product	λ

Horizontal Layout

The horizontal layout is carried out in order to minimize the device parasitics. Figure 3.6 shows the essential parasitic resistances and capacitances for a schematic bipolar structure containing two base contacts. The various RC constants in Fig. 3.6 introduce time delays. For conventional bipolar transistors, such parasitics often limit device speed. In contrast, the self-alignment technology applied in advanced bipolar transistor fabrication allows for efficient suppression of the parasitics.

In horizontal layout, f_{max} serves as a first-order indicator in the extrinsic optimization procedure because of its dependence on C_{BC} and (total) R_B. These two parasitics are strongly connected to the geometrical layout of the device. The more advanced τ_d calculation takes all major parasitics into account under given load conditions, thus providing good insight into the various time delay contributions of a bipolar logic gate.[55]

From Fig. 3.6, it is seen that the collector resistance is divided into three parts. Apart from the epi-layer and buried layer previously discussed, the collector contact also adds a series resistance. Provided the epi-layer is not too thick, the transistor is equipped with a deep phosphorus plug from the collector contact down to the buried layer, thus reducing the total R_C.

As illustrated in Fig. 3.6, the base resistance is divided into intrinsic (R_{Bi}) and extrinsic (R_{Bx}) components. The former is the pinched base resistance situated directly under the emitter diffusion, whereas the latter constitutes the base regions contacting the intrinsic base. The intrinsic part decreases with the current due to the lateral voltage drop in the base region.[56] At high current densities, this causes current crowding effects at the emitter diffusion edges. This results in a reduced onset for high-current effects in the transistor. The extrinsic base resistance is bias independent and must be kept as small as possible (e.g., by utilizing self-alignment architectures). By designing a device layout with two or more base contacts surrounding the emitter, the final R_B is further reduced at the expense of chip area. Apart from enhancing f_{max}, the R_B reduction is also beneficial for device noise performance.

The layout of the emitter is crucial since the effective emitter area defines the intrinsic device cross-section.[57] The minimum emitter area, within the lithography constraints, is determined by the operational collector current and the critical current density where high-current effects start to occur.[58] Eventually, a tradeoff must be made between the base resistance and device capacitances as a function of emitter geometry; this choice is largely dictated by the device application. Long, narrow emitter stripes, meaning

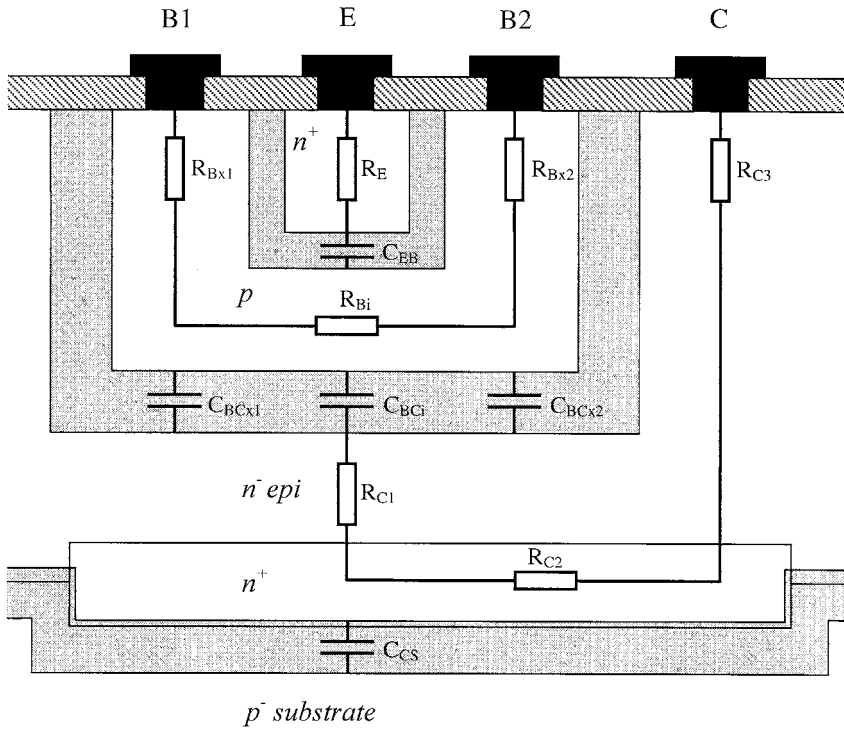


FIGURE 3.6 Schematic view of the parasitic elements in a bipolar transistor equipped with two base contacts. R_E = emitter resistance, R_{Bi} = intrinsic base resistance, R_{Bx} = extrinsic base resistance, R_C = collector resistance, C_{EB} = emitter-base capacitance, C_{BCi} = intrinsic base-collector capacitance, C_{BCx} = extrinisic base-collector capacitance, and C_{CS} = collector-substrate capacitance. Gray areas denote depletion regions. Contact resistances are not shown.

a reduction in the base resistance, are frequently used. The emitter resistance is usually non-critical for conventional devices; however, for polysilicon emitters, the emitter resistance may become a concern in very small-geometry layouts.[3]

Of the various junction capacitances in Fig. 3.6, the collector-base capacitance is the most significant. This parasitic is also divided into intrinsic (C_{BCi}) and extrinsic (C_{BCx}) contributions. Similar to R_{Bx}, the C_{BCx} is kept low by using self-aligned schemes. For example, the fabrication of a SIC causes an increase only in C_{BCi}, whereas C_{BCx} stays virtually unaffected. The collector-substrate capacitance C_{CS} is one of the minor contributors to f_T; the C_{CS} originates from the depletion regions created in the epi-layer and under the buried layer. C_{CS} will become significant at very high frequencies due to substrate coupling effects.[59]

3.3 Conventional Bipolar Technology

Conventional bipolar technology is based on the device designs developed during the 1960s and 1970s. Despite its age, the basic concept still constitutes a workhorse in many commercial analog processes where ultimate speed and high packing density are not of primary importance. In addition, a conventional bipolar component is often implemented in low-cost BiCMOS processes.

Junction-Isolated Transistors

The early planar transistor technology took advantage of a reverse-biased pn junction in providing the necessary isolation between components. One of the earliest junction-isolated transistors, the so-called triple-diffused process, is simply based on three ion implantations and subsequent diffusion.[60] This device

has been integrated into a standard CMOS process using one extra masking step.[61] The triple-diffused bipolar process, however, suffers from a large collector resistance due to the absence of a subcollector, and the npn performance will be low.

By far, the most common junction-isolated transistor is represented by the device cross-section of Fig. 3.7, the so-called buried-collector process.[60] This device is based on the concept previously shown in Fig. 3.2 but with the addition of an n^+-collector plug and isolation. This is provided by the diffused p^+-regions surrounding the transistor. The diffusion of the base and emitter impurities into the epi-layer allows relatively good control of the base width (more details of the fabrication is given in the next section on oxide-isolated transistors).

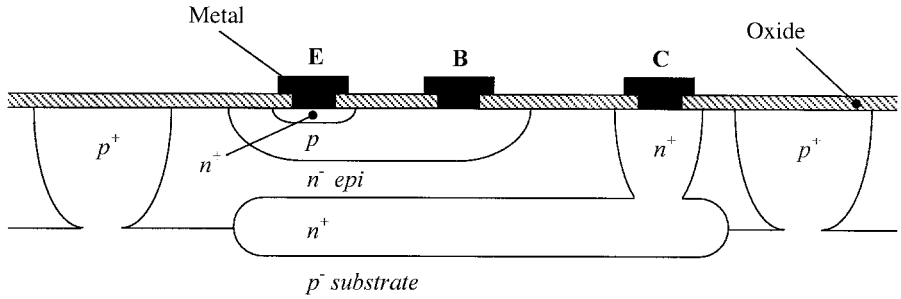


FIGURE 3.7 Cross-section of the buried-collector transistor with junction isolation and collector plug.

A somewhat different approach of the buried-collector process is the so-called collector-isolation diffusion.[62] This process requires a p-type epi-layer after formation of the subcollector. An n^+-diffusion serves both as isolation and the collector plug to the buried layer. After an emitter diffusion, the p-type epi-layer will constitute the base of the final device. Compared to the buried-layer collector process, the collector-isolated device concept does not result in very accurate control over the final base width.

The main disadvantage of the junction-isolated transistor is the relatively large chip area occupied by the isolation region, thus precluding the use of such a device in any VLSI application. Furthermore, high-speed operation is ruled out because of the large parasitic capacitances associated with the junction isolation and the relatively deep diffusions involved. Indeed, many of the conventional junction-isolated processes were designed for doping from the gas phase at high temperatures.

Oxide-Isolated Transistors

Oxide isolation permits a considerable reduction in the lateral and vertical dimensions of the buried-layer collector process. The reason is that the base and collector contacts can be extended to the edge of the isolation region. More chip area can be saved by having the emitter walled against the oxide edge. The principal difference between scaling of junction-and oxide-isolated transistors is visualized in Fig. 3.8.[63] The device layouts are Schottky clamped; that is, the base contact extends over the collector region. This hinders the transistor to enter saturation mode under device operation. In Fig. 3.8(b), the effective surface area of the emitter contact has been reduced by a so-called washed emitter approach: since the oxide formed on the emitter window during emitter diffusion has a much higher doping concentration than its surroundings, this particular oxide can be removed by a mask-less wet etching. Hence, the emitter contact becomes self-aligned to the emitter diffusion area.

The process flow including mask layouts for an oxide-isolated bipolar transistor of the buried-layer collector type is shown in Fig. 3.9.[64] After formation of the subcollector by arsenic implantation through an oxide mask in the p^--substrate, the upper collector layer is grown epitaxially on top (Fig. 3.9(a)). The device isolation is fabricated by local oxidation of silicon (LOCOS) or recessed oxide (ROX) process

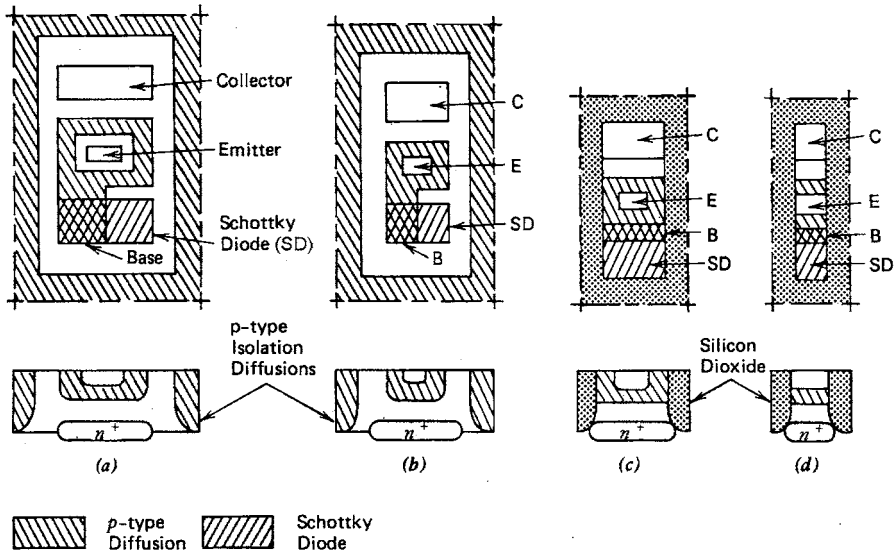


FIGURE 3.8 Device layout and cross-section demonstrating scaling of (a)-(b) junction-isolated and (c)-(d) oxide-isolated bipolar transistors (after Ref. 63, copyright© 1986, Wiley).

(Figs. 3.9(b) to (d)). The isolation mask in Fig. 3.9(b) is aligned to the buried layer using the step in the silicon (Fig. 3.9(a)) originating from the enhanced oxidation rate for highly doped n^+-silicon compared to the p^--substrate during activation of the buried layer. The ROX is thermally grown (Fig. 3.9(d)) after the boron field implantation (or chan-stop) (Fig. 3.9(c)). This p^+-implant is necessary for suppressing a conducting channel otherwise present under the ROX. The base is then formed by ion implantation of boron or BF_2 through a screen oxide (Fig. 3.9(d)); in the simple device of Fig. 3.9, a single base implantation is used; in a more advanced bipolar process, the fabrication of the intrinsic and extrinsic base must be divided into one low dose and one high dose implantation, respectively, adding one more mask to the total flow. After base formation, an emitter/base contact mask is patterned in a thermally grown oxide (Fig. 3.9(e)). The emitter is then implanted using a heavy dose arsenic implant (Fig. 3.9(f)). An n^+ contact is simultaneously formed in the collector window. After annealing, the device is ready for metallization and passivation.

Apart from the strong reduction in isolation capacitances, the replacement of a junction-isolated process with an oxide-isolated process also adds other high-speed features such as thinner epitaxial layer and shallower emitter/base diffusions. A typical base width is a few 1000 Å and the resulting f_T typically lies in the range of 1 to 10 GHz. The doping of the epitaxial layer is determined by the required breakdown voltage. Further speed enhancement of the oxide-isolated transistor is difficult due to the parasitic capacitances and resistances originating from contact areas and design-rule tolerances related to alignment accuracy.

Lateral pnp Transistors

The conventional npn flow permits the bipolar designer to simultaneously create a lateral pnp transistor. This is made by placing two base diffusions in close proximity to each other in the epi-layer, one of them (pnp-collector) surrounding the other (pnp-emitter) (see Fig. 3.10). In general, the lateral pnp device exhibits poor performance since the base width is determined by lithography constraints rather than vertical base control as in the npn device. In addition, there will be electron injection from the subcollector into the p-type emitter, thus reducing emitter efficiency.

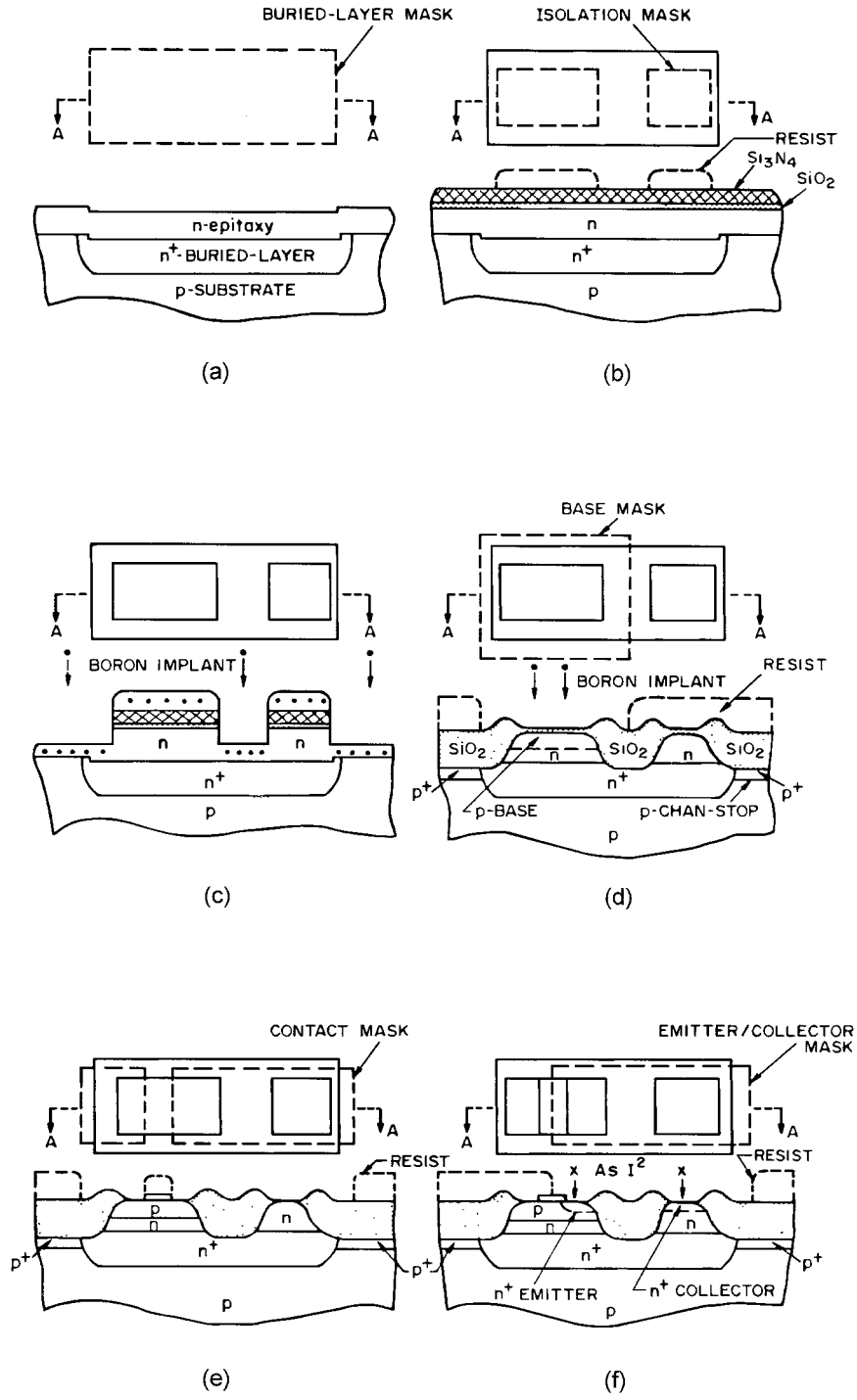


FIGURE 3.9 Layout and cross-section of the fabrication sequence for an oxide-isolated buried-collector transistor (after Ref. 64, copyright© 1983, McGraw-Hill).

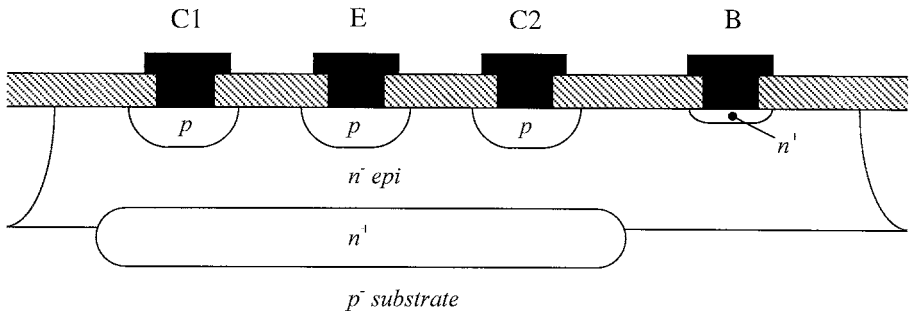


FIGURE 3.10 Schematic cross-section of the lateral pnp transistor.

3.4 High-Performance Bipolar Technology

The development of a high-performance bipolar technology for integrated circuits signified a large step forward, both with respect to speed and packing density of bipolar transistors. A representative device cross-section of a so-called double-poly transistor is depicted in Fig. 3.11. The important characteristics for this bipolar technology are the polysilicon emitter contact, the advanced device isolation, and the self-aligned structure. These three features are discussed here with an emphasis on self-alignment where the two basic process flows are outlined — the single-poly and double-poly transistor.

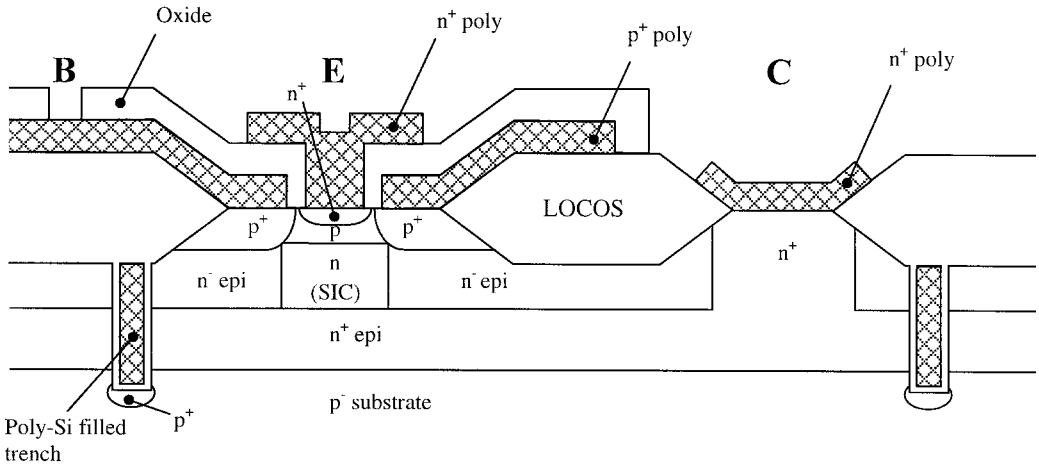


FIGURE 3.11 A double-poly self-aligned bipolar transistor with deep-trench isolation, polysilicon emitter and SIC. Metallization is not shown.

Polysilicon Emitter Contact

The polysilicon emitter contact is fabricated by a shallow diffusion of n-type species (usually arsenic) from an implanted n^+- polysilicon layer into the silicon substrate[65] (see emitter region in Fig. 3.11). The thin oxide sandwiched between the poly- and monosilicon is partially or fully broken up during contact formation. The mechanism behind the improved current gain is strongly related to the details of the interface between the polysilicon layer and the monosilicon substrate.[52] Hence, the cleaning procedure of the emitter window surface before polysilicon deposition must be carefully engineered for process robustness. Otherwise, the average current gain from wafer-to-wafer will exhibit unacceptable variations.

The emitter window preparation and subsequent drive-in anneal conditions can also be used in tailoring the process with respect to gain and emitter resistance.

From a fabrication point of view, there are further advantages when introducing polysilicon emitter technology. By implanting into the polysilicon rather than into single-crystalline material, the total defect generation as well as related anomalous diffusion effects are strongly suppressed in the internal transistor after the drive-in anneal. Moreover, the risk for spiking of aluminum during the metallization process, causing short-circuiting of the pn junction, is strongly reduced compared to the conventional contact formation. As a result, some of the yield problems associated with monosilicon emitter fabrication are, to a large extent, avoided when utilizing polysilicon emitter technology.

Advanced Device Isolation

With advanced device isolation, one usually refers to the deep trenches combined with LOCOS or ROX as seen in Fig. 3.11.[66] The starting material before etching is then a double-epitaxial layer (n^+-n) grown on a lowly doped p^--substrate. The deep trench must reach all the way through the double epi-layer, meaning a high-aspect ratio reactive-ion etch. Hence, the trenches will define the extension of the buried layer collector for the transistor.

The main reason for introducing advanced isolation in bipolar technology is the need for a compact chip layout.[67] Quite naturally, the bipolar isolation technology has benefited from the trench capacitor development in the MOS memory area. The deep trench isolation allows bipolar transistors to be designed at the packing density of VLSI.

The fabrication of a deep-trench isolation includes deep-silicon etching, chan-stop p^+-implantation, an oxide/nitride stack serving as isolation, intrinsic polysilicon fill-up, planarization, and cap oxidation.[66] The deep-trench isolation is combined with an ordinary LOCOS or ROX isolation, which is added before or after trench formation. The most advanced isolation schemes take advantage of shallow-trench isolation rather than ordinary LOCOS after the deep-trench process; in this way, a very planar surface with no oxide lateral encroachment ("birds beak") is achieved after the planarization step. The concern regarding stress-induced crystal defects originating from trench etching requires careful attention so as not to seriously affect yield.

Self-Aligned Structures

Advanced bipolar transistors are based on self-aligned structures made possible by polysilicon emitter technology. As a result, the emitter-base alignment is not dependent on the overlay accuracy of the lithography tool. The device contacts can be separated without affecting the active device area. It is also possible to create structures where the base is self-aligned both to the collector and emitter, the so-called sidewall-based contact structure (SICOS).[68] This process, however, has not been able to compete successfully with the self-aligned schemes discussed below.

The self-aligned structures are divided into single-polysilicon (single-poly) and double-polysilicon (double-poly) architectures, as visualized in Fig. 3.12.[69] The double-poly structure refers to the emitter polysilicon and base polysilicon electrode, whereas the single-poly only refers to the emitter polysilicon. From Fig. 3.12, it is seen that the double-poly approach benefits from a smaller active area than the single-poly, manifested in a reduced base-collector capacitance. Moreover, the double-poly transistor in general exhibits a lower base resistance. The double-poly transistor, however, is more complex to fabricate than the single-poly device. On the other hand, by applying inside spacer technology for the double-poly emitter structure, the lithography requirements are not as strict as in the single-poly case where more conventional MOS design rules are used for definition of the emitter electrode.

Single-Poly Structure

The fabrication of a single-poly transistor has been presented in several versions, more or less similar to the traditional MOS flow. An example of a standard single-poly process is shown in Fig. 3.13.[70] After

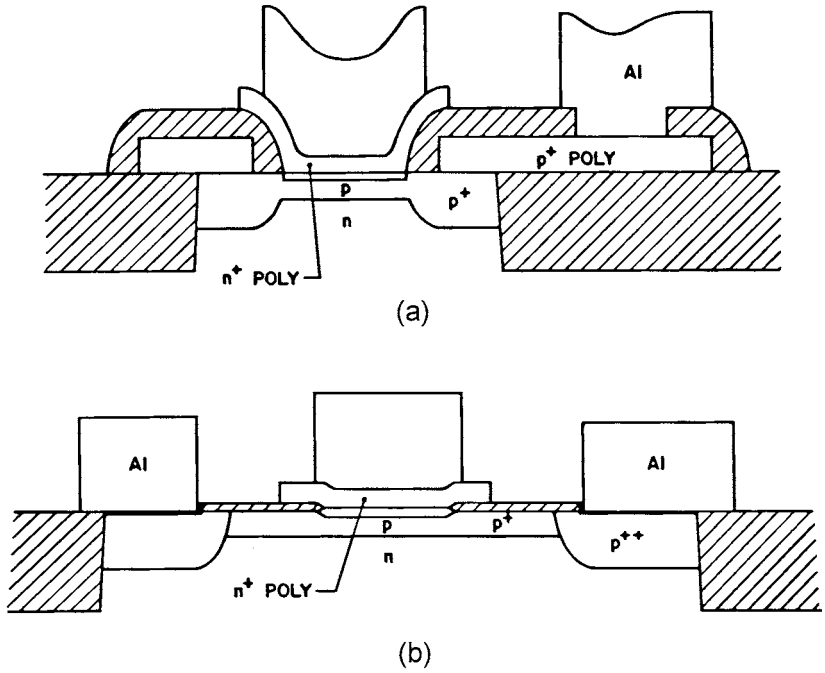


FIGURE 3.12 (a) Double-poly structure and (b) single-poly structure. Buried layer and collector contact are not shown (after Ref. 69, copyright© 1989, IEEE).

arsenic emitter implantation (Fig. 3.13(a)) and polysilicon patterning, a so-called base-link is implanted using boron ions (Fig. 3.13(b)). Oxide is then deposited and anisotropically etched to form outside spacers, followed by the heavy extrinsic base implantation (Fig. 3.13(c)). Shallow junctions (including emitter diffusion) are formed by rapid thermal annealing (RTA). A salicide or polycide metallization completes the structure (Fig. 3.13(d)).

The intrinsic base does not necessarily need to be formed prior to the extrinsic part. Li et al.[71] have presented a reverse extrinsic-intrinsic base scheme based on a disposable emitter pedestal with spacers. This leads to improved control over the intrinsic base width and a lower surface topography compared to the process represented in Fig. 3.13.

Another variation of the single-poly architecture is the so-called quasi-self-aligned process (see Fig. 3.14).[72] A base oxide is formed by thermal oxidation in the active area and an emitter window is opened (Fig. 13.14(a)). Following intrinsic base implantation, the emitter polysilicon is deposited, implanted, and annealed. The polysilicon emitter pedestal is then etched out (Fig. 3.14(b)). The extrinsic base process, junction formation, and metallization are essentially the same as in the single-poly process shown in Fig. 3.13. Note that in Fig. 13.4, the emitter-base formation is self-aligned to the emitter window in the oxide, not to the emitter itself, hence explaining the term quasi-self-aligned. As a result, a higher total base resistance is obtained compared to the standard single-poly process.

The boron implantation illustrated in Fig. 3.13(b) is an example of so-called base-link engineering aimed at securing the electrical contact between the heavily doped p^+-extrinsic base and the much lower doped intrinsic base. Too weak a base link will result in high total base resistance, whereas too strong a base link may create a lateral emitter-base tunnel junction leading to non-ideal base current characteristics.[73] Furthermore, a poorly designed base link jeopardizes matching between individual transistors since the final current gain may vary substantially with the emitter width.

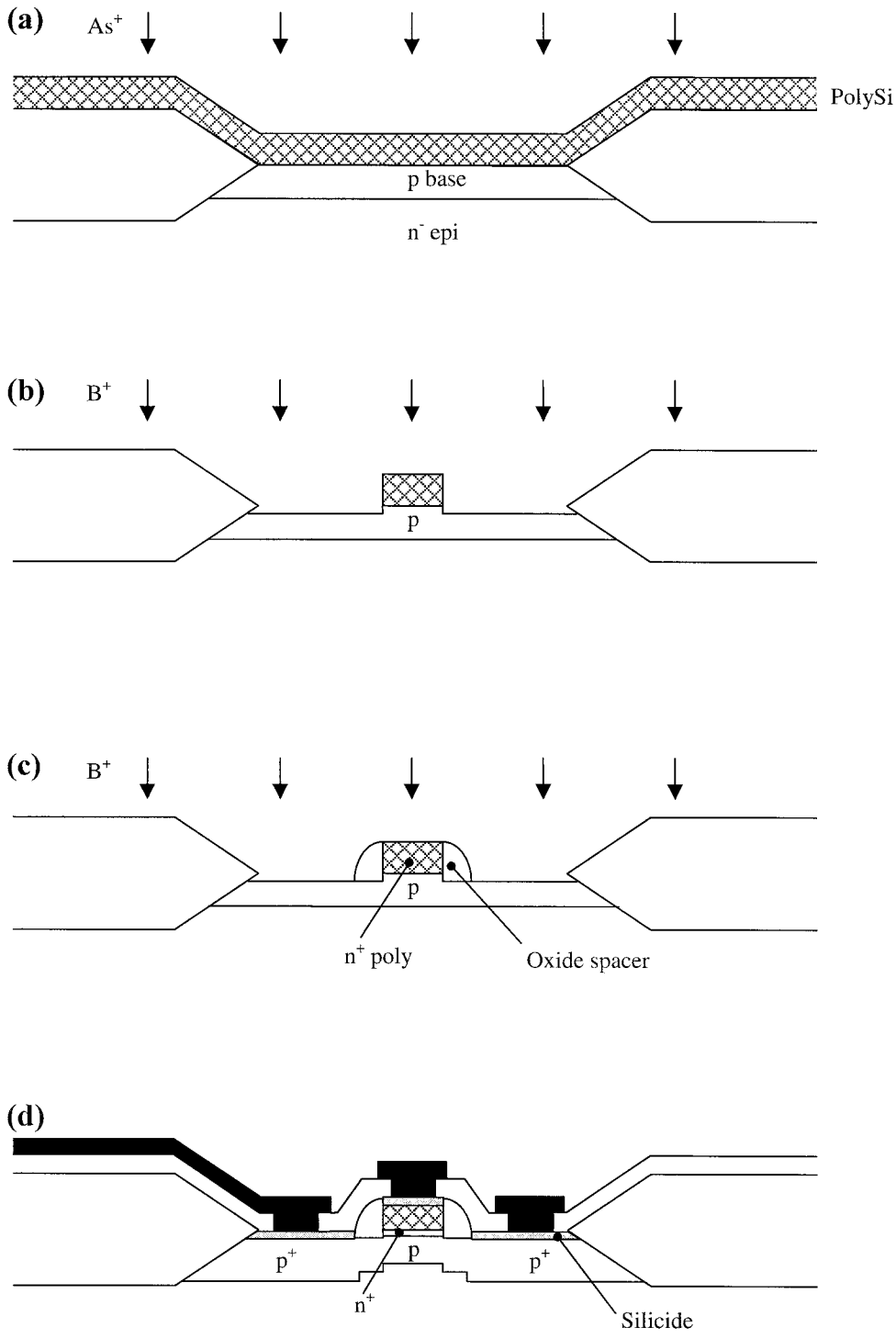


FIGURE 3.13 The single-poly, self-aligned process: (a) polyemitter implantation, (b) emitter etch and base link implantation, (c) oxide spacer formation and extrinsic base implantation, and (d) final device after junction formation and metallization.

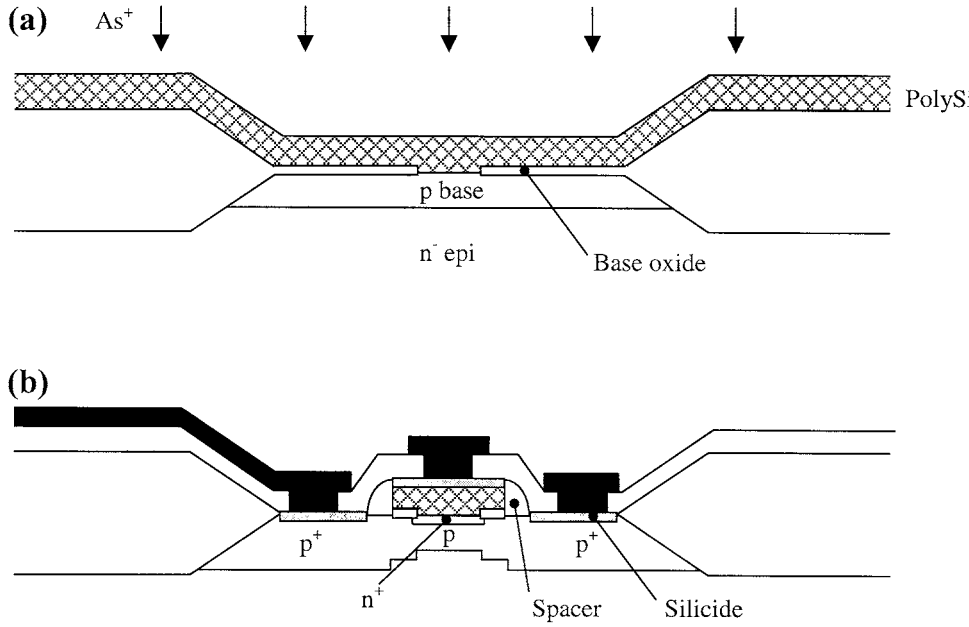


FIGURE 3.14 The single-poly, quasi-self-aligned process: (a) polyemitter implantation, (b) final device.

Double-Poly Structure

The double-poly structure originates from the classical IBM structure presented in 1981.[74] Most high-performance commercial processes today are based on double-poly technology. The number of variations are less than for the single-poly, mainly with different aspects on base-link engineering, spacer technology, and SIC formation. One example of a double-poly fabrication is presented in Fig. 3.15. After deposition of the base polysilicon and oxide stack, the emitter window is opened (Fig. 3.15(a)) and thermally oxidized. During this step, p^+-impurities from the base polysilicon diffuse into the monosilicon, thus forming the extrinsic base. In addition, the oxidation repairs the crystal damage caused by the dry etch when opening the emitter window. A thin silicon nitride layer is then deposited, the intrinsic base is implanted using boron, followed by the fabrication of amorphous silicon spacers inside the emitter window (Fig. 3.15(b)). The nitride is exposed to a short dry etch, the spacers are removed, and the thin oxide is opened up by an HF dip. Deposition and implantation of the polysilicon emitter film is carried out (Fig. 3.15(c)). The structure is patterned and completed by RTA emitter drive-in and metallization (Fig. 3.15(d)). The emitter will thus be fully self-aligned with respect to the base. Note that the inside spacer technology implies that the actual emitter width will be significantly less than the drawn emitter width.

The definition of the polyemitter in the single- and double-poly process inevitably leads to some overetching into the epi-layer, see Figs. 3.13(b) and 3.15(a), respectively. The final recessed region will make control over base-link formation more awkward.[75,76] In fact, the base link will depend both on the degree of overetch as well as the implantation parameters.[77] This situation is of no concern for the quasi-self-aligned process where the etch of the polysilicon emitter stops on the base oxide.

In a modification of the double-poly process, a more advanced base-link technology is proposed.[78] After extrinsic base drive-in and emitter window opening, BF_2-implanted poly-spacers are formed inside the emitter window. The boron is out-diffused through the emitter oxide, thus forming the base link. The intrinsic base is subsequently formed by conventional implantation through the emitter window. New dielectric inside spacers are formed prior to polysilicon emitter deposition, followed by arsenic implantation and emitter drive-in.

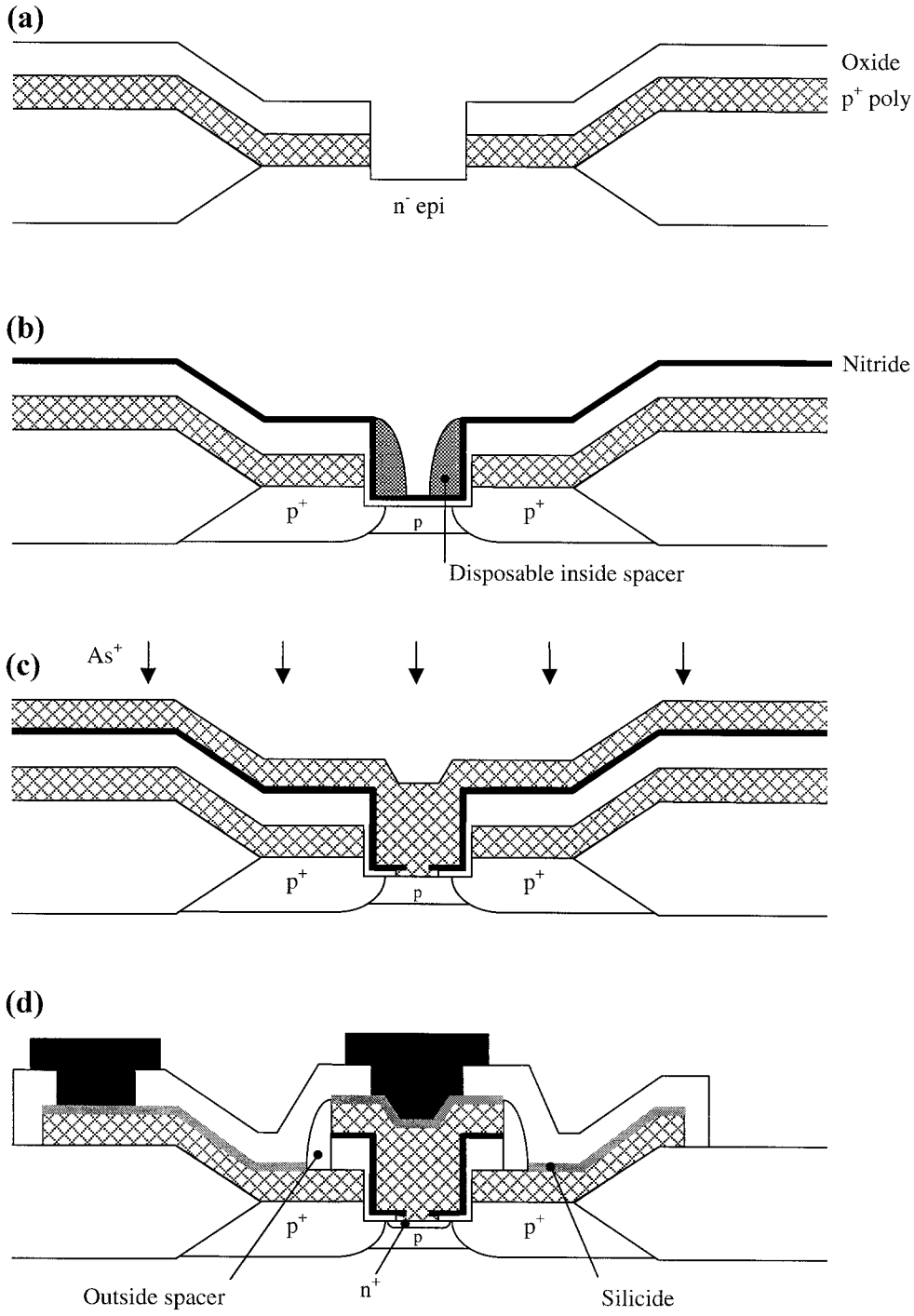


FIGURE 3.15 The double-poly, self-aligned process: (a) emitter window etch, (b) intrinsic base implantation through thin oxide/nitride stack followed by inside spacer formation, (c) polyemitter implantation, (d) final device after emitter drive-in and metallization.

Also, vertical pnp bipolar transistors based on the double-poly concept have been demonstrated.[79] Either boron or BF_2 is used for the polyemitter implantation. A pnp device with f_T of 35 GHz has been presented in a classical double-poly structure.[80]

3.5 Advanced Bipolar Technology

This chapter section treats state-of-the-art bipolar technologies reported during the 1990s (but not necessarily put into production). Alongside the traditional down-scaling in design rules, efforts have focused on new innovations in emitter and base electrode fabrication. A key issue has been the integration of epitaxial Si or SiGe intrinsic base into the standard npn process flow. This section concludes with an outlook on the future trends in bipolar technology after the year 2000.

Implanted Base

Today's most advanced commercial processes are specified with an f_T around 30 GHz. The major developments are being carried out using double-poly technology, although new improvements have also been reported for single-poly architectures.[72,81] For double-poly transistors, it was demonstrated relatively early that by optimizing a very low intrinsic base implant energy below 10 keV, devices with an f_T around 50 GHz are possible to fabricate.[82] The emitter out-diffusion is performed by a combined furnace anneal and RTA. In this way, the intrinsic base width is controlled below 1000 Å, whereas the emitter depth is only around 250 Å.

Since ion implantation is associated with a number of drawbacks such as channeling, shadowing effects, and crystal defects, it may be difficult to reach an f_T above 50 to 60 GHz based on such a technology. The intrinsic base implantation has been replaced by rapid vapor deposition using B_2H_6 gas around 900°C.[83] The in-diffused boron profile will form a thin and low-resistive base. Also, the emitter implantation can be removed by utilizing *in situ* doped emitter technology (e.g., AsH_3 gas during polysilicon deposition).[84] Two detrimental effects are then avoided; namely, emitter perimeter depletion and the emitter plug effect.[85] The former effect causes a reduced doping concentration close to the emitter perimeter, whereas the latter implies the plugging of doping atoms in narrow emitter windows causing shallower junctions compared to larger openings on the same chip.

Arsenic came to replace phosphorus as the emitter impurity during the 1970s, mainly because of the emitter push-effect plaguing phosphorus monosilicon emitters. The phosphorus emitter has, however, experienced a renaissance in advanced bipolar transistors by introducing the so-called *in situ* phosphorus doped polysilicon (IDP) emitter.[86] One motivation for using IDP technology is the reduction in final emitter resistance compared to the traditional As polyemitter, in particular for aggressively down-scaled devices with very narrow emitter windows. In addition, the emitter drive-in for an IDP emitter is carried out at a lower thermal budget than the corresponding arsenic emitter due to the difference in diffusivity between the impurity atoms. Using IDP and RVD, very high f_T values (above 60 GHz) have been realized.[83] It has been suggested that the residual stress of the IDP emitter and the interfacial oxide between the poly- and the monosilicon creates a heteroemitter action for the device, thus explaining the high current gains of IDP bipolar transistors.[87]

Base electrode engineering in advanced devices has become an important field in reducing the total base resistance, thus improving f_{max} of the transistor. One straightforward method in lowering the base sheet resistance is by shunting the base polysilicon with an extended silicide across the total base electrode. This has recently been demonstrated in an $f_{max} = 60$ GHz double-poly process.[88] A still more effective concept is to integrate metal base electrodes.[89] This approach is combined with *in situ* doped boron polysilicon base electrodes as well as an IDP emitter in a double-poly process (see Fig. 3.16). The tungsten electrodes are fully self-aligned using WF_6-selective deposition. The technology, denoted SMI (self-aligned metal IDP), has been applied together with RVD base formation. The bipolar process was shown to produce f_T and f_{max} figures of 100 GHz at a breakdown voltage of 2.5 V.[90]

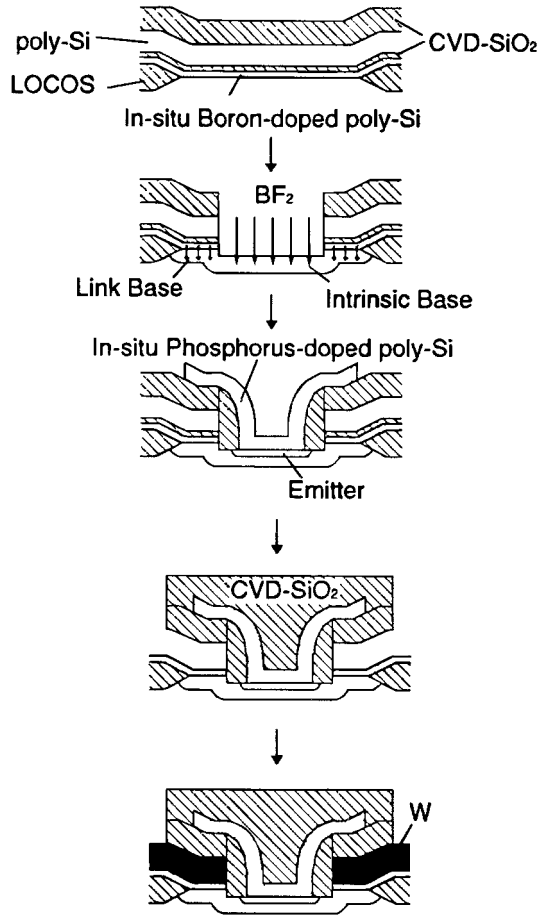


FIGURE 3.16 The self-aligned metal IDP process using selective deposition of tungsten base electrodes (after Ref. 89, copyright© 1997, IEEE).

Epitaxial Base

By introducing epitaxial film growth techniques for intrinsic base formation, the base width is readily controlled on the order of some hundred angstroms. Both selective and non-selective epitaxial growth (SEG and NSEG, respectively) have been reported. One example of a SEG transistor flow is illustrated in Fig. 3.17.[91] Not only the epitaxial base, but also the n^--collector is grown using SEG. The p^+-poly overhangs warrant a strong base link between the SEG intrinsic base and the base electrode. This $f_T = 44$ GHz process was capable of delivering divider circuits working at 25 GHz.

A natural extension of the Si epitaxy is to apply the previously mentioned SiGe epitaxy, thus creating a heterojunction bipolar transistor (HBT). For example, the transistor process in Fig. 3.17 was later extended to a SiGe process with $f_T = 61$ GHz and $f_{max} = 74$ GHz.[92] Apart from high speed, low base resistance is a trademark for many SiGe bipolar processes. For details of the SiGe HBT, the reader is referred to Chapter 5. Here, only some process integration points of view are given of this very important technology for advanced silicon-based bipolar devices during the 1990s.

While the first world records in terms of f_T and f_{max} were broken for non-self-aligned structures or mesa HBTs, planar self-aligned process schemes taking advantage of the benefits using SiGe have

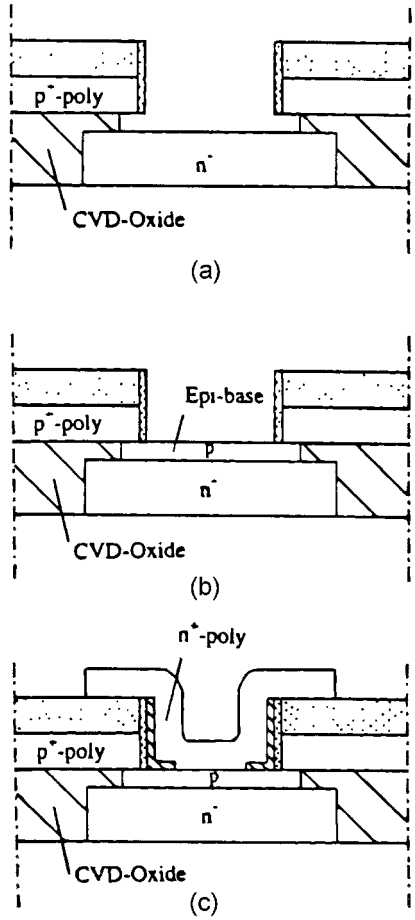


FIGURE 3.17 Process demonstrating selective epitaxial growth: (a) self-aligned formation of p^+-poly overhangs, (b) selective epitaxial growth of the intrinsic base, (c) emitter fabrication (after Ref. 91, copyright© 1992, IEEE).

subsequently been demonstrated. In recent years, both single-poly and double-poly SiGe transistors exhibiting excellent dc and ac properties have been presented. This also includes the quasi-self-aligned approach where a fully CMOS compatible process featuring an f_{max} of 71 GHz has been shown.[93] A similar concept featuring NSEG, so-called differential epitaxy, has been applied in the design of an HBT with a single-crystalline emitter rather than a polysilicon emitter.[94] This process, however, requires a very low thermal budget because of the high boron and germanium content in the intrinsic base of the transistor. Excellent properties for RF applications are made possible by this approach, such as high f_T and f_{max} around 50 GHz, combined with good dc properties and low noise figures.[95]

In double-poly structures, the extrinsic base is usually deposited prior to SiGe base epitaxy, which is then carried out by SEG (as shown in Fig. 3.17). Several groups report τ_d below 20 ps using this approach. One example of the most advanced bipolar technologies is the SiGe double-poly structure shown in Fig. 3.18.[96] This technology features the SMI base electrode technology, together with SEG of SiGe. The device is isolated by oxide-filled trenches. The reported τ_d was 7.7 ps and the f_{max} was 108 GHz, thus approaching the SiGe HBT mesa record of 160 GHz.[97] A device similar to the one in Fig. 3.18 was also reported to yield an f_T of 130 GHz.[98]

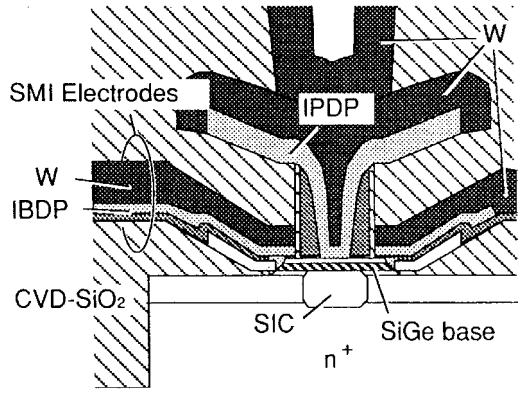


FIGURE 3.18 A state-of-the-art bipolar device featuring SMI electrodes, selectively grown epitaxial SiGe base, *in situ* doped polysilicon emitter/base and oxide-filled trenches (after Ref. 96, copyright© 1998, IEEE).

Future Trends

The shrinking of dimensions in bipolar devices, in particular for digital applications, will proceed one or two generations behind the MOS frontier, leading to a further reduction in τ_d. Several of the concepts reviewed above for advanced bipolar components are expected to be introduced in the next commercial high-performance processes; for example, *in situ* doped emitter and low-resistivity base electrodes. Similar to CMOS, the overall temperature budget must be reduced in processing. Evidently, bipolar technology in the future also continues to benefit from the progress made in CMOS technology; for example, in isolation and back-end processing. CMOS compatibility will be a general requirement for the majority of bipolar process development because of the strong interest in mixed-signal BiCMOS processes.

Advanced isolation technology combining deep and shallow trenches, perhaps on silicon-on-insulator (SOI) or high-resistivity substrates, marks one key trend in future bipolar transistors. Bipolar technology based on SOI substrates may well be accelerated by the current introduction of SOI into CMOS production. However, thermal effects for high current drive bipolar devices must be solved when using SOI. An interesting low-cost alternative insulating substrate for RF-bipolar technology is the silicon-on-anything concept recently presented.[99] In addition, copper metallization will be introduced for advanced bipolars provided that intermetal dielectrics as well as passive components are developed to meet this additional advantage.[100]

The epitaxial base constitutes another important trend where both Si and SiGe are expected to enhance performance in the high-frequency domain, although the introduction may be delayed due to progress in ion-implanted technology. In this respect, SEG technology has yet to prove its manufacturability.

Future Si-based bipolar technology with f_T and f_{max} greater than 100 GHz will continue to play an important role in small-density, high-performance designs. The most important applications are found in communication systems in the range 1 to 10 GHz (wireless telephony and local area networks) and 10 to 70 GHz (microwave and optical-fiber communication systems) where Si and/or SiGe bipolar technologies are expected to seriously challenge existing III-V technologies.[101]

Acknowledgments

We are grateful to G. Malm and M. Linder for carrying out the process simulations. The support from the Swedish High-Frequency Bipolar Technology Consortium is greatly acknowledged.

References

1. Ning, T. H. and Tang, D. D., Bipolar Trends, *Proc. IEEE*, 74, 1669, 1986.
2. Nakamura, T. and Nishizawa, H., Recent progress in bipolar transistor technology, *IEEE Trans. Electron Dev.*, 42, 390, 1995.
3. Warnock, J. D., Silicon bipolar device structures for digital applications: Technology trends and future directions, *IEEE Trans. Electron Dev.*, 42, 377, 1995.
4. Wilson, G. R., Advances in bipolar VLSI, *Proc. IEEE*, 78, 1707, 1990.
5. Barber, H. D., Bipolar device technology challenge and opportunity, *Can. J. Phys.*, 63, 683, 1985.
6. Baltus, P., Influence of process- and device parameters on the performance of portable rf communication circuits, *Proceedings of the 24th European Solid State Device Research Conference,* Hill, C. and Ashburn, P., Eds., 1994, 3.
7. Burghartz, J. N., BiCMOS process integration and device optimization: Basic concepts and new trends, *Electrical Eng.*, 79, 313, 1996.
8. Ashburn, P., *Design and Realization of Bipolar Transistors*, Wiley, Chichester, 1988.
9. Roulston, D. J., *Bipolar Semiconductor Devices*, McGraw-Hill, New York, 1990.
10. Tang, D. D. and Solomon, P. M., Bipolar transistor design for optimized power-delay logic circuits, *IEEE J. of Solid-State Circuits*, SC-14, 679, 1979.
11. Chor, E.-F., Brunnschweiler, A., and Ashburn, P., A propagation-delay expression and its application to the optimization of polysilicon emitter ECL processes, *IEEE J. of Solid-State Circuits*, 23, 251, 1988.
12. Stork, J. M. C., Bipolar transistor scaling for minimum switching delay and energy dissipation, in *1988 Int. Electron Devices Meeting Tech. Dig.*, 1988, 550.
13. Prinz, E. J. and Sturm, J. C., Current gain — Early voltage products in heterojunction bipolar transistors with nonuniform base bandgaps, *IEEE Electron. Dev. Lett.*, 12, 661, 1991.
14. Kurishima, K., An analytical expression of f_{max} for HBT's, *IEEE Trans. Electron Dev.*, 43, 2074, 1996.
15. Taylor, G. W. and Simmons, J. G., Figure of merit for integrated bipolar transistors, *Solid State Electronics*, 29, 941, 1986.
16. Hurkx, G. A. M., The relevance of f_T and f_{max} for the speed of a bipolar CE amplifier stage, *IEEE Trans. Electron Dev.*, 44, 775, 1997.
17. Larson, L. E., Silicon bipolar transistor design and modeling for microwave integrated circuit applications, in *1996 Bipolar Circuits Technol. Meeting Tech. Dig.*, 1996, 142.
18. Fang, W., Accurate analytical delay expressions for ECL and CML circuits and their applications to optimizing high-speed bipolar circuits, *IEEE J. of Solid-State Circuits*, 25, 572, 1990.
19. Silvaco International, 1997: VWF Interactive tools, *Athena User's Manual,* 1997.
20. Silvaco International, 1997: VWF Interactive tools, *Atlas User's Manual,* 1997.
21. Roulston, D. J., *BIPOLE3 User's Manual*, BIPSIM, Inc., 1996.
22. Alvarez, A. R., Abdi, B. L., Young, D. L., Weed, H. D., Teplik, J., and Herald, E. R., Application of statistical design and response surface methods to computer-aided VLSI device design, *IEEE Trans. Comp.-Aided Design*, 7, 272, 1988.
23. Johnson, E. O., Physical limitations on frequency and power parameters of transistors, *RCA Rev.*, 26, 163, 1965.
24. Ng, K. K., Frei, M. R., and King, C. A., Reevaluation of the $ftBV_{CEO}$ limit in Si bipolar transistors, *IEEE Trans. Electron Dev.*, 45, 1854, 1998.
25. Kirk, C. T., A theory of transistor cut-off frequency falloff at high current densities, *IRE Trans. Electron. Devices*, ED9, 164, 1962.
26. Roulston, D. J., *Bipolar Semiconductor Devices*, McGraw-Hill, New York, 1990, p. 257.
27. Konaka, S., Amemiya, Y., Sakuma, K., and Sakai, T., A 20 ps/G Si bipolar IC using advanced SST with collector ion implantation, in *1987 Ext. Abstracts 19th Conf. Solid-State Dev. Mater.*, Tokyo, 1987, 331.

28. Lu, P.-F. and Chen, T.-C., Collector-base junction avalanche effects in advanced double-poly self-aligned bipolar transistors, *IEEE Trans. Electron Dev.*, 36, 1182, 1989.
29. Tang, D. D. and Lu, P.-F., A reduced-field design concept for high-performance bipolar transistors, *IEEE Electron. Dev. Lett.*, 10, 67, 1989.
30. Ugajin M., Konaka, S., Yokohama K., and Amemiya, Y., A simulation study of high-speed hetero-emitter bipolar transistors, *IEEE Trans. Electron Dev.*, 36, 1102, 1989.
31. Inou, K. et al., 52 GHz epitaxial base bipolar transistor with high Early voltage of 26.5 V with box-like base and retrograded collector impurity profiles, in *1994 Bipolar Circuits Technol. Meeting Tech. Dig.*, 1994, 217.
32. Ikeda, T., Watanabe, A., Nishio, Y., Masuda, I., Tamba, N., Odaka, M., and Ogiue, K., High-speed BiCMOS technology with a buried twin well structure, *IEEE Trans. Electron Dev.*, 34, 1304, 1987.
33. Kumar, M. J., Sadovnikov, A. D., and Roulston, D. J., Collector design tradeoffs for low voltage applications of advanced bipolar transistors, *IEEE Trans. Electron Dev.*, 40, 1478, 1993.
34. Kumar, M. J. and Datta, K., Optimum collector width of VLSI bipolar transistors for maximum f_{max} at high current densities, *IEEE Trans. Electron Dev.*, 44, 903, 1997.
35. Roulston, D. J. and Hébert, F., Optimization of maximum oscillation frequency of a bipolar transistor, *Solid State Electronics*, 30, 281, 1987.
36. Early, J. M., Effects of space-charge layer widening in junction transistors, *Proc. IRE*, 42, 1761, 1954.
37. Shafi, Z. A., Ashburn, P., and Parker, G., Predicted propagation delay of Si/SiGe heterojunction bipolar ECL circuits, *IEEE J. of Solid-State Circuits*, 25, 1268, 1990.
38. Stork, J. M. C. and Isaac, R. D., Tunneling in base-emitter junctions, *IEEE Trans. Electron Dev.*, 30, 1527, 1983.
39. del Alamo, J. and Swanson, R. M., Forward-biased tunneling: A limitation to bipolar device scaling, *IEEE Electron. Dev. Lett.*, 7, 629, 1986.
40. Roulston, D. J., *Bipolar Semiconductor Devices*, McGraw-Hill, New York, 1990, 220 ff.
41. Van Wijnen, P. J. and Gardner, R. D., A new approach to optimizing the base profile for high-speed bipolar transistors, *IEEE Electron. Dev. Lett.*, 4, 149, 1990.
42. Suzuki, K., Optimum base doping profile for minimum base transit time, *IEEE Trans. Electron Dev.*, 38, 2128, 1991.
43. Yuan, J. S., Effect of base profile on the base transit time of the bipolar transistor for all levels of injection, *IEEE Trans. Electron Dev.*, 41, 212, 1994.
44. Ashburn, P. and Morgan, D. V., Heterojunction bipolar transistors, in *Physics and Technology of Heterojunction Devices,* Morgan D. V. and Williams R. H., Eds., Peter Peregrinus Ltd., London, 1991, chap. 6.
45. Harame, D. L., Stork, J. M. C., Meyerson, B. S., Hsu, K. Y.-J., Cotte, J., Jenkins, K. A., Cressler, J. D., Restle, P., Crabbé, E. F., Subbana, S., Tice, T. E., Scharf, B. W., and Yasaitis, J. A., Optimization of SiGe HBT technology for high speed analog and mixed-signal applications, in *1993 Int. Electron Devices Meeting Tech. Dig.*, 1993, 71.
46. Prinz, E. J., Garone, P. M., Schwartz, P. V., Xiao, X., and Sturm, J. C., The effects of base dopant outdiffusion and undoped $Si_{1-x}Ge_x$ junction spacer layers in $Si/Si_{1-x}Ge_x/Si$ heterojunction bipolar transistors, *IEEE Electron. Dev. Lett.*, 12, 42, 1991.
47. Hueting, R. J. E., Slotboom, J. W., Pruijmboom, A., de Boer, W. B., Timmering, C. E., and Cowern, N. E. B., On the optimization of SiGe-base bipolar transistors, *IEEE Trans. Electron Dev.*, 43, 1518, 1996.
48. Kerr, J. A. and Berz, F., The effect of emitter doping gradient on f_T in microwave bipolar transistors, *IEEE Trans. Electron Dev.*, ED-22, 15, 1975.
49. Slotboom, J. W. and de Graaf, H. C., Measurement of bandgap narrowing in silicon bipolar transistors, *Solid State Electronics*, 19, 857, 1976.
50. Cuthbertson, A. and Ashburn, P., An investigation of the tradeoff between enhanced gain and base doping in polysilicon emitter bipolar transistors, *IEEE Trans. Electron Dev.*, ED-32, 2399, 1985.

51. Ning, T. H. and Isaac, R. D., Effect on emitter contact on current gain of silicon bipolar devices, *IEEE Trans. Electron Dev.*, ED-27, 2051, 1980.
52. Post, R. C., Ashburn, P., and Wolstenholme, G. R., Polysilicon emitters for bipolar transistors: A review and re-evaluation of theory and experiment, *IEEE Trans. Electron Dev.*, 39, 1717, 1992.
53. Solomon, P. M. and Tang, D. D., Bipolar circuit scaling, in *1979 IEEE International Solid-State Circuits Conference Tech. Dig.*, 1979, 86.
54. Ning, T. H., Tang, D. D., and Solomon, P. M., Scaling properties of bipolar devices, in *1980 Int. Electron Devices Meeting Tech. Dig.*, 1980, 61.
55. Ashburn, P., *Design and Realization of Bipolar Transistors*, Wiley, Chichester, 1988, chap. 7.
56. Lary, J. E. and Anderson, R. L., Effective base resistance of bipolar transistors, *IEEE Trans. Electron Dev.*, ED-32, 2503, 1985.
57. Rein, H.-M., Design considerations for very-high-speed Si-bipolar IC's operating up to 50 Gb/s, *IEEE J. of Solid-State Circuits*, 8, 1076, 1996.
58. Schröter, M. and Walkey, D. J., Physical modeling of lateral scaling in bipolar transistors, *IEEE J. of Solid-State Circuits*, 31, 1484, 1996.
59. Pfost, M., Rein, H.-M., and Holzwarth, T., Modeling substrate effects in the design of high-speed Si-bipolar IC's, *IEEE J. of Solid-State Circuits*, 31, 1493, 1996.
60. Lohstrom, J., Devices and circuits for bipolar (V)LSI, *Proc. IEEE*, 69, 812, 1981.
61. Wolf, S., *Silicon Processing for the VLSI Area*, Vol. 2, Lattice Press, Sunset Beach, 1990, 532-533.
62. *ibid.*, p. 16-17.
63. Muller, R. S. and Kamins, T. I., *Device Electronics for Integrated Circuits*, 2nd ed., Wiley, New York, 1986, 307.
64. Parrillo, L. C., VLSI process integration, in *VLSI Technology*, Sze, S. M., Ed., McGraw-Hill, Singapore, 1983, 449 ff.
65. Ashburn, P., Polysilicon emitter technology, in *1989 Bipolar Circuits Technol. Meeting Tech. Dig.*, 1989, 90.
66. Li, G. P., Ning, T. H., Chuang, C. T., Ketchen, M. B., Tang, D.D., and Mauer, J., An advanced high-performance trench-isolated self-aligned bipolar technology, *IEEE Trans. Electron Dev.*, ED-34, 2246, 1987.
67. Tang, D. D., Solomon, P. M., Isaac, R. D., and Burger, R. E., 1.25 μm deep-groove-isolated self-aligned bipolar circuits, *IEEE J. of Solid-State Circuits*, SC-17, 925, 1982.
68. Yano, K., Nakazato, K., Miyamoto, M., Aoki, M., and Shimohigashi, K., A high-current-gain low-temperature pseudo-HBT utilizing a sidewall base-contact structure (SICOS), , *IEEE Trans. Electron Dev.*, 10, 452, 1989.
69. Tang, D. D.-L., Chen, T.-C., Chuang, C. T., Cressler, J. D., Warnock, J., Li, G.-P., Polcari, M. R., Ketchen, M. B., and Ning, T. H., The design and electrical characteristics of high-performance single-poly ion-implanted bipolar transistors, *IEEE Trans. Electron Dev.*, 36, 1703, 1989.
70. de Jong, J. L., Lane, R. H., de Groot, J. G., and Conner, G. W., Electron recombination at the silicided base contact of an advanced self-aligned polysilicon emitter, in *1988 Bipolar Circuits Technol. Meeting Tech. Dig.*, 1988, 202.
71. Li, G. P., Chen, T.-C., Chuang, C.-T., Stork, J. M. C., Tang, D. D., Ketchen, M. B., and Wang, L.-K., Bipolar transistor with self-aligned lateral profile, *IEEE Electron. Dev. Lett.*, EDL-8, 338, 1987.
72. Niel, S., Rozeau, O., Ailloud, L., Hernandez, C., Llinares, P., Guillermet, M., Kirtsch, J., Monroy, A., de Pontcharra, J., Auvert, G., Blanchard, B., Mouis, M., Vincent, G., and Chantre, A., A 54 GHz f_{max} implanted base 0.35 μm single-polysilicon bipolar transistor, in *1997 Int. Electron Devices Meeting Tech. Dig.*, 1997, 807.
73. Tang, D. D., Chen, T.-C., Chuang, C.-T., Li, G. P., Stork, J. M. C., Ketchen, M. B., Hackbarth, E., and Ning, T. H., Design considerations of high-performance narrow-emitter bipolar transistors, *IEEE Electron. Dev. Lett.*, EDL-8, 174, 1987.

74. Ning, T. H., Isaac, R. D., Solomon, P. M., Tang, D. D.-L., Yu, H.-N., Feth, G. C., and Wiedmann, S. K., Self-aligned bipolar transistors for high-performance and low-power-delay VLSI, *IEEE Trans. Electron Dev.*, ED-28, 1010, 1981.
75. Chantre, A., Festes, G., Giroult-Matlakowski, G., and Nouailhat, An investigation of nonideal base currents in advanced self-aligned "etched-polysilicon" emitter bipolar transistors, *IEEE Trans. Electron Dev.*, 38, 1354, 1991.
76. Sun, S. W., Denning, D., Hayden, J. D., Woo, M., Fitch, J. T., and Kaushik, V., A nonrecessed-base, self-aligned bipolar structure with selectively deposited polysilicon emitter, *IEEE Trans. Electron Dev.*, 39, 1711, 1992.
77. Chuang, C.-T., Li, G. P., and Ning, T. H., Effect of off-axis implant on the characteristics of advanced self-aligned bipolar transistors, *IEEE Electron. Dev. Lett.*, EDL-8, 321, 1987.
78. Hayden, J. D., Burnett, J. D., Pfiester, J. R., and Woo, M. P., A new technique for forming a shallow link base in a double polysilicon bipolar transistor, *IEEE Trans. Electron Dev.*, 41, 63, 1994.
79. Maritan, C. M. and Tarr, N. G., Polysilicon emitter p-n-p transistors, *IEEE Trans. Electron Dev.*, 36, 1139, 1989.
80. Warnock, J., Lu, P.-F., Cressler, J. D., Jenkins, K. A., and Sun, J. Y. C., 35 GHz/35 psec ECL pnp technology, in *1990 Int. Electron Devices Meeting Tech. Dig.*, 1990, 301.
81. Chantre, A., Gravier, T., Niel, S., Kirtsch, J., Granier, A., Grouillet, A., Guillermet, M., Maury, D., Pantel, R., Regolini, J. L., and Vincent, G., The design and fabrication of 0.35 μm single-polysilicon self-aligned bipolar transistors, *Jpn. J. Appl. Phys.*, 37, 1781, 1998.
82. Warnock, J., Cressler, J. D., Jenkins, K. A., Chen, T.-C., Sun, J. Y.-C., and Tang, D. D., 50-GHz self-aligned silicon bipolar transistors with ion-implanted base profiles, *IEEE Electron. Dev. Lett.*, 11, 475, 1990.
83. Uchino, T., Shiba, T., Kikuchi, T., Tamaki, Y., Watanabe, A., and Kiyota, Y., Very-high-speed silicon bipolar transistors with *in situ* doped polysilicon emitter and rapid vapor-phase doping base, *IEEE Trans. Electron Dev.*, 42, 406, 1995.
84. Burghartz, J. N., Megdnis, A. C., Cressler, J. D., Sun, J. Y.-C., Stanis, C. L., Comfort, J. H., Jenkins, K. A., and Cardone, F., Novel *in-situ* doped polysilicon emitter process with buried diffusion source (BDS), *IEEE Electron. Dev. Lett.*, 12, 679, 1991.
85. Burghartz, J. N., Sun, J. Y.-C., Stanis, C. L., Mader, S. R., and Warnock, J. D., Identification of perimeter depletion and emitter plug effects in deep-submicrometer, shallow-junction polysilicon emitter bipolar transistors, *IEEE Trans. Electron Dev.*, 39, 1477, 1992.
86. Shiba, T., Uchino, T., Ohnishi, K., and Tamaki, Y., *In situ* phosphorus-doped polysilicon emitter technology for very high-speed small emitter bipolar transistors, *IEEE Trans. Electron Dev.*, 43, 889, 1996.
87. Kondo, M., Shiba, T., and Tamaki, Y., Analysis of emitter efficiency enhancement induced by residual stress for *in situ* phosphorus-doped emitter transistors, *IEEE Trans. Electron Dev.*, 44, 978, 1997.
88. Böck, J., Meister, T. F., Knapp, H., Aufinger, K., Wurzer. M., Gabl, R., Pohl, M., Boguth, S., Franosch, M., and Treitinger, L., 0.5 μm / 60 GHz f_{max} implanted base Si bipolar technology, in *1998 Bipolar Circuits Technol. Meeting Tech. Dig.*, 1998, 160.
89. Onai, T., Ohue, E., Tanabe, M., and Washio, K., 12-ps ECL using low-base-resistance Si bipolar transistor by self-aligned metal/IDP technology, *IEEE Trans. Electron Dev.*, 44, 2207, 1997.
90. Kiyota, Y., Ohue, E., Washio, K., Tanabe, M., and Inade, T., Lamp-heated rapid vapor-phase doping technology for 100-GHz Si bipolar transistors, in *1996 Bipolar Circuits Technol. Meeting Tech. Dig.*, 1996, 173.
91. Meister, T. F., Stengl, R., Meul, H. W., Packan, P., Felder, A., Klose, H., Schreiter, R., Popp, J., Rein, H. M., and Treitinger, L., Sub-20 ps silicon bipolar technology using selective epitaxial growth, in *1992 Int. Electron Devices Meeting Tech. Dig.*, 1992, 401.

92. Meister, T. F., Schäfer, H., Franosch, M., Molzer, W., Aufinger, K., Scheler, U., Walz, C., Stolz, M., Boguth, S., and Böck, J., SiGe base bipolar technology with 74 GHz f_{max} and 11 ps gate delay, in *1995 Int. Electron Devices Meeting Tech. Dig.*, 1995, 739.
93. Chantre, A., Marty, M., Regolini, J. L., Mouis, M., de Pontcharra, J., Dutartre D., Morin, C., Gloria, D., Jouan, S., Pantel, R., Laurens, M., and Monroy, A., A high performance low complexity SiGe HBT for BiCMOS integration, in *1998 Bipolar Circuits Technol. Meeting Tech. Dig.*, 1998, 93.
94. Schüppen, A., König, U., Gruhle, A., Kibbel, H., and Erben, U., The differential SiGe-HBT, *Proceedings of the 24th European Solid State Device Research Conference,* Hill, C. and Ashburn, P., Eds., 1994, 469.
95. Schüppen, A., Dietrich, H., Seiler, U., von der Ropp, H., and Erben, U., A SiGe RF technology for mobile communication systems, *Microwave Engineering Europe*, June 1998, 39.
96. Ohue, E., Oda, K., Hayami, R., and Washio, K., A 7.7 ps CML using selective-epitaxial SiGe HBTs, *1998 Bipolar Circuits Technol. Meeting Tech. Dig.*, 1998, 97.
97. Schüppen, A., Erben, U., Gruhle, A., Kibbel, H., Schumacher, H., and König, U., Enhanced SiGe heterojunction bipolar transistors with 160 GHz-f_{max}, *1995 Int. Electron Devices Meeting Tech. Dig.*, 1995, 743.
98. Oda, K., Ohue, E., Tanabe, M., Shimamoto, H., Onai, T., and Washio, K., 130-GHz f_T SiGe HBT technology, in *1997 Int. Electron Devices Meeting Tech. Dig.*, 1997, 791.
99. Dekker, R., Baltus, P., van Deurzen, M., v.d. Einden, W., Maas, H., and Wagemans, A., An ultra low-power RF bipolar technology on glass, *1997 Int. Electron Devices Meeting Tech. Dig.*, 1997, 921.
100. Hashimoto, T., Kikuchi, T., Watanabe, K., Ohashi, N., Saito, N., Yamaguchi, H., Wada, S., Natsuaki, N., Kondo, M., Kondo, S., Homma, Y., Owada, N., and Ikeda, T., A 0.2 μm bipolar-CMOS technology on bonded SOI with copper metallization for ultra high-speed processors, *1998 Int. Electron Devices Meeting Tech. Dig.*, 1998, 209.
101. König, U., SiGe & GaAs as competitive technologies for RF-applications, *1998 Bipolar Circuits Technol. Meeting Tech. Dig.*, 1998, 87.

Cristoloveanu, S.“Silicon on Insulatpr Technology”
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

4

Silicon on Insulator Technology

Sorin Cristoloveanu
Institut National Polytechnique de Grenoble

4.1 Introduction

Silicon on Insulator (SOI) technology (more specifically, silicon on sapphire) was originally invented for the niche of radiation-hard circuits. In the last 20 years, a variety of SOI structures have been conceived with the aim of dielectrically separating, using a buried oxide (Fig. 4.1(b)), the active device volume from the silicon substrate.[1] Indeed, in an MOS transistor, only the very top region (0.1–0.2 μm thick, i.e., less than 0.1% of the total thickness) of the silicon wafer is useful for electron transport and device operation, whereas the substrate is responsible for detrimental, parasitic effects (Fig. 4.1(a)).

More recently, the advent of new SOI materials (Unibond, ITOX) and the explosive growth of portable microelectronic devices have attracted considerable attention on SOI for the fabrication of low-power (LP), low-voltage (LV), and high-frequency (HF) CMOS circuits.

The aim of this chapter is to overview the state-of-the-art of SOI technologies, including the material synthesis (Section 4.2), the key advantages of SOI circuits (Section 4.3), the structure and performance of typical devices (Section 4.4), and the operation modes of fully depleted (Section 4.5), and partially depleted SOI MOSFETs (Section 4.6). Section 4.7 is dedicated to short-channel effects. The main challenges that SOI is facing, in order to successfully compete with bulk–Si in the commercial arena, are critically discussed in Section 4.8.

4.2 Fabrication of SOI Wafers

Many techniques, more or less mature and effective, are available for the synthesis of SOI wafers.[1]

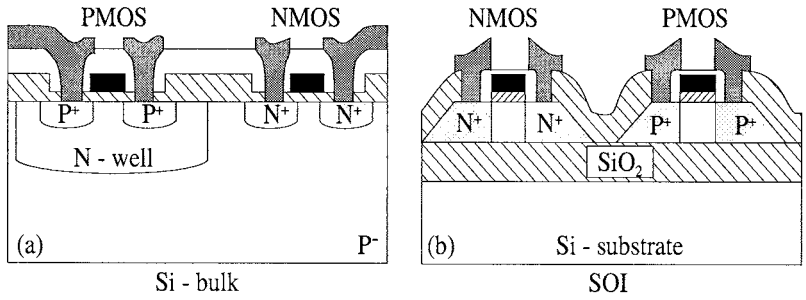


FIGURE 4.1 Basic architecture of MOS transistors in (a) bulk silicon and (b) SOI.

Silicon on Sapphire

Silicon on sapphire (SOS, Fig. 4.2(a_1)) is the initial member of SOI family. The epitaxial growth of Si films on Al_2O_3 gives rise to small silicon islands that eventually coalesce. The interface transition region contains crystallographic defects due to the lattice mismatch and Al contamination from the substrate. The electrical properties suffer from lateral stress, in-depth inhomogeneity of SOS films, and defective transition layer.[2]

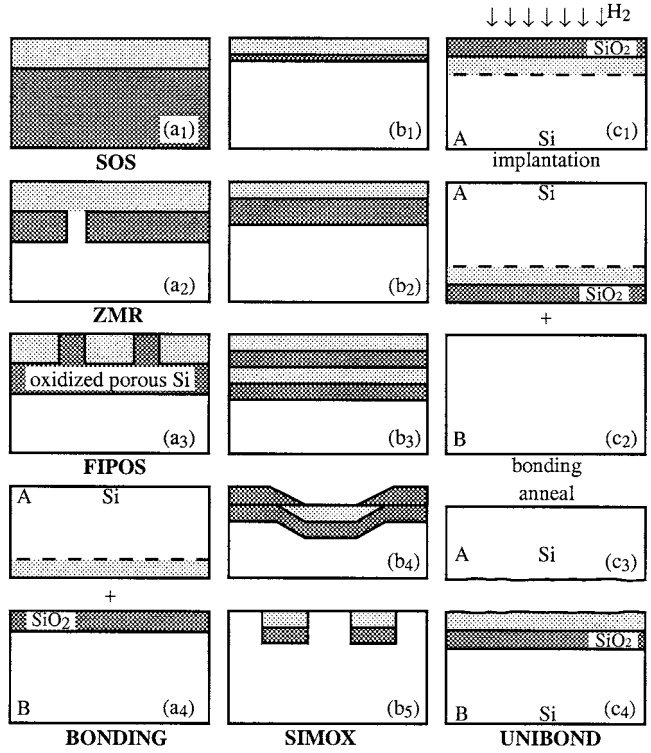


FIGURE 4.2 SOI family: (a) SOS, ZMR, FIPOS, and wafer bonding, (b) SIMOX variants, (c) UNIBOND processing sequence.

SOS has recently undergone a significant lifting: larger wafers and thinner films with higher crystal quality. This improvement is achieved by *solid-phase epitaxial regrowth*. Silicon ions are implanted to amorphise the film and erase the memory of damaged lattice and interface. Annealing allows the epitaxial regrowth of the film, starting from the 'seeding' surface towards the $Si–Al_2O_3$ interface. The result is visible in terms of higher carrier mobility and lifetime; 100-nm thick SOS films with good quality have recently been grown on 6 in. wafers.[3]

Thanks to the 'infinite' thickness of the insulator, SOS looks promising for the integration of RF and radiation-hard circuits.

ELO and ZMR

The *epitaxial lateral overgrowth* (ELO) method consists of growing a single-crystal Si film on a seeded and, often, patterned oxide (Fig. 4.2(a_2)). Since the epitaxial growth proceeds in both lateral and vertical directions, the ELO process requires a post-epitaxy thinning of the Si film. Alternatively, poly-silicon can be deposited directly on SiO_2; subsequently, *zone melting recrystallization* (ZMR) is achieved by scanning high-energy sources (lamps, lasers, beams, or strip heaters) across the wafer. The ZMR process can be seeded or unseeded; it is basically limited by the lateral extension of single-crystal regions, free from grain subboundaries and associated defects. ELO and ZMR are basic techniques for the integration of 3-D stacked circuits.

FIPOS

The FIPOS method (*full isolation by porous oxidized silicon*) makes use of the very large surface-to-volume ratio (10^3 cm^2 per cm^3) of porous silicon which is, thereafter, subject to selective oxidation (Fig. 4.2(a_3)). The critical step is the conversion of selected p-type regions of the Si wafer into porous silicon, via anodic reaction. FIPOS may enlighten Si technology because there are prospects, at least from a conceptual viewpoint, for combining electroluminescent porous Si devices with fast SOI–CMOS circuits.

SIMOX

In the last decade, the dominant SOI technology was SIMOX (*separation by implantation of oxygen*). The buried oxide (BOX) is synthesized by internal oxidation during the deep implantation of oxygen ions into a Si wafer. Annealing at high temperature (1320°C, for 6 h) is necessary to recover a suitable crystalline quality of the film. High current implanters (100 mA) have been conceived to produce 8 in. wafers with good thickness uniformity, low defect density (except threading dislocations: 10^4–10^6 cm^{-2}), sharp $Si–SiO_2$ interface, robust BOX, and high carrier mobility.[4]

The family of SOI structures is presented in Figure 4.2(b):

- Thin and thick Si films fabricated by adjusting the implant energy.
- Low-dose SIMOX: a dose of 4×10^{17} O^+/cm^2 and an additional oxygen-rich anneal for enhanced BOX integrity (ITOX process) yield a 0.1-μm thick BOX (Fig. 4.2(b_1)).
- Standard SIMOX obtained with 1.8×10^{18} O^+/cm^2 implant dose, at 190keV and 650°C; the thicknesses of the Si film and BOX are roughly 0.2 μm and 0.4 μm, respectively (Fig. 4.2(b_2)).
- Double SIMOX (Fig. 4.2(b_3)), where the Si layer sandwiched between the two oxides can serve for interconnects, wave guiding, additional gates, or electric shielding.
- Laterally-isolated single-transistor islands (Fig. 4.2(b_4)), formed by implantation through a patterned oxide.
- Interrupted oxides (Fig. 4.2(b_5)) which can be viewed as SOI regions integrated into a bulk Si wafer.

Wafer Bonding

Wafer bonding (WB) and etch-back stands as a rather mature SOI technology. An oxidized wafer is mated to another SOI wafer (Fig. 4.2(a_4)). The challenge is to drastically thin down one side of the bonded structure in order to reach the targeted thickness of the silicon film. Etch-stop layers can be achieved by doping steps (P^+/P^-, P/N) or porous silicon (Eltran process).[5] The advantage of wafer bonding is to provide unlimited combinations of BOX and film thicknesses, whereas its weakness comes from the difficulty to produce ultra-thin films with good uniformity.

UNIBOND

A recent, revolutionary bonding-related process (UNIBOND) uses the deep implantation of hydrogen into an oxidized Si wafer (Fig. 4.2(c_1)) to generate microcavities and thus circumvent the thinning problem.[6] After bonding wafer *A* to a second wafer *B* and subsequent annealing to enhance the bonding strength (Fig. 4.2(c_2)), the hydrogen-induced microcavities coalesce. The two wafers separate, not at the bonded interface but at a depth defined by the location of hydrogen microcavities. This mechanism, named *Smart-cut*, results in a rough SOI structure (Fig. 4.2(c_4)). The process is completed by touch-polishing to erase the surface roughness.

The extraordinary potential of the Smart-cut approach comes from several distinct advantages: (1) the etch-back step is avoided, (2) the second wafer (Fig. 4.2(c_3)) being recyclable, UNIBOND is a single-wafer process, (3) only conventional equipment is needed for mass production, (4) relatively inexpensive 12 in. wafers are manufacturable, and (5) the thickness of the silicon film and/or buried oxide can be adjusted to match most device configurations (ultra-thin CMOS or thick-film power transistors and sensors). The defect density in the film is very low, the electrical properties are excellent, and the BOX quality is comparable to that of the original thermal oxide. The Smart-cut process is adaptable to a variety of materials: SiC or III–V compounds on insulator, silicon on diamond, etc. Smart-cut can be used to transfer already fabricated bulk-Si CMOS circuits on glass or on other substrates.

4.3 Generic Advantages of SOI

SOI circuits consist of single-device islands dielectrically isolated from each other and from the underlying substrate (Fig. 4.1(b)). The lateral isolation offers more compact design and simplified technology than in bulk silicon; there is no need of wells or interdevice trenches. In addition, the vertical isolation renders the *latch-up* mechanisms impossible.

The source and drain regions extend down to the buried oxide; thus, the junction surface is minimized. This implies reduced leakage currents and junction capacitances, which further translates into improved speed, lower power, and wider temperature range of operation.

The limited extension of drain and source regions allows SOI devices to be less affected by short-channel effects, originated from 'charge sharing' between gate and junctions. Besides the outstanding tolerance of transient radiation effects, SOI MOSFETs experience a lower electric-field peak than in bulk Si and are potentially more immune to hot carrier damage.

It is in the highly competitive domain of LV/LP circuits, operated with one-battery supply (0.9–1.5V), that SOI can express its entire potential. A small gate voltage gap is suited to switch a transistor from off- to on-state. SOI offers the possibility to achieve a quasi-ideal subthreshold slope (60mV/decade at room temperature); hence, a threshold voltage shrunk below 0.3V. Low leakage currents limit the *static* power dissipation, as compared to bulk Si, whereas, the *dynamic* power dissipation is minimized by the combined effects of low parasitic capacitances and reduced voltage supply.

Two arguments can be given to outline unequivocally the advantage of SOI over bulk Si:

- Operation at similar *voltage* consistently shows about 30% increase in performance, whereas operation at similar *low-power* dissipation yields as much as 300% performance gain in SOI. It is

believed, at least in the SOI community, that SOI circuits of generation (n) and bulk-Si circuits from the *next* generation ($n + 1$) perform comparably.

- Bulk Si technology does attempt to mimic a number of features that are natural in SOI: the double-gate configuration is reproduced by processing surrounded-gate vertical MOSFETs on bulk Si, full depletion is approached by tailoring a low-high step doping, and the dynamic-threshold operation is borrowed from SOI.

The problem for SOI is that such an enthusiastic list of merits did not perturb the fantastic progress and authority of bulk Si technology. There has been no room or need so far for an alternative technology such as SOI. However, the SOI community remains confident that the SOI advantages together with the predictable approach of bulk-Si limits will be enough for SOI to succeed soon.

4.4 SOI Devices

CMOS Circuits

High-performance SOI CMOS circuits, compatible with LV/LP and high-speed ULSI applications have been repeatedly demonstrated on submicron devices. Quarter-micron ring oscillators showed delay times of 14 ps/stage at 1.5 V[7] and of 45 ps/stage at 1V.[8] PLL operated at 2.5 V and 4 GHz dissipate 19 mW only.[8] Microwave SOS MOSFETs, with T-gate configuration, had 66-MHz maximum frequency and low noise figure.[3]

More complex SOI circuits, with direct impact on mainstream microelectronics, have also been fabricated: 0.5 V–200 MHz microprocessor,[9] 4 Mb SRAM,[10] 16 Mb and 1 Gb DRAM,[11] etc.[1,12] Several companies (IBM, Motorola, Sharp) have announced the imminent commercial deployment of 'SOI-enhanced' PC processors and mobile communication devices.

CMOS SOI circuits show capability of successful operation at temperatures higher than 300°C: the leakage currents are much smaller and the threshold voltage is less temperature sensitive (≈0.5mV/°C for fully depleted MOSFETs) than in bulk Si.[13] In addition, many SOI circuits are radiation-hard, able to sustain doses above 10 Mrad.

Bipolar Transistors

As a consequence of the small film thickness, most bipolar transistors have a lateral configuration. The implementation of BiCMOS technology on SOI has resulted in devices with a cutoff frequency above 27 GHz.[14] Hybrid MOS–bipolar transistors with increased current drive and transconductance are formed by connecting the gate to the floating body (or base); the MOSFET action governs in strong inversion whereas, in weak inversion, the bipolar current prevails.[12]

Vertical bipolar transistors have been processed in thick-film SOI (wafer bonding or epitaxial growth over SIMOX). An elegant solution for thin-film SOI is to replace the buried collector by an inversion layer activated by the back gate.[12]

High-Voltage Devices

Lateral double-diffused MOSFETs (DMOS), with long drift region, were fabricated on SIMOX and showed 90 V/1.3A capability.[15] Vertical DMOS can be accommodated in thicker wafer-bonding SOI.

The SIMOX process offers the possibility to synthesize locally a buried oxide ('interrupted' SIMOX, Fig. 4.2(b_5)). Therefore, a vertical power device (DMOS, IGBT, UMOS, etc.), located in the bulk region of the wafer, can be controlled by a low-power CMOS/SOI circuit (Fig. 4.3(a)). A variant of this concept is the 'mezzanine' structure, which served for the fabrication of a 600V/25A smart-power device.[16] Double SIMOX (Fig. 4.2(b_3)) has also been used to combine a power MOSFET with a double-shielded high-voltage lateral CMOS and an intelligent low-voltage CMOS circuit.[17]

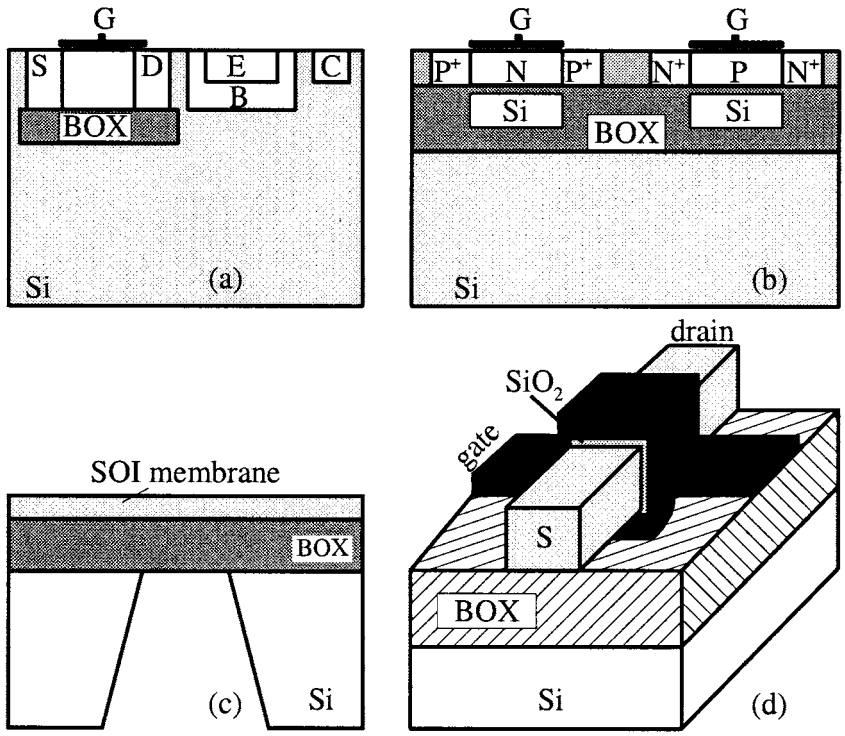


FIGURE 4.3 Examples of innovative SOI devices: (a) combined bipolar (or high power) bulk-Si transistor with low-voltage SOI CMOS circuits, (b) dual-gate transistors, (c) pressure sensor, and (d) gate-all-around (GAA) MOSFET.

Innovative Devices

Most innovative devices make use of special SOI features, including the possibility to (1) combine bulk Si and SOI on a single chip (Fig. 4.3(a)), (2) adjust the thickness of the Si overlay and buried oxide, and (3) implement additional gates in the buried oxide (Fig. 4.3(b)), by ELO process or by local oxidation of the sandwiched Si layer in double SIMOX (Fig. 4.2(b_3)).

SOI is an ideal material for microsensors because the Si/BOX interface gives a perfect etch-stop mark, making it possible to fabricate very thin membranes (Fig. 4.3(c)). Transducers for detection of pressure, acceleration, gas flow, temperature, radiation, magnetic field, etc. have successfully been integrated on SOI.[1,16]

The feasibility of three-dimensional circuits has been demonstrated on ZMR structures. For example, an image-signal processor is organized in three levels: photodiode arrays in the upper SOI layer, fast A/D converters in the intermediate SOI layer, and arithmetic units and shift registers in the bottom bulk Si level.[18]

The *gate all–around* (GAA) transistor of Fig. 4.3(d), based on the concept of volume inversion, is fabricated by etching a cavity into the BOX and wrapping the oxidized transistor body into a poly-Si gate.[12] Similar devices include the Delta transistor[19] and various double-gate MOSFETs.

The family of SOI devices also includes optical waveguides and modulators, microwave transistors integrated on high-resistivity SIMOX, twin-gate MOSFETs, and other exotic devices.[1,12] They do not belong to science fiction: the devices have already been demonstrated in terms of technology and functionality… even if most people still do not believe that they can operate.

4.5 Fully–Depleted SOI Transistors

In SOI MOSFETs (Fig. 4.1(b)), inversion channels can be activated at both the front Si–SiO_2 interface (via gate modulation V_{G_1}) and back Si–BOX interface (via substrate, back-gate bias V_{G_2}).

Full depletion means that the depletion region covers the entire transistor body. The depletion charge is constant and cannot extend according to the gate bias. A better coupling develops between the gate bias and the inversion charge, leading to enhanced drain current. In addition, the front- and back-surface potentials become coupled too. The coupling factor is roughly equal to the thickness ratio between gate oxide and buried oxide. The electrical characteristics of one channel vary remarkably with the bias applied to the opposite gate. Due to *interface coupling*, the front-gate measurements are all reminiscent of the back-gate bias and quality of the buried oxide and interface.

Totally new $I_D(V_G)$ relations apply to fully depleted SOI–MOSFETs whose complex behavior is controlled by both gate biases. The typical characteristics of the front-channel transistor are schematically illustrated in Fig. 4.4, for three distinct bias conditions of the back interface (inversion, depletion, and accumulation), and will be explained next.

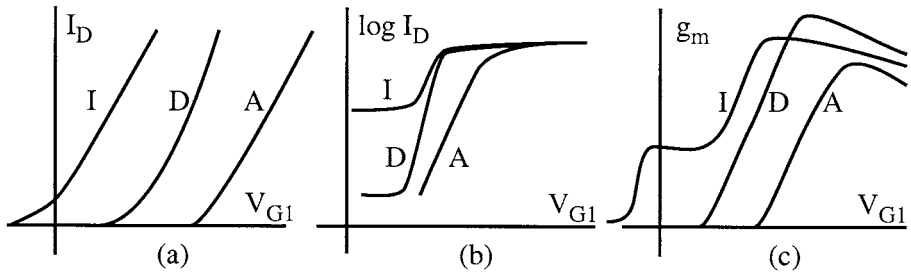


FIGURE 4.4 Generic front-channel characteristics of a fully depleted n-channel SOI MOSFET for accumulation (A), depletion (D), and inversion (I) at the back interface: (a) $I_D(V_{G_1})$ curves in strong inversion, (b) log $I_D(V_{G_1})$ curves in weak inversion, and (c) transconductance $g_m(V_{G_1})$ curves.

Threshold Voltage

The lateral shift of $I_D(V_{G_1})$ curves (Fig. 4.4(a)) is explained by the linear variation of the front-channel threshold voltage, $V_{T_1}^{dep}$, with back-gate bias. This *potential coupling* causes $V_{T_1}^{dep}$ to decrease linearly, with increasing V_{G_2}, between two plateaus corresponding, respectively, to accumulation and inversion at the back interface[20]:

$$V_{T_1}^{dep} = V_{T_1}^{acc} - \frac{C_{si}C_{ox_2}(V_{G_2} - V_{G_2}^{acc})}{C_{ox_1}(C_{ox_2} + C_{si} + C_{it_2})} \tag{4.1}$$

where $V_{T_1}^{acc}$ is the threshold voltage when the back interface is accumulated

$$V_{T_1}^{acc} = \Phi_{fb_1} + \frac{C_{ox_1} + C_{si} + C_{it_1}}{C_{ox_1}} 2\Phi_F - \frac{Q_{si}}{2C_{ox_1}} \tag{4.2}$$

and $V_{G_2}^{acc}$ is given by

$$V_{G_2}^{acc} = \Phi_{fb_2} - \frac{C_{si}}{C_{ox_2}} 2\Phi_F - \frac{Q_{si}}{2C_{ox_2}} \tag{4.3}$$

In the above equations, C_{si}, C_{ox}, and C_{it} are the capacitances of the fully depleted film, oxide, and interface traps, respectively; Q_{si} is the depletion charge, Φ_F is the Fermi potential, and Φ_{fb} is the flat-band potential. The subscripts 1 and 2 hold for the front- or the back-channel parameters and can be interchanged to account for the variation of the back-channel threshold voltage V_{T_2} with V_{G_1}.

The difference between the two plateaus, $\Delta V_{T_1} = (C_{si}/C_{ox_1})\,2\Phi_F$, slightly depends on doping, whereas the slope does not. We must insist on the polyvalence of Eqs.(4.1) to (4.3) as compared to the simple case of bulk Si MOSFETs (or partially depleted MOSFETs), where

$$V_{T_1} = \Phi_{fb_1} + \left(1 + \frac{C_{it_1}}{C_{ox_1}}\right)2\Phi_F + \frac{\sqrt{4q\varepsilon_{si}N_A\Phi_F}}{C_{ox_1}} \tag{4.4}$$

The extension to p-channels or accumulation-mode SOI–MOSFETs is also straightforward.[1]

In fully depleted MOSFETs, the threshold voltage decreases in thinner films (i.e., reduced depletion charge) until quantum effects arise and lead to the formation of a 2-D subband system. In ultra-thin films ($t_{si} \leq 10$ nm), the separation between the ground state and the bottom of the conduction band increases with reducing thickness: a V_T rebound is then observed.[21]

Subthreshold Slope

For depletion at the back interface, the subthreshold slope (Fig. 4.4(b)) is very steep and the subthreshold *swing S* is given by[22]:

$$S_1^{dep} = 2.3\frac{kT}{q}\left(1 + \frac{C_{it_1}}{C_{ox_1}} + \alpha_1\frac{C_{si}}{C_{ox_1}}\right) \tag{4.5}$$

The interface coupling coefficient α_1

$$\alpha_1 = \frac{C_{ox_2} + C_{it_2}}{C_{si} + C_{ox_2} + C_{it_2}} < 1 \tag{4.6}$$

accounts for the influence of back interface traps C_{it_2} and buried oxide thickness C_{ox_2} on the front channel current.[22]

In the ideal case, where $C_{it_{1,2}} \cong 0$ and the buried oxide is much thicker than both the film and the gate oxide (i.e., $\alpha_1 \cong 0$), the swing approaches the theoretical limit $S_1^{dep} \cong 60$ mV/decade at 300K. Accumulation at the back interface does decouple the front inversion channel from back interface defects but, in turn, makes α_1 tend to unity (as in bulk–Si or partially depleted MOSFETs), causing an overall degradation of the swing.

It is worth noting that the above simplified analysis and equations are valid only when the buried oxide is thick enough, such that substrate effects occurring underneath the BOX can be overlooked. The capacitances of the BOX and Si substrate are actually connected in series. Therefore, the swing may depend, essentially for thin buried oxides, on the density of the traps and surface charge (accumulation, depletion, or inversion) at the *third* interface: BOX–Si substrate. The general trend is that the subthreshold slope improves for thinner silicon films and thicker buried oxides.

Transconductance

For strong inversion and ohmic region of operation, the front-channel drain current and transconductance are given by

$$I_D = \frac{C_{ox_1}WV_D}{L} \cdot \frac{\mu_1}{1 + \theta_1(V_{G_1} - V_{T_1}(V_{G_2}))} \cdot (V_{G_1} - V_{T_1}(V_{G_2})) \tag{4.7}$$

$$g_{m_1} = \frac{C_{ox_1} W V_D}{L} \cdot \frac{\mu_1}{[1 + \theta_1(V_{G_1} - V_{T_1}(V_{G_2}))]^2} \tag{4.8}$$

where μ_1 is the mobility of front-channel carriers, and θ_1 is the mobility attenuation coefficient.

The complexity of the transconductance curves in fully depleted MOSFETs (Fig. 4.4(c)) is explained by the influence of the back gate bias via $V_{T_1}(V_{G_2})$. The effective mobility and transconductance peak are maximum for depletion at the back interface, due to combined effects of reduced vertical field and series resistances.

An unusual feature is the distortion of the transconductance (curve I, Fig. 4.4(c)), which reflects the possible activation of the back channel, far before the inversion charge build-up is completed at the front channel.[23] While the front interface is still depleted, increasing V_{G_1} reduces the back threshold voltage and eventually opens the *back* channel. The plateau of the front-channel transconductance (Fig. 4.4(c)) can be used to derive directly the back-channel mobility.

Volume Inversion

In thin and low-doped films, the simultaneous activation of front and back channels induces by continuity (i.e., *charge coupling*) the onset of *volume inversion*.[24] Unknown in bulk Si, this effect enables the inversion charge to cover the whole film. Self-consistent solutions of Poisson and Schrödinger equations indicate that the maximum density of the inversion charge is reached in the middle of the film. This results in increased current drive and transconductance, attenuated influence of interface defects (traps, fixed charges, roughness), and reduced $1/f$ noise.

Double-gate MOSFETs (*DELTA* and *GAA* transistors), designed to take full advantage from volume inversion, also benefit from reduced short-channel effects (V_T drop, punch-through, DIBL, hot-carrier injection, etc.), and are therefore very attractive, if not unique, devices for down-scaling below 30-nm gate length.

Defect Coupling

In fully depleted MOSFETs, carriers flowing at one interface may sense the presence of defects located at the opposite interface. *Defect coupling* is observed as an apparent degradation of the front-channel properties, which is actually induced by the buried oxide damage. This unusual mechanism is notorious after back interface degradation via radiation or hot-carrier injection (see also Fig. 4.7 in Section 4.7).

4.6 Partially Depleted SOI Transistors

In partially depleted SOI MOSFETs, the depletion charge controlled by one or both gates does not extend from an interface to the other. A neutral region subsists and, therefore, the interface coupling effects are disabled. When the body is grounded (via independent body contacts or body-source ties), partially depleted SOI transistors behave very much like bulk–Si MOSFETs and most of the standard $I_D(V_G, V_D)$ equations and design concepts apply. If body contacts are not supplied, so-called *floating-body* effects arise, leading to detrimental consequences, which will be explained next.

The *kink* effect is due to majority carriers, generated by impact ionization, that collect in the transistor body. The body potential is raised, which reduces the threshold voltage. This feedback gives rise to extra drain current (kink) in $I_D(V_D)$ characteristics (Fig. 4.5(a)), which is annoying in analog circuits.

In weak inversion and for high drain bias, a similar positive feedback (increased inversion charge → more impact ionization → body charging → threshold voltage lowering) is responsible for negative resistance regions, hysteresis in log $I_D(V_G)$ curves, and eventually latch (loss of gate control, Fig. 4.5(b)).

The floating body may also induce transient effects. A drain current *overshoot* is observed when the gate is turned on (Fig. 4.5(c)). Majority carriers are expelled from the depletion region and collect in

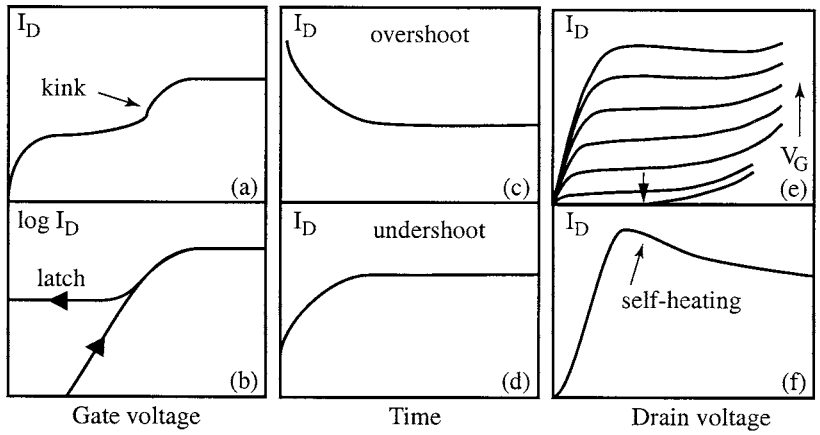


FIGURE 4.5 Parasitic effects in partially depleted SOI MOSFETs: (a) kink in $I_D(V_D)$ curves, (b) latch in $I_D(V_G)$ curves, (c) drain current overshoot, (d) current undershoot, (e) premature breakdown, and (f) self-heating.

the neutral body, increasing the potential. Equilibrium is reached through electron-hole recombination, which eliminates the excess majority carriers, making the drain current decrease gradually with time. A reciprocal *undershoot* occurs when the gate is switched from strong to weak inversion: the current now increases with time (Fig. 4.5(d)) as the majority carrier generation allows the depletion depth to shrink gradually. In short-channel MOSFETs, the transient times are dramatically reduced because of the additional contribution of source and drain junctions to establish equilibrium.

An obvious solution to alleviate floating-body effects is to sacrifice chip space for designing body contacts. The problem is that, in ultra-thin films with large sheet resistance, the body contacts are far from ideal. Their intrinsic resistance does not allow the body to be perfectly grounded and may generate additional noise. A floating body is then preferable to a poor body contact.

An exciting partially depleted device is the *dynamic-threshold* DT-MOS transistor. It is simply configured by interconnecting the gate and the body. As the gate voltage increases in weak inversion, the simultaneous raise in body potential makes the threshold voltage decrease. DT-MOSFETs achieve perfect gate-charge coupling, maximum subthreshold slope, and enhanced current, which are attractive features for LV/LP circuits.

4.7 Short–Channel Effects

In both fully and partially depleted MOSFETs with submicron length, the source–body junction can easily be turned on. The inherent activation of the lateral bipolar transistor has favorable (extra current flow in the body) or detrimental (premature breakdown, Fig. 4.5(e)) consequences. The breakdown voltage is evaluated near threshold, where the bipolar action prevails. The breakdown voltage is especially lowered for n-channels, shorter devices, thinner films, and higher temperatures. As expected, the impact ionization rate and related floating-body effects are attenuated at high temperature. However, the bipolar gain increases dramatically with temperature and accentuates the bipolar action: lower breakdown and latch voltages.[13]

Another concern is *self-heating*, induced by the power dissipation of short-channel MOSFETs and exacerbated by the poor thermal conductivity of the surrounding SiO_2 layers. Self-heating is responsible for mobility degradation, threshold voltage shift, and negative differential conductance shown in Fig. 4.5(f). The temperature rise can exceed 100 to 150°C in SOI, which is far more than in bulk Si.[25] Electromigration may even be initiated by the resulting increase in interconnect temperature. Thin buried oxides (≤100 nm) and thicker Si films (≥100 nm) are suitable when self-heating becomes a major issue.

A familiar short-channel effect is the threshold voltage roll-off due to charge sharing between the gate and source and drain terminals. The key parameters in SOI are the doping level, film thickness, and BOX thickness.[26] Ultra-thin, fully depleted MOSFETs show improved performance in terms of both V_T roll-off and drain-induced barrier lowering (DIBL) as compared to partially depleted SOI or bulk Si transistors (Fig. 4.6(a)).[27] The worst case happens when the film thickness corresponds to the transition between full and partial depletion. An additional origin of V_T roll-off in fully depleted MOSFETs is the field penetration into the buried oxide. An obvious solution is again the use of relatively thin buried oxides.

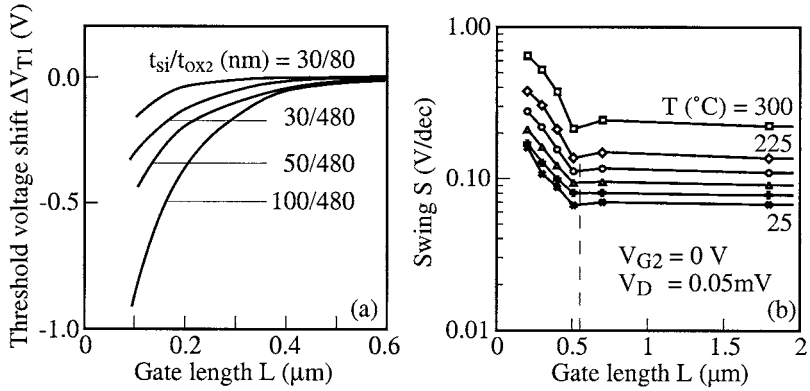


FIGURE 4.6 Typical short-channel effects in fully depleted SOI MOSFETs: (a) threshold voltage roll-off for different thicknesses of film and buried oxide[26] and (b) subthreshold swing degradation below 0.2-μm channel length for various temperatures.[13]

A degradation of the subthreshold swing is observed in very short ($L \leq 0.1$–0.5 μm), fully depleted MOSFETs (Fig. 4.6(b)). Two effects are involved: (1) conventional charge sharing, and (2) a surprising non-uniform coupling effect. We have seen that the subthreshold swing is minimum for depletion and increases for inversion at the back interface. In very short transistors, the lateral profile of the back interface potential can be highly inhomogeneous: from depletion in the middle of the channel to weak inversion near the channel ends, due to the proximity of source and drain regions. This localized weak inversion region explains the degradation of the swing.[13]

The transconductance is obviously improved in deep submicron transistors. Velocity saturation occurs as in bulk silicon. The main short-channel limitation of the transconductance comes from series resistance effects.

The lifetime of submicron MOSFETs is affected by hot-carrier injection into the gate oxide(s). The degradation mechanisms are more complex in SOI than in bulk-Si, due to the presence of two oxides, two channels, and related coupling mechanisms. For example, in Fig. 4.7, the front-channel threshold voltage is monitored during back channel stress. The shift ΔV_{T_1}, measured for $V_{G_2} = 0$ (depleted back interface), would imply that many defects are being generated at the front interface. Such a conclusion is totally negated by measurements performed with $V_{G_2} = -40V$: the influence of buried-oxide defects is now masked by the accumulation layer and indeed the apparent front-interface damage disappears ($\Delta V_{T_1} \cong 0$).[28]

In n-channels, the defects are created at the interface where the electrons flow; exceptionally, injection into the opposite interface may arise when the transistor is biased in the breakdown region. Although the device lifetime is relatively similar in bulk Si and SOI, the influence of stressing bias is different: SOI MOSFETs degrade less than bulk Si MOSFETs for $V_G \cong V_D/2$ (i.e., for maximum substrate current) and more for $V_G \cong V_T$ (i.e., enhanced hole injection). Device aging is accelerated by accumulating the back interface.[28]

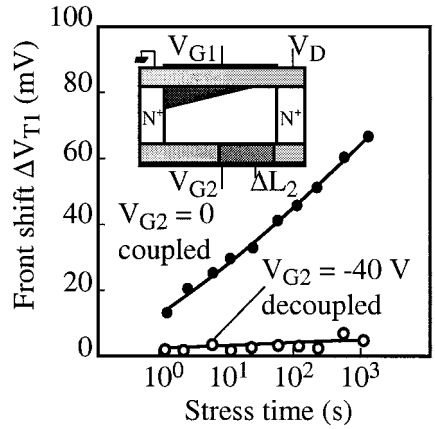


FIGURE 4.7 Front-channel threshold voltage shift during back-channel stress in a SIMOX MOSFET.[28] The apparent degradation of the gate oxide disappears when the stress-induced defects in the buried oxide are masked by interface accumulation ($V_{G_2} = -40$ V).

In p-channels, the key mechanism involves the electrons generated by front-channel impact ionization, which become trapped into the buried oxide. An apparent degradation of the front interface again occurs via coupling.[28]

4.8 SOI Challenges

SOI stands already as a pretty mature technology. However, there are still serious challenges in various domains — fundamental and device physics, technology, device modeling, and circuit design — before SOI will become fully competitive in the commercial market.

The minimum dimensions thus far achieved for SOI MOSFETs are: 70 nm length, 10 nm width (quantum wires), and 1 to 2 nm thickness.[21] When these features will be cumulative in a single transistor, the body volume ($\leq 10^{-18}$ cm^3!) will contain 10^4–10^5 silicon atoms and 0 to 1 defects. The body doping (10^{17}–10^{18} cm^{-3}) will be provided by a unique impurity, whose location may become important. Moreover, quantum transport phenomena are already being observed in ultra-thin SOI transistors. It is clear that new physical concepts, ideas, and modeling tools will be needed to account for minimum-size mechanisms in order to take advantage of them.

As far as the technology is concerned, a primary challenge is the mass-production of SOI wafers with large diameter (≥12 in.), low defect content, and reasonable cost (2 to 3 times higher than for bulk-Si wafers). The thickness uniformity of the silicon layer is especially important for fully depleted MOSFETs because it governs the fluctuations of the threshold voltage. It is predictable that several SOI technologies will not survive, except for special niches.

There is a demand for appropriate characterization techniques, either imported from other semiconductors or entirely conceived for SOI.[1] Such a pure SOI technique is the pseudo-MOS transistor (Ψ–MOSFET).[29] Ironically, it behaves very much like the MOS device that Shockley attempted to demonstrate 50 years ago but, at that time, he didn't have the chance to know about SOI. The inset of Figure 4.8 shows that the Si substrate is biased as a gate and induces a conduction channel (inversion or accumulation) at the film–oxide interface. Source and drain probes are used to measure $I_D(V_G)$ characteristics. The Ψ–MOSFET does not require any processing, hence valuable information is directly available: quality of the film, interface and oxide, electron/hole mobilities, and lifetime.

Full CMOS processing must address typical SOI requirements such as the series resistance reduction in ultra-thin MOSFETs (via local film oxidation, elevated source and drain structures, etc.), the lowering

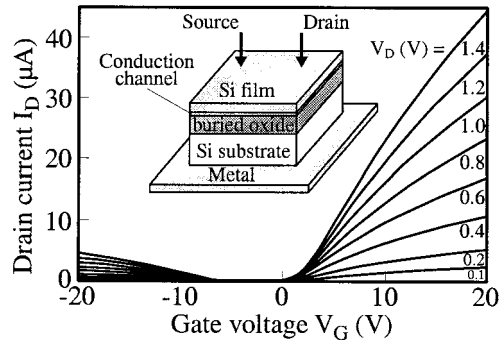


FIGURE 4.8 Pseudo-MOSFET transistor and $I_D(V_G)$ characteristics in SOI.

of the source–body barrier by source engineering (silicidation, Si–Ge, etc.), the control of the parasitic bipolar transistor, and the limitation of self-heating effects. It is now clear that the best of SOI is certainly not achievable by simply using a very good bulk-Si technology. For example, double-gate SOI MOSFETs deserve special processing and design.

According to process engineers and circuit designers, partially depleted SOI MOSFETs are more user friendly as they maintain the flavor of bulk-Si technology. On the other hand, fully depleted transistors have superior capability; they need to be domesticated in order to become more tolerant to short-channel effects. A possible solution, which requires further investigation, is the incorporation of a ground plane underneath the buried oxide.

Advanced modeling is required for correct transcription of the transistor behavior, including the transient effects due to body charging and discharging, floating-body mechanisms, bipolar transistor, dual-gate operation, quantum effects, self-heating, and short-channel limitations. Based on such physical models, compact models should then be conceived for customized simulation and design.

It is obvious that SOI does need SOI-dedicated CAD libraries. This implies a substantial amount of work which, in turn, will guarantee that the advantages and peculiar constraints of SOI devices are properly accounted for. The optimum configuration of memories, microprocessors, DSP, etc., will most likely be different in SOI as compared to bulk. Not only can SOI afford to combine fully/partially depleted, low/high power, and DT-MOSFETs in a single chip, but also the basic mechanisms of operation may differ.

4.9 Conclusion

For the next millennium, SOI offers the opportunity to integrate high-performance and/or innovative devices that can push away the present frontiers of the CMOS down-scaling. SOI will play a significant role in the future of microelectronics if subsisting problems can be rapidly solved. The short-term prospects of SOI-based microelectronics will also closely depend on the penetration rate of LV/LP SOI circuits into the market. Not only does SOI offer enhanced performance, but also most of SOI disadvantages (self-heating, hot carriers, early breakdown, etc.) tend to disappear for operation at low voltage.

A key challenge is associated with industrial strategy, which must be oriented to overcome the bulk-Si monocultural barrier. Designers, process engineers, and managers are extremely busy loading the bulk-Si machine. When, eventually, they can afford to take a careful look at the assets of SOI technology, they will realize the immediate and long-term benefits offered in terms of performance and scaling extensions.

This is so because SOI is not a totally different technology — it is just a metamorphosis of silicon.

References

1. Cristoloveanu, S. and Li, S. S., *Electrical Characterization of SOI Materials and Devices*, Kluwer, Norwell, 1995.
2. Cristoloveanu, S., "Silicon films on sapphire," *Rep. Prog. Phys.*, **3**, 327, 1987.
3. Johnson, R. A., de la Houssey, P. R., Chang, C. E., Chen, P.–F., Wood, M. E., Garcia, G. A., Lagnado, I., and Asbeck, P. M., "Advanced thin-film silicon-on-sapphire technology: microwave circuit applications," *IEEE Trans. Electron Devices*, **45**, 1047, 1998.
4. Cristoloveanu, S., "A review of the electrical properties of SIMOX substrates and their impact on device performance," *J. Electrochem. Soc.*, **138**, 3131, 1991.
5. Sato, N., Ishii, S., Matsumura, S., Ito, M., Nakayama, J., and Yonehara, T., "Reduction of crystalline defects to 50/cm^2 in epitaxial layers over porous silicon for Eltran," *IEEE Int. SOI Conf.*, Stuart, FL, 1998.
6. Burel, M., "Silicon on insulator material technology," *Electronics Lett.*, **31**, 1201, 1995.
7. Chen, J., Parke, S., King, J., Assaderaghi, F., Ko, P., and Hu, C., "A high-speed SOI technology with 12 ps/18 ps gate delay operating at 5V/1.5V," *IEDM Techn. Dig.*, 35, 1992.
8. Tsuchiya, T., Ohno, T. , and Kado, Y., "Present status and potential of subquarter-micron ultra-thin-film CMOS/SIMOX technology," *SOI Technology and Devices*, Electrochem. Soc., Pennington, 401, 1994.
9. Fuse, T. et al., "A 0.5V 200MHz 1–stage 32b ALU using a body bias controlled SOI pass-gate logic," *ISSCC Techn. Digest*, 286, 1997.
10. Schepis, D. J., et al., "A 0.25 μm CMOS SOI technology and its application to 4 Mb SRAM," *IEDM Techn. Dig.*, 587, 1997.
11. Koh, Y.–H. et al., "1 Gigabit SOI DRAM with fully bulk compatible process and body-contacted SOI MOSFET structure," *IEDM Techn. Dig.*, 579, 1997.
12. Colinge, J.–P., *SOI Technology: Materials to VLSI* (2nd ed.), Kluwer, Boston, 1997.
13. Cristoloveanu, S., and Reichert, G., "Recent advances in SOI materials and device technologies for high temperature," in 1998 High Temperature Electronic Materials, Devices and Sensors Conference, *IEEE Electron Devices Soc.*, 86, 1998.
14. Hiramoto, T. , Tamba, N., Yoshida M., et al., "A 27 GHz double polysilicon bipolar technology on bonded SOI with embedded 58 μm^2 CMOS memory cell for ECL–CMOS SRAM applications," *IEDM Techn. Dig.*, 39, 1992.
15. O'Connor, J. M., Luciani, V. K., and Caviglia, A. L., "High voltage DMOS power FETs on thin SOI substrates," *IEEE Int. SOI Conf. Proc.*, 167, 1990.
16. Vogt, H., "Advantages and potential of SOI structures for smart sensors," *SOI Technology and Devices*, Electrochem. Soc., Pennington, 430, 1994.
17. Ohno, T., Matsumoto, S., and Izumi, K., "An intelligent power IC with double buried-oxide layers formed by SIMOX technology," *IEEE Trans. Electron Devices*, **40**, 2074, 1993.
18. Nishimura, T., Inoue, Y., Sugahara, K., Kusonoki, S., Kumamoto, T., Nakagawa, S., Nakaya, M., Horiba, Y., and Akasaka, Y., "Three dimensional IC for high performance image signal processor," *IEDM Dig.*, 111, 1987.
19. Hisamoto, D., Kaga, T., and Takeda, E., "Impact of the vertical SOI 'DELTA' structure on planar device technology," *IEEE Trans. Electron Dev.*, **38**, 1419, 1991.
20. Lim, H.–K. and Fossum, J. G., "Threshold voltage of thin-film silicon on insulator (SOI) MOSFETs," *IEEE Trans. Electron Dev.*, **30**, 1244, 1983.
21. Ohmura, Y., Ishiyama, T., Shoji, M., and Izumi, K., "Quantum mechanical transport characteristics in ultimately miniaturized MOSFETs/SIMOX," *SOI Technology and Devices*, Electrochem. Soc., Pennington, 199, 1996.
22. Mazhari, B., Cristoloveanu, S. , Ioannou D. E., and Caviglia, A. L., "Properties of ultra-thin wafer-bonded silicon on insulator MOSFETs," *IEEE Trans. Electron Dev.*, **ED–38**, 1289, 1991.

23. Ouisse, T., Cristoloveanu, S., and Borel, G., "Influence of series resistances and interface coupling on the transconductance of fully depleted silicon-on-insulator MOSFETs," *Solid-State Electron.*, **35**, 141, 1992.
24. Balestra, F., Cristoloveanu, S., Bénachir, M., Brini, J., and Elewa,T., "Double-gate silicon on insulator transistor with volume inversion: a new device with greatly enhanced performance," *IEEE Electron Device Lett.*, **8**, 410, 1987.
25. Su, L. T., Goodson, K. E., Antoniadis, D. A., Flik, M. I., and Chung, J. E., "Measurement and modeling of self-heating effects in SOI n-MOSFETs," *IEDM Dig.*, 111, 1992.
26. Ohmura, Y., Nakashima, S., Izumi, K., and Ishii, T., "0.1-μm-gate, ultra-thin film CMOS device using SIMOX substrate with 80-nm-thick buried oxide layer," *IEDM Techn. Dig.*, 675, 1991.
27. Balestra, F. and Cristoloveanu, S., "Special mechanisms in thin-film SOI MOSFETs," *Microelectron. Reliab.*, **37**, 1341, 1997.
28. Cristoloveanu, S., "Hot-carrier degradation mechanisms in silicon-on-insulator MOSFETs," *Microelectron. Reliab.*, **37**, 1003, 1997.
29. Cristoloveanu, S. and Williams, S., "Point contact pseudo–MOSFET for *in-situ* characterization of as-grown silicon on insulator wavers," *IEEE Electron Device Lett.*, **13**, 102, 1992.

Cressler, J.D. “SiGe Technology”
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

5

SiGe Technology

John D. Cressler
Auburn University

5.1 Introduction

The concept of bandgap engineering has been used for many years in compound semiconductors such as gallium arsenide (GaAs) and indium phosphide (InP) to realize a host of novel electronic devices. A bandgap-engineered transistor is compositionally altered in a manner that improves a specific device metric (e.g., speed). A transistor designer might choose, for instance, to make a bipolar transistor that has a GaAs base and collector region, but which also has a AlGaAs emitter. Such a device has electrical properties that are inherently superior to what could be achieved using a single semiconductor. In addition to simply combining two different materials (e.g., AlGaAs and GaAs), bandgap engineering often involves compositional grading of materials within a device. For instance, one might choose to vary the Al content in an AlGaAs/GaAs transistor from a mole fraction of 0.4 to 0.6 across a given distance within the emitter.

Device designers have long sought to combine the bandgap engineering techniques enjoyed in compound semiconductors technologies with the fabrication maturity, high yield, and hence low cost associated with conventional silicon (Si) integrated circuit manufacturing. Epitaxial silicon-germanium (SiGe) alloys offer considerable potential for realizing viable bandgap-engineered transistors in the Si material system. This is exciting because it potentially allows Si electronic devices to achieve performance levels that were once thought impossible, and thus dramatically extends the number of high-performance circuit applications that can be addressed using Si technology. This chapter reviews the recent progress in both SiGe heterojunction bipolar transistor (HBT) technology and SiGe field effect transistor (FET) technology.

5.2 SiGe Strained Layer Epitaxy

Si and Ge, being chemically compatible elements, can be intermixed to form a stable alloy. Unfortunately, however, the lattice constant of Si is about 4.2% smaller than that of Ge. The difficulties associated with realizing viable SiGe bandgap-engineered transistors can be traced to the problems encountered in growing high-quality, defect-free epitaxial SiGe alloys in the presence of this lattice mismatch. For electronic applications, it is essential to obtain a SiGe film that adopts the same lattice constant as the underlying Si substrate with perfect alignment across the growth interface. In this case, the resultant SiGe alloy is under compressive strain. This strained SiGe film is thermodynamically stable only under a narrow range of conditions that depend on the film thickness and the effective strain (determined by

the Ge concentration).[1] The critical thickness below which the grown film is unconditionally stable depends reciprocally on the effective strain (Fig. 5.1). Thus, for practical electronic device applications, SiGe alloys must be thin (typically <100–200 nm) and contain only modest amounts of Ge (typically <20–30%). It is essential for electronic devices that the SiGe films remain thermodynamically stable so that conventional Si fabrication techniques such as high-temperature annealing, oxidation, and ion implantation can be employed without generating defects.

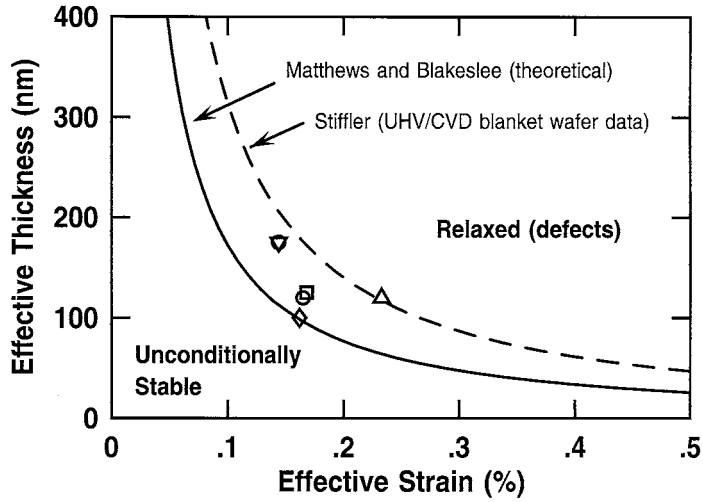


FIGURE 5.1 Effective thickness versus effective strain for SiGe strained layer epitaxy. Shown are a theoretical stability curve (Matthews-Blakeslee), and an empirical curve for UHV/CVD SiGe epitaxy. The symbols represent actual films used in UHV/CVD SiGe HBTs.

From an electronic device viewpoint, the property of the strained SiGe alloy that is most often exploited is the reduction in bandgap with strain and Ge content (roughly 75 meV per 10% Ge).[2] This band offset appears mostly in the valence band, which is particularly useful for realizing n-p-n bipolar transistors and p-channel FETs.[3] While these band offsets are modest compared to those that can be achieved in III–V semiconductors, the Ge content can be compositionally graded to produce local electric fields that aid carrier transport. For instance, in a SiGe HBT, the Ge content might be graded from 0 to 15% across distances as short as 50 to 60 nm, producing built-in drift fields as large as 15 to 20 kV · cm^{-1}. Such fields can rapidly accelerate the carriers to scattering-limited velocity (1×10^7 cm · s^{-1}), thereby improving the transistor frequency response. Another benefit of using SiGe strained layers is the enhancement in carrier mobility. This advantage will be exploited in SiGe channel FETs, as discussed below.

Epitaxial SiGe strained layers on Si substrates can be successfully grown today by a number of different techniques, including molecular beam epitaxy (MBE), ultra-high vacuum/chemical vapor deposition (UHV/CVD), rapid-thermal CVD (RTCVD), atmospheric pressure CVD (APCVD), and reduced pressure CVD (RPCVD). Each growth technique has advantages and disadvantages, but it is generally agreed that UHV/CVD,[4] has a number of appealing features for the commercialization of SiGe integrated circuits. The features of UHV/CVD that make it particularly suitable for manufacturing include: (1) batch processing on up to 16 wafers simultaneously, (2) excellent doping and thickness control on large (e.g., 200 mm) wafers, (3) very low background oxygen and carbon concentrations, (4) compatibility with patterned wafers and hence conventional Si bipolar and CMOS fabrication techniques, and (5) the ability to compositionally grade the Ge content in a controllable manner across short distances. The experimental results presented in this chapter are based on the UHV/CVD growth technique as practiced at IBM Corporation, and are representative of the state-of-the-art in SiGe technology.

5.3 The SiGe Heterojunction Bipolar Transistor (HBT)

The SiGe HBT is by far the most mature Si-based bandgap-engineered electronic device.[5] The first SiGe HBT was reported in 1987,[6,7] and began commercial production in 1998. Significant steps along the path to manufacturing included the first demonstration of high-frequency (75 GHz) operation of a SiGe HBT in a non-self-aligned structure in early 1990.[8] This result garnered much attention worldwide because the performance of the SiGe HBT was roughly twice what a state-of-the-art Si BJT could achieve (Fig. 5.2). The first fully integrated, self-aligned SiGe HBT technology was demonstrated later in 1990,[9] the first fully integrated 0.5-μm SiGe BiCMOS technology (SiGe HBT + Si CMOS) in 1992,[10] and SiGe HBTs with frequency response above 100 GHz in 1993 and 1994.[11,12] A number of research laboratories around the world have demonstrated robust SiGe HBT technologies, including IBM,[13–17] Daimler-Benz/TEMIC,[18–21] NEC,[22–24] and Siemens.[25] More recent work has begun to focus on practical SiGe HBT circuits for radio-frequency (RF) and microwave applications.[26–28]

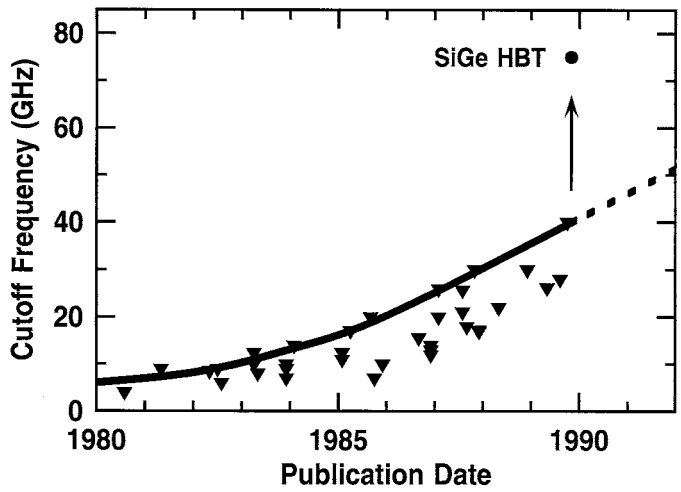


FIGURE 5.2 Transistor cutoff frequency as a function of publication date comparing Si BJT performance and the first SiGe HBT result.

Because the intent in SiGe technology is to combine bandgap engineering with conventional Si fabrication techniques, most SiGe HBT technologies appear very similar to conventional Si bipolar technologies. A typical device cross-section is shown in Fig. 5.3. This SiGe HBT has a planar, self-aligned structure with a conventional polysilicon emitter contact, silicided extrinsic base, and deep- and shallow-trench isolation. A 3-5 level, chemical-mechanical-polishing (CMP) planarized, W-stud, AlCu CMOS metalization scheme is used. The extrinsic resistive and capacitive parasitics are intentionally minimized to improve the maximum oscillation frequency (f_{max}) of the transistor. Observe that the Ge is introduced only into the thin base region of the transistor, and is deposited with a thickness and Ge content that ensures the film is thermodynamically stable (examples of real SiGe HBT profiles are shown as symbols in stability space in Fig. 5.1). The *in situ* boron-doped, graded SiGe base is deposited across the entire wafer using the ultra-high vacuum/chemical vapor deposition (UHV/CVD) technique. In areas that are not covered by oxide, the UHV/CVD film consisting of an intrinsic-Si/strained boron-doped SiGe/intrinsic-Si stack is deposited as a perfect single-crystal layer on the Si substrate. Over the oxide, the deposited layer is polycrystalline (poly) and will serve either as the extrinsic base contact of the SiGe HBT, the poly-on-oxide resistor, or the gate electrode of the Si CMOS devices. The metallurgical base and single-crystal emitter widths range from 75 to 90 nm and 25 to 35 nm, respectively. A masked phosphorous implant is used to tailor the intrinsic collector profile for optimum frequency response at high current densities.

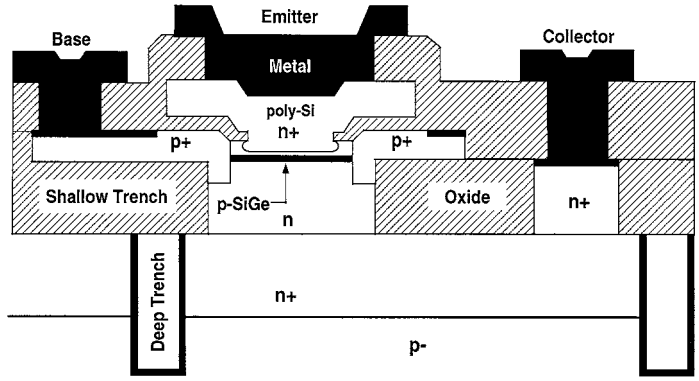


FIGURE 5.3 Schematic cross-section of a self-aligned UHV/CVD SiGe HBT.

A conventional deep-trench/shallow-trench bipolar isolation scheme is used, as well as a conventional arsenic-doped polysilicon emitter layer. This approach ensures that the SiGe HBT is compatible with commonly used (low-cost) bipolar/CMOS fabrication processes. A typical doping profile measured by secondary ion mass spectroscopy (SIMS) of the resultant SiGe HBT is shown in Fig. 5.4.

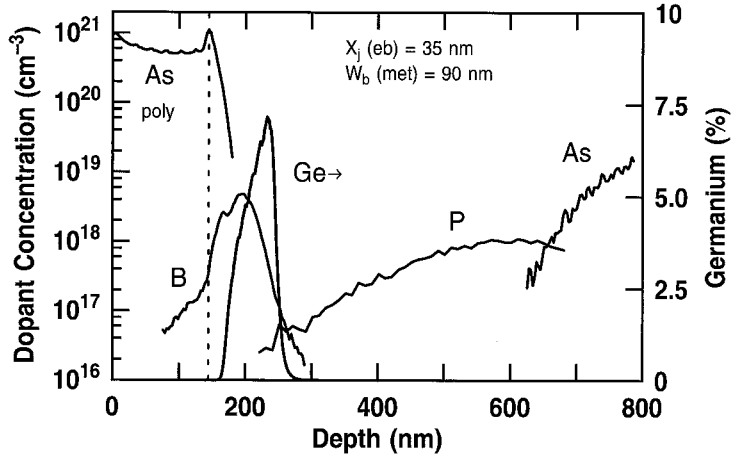


FIGURE 5.4 Secondary ion mass spectroscopy (SIMS) doping profile of a graded-base SiGe HBT.

The smaller base bandgap of the SiGe HBT can be exploited in three major ways, and is best illustrated by examining an energy band diagram comparing a SiGe HBT with a Si BJT (Fig. 5.5). First, note the reduction in base bandgap at the emitter–base junction. The reduction in the potential barrier at the emitter–base junction in a SiGe HBT will exponentially increase the collector current density and, hence, current gain ($\beta = J_C/J_B$) for a given bias voltage compared to a comparably designed Si BJT. Compared to a Si BJT of identical doping profile, this enhancement in current gain is given by:

$$\frac{J_{C,SiGe}}{J_{C,Si}} = \frac{\beta_{SiGe}}{\beta_{Si}} = \gamma\eta \frac{\Delta E_{g,Ge}(grade)/kT \; e^{\Delta E_{g,Ge}(0)/kT}}{1 - e^{-\Delta E_{g,Ge}(grade)/kT}} \tag{5.1}$$

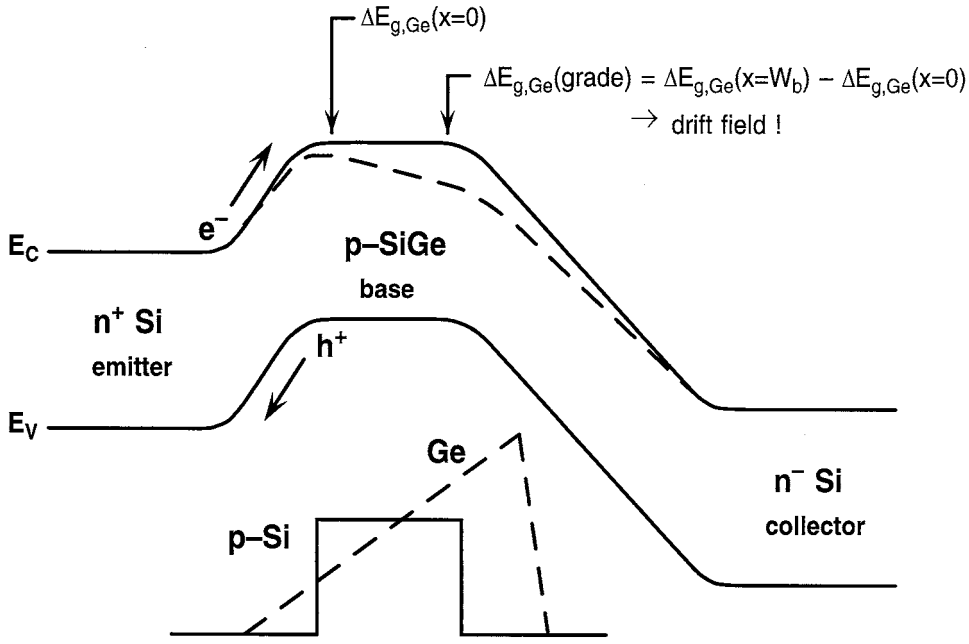


FIGURE 5.5 Energy band diagram for both a Si BJT and a graded-base SiGe HBT.

where $\gamma = N_C N_V$ (SiGe)/$N_C N_V$(Si) is the ratio of the density-of-states product between SiGe and Si, and $\eta = D_{nb}$(SiGe)/D_{nb}(Si) accounts for the differences between the electron and hole mobilities in the base. The position dependence of the band offset with respect to Si is conveniently expressed as a bandgap grading term ($\Delta E_{g,Ge}$(grade) = $\Delta E_{g,Ge}$(Wb) – $\Delta E_{g,Ge}$(0)). As can be seen in Fig. 5.6, which compares the measued Gummel characteristics for two identically constructed SiGe HBTs and Si BJTs, these theoretical expectations are clearly borne out in practice.

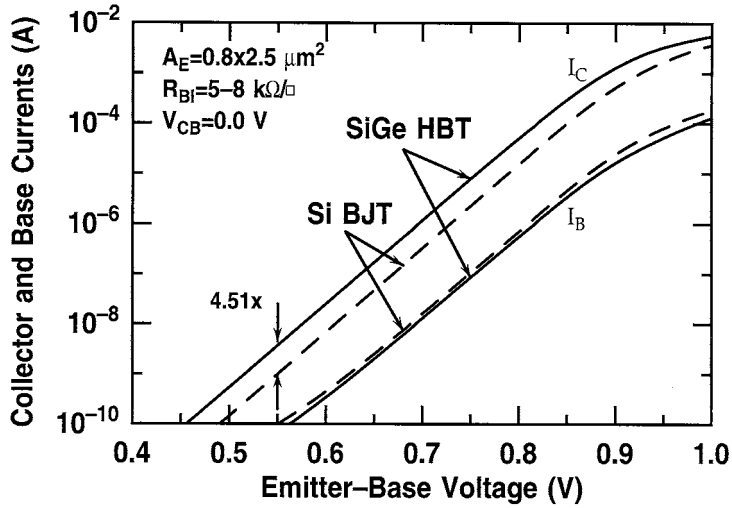


FIGURE 5.6 Measured current/voltage characteristics of both SiGe HBT and Si BJT with a comparable doping profile.

Secondly, if the Ge content is graded across the base region of the transistor, the conduction band edge becomes position dependent (refer to Fig. 5.5), inducing an electric field in the base that accelerates the injected electrons. The base transit time is thereby shortened and the frequency response of the transistor is improved according to:

$$\frac{\tau_{b,\,SiGe}}{\tau_{b,\,Si}} = \frac{f_{T,\,Si}}{f_{T,\,SiGe}} = \frac{2}{\eta}\left(\frac{kT}{\Delta E_{g,\,Ge}(grade)}\right)\left[1 - \frac{1 - e^{-\Delta E_{g,\,Ge}(grade)/kT}}{\Delta E_{g,\,Ge}(grade)/kT}\right] \tag{5.2}$$

Figure 5.7 compares the measured unity gain cutoff frequency (f_T) of a SiGe HBT and a comparably constructed Si BJT, and shows that an improvement in peak f_T of 1.7X can be obtained with relatively modest Ge profile grading (0–7.5% in this case). More recent studies demonstrate that peak cutoff frequencies in excess of 100 GHz can be obtained using more aggressively designed (although still stable) Ge profiles[12] (see Fig. 5.8).

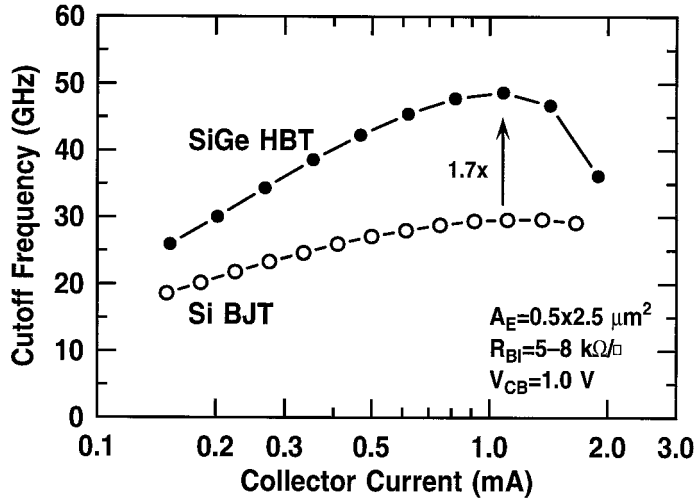


FIGURE 5.7 Measured cutoff frequency as a function of bias current for both SiGe HBT and Si BJT with a comparable doping profile.

The final advantage of using a graded Ge profile in a SiGe HBT is the improvement in the output conductance of the transistor, an important analog design metric. For a graded-base SiGe HBT, the Early voltage (a measure of output conductance) increases exponentially compared to a Si BJT of comparable doping, according to:

$$\frac{V_{A,\,SiGe}}{V_{A,\,Si}} = e^{\Delta E_{g,\,Ge}(grade)/kT}\left[\frac{1 - e^{-\Delta E_{g,\,Ge}(grade)/kT}}{(\Delta E_{g,\,Ge}(grade)/kT)}\right] \tag{5.3}$$

In essence, the position dependence of the bandgap in the graded-base SiGe HBT weights the base profile toward the collector region, making it harder to deplete the base with collector-base bias, hence yielding a larger Early voltage. A transistor with a high Early voltage has a very flat common-emitter output characteristic, and hence low output conductance. For the device shown in Fig. 5.6, the Early voltage increases from 18 V in the Si BJT to 53 V in the SiGe HBT, a 3X improvement.

SiGe HBTs have been successfully integrated with conventional high-performance Si CMOS to realize a SiGe BiCMOS technology.[15] The SiGe HBT and ac performance in this SiGe BiCMOS technology is

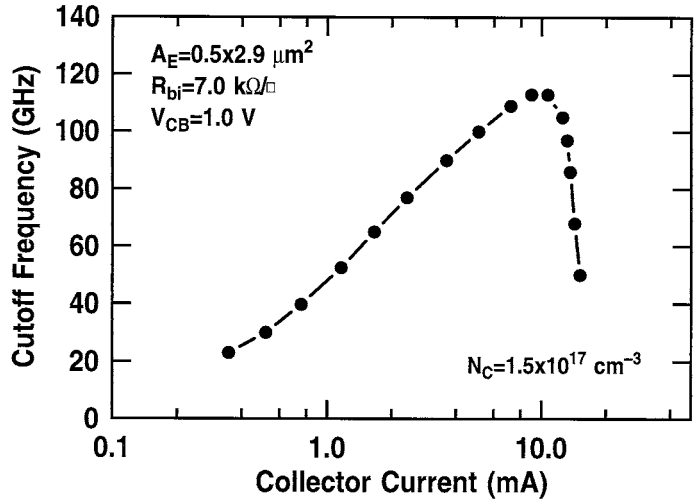


FIGURE 5.8 Cutoff frequency as a function of bias current for an aggressively designed UHV/CVD SiGe HBT.

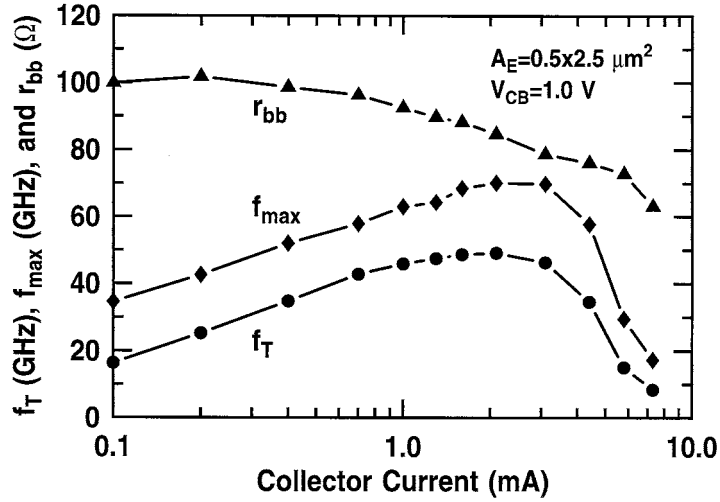


FIGURE 5.9 Cutoff frequency, maximum oscillation frequency, and small-signal base resistance for a small-geometry, state-of-the-art SiGe HBT.

shown in Fig. 5.9, and is not compromised by the addition of the CMOS devices. Table 5.1 shows the suite of resultant elements in this SiGe BiCMOS technology. Two SiGe HBTs are available, one with a reduced collector implant and hence higher BV_{CEO} (5.3 V vs. 3.3 V) that is suitable for RF power applications. Table 5.2 gives the typical SiGe HBT device parameters at 300K.

Bandgap-engineered SiGe HBTs have other attractive features that make them ideal candidates for certain circuit applications. For instance, Si BJT technology is well known to have superior low-frequency noise properties compared to compound semiconductor technologies. Low-frequency noise is often a major limitation for RF and microwave systems because it directly limits the spectral purity of the transmitted signal. Recent work suggests that SiGe HBTs have low-frequency properties as good as or better than Si BJTs, superior to that obtained in AlGaAs/GaAs HBTs and Si CMOS (Fig. 5.10).[29,30] The

TABLE 5.1 Elements in IBM's SiGe BiCMOS Technology

Element	Characteristics
Standard SiGe HBT	47 GHz f_T at BV_{CEO} = 3.3V
High-breakdown SiGe HBT	28 GHz f_T at BV_{CEO} = 5.3V
Si CMOS	0.36 mm L_{eff} for 2.5 V V_{DD}
Gated lateral pnp	1.0 GHz f_T
Polysilicon resistor	342 Ω/□
Ion-implanted resistor	1,600 Ω/□
Thin-oxide decoupling capacitor	1.52 fF/μm²
MIM precision capacitor	0.70 fF/μm²
Inductor (6-turn)	10 nH with Q = 10 at 1.0 GHz
Schottky barrier diode	213 mV at 100 μA
p-i-n diode	790 mV at 100 μA
Varactor diode	810 mV at 100 μA
ESD diode	1.2 kV

TABLE 5.2 Typical Parameters for a SiGe HBT with $A_E = 0.5 \times 2.5$ μm². All ac Parameters Were Measured at V_{CB} = 1.0V and f_{max} was Extracted Using MAG.

Parameter	Standard SiGe HBT	High-BVCEO SiGe HBT
Peak β	113	97
V_A (V)	61	132
βV_A (V)	6,893	12,804
Peak f_T (GHz)	48	28
rbb at peak f_T (Ω)	80	N/A
Peak f_{max} (GHz)	69	57
BV_{CEO} (V)	3.3	5.3
BV_{EBO} (V)	4.2	4.1
Peak $f_T \times BV_{CEO}$ (GHz V)	158	143

broadband (RF) noise in SiGe HBTs is competitive with GaAs MESFET technology and superior to Si BJTs. In addition, SiGe HBTs have recently been shown to be very robust with respect to ionizing radiation, an important feature for space-based electronic systems.[31,32] Finally, cooling enhances all of the advantages of a SiGe HBT. In striking contrast to a Si BJT, which strongly degrades with cooling, the current gain, Early voltage, cutoff frequency, and maximum oscillation frequency (f_{max}) all improve significantly as the temperature drops.[33–35] This means that the SiGe HBT is well-suited for operation in the cryogenic environment (e.g., 77K), historically the exclusive domain of Si CMOS and III–V compound semiconductor technologies. Cryogenic electronics is in growing use in both military and commercial applications such as space-based satellites, high-sensitivity instrumentation, high-T_C superconductors, and future cryogenic computers.

5.4 The SiGe Heterojunction Field Effect Transistor (HFET)

The effective carrier mobility (μ_{eff}) is the fundamental parameter that limits the speed of field effect transistors (FETs), and SiGe bandgap engineering can be used in two principal ways to significantly improve the mobility and, hence, the speed of the device. First, the valence band offset associated with SiGe can be used to spatially confine carriers such that they are effectively removed from the Si/SiO_2 interface. The surface roughness scattering associated with the Si/SiO_2 interface degrades the mobility in a conventional MOSFET, particularly at high gate bias. If, in addition, the holes are confined to a region of the semiconductor that is intentionally left undoped, the result is a reduction in ionized impurity scattering and, hence, further increase in mobility.[2] This is exactly the approach taken in bandgap-engineered compound semiconductor FETs known as HEMTs (high electron mobility transistor). Second, the strain associated with SiGe epitaxy is known to lift the degeneracy of the light and heavy hole valence bands, resulting in a reduced hole effective mass and hence higher hole mobility.[3]

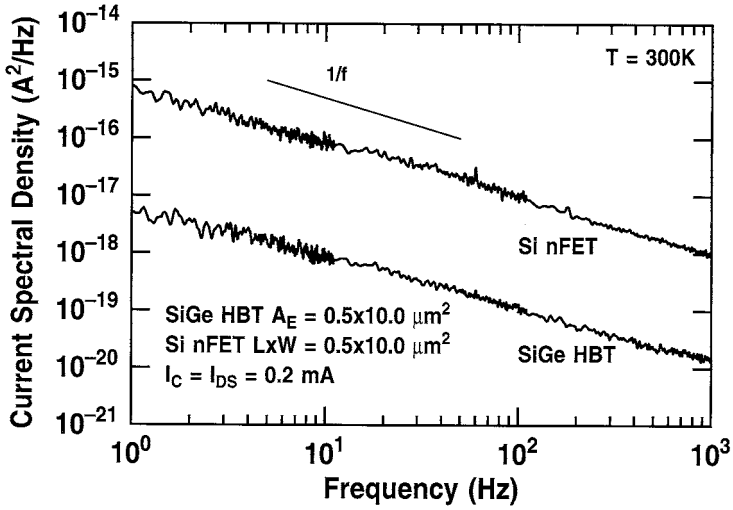


FIGURE 5.10 Low-frequency noise spectra of SiGe HBT and a conventional Si nFET of identical geometry.

Because it is the hole mobility that is improved in strained SiGe, and the valence band offset can be used to confine these holes, the p-channel FET (pFET) is the logical device to pursue with SiGe bandgap engineering. A SiGe pFET with an improved hole mobility is particularly desirable in CMOS technology because the conventional Si pFET has a mobility that is about a factor of two lower than the Si nFET mobility. This mobility asymmetry between pFET and nFET in conventional Si CMOS technology requires a doubling of the pFET gate width, and thus a serious real estate penalty, to obtain proper switching characteristics in the CMOS logic gate. If SiGe can be used to equalize the pFET and nFET mobilities, then substantial area advantages can be realized in CMOS logic gates, and tighter packing densities achieved.

It is interesting to note that despite the fact that the SiGe HBT is the most commercially mature SiGe device, the first SiGe FET (actually a HEMT structure) predates the first SiGe HBT by one year, having been first reported in 1986.[36,37] A number of different SiGe pFET designs have been successfully demonstrated[38–43] with improvements in mobility as high at 50% at 0.25-μm gate lengths.[40] Figure 5.11 shows perhaps the simplest configuration of a SiGe pFET, which consists of a SiGe hole channel buried underneath a thin Si cap layer and the conventional thermal oxide.[44,45] In this case, the entire device is fabricated on a silicon-on-sapphire substrate to improve microwave performance. Significant mobility advantage can be realized over a comparable Si pFET (Fig. 5.12).

Because a complementary circuit configuration offers many advantages from a power dissipation standpoint, the realization of n-channel SiGe devices is highly desirable. Strictly speaking, this is not possible in strained SiGe on Si substrates because only a valence band offset is available and the electrons cannot be confined as in the SiGe pFET. Fortunately, however, recent work[46–51] using strained Si on relaxed SiGe layers has proven particularly promising because it provides a conduction band offset and enhanced electron mobility compared to Si.

The fabrication of strained Si nFETs on relaxed SiGe layers is, in general, more complicated than fabricating strained SiGe pFETs in Si substrates. For the strained Si nFET, a graded SiGe buffer layer is used to reduce and confine the dislocations associated with the growth of relaxed SiGe layers.[46] Using this technique, both strained Si nFETs and strained SiGe pFETs can be jointly fabricated to form a Si-based heterojunction CMOS (HCMOS) technology. Figure 5.13 shows a schematic cross-section of such a Si/SiGe HCMOS technology,[51] and represents the state-of-the-art in the field of Si/SiGe HCMOS. Observe that the conducting channels of both transistors are grown in a single step (using UHV/CVD in this case), and electron and hole confinement occurs in the strained SiGe and strained Si for the pFET and nFET, respectively. In this technology, both the pFET and nFET are realized in a planar structure,

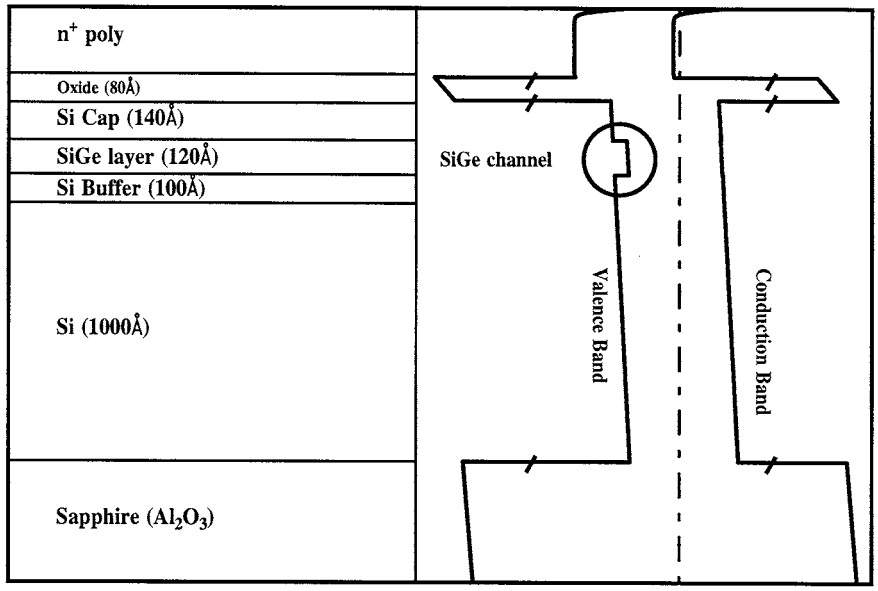


FIGURE 5.11 Schematic cross-section of SiGe pFET on silicon-on-sapphire (SOS).

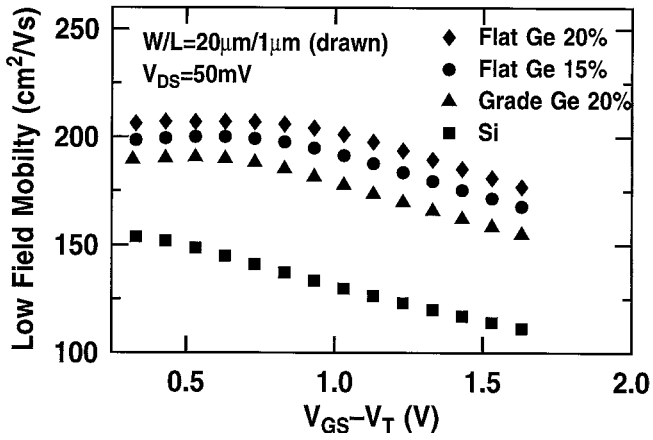


FIGURE 5.12 Effective mobility as a function of gate drive comparing SiGe pFETs on SOS and conventional Si pFETs on SOS.

and are expected to show substantial improvements in performance over conventional Si CMOS. With layer and doping profile optimization, the parasitic surface channel (which degrades mobility) can be minimized. Simulation results of anticipated circuit performance indicate that substantial improvements (4 to 6X) in power-delay product over conventional Si CMOS can be obtained using Si/SiGe HCMOS technology at 0.2-μm effective gate length.

5.5 Future Directions

The future developments in SiGe electronic devices will likely follow two paths. The first path will be toward increased integration and scaling of existing SiGe technologies. There is already a clear trend toward integrating SiGe HBTs with conventional Si CMOS to form a SiGe BiCMOS technology.[14,15]

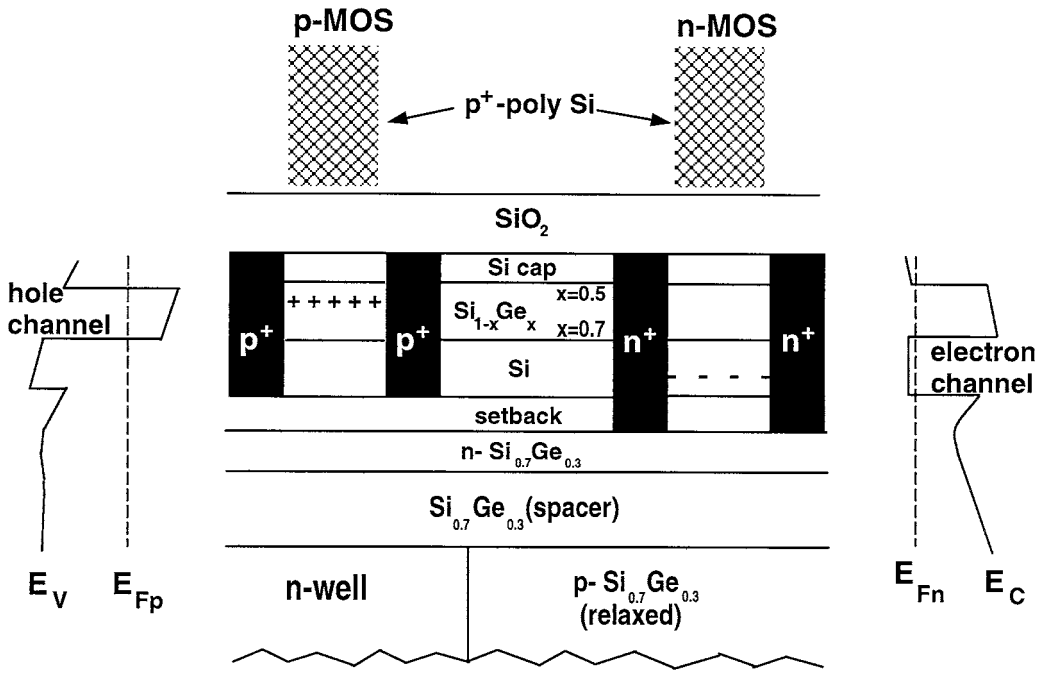


FIGURE 5.13 Schematic device cross-section of SiGe CMOS technology, consisting of a strained SiGe pFET and a strained Si nFET (after Ref. 51).

Eventually, one would like to merge SiGe HBTs, Si CMOS, Si/SiGe HCMOS, and SiGe photonic devices on the same chip to provide a complete solution for future RF and microwave electronic and optoelectronic transceivers containing precision analog functions (mixers, LNAs, VCOs, DACs, ADCs, power amps, etc.), digital signal processing and computing functions (traditionally done in CMOS), as well as integrated photonic detectors and possibly transmitters.

The second path in SiGe electronic devices will be to pursue new Si-based material systems that offer the promise of lattice-matching to Si substrates. The intent here is to remove the stability constraints that SiGe device designers currently face, and that limit the useful range of Ge content in practical SiGe devices. The most promising of these new materials is silicon-germanium-carbon (SiGeC). The lattice constant of C is larger than Si and thus can be used reduce the strain in a SiGe film. Theoretically, it would only take 1% C to lattice match a 9% Ge film to a Si substrate, 2% C for 18% Ge, etc. While research is just beginning on the growth of device-quality SiGeC films, and the properties of the resultant films has not been firmly established, the SiGeC material system clearly offers exciting possibilities for the future evolution of SiGe technology. In addition, it has recently been shown that low concentrations of C can serve to dramatically reduce boron diffusion in conventional SiGe HBTs.[52] This has the potential to allow much more aggressive SiGe HBT profiles to be realized with stable SiGe strained layers. More research is required to quantify the impact of C on the device electrical characteristics, although initial studies appear promising.

Acknowledgments

The author would like to thank D.L. Harame, B.S. Meyerson, S. Subbanna, D. Ahlgren, M. Gilbert, K. Ismail, and the members of the SiGe team at IBM Corporation, as well as the past and present members of the Auburn University SiGe research group (A. Joseph, D. Richey, L. Vempati, S. Mathew, J. Roldán, G. Bradford, G. Niu, B. Ansley, K. Shivaram, G. Banerjee, S. Zhang, S. Salmon, and

U. Gogineni) for their contributions to this work. The support of the Alabama Microelectronics Science and Technology Center, ONR, DARPA, NRL, U.S. Army SSDC, Navy NCCOSC, Navy Crane, and MRC are gratefully acknowledged.

References

1. Matthews, J. W. and Blakeslee, A. E., *Journal of Crystal Growth*, **27**, 118,1974.
2. People, R., *IEEE Journal of Quantum Electronics*, **22**, 1696, 1986.
3. Meyerson, B. S., *Proceedings of the IEEE*, **80**, 1592, 1992.
4. Meyerson, B. S., *Applied Physics Letters*, **48**, 797, 1986.
5. Cressler, J. D., *IEEE Spectrum*, 49, March 1995.
6. Iyer, S. S., Patton, G. L., Delage, S. L., Tiwari, S., and Stork, J. M. C., in *Technical Digest of the IEEE International Electron Device Meeting*, 874, 1987.
7. Patton, G. L., Iyer, S. S., Delage, S. L., Tiwari, S., and Stork, J. M. C., *IEEE Electron Device Letters*, **9**, 165, 1988.
8. Patton, G. L., Comfort, J. H., Meyerson, B. S., Crabbé, E. F., Scilla, G. J., de Fresart, E., Stork, J. M. C., Sun, J. Y.-C., Harame, D. L., and Burghartz, J., *IEEE Electron Device Letters*, **11**, 171, 1990.
9. Comfort, J. H., Patton, G. L., Cressler, J. D., Lee, W., Crabbé, E. F., Meyerson, B. S., Sun, J. Y.-C., Stork, J. M. C., Lu, P.-F., Burghartz, J. N., Warnock, J., Scilla, G., Toh, K.-Y., D'Agostino, M., Stanis, C., and Jenkins, K., in *Technical Digest of the International Electron Device Meeting*, 21, 1990.
10. Harame, D. L., Crabbé, E. F., Cressler, J. D., Comfort, J. H., Sun, J. Y.-C., Stiffler, S. R., Kobeda, E., Gilbert, M., Malinowski, J., Dally, A. J., Ratanaphanyarat, S., Saccamango, M. J., Rausch, W., Cotte, J., Chu, C., and Stork, J. M. C., in *Technical Digest of the International Electron Device Meeting*, 19, 1992.
11. Schuppen, A., Gruhle, A., Erben, U., Kibbel, H., and Konig, U., in *Technical Digest of the International Electron Device Meeting*, 377, 1994.
12. Crabbé, E. F., Meyerson, B. S., Stork, J. M. C., and Harame, D. L., in *Technical Digest of the International Electron Device Meeting*, 83, 1993.
13. Harame, D. L., Stork, J. M. C., Meyerson, B. S., Hsu, K. Y.-J., Cotte, J., Jenkins, K. A., Cressler, J. D., Restle, P., Crabbé, E. F., Subbanna, S., Tice, T. E., Scharf, B. W., and Yasaitis, J. A., in *Technical Digest of the International Electron Device Meeting*, 71, 1993.
14. Harame, D. L., Schonenberg, K., Gilbert, M., Nguyen-Ngoc, D., Malinowski, J., Jeng, S.-J., Meyerson, B. S., Cressler, J. D., Groves, R., Berg, G., Tallman, K., Stein, K., Hueckel, G., Kermarrec, C., Tice, T., Fitzgibbons, G., Walter, K., Colavito, D., Houghton, T., Greco, N., Kebede, T., Cunningham, B., Subbanna, S., Comfort, J. H., and Crabbé, E. F., in *Technical Digest of the International Electron Device Meeting*, 437, 1994.
15. Nguyen-Ngoc, D., Harame, D. L., Malinowski, J. C., Jeng, S.-J., Schonenberg, K. T., Gilbert, M. M., Berg, G., Wu, S., Soyuer, M., Tallman, K. A., Stein, K. J., Groves, R. A., Subbanna, S., Colavito, D., Sunderland, D. A., and Meyerson, B. S., in *Proceedings of the Bipolar/BiCMOS Circuits and Technology Meeting*, 89, 1995.
16. Harame, D. L., Comfort, J. H., Cressler, J. D., Crabbé, E. F., Sun, J. Y.-C., Meyerson, B. S., and Tice, T., *IEEE Transactions on Electron Devices*, 42, 455, 1995.
17. Harame, D. L., Comfort, J. H., Cressler, J. D., Crabbé, E. F., Sun, J. Y.-C., Meyerson, B. S., and Tice, T., *IEEE Transactions on Electron Devices*, **42**, 469, 1995.
18. Gruhle, A., *Journal of Vacuum Science and Technology B*, **11**, 1186, 1993.
19. Gruhle, A., Kibbel, H., Konig, U., Erben, U., and Kasper, E., *IEEE Electron Device Letters*, **13**, 206, 1992.
20. Schuppen, A., in *Technical Digest of the International Electron Device Meeting*, 743, 1995.
21. Schuppen, A., Dietrich, H., Gerlach, S., Arndt, J., Seiler, U., Gotzfried, A., Erben, U., and Schumacher, H., in *Proceedings of the Bipolar/BiCMOS Circuits and Technology Meeting*, 130, 1996.
22. Sato, F., Takemura, H., Tashiro, T., Hirayama, T., Hiroi, M., Koyama, K., and Nakamae, M., in *Technical Digest of the International Electron Device Meeting*, 607, 1992.

23. Sato, F., Hashimoto, T., Tatsumi, T., and Tasshiro, T., *IEEE Transactions on Electron Devices*, **42**, 483, 1995.
24. Sato, F., Tezuka, H., Soda, M., Hashimoto, T., Suzaki, T., Tatsumi, T., Morikawa, T., and Tashiro, T., in *Proceedings of the Bipolar/BiCMOS Circuits and Technology Meeting*, 158, 1995.
25. Meister, T. F., Schafer, H., Franosch, M., Molzer, W., Aufinger, K., Scheler, U., Walz, C., Stolz, M., Boguth, S., and Bock, J., in *Technical Digest of the International Electron Device Meeting*, 739, 1995.
26. Soyuer, M., Burghartz, J., Ainspan, H., Jenkins, K. A., Xiao, P., Shahani, A., Dolan, M., and Harame, D. L., in *Proceedings of the Bipolar/BiCMOS Circuits and Technology Meeting*, 169, 1996.
27. Schumacher, H., Erben, U., Gruhle, A., Kibbelk, H., and Konig, U., in *Proceedings of the Bipolar/BiCMOS Circuits and Technology Meeting*, 186, 1995.
28. Sato, F., Hashimoto, T., Tatsumi, T., Soda, M., Tezuka, H., Suzaki, T., and Tashiro, T., in *Proceedings of the Bipolar/BiCMOS Circuits and Technology Meeting*, 82, 1995.
29. Cressler, J. D., Vempati, L., Babcock, J. A., Jaeger, R. C., and Harame, D. L., *IEEE Electron Device Letters*, **17**, 13, 1996.
30. Vempati, L., Cressler, J. D., Babcock, J. A., Jaeger, R. C., and Harame, D. L., *IEEE Journal of Solid-State Circuits*, **31**, 1458, 1996.
31. Babcock, J. A., Cressler, J. D., Vempati, L., Clark, S. D., Jaeger, R. C., and Harame, D. L., *IEEE Electron Device Letters*, **16**, 351, 1995.
32. Babcock, J. A., Cressler, J. D., Vempati, L., Jaeger, R. C., and Harame, D. L., *IEEE Transactions on Nuclear Science*, **42**, 1558, 1995.
33. Cressler, J. D., Crabbé, E. F., Comfort, J. H., Sun, J.Y.-C., and Stork, J. M. C., *IEEE Electron Device Letters*, **15**, 472, 1994.
34. Cressler, J. D., Comfort, J. H., Crabbé, E. F., Patton, G. L., Stork, J. M. C., Sun, J. Y.-C., and Meyerson, B. S., *IEEE Transactions on Electron Devices*, **40**, 525, 1993.
35. Cressler, J. D., Crabbé, E. F., Comfort, J. H., Stork, J. M. C., and Sun, J. Y.-C., *IEEE Transactions on Electron Devices*, **40**, 542, 1993.
36. Pearsall, T. P. and Bean, J. C., *IEEE Electron Device Letters*, **7**, 308, 1986.
37. Daembkes, H., Herzog, H.-J., Jorke, H., Kibbel, H., and Kasper, E., in *Technical Digest of the International Electron Device Meeting*, 768, 1986.
38. Wang, P. J., Meyerson, B. S., Fang, F. F., Nocera, J., and Parker, B., *Applied Physics Letters*, **55**, 2333, 1989.
39. Nayak, D. K., Woo, J. C. S., Park, J. S., Wang, K. L., and McWilliams, K. P., *IEEE Electron Device Letters*, **12**, 154, 1991.
40. Kesan, V. P., Subbanna, S., Restle, P. J., Tejwani, M. J., Altken, J. M., Iyer, S. S., and Ott, J. A., in *Technical Digest of the International Electron Device Meeting*, 25, 1991.
41. Verdonckt-Vanderbroek, S., Crabbé, E. F., Meyerson, B. S., Harame, D. L. , Restle, P. J., Stork, J. M. C., Megdanis, A. C., Stanis, C. L., Bright, A. A., Kroesen, G. M. W., and Warren, A. C., *IEEE Electron Device Letters*, **12**, 447, 1991.
42. Verdonckt-Vanderbroek, S., Crabbé, E. F., Meyerson, B. S., Harame, D. L., Restle, P. J., Stork, J. M. C., and Johnson, J. B., *IEEE Transactions on Electron Devices*, **41**, 90, 1994.
43. Garone, P. M., Venkataraman, V., and Sturm, J. C., *IEEE Electron Device Letters*, **12**, 230, 1991.
44. Mathew, S., Niu, G., Dubbelday, W., Cressler, J. D., Ott, J., Chu, J., Mooney, P., Kavanaugh, K., Meyerson, B., and Lagnado, I., in *Technical Digest of the IEEE International Electron Devices Meeting*, 815, 1997.
45. Mathew, S., Ansley, W., Dubbelday, W., Cressler, J., Ott, J., Chu, J., Meyerson, B., Kavanaugh, K., Mooney, P., and Lagnado, I., in *Technical Digest of the IEEE Device Research Conference*, 130, 1997.
46. Meyerson, B. S., Uram, K., and LeGoues, F., *Applied Physics Letters*, **53**, 2555, 1988.
47. Ismail, K., Meyerson, B. S., Rishton, S., Chu, J., Nelson, S., and Nocera, J., *IEEE Electron Device Letters*, **13**, 229, 1992.
48. Ismail, K., Nelson, S. F., Chu, J. O., and Meyerson, B. S., *Applied Physics Letters*, **63**, 660, 1993.
49. Welser, J., Hoyt, J. L., and Gibbons, J. F., *IEEE Electron Device Letters*, **15**, 100, 1994.

50. Ismail, K., Chu, J. O., and Meyerson, B. S., *Applied Physics Letters*, **64**, 3124, 1994.
51. Sadek, A., Ismail, K., Armstrong, M. A., Antoniadis, D. A., and Stern, F., *IEEE Transactions on Electron Devices*, **43**, 1224, 1996.
52. Lanzerotti, L., Sturm, J., Stach, E., Hull, R., Buyuklimanli, T., and Magee, C., in *Technical Digest of the IEEE International Electron Device Meeting*, 249, 1996.

Neudeck, P.G. “SiC Technology”
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

6

SiC Technology

Philip G. Neudeck
NASA Glenn Research Center at Lewis Field

6.1 Introduction

Silicon carbide (SiC)-based semiconductor electronic devices and circuits are presently being developed for use in high-temperature, high-power, and/or high-radiation conditions under which conventional semiconductors cannot adequately perform. Silicon carbide's ability to function under such extreme conditions is expected to enable significant improvements to a far-ranging variety of applications and systems. These range from greatly improved high-voltage switching[1–4] for energy savings in public electric power distribution and electric motor drives, to more powerful microwave electronics for radar and communications,[5–7] to sensors and controls for cleaner-burning more fuel-efficient jet aircraft and automobile engines. In the particular area of power devices, theoretical appraisals have indicated that SiC power MOSFETs and diode rectifiers would operate over higher voltage and temperature ranges, have superior switching characteristics, and yet have die sizes nearly 20 times smaller than correspondingly rated silicon-based devices.[8] However, these tremendous theoretical advantages have yet to be realized in experimental SiC devices, primarily due to the fact that SiC's relatively immature crystal growth and device fabrication technologies are not yet sufficiently developed to the degree required for reliable incorporation into most electronic systems.[9]

This chapter briefly surveys the SiC semiconductor electronics technology. In particular, the differences (both good and bad) between SiC electronics technology and well-known silicon VLSI technology are highlighted. Projected performance benefits of SiC electronics are highlighted for several large-scale applications. Key crystal growth and device-fabrication issues that presently limit the performance and capability of high-temperature and/or high-power SiC electronics are identified.

6.2 Fundamental SiC Material Properties

SiC Crystallography: Important Polytypes and Definitions

Silicon carbide occurs in many different crystal structures, called polytypes. A comprehensive introduction to SiC crystallography and polytypism can be found in Ref. 10. Despite the fact that all SiC polytypes chemically consist of 50% carbon atoms covalently bonded with 50% silicon atoms, each SiC polytype has its own distinct set of electrical semiconductor properties. While there are over 100 known polytypes of SiC, only a few are commonly grown in a reproducible form acceptable for use as an electronic semiconductor. The most common polytypes of SiC presently being developed for electronics are 3C-SiC, 4H-SiC, and 6H-SiC. 3C-SiC, also referred to as β-SiC, is the only form of SiC with a cubic crystal lattice structure. The non-cubic polytypes of SiC are sometimes ambiguously referred to as α-SiC. 4H-SiC and 6H-SiC are only two of many possible SiC polytypes with hexagonal crystal structure. Similarly, 15R-SiC is the most common of many possible SiC polytypes with a rhombohedral crystal structure.

Because some important electrical device properties are non-isotropic with respect to crystal orientation, lattice site, and surface polarity, some further understanding of SiC crystal structure and terminology is necessary. As discussed much more thoroughly in Ref. 10, different polytypes of SiC are actually composed of different stacking sequences of Si-C bilayers (also called Si-C double layers), where each single Si-C bilayer can simplistically be viewed as a planar sheet of silicon atoms coupled with a planar sheet of carbon atoms. The plane formed by a bilayer sheet of Si and C atoms is known as the basal plane, while the crystallographic *c*-axis direction, also known as the stacking direction or the [0001] direction, is defined normal to Si-C bilayer plane. Figure 6.1 schematically depicts the stacking sequence of the 6H-SiC polytype, which requires six Si-C bilayers to define the unit cell repeat distance along the *c*-axis [0001] direction. The [1100] direction depicted in Fig. 6.1 is often referred to as the a-axis direction. The silicon atoms labeled "h" or "k" in Figure 1 denote Si-C double layers that reside in "quasi-hexagonal" or "quasi-cubic" environments with respect to their immediately neighboring above and below bilayers. SiC is a polar semiconductor across the *c*-axis, in that one surface normal to the *c*-axis is terminated with silicon atoms, while the opposite normal *c*-axis surface is terminated with carbon atoms. As shown in Fig. 6.1, these surfaces are typically referred to as "silicon face" and "carbon face" surfaces, respectively.

SiC Semiconductor Electrical Properties

Owing to the differing arrangements of Si and C atoms within the SiC crystal lattice, each SiC polytype exhibits unique fundamental electrical and optical properties. Some of the more important semiconductor electrical properties of the 3C, 4H, and 6H silicon carbide polytypes are given in Table 6.1. Much more detailed electrical properties can be found in Refs. 11 to 13 and references therein. Even within a given polytype, some important electrical properties are non-isotropic, in that they are a strong functions of crystallographic direction of current flow and applied electric field (e.g., electron mobility for 6H-SiC). Dopants in SiC can incorporate into energetically inequivalent quasi-hexagonal (*h*) C-sites or Si-sites, or quasi-cubic (*k*) C-sites or Si-sites (only Si-sites are *h* or *k* labeled in Fig. 6.1). While all dopant ionization energies associated with various dopant incorporation sites should normally be considered for utmost accuracy, Table 6.1 lists only the shallowest ionization energies of each impurity.

For comparison, Table 6.1 also includes comparable properties of silicon and GaAs. Because silicon is the semiconductor employed in most commercial solid-state electronics, it is the yardstick against which

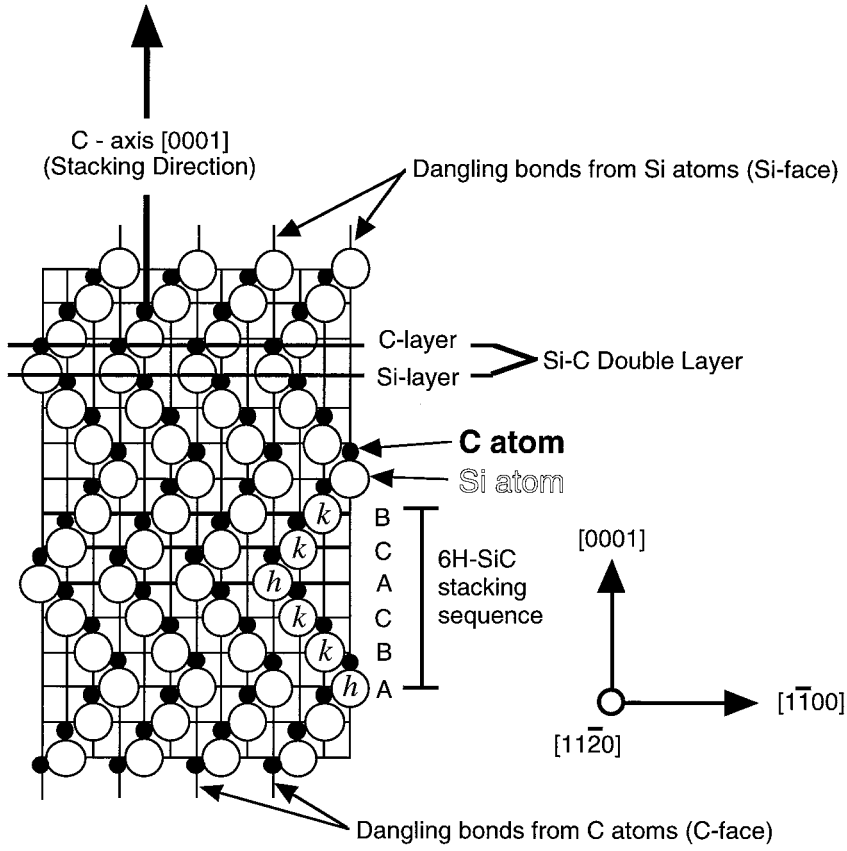


FIGURE 6.1 Schematic cross-section {(1120) plane} of the 6H-SiC polytype. (Modified from Ref. 10. With permission)

TABLE 6.1 Comparison of Selected Important Semiconductors of Major SiC Polytypes with Silicon and GaAs

Property	Silicon	GaAs	4H-SiC	6H-SiC	3C-SiC
Bandgap (eV)	1.1	1.42	3.2	3.0	2.3
Relative dielectric constant	11.9	13.1	9.7	9.7	9.7
Breakdown field $N_D = 10^{17}$ cm^{-3} (MV/cm)	0.6	0.6	//*c*-axis: 3.0	// *c*-axis: 3.2 $\perp$ *c*-axis: >1	>1.5
Thermal conductivity (W/cm-K)	1.5	0.5	3–5	3–5	3–5
Intrinsic carrier concentration (cm^{-3})	10^{10}	1.8×10^6	~10^{-7}	~10^{-5}	~10
Electron mobility @ $N_D = 10^{16}$ cm^{-3} (cm^2/V-s)	1200	6500	//*c*-axis: 800 $\perp$ *c*-axis: 800	//*c*-axis: 60 $\perp$ *c*-axis: 400	750
Hole mobility @ $N_A = 10^{16}$ cm^{-3} (cm^2/V-s)	420	320	115	90	40
Saturated electron velocity (10^7 cm/s)	1.0	1.2	2	2	2.5
Donor dopants & shallowest ionization energy (meV)	P: 45 As: 54	Si: 5.8	N: 45 P: 80	N: 85 P: 80	N: 50
Acceptor dopants & shallowest ionization energy (meV)	B: 45	Be, Mg, C: 28	Al: 200 B: 300	Al: 200 B: 300	Al: 270
1998 Commercial wafer diameter (cm)	30	15	5	5	None

Note: Data compiled from Refs. 11–13, 15, and references therein. (With permission.)

other semiconductor materials must be evaluated. To varying degrees, the major SiC polytypes exhibit advantages and disadvantages in basic material properties compared to silicon. The most beneficial inherent material superiorities of SiC over silicon listed in Table 6.1 are its exceptionally high breakdown electric field, wide bandgap energy, high thermal conductivity, and high carrier saturation velocity. The electrical device performance benefits that each of these properties enable are discussed in the next section, as are system-level benefits enabled by improved SiC devices.

6.3 Applications and Benefits of SiC Electronics

Two of the most beneficial advantages that SiC-based electronics offer are in the areas of high-temperature device operation and high-power device operation. The specific SiC device physics that enables high-temperature and high-power capabilities will be examined first, followed by several examples of revolutionary system-level performance improvements these enhanced capabilities enable.

High-Temperature Device Operation

The wide bandgap energy and low intrinsic carrier concentration of SiC allow SiC to maintain semiconductor behavior at much higher temperatures than silicon, which in turn permits SiC semiconductor device functionality at much higher temperatures than silicon. As discussed in basic semiconductor physics textbooks,[14,15] semiconductor electronic devices function in the temperature range where intrinsic carriers are negligible so that conductivity is controlled by intentionally introduced dopant impurities. Furthermore, the intrinsic carrier concentration n_i is a fundamental prefactor to well-known equations governing undesired junction reverse-bias leakage currents.[15–18] As temperature increases, intrinsic carriers increase exponentially so that undesired leakage currents grow unacceptably large, and eventually at still higher temperatures, the semiconductor device operation is overcome by uncontrolled conductivity as intrinsic carriers exceed intentional device dopings. Depending on specific device design, the intrinsic carrier concentration of silicon generally confines silicon device operation to junction temperatures less than 300°C. The much smaller intrinsic carrier concentration of SiC theoretically permits device operation at junction temperatures exceeding 800°C; 600°C SiC device operation has been experimentally demonstrated on a variety of SiC devices (Section 6.6).

High-Power Device Operation

The high breakdown field and high thermal conductivity of SiC, coupled with high operational junction temperatures, theoretically permit extremely high-power densities and efficiencies to be realized in SiC devices. Figures 6.2 and 6.3 demonstrate the theoretical advantage of SiC's high breakdown field compared to silicon in shrinking the drift-region and associated parasitic on-state resistance of a 3000 V rated unipolar power MOSFET device.[8] The high breakdown field of SiC relative to silicon enables the blocking voltage region to be roughly 10X thinner and 10X heavier-doped, permitting a roughly 100-fold decrease in the dominant blocking region (N-Drift Region) resistance R_D of Fig. 6.2 for the SiC device relative to an identically rated 3000-V silicon power MOSFET.

Significant energy losses in many silicon high-power system circuits, particularly hard-switching motor drive and power conversion circuits, arise from semiconductor switching energy loss.[1,19] While the physics of semiconductor device switching loss are discussed in detail elsewhere,[15–17] switching energy loss is often a function of the turn-off time of the semiconductor switching device, generally defined as the time lapse between when a turn-off bias is applied and the time that the device actually cuts off most current flow. The faster a device turns off, the smaller its energy loss in a switched power conversion circuit. For device-topology reasons discussed in Refs. 8, 20–22, SiC's high breakdown field and wide energy bandgap enable much faster power switching than is possible in comparably volt-amp rated silicon power-switching devices. Therefore, SiC-based power converters could operate at higher switching frequencies with much greater efficiency (i.e., less switching energy loss). Higher switching frequency in

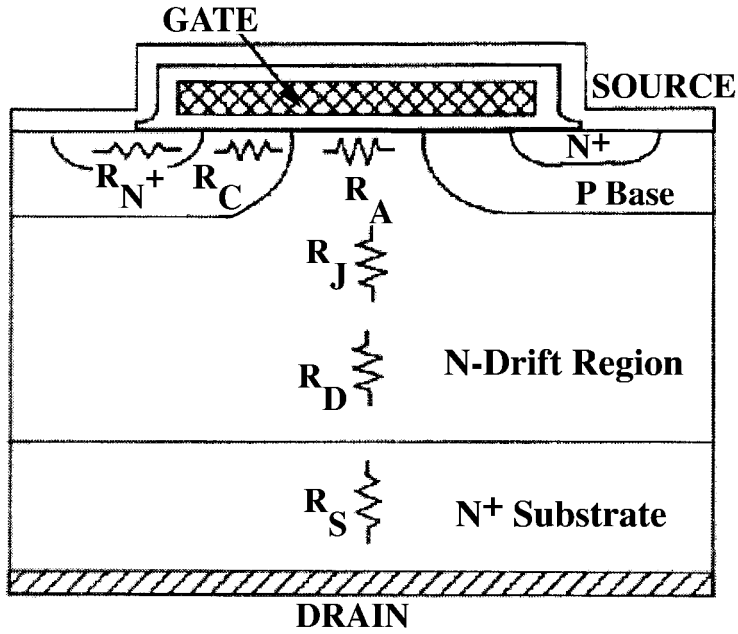


FIGURE 6.2 Cross-section of power MOSFET structure showing various internal resistances. The resistance R_D of the N-Drift Region is the dominant resistance in high-voltage power devices. (From Ref. 8. With permission.)

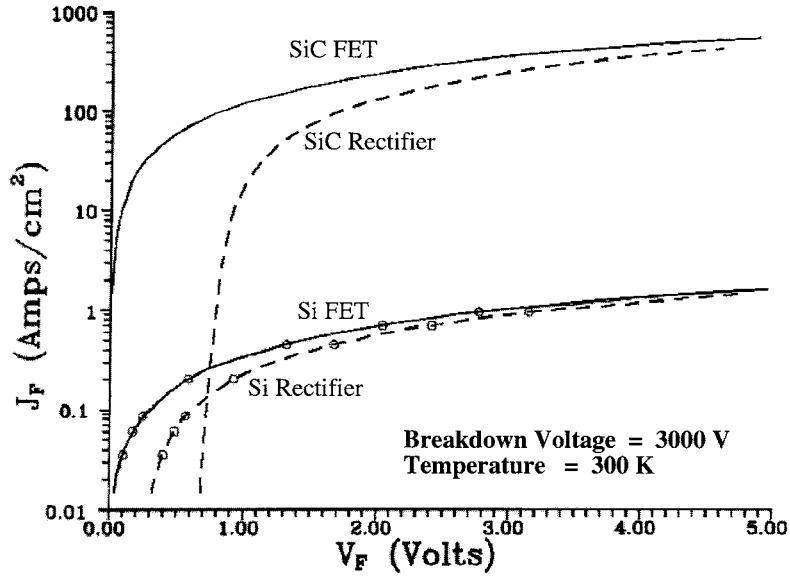


FIGURE 6.3 Simulated forward conduction characteristics of ideal Si and SiC 3000 V power MOSFETs and Schottky rectifiers. The high breakdown field of SiC relative to silicon (Table 6.1) enables the blocking voltage region (N-Drift Region in Fig. 6.2) to be roughly 10X thinner and 10X heavier-doped, permitting a roughly 100-fold increase in on-state current density for the 3000-V SiC devices relative to 3000-V silicon devices. (From Ref. 8. With permission.)

power converters is highly desirable because it permits use of proportionally smaller capacitors, inductors, and transformers, which in turn can greatly reduce overall system size and weight.

While SiC's smaller on-resistance and faster switching help minimize energy loss and heat generation, SiC's higher thermal conductivity enables more efficient removal of waste heat energy from the active device. Because heat energy radiation efficiency increases greatly with increasing temperature difference between the device and the cooling ambient, SiC's ability to operate at high junction temperatures permits much more efficient cooling to take place, so that heatsinks and other device-cooling hardware (i.e., fan cooling, liquid cooling, air conditioning, etc.) typically needed to keep high-power devices from overheating can be made much smaller or even eliminated.

While the preceding discussion focused on high-power switching for power conversion, many of the same arguments can be applied to devices used to generate and amplify RF signals used in radar and communications applications. In particular, the high breakdown voltage and high thermal conductivity, coupled with high carrier saturation velocity, allow SiC microwave devices to handle much higher power densities than their silicon or GaAs RF counterparts, despite SiC's disadvantage in low-field carrier mobility (Section 6.6).[6,7,23]

System Benefits of High-Power, High-Temperature SiC Devices

Uncooled operation of high-temperature and/or high-power SiC electronics would enable revolutionary improvements to aerospace systems. Replacement of hydraulic controls and auxiliary power units with distributed "smart" electromechanical controls capable of harsh-ambient operation will enable substantial jet-aircraft weight savings, reduced maintenance, reduced pollution, higher fuel efficiency, and increased operational reliability.[24–26] SiC high-power solid-state switches will also enable large efficiency gains in electric power management and control. Performance gains from SiC electronics could enable the public power grid to provide increased consumer electricity demand without building additional generation plants, and improve power quality and operational reliability through "smart" power management. More efficient electric motor drives will benefit industrial production systems as well as transportation systems such as diesel-electric railroad locomotives, electric mass-transit systems, nuclear-powered ships, and electric automobiles and buses.

From the above discussions, it should be apparent that SiC high-power and/or high-temperature solid-state electronics promise tremendous advantages that could significantly impact transportation systems and power usage on a global scale. By improving the way in which electricity is distributed and used, improving electric vehicles so that they become more viable replacements for internal combustion-engine vehicles, and improving the fuel efficiency and reducing pollution of remaining fuel-burning engines and generation plants, SiC electronics promises the potential to better the daily lives of all citizens of planet Earth.

6.4 SiC Semicondcutor Crystal Growth

As of this writing, much of the outstanding theoretical promise of SiC electronics highlighted in the previous section has largely gone unrealized. A brief historical examination quickly shows that serious shortcomings in SiC semiconductor material manufacturability and quality have greatly hindered the development of SiC semiconductor electronics. From a simple-minded point of view, SiC electronics development has very much followed the general rule of thumb that a solid-state electronic device can only be as good as the semiconductor material from which it is made.

Historical Lack of SiC Wafers

Most of silicon carbide's superior intrinsic electrical properties have been known for decades. At the genesis of the semiconductor electronics era, SiC was considered an early transistor material candidate, along with germanium and silicon. However, reproducible wafers of reasonable consistency, size, quality,

and availability are a prerequisite for commercial mass-production of semiconductor electronics. Many semiconductor materials can be melted and reproducibly recrystallized into large, single crystals with the aid of a seed crystal, such as in the Czochralski method employed in the manufacture of almost all silicon wafers, enabling reasonably large wafers to be mass-produced. However, because SiC sublimes instead of melting at reasonably attainable pressures, SiC cannot be grown by conventional melt-growth techniques. This prevented the realization of SiC crystals suitable for mass-production until the late 1980s. Prior to 1980, experimental SiC electronic devices were confined to small (typically ~1 cm^2), irregularly shaped SiC crystal platelets (Fig. 6.4, right side) grown as a by-product of the Acheson process for manufacturing industrial abrasives (e.g., sandpaper)[27] or by the Lely process.[28] In the Lely process, SiC sublimed from polycrystalline SiC powder at temperatures near 2500°C are randomly condensed on the walls of a cavity forming small hexagonally shaped platelets. While these small, nonreproducible crystals permitted some basic SiC electronics research, they were clearly not suitable for semiconductor mass-production. As such, silicon became the dominant semiconductor fueling the solid-state technology revolution, while interest in SiC-based microelectronics was limited.

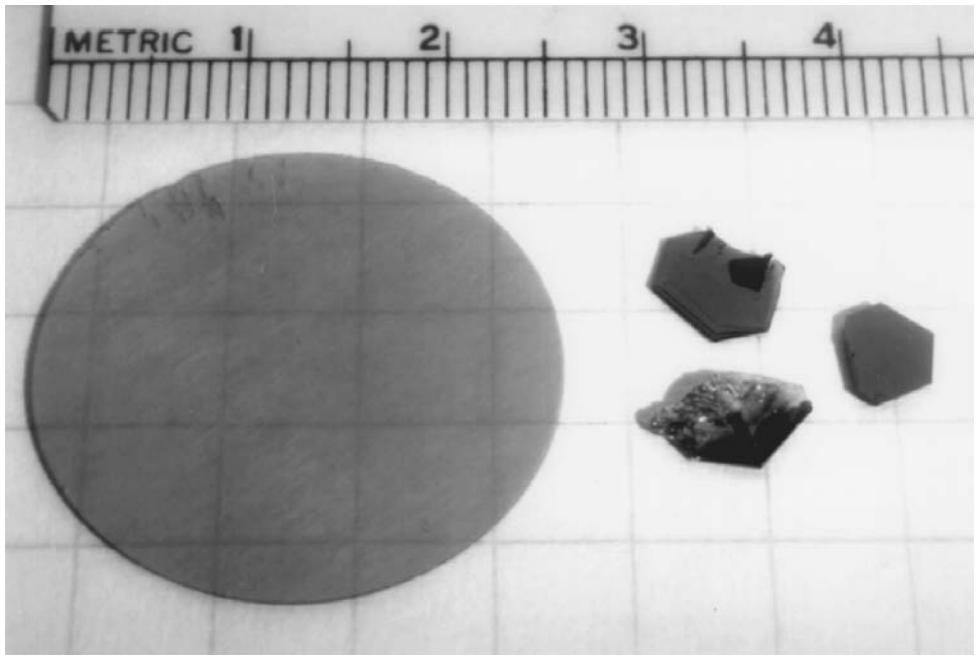


FIGURE 6.4 Mass-produced 2.5-cm diameter 6H-SiC wafer manufactured circa 1990 via seeded sublimation by Cree Research (left), and 6H-SiC Lely and Acheson platelet crystals (right) representative of single-crystal SiC substrates available prior to 1989. 5.1-cm diameter seeded sublimation SiC wafers entered the commercial market in 1997.

Growth of 3C-SiC on Large-Area (Silicon) Substrates

Despite the absence of SiC substrates, the potential benefits of SiC hostile-environment electronics nevertheless drove modest research efforts aimed at obtaining SiC in a manufacturable wafer form. Toward this end, the heteroepitaxial growth of single-crystal SiC layers on top of large-area silicon substrates was first carried out in 1983,[29] and subsequently followed by a great many others over the years using a variety of growth techniques. Primarily due to large differences in lattice constant (20% difference between SiC and Si) and thermal expansion coefficient (8% difference), heteroepitaxy of SiC using silicon as a substrate always results in growth of 3C-SiC with a very high density of crystallographic

structural defects such as stacking faults, microtwins, and inversion domain boundaries.[30,31] Furthermore, the as-grown surface morphology of 3C-SiC grown on silicon is microscopically textured, making submicron lithography somewhat problematic. Other large-area wafer materials, such as sapphire, silicon-on-insulator, TiC, etc.), have been employed as substrates for heteroepitaxial growth of SiC epilayers, but the resulting films have been of comparably poor quality with high crystallographic defect densities.

While some limited semiconductor electronic devices and circuits have been implemented in 3C-SiC grown on silicon,[32,33] the performance of these electronics can be summarized as severely limited by the high density of crystallographic defects to the degree that almost none of the operational benefits discussed in Section 6.3 have been viably realized. Among other problems, the crystal defects "leak" parasitic current across reverse-biased device junctions where current flow is not desired. Because excessive crystal defects lead to electrical device shortcomings, there are as yet no commercial electronics manufactured in 3C-SiC grown on large-area substrates.

Despite the lack of major technical progress, there is strong economic motivation to continue to pursue heteroepitaxial growth of SiC on large-area substrates, as this would provide cheap wafers for SiC electronics that would be immediately compatible with silicon integrated circuit manufacturing equipment. If ongoing work ever solves the extremely challenging crystallographic defect problems associated with the heteroepitaxial growth of SiC, it would likely become the material of choice for mass-production of SiC-based electronics. Given the present electrical deficiencies of heteroepitaxial SiC, 3C-SiC grown on silicon is more likely to be commercialized as a mechanical material in microelectromechanical systems (MEMS) applications (Section 6.6) instead of being used purely as a semiconductor in traditional solid-state electronics.

Sublimation Growth of SiC Wafers

In the late 1970s, Tairov and Tzvetkov established the basic principles of a modified seeded sublimation growth process for growth of 6H-SiC.[34,35] This process, also referred to as the modified Lely process, was a breakthrough for SiC in that it offered the first possibility of reproducibly growing acceptably large single crystals of SiC that could be cut and polished into mass-produced SiC wafers. The basic growth process is based on heating polycrystalline SiC source material to ~2400°C under conditions where it sublimes into the vapor phase and subsequently condenses onto a cooler SiC seed crystal. This produces a somewhat cylindrical boule of single-crystal SiC that grows taller at a rate of a few millimeters per hour. To date, the preferred orientation of the growth in the sublimation process is such that vertical growth of a taller cylindrical boule proceeds along the [0001] crystallographic *c*-axis direction (i.e., vertical direction in Fig. 6.1). Circular "*c*-axis" wafers with surfaces that lie normal (perpendicular) to the *c*-axis can be sawed from the roughly cylindrical boule. While other growth orientations (such as growth along the a-axis) continue to be investigated, the electronic quality of this material has thus far proven inferior to *c*-axis-grown wafers.[36,37]

Commercially Available SiC Wafers

After years of further development of the sublimation growth process, Cree Research became the first company to sell 2.5-cm diameter semiconductor wafers of 6H-SiC (Fig. 6.4, left side) in 1989.[38] Only with the development of the modified Lely seeded sublimation growth technique have acceptably large and reproducible single-crystal SiC wafers of usable electrical quality become available. Correspondingly, the vast majority of silicon carbide semiconductor electronics development has taken place since 1990. Other companies have subsequently entered the SiC wafer market, and sublimation grown wafers of the 4H-SiC polytype have also been commercialized, as summarized in Table 6.2.

Commercially available 4H- and 6H-SiC wafer specifications are given in Table 6.2. N-type, p-type, and semi-insulating SiC wafers are commercially available at different prices. Wafer size, cost, and quality are all very critical to the manufacturability and process yield of mass-produced semiconductor microelectronics. Compared to commonplace silicon and GaAs wafer standards, present-day 4H- and 6H-SiC wafers are small, expensive, and generally of inferior quality. In addition to high densities of crystalline

TABLE 6.2 Commercial Vendors and Specifications of Selected Sublimation-Grown SiC Single-Crystal Wafers.

Vendor [Ref.]	Year	Product	Wafer Diameter	Micropipes (#/cm^2)	Price (U.S.$)
Cree	1993	6H n-type, Si-face, R-Grade	3.0 cm	200–1000	1000
[38]		6H n-type, Si-face, P-Grade	3.0 cm	200–1000	2900
		6H n-type, C-face, P-Grade	3.0 cm	200–1000	3000
		6H p-type, Si-face, P-Grade	3.0 cm	200–1000	3300
		4H n-type, Si-face, R-Grade	3.0 cm	200–1000	3800
Cree	1997	4H n-type, Si-face, R-Grade	3.5 cm	100–200	750
[38]		4H n-type, Si-face, P-Grade	3.5 cm	100–200	1300
		4H n-type, Si-face, P-Grade	3.5 cm	<30	2300
	1998	4H n-type, Si-face, R-Grade	5.1 cm	<200	2100
		4H n-type, Si-face, P-Grade	5.1 cm	<200	3100
	1997	4H p-type, Si-face, R-Grade	3.5 cm	<200	1900
		4H Semi-Insulating, R-Grade	3.5 cm	<200	4800
		6H n-type, Si-face, P-Grade	3.5 cm	<200	1000
		6H p-type, Si-face, P-Grade	3.5 cm	<200	2200
Nippon Steel [142]	1997	4H n-type	2.5 cm	NA	NA
SiCrystal [143]	1997	4H n-type, Quality I	3.5 cm	<200	1200
		4H n-type, Quality III	3.5 cm	400–1000	900
		4H n-type, Quality I	2.5 cm	<200	600
		6H n-type, Quality I	3.5 cm	<200	1200
Sterling and	1998	6H n-type	3.5 cm	<100	800
ATMI/Epitronics		4H n-type	3.5 cm	<100	800
[144][145]					

defects such as micropipes and closed-core screw dislocations discussed in the next subsection, commercial SiC wafers also exhibit significantly rougher surfaces, and larger warpage and bow than is typical for silicon and GaAs wafers.[39] This disparity is not surprising considering that silicon and GaAs wafers have undergone several decades of commercial process refinement, and that SiC is an extraordinarily hard material, making it very difficult to properly saw and polish. Nevertheless, ongoing wafer sawing and polishing process improvements should eventually alleviate wafer surface quality deficiencies.

SiC Wafer Crystal Defects

While the specific electrical effects of SiC crystal defects are discussed later in Section 6.6, the micropipe defect (Table 6.2) is regarded as the most damaging defect that is limiting upscaling of SiC electronics capabilities.[9,40] A micropipe is a screw dislocation with a hollow core and a larger Burgers vector, which becomes a tubular void (with a hollow diameter on the order of micrometers) in the SiC wafer that extends roughly parallel to the crystallographic *c*-axis normal to the polished *c*-axis wafer surface.[41–44] Sublimation-grown 4H- and 6H-SiC wafers also contain high densities of closed-core screw dislocation defects that, like micropipes, cause a considerable amount of localized strain and SiC lattice deformation.[42,43,45,46] Similar to horizontal branches on a tree with its trunk running up the *c*-axis, dislocation loops emanate out along the basal plane from screw dislocations.[41,47] As shown in Table 6.2, micropipe densities in commercial SiC wafers have shown steady improvement over a 5-year period, leading to wafers with less than 30 micropipes per square centimeter of wafer area. However, as discussed in Section 6.6, SiC wafer improvement trends will have to accelerate if some of SiC's most beneficial high-power applications are going to reach timely commercial fruition.

SiC Epilayers

Most SiC electronic devices are not fabricated directly in sublimation-grown wafers, but are instead fabricated in much higher quality epitaxial SiC layers that are grown on top of the initial sublimation-grown wafer. Well-grown SiC epilayers have superior electrical properties and are more controllable and

reproducible than bulk sublimation-grown SiC wafer material. Therefore, the controlled growth of high-quality epilayers is highly important in the realization of useful SiC electronics.

SiC Epitaxial Growth Processes

An interesting variety of SiC epitaxial growth methodologies — ranging from liquid-phase epitaxy, molecular beam epitaxy, and chemical vapor deposition — has been investigated.[11,12,32] The chemical vapor deposition (CVD) growth technique is generally accepted as the most promising method for attaining epilayer reproducibility, quality, and throughputs required for mass-production. In simplest terms, variations of SiC CVD are carried out by heating SiC substrates in a chamber with flowing silicon and carbon containing gases that decompose and deposit Si and C onto the wafer, allowing an epilayer to grow in a well-ordered single-crystal fashion under well-controlled conditions. Conventional SiC CVD epitaxial growth processes are carried out at substrate growth temperatures between 1400 and 1600°C at pressures from 0.1 to 1 atm, resulting in growth rates on the order of micrometers per hour.[39,48–50] Higher-temperature (up to 2000°C) SiC CVD growth processes are also being pioneered to obtain higher SiC epilayer growth rates — on the order of hundreds of micrometers per hour.[51]

SiC Homoepitaxial Growth

Homoepitaxial growth, whereby the polytype of the SiC epilayer matches the polytype of the SiC substrate, is accomplished by step-controlled epitaxy.[39,49,52] Step-controlled epitaxy is based on growing epilayers on an SiC wafer polished at an angle (called the "tilt-angle" or "off-axis angle") of typically 3° to 8° off the (0001) basal plane, resulting in a surface with atomic steps and flat terraces between steps, as schematically depicted in Fig. 6.5. When growth conditions are properly controlled and there is a sufficiently short distance between steps, Si and C atoms impinging onto the growth surface find their way to steps where they bond and incorporate into the crystal. Thus, ordered lateral "step flow" growth takes place, which enables the polytypic stacking sequence of the substrate to be exactly mirrored in the growing epilayer. When growth conditions are not properly controlled or when steps are too far apart (as can occur with SiC substrate surfaces that are polished to within less than 1° of the basal plane), growth adatoms can nucleate and bond in the middle of terraces instead of at the steps; this leads to heteroepitaxial growth of poor-quality 3C-SiC.[39,49] To help prevent spurious nucleation of 3C-SiC "triangular inclusions" during epitaxial growth, most commercial 4H- and 6H-SiC substrates are polished to tilt angles of 8° and 3.5° off the (0001) basal plane, respectively.

It is important to note that most present-day, as-grown SiC epilayers contain varying densities of undesirable surface morphological features that could affect SiC device processing and performance.[39,48] In addition to "triangular inclusions," these include "growth pits" as well as large macrosteps formed by

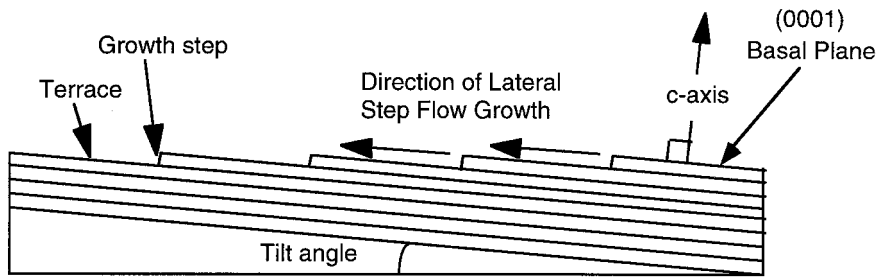


FIGURE 6.5 Cross-sectional schematic representation of "off-axis" polished SiC surface used for homoepitaxial growth. When growth conditions are properly controlled and there is a sufficiently short distance between steps, Si and C atoms impinging onto the growth surface find their way to steps where they bond and incorporate into the crystal. Thus, ordered lateral "step-flow" growth takes place, which enables the polytypic stacking sequence of the substrate to be exactly mirrored in the growing epilayer. (Modified from Ref. 10. With permission.)

coalescence of multiple SiC growth steps (i.e., "step bunching") during epitaxy. Pre-growth wafer polishing as well as growth initiation procedures have been shown to strongly impact the formation of undesirable epitaxial growth features.[39,48] Further optimization of pre-growth treatments and epitaxial growth initiation processes are expected to reduce undesired morphological growth features.

SiC Epilayer Doping

In situ doping during CVD epitaxial growth is primarily accomplished through the introduction of nitrogen (usually N_2) for n-type and aluminum (usually trimethyl- or triethylaluminum) for p-type epilayers.[12] Some alternative dopants such as phosphorous, boron, and vanadium have also been investigated for n-type, p-type, and semi-insulating epilayers, respectively. While some variation in epilayer doping can be carried out strictly by varying the flow of dopant gasses, the site-competition doping methodology[53,54] has enabled a much broader range of SiC doping to be accomplished. In addition, site-competition epitaxy has also made moderate epilayer dopings more reliable and repeatable. The site-competition dopant-control technique is based on the fact that many dopants of SiC preferentially incorporate into either Si lattice sites or C lattice sites. As an example, nitrogen preferentially incorporates into lattice sites normally occupied by carbon atoms. By epitaxially growing SiC under carbon-rich conditions, most of the nitrogen present in the CVD system (whether it is a residual contaminant or intentionally introduced) can be excluded from incorporating into the growing SiC crystal. Conversely, by growing in a carbon-deficient environment, the incorporation of nitrogen can be enhanced to form very heavily doped epilayers for ohmic contacts. Aluminum, which is opposite to nitrogen, prefers the Si-site of SiC, and other dopants have also been controlled through site competition by properly varying the Si/C ratio during crystal growth. SiC epilayer dopings ranging from 9×10^{14} to 1×10^{19} cm^{-3} are commercially available, and researchers have reported obtaining dopings nearly a factor of 10 larger and smaller than this range for n-type and p-type dopings. Commercial epilayer thickness and doping tolerances are presently specified at 25% and 100%, respectively,[38] while doping uniformities of 7% and thickness uniformities of 4% over a 30-mm wafer have been reported in developmental research.[48]

SiC Epilayer Crystal Defects

Improvements in epilayer quality are needed as SiC electronics upscale toward production integrated circuits, as there are presently many observable defects present in state-of-the-art SiC homoepilayers. Non-ideal surface morphological features, such as "growth pits," 3C-SiC triangular inclusions ("triangle defects") introduced in Section 6.4, are generally more prevalent in 4H-SiC epilayers than 6H-SiC epilayers. Most of these features appear to be manifestations of non-optimal "step flow" during epilayer growth arising from substrate defects, non-ideal substrate surface finish, contamination, and/or unoptimized epitaxial growth conditions. While by no means trivial, it is anticipated that SiC epilayer surface morphology will greatly improve as refined substrate preparation and epilayer growth processes are developed.

Many impurities and crystallographic defects found in sublimation-grown SiC wafers do not propagate into SiC homoepitaxial layers. For example, basal-plane dislocation loops emanating from micropipes and screw dislocations in sublimation-grown SiC wafers (Section 6.4) are not generally observed in SiC epilayers.[47] Unfortunately, however, screw dislocations (both micropipes and closed-core screw dislocations) present in commercial *c*-axis wafers do replicate themselves up the crystallographic *c*-axis into SiC homoepilayers grown on commercial wafers. Therefore, as discussed later in Section 6.6, devices fabricated in commercial epilayers are still subject to electrical performance and yield limitations imposed by commercial substrate screw-dislocation defect densities.

Alternative Growth Methods to Reduce SiC Epilayer Dislocations

As of this writing, there is no known practical method of realizing screw-dislocation-free 4H- or 6H-SiC homoepilayers on conventional sublimation-grown substrates. Some non-conventional epitaxial growth techniques have been attempted in an effort to prevent the propagation of micropipes into an epilayer.[55,56] While these approaches have scored modest success in closing and covering up micropipes, to date there

has been little, if any, improvement demonstrated in electrical devices fabricated in the resulting material. This is perhaps due to the fact that screw dislocations and associated harmful stresses may still be present in the epilayer, despite the fact that some open cores may have been converted to closed cores.

Because screw dislocations propagate up the *c*-axis, one could conceivably alleviate screw dislocations by growing epilayers on SiC wafers with their surface parallel to the *c*-axis using "a-axis" wafers. Unfortunately, efforts directed at realizing a-axis wafers and epilayers have to date been much less successful than *c*-axis wafers and epilayers, primarily because defects that form and propagate up the basal plane (the vertical wafer and epilayer growth direction in a-axis-oriented wafers) have proven more harmful and difficult to eliminate than screw dislocations in conventional *c*-axis wafers and epilayers.[36,37]

Selected-area epitaxial growth techniques have recently led to startling reductions in GaN epilayer defect densities.[57] While selective-area epitaxial growth of 3C-SiC has been demonstrated, the applicability of similar techniques to realizing superior electrical-quality SiC will be much more difficult due to the step-flow homoepitaxial growth mechanism of α-SiC as well as high growth temperatures (>1400°C), which are incompatible with conventional growth-masking materials like SiO_2.

6.5 SiC Device Fundamentals

In order to minimize the development and production costs of SiC electronics, it is essential that SiC device fabrication take advantage of the existing silicon and GaAs wafer processing infrastructure as much as possible. As will be discussed in this section, most of the steps necessary to fabricate SiC electronics starting from SiC wafers can be accomplished using somewhat modified commercial silicon electronics processes and fabrication tools.

Choice of Polytype for Devices

As discussed in Section 6.4, 4H-SiC and 6H-SiC are the far superior forms of semiconductor device quality SiC commercially available in mass-produced wafer form. Therefore, only 4H-SiC and 6H-SiC device processing methods will be explicitly considered in the rest of this section. It should be noted, however, that most of the processing methods discussed in this section are applicable to other polytypes of SiC, except for the case of 3C-SiC grown on silicon where all processing temperatures need to be kept well below the melting temperature of silicon (~1400°C).

It is generally accepted that 4H-SiC's substantially higher carrier mobility and shallower dopant ionization energies compared to 6H-SiC (Table 6.1) should make it the polytype of choice for most SiC electronic devices, provided that all other device processing, performance, and cost-related issues play out as being roughly equal between the two polytypes. Furthermore, the inherent mobility anisotropy that degrades conduction parallel to the crystallographic *c*-axis in 6H-SiC[58] will particularly favor 4H-SiC for vertical power device configurations (Section 6.6).

SiC Selective Doping: Ion Implantation

The fact that diffusion coefficients of most SiC dopants are negligibly small below ~1800°C is excellent for maintaining device junction stability, because dopants do not undesirably diffuse as the device is operated long-term at high-temperatures. Unfortunately, however, this characteristic also precludes the use of conventional dopant diffusion, a highly useful technique widely employed in silicon microelectronics manufacturing for patterned doping of SiC.

Laterally patterned doping of SiC is carried out by ion implantation. This somewhat restricts the depth to which most dopants can be conventionally implanted to less than 1 μm using conventional dopants and implantation equipment. Compared to silicon processes, SiC ion-implantation requires a much higher thermal budget to achieve acceptable dopant implant electrical activation. Summaries of ion-implantation processes for various dopants can be found in Refs. 11, 59, and 60. Most of these processes are based on carrying out implantation at elevated temperatures (~500 to 800°C) using a

patterned high-temperature masking material. The elevated temperature during implantation promotes some lattice self-healing during the implant, so that damage and segregation of displaced silicon and carbon atoms does not become excessive, especially in high-dose implants often employed for ohmic contact formation.[59,60] Co-implantation of carbon with p-type dopants has recently been investigated as a means to improve the electrical conductivity of implanted p-type contact layers.[61]

Following implantation, the patterning mask is stripped and a much higher temperature (~1200 to 1800°C) anneal is carried out to achieve maximum electrical activation of dopant donor or acceptor ions. The final annealing conditions are crucial for obtaining desired electrical properties from ion-implanted layers. At higher implant anneal temperatures, the SiC surface morphology can seriously degrade as damage-assisted sublimation etching of the SiC surface begins to take place.[62] Because sublimation etching is driven primarily by loss of silicon from the crystal surface, annealing in silicon overpressures can be used to prevent surface degradation during high-temperature anneals. Such overpressure can be achieved by close-proximity solid sources, such as using an enclosed SiC crucible with SiC lid and/or SiC powder near the wafer, or by annealing in a silane-containing atmosphere.

SiC Contacts and Interconnects

All useful semiconductor electronics require conductive signal paths in and out of each device, as well as conductive interconnects to carry signals between devices on the same chip and to external circuit elements that reside off-chip. While SiC itself is theoretically capable of fantastic operation under extreme conditions (Section 6.3), such functionality is useless without contacts and interconnects that are also capable of operation under the same conditions to enable complete extreme-condition circuit functionality. Previously developed conventional contact and interconnect technologies will likely not be sufficient for reliable operation in extreme conditions that SiC enables. The durability and reliability of metal–semiconductor contacts and interconnects are two of the main factors limiting the operational high-temperature limits of SiC electronics. Similarly, SiC high-power device contacts and metallizations will have to withstand both high-temperature and high current density stress never before encountered in silicon power electronics experience.

The subject of metal–semiconductor contact formation is a very important technical field, too broad to be discussed in detail here. For general background discussions on metal–semiconductor contact physics and formation, the reader should consult narratives presented in Refs. 15 and 63. These references primarily discuss ohmic contacts to conventional narrow-bandgap semiconductors such as silicon and GaAs. Specific overviews of SiC metal–semiconductor contact technology can be found in Refs. 64 to 67.

As discussed in Refs. 64 to 67, there are both similarities and a few differences between SiC ohmic contacts and ohmic contacts to conventional narrow-bandgap semiconductors (e.g., silicon, GaAs). The same basic physics and current transport mechanisms that are present in narrow-bandgap contacts — such as surface states, Fermi-pinning, thermionic emission, and tunneling — also apply to SiC contacts. A natural consequence of the wider bandgap of SiC is higher effective Schottky barrier heights. Analogous with narrow-bandgap ohmic contact physics, the microstructural and chemical state of the SiC–metal interface is crucial to contact electrical properties. Therefore, pre-metal-deposition surface preparation, metal deposition process, choice of metal, and post-deposition annealing can all greatly impact the resulting performance of metal–SiC contacts. Because the chemical nature of the starting SiC surface is strongly dependent on surface polarity, it is not uncommon to obtain significantly different results when the same contact process is applied to the silicon face surface vs. the carbon face surface.

SiC Ohmic Contacts

Ohmic contacts serve the purpose of carrying electrical current into and out of the semiconductor, ideally with no parasitic resistance. The properties of various ohmic contacts to SiC reported to date are summarized in Refs. 66 and 67. While SiC specific ohmic contact resistances at room temperature are generally higher than in contacts to narrow-bandgap semiconductors, they are nevertheless sufficiently low for most envisioned SiC applications. Lower specific contact resistances are usually obtained to n-type 4H- and

6H-SiC (~10^{-4} to 10^{-6} ohm-cm^2) than to p-type 4H- and 6H-SiC (~10^{-3} to 10^{-5} ohm-cm^2). Consistent with narrow-bandgap ohmic contact technology, it is easier to make low-resistance ohmic contacts to heavily doped SiC. While it is possible to achieve ohmic contacts to lighter-doped SiC using high-temperature annealing, the lowest-resistance ohmic contacts are most easily implemented on SiC degenerately doped by site competition (Section 6.4) or high-dose ion implantation (Section 6.5). If the SiC doping is sufficiently degenerate, many metals deposited on a relatively clean SiC surface are ohmic in the "as deposited" state.[68] Regardless of doping, it is common practice in SiC to thermally anneal contacts to obtain the minimum possible ohmic contact resistance. Most SiC ohmic contact anneals are performed at temperatures around 1000°C in non-oxidizing environments. Depending on the contact metallization employed, this anneal generally causes limited interfacial reactions (usually metal-carbide or metal-silicide formation) that broaden and/or roughen the metal–semiconductor interface, resulting in enhanced conductivity through the contact.

Truly enabling harsh-environment SiC electronics will require ohmic contacts that can reliably withstand prolonged harsh-environment operation. Most reported SiC ohmic metallizations appear sufficient for long-term device operation up to 300°C. SiC ohmic contacts that withstand heat soaking under no electrical bias at 500 to 600°C for hundreds or thousands of hours in non-oxidizing gas or vacuum environments have also been demonstrated. In air, however, there has only been demonstration to date of a contact that can withstand heat soaking (no electrical bias) for 60 hours at 650°C.[69] Some very beneficial aerospace systems will require simultaneous high-temperature ($T > 300$°C) and high current density operation in oxidizing air environments. Electromigration, oxidation, and other electrochemical reactions driven by high-temperature electrical bias in a reactive oxidizing environment are likely to limit SiC ohmic contact reliability for the most demanding applications. The durability and reliability of SiC ohmic contacts is one of the critical factors limiting the practical high-temperature limits of SiC electronics.

SiC Schottky Contacts

Rectifying metal–semiconductor Schottky barrier contacts to SiC are useful for a number of devices, including metal–semiconductor field-effect transistors (MESFETs) and fast-switching rectifiers. References 64, 65, 67, and 70 summarize electrical results obtained in a variety of SiC Schottky studies to date. Due to the wide bandgap of SiC, almost all unannealed metal contacts to lightly doped 4H- and 6H-SiC are rectifying. Rectifying contacts permit extraction of Schottky barrier heights and diode ideality factors by well-known current-voltage (I-V) and capacitance-voltage (C-V) electrical measurement techniques.[63] While these measurements show a general trend that Schottky junction barrier height does somewhat depend on metal–semiconductor workfunction difference, the dependence is weak enough to suggest that surface state charge also plays a significant role in determining the effective barrier height of SiC Schottky junctions. At least some experimental scatter exhibited for identical metals can be attributed to cleaning and metal deposition process differences, as well as different barrier height measurement procedures. The work by Teraji et al.,[71] in which two different surface cleaning procedures prior to titanium deposition lead to ohmic behavior in one case and rectifying behavior in the other, clearly shows the important role that process recipe can play in determining SiC Schottky contact electrical properties.

It is worth noting that barrier heights calculated from C-V data are often somewhat higher than barrier heights extracted from I-V data taken from the same diode. Furthermore, the reverse current drawn in experimental SiC diodes, while small, is nevertheless larger than expected based on theoretical substitution of SiC parameters into well-known Schottky diode reverse leakage current equations developed for narrow-bandgap semiconductors. Bhatnagar et al.[72] proposed a model to explain these behaviors, in which localized surface defects, perhaps elementary screw dislocations where they intersect the SiC-metal interface, cause locally reduced junction barriers in the immediate vicinity of defects. Because current is exponentially dependent on Schottky barrier height, this results in the majority of measured current flowing at local defect sites instead of evenly distributed over the entire Schottky diode area. In addition to local defects, electric field crowding along the edge of the SiC Schottky barrier can also lead to increased reverse-bias leakage current and reduced reverse breakdown voltage.[15,16,63]

Schottky diode edge termination techniques to relieve electric field edge crowding and improve Schottky rectifier reverse properties are discussed later in Section 6.6. Quantum mechanical tunneling of carriers through the barrier may also account for some excess reverse leakage current in SiC Schottky diodes.[73]

The high-temperature operation of rectifying SiC Schottky diodes is primarily limited by reverse-bias thermionic leakage of carriers over the junction barrier. Depending on the specific application and the barrier height of the particular device, SiC Schottky diode reverse leakage currents generally grow to excessive levels at around 300 to 400°C. As with ohmic contacts, electrochemical interfacial reactions must also be considered for long-term Schottky diode operation at the highest temperatures.

Patterned Etching of SiC for Device Fabrication

At room temperature, no known wet chemical etches single-crystal SiC. Therefore, most patterned etching of SiC for electronic devices and circuits is accomplished using dry etching techniques. The reader should consult Ref. 74; it contains an excellent summary of dry SiC etching results obtained to date. The most commonly employed process involves reactive ion etching (RIE) of SiC in fluorinated plasmas. Sacrificial etch masks (often aluminum metal) are deposited and photolithographically patterned to protect desired areas from being etched. The SiC RIE process can be implemented using standard silicon RIE hardware, and typical 4H- and 6H-SiC RIE etch rates are on the order of hundreds of angstroms per minute. Well-optimized SiC RIE processes are typically highly anisotropic with little undercutting of the etch mask, leaving smooth surfaces. One of the keys to achieving smooth surfaces is preventing "micromasking," wherein masking material is slightly etched and randomly redeposited onto the sample, effectively masking very small areas on the sample that were intended for uniform etching. This can result in "grass"-like etch-residue features being formed in the unmasked regions, which is undesirable in most cases. In special cases, RIE etching under conditions promoting micromasking is useful in greatly roughening the SiC surface to reduce the contact resistance of subsequently deposited ohmic metallizations.

While RIE etch rates are sufficient for many electronic applications, much higher SiC etch rates are necessary to carve features on the order of tens to hundreds of micrometers deep that are needed to realize advanced sensors, microelectromechanical systems (MEMS), and some very high-voltage power device structures. High-density plasma dry etching techniques, such as electron cyclotron resonance (ECR) and inductively coupled plasma (ICP), have been developed to meet the need for deep-etching of SiC. Residue-free patterned etch rates exceeding a thousand angstroms a minute have been demonstrated.[74–76]

Patterned etching of SiC at very high etch rates has also been demonstrated using photo-assisted and dark electrochemical wet etching.[77,78] By choosing proper etching conditions, this technique has demonstrated a very useful dopant-selective etch-stop capability. However, there are major incompatibilities of the electrochemical process that make it undesirable for VLSI mass-production, including extensive pre-etching and post-etching sample preparation, etch isotropy and mask undercutting, and somewhat non-uniform etching across the sample.

SiC Insulators: Thermal Oxides and MOS Technology

The vast majority of semiconductor integrated circuit chips in use today rely on silicon metal-oxide-semiconductor field effect transistors (MOSFETs), whose electronic advantages and operational device physics are summarized in Choma's chapter on devices and their models and elsewhere.[15,16,79] Given the extreme usefulness and success of MOSFET-based electronics in VLSI silicon, it is naturally desirable to implement high-performance inversion channel MOSFETs in SiC. Like silicon, SiC forms a thermal SiO_2 oxide when it is sufficiently heated in an oxygen environment. While this enables SiC MOS technology to somewhat follow the highly successful path of silicon MOS technology, there are nevertheless important differences in insulator quality and device processing that are presently preventing SiC MOSFETs from realizing their full beneficial potential. While the following discourse attempts to quickly highlight key issues facing SiC MOSFET development, more detailed insights can be found in Refs. 80 to 83. In

highlighting the difficulties facing SiC MOSFET development, it is important to keep in mind that early silicon MOSFETs faced similar developmental challenges that took many years of dedicated research effort to successfully overcome.

From a purely electrical point of view, there are two prime operational deficiencies of SiC oxides and MOSFETs compared to silicon MOSFETs. First, effective inversion channel mobilities in most SiC MOSFETs are much lower (typically well under 100 cm^2/V-s for inversion electrons) than one would expect based on silicon inversion channel MOSFET carrier mobilities. This seriously reduces the transistor gain and current-carrying capability of SiC MOSFETs, so that SiC MOSFETs are not nearly as advantageous as theoretically predicted. Second, SiC oxides have not proven as reliable and immutable as well-developed silicon oxides, in that SiC MOSFETs are more prone to threshold voltage shifts, gate leakage, and oxide failures than comparably biased silicon MOSFETs. The excellent works by Cooper[80] and Brown et al.[83] discuss noteworthy differences between the basic electrical properties of n-type versus p-type SiC MOS devices. SiC MOSFET oxide electrical performance deficiencies appear mostly attributable to differences between silicon and SiC thermal oxide quality and interface structure that cause the SiC oxide to exhibit undesirably higher levels of interface state densities (~10^{11} to 10^{13} $eV^{-1}cm^{-2}$), fixed oxide charges (~10^{11} to 10^{12} cm^{-2}), charge trapping, carrier oxide tunneling, and roughness-related scattering of inversion channel carriers.

One of the most obvious differences between thermal oxidation of silicon and SiC to form SiO_2 is the presence of C in SiC. While most of the C in SiC converts to gaseous CO and CO_2 and escapes the oxide layer during thermal oxidation, leftover C species residing near the SiC–SiO_2 interface nevertheless appear to have a detrimental impact on SiO_2 electrical quality.[80,81] Cleaning treatments and oxidation/anneal recipes aimed at reducing interfacial C appear to improve SiC oxide quality. Another procedure employed to minimize detrimental carbon effects has been to form gate oxides by thermally oxidizing layers of silicon deposited on top of SiC.[84] Likewise, deposited insulators also show promise toward improving SiC MOSFET characteristics, as Sridevan et al.[85] have recently reported greatly improved SiC inversion channel carrier mobilities (>100 cm^2/V-s) using thick deposited gate insulators.

SiC surfaces are well known to be much rougher than silicon surfaces, due to off-angle polishing needed to support SiC homoepitaxy (Fig. 6.5) as well as step-bunching (particularly pronounced in 4H-SiC) that occurs during SiC homoepilayer growth (Section 6.4).[39,86] The impact of surface morphology on inversion channel mobility is highlighted by the recent work of Scharnholz et al.,[87] in which improved mobility (>100 cm^2/V-s) was obtained by specifically orienting SiC MOSFETs in a direction such that current flowed parallel to surface step texture. The interface roughness of SiC may also be a factor in poor oxide reliability by assisting unwanted injection of carriers that damage and degrade the oxide.

As Agarwal et al.[88] have pointed out, the wide bandgap of SiC reduces the potential barrier impeding tunneling of damaging carriers through SiC thermal oxides, so that perfectly grown oxides on atomically smooth SiC would not be as reliable as silicon thermal oxides. Therefore, it is highly probable that alternative gate insulators will have to be developed for optimized implementation of inversion-channel SiC FETs for the most demanding high-power and/or high-temperature electronic applications.

SiC Device Packaging and System Considerations

Hostile-environment SiC semiconductor devices and ICs are of little advantage if they cannot be reliably packaged and connected to form a complete system capable of hostile-environment operation. With proper materials selection, modifications of existing IC packaging technologies appear feasible for non-power SiC circuit packaging up to 300°C.[89,90] Prototype electronic packages that can withstand over a thousand hours of heat soaking without electrical bias at 500°C have been demonstrated.[91] Much work remains before electronics system packaging can meet the needs of the most demanding aerospace electronic applications, whose requirements include high-power operation in high-vibration, 500 to 600°C, oxidizing-ambient environments. Similarly, harsh-environment passive components, such as inductors, capacitors, and transformers, must also be developed for operation in demanding conditions before the full system-level benefits of SiC electronics discussed in Section 6.3 can be successfully realized.

6.6 SiC Electronic Devices and Circuits

This section briefly summarizes a variety of SiC electronic device designs broken down by major application areas. The operational performance of experimental SiC devices is compared to theoretically predicted SiC performance as well as the capabilities of existing silicon and GaAs devices. SiC process and materials technology issues limiting the capabilities of various SiC device topologies are highlighted as key issues to be addressed in further SiC technology maturation.

SiC Optoelectronic Devices

The wide bandgap of SiC is useful for realizing short wavelength blue and ultraviolet (UV) optoelectronics. 6H-SiC-based blue pn junction light-emitting diodes (LEDs) were the first silicon carbide-based devices to reach high-volume commercial sales. These epitaxially grown, dry-etch, mesa-isolated pn junction diodes were the first mass-produced LEDs to cover the blue (~250 to 280 nm peak wavelength) portion of the visible color spectrum, which in turn enabled the realization of the first viable full-color LED-based displays.[92] Because the SiC bandgap is indirect (i.e., the conduction minimum and valence band maximum do not coincide in crystal momentum space), luminescent recombination in the LEDs is governed by inherently inefficient indirect transitions mediated by impurities and phonons.[93] Therefore, the external quantum efficiency of SiC blue LEDs (i.e., percentage of light energy output obtained vs. electrical energy input) was limited to well below 1%. While commercially successful during the 1989 to 1995 timeframe, SiC-based blue LEDs have now been totally obsoleted by the emergence of much brighter, much more efficient, direct-bandgap GaN blue LEDs.

SiC has proven much more efficient at absorbing short-wavelength light, which has enabled the realization of SiC UV-sensitive photodiodes that serve as excellent flame sensors in turbine-engine combustion monitoring and control.[92,94] The wide bandgap of 6H-SiC is useful for realizing low photodiode dark currents, as well as sensors that are blind to undesired near-infrared wavelengths produced by heat and solar radiation. Commercial SiC-based UV flame sensors, again based on epitaxially grown, dry-etch, mesa-isolated 6H-SiC pn junction diodes, have successfully reduced harmful pollution emissions from gas-fired, ground-based turbines used in electrical power generation systems. Prototype SiC photodiodes are also being developed to improve combustion control in jet-aircraft engines.[95]

SiC RF Devices

The main use of SiC RF devices appears to lie in high-frequency solid-state high-power amplification at frequencies from around 600 MHz (UHF-band) to perhaps around 10 GHz (X-band). As discussed in better detail in Refs. 6, 7, 23, 96, and 97, the high breakdown voltage and high thermal conductivity, coupled with high carrier saturation velocity, allow SiC RF transistors to handle much higher power densities than their silicon or GaAs RF counterparts, despite SiC's disadvantage in low-field carrier mobility (Section 6.2). This power output advantage of SiC is briefly illustrated in Fig. 6.6 for the specific case of a Class A MESFET-based RF amplifier. The maximum theoretical RF power of a Class A MESFET operating along the DC load line shown in Fig. 6.6 is approximated by[7]:

$$P_{max} = \frac{I_{dson}(V_b - V_{knee})}{8} \tag{6.1}$$

The higher breakdown field of SiC permits higher drain breakdown voltage (V_b), permitting RF operation at higher drain biases. Given that there is little degradation in I_{dson} and V_{knee} for SiC vs. GaAs and silicon, the increased drain voltage directly leads to higher SiC MESFET output power densities. The higher thermal conductivity of SiC is also crucial in minimizing channel self-heating so that phonon scattering does not seriously degrade channel carrier velocity and I_{dson}. As discussed in Refs. 7 and 97, similar RF output power arguments can be made for SiC-based static induction transistors (SITs).

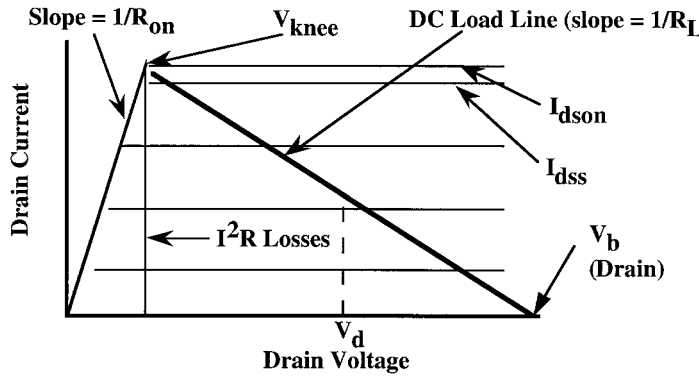


FIGURE 6.6 Piecewise linear MESFET drain characteristic showing DC load line used in Class A RF amplifier operation. The higher breakdown voltage V_b enabled by SiC's higher breakdown field enables operation at higher drain biases, leading to higher RF power densities. (From Ref. 7. With permission.)

The high power density of high-frequency SiC transistors could prove very useful in realizing solid-state transmitters for cell phone base stations, high-definition television (HDTV) transmitters, and radar transmitters, because it reduces the number of devices needed to generate sufficient RF power for these applications. Fewer transistors capable of operating at higher temperatures reduces matching and cooling requirements, leading to reduced overall size and cost of these systems. While excellent for fixed-base high-power RF transmission systems, SiC RF transistors are not well suited for portable handheld RF transceivers where drain voltage and power are restricted to function within the operational limitations of small-sized battery packs.

Because rapid progress is being made toward improving the capabilities of SiC RF power transistors, the reader should consult the latest electron device literature for up-to-date SiC RF transistor capabilities. A late-1997 summary of solid-state high-power RF amplification transistor results, including 4H-SiC, 6H-SiC, silicon, GaAs, and GaN device results, is given in Fig. 6.7.[7] Despite the fact that SiC RF transistors are not nearly as optimized, they have still demonstrated higher power densities than silicon and GaAs RF power transistors. The commercial availability of semi-insulating SiC substrates to minimize parasitic capacitances is crucial to the high-frequency performance of SiC RF MESFETs. MESFET devices fabricated on semi-insulating substrates are conceivably less susceptible to adverse yield consequences arising from micropipes than vertical high-power switching devices, primarily because a *c*-axis micropipe can no longer short together two conducting sides of a high field junction in most areas of the lateral channel MESFET structure. In addition to micropipes, other nonidealities, such as variations in epilayer doping and thickness, surface morphological defects, and slow charge trapping/detrapping phenomena causing unwanted device I-V drift,[98] also limit the yield, size, and manufacturability of SiC RF transistors. However, increasingly beneficial SiC RF transistors should continue to evolve as SiC crystal quality and device processing technology continues to improve.

In addition to high-power RF transistors, SiC mixer diodes show excellent promise for reducing undesired intermodulation interference in RF receivers.[99] More than 20-dB dynamic range improvement was demonstrated using non-optimized SiC Schottky diode mixers. Following further development and optimization, SiC-based mixers should improve the interference immunity of a number of RF systems where receivers and high-power transmitters are closely located, as well as improve the reliability and safety of flight RF-based avionics instruments used to guide aircraft in low-visibility weather conditions.

High-Temperature Signal-Level Devices

Most analog signal conditioning and digital logic circuits are considered "signal level" in that individual transistors in these circuits do not require any more than a few milliamperes of current and less than

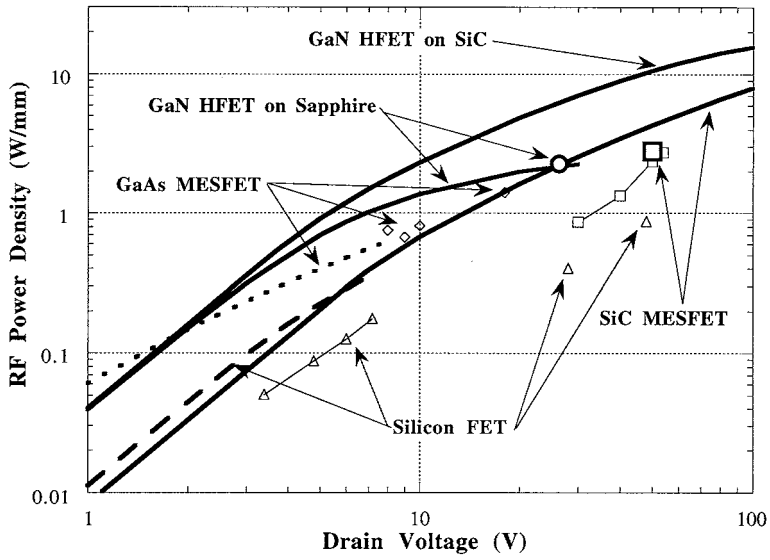


FIGURE 6.7 Theoretical (lines) and experimental (symbols) RF power densities of RF transistors fabricated in silicon, GaAs, SiC, and GaN as of late 1997. (From Ref. 96. With permission.)

20 V to function properly. Commercially available silicon-on-insulator circuits can perform complex digital and analog signal-level functions up to 300°C when high-power output is not required.[100,101] Aside from ICs where it is advantageous to combine signal-level functions with high-power or unique SiC sensors/MEMS onto a single chip, more expensive SiC circuits solely performing low-power signal-level functions appear largely unjustifiable for low-radiation applications at temperatures below 250 to 300°C.

Achieving long-term operational reliability is one of the primary challenges of realizing 300 to 600°C devices and circuits. Circuit technologies that have been used to successfully implement VLSI circuits in silicon and GaAs, such as CMOS, ECL, BiCMOS, DCFL, etc., are to varying degrees candidates for T > 300°C SiC integrated circuits. High temperature gate-insulator reliability (Section 6.5) is critical to the successful realization of MOSFET-based integrated circuits. Gate-to-channel Schottky diode leakage limits the peak operating temperature of SiC MESFET circuits to around 400°C (Section 6.5). Prototype bipolar SiC transistors have exhibited poor gains,[102] but improvements in SiC crystal growth and surface passivation should improve SiC BJT gains.[103] As discussed in Section 6.5, a common obstacle to all technologies is reliable long-term operation of contacts, interconnect, passivation, and packaging at T > 300°C. Because signal-level circuits are operated at relatively low electric fields well below the electrical failure voltage of most micropipes, micropipes affect signal-level circuit process yields to a much lesser degree than they affect high-field power device yields. Nonidealities in SiC epilayers, such as variations in epilayer doping and thickness, surface morphological defects, and slow charge trapping/detrapping phenomena causing unwanted device I–V drift, presently limit the yield, size, and manufacturability of SiC high-temperature integrated circuits.[83] However, continued progress in maturing SiC crystal growth and device fabrication technology should eventually enable the realization of SiC VLSI circuits.

Robust circuit designs that accommodate large changes in device operating parameters with temperature will be necessary for circuits to function successfully over the very wide temperature ranges (as large as 650°C spread) enabled by SiC. While there are similarities to silicon device behavior as a function of temperature, there are also significant differences that will present challenges to SiC integrated circuit designers. For example, in silicon devices, dopant atoms are fully ionized at standard operating temperatures of interest, so that free carrier concentrations correspond with dopant impurity concentrations.[14,15] Therefore, the resistivity of silicon increases with increasing temperature as phonon scattering reduces carrier mobility. SiC device layers, on the other hand, are significantly "frozen-out" due to deeper donor

and acceptor dopant ionization energies, so that non-trivial percentages of dopants are not ionized to produce free carriers that carry current at or near room temperature. Thus, the resistivity of SiC layers can sometimes initially decrease with increasing temperature as dopant atoms ionize to contribute more current-conducting free carriers, then decrease similar to silicon after most dopant atoms have ionized and increased phonon scattering degrades free carrier mobility. Thus, SiC transistor parameters can exhibit temperature variations not found in silicon devices, so that new device-behavior models are sometimes necessary to carry out proper design of wider-temperature range SiC integrated circuits. Because of carrier freeze-out effects, it will be difficult to realize SiC-based ICs operational at temperatures much lower than –55°C (the lower end of the U.S. Mil-Spec. temperature range).

Small-scale prototype logic and analog amplifier SiC-based ICs (one of which is shown in Fig. 6.8) have been demonstrated using SiC variations of NMOS, CMOS, JFET, and MESFET device topologies.[33,83,104–108] These prototypes are not commercially viable as of this writing, largely due to their high cost, unproven reliability, and limited temperature range that is mostly covered by silicon-on-insulator-based circuitry. However, increasingly capable and economical SiC integrated circuits will continue to evolve as SiC crystal growth and device fabrication technology continues to improve.

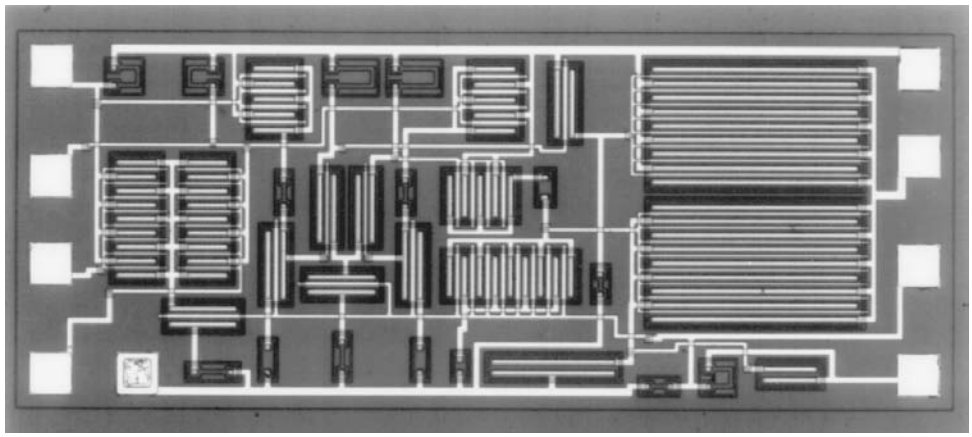


FIGURE 6.8 Optical micrograph of 1 × 2 mm^2 300°C 6H-SiC operational amplifier integrated circuit. The chip contains 14 depletion-mode N-channel MOSFETs integrated with 19 resistors (From Ref. 105. With permission)

SiC High-Power Switching Devices

Operational Limitations Imposed by SiC Material Quality

As discussed in Section 6.3, the most lucrative system benefits of SiC electronics arguably lie in high-power devices. Unfortunately, these devices are also the most susceptible to present-day deficiencies in SiC material quality and process variations, mostly because they operate at high electric fields and high current densities that place the greatest electrical stresses on the semiconductor and surrounding device materials. Prototype SiC devices have demonstrated excellent area-normalized performance, often well beyond (>10X) the theoretical power density of silicon power electronics (Fig. 6.9). However, the presence of micropipe crystal defects has thus far prevented scale-up of small-area prototypes to large areas that can reliably deliver high total operating currents in large-scale power systems, as discussed in Section 6.4 and Refs. 9 and 40. Rectifying power device junctions responsible for OFF-state blocking fail at micropipe defects, leading to undesired (often damaging) localized current flow through micropipes at unacceptably low electric fields well below the critical reverse-breakdown field of defect-free SiC. Over the last decade, SiC micropipe densities have dropped from several hundred per square centimeter of wafer area to tens per square centimeter of wafer area (Section 6.4, Table 6.2), resulting in corresponding improvements in peak SiC device operating currents from less than 1 A to 10s of A. However, further defect reductions

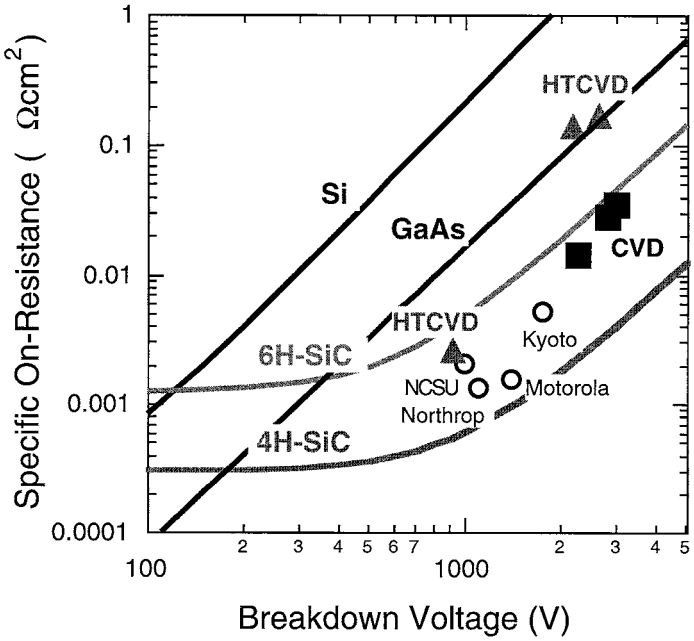


FIGURE 6.9 Experimental SiC (symbols) and theoretical SiC and silicon (lines) Schottky diode specific on resistance plotted as a function of off-state blocking voltage. While the graph clearly shows that the area-normalized performance of small-area SiC devices is orders of magnitude better than silicon, these results have not been successfully upscaled to realize large-area high-current SiC devices due to high densities of device-degrading defects present in commercial SiC wafers and epilayers. (From Ref. 117. With permission.)

of at least an additional order of magnitude will be necessary before reasonably good power device yields and currents will be obtained.

In addition to micropipe defects, the density of non-hollow core (elementary) screw dislocation defects in SiC wafers and epilayers has been measured on the order of several thousands per square centimeter of wafer area (Section 6.4). While these defects are not nearly as detrimental to device performance as micropipes, recent experiments have shown that they degrade the leakage and breakdown characteristics of pn junctions.[109,110] Less direct experimental evidence exists to suggest that elementary screw dislocations may also cause localized reductions in minority carrier diffusion lengths[111,112] and non-uniformities and catastrophic localized failure to high-voltage Schottky rectifiers under reverse bias.[72,113] While localized breakdown is well-known to adversely degrade silicon device reliability in high-power switching applications, the exact impact of localized breakdown in SiC devices has yet to be quantified. If it turns out that SiC power devices roughly adhere to the same reliability physics well-known for silicon power devices, it is possible that SiC devices containing non-hollow core screw dislocations could prove unacceptably unreliable for use in the most demanding high-power conversion applications, such as large-motor control and public power distribution. Thus, these applications might require much larger (i.e., much longer-term) improvements in SiC material quality so as to eliminate all screw dislocations (both hollow core and non-hollow core) from any given device.

SiC High-Voltage Edge Termination

For SiC power devices to successfully function at high voltages, peripheral breakdown due to edge-related electric field crowding[15,16,63] must be avoided through careful device design and proper choice of insulating/passivating dielectric materials. The peak voltage of most prototype high-voltage SiC devices has been limited by often destructive edge-related breakdown, especially in SiC devices capable of blocking multiple kilovolts.[114,115] In addition, most testing of multi-kilovolt SiC devices has required the device to

be immersed in specialized high-dielectric strength fluids or gas atmospheres to minimize damaging electrical arcing and surface flashover at device peripheries.[114,116,117]

A variety of edge termination methodologies, many of which were originally pioneered in silicon high-voltage devices, have been applied to prototype SiC power devices with varying degrees of success. Some of these approaches include tailored dopant guard rings,[122–124] tailored etches,[118–121] neutral ion implant damage rings,[125,126] metal guard rings,[127] and anode contact-insulator overlap.[128,129] The higher voltages and higher local electric fields of SiC power devices will place larger stresses on packaging and on wafer insulating materials, so it is unclear that traditional materials used to insulate/passivate silicon high-voltage devices will prove sufficient for reliable use in SiC high-voltage devices, especially if those devices are to be operated at high-temperatures.

SiC High-Power Rectifiers

The high-power diode rectifier is a critical building block of power conversion circuits. A good review of experimental SiC rectifier results is given in Ref. 67. As discussed in Refs. 8 and 20 to 22, the most important SiC diode rectifier device design tradeoffs roughly parallel well-known silicon rectifier tradeoffs, except for the fact that numbers for current densities, voltages, power densities, and switching speeds are much higher in SiC. SiC's high breakdown field and wide energy bandgap permit operation of SiC metal–semiconductor Schottky diodes at much higher voltages (i.e., kilovolts) and current densities (kA/cm^2) than is practical with silicon-based Schottky diodes. A drawback of the wide bandgap of SiC is that it requires larger forward bias voltages (~1 V) to reach the turn-on "knee" where significant ON-state current begins flowing, and this can lead to an undesirable increase in ON-state power dissipation. However, the benefits of 100X decreased drift region resistance and much faster dynamic switching should greatly overcome SiC ON-state knee voltage disadvantages in most high-power systems. Figure 6.9 summarizes experimental Schottky diode specific on-resistance versus breakdown voltage results published up through 1997.[117] For blocking voltages up to 3 kV, unipolar SiC Schottky rectifiers offer lower turn-on voltages (~1 to 2 V vs. ~3 V) and faster switching speeds (due to no appreciable minority carrier injection/charge storage) than SiC pn junctions.[21,22]

In rectifiers that block over ~3 kV, bipolar minority carrier charge injection (i.e., conductivity modulation) should enable SiC pn diodes to carry higher current densities than unipolar Schottky diodes whose drift regions conduct solely using dopant-atom majority carriers.[20–22] SiC pn junction blocking voltages as high as 5.5 kV have been realized as of this writing,[130] and further blocking voltage improvements are expected as SiC materials growth and processing further improve. Consistent with silicon rectifier experience, SiC pn junction generation-related reverse leakage is usually smaller than thermionic-assisted Schottky diode reverse leakage. While it has not yet been experimentally verified for SiC, silicon power device experience[131] strongly suggests that SiC pn junction rectifiers should offer significantly better reverse breakdown immunity to overvoltage/overcurrent faults that can occur in high-power switching circuits with large inductors than SiC Schottky rectifiers. As with silicon bipolar devices, reproducible localized control of minority carrier lifetime will be essential in optimizing the switching-speed versus ON-state current density performance tradeoffs of SiC bipolar devices for specific applications. SiC minority carrier lifetimes on the order of several microseconds have been obtained in high-quality epilayers,[132] and lifetime reduction via intentional impurity incorporation and introduction of radiation-induced structural defects appears feasible.

Hybrid Schottky/pn rectifier structures first developed in silicon that combine pn junction reverse blocking with low Schottky forward turn-on should prove extremely useful to realizing application-optimized SiC rectifiers.[133,134] Similarly, combinations of dual Schottky metal structures and trench pinch rectifier structures can also be used to optimize SiC rectifier forward turn-on and reverse leakage properties.[135]

SiC High-Power Switching Transistors

Three terminal power switches that use small drive signals to control large voltages and currents are also a critical building blocks of high-power conversion circuits. As well summarized in Ref. 21, a variety of

prototype three-terminal SiC power switches have been demonstrated in recent years. For the most part, SiC solid-state switches are based on well-known silicon device topologies, like the thyristor, vertical MOSFETs, IGBT, GTO, etc., that try to maximize power density via vertical current flow using the substrate as one of the device terminals. Because these switches all contain high-field junctions responsible for blocking current flow in the OFF-state, their maximum operating currents are primarily restricted by the material quality deficiencies discussed in Section 6.6. Therefore, while blocking voltages over 2 kV have been demonstrated in low-current devices,[116] experimental SiC power switches have only realized modest current ratings (under 1 A in most devices).

Silicon power MOSFETs and IGBTs are extremely popular in power circuits, largely because their MOS gate drives are well insulated and require little drive signal power, and the devices are "normally off" in that there is no current flow when the gate is unbiased at 0 V. However, as discussed in Section 6.5, the performance and reliability of SiC power device structures with inversion channel MOS field-effect gates (i.e., MOSFETs, IGBTs, etc.) are limited by poor inversion channel mobilities and questionable oxide reliability at high-temperatures. Thus, SiC device structures that do not rely on high-quality gate oxides, such as the thyristor, appear more favorable for more immediate realization, despite some non-trivial drawbacks in operational circuit design and switching speed.

Recently, some non-traditional power switch topologies have been proposed to somewhat alleviate SiC oxide and material quality deficiencies while maintaining normally off insulated gate operation. Shenoy et al.[136] and Hara[137] respectively, have implemented lateral and vertical doped-channel depletion/accumulation mode power SiC MOSFETs that can be completely depleted by built-in potentials at zero gate bias so that they are "normally off." Spitz et al.[116] recently demonstrated high-voltage SiC lateral MOSFETs implemented on semi-insulating substrates. These devices could conceivably reduce the adverse yield consequences of micropipes, because a *c*-axis micropipe can no longer short together two conducting sides of a high-field junction in most regions of the device. With the assistance of lateral surface electric field tailoring techniques, Baliga[138] has suggested that lateral-conduction SiC power devices could deliver better power densities than traditional vertical SiC power device structures. Baliga has also proposed the advantageous high-voltage switching by pairing a high-voltage SiC MESFET or JFET with a lower-voltage silicon power MOSFET.[138]

SiC for Sensors and Microelectromechanical Systems (MEMS)

Silicon carbide's high-temperature capabilities have enabled the realization of catalytic metal–SiC and metal–insulator–SiC (MIS) prototype gas sensor structures with great promise for emissions monitoring applications.[139,140] High-temperature operation of these structures, not possible with silicon, enables rapid detection of changes in hydrogen and hydrocarbon content to sensitivities of parts-per-million in very small-sized sensors that could easily be placed unobtrusively anywhere on an engine. Once they have been more fully developed, these sensors could assist in active combustion control to reduce harmful pollution emissions from automobile and aircraft engines.

Hesketh's chapter on micromachining describes conventional silicon-based microelectromechanical systems (MEMS). While the previous sections in this chapter have centered on the use of SiC for traditional semiconductor electronic devices, SiC is also likely to play a significant role in emerging MEMS applications.[141] In addition to high-temperature electrical operation, SiC has excellent mechanical properties vital to durable operation of microsystems that address some shortcomings of silicon-based MEMS, such as extreme hardness and low friction, reducing mechanical wear-out as well as excellent chemical inertness to corrosive ambients. Unfortunately, the same properties that make SiC more durable than silicon also make SiC more difficult to process into MEMS structures than silicon. Nevertheless, SiC-based pressure sensors, accelerometers, resonators (Fig. 6.10), and other MEMS systems are being developed for use in harsh-environment applications beyond the reach of silicon-based microsystems. The general approaches to fabricating harsh-environment MEMS structures in SiC and prototype SiC-MEMS results obtained to date are discussed in Ref. 141.

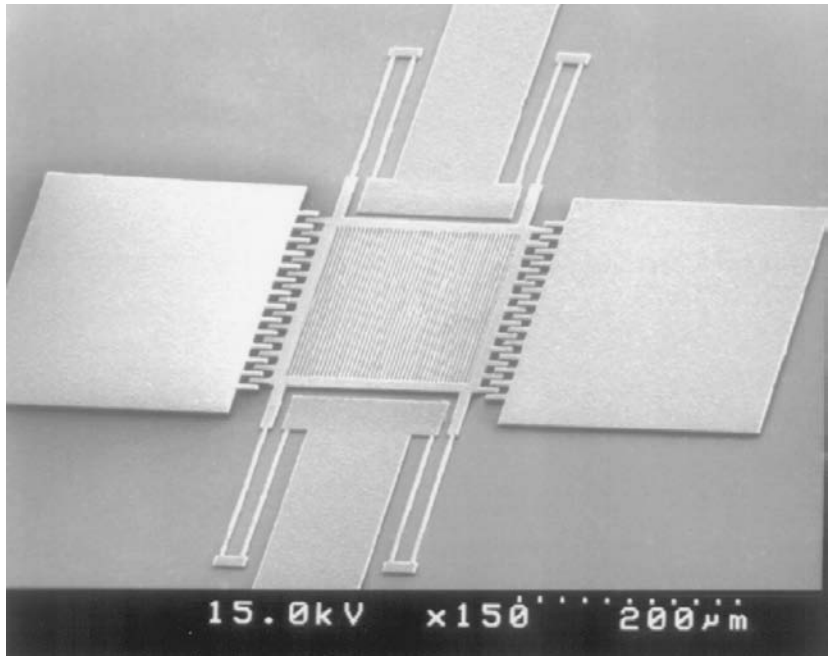


FIGURE 6.10 Micromachined SiC-based lateral resonator device. The excellent mechanical and electrical properties of SiC are enabling the development of harsh-environment microelectromechanical systems (MEMS) for operation beyond the limits of conventional silicon-based MEMS. (From Prof. M. Mehregany, Case Western Reserve University. With permission.)

6.7 Further Recommended Reading

This chapter has presented a brief summary overview of evolving SiC semiconductor device technology. The following publications, which were heavily referenced in this chapter, are highly recommended as advanced, supplemental reading that more completely covers the work being done to develop SiC electronics in much greater technical detail than possible within this short chapter. Reference 12 is a two-volume collection of invited in-depth papers from recognized leaders in SiC technology development that first appeared in special issues of the journal *Physica Status Solidi* (a, 162, No. 1) and (b, 202, No. 1) in 1997. Reference 11 is a two-volume collection of papers from the *7th International Conference on Silicon Carbide, III-Nitrides, and Related Materials* held in Stockholm, Sweden, in September 1997. As SiC electronics is evolving rapidly to fulfill the needs of a steadily increasing array of applications, the reader should consult the current literature for updates on SiC device capabilities. One of the best ongoing sources of SiC electronics information is the *International Conference on Silicon Carbide and Related Materials*, which is held every two years. A meeting was scheduled for October 1999 in Research Triangle Park, North Carolina (internet Web site: www.ISCRM99.ncsu.edu). In addition, a variety of Internet Web sites contain useful SiC information and links, including www.grc.nasa.gov/WWW/SiC/SiC.html, www.hiten.com, www.cree.com, www.ecn.purdue.edu/WBG/, www.sterling-semiconductor.com/, www.imc.kth.se/sic/, and www.ifm.liu.se/Matephys/new_page/research/sic/index.html, among others that can be easily located using widely available World Wide Web Internet search engine services.

References

1. Baliga, B. J., Power Semiconductor Devices for Variable-Frequency Drives, *Proceedings of the IEEE*, 82, 1112, 1994.

2. Baliga, B. J., Power ICs in the Saddle, *IEEE Spectrum*, 32, 34, 1995.
3. Baliga, B. J., Trends in Power Semiconductor Devices, *IEEE Transactions on Electron Devices*, 43, 1717, 1996.
4. Heydt, G. T. and Skromme, B. J., Applications of High Power Electronic Switches in the Electric Power Utility Industry and the Needs for High Power Switching Devices, Power Semiconductor Materials and Devices, *Materials Research Society Symposia Proceedings*, 483, Pearton, S. J., Shul, R. J., Wolfgang, E., Ren, F. and Tenconi, S., Eds., Materials Research Society, Warrandale, PA, 1998, 3.
5. Trew, R. J., Yan, J.-B., and Mock, P. M., The Potential of Diamond and SiC Electronic Devices for Microwave and Millimeter-Wave Power Applications, *Proceedings of the IEEE*, 79, 598, 1991.
6. Weitzel, C. E., Palmour, J. W., Carter, C. H., Jr., Moore, K., Nordquist, K. J., Allen, S., Thero, C., and Bhatnagar, M., Silicon Carbide High Power Devices, *IEEE Transactions on Electron Devices*, 43, 1732, 1996.
7. Weitzel, C. E. and Moore, K. E., Silicon Carbide and Gallium Nitride RF Power Devices, Power Semiconductor Materials and Devices, *Materials Research Society Symposia Proceedings*, 483, Pearton, S. J., Shul, R. J., Wolfgang, E., Ren, F. and Tenconi, S., Eds., Materials Research Society, Warrandale, PA, 1998, 111.
8. Bhatnagar, M. and Baliga, B. J., Comparison of 6H-SiC, 3C-SiC, and Si for Power Devices, *IEEE Transactions on Electron Devices*, 40, 645, 1993.
9. Neudeck, P. G., Progress in Silicon Carbide Semiconductor Electronics Technology, *Journal of Electronic Materials*, 24, 283, 1995.
10. Powell, J. A., Pirouz, P. and Choyke, W. J., Growth and Characterization of Silicon Carbide Polytypes for Electronic Applications, *Semiconductor Interfaces, Microstructures, and Devices: Properties and Applications*, Feng, Z. C., Eds., Institute of Physics Publishing, Bristol, United Kingdom, 1993, 257.
11. Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., *Silicon Carbide, III-Nitrides, and Related Materials, Materials Science Forum*, 264-268, Trans Tech Publications, Switzerland, 1998.
12. Choyke, W. J., Matsunami, H., and Pensl, G., Silicon Carbide — *A Review of Fundamental Questions and Applications to Current Device Technology*, Wiley-VCH, Berlin, 1997.
13. Harris, G. L., *Properties of SiC*, EMIS Datareviews Series, 13, The Institute of Electrical Engineers, London, 1995.
14. Pierret, R. F., *Advanced Semiconductor Fundamentals*, Modular Series on Solid State Devices, Vol. 6, Addison-Wesley, Reading, MA, 1987.
15. Sze, S. M., *Physics of Semiconductor Devices, 2nd. ed.*, Wiley-Interscience, New York, 1981.
16. Baliga, B. J., *Modern Power Devices*, 1st ed., John Wiley & Sons, New York, 1987.
17. Neudeck, G. W., *The PN Junction Diode, Modular Series on Solid State Devices*, 2, 2nd ed., Addison-Wesley, Reading, MA, 1989.
18. Neudeck, G. W., *The Bipolar Junction Transistor*, Modular Series on Solid State Devices, 3, 2nd ed., Addison-Wesley, Reading, MA, 1989.
19. Divan, D., Low-Stress Switching for Efficiency, *IEEE Spectrum*, 33, 33, 1996.
20. Ruff, M., Mitlehner, H. and Helbig, R., SiC Devices: Physics and Numerical Simulation, *IEEE Transactions on Electron Devices*, 41, 1040, 1994.
21. Chow, T. P., Ramungul, N. and Ghezzo, M., Wide Bandgap Semiconductor Power Devices, Power Semiconductor Materials and Devices, *Materials Research Society Symposia Proceedings*, 483, Pearton, S. J., Shul, R. J., Wolfgang, E., Ren, F. and Tenconi, S., Eds., Materials Research Society, Warrandale, PA, 1998, 89.
22. Bakowski, M., Gustafsson, U., and Lindefelt, U., Simulation of SiC High Power Devices, *Physica Status Solidi (a)*, 162, 421, 1997.
23. Trew, R. J., Experimental and Simulated Results of SiC Microwave Power MESFETs, *Physica Status Solidi (a)*, 162, 409, 1997.
24. Nieberding, W. C. and Powell, J. A., High Temperature Electronic Requirements in Aeropropulsion Systems, *IEEE Trans. on Industrial Electronics*, 29, 103, 1982.

25. Carlin, C. M. and Ray, J. K., The Requirements for High Temperature Electronics in a Future High Speed Civil Transport (HSCT), *Second International High Temperature Electronics Conference*, Charlotte, NC, One, King, D. B. and Thome, F. V., Eds., Sandia National Laboratories, Albuquerque, NM, 1994, I-19.
26. Reinhardt, K. C. and Marciniak, M. A., Wide-Bandgap Power Electronics for the More Electric Aircraft, *Transactions 3rd International High Temperature Electronics Conference*, Albuquerque, NM, 1, Sandia National Laboratories, Albuquerque, NM, 1996, I-9.
27. Acheson, A. G., England Patent 17911, 1892.
28. Lely, J. A., Darstellung von Einkristallen von Silicium carbid und Beherrschung von Art und Menge der eingebautem Verunreingungen, *Ber. Deut. Keram.* Ges., 32, 229, 1955.
29. Nishino, S., Powell, J. A., and Will, H. A., Production of Large-Area Single-Crystal Wafers of Cubic SiC for Semiconductor Devices, *Applied Physics Letters*, 42, 460, 1983.
30. Pirouz, P., Chorey, C. M., and Powell, J. A., Antiphase Boundaries in Epitaxially Grown Beta-SiC, *Applied Physics Letters*, 50, 221, 1987.
31. Pirouz, P., Chorey, C. M., Cheng, T. T. and Powell, J. A., Lattice Defects in β-SiC Grown Epitaxially on Silicon Substrates, Heteroepitaxy on Silicon II, *Materials Research Society Symposia Proceedings*, 91, Fan, J. C., Phillips, J. M., and Tsaur, B.-Y., Eds., Materials Reserach Society, Pittsburgh, PA, 1987, 399.
32. Davis, R. F., Kelner, G., Shur, M., Palmour, J. W., and Edmond, J. A., Thin Film Deposition and Microelectronic and Optoelectronic Device Fabrication and Characterization in Monocrystalline Alpha and Beta Silicon Carbide, *Proceedings of the IEEE*, 79, 677, 1991.
33. Harris, G. L., Wongchotigul, K., Henry, H., Diogu, K., Taylor, C., and Spencer, M. G., Beta SiC Schottky Diode FET Inverters Grown on Silicon, Silicon Carbide and Related Materials: *Proceedings of the Fifth International Conference, Institute of Physics Conference Series*, 137, Spencer, M. G., Devaty, R. P., Edmond, J. A., Kahn, M. A., Kaplan, R., and Rahman, M., Eds., IOP Publishing, Bristol, United Kingdom, 1994, 715.
34. Tairov, Y. M. and Tsvetkov, V. F., Investigation of Growth Processes of Ingots of Silicon Carbide Single Crystals, *Journal of Crystal Growth*, 43, 209, 1978.
35. Tairov, Y. M. and Tsvetkov, V. F., General Principals of Growing Large-Size Single Crystals of Various Silicon Carbide Polytypes, *Journal of Crystal Growth*, 52, 146, 1981.
36. Eldridge, G. W., Barrett, D. L., Burk, A. A., Hobgood, H. M., Siergiej, R. R., Brandt, C. D., Tischler, M. A., Bilbro, G. L., Trew, R. J., Clark, W. H., and Gedridge, R. W., Jr., High Power Silicon Carbide IMPATT Diode Development, *2nd Annual AIAA SDIO Interceptor Technology Conference*, Albuquerque, NM, American Institute of Aeronautics and Astronautics, Report 93-2703, Washington D.C., 1993.
37. Takahashi, J. and Ohtani, N., Modified-Lely SiC Crystals Grown in [1100] and [1120] Directions, *Physica Status Solidi (b)*, 202, 163, 1997.
38. Cree Research, Inc., 4600 Silicon Drive, Durham, NC 27703, http://www.cree.com.
39. Powell, J. A. and Larkin, D. J., Processed-Induced Morphological Defects in Epitaxial CVD Silicon Carbide, *Physica Status Solidi (b)*, 202, 529, 1997.
40. Neudeck, P. G. and Powell, J. A., Performance Limiting Micropipe Defects in Silicon Carbide Wafers, *IEEE Electron Device Letters*, 15, 63, 1994.
41. Yang, J.-W., SiC: Problems in Crystal Growth and Polytypic Transformation, Ph. D. dissertation, Case Western Reserve University, Cleveland, OH, 1993.
42. Si, W., Dudley, M., Glass, R., Tsvetkov, V., and Carter, C. H., Jr., Hollow-Core Screw Dislocations in 6H-SiC Single Crystals: A Test of Frank's Theory, *Journal of Electronic Materials*, 26, 128, 1997.
43. Si, W. and Dudley, M., Study of Hollow-Core Screw Dislocations in 6H-SiC and 4H-SiC Single Crystals, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 429.

44. Heindl, J., Strunk, H. P., Heydemann, V. D., and Pensl, G., Micropipes: Hollow Tubes in Silicon Carbide, *Physica Status Solidi (a)*, 162, 251, 1997.
45. Wang, S., Dudley, M., Carter, C. H., Jr., Tsvetkov, V. F. and Fazi, C., Synchrotron White Beam Topography Studies of Screw Dislocations in 6H-SiC Single Crystals, Applications of Synchrotron Radiation Techniques to Materials Science, *Mat. Res. Soc. Proc.*, 375, Terminello, L., Shinn, N., Ice, G., D'Amico, K., and Perry, D., Eds., Materials Research Society, Warrandale, PA, 1995, 281.
46. Dudley, M., Wang, S., Huang, W., Carter, C. H., Jr., and Fazi, C., White Beam Synchrotron Topographic Studies of Defects in 6H-SiC Single Crystals, *Journal of Physics D: Applied Physics*, 28, A63, 1995.
47. Wang, S., Dudley, M., Carter, C. H., Jr., and Kong, H. S., X-Ray Topographic Studies of Defects in PVT 6H-SiC Substrates and Epitaxial 6H-SiC Thin Films, Diamond, SiC and Nitride Wide Bandgap Semiconductors, *Materials Research Society Symposium Proceedings*, 339, Carter, C. H., Jr., Gildenblat, G., Nakamura, S. and Nemanich, R. J., Eds., Materials Research Society, Pittsburgh, PA, 1994, 735.
48. Burk, A. A., Jr. and Rowland, L. B., Homoepitaxial VPE Growth of SiC Active Layers, *Physica Status Solidi (b)*, 202, 263, 1997.
49. Kimoto, T., Itoh, A., and Matsunami, H., Step-Controlled Epitaxial Growth of High-Quality SiC Layers, *Physica Status Solidi (b)*, 202, 247, 1997.
50. Rupp, R., Makarov, Y. N., Behner, H., and Wiedenhofer, A., Silicon Carbide Epitaxy in a Vertical CVD Reactor: Experimental Results and Numerical Process Simulation, *Phys. Stat. Sol. (b)*, 202, 281, 1997.
51. Kordina, O., Hallin, C., Henry, A., Bergman, J. P., Ivanov, I., Ellison, A., Son, N. T., and Janzen, E., Growth of SiC by "Hot-Wall" CVD and HTCVD, *Physica Status Solidi (b)*, 202, 321, 1997.
52. Kong, H. S., Glass, J. T., and Davis, R. F., Chemical Vapor Deposition and Characterization of 6H-SiC Thin Films on Off-Axis 6H-SiC Substrates, *Journal of Applied Physics*, 64, 2672, 1988.
53. Larkin, D. J., Neudeck, P. G., Powell, J. A., and Matus, L. G., Site-Competition Epitaxy for Superior Silicon Carbide Electronics, *Applied Physics Letters*, 65, 1659, 1994.
54. Larkin, D. J., SiC Dopant Incorporation Control Using Site-Competition CVD, *Phys. Stat. Sol. (b)*, 202, 305, 1997.
55. Rendakova, S. V., Nikitina, I. P., Tregubova, A. S., and Dmitriev, V. A., Micropipe and Dislocation Density Reduction in 6H-SiC and 4H-SiC Structures Grown by Liquid Phase Epitaxy, *Journal of Electronic Materials*, 27, 292, 1998.
56. Khlebnikov, I., Sudarshan, T. S., Madangarli, V., and Capano, M. A., A Technique for Rapid Thick Film SiC Epitaxial Growth, Power Semiconductor Materials and Devices, *Materials Research Society Symposia Proceedings*, 483, Pearton, S. J., Shul, R. J., Wolfgang, E., Ren, F., and Tenconi, S., Eds., Materials Research Society, Warrandale, PA, 1998, 123.
57. Nam, O. H., Zheleva, T. S., Bremser, M. D., and Davis, R. F., Lateral Epitaxial Overgrowth of GaN Films on SiO_2 Areas Via Metalorganic Vapor Phase Epitaxy, *Journal of Electronic Materials*, 27, 233, 1998.
58. Schaffer, W. J., Negley, G. H., Irvine, K. G., and Palmour, J. W., Conductivity Anisotropy in Epitaxial 6H and 4H SiC, Diamond, SiC, and Nitride Wide-Bandgap Semiconductors, *Materials Research Society Symposia Proceedings*, 339, Carter, C. H., Jr., Gildenblatt, G., Nakamura, S., and Nemanich, R. J., Eds., Materials Research Society, Pittsburgh, PA, 1994, 595.
59. Troffer, T., Schadt, M., Frank, T., Itoh, H., Pensl, G., Heindl, J., Strunk, H. P., and Maier, M., Doping of SiC by Implantation of Boron and Aluminum, *Physica Status Solidi (a)*, 162, 277, 1997.
60. Kimoto, T., Itoh, A., Inoue, N., Takemura, O., Yamamoto, T., Nakajima, T., and Matsunami, H., Conductivity Control of SiC by In-Situ Doping and Ion Implantation, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 675.

61. Zhao, J. H., Tone, K., Weiner, S. R., Caleca, M. A., Du, H., and Withrow, S. P., Evaluation of Ohmic Contacts to P-Type 6H-SiC Created by C and Al Coimplantation, *IEEE Electron Device Letters*, 18, 375, 1997.
62. Capano, M. A., Ryu, S., Melloch, M. R., Cooper, J. A., Jr., and Buss, M. R., Dopant Activation and Surface Morphology of Ion Implanted 4H- and 6H-Silicon Carbide, *Journal of Electronic Materials*, 27, 370, 1998.
63. Rhoderick, E. H. and Williams, R. H., Metal-Semiconductor Contacts, *Monographs in Electrical and Electronic Engineering*, 19, Clarendon Press, Oxford, UK, 1988.
64. Porter, L. M. and Davis, R. F., A Critical Review of Ohmic and Rectifying Contacts for Silicon Carbide, *Materials Science and Engineering B*, B34, 83, 1995.
65. Bozack, M. J., Surface Studies on SiC as Related to Contacts, *Physica Status Solidi (b)*, 202, 549, 1997.
66. Crofton, J., Porter, L. M., and Williams, J. R., The Physics of Ohmic Contacts to SiC, *Physica Status Solidi (b)*, 202, 581, 1997.
67. Saxena, V. and Steckl, A. J., Building Blocks for SiC Devices: Ohmic Contacts, Schottky Contacts, and p-n Junctions, *Semiconductors and Semimetals*, 52, Academic Press, New York, 1998, 77.
68. Petit, J. B., Neudeck, P. G., Salupo, C. S., Larkin, D. J., and Powell, J. A., Electrical Characteristics and High Temperature Stability of Contacts to N- and P-Type 6H-SiC, Silicon Carbide and Related Materials, *Institute of Physics Conference Series*, 137, Spencer, M. G., Devaty, R. P., Edmond, J. A., Kahn, M. A., Kaplan, R. and Rahman, M., Eds., IOP Publishing, Bristol, 1994, 679.
69. Okojie, R. S., Ned, A. A., Provost, G. and Kurtz, A. D., Characterization of Ti/TiN/Pt Contacts on N-Type 6H-SiC Epilayer at 650°C, *1998 4th International High Temperature Electronics Conference*, Albuquerque, NM, IEEE, Piscataway, NJ, 1998, 79.
70. Itoh, A. and Matsunami, H., Analysis of Schottky Barrier Heights of Metal/SiC Contacts and Its Possible Application to High-Voltage Rectifying Devices, *Physica Status Solidi (a)*, 162, 389, 1997.
71. Teraji, T., Hara, S., Okushi, H., and Kajimura, K., Ideal Ohmic Contact to N-Type 6H-SiC by Reduction of Schottky Barrier Height, *Applied Physics Letters*, 71, 689, 1997.
72. Bhatnagar, M., Baliga, B. J., Kirk, H. R., and Rozgonyi, G. A., Effect of Surface Inhomogenities on the Electrical Characteristics of SiC Schottky Contacts, *IEEE Transactions on Electron Devices*, 43, 150, 1996.
73. Crofton, J. and Sriram, S., Reverse Leakage Current Calculations for SiC Schottky Contacts, *IEEE Transactions on Electron Devices*, 43, 2305, 1996.
74. Yih, P. H., Saxena, V., and Steckl, A. J., A Review of SiC Reactive Ion Etching in Fluorinated Plasmas, *Physica Status Solidi (b)*, 202, 605, 1997.
75. Cao, L., Li, B., and Zhao, J. H., Inductively Coupled Plasma Etching of SiC for Power Switching Device Fabrication, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 833.
76. McLane, G. F. and Flemish, J. R., High Etch Rates of SiC in Magnetron Enhanced SF_6 Plasmas, *Applied Physics Letters*, 68, 3755, 1996.
77. Shor, J. S. and Kurtz, A. D., Photoelectrochemical Etching of 6H-SiC, *Journal of the Electrochemical Society*, 141, 778, 1994.
78. Shor, J. S., Kurtz, A. D., Grimberg, I., Weiss, B. Z., and Osgood, R. M., Dopant-Selective Etch Stops in 6H and 3C SiC, *Journal of Applied Physics*, 81, 1546, 1997.
79. Pierret, R. F., Field Effect Devices, *Modular Series on Solid State Devices*, 4, Addison-Wesley, Reading, MA, 1983.
80. Cooper, J. A., Jr., Advances in SiC MOS Technology, *Physica Status Solidi (a)*, 162, 305, 1997.
81. Afanasev, V. V., Bassler, M., Pensl, G., and Schulz, M., Intrinsic SiC/SiO_2 Interface States, *Physica Status Solidi (a)*, 162, 321, 1997.
82. Ouisse, T., Electron Transport at the SiC/SiO_2 Interface, *Physica Status Solidi (a)*, 162, 339, 1997.

83. Brown, D. M., Downey, E., Ghezzo, M., Kretchmer, J., Krishnamurthy, V., Hennessy, W., and Michon, G., Silicon Carbide MOSFET Integrated Circuit Technology, *Physica Status Solidi (a)*, 162, 459, 1997.
84. Tan, J., Das, M. K., Cooper, J. A., Jr., and Melloch, M. R., Metal-Oxide-Semiconductor Capacitors Formed by Oxidation of Polycrystalline Silicon on SiC, *Applied Physics Letters*, 70, 2280, 1997.
85. Sridevan, S. and Baliga, B. J., Inversion Layer Mobility in SiC MOSFETs, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 997.
86. Powell, J. A., Larkin, D. J., and Abel, P. B., Surface Morphology of Silicon Carbide Epitaxial Films, *Journal of Electronic Materials*, 24, 295 , 1995.
87. Scharnholz, S., Stein von Kamienski, E., Golz, A., Leonhard, C., and Kurz, H., Dependence of Channel Mobility on the Surface Step Orientation in Planar 6H-SiC MOSFETs, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 1001.
88. Agarwal, A. K., Seshadri, S., and Rowland, L. B., Temperature Dependence of Fowler-Nordheim Current in 6H- and 4H-SiC MOS Capacitors, *IEEE Electron Device Letters*, 18, 592, 1997.
89. Grzybowski, R. R. and Gericke, M., 500°C Electronics Packaging and Test Fixturing, *Second International High Temperature Electronic Conference*, Charlotte, NC, 1, King, D. B., and Thome, F. V., Eds., Sandia National Laboratories, Albuquerque, NM, 1994, IX-41.
90. Bratcher, M., Yoon, R. J., and Whitworth, B., Aluminum Nitride Package for High Temperature Applications, *Transactions 3rd International High Temperature Electronics Conference*, Albuquerque, NM, 2, Sandia National Laboratories, Albuquerque, NM, 1996, P-21.
91. Salmon, J. S., Johnson, R. W., and Palmer, M., Thick Film Hybrid Packaging Techniques for 500°C Operation, *1998 4th International High Temperature Electronics Conference*, Albuquerque, NM, IEEE, Piscataway, NJ, 1998, 103.
92. Edmond, J., Kong, H., Suvorov, A., Waltz, D., and Carter, C., Jr., 6H-Silicon Carbide Light Emitting Diodes and UV Photodiodes, *Physica Status Solidi (a)*, 162, 481, 1997.
93. Bergh, A. A. and Dean, P. J., *Light-Emitting Diodes*, Clarendon Press, Oxford, 1976.
94. Brown, D. M., Downey, E., Kretchmer, J., Michon, G., Shu, E., and Schneider, D., SiC Flame Senors for Gas Turbine Control Systems, *Solid-State Electronics*, 42, 755, 1998.
95. Przybylko, S. J., Developments in Silicon Carbide for Aircraft Propulsion System Applications, *AIAA/SAE/ASME/ASEE 29th Joint Propulsion Conference and Exhibit*, American Institute of Aeronautics and Astronautics, Report 93-2581, Washington D.C., 1993.
96. Weitzel, C., Pond, L., Moore, K., and Bhatnagar, M., Effect of Device Temperature on RF FET Power Density, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 969.
97. Sriram, S., Siergiej, R. R., Clarke, R. C., Agarwal, A. K., and Brandt, C. D., SiC for Microwave Power Transistors, *Physica Status Solidi (a)*, 162, 441, 1997.
98. Noblanc, O., Arnodo, C., Chartier, E., and Brylinski, C., Characterization of Power MESFETs on 4H-SiC Conductive and Semi-Insulating Wafers, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 949.
99. Fazi, C. and Neudeck, P., Use of Wide-Bandgap Semiconductors to Improve Intermodulation Distortion in Electronic Systems, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 913.
100. AlliedSignal Microelectronics and Technology Center, 9140 Old Annapolis Road, Columbia, MD 21045, http://www.mtcsemi.com.

101. Honeywell Solid State Electronics Center, 12001 State Highway 55, Plymouth, MN 55441, http://www.ssec.honeywell.com.
102. Wang, Y., W.Xie, Cooper, J. A., Jr., Melloch, M. R., and Palmour, J. W., Mechanisms Limiting Current Gain in SiC Bipolar Junction Transistors, Silicon Carbide and Related Materials 1995, *Institute of Physics Conference Series*, 142, Nakashima, S., Matsunami, H., Yoshida, S., and Harima, H., Eds., IOP Publishing, Bristol, U.K., 1996, 809.
103. Neudeck, P. G., Perimeter Governed Minority Carrier Lifetimes in 4H-SiC p^+n Diodes Measured by Reverse Recovery Switching Transient Analysis, *Journal of Electronic Materials*, 27, 317, 1998.
104. Xie, W., Cooper, J. A., Jr., and Melloch, M. R., Monolithic NMOS Digital Integrated Circuits in 6H-SiC, *IEEE Electron Device Letters*, 15, 455, 1994.
105. Brown, D. M., Ghezzo, M., Kretchmer, J., Krishnamurthy, V., Michon, G., and Gati, G., High Temperature Silicon Carbice Planar IC Technology and First Monolithic SiC Operational Amplifier IC, *Second International High Temperature Electronics Conference*, Charlotte, NC, 1, Sandia National Laboratories, Albuquerque, NM, 1994, XI-17.
106. Ryu, S. H., Kornegay, K. T., Cooper, J. A., Jr., and Melloch, M. R., Digital CMOS IC's in 6H-SiC Operating on a 5-V Power Supply, *IEEE Transactions on Electron Devices*, 45, 45, 1998.
107. Diogu, K. K., Harris, G. L., Mahajan, A., Adesida, I., Moeller, D. F., and Bertram, R. A., Fabrication and Characterization of a 83 MHz High Temperature β-SiC MESFET Operational Amplifier with an AlN Isolation Layer on (100) 6H-SiC, *54th Annual IEEE Device Research Conference*, Santa Barbara, CA, IEEE, Piscataway, NJ, 1996, 160.
108. Neudeck, P. G., 600°C Digital Logic Gates, *NASA Lewis 1998 Research & Technology Report*, 1999.
109. Neudeck, P. G., Huang, W., and Dudley, M., Breakdown Degradation Associated with Elementary Screw Dislocations in 4H-SiC P^+N Junction Rectifiers, Power Semiconductor Materials and Devices, *Materials Research Society Symposia Proceedings*, 483, Pearton, S. J., Shul, R. J., Wolfgang, E., Ren, F., and Tenconi, S., Eds., Materials Research Society, Warrandale, PA, 1998, 285.
110. Neudeck, P. G., Huang, W., Dudley, M., and Fazi, C., Non-Micropipe Dislocations in 4H-SiC Devices: Electrical Properties and Device Technology Implications, Wide-Bandgap Semiconductors for High Power, High Frequency and High Temperature, *Materials Research Society Symposia Proceedings*, 512, Denbaars, S., Shur, M. S., Palmour, J., and Spencer, M., Eds., Materials Research Society, Warrandale, PA, 1998, 107.
111. Doolittle, W. A., Rohatgi, A., Ahrenkiel, R., Levi, D., Augustine, G., and Hopkins, R. H., Understanding the Role of Defects in Limiting the Minority Carrier Lifetime in SiC, Power Semiconductor Materials and Devices, *Materials Research Society Symposia Proceedings*, 483, Pearton, S. J., Shul, R. J., Wolfgang, E., Ren, F., and Tenconi, S., Eds., Materials Research Society, Warrandale, PA, 1998, 197.
112. Hubbard, S. M., Effect of Crystal Defects on Minority Carrier Diffusion Length in 6H SiC Measured Using the Electron Beam Induced Current Method, Master of Science dissertation, Case Western Reserve University, Cleveland, OH, 1998.
113. Raghunathan, R. and Baliga, B. J., Role of Defects in Producing Negative Temperature Dependence of Breakdown Voltage in SiC, *Applied Physics Letters*, 72, 3196, 1998.
114. Neudeck, P. G., Larkin, D. J., Powell, J. A., Matus, L. G., and Salupo, C. S., 2000 V 6H-SiC p-n Junction Diodes Grown by Chemical Vapor Deposition, *Applied Physics Letters*, 64, 1386, 1994.
115. Domeij, M., Breitholtz, B., Linnros, J., and Ostling, M., Reverse Recovery and Avalanche Injection in High Voltage SiC PIN Diodes, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Morkoc, H., Pensl, G., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 1041.
116. Spitz, J., Melloch, M. R., Cooper, J. A., Jr., and Capano, M. A., 2.6 kV 4H-SiC Lateral DMOSFET's, *IEEE Electron Device Letters*, 19, 100, 1998.

117. Kimoto, T., Wahab, Q., Ellison, A., Forsberg, U., Tuominen, M., Yakimova, R., Henry, A., and Janzen, E., High-Voltage (>2.5kV) 4H-SiC Schottky Rectifiers Processed on Hot-Wall CVD and High-Temperature CVD Layers, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 921.
118. Peters, D., Schorner, R., Holzlein, K. H., and Friedrichs, P., Planar Aluminum-Implanted 1400 V 4H Silicon Carbide p-n Diodes with Low On Resistance, *Applied Physics Letters*, 71, 2996, 1997.
119. Ueno, K., Urushidani, T., Hashimoto, K., and Seki, Y., The Guard-Ring Termination for the High Voltage SiC Schottky Barrier Diodes, *IEEE Electron Device Letters*, 16, 331, 1995.
120. Itoh, A., Kimoto, T., and Matsunami, H., Excellent Reverse Blocking Characteristics of High-Voltage 4H-SiC Schottky Rectifiers with Boron-Implanted Edge Termination, *IEEE Electron Device Letters*, 17, 139, 1996.
121. Singh, R. and Palmour, J. W., Planar Terminations in 4H-SiC Schottky Diodes with Low Leakage and High Yields, *9th International Symposium on Power Semiconductor Devices and IC's*, IEEE, Piscataway, NJ, 1997, 157.
122. Ramungul, N., Khemka, V., Chow, T. P., Ghezzo, M., and Kretchmer, J., Carrier Lifetime Extraction from a 6H-SiC High-Voltage P-i-N Rectifier Reverse Recovery Waveform, Silicon Carbide, III-Nitrides, and Related Materials 1997, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 1065.
123. Konstantinov, A. O., Wahab, Q., Nordell, N., and Lindefelt, U., Ionization Rates and Critical Fields in 4H Silicon Carbide, *Applied Physics Letters*, 71, 90, 1997.
124. Harris, C. I., Konstantinov, A. O., Hallin, C., and Janzen, E., SiC Power Device Passivation Using Porous SiC, *Applied Physics Letters*, 66, 1501, 1995.
125. Alok, D., Baliga, B. J., and McLarty, P. K., A Simple Edge Termination for Silicon Carbide Devices with Nearly Ideal Breakdown Voltage, *IEEE Electron Device Letters*, 15, 394, 1994.
126. Alok, D. and Baliga, B., SiC Device Edge Termination Using Finite Area Argon Implantation, *IEEE Transactions on Electron Devices*, 44, 1013, 1997.
127. Raghunathan, R. and Baligà, B. J., EBIC Measurements of Diffusion Lengths in Silicon Carbide, *1996 Electronic Materials Conference*, Santa Barbara, CA, TMS, Warrendale, PA, 1996, 18.
128. Su, J. N. and Steckl, A. J., Fabrication of High Voltage SiC Schottky Barrier Diodes by Ni Metallization, Silicon Carbide and Related Materials 1995, *Institute of Physics Conference Series*, 142, Nakashima, S., Matsunami, H., Yoshida, S., and Harima, H., Eds., IOP Publishing, Bristol, United Kingdom, 1996, 697.
129. Brezeanu, G., Fernandez, J., Millan, J., Badila, M., and Dilimot, G., MEDICI Simulation of 6H-SiC Oxide Ramp Profile Schottky Structure, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 941.
130. Singh, R., Irvine, K. G., Kordina, O., Palmour, J. W., Levinshtein, M. E., and Rumyanetsev, S. L., 4H-SiC Bipolar P-i-N Diodes With 5.5 kV Blocking Voltage, *56th Annual Device Research Conference*, Charlottesville, VA, IEEE, Piscataway, NJ, 1998, 86.
131. Ghose, R. N., *EMP Environment and System Hardness Design*, D. White Consultants, Gainesville, VA, 1984, 4.1.
132. Kordina, O., Bergman, J. P., Hallin, C., and Janzen, E., The Minority Carrier Lifetime of N-Type 4H- and 6H-SiC Epitaxial Layers, *Applied Physics Letters*, 69, 679, 1996.
133. Held, R., Kaminski, N., and Niemann, E., SiC Merged P-N/Schottky Rectifiers for High Voltage Applications, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 1057.

134. Dahlquist, F., Zetterling, C. M., Ostling, M., and Rottner, K., Junction Barrier Schottky Diodes in 4H-SiC and 6H-SiC, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 1061.
135. Schoen, K. J., Henning, J. P., Woodall, J. M., Cooper, J. A., Jr., and Melloch, M. R., A Dual-Metal-Trench Schottky Pinch-Rectifier in 4H-SiC, *IEEE Electron Device Letters*, 19, 97, 1998.
136. Shenoy, P. M. and Baliga, B. J., The Planar 6H-SiC ACCUFET: A New High-Voltage Power MOSFET Structure, *IEEE Electron Device Letters*, 18, 589, 1997.
137. Hara, K., Vital Issues for SiC Power Devices, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 901.
138. Baliga, B. J., Prospects For Development of SiC Power Devices, Silicon Carbide and Related Materials 1995, *Institute of Physics Conference Series*, 142, Nakashima, S., Matsunami, H., Yoshida, S. and Harima, H., Eds., IOP Publishing, Bristol, United Kingdom, 1996, 1.
139. Hunter, G. W., Neudeck, P. G., Chen, L. Y., Knight, D., Liu, C. C., and Wu, Q. H., SiC-Based Schottky Diode Gas Sensors, Silicon Carbide, III-Nitrides, and Related Materials, *Materials Science Forum*, 264-268, Pensl, G., Morkoc, H., Monemar, B., and Janzen, E., Eds., Trans Tech Publications, Switzerland, 1998, 1093.
140. Lloyd Spetz, A., Baranzahi, A., Tobias, P., and Lundstrom, I., High Temperature Sensors Based on Metal-Insulator-Silicon Carbide Devices, *Physica Status Solidi (a)*, 162, 493, 1997.
141. Mehregany, M., Zorman, C., Narayanan, N., and Wu, C. H., Silicon Carbide MEMS for Harsh Environments, *Proceedings of the IEEE*, 14, 1998.
142. Nippon Steel Corporation, 5-10-1 Fuchinobe, Sagamihara, Kanagawa 229, Japan.
143. SiCrystal AG, Heinrich-Hertz-Platz 2, D-92275 Eschenfelden http://www.sicrystal.de.
144. Sterling Semiconductor, 22660 Executive Drive, Suite 101, Sterling, VA 20166, http://www.sterling-semicondcutor.com/.
145. Epitronics Corporation, 550 West Juanita Ave., Mesa, AZ 85210, http://www.epitronics.com.

Lotfi, A. “Passive Components”
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

7

Passive Components

Ashraf Lotfi
Bell Laboratories
Lucent Technologies

7.1 Magnetic Components

Integration Issues

It is well known and recognized that magnetic components should be avoided when designing integrated circuits due to their lack of integrability. New developments in the field of magnetic component fabrication are promising devices that can be integrated and miniaturized using monolithic fabrication techniques as opposed to today's bulk methods. The driving forces for such developments rest in certain applications that benefit or rely on inductive or magnetically coupled devices using ferromagnetic media. Examples of such applications include tuned RF tanks, matching networks, dc-dc power conversion and regulation, network filters, and line isolators/couplers.

Emerging applications requiring more mobility, lower power dissipation, and smaller component and system sizes have been drivers for the development of highly integrated systems and/or subsystems. In order to match these trends, it has become necessary to be able to integrate high-quality magnetic devices (i.e., inductors and transformers) with the systems they operate in as opposed to being stand-alone discrete devices. Not only does their discrete nature prevent further miniaturization, but their very nature also hampers improved performance (e.g., speed).

The main features of a monolithic magnetic device include:

1. High values of inductance compared to air core spirals
2. Enhanced high-frequency performance
3. Energy storage, dc bias, and power handling capabilities
4. Use of ferromagnetic materials as a magnetic core
5. Photolithographic fabrication of windings and magnetic core
6. Multi-layer mask fabrication for complete magnetic device design
7. Standard or semi-standard IC processing techniques

Due to the use of standard photolithography, etching, and patterning methods for their fabrication, monolithic magnetic devices may appear compatible with IC processes. However, two main characteristics make these devices more suitably fabricated off-line from a mainstream IC process:

- Coarser design rules. Usually, magnetic device designs do not require sub-micron geometries as demanded by semiconductor designs. This discrepancy means that an expensive sub-micron process for these components would unnecessarily raise the device cost.
- Use of ferromagnetic core materials. The use of iron, cobalt, nickel, and their alloys is at the heart of a high-quality magnetic device. Some of these materials are alien and contaminating to semiconductor cleanrooms. As a result, processing sequences and logistics for full integration with semiconductors is still in the development phase.

With these two major differences, integration of magnetics and semiconductors may require the use of multi-chip modules or single package multi-die cases. Full integration into a single monolithic die requires separate processing procedures using the same substrate.

The construction of a monolithic micromagnetic device fabricated on a substrate such as silicon or glass is shown in Fig. 7.1. In this diagram, the magnetic layer is sandwiched between upper and lower conductor layers that are connected together by means of an electrically conducting via. This structure is referred to as a *toroidal device* from its discrete counterpart.

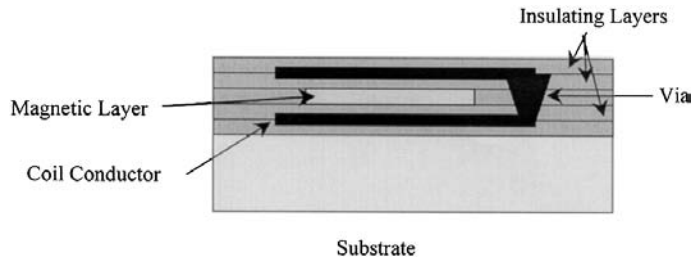


FIGURE 7.1 Cross-section of a monolithic micromagnetic device fabricated using IC methods.

Conversely, a dual structure can be made where two magnetic layers sandwich the conductor layer (or layers). This dual structure can be referred to as an *EE device* since it is derived from the standard discrete "EE" core type. In either case, as required by the operation of any magnetically coupled device, the magnetic flux path in the magnetic film and the current flow in the coil conductor are orthogonal, in accordance with Ampere's circuital law. Interlayer insulation between conductors and the magnetic layer(s) is necessary, both to reduce capacitive effects and to provide a degree of electrical voltage breakdown. These parameters are affected by the choice of insulator systems used in microfabricated circuits, due to the differing values of dielectric constants and breakdown voltages used. Some commonly used insulator systems include silicon dioxide and polyimide, each of which has distinctly different processing methods and physical characteristics. Conductor layers for the coil windings can be fabricated using standard aluminum metallization. In some cases, copper conductors are a better choice due to their higher conductivity and hence lower resistive losses. This is especially important if the device is to handle any significant power. The magnetic film layer is a thin film of chosen magnetic material typically between 1 and 10 μm in thickness. Such materials can be routinely deposited by standard techniques such as sputtering or electrodeposition. The specific method chosen must yield magnetic films with the desired properties, namely permeability, parallel loss resistance, and maximum flux density. These parameters vary with the deposition conditions and techniques, such that significant development and optimization has occurred to produce the desired results.

Since the design may call for energy storage and hence gaps in the core, the fabrication method can be modified to incorporate these features. Figure 7.2 shows the geometry of a planar magnetic core with a gap produced by photolithography. In this case, the gap is formed as a result of the artwork generated for the core design. Figure 7.3 shows the design of a gap using multi-layer magnetic films. The energy storage region exists in the edge insulation region between the two magnetic layers.

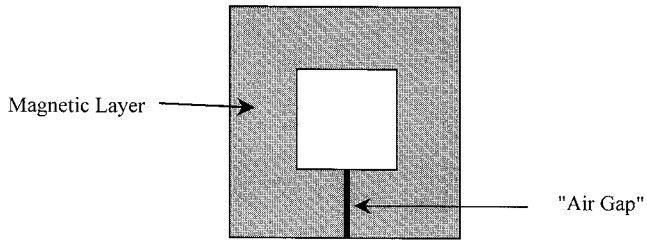


FIGURE 7.2 Planar, single-layer magnetic core configuration for energy storage (top view).

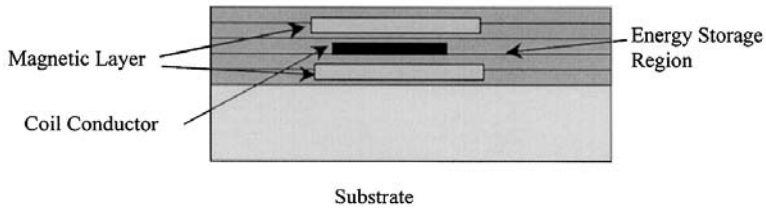


FIGURE 7.3 Multiple magnetic layers for energy storage micromagnetics (cross-sectional view).

The fabrication and construction of conductors for the coil usually involve depositing standard interconnect metals (e.g., aluminum) by sputter deposition. The thicknesses are chosen based on the current-carrying capability and the frequency of operation, as well as the desired configuration (inductor or transformer). The dc resistance of the conductors must be minimized to reduce dc losses, but the conductor thickness and arrangement must also result in minimal ac losses. This can be accomplished by reduced resistivity (i.e., copper vs. aluminum) and by multiple conductor layers to reduce skin and proximity effects at high frequencies.

Designs for Integrated Circuits

Unlike discrete magnetic components, monolithic micromagnetic devices are designed to operate at substantially higher frequencies. Due to their integrated nature and smaller physical size, interconnection and coupling parasitics are lower, thus enabling high-frequency response. However, the smaller physical size also places upper limits on such characteristics as inductance, current levels, power levels, and dissipation. With these limits, the maximum energy storage, *E*, is lower. In any inductor, the energy stored due to the current flowing (*I*) is related to the magnetic fields in the volume of the device by:

$$E = \frac{1}{2}LI^2 = \frac{1}{2}\iiint \bar{B} \cdot \bar{H} dv \tag{7.1}$$

where *L* is the transformer or inductor's inductance in Henries, and *I* (A) is the maximum current carried by the corresponding winding. This is related to the magnetic flux density, *B*, and magnetic field, *H*, present in the volume of the device. So, for a small physical volume, one can see from Eq. (7.1) that the energy stored is also small. This limited energy storage capability limits these devices to operation in low-power circuits. In order to obtain a high *B-H* product for more energy storage, a combination of high-permeability and low-permeability regions should be fabricated (i.e., a gap in the high permeability path is introduced). This gap region helps to maintain a high flux density as well as an appreciable field. The highly permeable region, on the other hand, while being able to maintain high flux density does not support large magnetic fields due to the fundamental relationship between magnetic field and flux:

$$\bar{B} = \mu_0 \mu_r \bar{H} \tag{7.2}$$

In Eq. (7.2), μ_0 is the permeability of vacuum ($4\pi \times 10^{-7}$ H/m) and μ_r is the relative permeability of the medium in which the magnetic field produces the corresponding magnetic flux density. The size of this gap determines both the energy storage levels and the inductance attainable (which is lower than the inductance attainable without a gap). In micromagnetic fabrication, two approaches may be taken to create this "air gap" region. One is to introduce a planar lithographical feature into the core structure (Fig. 7.2), and the other is to rely on multiple magnetic core layers separated by insulating layers (Fig. 7.3). The drawback of the lithographical gap is the limits imposed by the design rules. In this case, the gap may not be any smaller than the minimum design rule, which can be quite coarse. Excessive gap sizes result in very low inductance, requiring an increase in number of turns to compensate for this drop. Consequently, electrical losses in these windings increase and also the fabrication becomes more complicated. The drawback of multiple magnetic core layers is the need to add another level of processing to obtain at least a second (or more) magnetic layer(s). The stack-up of these layers and the edge terminations determine the amount of energy storage possible in the device. Unlike the lithographically produced gap, the energy storage in this case is much more difficult to estimate due to the two-dimensional nature of the edge termination fields in the gap region surrounding the multi-layer magnetic cores. In uniform field cases, the energy stored in volume of the gap can be obtained from Eqs. (7.1) and (7.2) due to the continuity and uniformity of the flux density vector in both the core and gap regions, giving:

$$E = \frac{1}{2} L I^2 \approx \frac{1}{2} \frac{B^2 V_{gap}}{\mu_0} \tag{7.3}$$

where V_{gap} is the volume of the gap region in cubic meters (m^3). The approximation is valid as long as the gap region carries a uniform flux density and is "magnetically long" compared to the length of the highly permeable core region (i.e., gap length/$\mu_{r\,gap}$ >> core length/$\mu_{r\,mag}$). Usually, this condition can be satisfied with most ferromagnetic materials of choice, but some ferromagnetic materials may have low enough permeabilities to render this approximation invalid. In this event, some energy is stored within the ferromagnetic material and Eq. (7.3) should be modified. Eq. (7.3) is very useful in determining the size of gap needed to support the desired inductance and current levels for the device. For example, if a 250-nH inductor operating at 250 mA of current bias were needed, the gap volume necessary to support these specifications would be about 2×10^{-5} mm^3, assuming a material with a maximum flux density of 1.0 T. In the planar device of Fig. 7.2 with nominal magnetic film dimensions of 2 μm in the normal direction and 200 μm in the planar direction, the required gap width would be about 5 μm. Since the gap in this case is obtained by photolithography, the minimum feature size for this process would need to be 5 μm. If a different material of lower maximum flux density capability of 0.5 T were used instead, the rated current level of 250 mA would have to be downgraded to 62 mA to prevent saturation of the magnetic material. Conversely, the gap length of 5 μm could be increased to 20 μm while maintaining the same current level, assuming adjustments are made to the turns to maintain the desired inductance. Such tradeoffs are common, but are more involved due to the interaction of gap size with inductance level and number of turns.

Another aspect of the design is the conductor for coil windings for an inductor or for primary and secondary windings in the case of a transformer. The number of turns is usually selected based on the desired inductance and turns ratio (for a transformer), which are typically circuit design parameters. As is well known, the number of turns around a magnetic core gives rise to an inductance, *L*, given by:

$$L = \frac{\mu_0 \mu_r N^2 A}{l} H \tag{7.4}$$

In this relation, N is the number of turns around a magnetic core of cross-sectional area A (m^2) and magnetic path length l (m). The inductance is reduced by the presence of a gap since this will serve to increase the path length. The choice of conductor thickness is always made in light of the ac losses occurring when conductors carry high-frequency currents. The conductors will experience various current redistribution effects due to the presence of eddy currents induced by the high-frequency magnetic fields surrounding the conductors. The well-known *skin effect* is one of such effects. Current will crowd toward the surface of the conductor and flow mainly in a thickness related to the skin depth, δ,

$$\delta = \frac{1}{\sqrt{\pi f \mu_0 \sigma}} \text{ m} \tag{7.5}$$

For a copper conductor, $\delta = 66/\sqrt{f(\text{MHz})}$ μm. At 10 MHz, the skin depth in copper is 20 μm, placing an upper limit on conductor thickness. When the interconnect metallization is aluminum, $\delta = 81/\sqrt{f(\text{MHz})}$ μm, so the upper limit at 10 MHz becomes 25 μm of metal thickness. Usually, the proximity of conductors to one another forces further optimization due to the introduction of losses due to eddy currents induced by neighboring conductors. In this case, the conductor thickness should be further adjusted with respect to the skin depth to reduce the induced eddy currents. The increase in conductor resistance due to the combined skin and proximity effects in a simple primary-secondary winding metallization scheme (shown in Fig. 7.4) can be calculated as an increase over the dc resistance of the conductor from:

$$Rac = Rdc \cdot \frac{1}{2}\frac{h}{\delta}\left\{\frac{\sinh h/\delta + \sin h/\delta}{\cosh h/\delta - \cos h/\delta} + \frac{\sinh h/\delta - \sin h/\delta}{\cosh h/\delta + \cos h/\delta}\right\} \tag{7.6}$$

In this relationship, h is the thickness of the metallization being used and Rac is obtained once the dc resistance (Rdc, also a function of h) is known. A distinct minimum for Rac can be obtained and yields the lowest possible ac resistance when:

$$\frac{h}{\delta} = \frac{\pi}{2} \tag{7.7}$$

with a corresponding minimum value of ac resistance of:

$$Rac = \frac{\pi}{2} Rdc \tanh\frac{\pi}{2} = 1.44 Rdc \tag{7.8}$$

When the geometry differs from the simple primary-to-secondary interface of Fig. 7.4 to more turns, layers, shapes, etc., a more complicated analysis is necessary.[4] The simple relation of Eq. (7.7) is no longer

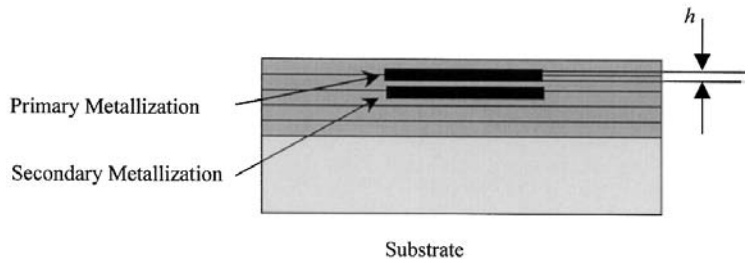


FIGURE 7.4 Configuration of primary and secondary transformer metallization for ac resistance calculation.

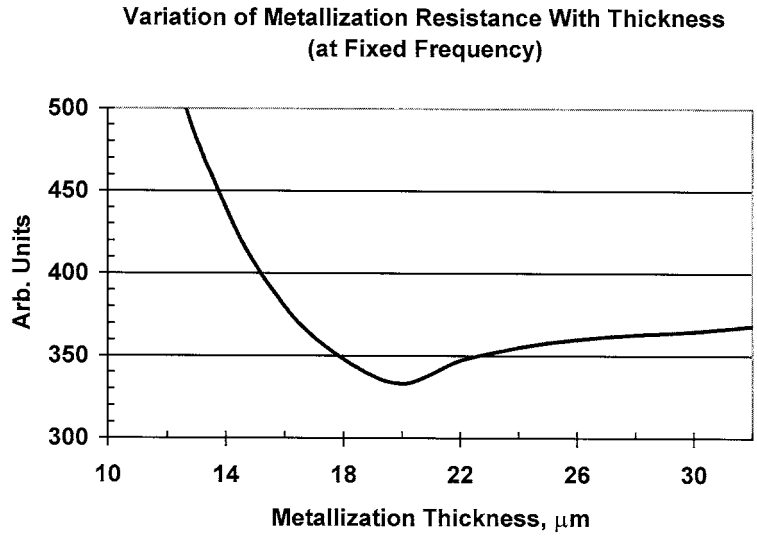


FIGURE 7.5 The variation of high-frequency metallization resistance with metal thickness for the configuration in Fig. 7.4. A distinct minimum is observed due to skin and proximity effects.

valid. Nevertheless, this relation provides a very good starting point for many designs. A qualitative explanation of this behavior stems from the fact that a thicker metallization will produce less dc resistance, but provides poor ac utilization due to current crowding near the surface. On the other hand, a thinner metallization increases the dc resistance, while providing better ac conductor utilization. The optimum situation is somewhere in between these two extreme cases, as can be seen from Fig. 7.5.

The principles presented in this section regarding design issues are at the core of every magnetic component design for integrated circuits. However, many design details — especially at elevated frequencies — are beyond the scope of this text. It is important to note that many of the limitations on high-frequency designs (100 MHz and higher) are imposed by the properties of the magnetic materials used in the cores of these devices.

Magnetic Core Materials

The most common magnetic materials used for discrete magnetic components operating at higher frequencies are ferrites. This is mainly due to their high resistivity (1 to 10 Ωm). Despite a low saturation flux density of about 0.3T, such a high resistivity makes ferrites suitable for applications up to 1 MHz, where hysteresis core losses are still limited. When the frequency is raised over 1MHz, the core losses become excessive, thus degrading the quality factor and efficiency of the circuit. Moreover, the permeability of all magnetic materials experiences a roll-off beyond a maximum upper frequency. Commonly used ferrites (e.g., MnZn ferrites) operate up to 1 to 2 MHz before permeability roll-off occurs. Higher roll-off frequencies are available, but with higher loss factors (e.g., NiZn ferrites). Ferrites, however, are not amenable to integrated circuit fabrication since they are produced by a high-temperature sintering process. In addition, their low flux saturation levels would not result in the smallest possible device per unit area. A set of more suitable materials for integrated circuit fabrication are the magnetic metal alloys usually derived from iron, cobalt, or nickel. These alloys can be deposited as thin films using IC fabrication techniques such as sputtering or electrodeposition and possess saturation flux levels of 0.8T to as high as 2.0T. Their main drawback due to their metallic nature is a much lower resistivity. Permalloy, a common magnetic alloy (80% nickel and 20% iron) has a resistivity of 20×10^{-8} Ωm, with a saturation flux density of 0.8T. Other materials such as sendust (iron-aluminum-silicon) have improved resistivity of 120×10^{-8} Ωm and saturation flux density of 0.95T.

To overcome the problem of low resistivity, the magnetic layers must be deposited in thin films with limited thickness. Since eddy currents flow in the metallic films at high frequencies, their effect can be greatly reduced by making the film thickness less than a skin depth. The skin depth in the magnetic film, δ_m, is given by:

$$\delta_m = \frac{1}{\sqrt{\pi f \mu_o \mu_r \sigma}} \text{ m} \tag{7.9}$$

In a thin film of permalloy ($\mu_r = 2000$), the skin depth at 10 MHz is 3 μm. In order to limit eddy current losses in the film, its thickness must be chosen to be less than 3 μm. This limitation will conflict with the inductance requirement, since a larger inductance requires a thicker magnetic film (see Eq. 7.4). Such difficulties can be overcome by depositing the magnetic film in multiple layers insulated from one another to restrict eddy current circulation. Such a structure would still provide the overall thickness needed to achieve the specified inductance while limiting the eddy current loss factor. The use of multi-layers also allows the reduction of die size due to the build-up of magnetic core cross-section (A in Eq. 7.4) in vertical layers, rather than by increasing the planar dimensions. As a result, it can be seen that a tradeoff exists between number of layers and die size to yield the most economical die cost.

In addition to eddy current losses due to the low magnetic metal resistivity, hysteresis losses occur in any magnetic material due to the traversing of the non-linear *B-H* loop at the frequency of operation. This is due to the loss of energy needed to rotate magnetic domains within the material. This loss is given by

$$P_{hys} = f \cdot \oint \bar{H} \cdot d\bar{B} \tag{7.10}$$

which is the area enclosed by the particular *B-H* loop demanded by the circuit operation and f is the frequency of operation. Total loss is expressed in many forms, depending on the application. In many cases, it is given in the form of a "parallel" or "shunt resistance" (Fig. 7.6) and it therefore presents a reduction in impedance to the source as well as a reduction in the overall quality factor of the inductor or transformer. It also represents a finite power loss since this loss is simply V^2/R_p watts, where V is the applied voltage.

Notice that R_p is a non-linear resistance with both frequency and flux level dependencies. It can be specified at a given frequency and flux level and is usually experimentally measured. It can also be extracted from core loss data usually available in the form

$$P = kf^{\alpha}B^{\beta} = \frac{V^2}{R_p} \tag{7.11}$$

In this relation, k, α, and β are constants for the material at hand. This model is useful for circuit simulation purposes, thereby avoiding the non-linear properties of the magnetic material. Care, however, should be exercised in using such models since with a large enough excitation, the value of the shunt resistor changes.

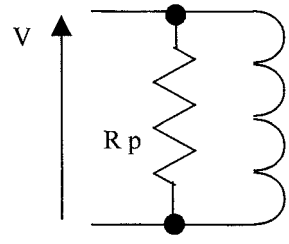


FIGURE 7.6 Magnetic material losses are represented by a parallel shunt resistance.

7.2 Air Core Inductors

Air core inductors do not use a magnetic core to concentrate the lines of magnetic flux. Instead, the flux lines exist in the immediate neighborhood of the coils without tight confinement. As a result, the inductance of an air core coil is considerably lower than one with a magnetic core. In fact, at low frequencies, the quality factor is reduced by a factor of μ_r when the loss factor is small. At high frequencies, however, the loss factor may be so high that the addition of a magnetic core and increased inductance actually ends up degrading the quality factor below the air core value. Discrete air core inductors have been used in RF applications and have been wound using discrete magnet or Litz wire onto forming cylinders.

For integrated circuits, especially RF and microwave circuits, spiral metallization deposited on a substrate is a common means to obtain small amounts of inductance with relatively high quality factors. Inductance values that can be obtained by these techniques are usually in the low nH range (1 to 20 nH). Estimating inductance values using air cores is much more complicated than in the case of highly permeable cores due to the lack of flux concentration. Formulas have been derived for different spiral air coil shapes (*assuming perfectly insulating substrates*) and are tabulated in several handbooks for inductance calculations.[6–9] An example of a useful inductance formula[9] is:

$$L = 4\pi \times 10^{-7} N^2 r_{avg} \left\{ \ln \frac{8 r_{avg}}{(r_{out} - r_{in})} + \frac{1}{24} \left(\frac{(r_{out} - r_{in})}{r_{avg}} \right)^2 \left(\ln \frac{8 r_{avg}}{(r_{out} - r_{in})} + 3.583 \right) - 0.5 \right\} \quad (7.12)$$

In this formula, r_{out} and r_{in} are the outer and inner radii of the spiral, respectively, and the average radius r_{avg} is

$$r_{avg} = \frac{(r_{out} + r_{in})}{2} \quad (7.13)$$

The formula is an approximation that loses accuracy as the device size becomes large (i.e., large r_{out} with respect to r_{in}).

The loss factors of such devices are strongly influenced by the non-idealities of the substrates and insulators on which they are deposited. For example, aluminum spiral inductors fabricated on silicon with highly doped substrates and epitaxial layers can have significant reductions in quality factor due to the conductivity of the underlying layers. These layers act as ground planes, producing the effect of an image of the spiral underneath. This in turn causes a loss in inductance. This can be as much as 30 to 60% when compared to a spiral over a perfect insulator. In addition, an increase in the loss factor occurs due to circulating eddy currents in the conductive under-layers. Increases in the effective resistance of 5 to 10 times the perfect insulator case are possible, increasing with increased frequency. All these effects can be seen to degrade the performance of these inductors, thus requiring design optimization.[10–12]

These substrate effects appear in the form of coupling capacitances from the spiral metal to the substrates, as well as spreading resistances in the substrate itself. The spreading resistance is frequency dependent, increasing with higher frequency. The amount of coupling to the substrate depends on the coupling capacitances and hence the separation of the spiral from the substrate. This distance is the dielectric thickness used in the IC process. Only with very large dielectric thicknesses are the substrate effects negligible. In practical cases where it is relatively thin and limited to a few microns, the effects are very large, giving an overall quality factor, Q, which is significantly lower than the Q of the spiral without the substrate. Fig. 7.7 shows a typical degradation curve of Q on a resistive substrate for "thick" and "thin" separations or dielectric thicknesses. The trends of this curve are also similar if the dielectric thickness variable is replaced by the substrate resistivity as a variable. The exact amount of degradation depends on the separation involved, the dielectric constant, and the resistivity of the substrate. With

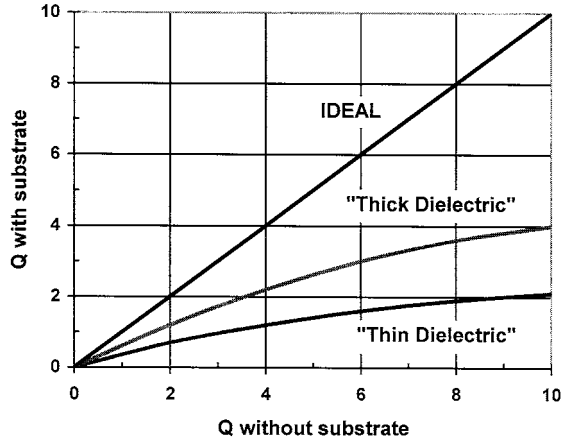


FIGURE 7.7 Degradation of inductor quality factor by placement on a resistive substrate.

these quantities known, it is possible to construct a circuit model to include these effects and hence solve for the overall quality factor, including the substrate effects.

In order to improve the inductor quality factor on a resistive substrate, some design solutions are possible. One solution to this problem is to increase the substrate resistivity. Another is to design a spiral with small footprint to reduce coupling to the substrate. In order to offset the increased resistance (which also reduces Q), thicker metallization would be necessary and clearly a tradeoff situation arises requiring some design optimization by circuit modeling or, more accurately, by electromagnetic finite-element analysis.

7.3 Resistors

Resistors have been available for use in integrated circuits for many years.[13–16] Some of these are made in silicon and thus are directly integrated with the rest of the IC process. Others, similar to the magnetic device case, are thin-film resistors fabricated in an off-line process that is not necessarily compatible with silicon IC processing. Integrated silicon resistors offer simplicity in fabrication but have less than ideal characteristics with loose tolerances. For this reason, many circuits rely on the ratio of resistor values rather than on their absolute values. Thin-film resistors, on the other hand, are far superior, offering tight tolerances and the ability to trim their absolute value down to very precise values. They also display more stability in terms of temperature and frequency dependence.[17]

Usually, resistors in integrated circuits are characterized in terms of their sheet resistance rather than their absolute resistance value. Sheet resistance, R_{sheet}, is defined as the resistance of a resistive strip with equal length and width so that

$$R_{sheet} = \frac{\rho}{t}(\Omega/\square) \tag{7.14}$$

where ρ is the material resistivity ($\Omega\cdot$m) and t is its thickness (m). Once R_{sheet} is given, the resulting resistor value is obtained by multiplying by its length-to-width aspect ratio. In order to avoid very high aspect ratios, an appropriate sheet resistivity should be used. For example, with $R_{sheet} = 10\ \Omega/\square$ a 10:1 length-to-width ratio would give a 100-Ω resistor. However, to obtain a 1-kΩ resistor, it would be better to use a different material with, for example, $R_{sheet} = 100\ \Omega/\square$ with the same 10:1 ratio instead of using a 100:1 ratio with the low-resistivity material.

Integrated Semiconductor Resistors

In this category, the existing semiconductor is used as the resistive material. The resistor may be fabricated at a number of stages during the IC process, giving rise to different resistors with different characteristics. Some of the most common include:

Diffused Resistors

This can be formed during either the base or emitter diffusion of a bipolar process. For an npn process, the base diffusion resistor is a p-type of moderate sheet resistivity, typically in the range of 100 to 200 Ω /□. This can provide resistors in the 50 to 10 kΩ range. The heavily doped n^+ emitter diffusion will produce an n^+-type resistor with low sheet resistivity of 2 to 10 Ω/□. This can provide resistors with low values in the 1 to 100 Ω range. Due to tolerances in the photolithographic and etching processes, the tolerance in the absolute resistance can be as high as range. Due to tolerances on the photolithographic and etching processes, the tolerance on the absolute resistance can be as high as ±30%. However, resistor pairs can be matched closely in temperature coefficients and doping profiles, especially when placed side-by-side on the chip, so that the resultant tolerance of the resistor ratio can be made to be less than ±1%. Since a diffusion resistor is based on a p-type base over an n-type epitaxy, or an n^+-type emitter over a p-type base, it is essential that the formed p-n junctions are always reverse-biased to ensure that current flows in the intended portion of the resistor. The presence of such a reverse-biased p-n junction also introduces a distributed capacitance from the resistor body to the substrate. This will cause high-frequency degradation, whereby the resistor value drops from its nominal design value to a lower impedance value due to the shunting capacitance.

Pinched Resistors

A variation to the diffused resistor that is used to increase the sheet resistivity of base region is to use the n^+-type emitter as a means to reduce the cross-sectional area of the base region, thereby increasing the sheet resistivity. This can increase the sheet resistance to about 1 kΩ/□. In this case, one end of the n^+-type emitter must be tied to one end of the resistor to contain all current flow to the pinched base region.

Epitaxial Resistors

High resistor values can be formed using the epitaxial layer since it has higher resistivity than other regions. Epitaxial resistors can have sheet resistances around 5 kΩ /□. However, epitaxial resistors have even looser tolerances due to the wide tolerances on both epitaxial resistivity and epitaxial layer thickness.

MOS Resistors

A MOSFET can be biased to provide a non-linear resistor. Such a resistor provides much greater values than diffused ones while occupying a much smaller area. When the gate is shorted to the drain in a MOSFET, a quadratic relation between current and voltage exists and the device conducts current only when the voltage exceeds the threshold voltage. Under these circumstances, the current flowing in this resistor (i.e., the MOSFET drain current) depends on the ratio of channel width-to-length. Hence, to increase the resistor value, the aspect ratio of the MOSFET should be reduced to give longer channel length and narrower channel width.

Thin-Film Resistors

As mentioned before in the magnetic core case, a resistive thin-film layer can be deposited (e.g., by sputtering) on the substrate to provide a resistor with very tight absolute-value tolerance. In addition, given a large variety of resistor materials, a wide range of resistor values can be obtained in small footprints, thereby providing very small parasitic capacitances and small temperature coefficients. Some common thin-film resistor materials include tantalum, tantalum nitride, and nickel-chromium. Unlike semiconductor resistors, thin-film resistors can be laser trimmed to adjust their values to very high accuracies of up to 0.01%. Laser trimming can only increase the resistor value since the fine beam

evaporates a portion of the thin-film material. By its nature, laser trimming is a slow and costly operation that is only justified when very high accuracy on absolute values is necessary.

7.4 Capacitors

As in the inductor case, the limitation on integrated capacitors is die size, due to the limited capacitance/unit area available on a die. These limitations are imposed by the dielectrics used with their dielectric constants and breakdown voltages. Most integrated capacitors are either junction capacitors or MOS capacitors.

Junction Capacitors

A *junction capacitor* is formed when a p-n junction is reversed-biased. This can be formed using the base–emitter, base–collector, or collector–substrate junctions of an npn structure in bipolar ICs. Of course, the particular junction must be maintained in reverse-bias to provide the desired capacitance. Since the capacitance arises from the parallel plate effect across the depletion region, whose thickness in turn is voltage dependent, the capacitance is also voltage dependent, decreasing with increased reverse-bias. The capacitance depends on the reverse-voltage in the following form:

$$C(V) = \frac{C_0}{(1 + V/\psi_0)^n} \tag{7.15}$$

The built-in potential, Ψ_0, depends on the impurity concentrations of the junction being used. For example, $\Psi_0 = 0.7$ V for a typical bipolar base–emitter junction. The exponent n depends on the doping profile of the junction. The approximations $n = 1/2$ for a step junction and $n = 1/3$ for a linearly graded junction are commonly used. The resultant capacitance depends on C_0, the capacitance per unit area with zero bias applied. This depends on the doping level and profile. The base–emitter junction provides the highest capacitance per unit area, around 1000 pF/mm^2 with a low breakdown voltage (~5 V). The base–collector junction provides about 100 pF/mm^2 with a higher breakdown voltage (~40 V).

MOS Capacitors

MOS capacitors are usually formed as parallel plate devices with a top metallization and a high conductivity n^+ emitter diffusion as the two plates, with a thin oxide dielectric sandwiched in between. The oxide is usually a thin layer of SiO_2 with a relative dielectric constant ε_r of 3 to 4, or Si_3N_4 with ε_r of 5 to 8. Since the capacitance obtained is $\varepsilon_0\varepsilon_r A/t_{oxide}$, the oxide thickness, t_{oxide} is critical. The lower limit on the oxide thickness depends on the process yields and tolerances, as well as the desired breakdown voltage and reliability. MOS capacitors can provide around 1000 pF/mm^2, with breakdown voltages up to 100 V. Unlike junction capacitors, MOS capacitors are voltage independent and can be biased either positively or negatively. Their breakdown, however, is destructive since the oxide fails permanently. Care should be taken to prevent overvoltage conditions.

References

1. Saleh, N. and Qureshi, A., "Permalloy thin-film inductors," *Electronics Letters*, vol. 6, no. 26, pp. 850-852, 1970.
2. Soohoo, R., "Magnetic film inductors for integrated circuit applications," *IEEE Trans. Magn.*, vol. MAG-15, pp. 1803, 1979.
3. Mino, M. et al., "A new planar microtransformer for use in micro-switching converters," *IEEE Trans. Magn.*, vol. 28, pp. 1969, 1992.

4. Vandelac, J. and Ziogas, P. , "A novel approach for minimizing high frequency transformer copper loss," *IEEE Trans. Power Elec.*, vol. 3, no. 3, pp. 266-76, 1988.
5. Sato, T., Tomita, H., Sawabe, A., Inoue, T., Mizoguchi, T., and Sahashi, M., "A magnetic thin film inductor and its application to a MHz Switching dc-dc converter," *IEEE Trans. Magn.*, vol. 30, no. 2, pp. 217-223, 1994.
6. Grover, F., *Inductance Calculations*, Dover Publishing, New York, 1946.
7. Welsby, V., *Theory and Design of Inductance Coils*, MacDonald & Co., London, 2nd ed., 1960.
8. Walker, C., *Capacitance, Inductance, and Crosstalk Analysis*, Artech House, Boston, 1990.
9. Gupta, K. C., Garg, R., and Chadha, R., *Computer-Aided Design of Microwave Circuits*, Artech House, Dedham, MA, 1981.
10. Remke, R. and Burdick, G.,"Spiral inductors for hybrid and microwave applications," *Proc. 24th Electron Components Conf.*, May 1974, pp. 152-161.
11. Arnold, R. and Pedder, J.,"Microwave characterization of microstrip lines and spiral inductors in MCM-D technology," *IEEE Trans. Comp., Hybrids, and Manuf. Tech.*, vol. 15, pp. 1038-45, 1992.
12. Nguyen, N. M. and Meyer, R. G.,"Si IC-compatible inductors and LC passive filters," *IEEE J. Solid-State Circuits*, vol. 25, pp. 1028-1031, Aug. 1990.
13. Glaser, A. and Subak-Sharpe, G., *Integrated Circuit Engineering*, Addison-Wesley, 1977.
14. Goodge, M. E., *Semiconductor Device Technology*, Howard Sams & Co., Inc., Indiana, 1983.
15. Grebene, A. B., *Bipolar and MOS Analog Integrated Circuit Design*, John Wiley, New York, 1984.
16. Gray, P. R. and Meyer, R. G., *Analysis and Design of Analog Integrated Circuits*, John Wiley, New York, 1993.
17. Sergent, J. E. and Harper, C. A., *Hybrid Microelectronics Handbook*, McGraw-Hill, New York, 1995.
18. Levy, R. A., *Microelectronic Materials and Processes*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1989.

Nakagawa, A. “Power IC Technologies”
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

8

Power IC Technologies

Akio Nakagawa
Toshiba Corporation

8.1 Introduction

VLSI technology has advanced so greatly that Gigabit DRAMs have become a reality, and the technology faces an optical lithography limit. Microelectronics mostly advances signal processing LSIs such as memories and microprocessors. Power systems and the related circuits cannot be outside the influence of VLSI technology.[1] It would be quite strange for power systems alone to still continue to consume a large space while brains become smaller and smaller. On the other hand, almost all of the systems require actuators or power devices to control motors, displays, and multimedia equipment. The advances in microelectronics have made it possible to integrate large-scale circuits in a small silicon chip, ending up in high system performance and resultant system miniaturization. The system miniaturization inevitably necessitated power IC development. Typical early power ICs were audio power amplifiers, which used bipolar transistors as output devices. The pn junction isolation method was well suited to integrate bipolar transistors with control circuits.

Real advancements in intelligent power ICs were triggered by the invention of power DMOSFETs[2] in the 1970s. DMOS transistors have ideal features for output devices of power ICs. No driving dc current is necessary, and large currents can be controlled simply by changing the gate voltage. In addition, DMOS switching speed is sufficiently fast.

The on-resistance of vertical DMOSFETs has been greatly reduced year by year with advances in fine lithography in LSI technology. In the mid-1980s, the new concept "Smart Power"[3] was introduced. Smart Power integrates bipolar and CMOS devices with vertical DMOS, using a process primarily optimized for poly-silicon gate self-aligned DMOS. The main objective is to integrate control and protection circuits with vertical power devices, not only to increase device reliability and performance, but also to realize easy use of power devices. The concept of Smart Power was applied to high-voltage vertical DMOS with

drain contact on the back side of the chip because discrete DMOS technology was already well advanced in the early 1980s. The main application field was automotive, replacing mechanical relays and eliminating wire harnesses.

As the technology of microlithography has further advanced, the on-resistance of DMOS, especially low-voltage DMOS, has continuously decreased. In the early 1990s, the on-resistance of low-voltage lateral DMOS became lower than that of bipolar transistors.[4] It was even realized that low-voltage lateral DMOS is superior to vertical planar discrete DMOS since fine lithography does not contribute to a decrease in on-resistance of vertical DMOS because of JFET resistance. Recently, with the introduction of a 0.6-μm design rule, lateral DMOS has become predominant over the wide voltage range — from 20 V up to 150 V. Mixed technology, called BCD,[4] integrating BiCMOS and DMOS, is now widely accepted for low-voltage power ICs.

For high-voltage power ICs, DMOS is not suitable for output devices because of a high on-resistance. Thyristor-like devices, such as GTOs, have conventionally been used for high-voltage applications. Integration of thyristor-like devices needs a method of dielectric device isolation (DI). The conventional DI method, called EPIC,[5] has been used for high-voltage telecommunication ICs, called SLIC. However, it has problems of high cost and large wafer warpage. In 1985 and 1986, wafer direct-bonding technology was invented,[6,7] and low-cost DI wafers became available. Wafer warpage of directly bonded SOI wafers is very small. This made it possible not only to fabricate large-diameter (8-in.) SOI wafers, but also to apply advanced lithography to DI power ICs. The chip size of DI power ICs can be reduced by narrow-trench isolation and by the use of high-performance lateral IGBTs. The low-cost DI wafers and the chip size reduction have widened the application fields of DI power ICs, covering automotive, motor control, and PDP drivers.

8.2 Intelligent Power ICs

Figure 8.1 shows typical functions integrated into intelligent power ICs. The most important feature of the intelligent power IC is that a large power can be controlled by logic-level input signals and all the cumbersome circuits such as driving circuits, sense circuits, and protection circuits required for power device control are inside the power ICs.

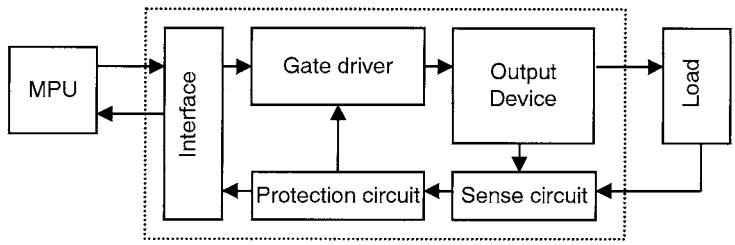


FIGURE 8.1 Typical integrated functions in intelligent power ICs.

Technologies for realizing such power ICs are classified into three categories, as shown in Figs. 8.2 to 8.4. These are self-isolation, junction isolation, and dielectric isolation. Self-isolation is a method that does not use any special means to isolate each device, and each device structure automatically isolates itself from the other.

Junction isolation (JI) is a method that uses reverse-biased junction depletion layers to isolate each device. JI is the most frequently used method for low-voltage power ICs, using bipolar transistor or DMOS outputs.

Junction isolation is not sufficient to isolate IGBTs or thyristors. Dielectric isolation is a method that uses silicon dioxide film to isolate devices and thus offers complete device isolation. Although the EPIC method has conventionally been used, the high cost of wafer fabrication has been a problem. Recently, wafer direct-bonding technology was invented and bonded SOI wafers are available at a lower price.

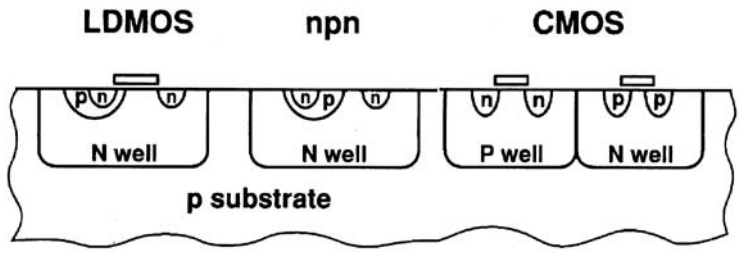


FIGURE 8.2 Self-isolation technology.

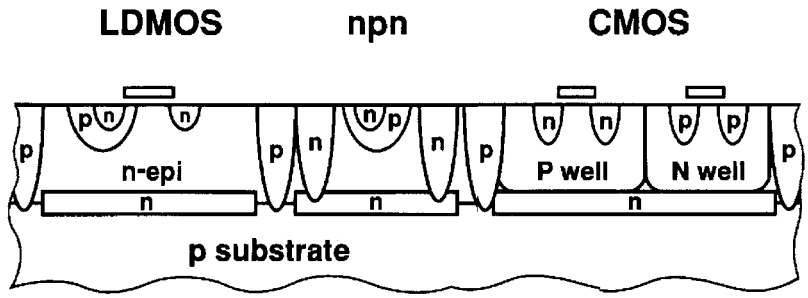


FIGURE 8.3 Junction isolation (JI) technology.

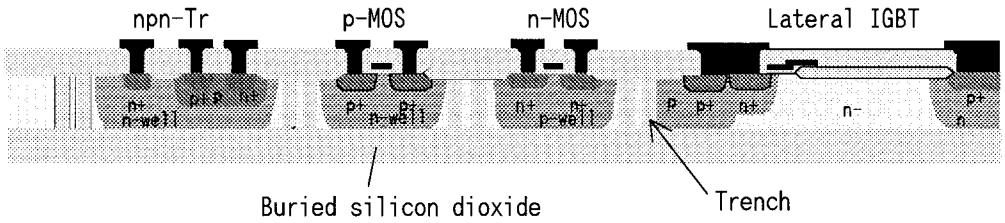


FIGURE 8.4 Dielectric isolation (DI) technology.

pn Junction Isolation

One of the fundamental issues in integrated circuits is how to electrically isolate each device from the others. pn junction isolation is the most familiar method and has been used since the beginning of bipolar IC history. Figure 8.5 shows the cross-section of a typical junction isolation structure. First, an n-type

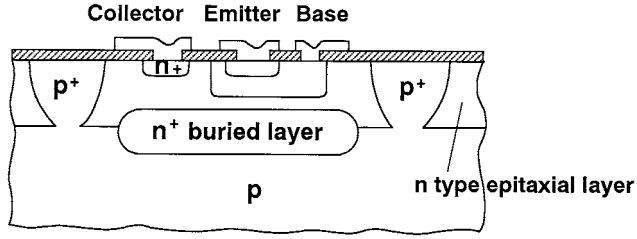


FIGURE 8.5 Junction isolation structure.

epitaxial layer is formed on p-type silicon substrate. p-type diffusion layers are then formed to reach the p-type substrate, resulting in isolated n-type islands surrounded by p-type regions. By keeping the substrate potential in the lowest level, the pn junctions, surrounding the islands, are reverse-biased and the depletion layers are formed to electrically isolate each island from the others.

If this method is applied to high-voltage power ICs, a thick n-type epitaxial layer is required and deep isolation diffusions are necessary. Deep diffusion accompanies large lateral diffusion, ending up in a large isolation area. One solution for this is to use buried p^+ diffusion layers for upward isolation diffusions, as shown in Fig. 8.6. However, 200 V is a practical limit for conventional pn junction isolation.

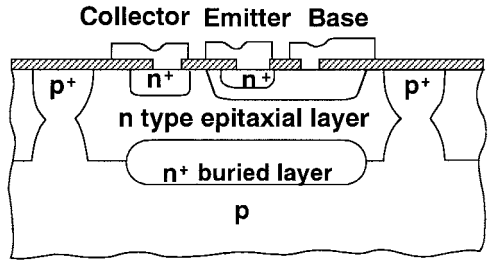


FIGURE 8.6 Junction isolation with upward isolation diffusions.

A variety of methods have been proposed to overcome this voltage limit. Figure 8.7 shows a typical example for this.[8] A shallow hole is formed where a high-voltage device is formed before the n-type epitaxial growth. This allows a locally thicker n-type epitaxial layer for high-voltage transistors.

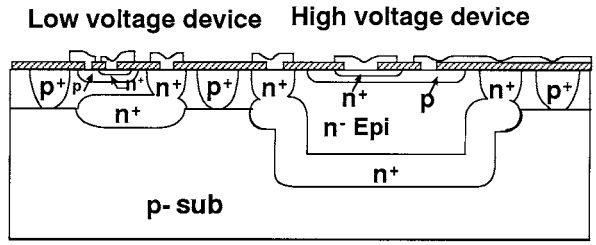


FIGURE 8.7 An example to overcome the junction isolation voltage limit.

Another distinguished example is shown in Fig. 8.8, where an n^+-substrate is used in place of a p-type substrate. p-type and n-type epitaxial layers are subsequently formed. This example makes it possible to integrate a vertical DMOSFET with a backside drain contact with junction-isolated BiCMOS control circuits. This structure was proposed as "Smart Power" in the mid-1980s.

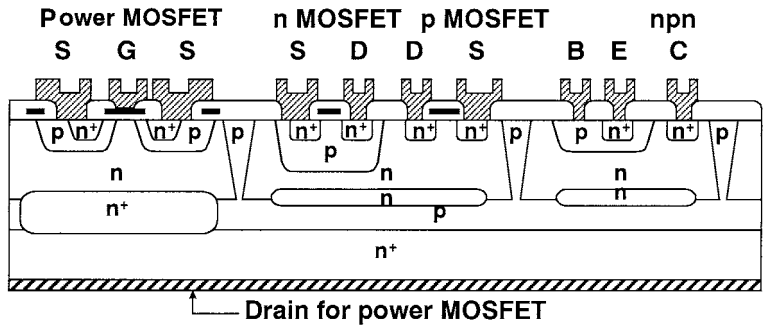


FIGURE 8.8 Junction isolation with VDMOSFET.

Impact of Dielectric Isolation

Dielectric isolation (DI) is a superior method for integrating many kinds of devices in a single chip. DI has many advantages[9–11] over junction-isolation techniques, including,

1. Virtually all integrated components can be treated as if they were discrete devices, so that circuit design becomes easy.
2. Bipolar devices, including thyristors, can be integrated without any difficulties.
3. Coupling between two devices can be minimized, thus attaining better IC performances: no latch-up, high speed, large noise immunity, and ruggedness.
4. High-temperature operation is feasible because there are virtually no parasitics and leakage current is low.
5. Radiation hardness for space use.

Figure 8.9 shows a cross-section of the conventional DI, called EPIC. The crystalline silicon islands completely surrounded by silicon dioxide film are floating in the supporting substrate made of a thick polysilicon layer. The fabrication process of EPIC wafers is complicated and illustrated in Fig. 8.10. The problem with EPIC is the high cost of wafers and large wafer warpage. The development of the EPIC method was initiated by the early works of J.W. Lathlop et al.,[12] and J. Bouchard et al.,[13] in 1964. The EPIC method was first applied to high-speed bipolar ICs owing to its low parasitic capacitance.

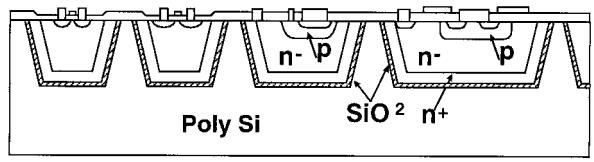


FIGURE 8.9 Dielectric isolation with EPIC technology.

Early work on high-voltage integrated circuits was triggered by the need for display drivers and high-voltage telecommunication circuits. Efforts to achieve high-voltage lateral MOSFETs started in the early 1970s, and the 800-V lateral MOSFET, using RESURF concept and DMOS (DSA[2]) technology, was developed for display drivers in 1976, before the RESURF concept was fully established.[14]

The need for high-voltage SLICs advanced the EPIC technology because it required electrically floating high-voltage bi-directional switches, which were realized only by the DI technique.

A variety of dielectric isolation methods, classified as silicon on insulator (SOI) technology, were invented in the 1970s. These are SOS (silicon on sapphire[15]), SIMOX,[16] and recrystallized poly-silicon such as ZMR.[17] And, silicon wafer direct-bonding (SDB)[6,7] was proposed in 1985.

The SOI wafer structure is simple. A single crystalline silicon layer is formed on the buried oxide layer or insulator substrate. Major methods are SIMOX and wafer bonding. SIMOX is a method that forms a buried oxide layer by a high dose of oxygen ion implantation and subsequent high-temperature annealing. Wafer bonding is a method that bonds an oxidized wafer and a substrate wafer at room temperature and strengthens the bond by annealing at high temperature. The thickness of the bonded SOI layer is adjusted by mechanical grinding and polishing.

In the late 1980s, MOS gate power device technology was greatly improved. In particular, the success of the MOS bipolar composite devices such as IGBTs[18,19] and MCTs[20] made it possible to control a large current by the MOS gate. The large current-handling capability of IGBTs has accelerated adopting DI with IGBT outputs.

In the early SLICs, double-injection devices with current control gates such as gated diodes and GTOs were used for such switches.[21] Recently developed SLICs (telecommunication ICs) have adopted lateral IGBTs or MOS gated thyristors because of the ease of gate drive. All the commercialized SLICs, so far,

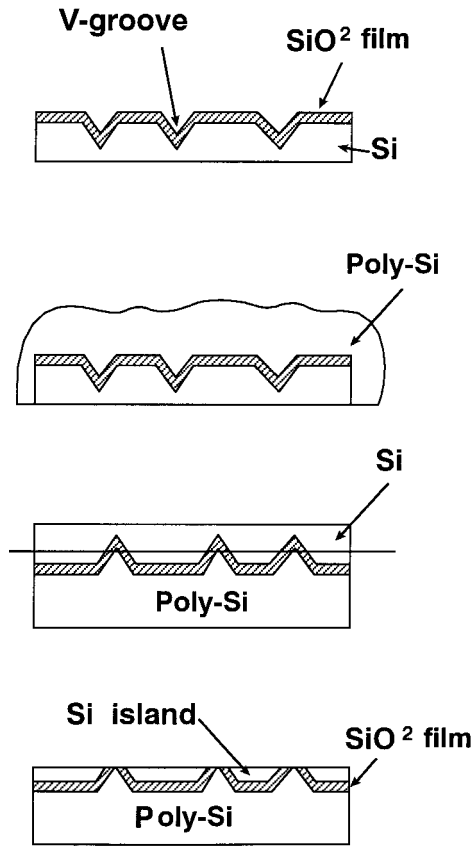


FIGURE 8.10 Fabrication process of EPIC wafers. A very thick poly-crystalline silicon layer is deposited on oxidized single-crystal silicon with a grooved surface. The crystalline silicon is grounded and polished so that the silicon layers isolate each other by the grooves.

have adopted the conventional DI method. The success of SLIC was supported by the fact that monolithic integration and added function deserved expensive DIs for telecommunications application.

In the 1990s, wafer bonding technology was well established and low-cost SOI wafers were made available. A low-cost DI method realized by SOI technology, using several micron thick or less silicon layers, changed the situation of DI research and widened the application fields.

If the silicon layer is thin, devices in the SOI layer are isolated with narrow trenches. This makes SOI technology very attractive for high-voltage applications because chip size can be reduced and resultant chip cost is reduced. The SOI technology widened the application field of DI toward consumer use.

High-voltage SOI research work started in the early 1990s. Research efforts have been directed toward:

1. Monolithic device integration of multiple number of high-voltage, high-current devices with control circuits
2. ICs allowing high temperature operation and ruggedness
3. Low-cost DI power IC process development
4. High-current, high-speed, MOS-controlled lateral output devices with self-protection functions

8.3 High-Voltage Technology

It is quite important to realize a high breakdown voltage in an integrated device structure. There are two major techniques for high-voltage power ICs. These are the field plate and resurf techniques.

Field Plate

It is very important to realize a high breakdown voltage in a planar device structure. In other words, it is ideal if a one-dimensional pn junction breakdown voltage is realized in an actual pn junction, formed by thermal impurity diffusion. Actual pn junctions consist of cylindrical junctions and spherical junctions near the surface. Generally, the breakdown voltage of cylindrical or spherical junctions is significantly lower than that of an ideal 1-D planar junction, if junction curvature is small.

A *field plate* is a simple and frequently used technique to increase the breakdown voltage of an actual planar junction. Figure 8.11 shows an example. Field plates, placed on the thick-field oxide, induce depletion layers underneath themselves. The curvature of the formed depletion layers can be increased with the induced depletion layers, thereby relaxing the curvature effects of the field plate.

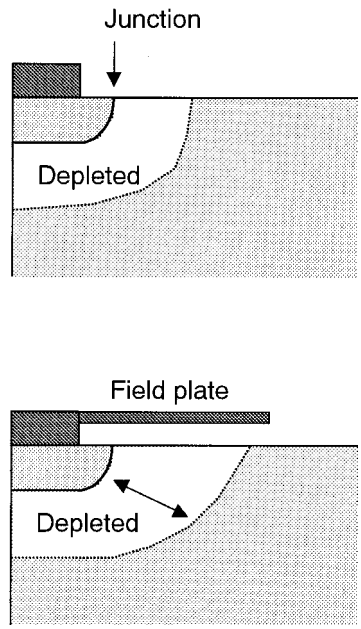


FIGURE 8.11 Field plate structure.

Resurf Technique

The *resurf* technique was originally proposed in 1979[14] as a method to obtain a high breakdown voltage in a conventional JI structure, where the breakdown voltage is limited by the thickness of the epitaxial layer. Figure 8.12 shows a high-voltage structure, where the depletion layer develops in the p-substrate and n-epitaxial-layer. If the epi-layer is thick or impurity doping is high (a), breakdown occurs before n-epi layer is completely depleted. If an appropriate epi-layer thickness is chosen (b), the epi-layer is completely depleted when breakdown occurs. The achieved breakdown voltage is very high because the depletion layer is sufficiently thick, both in lateral direction and vertical direction. The important point is that the total charge Q_c in the epi-layer is chosen so that the value satisfies the equation:

$$Q_c = \varepsilon E_c \tag{8.1}$$

where E_c denotes critical electric field in silicon (3×10^5 V/cm). This charge can be depleted just when the electric field becomes E_c or breakdown occurs. In other words, the epi-layer is completely depleted just when breakdown occurs, if the total epi-layer dose is Q_c/q, which is approximately $2 \times 10^{12}/cm^2$.

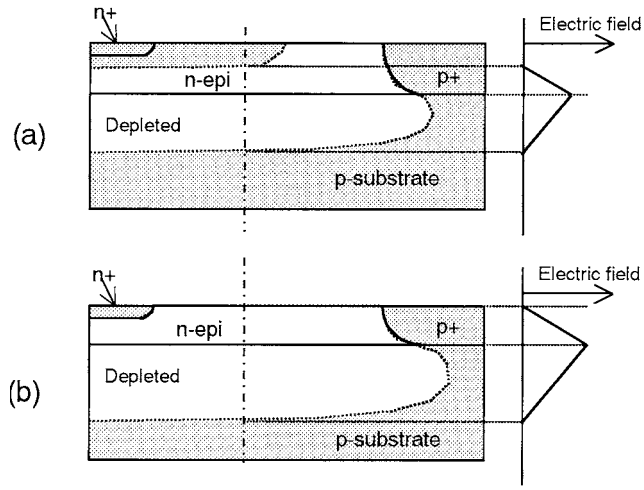


FIGURE 8.12 Resurf technique.

8.4 High-Voltage Metal Interconnection

In high-voltage power ICs, there must be interconnection layers crossing high-voltage junctions. These high-voltage interconnection layers may cause degradation of the breakdown voltage of high-voltage devices. These problems are often solved with a thicker insulator layer under the interconnection layers. However, special means are required if the breakdown voltage is over 400 V.

Figure 8.13 shows one of the methods to shield the influence of metal interconnection layers on the underlying devices. A spiral-shaped high-resistance poly-silicon layer, connecting source and drain electrodes, effectively shields the influence of the interconnection layer on the depletion layer.[22] This is because the potential of the high-resistance poly-silicon layer is determined by small leakage current.

Another typical example is multiple floating field plates. The cross-section of the structure is similar to Fig. 8.13. The difference is that the poly-silicon forms multiple closed field rings, which are electrically floating each other. Multiple floating field plates also prevent breakdown voltage reduction due to metal interconnection.

8.5 High-Voltage SOI Technology

SOI power ICs are classified into two categories from the viewpoint of SOI wafer structure. The difference is whether there is a buried n^+ layer on the buried oxide. Figure 8.14 shows a typical device structure, employing an n^+ buried layer on the buried oxide. The breakdown voltage is determined with the thickness of the high-resistivity n-layer or the thickness of the depletion layer. The maximum breakdown voltage is limited to below 100 V because of SOI layer thickness or practically available trench depth. For this case, SOI wafers are used as a simple replacement for conventional DI wafers.

Figure 8.15 shows another typical SOI power IC structure, employing a high-voltage lateral IGBT. The n^- drift layer is fully depleted by application of a high voltage. As the buried oxide and the depletion layer both share the applied voltage, high breakdown voltage is realized in a relatively thin SOI. This type of power IC fully enjoys the features of SOI technology:

1. Complete device isolation by trench technique and small isolation region
2. Virtually no parasitic active component
3. A high breakdown voltage exceeding 500 V is realized by applying a large portion of the voltage across the thick buried oxide

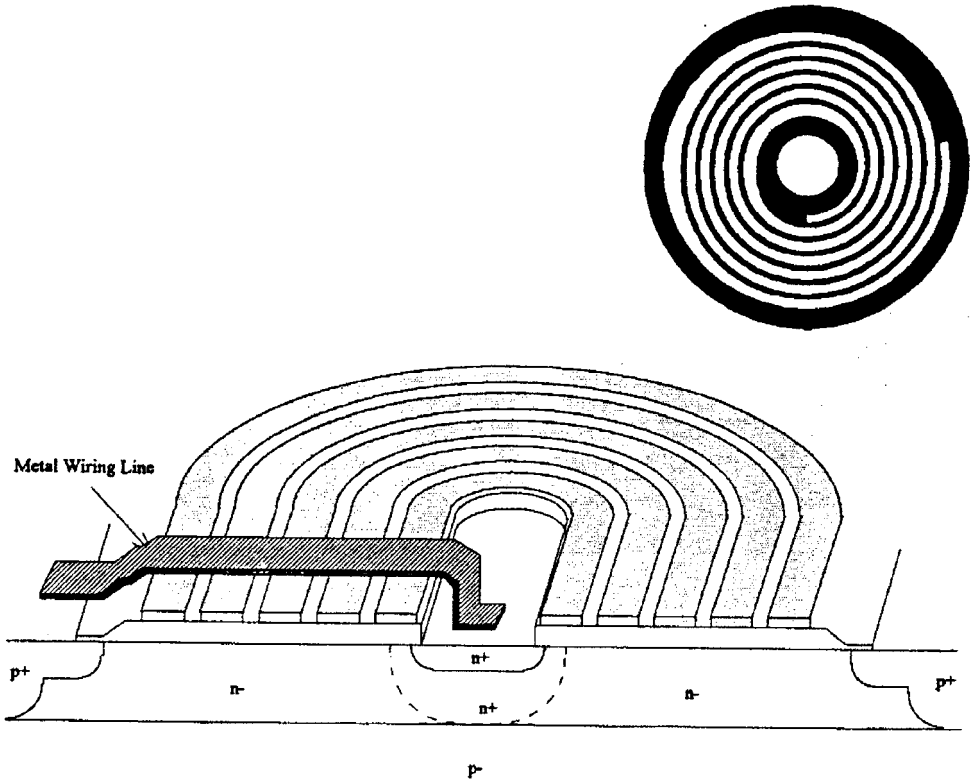


FIGURE 8.13 A method to shield the influence of the metal interconnection layer. (Copyright (1994) IEEE. With permission.)

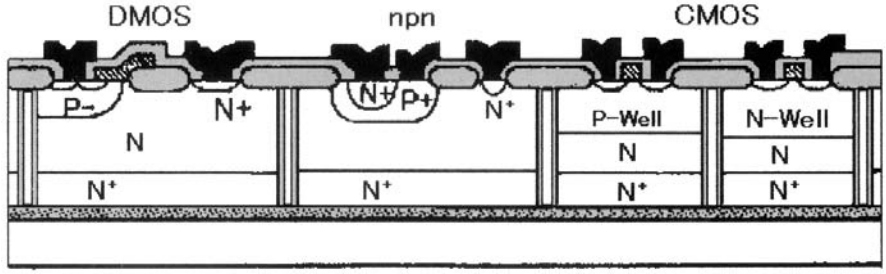


FIGURE 8.14 A high-voltage SOI device structure with n^+ buried layer on the buried oxide.

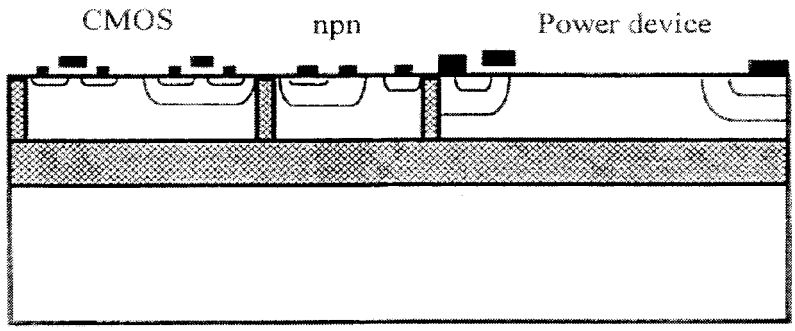


FIGURE 8.15 A high-voltage SOI device structure without n^+ buried layer on the buried oxide.

4. Small wafer warpage and fine lithography is applicable
5. High-temperature operation is possible

There are two big issues associated with high-voltage devices on SOI.[11] One is how to realize a high breakdown voltage under the influence of substrate ground potential. The other is how to attain a low on-resistance with a thin silicon layer. In the conventional DI, the wrap-around n^+ region (see Fig. 8.14) is used in the DI island to prevent the influence of substrate potential on the device breakdown voltage. However, for thin silicon layers, this method cannot be used. The bottom silicon dioxide layer simply works as an undoped layer as far as the Poisson equation is concerned. Thus, a SOI layer on a grounded silicon substrate structure behaves in a way similar to the structure of a doped n-type thin silicon layer on undoped silicon layer (corresponding to silicon dioxide) on a grounded p silicon substrate. Thus, the SOI layer works in the same way as a resurf layer.

A high breakdown voltage of a thin silicon layer device can be realized by sharing a large applied voltage with the buried dioxide film, whose breakdown field is far greater than that of silicon. The buried oxide film is able to sustain a large share of applied voltage, because the dielectric breakdown field is larger than that of silicon.

Figure 8.16 shows a typical SOI diode structure and its potential distribution. It is seen that almost a half of the voltage is applied across the buried oxide. Figure 8.17 shows the electric field distribution along the symmetry axis of the diode of Fig. 8.16. The electric field in the oxide is larger than that in silicon because the following relation holds.

The two electric field components $E_t(Si)$, $E_t(I)$, normal to the interface of the silicon and the bottom insulator layer, have the relation:

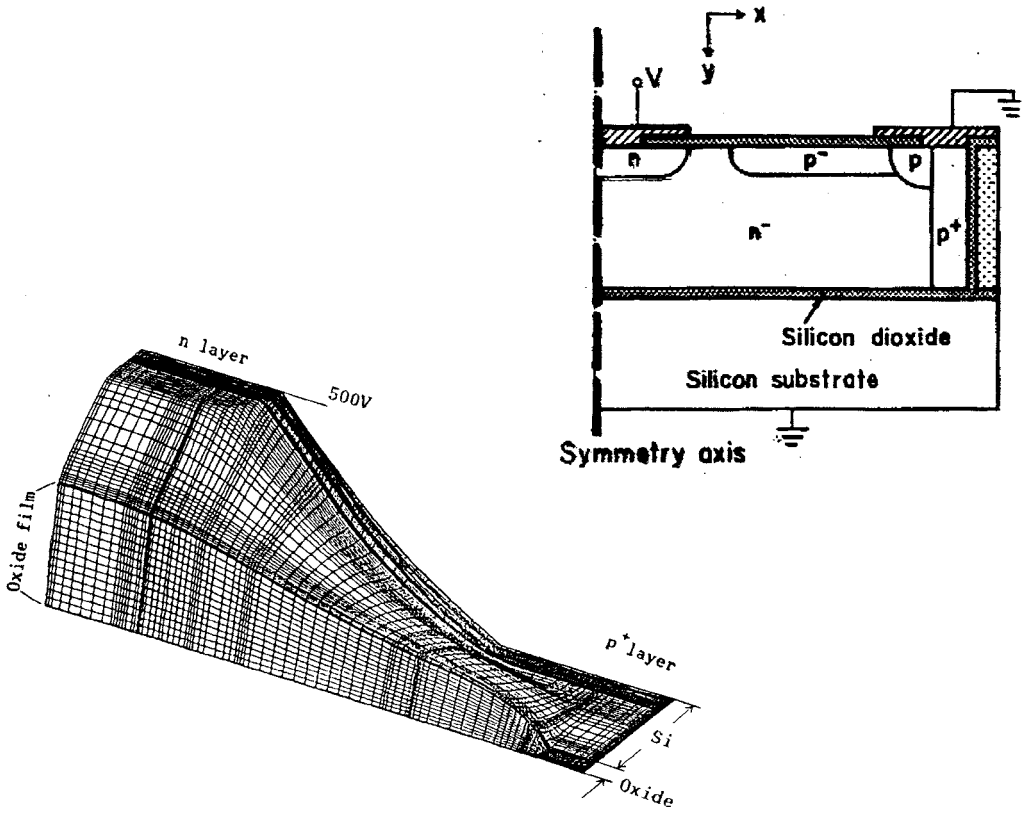


FIGURE 8.16 SOI diode and potential distribution in the diode.

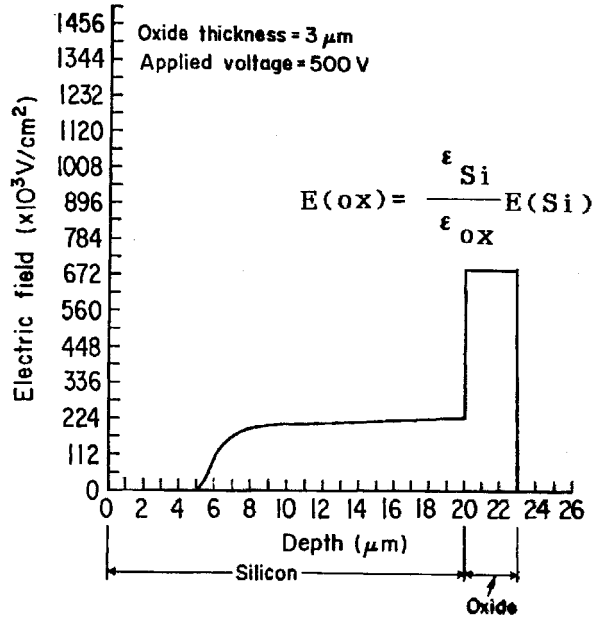


FIGURE 8.17 Electric field distribution along the symmetry axis of diode shown in Fig. 8.16.

$$\varepsilon(\mathrm{Si})E_t(\mathrm{Si}) = \varepsilon(\mathrm{I})E_t(\mathrm{I}) \tag{8.2}$$

where $\varepsilon(\mathrm{Si})$, $\varepsilon(\mathrm{I})$ denote dielectric constants for silicon and silicon dioxide, respectively. Using an insulator film with a lower dielectric constant will increase the device breakdown voltage because the insulator layer sustains a larger share of the applied voltage.

For optimized SOI diodes, the breakdown voltage is substantially limited to the breakdown voltage of the 1-D MOS diode portion, as illustrated in Fig. 8.18, consisting of n^+/n^-/oxide/substrate. Figure 8.19 shows the measured SOI device breakdown voltage as a function of SOI layer thickness with buried oxide thickness as a parameter. The calculated breakdown voltage of 1-D MOS diodes are shown together. A 500-V breakdown voltage can be obtained with a 13-μm thick SOI with 3-μm thick buried oxide.

It is very difficult to achieve a high breakdown voltage exceeding 600 V in simple SOI structures, because a thicker buried oxide layer of 4 μm or more is required. Maximum breakdown voltage is substantially limited with the breakdown voltage of the 1-D MOS diode and actually the realized

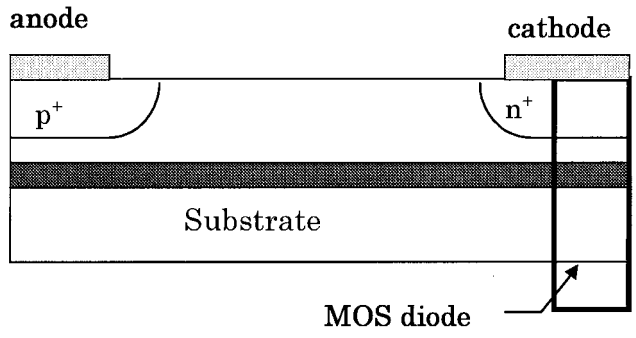


FIGURE 8.18 1-D MOS-diode structure in SOI diode.

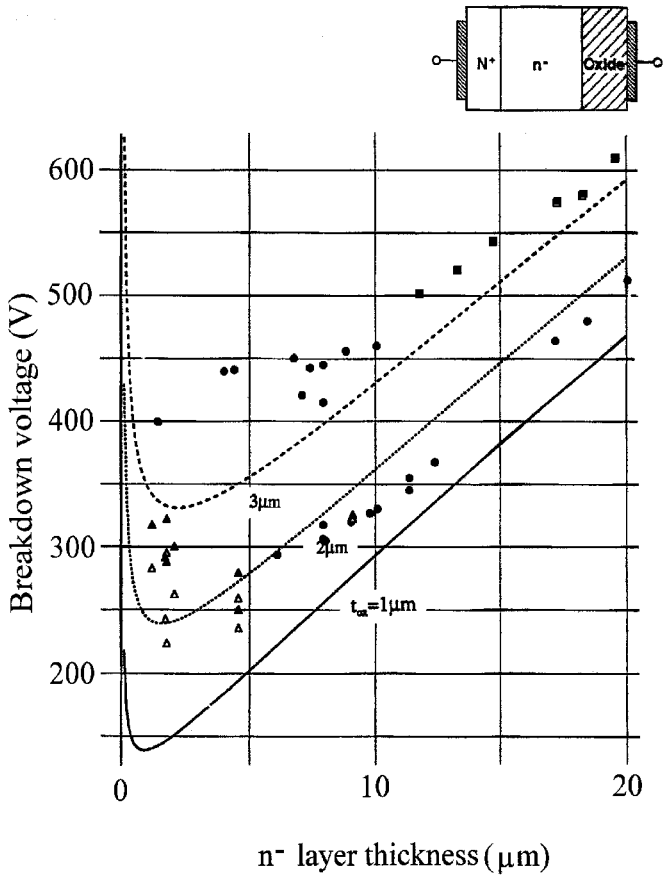


FIGURE 8.19 Measured and calculated SOI device breakdown voltage.

breakdown voltage is lower than this limit. If the influence of the substrate potential can be shielded, it is possible to achieve a higher breakdown voltage in the SOI device.

A new high-voltage SOI device structure, free from the above constraints, was proposed in 1991,[1] that realizes a 1200-V breakdown voltage.[23]

To improve the breakdown voltage, an SOI structure with a shallow n^+ layer diffused from the bottom of SOI layer was proposed.[24] Figure 8.20 shows the structure of an SOI diode with a shallow n^+ layer and the electric field strength in the MOS diode portion compared to that without a shallow n^+ layer. In general, if a larger portion of the applied voltage is carried with the bottom oxide layer, a higher breakdown voltage can be achieved. The problem is how to apply a higher electric field across the buried oxide without increasing the electric field strength in the SOI layer. This problem can be solved by placing a certain amount of positive charge on the SOI layer– buried oxide interface. The positive charge at the interface shields the high electric field in the buried oxide, so that a voltage across the oxide layer can be increased without applying a higher electric field in the SOI layer. The shallow n^+ layer diffused from the bottom is a practical technique to place the positive charge on the SOI layer–buried oxide interface, as shown in Fig 8.20. The required dose of the shallow n^+ layer is around $1 \times 10^{12}\text{cm}^{-2}$.

Very Thin SOI Case

Merchant et al.[25] showed that the SOI diode breakdown voltage is significantly enhanced if the SOI layer thickness is very thin, such as 0.1 μm. As shown in Fig. 8.19, reduction in the SOI layer thickness enhances

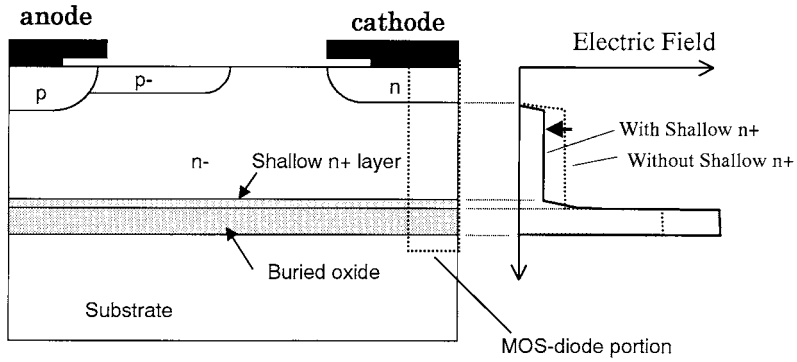


FIGURE 8.20 SOI diode with shallow n^+ layer diffused from the bottom of the SOI layer.

the breakdown voltage if the thickness is less than 1 μm. This is because the carrier path along the vertical high electric field is as short as the SOI layer thickness, so that the carriers reach the top or bottom surface of the SOI layer before ionizing a sufficient amount of carriers for avalanche multiplication along the path. They proposed a combination of the very thin SOI layer and a linearly graded impurity profile of the n-type silicon layer for a high-voltage $n^+n^-p^+$ lateral diode. A 700-V breakdown voltage was realized by this structure.

The exact 2-D simulations revealed that the ideal profile for a lateral diode is approximated by a function that is similar to a tangent function, as shown in Fig. 8.21. The important point is that the p-layer impurity profile should also be graded and that the linearly graded portion is terminated with the exponentially increasing ending portions. By using the proposed profile, a 5000-V lateral diode was predicted to be realized on 0.1-μm SOI on a 600-μm thick quartz substrate. A completely uniform lateral electric field is realized at 5000 V (see Fig. 8.21).

8.6 High-Voltage Output Devices

High-voltage output devices are the most important part of power ICs. An entire power system can be integrated on a single silicon chip if high-voltage power devices can be integrated with analog and digital circuits as well as MPUs. Recently, MOS gate power devices were adopted, primarily because of the low on-resistance and the ease of gate control. These are DMOSFETs and IGBTs.

Lateral Power MOSFET

pn junction-isolated power ICs are frequently used for low-voltage applications, where DMOS is the primary choice for output devices. Since the reliability of junction isolation is not sufficient, SOI DMOS power ICs will be used where high reliability is required. In this section, DMOS electrical characteristics are described, using mostly the junction-isolated DMOS data.

For above a 60-V breakdown voltage range, the vertical DMOS structure with upside surface drain contact (up-drain DMOS, see Fig. 8.22) has conventionally been used. However, recently, the lateral DMOS (LDMOS) structure (Fig. 8.23) tends to be used for the entire voltage range. This is because the LDMOS on-resistance can be directly improved by adopting finer lithography. On the other hand, up-drain vertical DMOS on-resistance includes the resistances of the buried n^+ layer and sinker plug diffusions, which are not improved by finer lithography.

Figure 8.24 shows state-of-the-art DMOS on-resistance as a function of breakdown voltage. The figure also shows state-of-the-art on-resistance for vertical discrete trench MOSFETs as a comparison. Black circles show lateral DMOS, and open squares show trench MOSFETs. Recently, battery-operated mobile equipment and computer peripherals have opened a large applications area, and lateral MOSFETs of the less than 60 V are the major output devices. It is astonishing that the state-of-the-art on-resistances of

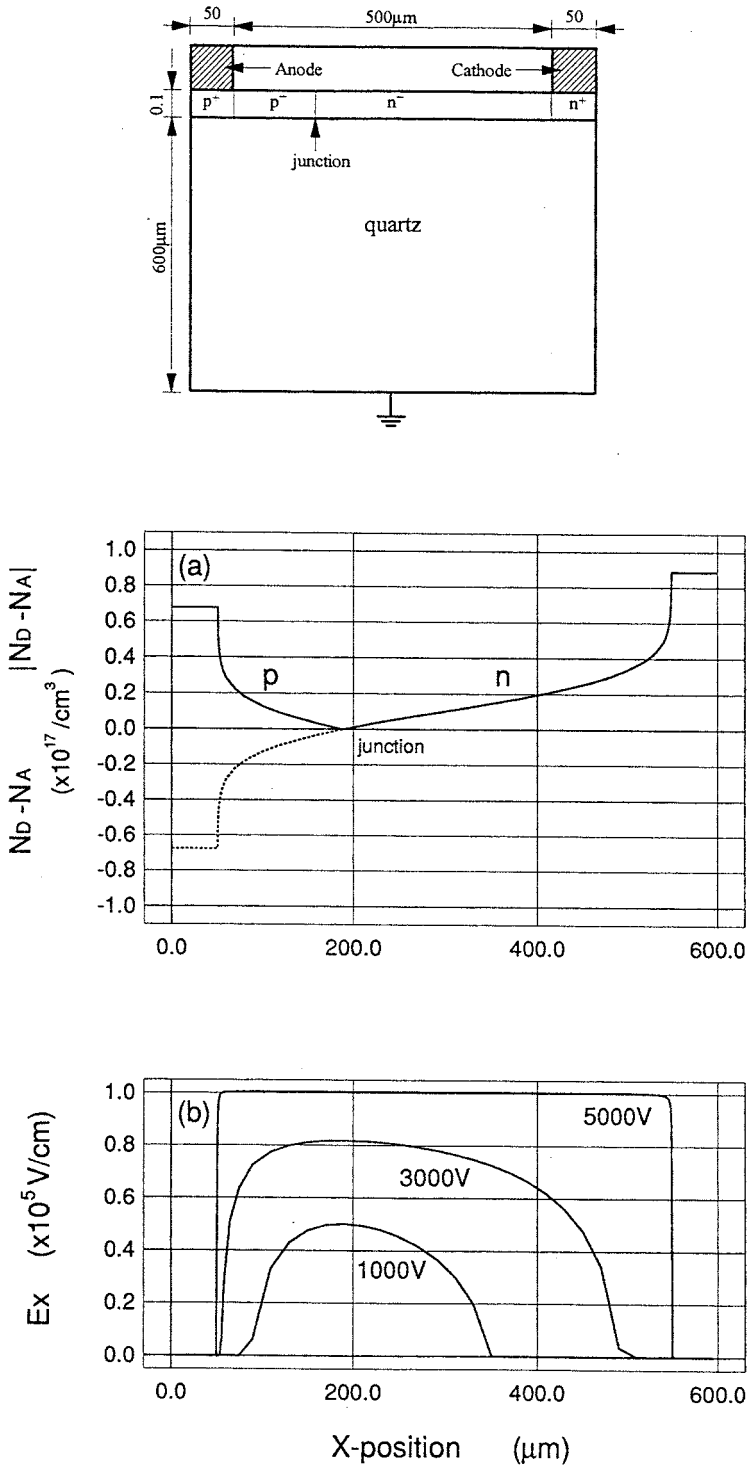


FIGURE 8.21 Calculated ideal profile for a lateral diode on thin SOI.

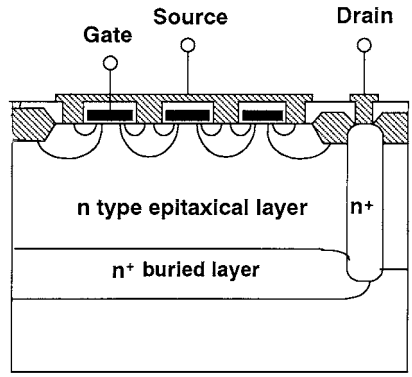


FIGURE 8.22 Vertical DMOS structure with upside surface drain contact.

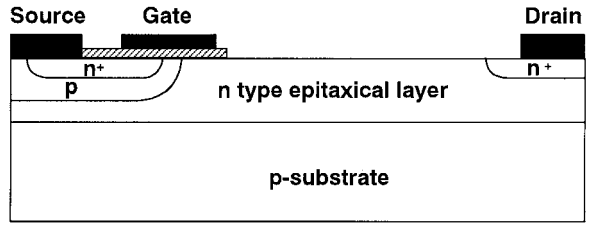


FIGURE 8.23 Lateral DMOS (LDMOS) structure.

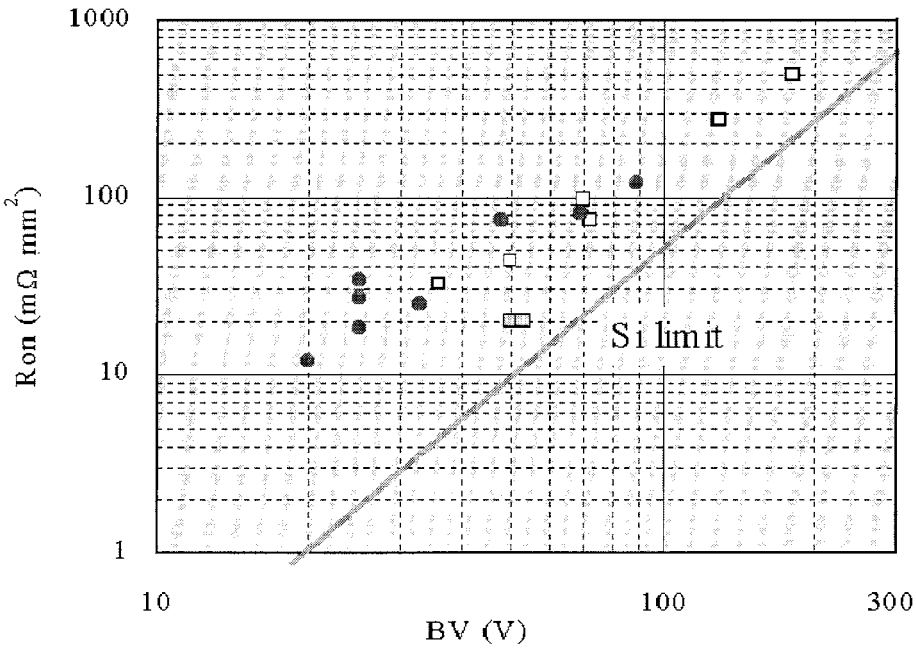


FIGURE 8.24 State-of-the-art lateral DMOS and vertical trench MOSFET on-resistance as a function of breakdown voltage (black circles show lateral DMOS, and open squares show trench MOSFETs).

lateral DMOS and vertical trench MOSFETs are almost the same. This implies that power ICs with a vertical DMOS output will be replaced by power ICs with a lateral DMOS output, if current capacity is small — for example, less than 10 A.

High-side switching operation is an important function in automotive applications, especially in case of H-bridges for motor control. The on-resistance of conventional junction-isolated, high-voltage MOSFETs, shown in Fig. 8.23, is significantly influenced by the source to substrate bias,[26] because the drift layer is depleted. However, in the SOI MOSFETs shown in Fig. 8.25, the drift layer is not depleted, but a hole inversion layer is formed. Thus, the substrate bias influence of SOI LDMOS is small.[26]

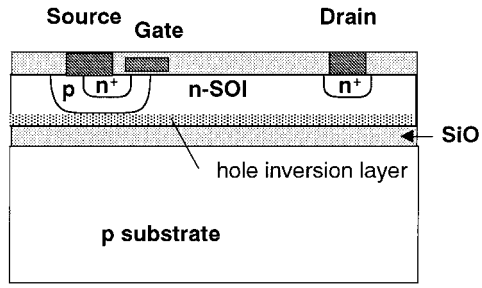


FIGURE 8.25 SOI MOSFET at high-side operation.

Figure 8.26 shows a 60-V DMOS in a 2-μm thick p-type SOI. The fabrication process is completely compatible with the CMOS process. The threshold voltage is controlled by channel implant. The experimentally obtained specific on-resistance of the 60-V LDMOS is 100 $m\Omega \cdot mm^2$.The developed power MOSFET is completely free from substrate bias influence.[27] This is because the hole accumulation layer is induced on the buried oxide, leaving the n-drift layer unchanged. These results indicate that this device can be used for high-side switches without on-resistance increase.

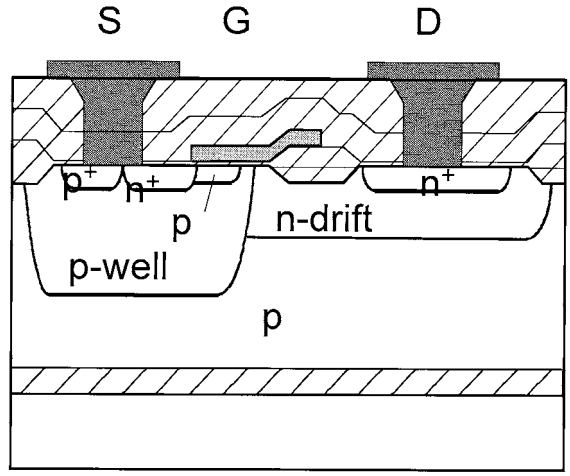


FIGURE 8.26 60-V DMOS in 5-μm thick p-type SOI.

Lateral IGBTs on SOI

IGBTs are suitable for high-voltage, medium current power ICs because of large current capability, as shown in Fig. 8.27. IGBT can be recognized as a pnp transistor driven by an n-channel MOSFET for a

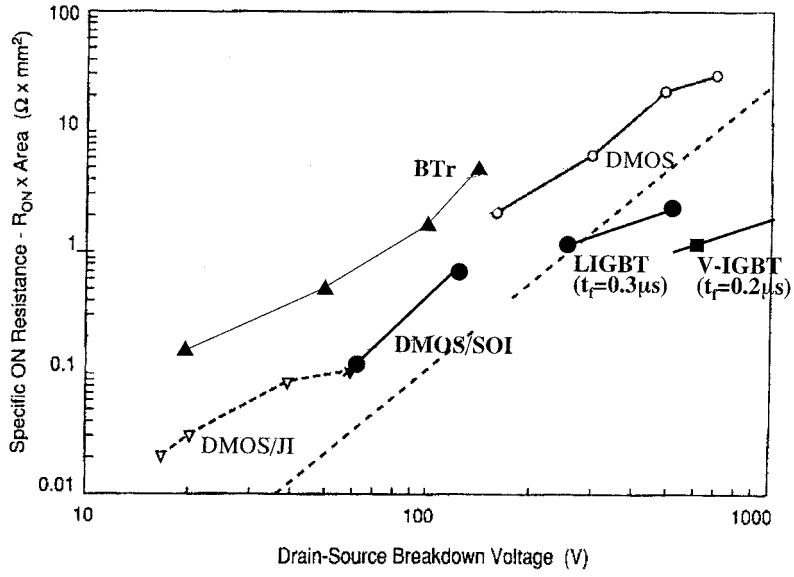


FIGURE 8.27 On-resistance versus breakdown voltage for bipolar transistor, DMOS, LIGBT, and VIGBT (discrete IGBTs).

first order approximation. IGBTs should be fabricated by conventional CMOS-compatible processes so that conventional CMOS circuit libraries can be utilized without changes.

The switching speed of bipolar power devices is conventionally controlled by introduction of a lifetime killer. However, the lifetime control process is not compatible with a conventional CMOS process. There are two ways to control the switching speed of power devices. One way is to use thin SOI layers. The switching speed of IGBTs improves as the SOI thickness decreases,[28] because the carrier lifetime is effectively decreased by the influence of large carrier recombination at the silicon dioxide interfaces.[29] The other way is to reduce the emitter efficiency of the p^+ drain or collector. The effective methods are (1) emitter short, (2) low-dose emitter,[30] (3) high-dose n buffer, and (4) forming an n^+ layer in the p^+ emitter.[31]

Figure 8.28 shows a cross-section of large current lateral IGBTs. Large current capability has been realized by adopting multiple surface channels. Figure 8.29 shows typical current voltage curves of the multi-channel LIGBT. The current is an exponential function of the drain bias (collector bias) for the low-voltage range. The current-voltage curves seem to have a 0.8-V offset voltage, just like a diode. The typical switching speed of the developed LIGBTs is 300 ns.

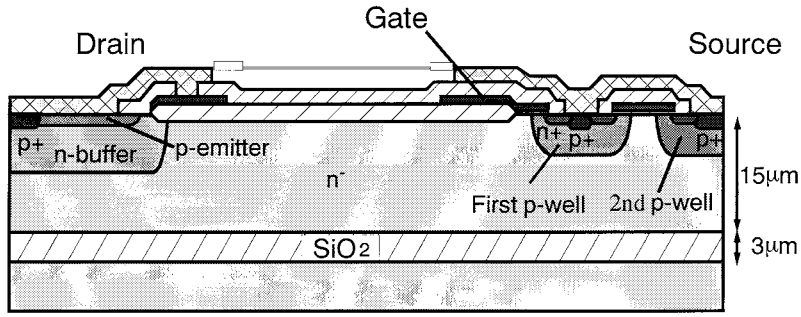


FIGURE 8.28 Cross-section of large current lateral IGBT.[30] (Copyright (1997) IEEE. With permission.)

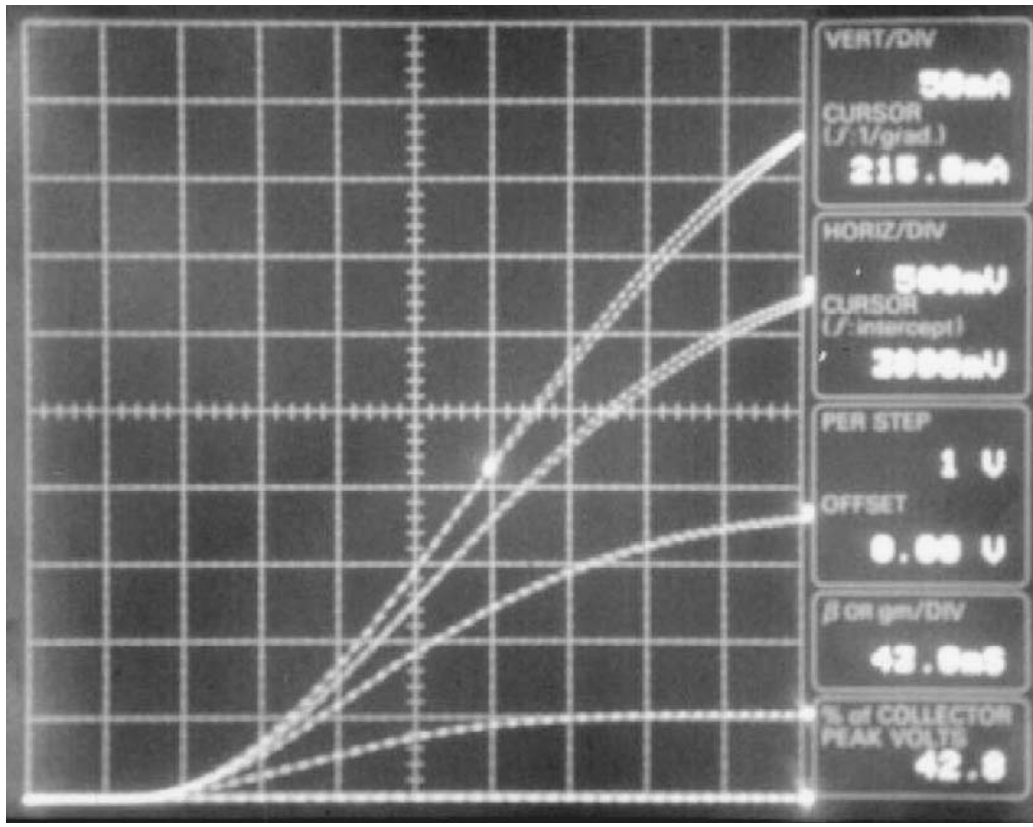


FIGURE 8.29 Typical current-voltage curves of a multi-channel LIGBT (Vertical scale: 50 mA/Div, Horizontal scale: 0.5 V/Div).

It is extremely important to increase the operating current density of LIGBTs in order to reduce chip size. This is because output devices occupy most of the chip area and the cost of the power ICs depends significantly on the size of the power devices. The current density of the developed LIGBT is 175 A/cm^2 for 3-V forward voltage.

8.7 Sense and Protection Circuit

Power ICs have a significant advantage over discrete devices in terms of power switch protection because the protection circuit can be integrated on the same chip. Figure 8.30(a) shows the over-temperature sense circuit. In this circuit, the junction temperature dependence of diode forward voltage drop is utilized to sense the temperature. The series diodes are located close to the power switches so that the junction temperature of the diodes responds to the temperature at the power switches. Comparing the forward voltage drop across the diodes with the reference voltage, the circuit senses the over-temperature at the power switches and the fault signal is fed back to the gate control logic.

Figure 8.30(b) shows the over-current sense circuit for bipolar power transistors. The magnitude of the current through the main transistor is reflected to the current through the sense transistor with the current mirror circuit configuration. Therefore, the voltage drop across the resistor in the collector of the sense transistor is proportional to the current through the main transistor. Comparing the voltage drop with the reference voltage, the circuit detects the over-current. Figure 8.30(c) shows the over-current sense circuit for IGBTs or MOSFETs, which has a similar configuration with Fig. 8.30(b). In this circuit, the over-current is detected by the voltage drop across the resistor in the sense IGBT (MOSFET) emitter (source).

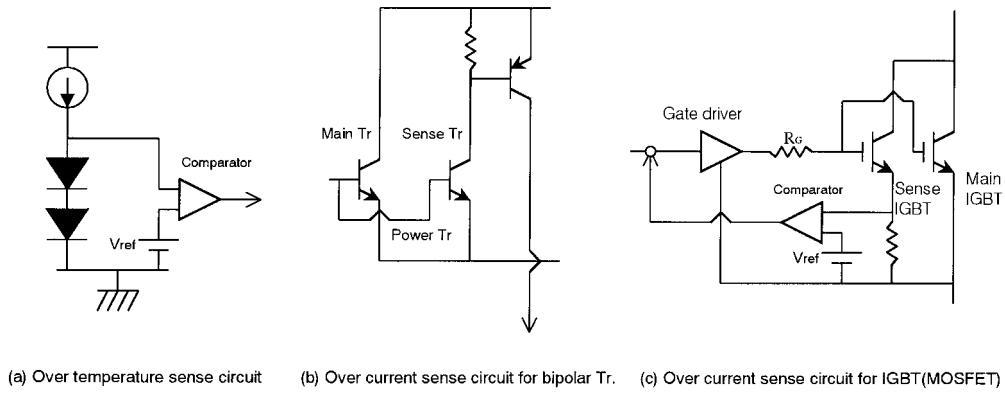


FIGURE 8.30 Sense circuits for over-temperature and over-current.

In short-circuit protection, the collector current should be squeezed or terminated as soon as the short-circuit operation is detected. For this purpose, the protection circuit directly draws down the gate voltage with the short feedback loop. Figure 8.31 shows a typical short-circuit protection circuit for IGBTs. When high current flows through the IGBT, the voltage drop across the emitter resistor of the sense IGBT directly drives the npn transistor. The transistor draws down the gate voltage so that the collector current is reduced to the level of a safe turn-off operation.

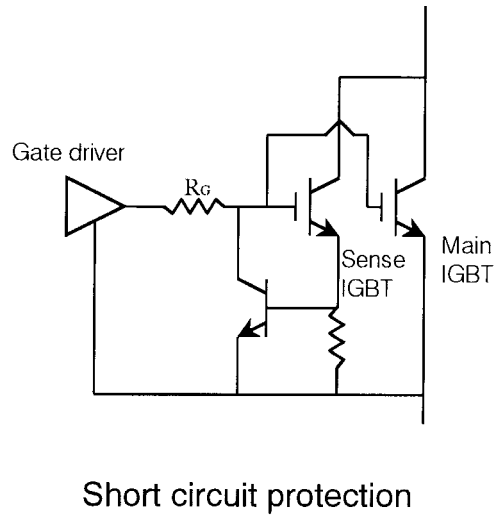


FIGURE 8.31 Typical short-circuit protection method for IGBTs.

8.8 Examples of High-Voltage SOI Power ICs with LIGBT Outputs

Current main applications of high voltage SOI power ICs are dc motor control and flat-panel display drivers. Recently, technologies for color plasma display panels have been greatly improved, and demands for PDP driver ICs have increased. There are several reports[32,33] on the development of such ICs using SOI wafers. Flat-panel display drivers have to integrate a large number of high-voltage devices. Trench isolation and LIGBTs are key techniques for reducing chip size and resultant cost.

Another large market is the motor control field. Home-use appliances use a number of small motors, which are directly controlled by a ac source line. Single-chip inverter ICs are able to reduce system size and increase system performance. Figure 8.32 shows a 500-V, 1-A, single-chip inverter IC[34] for dc brushless motors. It integrates six 500-V, 1-A LIGBTs, six 500-V diodes, control protection, and logic circuits.

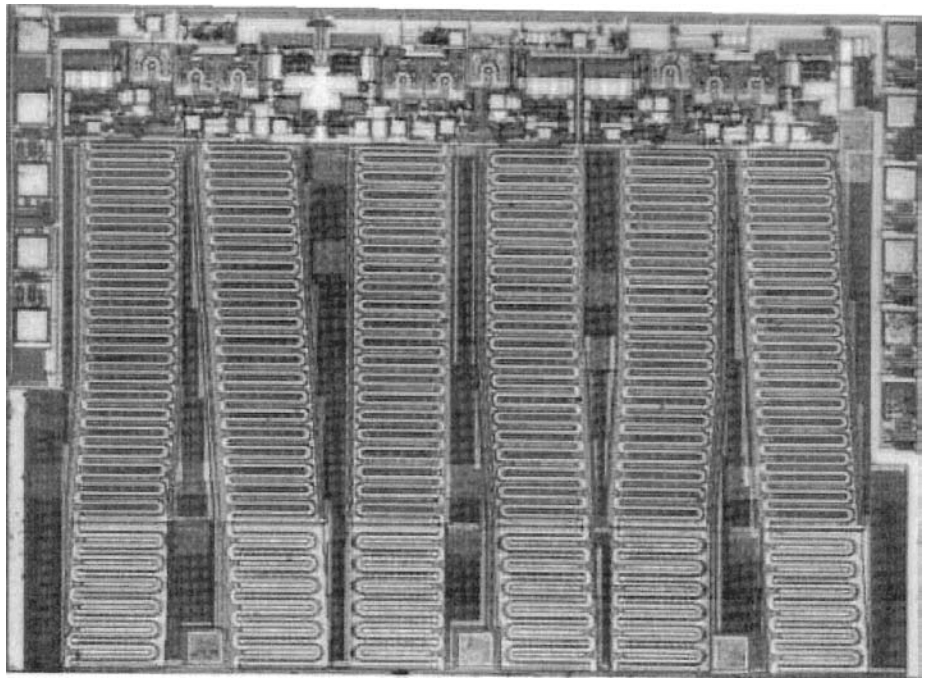


FIGURE 8.32 Photograph of a 500-V, 1-A single-chip inverter.

8.9 SOI Power ICs for System Integration

Another prospective application of SOI technology is in the automotive field, which requires large current DMOS outputs. Conventional pn junction-isolated power ICs are frequently used for these applications; however, the reliability of junction isolation is not sufficient. SOI DMOS power ICs will be used where high reliability is required.

For less than 100-V applications, the required thickness of the buried oxide layer is less than 1 μm. The warpage of the SOI wafers is very small; thus, fine lithography can be applied. The same CMOS circuit library can be used without changes because the same CMOS fabrication process can be applied without modification if a relatively thick SOI layer is used.

This section shows the possibility of integration of an MPU, together with BiCMOS analog circuits and 60-V power LDMOS. 4-bit MPUs, vertical npn, pnp, and 60-V power DMOS were fabricated on 2-μm SOI wafers by a conventional 0.8-μm BiCMOS process.[27] The 60-V DMOS used CMOS p-well without using self-alignment.

The fabricated 4-bit MPU, consisting of 30,000 FETs for the core, 6000 FETs for the cache, and 120,000 FETs for the ROM, operated at a 20% faster clock speed of 50 MHz at 25°C, as compared to 42 MHz of the bulk version MPU, and even operated at over 200°C. It was found that the clock speed could be improved and that a large latch-up immunity at high temperature was realized even if the MOSFETs were not isolated by trenches. The maximum operating temperature was more than 300°C. It was found that the yield of the MPU fabricated on SOI was the same as that on bulk wafers, verifying that the crystal

quality of the currently available SOI wafers was sufficiently good. It was also found that both SOI and bulk MPUs could be operated at 300°C if MPUs consisted of pure CMOS, although the power consumption of the bulk MPU was larger than that of the SOI MPUs.

One of the characteristic features of the SOI power IC structure, shown in Fig. 8.33, is that there are no buried layers for bipolar transistors. It was found that vertical npn and pnp transistors fabricated on the n-well and p-well layers exhibited sufficiently good characteristics, and the typical current gains h_{FE} for the vertical npn and pnp transistors were 80 and 30, respectively.

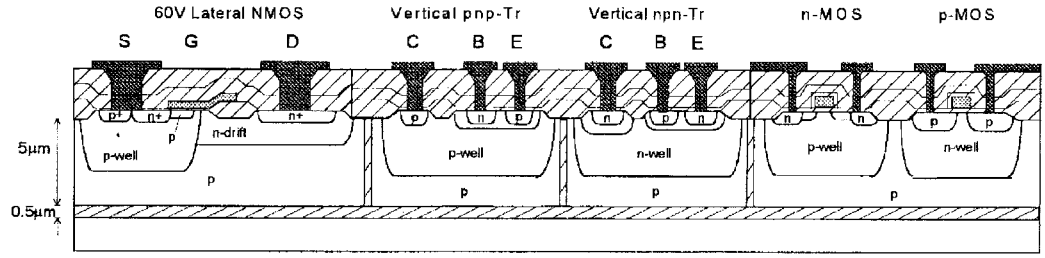


FIGURE 8.33 SOI power IC system integration.

All these results show that system integration including power LDMOS will be a reality in SOI wafers.

8.10 High-Temperature Operation of SOI Power ICs

The leakage current of SOI devices simply reduces as the SOI layer becomes thinner, as seen in Fig. 8.34.[35] Small leakage current enables high-temperature operation of SOI power ICs. It was experimentally shown

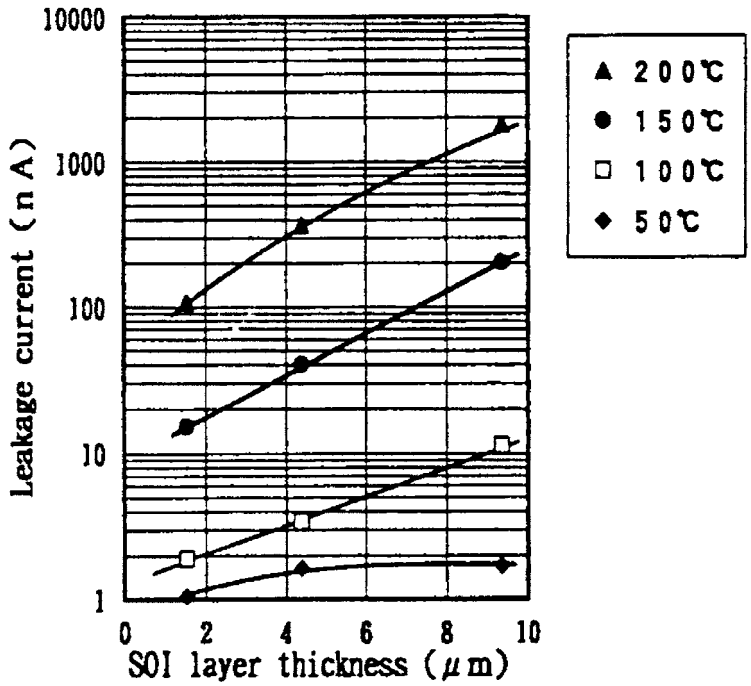


FIGURE 8.34 Leakage current vs. SOI layer thickness. (Copyright (1994) IEEE. With permission.)

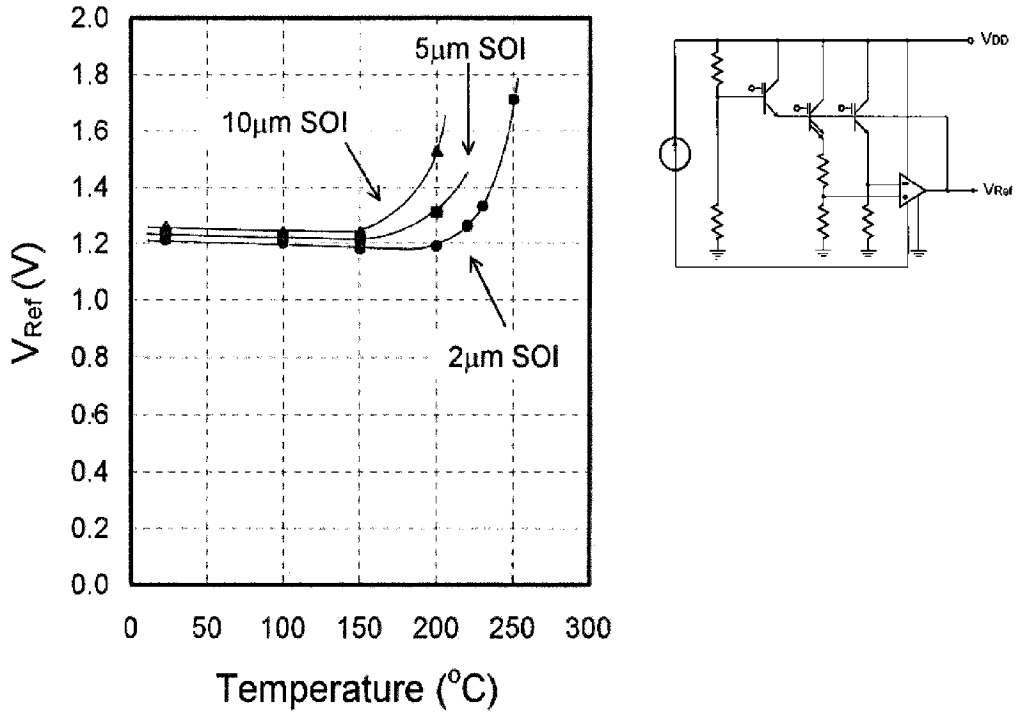


FIGURE 8.35 Output voltage of a bandgap reference circuit as a function of operation temperature. (Copyright (1995) IEEE. With permission.)

that IGBTs can be operated at a switching frequency of 20 kHz at 200°C if they are fabricated in thin SOI of less than 5 μm. The maximum operating temperature of analog circuits in SOI increases as the SOI layer becomes thinner. Figure 8.35 shows the output voltage of bandgap reference circuits as a function of temperature with SOI thickness as a parameter. In the circuits, each device was not trench-isolated. If all the devices are trench-isolated, much higher temperature operation can be expected.

The 200°C operation of 250-V, 0.5-A, three-phase, 1-chip inverter ICs fabricated in a 5-μm thick SOI layer was demonstrated.[36] CMOS circuits on SOI were found to be capable of operating at 300°C. CMOS-based analog circuits with a minimum number of bipolar transistors were adopted. It was found that the bandgap reference circuit operated at 250°C, which was higher than that expected from Fig. 8.34. This was probably because each device was trench-isolated. A DC brushless motor was successfully operated by the single-chip inverter IC at 200°C. The carrier frequency was 20 kHz.

References

1. Nakagawa, A., "Impact of dielectric isolation technology on power ICs (Invited)," *Proc. of ISPSD*, 1991, 16.
2. Tarui, Y., Hayashi, Y., and Sekigawa, T., "Diffusion self-aligned MOST: a new approach for high speed device," *Proc. of the 1st Conference on Solid State Devices*, Tokyo, 1969, 105.
3. Wrathall, R. S., Tam, D. T., Terry, L., and Rob, S. P., "Integrated circuits for the control of high power," *IEDM Tech. Digest*, 1983, 408.
4. Murari, B., Bertotti, F., and Vignola, G. A., *Smart Power ICs*, Springer-Verlag, 1995.
5. Beasom, J. D., "A process for simultaneous fabrication of vertical npn and pnp's and p-ch MOS devices," *IEDM Tech. Digest*, 1973, 41.

6. Shimbo, M., Furukawa, K., Fukuda, K., and Tanzawa, K., "Silicon-to-silicon direct bonding method," *J. Appl. Phys.*, vol. 60, p. 2987, 1986.
7. Ohashi, H., Ohura, J., Tsukakoshi, T., and Shimbo, M., "Improved dielectrically isolated device intergation by silicon wafer-wafer direct bonding (SDB) technique," *IEDM Tech. Digest*, 1986, 210.
8. Okabe, T., Sakamoto, K., and Hoya, K., "Semi-well isolation-based intelligent power IC technology," *Proc. of ISPSD*, 1988, 96.
9. Becke, H. W., "Approaches to isolation in high voltage integrated circuits (invited paper)," *IEDM Tech. Digest*, 1985, 724.
10. Rumennik, V., "Power devices are in the chips," *IEEE SPECTRUM*, July 1985, 42.
11. Nakagawa, A., Yasuhara, N., and Baba, Y., "New 500V output device structure on silicon oxide film," *Proc. of ISPSD*, 1990, 97.
12. Lathrop, J. W., "Semiconductor-networking technology-1964," *Proc. IEEE*, vol. 52, p. 1430, 1964.
13. Bouchard, J. and Hammmond, F. W., Abstract No. 165, "The iso-layer' process," *J. Electrochem. Soc.*, vol. 111, p. 197C, 1964.
14. Appels, J. A. and Vaes, H. M. J., "High voltage thin layer devices (RESUF DEVICES)," *IEDM Tech. Digest*, 1979, 238.
15. Rosen, R. S., Sprinter, M. R., and Tremain, R. E. Jr., "High voltage SOS/MOS devices and circuit elements: design, fabrication, and performance," *IEEE J. Solid State Circuits*, vol. SC-11, p. 431, 1976.
16. Izumi, Doken, M., and Ariyoshi, H., "C.M.O.S. devices fabricated on buried SiO2 layers formed by oxyzen implantation into silicon," *Electron. Lett.*, vol. 14, p. 593, 1978.
17. Geis, M. W., Flanders, D. C., and Smith, H. I., "Crystallographic orientation of silicon on an amorphous substrate using an artificial surface-relief grating and laser crystallization," *Appl. Physics Lett.*, vol. 35, p. 71, 1970.
18. Baliga, B. J., Adler, M. S., Gray, P. V., and Love, R. P., "The insulated gate rectifier (IGR) a new switching device," *IEDM Tech. Digest*, 1982, 264.
19. Nakagawa, A., Ohashi, H., Kurata, M., Yamaguchi, H., and Watanabe, K., "Non-latch-up 75A bipolar-mode MOSFET with large ASO," *IEDM Tech. Digest*, 1984, 860.
20. Temple, V. A. K., "MOS controlled thyristor," *IEDM Tech. Digest*, 1984, 282.
21. Kamei, T., "High voltage integrated circuits for telecommunication," *IEDM Tech. Digest*, 1981, 254.
22. Endo, K., Baba, Y., Udo, Y. Yasui, M., and Sano, Y., "A 500A 1-chip inverter IC with new electric field reduction structure," *Proc. of ISPSD*, 1994, 379.
23. Funaki, H., Yamaguchi, Y., Hirayama, K., and Nakagawa, A., "New 1200V MOSFET structure on SOI with SIPOS shielding layer," *Proc. of ISPSD*, 1998, 25.
24. Yasuhara, N., Nakagawa, A., and Furukawa, K., "SOI device structure inplementing 650V high voltage output devices on VLSIs," *IEDM Tech.. Digest*, 1991, 141.
25. Merchant, S., Arnold, E., Baumgart, H., Mukherjee, S., Pein, H., and Pinker, R., "Realization of high breakdown voltage (>700V) in thin SOI devices," *Proc. of ISPSD*, 1991, 141.
26. Arnold, E., Merchant, S., Amato, M., Mukherjee, S., and Pein, H., "Comparison of junction-isolated and SOI high-voltage devices operating in the source-follower mode," *Proc. of ISPSD*, 1992, 242.
27. Funaki, H., Yamaguchi, Y., Kawaguchi, Y., Terazaki, Y., Mochizzuki, H., and Nakagawa, A., "High voltage BiCDMOS technology on bonded 2μm SOI intergrating vertical npn, pnp, 60V-LDMOS and MPU, capable of 200°C operation," *IEDM Tech. Digest*, 1995, 967.
28. Yasuhara, N., Matsudai, T., and Nakagawa, A., "SOI thickness and buried oxide thickness dependencies of high voltage lateral IGBT switching characteristics," *Ext. Abstr. Int. Conf. SSDM*, 1993, 270.
29. Omura, I., Yasuhara, N., Nakagawa, A., and Suzuki, Y., "Numerical analysis of switching characteristics — switching speed enhancement by reducing the SOI thickness," *Proc. of ISPSD*, 1993, 248.
30. Funaki, H., Matsudai, T., Nakagawa, A., Yasuhara, N., and Yamaguchi, Y., "Multi-channel SOI lateral IGBTs with large SOA," *Proc. of ISPSD*, 1997, 33.

31. Yamaguchi, Y., Nakagawa, A., Yasuhara, N., Watanabe, K., and Ogura, T., "New anode structure for high voltage lateral IGBTs," *Ext. Abst. Int. Conf. SSDM*, 1990, 677.
32. Gonzalez, F., Shekhar, V., Chan, C., Choy, B., and Chen, N., "Fabrication of a 300V high current (300mA/output), smart-power IC using gate-controlled SCRs on bonded (BSOI) technology," *IEDM Tech. Digest*, 1995, 473.
33. Sumida, H., Hirabayashi, H., Shimabukuro, H., Takazawa, Y., and Shigeta, Y., "A high performance plasma display panel driver IC using SOI," *Proc. of ISPSD*, 1998, 137.
34. Nakagawa, A., Funaki, H., Yamaguchi, Y., and Suzuki, F., "Improvements in lateral IGBT design for 500V 3A one chip inverter ICs," *Proc. of ISPSD*, 1999, 321.
35. Matsudai, T., Yamaguchi, Y., Yasuara, N., Nakagawa, A., and Mochizki, H., "Thin SOI IGBT leakage current and a new device structure for high temperature operation," *Proc. of ISPSD,* 1994, 399.
36. Yamaguchi, Y., Yasuhara, N., Matsudai, T., and Nakagawa, A., "200°C High temperature operation of 250V 0.5A one chip inverter ICs in SOI," *Proc. of PCIM INTER'98*, Japan, 1998, 1.

Martin, S.S., et al “Noise in VLSI Technologies”
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

9

Noise in VLSI Technologies

Sam S. Martin
Thad Gabara
Kwok Ng
Bell Laboratories
Lucent Technologies

9.1 Introduction

The progress of VLSI technologies is a result of an intimate interaction between improvements in IC chip design and in device properties of the underlying process technologies. The semiconductor roadmap following Moore's law is responsible for an exponential decrease of minimum feature size of devices. The associated increase of device speed and decrease of supply voltage have strong implications for the available noise margin of a VLSI chip. This chapter addresses the main issues relevant for noise in VLSI technologies at various levels as indicated schematically in Fig. 9.1. At the microscopic level, the fundamental sources of noise associated with carrier transport are derived with emphasis on semiconductors. The next level deals with the noise properties of active devices and passive components. Three classes of active devices are chosen for illustration: bipolar junction transistors (BJTs), field effect transistors (FETs), and two terminal junction devices (diodes). Although the treatment of these device classes implies silicon-based technologies (Si-BJT, Si-MOSFET, Si-diode), it can generally be applied to other device classes as well (HBT, MESFET, etc.). Finally, the chip level noise is presented in terms of the major contributions being amplifier noise, oscillator noise, timing jitter, and interconnect noise. The evolution of VLSI technologies into the deep-sub-micron regime has significant implications for the treatment of fundamental noise mechanisms, which are briefly outlined in the last section of the chapter.

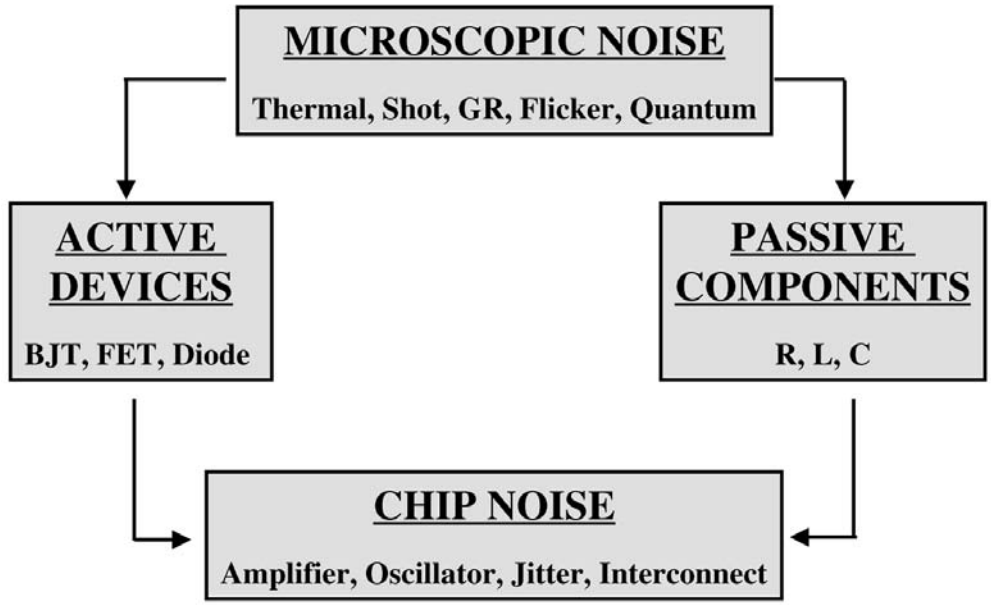


FIGURE 9.1 Schematic describing the various levels of noise studied in this chapter and their causal relationship.

9.2 Microscopic Noise

Thermal Noise

Thermal noise is, in general, associated with random motion of particles in a force-free environment. Since the mean available energy per degree of freedom is proportional to temperature, the resulting noise is referred to as thermal noise. Specifically, carrier transport in semiconductors is treated by considering energy distribution functions, such as the Fermi-Dirac distribution. The effect of local fields and scattering of carriers in position and momentum space is described by a change in the distribution function. This is expressed by the Boltzmann transport equation. Noise is a stochastic process and is characterized in terms of time-averaged quantities, as the instantaneous value cannot be predicted. The fundamental origins of thermal noise in semiconductors are microscopic velocity fluctuations. Velocity fluctuations occur due to various mechanisms, such as electron-phonon scattering, electron-electron scattering, etc. Since the same mechanisms give rise to macroscopic resistivity of a material, noise can be viewed as the microscopic property of transport. Consequently, noise is intimately related to bulk transport in a semiconductor. The voltage noise spectral density is given by:

$$S_{\Delta v}(\omega) = 2 \cdot \int_{-\infty}^{\infty} \langle \Delta v(t) \Delta v(t+\tau) \rangle e^{j\omega\tau} d\tau \tag{9.1}$$

In the above expression, $\Delta v(t)$ denotes the instantaneous velocity fluctuation, ω the frequency, and S the spectral density of noise. Velocity fluctuations can also be viewed as the driving force for diffusion, where the diffusion coefficient is given by:

$$D(\omega) = \int_0^{\infty} \langle \Delta v | \Delta v(t+\tau) \rangle e^{-j\omega\tau} d\tau \tag{8.2}$$

Therefore, the voltage noise spectral density can be directly related to the real part of the frequency-dependent diffusion function as:

$$S_{\Delta v}(\omega) = 4 \cdot \mathrm{Re}[D(\omega)] \tag{9.3}$$

The current noise spectral density due to a charge q in a slab of material of length L can be derived from the carrier velocity fluctuation as:

$$S_{\Delta i}(\omega) = \frac{q^2}{L^2} \cdot S_{\Delta v}(\omega) \tag{9.4}$$

The current noise spectral density of the whole slab of material of area A is then obtained by integrating the carrier density n(x) and the diffusion coefficient over the length of the sample, as given by:

$$S_I(\omega) = \frac{4q^2 A}{L^2} \cdot \int_0^L n(x) \mathrm{Re}[D(\omega, x)] dx \tag{9.5}$$

In the limit of uniform carrier density, thermal equilibrium, and neglecting quantum effects, the diffusion coefficient can be expressed in terms of the carrier mobility using the Einstein relation:

$$D = \frac{kT}{q}\mu \tag{9.6}$$

and the current noise spectral density is given by:

$$S_I(\omega) = 4kT \cdot \left(nq\mu \cdot \frac{A}{L} \right) \tag{9.7}$$

The above expression leads to the well-known Johnson thermal noise of a conductance $G = \sigma A/L$, where the conductivity is $\sigma = nq\mu$ and the carrier mobility is denoted by μ:

$$S_I(\omega) = 4kT \cdot G(\omega) \tag{9.8}$$

In general, the conductance is the real part of the frequency-dependent admittance of the material as indicated above. Note that the thermal noise expression given here is valid only when the mean scattering time of the carriers is negligible with respect to the inverse of the frequency at which the noise is measured. This assumption is true for most studies of semiconductors where the mean scattering times are on the order of 1 ps. The generalized expression for finite scattering times is discussed in a later section. The thermal noise expression also provides a direct relation between the microscopic noise and the bulk resistance (conductance), indicating the nature of noise as being due to the same type of scattering mechanisms as those causing resistance in a sample. A prominent feature of thermal noise is its direct dependence on temperature, giving rise to the possibility of measuring a thermodynamic quantity with noise.

Shot Noise

The discrete nature of particles undergoing an average flow in space, together with their velocity distribution, results in a random arrival at a fixed plane of incidence. The corresponding fluctuations of the flow are referred to as *shot noise*. Specifically, the discrete flow of charge in a field yields shot noise of current in semiconductors. Shot noise can be derived as being due to a random train of pulses corresponding to events, such as carrier emission from an electrode, carrier injection across a semiconductor junction, etc. The power spectral density of the resulting time-dependent waveform is given by:

$$S(\omega) = 2\nu \cdot \langle a^2 \rangle \cdot |F(\omega)|^2 \tag{9.9}$$

Here, $F(\omega)$ is the Fourier transform of the impulse function, ν the mean rate of the events, and $\langle a^2 \rangle$ the mean square amplitude of the pulses. For a current pulse $\delta i(t)$ due to the transit of a charge q through a sample, the Fourier transform is given by:

$$F(\omega) = \frac{1}{q} \cdot \int_0^{\tau_T} \delta i(t) \cdot e^{-j\omega\tau} dt \tag{9.10}$$

At sufficiently low frequencies, the transit time τ_T can be neglected with respect to $1/\omega$, and the current pulse can be considered a delta function. The Fourier transform function then becomes unity. The mean rate of current pulses is $\nu = I/q$, where I is the mean value of the current, and the mean pulse amplitude $\langle a \rangle = q$, so that the power spectral density of shot noise current is given by:

$$S_I(\omega) = 2qI \tag{9.11}$$

Note that the above expression is valid only for negligible transmit times of the carriers through a sample region with respect to the inverse of the frequency at which the noise is measured. The shot noise for non-negligible transit times and scattering times is treated in more detail later. One of the most significant properties of shot noise is its dependence on the mean current flow or bias in a semiconductor, in contrast to thermal noise which is dependent only on the temperature and resistivity of the material. Shot noise can also be measured only when the energy distribution function of the carriers can be probed by a reference plane of incidence, such as a p-n junction in a semiconductor.

Generation-Recombination Noise

The generation and recombination of charge carriers due to traps in semiconductors results in fluctuations of the current flow through the semiconductor. The temporal change in the carrier number N from a rate of generation g(t) and a rate of recombination r(t) is given by:

$$\frac{d(\Delta N)}{dt} = -\frac{\Delta N}{\tau} + \Delta g(t) - \Delta r(t) \tag{9.12}$$

Here, τ is the mean lifetime of the carriers. The noise spectral density of the carrier number fluctuations is found to be:

$$S_N(\omega) = 4\langle \Delta N^2 \rangle \cdot \frac{\tau}{1 + \omega^2\tau^2} \tag{9.13}$$

A current fluctuation is related to a carrier number fluctuation through $\Delta I/I = \Delta N/N$. The current noise spectral density from generation and recombination of carriers is then given by:

$$S_I(\omega) = 4\frac{I_0^2}{N_0^2}\langle \Delta N^2 \rangle \cdot \frac{\tau}{1+\omega^2\tau^2} \tag{9.14}$$

Here, I_0 denotes the mean current and N_0 the mean carrier number. The form of the above expression implies a constant noise power below a characteristic frequency $\omega_c = 1/\tau$, and a $1/\omega^2$ dependence (Lorentzian) at higher frequencies. This type of noise is observed in systems with a single well-defined trapping time constant or energy level of traps. The most significant feature of GR noise is its explicit frequency dependence, in contrast to the "white" spectral densities of thermal and shot noise. This frequency dependence has strong implications for semiconductors, since the characteristic frequency is well within the frequency range of most applications.

Flicker Noise

Flicker noise is, in general, a phenomenon observed in many systems with an inverse frequency dependence of the noise spectral density over a wide frequency regime. In semiconductors, the presence of energy traps can lead to generation and recombination of carriers and a corresponding frequency-dependent noise of the flicker noise type. A phenomenological approach for deriving the frequency dependence of flicker noise is to consider the expression for generation-recombination noise given by:

$$S_I^{\tau}(\omega) = A(I) \cdot \frac{\tau}{1+\omega^2\tau^2} \tag{9.15}$$

Here, A(I) summarizes the current-dependent pre-factor given earlier. A distribution of characteristic time constants given by a probability density $P(\tau)d\tau$ would result in a current noise spectral density of the form:

$$S_I(\omega) = A(I) \cdot \int_{\tau_0}^{\tau_1} \left(\frac{\tau}{1+\omega^2\tau^2} \cdot P(\tau)d\tau \right) \tag{9.16}$$

In the case of MOSFETs, the tunneling of carriers from the channel through an oxide to a trap at a distance x from the channel/oxide interface produces a probability distribution of time constants:

$$P(\tau) \propto \frac{1}{\tau} \tag{9.17}$$

Such a distribution of time constants yields the flicker noise expression for the current noise spectral density:

$$S_I(\omega) \propto \frac{A(I)}{\omega} \tag{9.18}$$

Note that the inverse frequency dependence of the noise is obtained from the integral above with the approximation: $1/\tau_1 < \omega < 1/\tau_0$. The most important characteristic of flicker noise is its explicit inverse frequency dependence, like GR noise. Another feature is the dependence of flicker noise magnitude on material properties such as trap densities, carrier mobility, etc. Consequently, no general form can be given for the flicker noise of semiconductors, but specific devices need to be considered.

Although the overall frequency dependence of flicker noise is universally found to be inversely proportional to frequency in semiconductors, the expression for the amplitude of flicker noise is strongly dependent on device technology. In general, for Si-BJTs and Si-MOSFETs, the flicker noise power is found to increase with bias current according to a power-law as given by:

$$S_I(\omega) = K \cdot \frac{I^{\gamma}}{\omega^{\alpha}} \tag{9.19}$$

The bias exponent γ has typical values between 1 and 2, and the frequency exponent α varies around unity by about 20%.

Quantum Limit

Quantum effects become relevant for noise processes for sufficiently high frequencies and low temperatures, such that a quantum of energy is comparable to or larger than the thermal energy:

$$\hbar\omega \geq kT \tag{9.20}$$

Consequently, the treatment of thermal or shot noise power requires modifying the mean available energy for each degree of freedom, which results in the following generalized expression for the spectral density of current noise:

$$S_I^Q(\omega) = S_I^C(\omega) \cdot \left(\frac{1}{2}\frac{\hbar\omega}{kT} + \frac{\hbar\omega/kT}{[e^{\hbar\omega/kT} - 1]} \right) \tag{9.21}$$

Here, the quantum correction (denoted by Q) to the classical noise power (denoted by C) is given by the expression in the parentheses, which includes both the Planck distribution function and the zero-point energy term. In the limit of low frequencies or high temperatures ($\hbar\omega << kT$); the expression in parentheses reduces to unity and the classical form for the noise is recovered.

A further implication of quantum effects is a fundamental lower limit on the noise power of a linear amplifier, which is given by:

$$\frac{\Delta P_{min}}{\Delta f} = \frac{(G-1)}{G} \cdot \hbar\omega_0 \tag{9.22}$$

Here, P denotes the noise power G of the amplifier gain and Δf is a frequency interval centered around the signal frequency ω_0. This limit on the noise is a result of the uncertainty principle applied to the number of photons and the corresponding phase of an electromagnetic wave.

9.3 Device Noise

The microscopic noise mechanisms described in the previous section are responsible for producing macroscopic noise in active semiconductor devices as well as passive components. This section briefly outlines the major noise contributions in several device technologies relevant for IC design. The intrinsic noise sources are related to measurable noise parameters and to properties suitable for device modeling.

Passive Components

The most relevant contribution to noise from passive components is the thermal noise of a resistance R given by:

$$S_v^t = 4kTR \tag{9.23}$$

The thermal noise is determined by the magnitude of the resistance and its equilibrium temperature only. It exhibits a "white" (constant) noise spectral density as function of frequency and becomes

frequency dependent at high frequencies, either due to the reactive components of a real resistor or fundamentally due to non-equilibrium effects.

Besides the thermal noise of resistors, the reactive components (inductors and capacitors) also play a major role in affecting noise in VLSI chips. The ideal reactive components are not associated with any noise sources. However, non-ideal properties such as series resistance in inductors and leakage currents in capacitors give rise to thermal and shot noise, respectively. Moreover, the reactive components introduce the frequency dependence of noise or cause phase shifts between voltage and current noise sources. Finally, the thermal noise bandwidth of a resistance R or a conductance G is limited by the series inductance L or the shunt capacitance C, respectively. The resulting integrated thermal noise powers are given by:

$$\langle i_n^2 \rangle = \frac{kT}{L} \tag{9.24a}$$

$$\langle v_n^2 \rangle = \frac{kT}{C} \tag{9.24b}$$

Here, i_n and v_n denote the noise current and noise voltage, respectively, integrated over the whole frequency bandwidth.

Diodes

A p-n junction is characterized by a depletion region and an energy barrier. The depletion region can be modeled by a frequency-dependent admittance, comprised of a diffusion plus depletion conductance $G_J(\omega)$ and a diffusion plus depletion capacitance $C_J(\omega)$. In the presence of a bias across the junction, the current-voltage characteristic is given by:

$$I_D = I_S \cdot \left\{ \exp\left(\frac{qV}{kT}\right) - 1 \right\} \tag{9.25}$$

Here, I_D is the total current through the diode, I_S the reverse saturation current, and V the forward voltage across the diode. The junction noise is shown to be arising from the shot noise due to current across the junction, and the thermal noise from the conductance of the junction:

$$S_I(\omega) = 2q(I_D + 2I_S) + 4kT[G_J(\omega) - G_J(0)] \tag{9.26}$$

Here, the frequency-dependent junction conductance is denoted by $G_J(\omega)$ and the low-frequency junction conductance is given by:

$$G_J(0) = \frac{q}{kT} \cdot (I_D + I_S) \tag{9.27}$$

Note that the general expression for the diode noise (Eq. 9.26) reduces to the pure shot noise form 2qI at low frequencies and to the thermal noise form 4kTG at zero bias.

Bipolar Junction Transistors

Figure 9.2 shows a schematic of a bipolar junction transistor (BJT) in a conventional device layout. The macroscopic noise sources in such a BJT are thermal noise from all resistive regions (especially from the

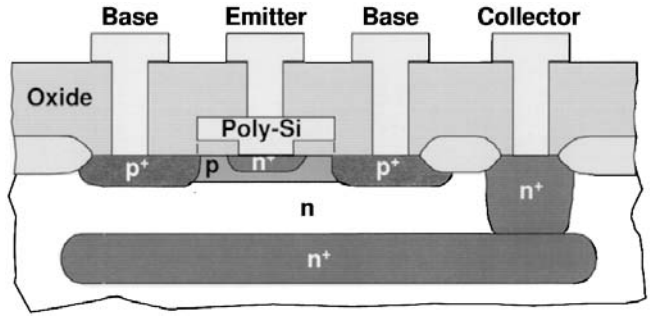


FIGURE 9.2 Schematic cross-section of an npn bipolar junction transistor in a conventional device layout.

extrinsic base region), shot noise from the base–emitter and base–collector junctions, and flicker noise from base and collector currents. These noise sources can be summarized as:

$$S^t_{R_x} = 4kTR_x \tag{9.28a}$$

$$S^s_{I_B} = 2qI_B \tag{9.28b}$$

$$S^s_{I_C} = 2qI_C \tag{9.28c}$$

$$S^f_{I_B} = K_B \cdot \frac{I_B^\gamma}{f^\alpha} \tag{9.28d}$$

$$S^f_{I_C} = K_C \cdot \frac{I_C^\gamma}{f^\alpha} \tag{9.28e}$$

The flicker noise magnitudes are strongly dependent on device processing and technology, and are summarized in the pre-factors K for each term. Note that, in general, the flicker noise exponents α and γ are different for base and collector currents. The resistor R_x denotes the total of base, emitter, and collector resistances, and the individual thermal noise terms need to be included appropriately within the device model. Since the noise sources are distributed in various regions of the device, it is useful to refer the noise either at the input or the output in order to estimate their contributions to the overall noise performance of the device. Here, we choose to refer the noise to the input of the device. The total input referred noise is given by taking into account the transconductance g_m:

$$S^{IN}_{tot} = S^t_{R_x} + (S^s_{I_B} + S^f_{I_B}) \cdot R_x^2 + (S^s_{I_C} + S^f_{I_C}) \cdot \frac{(R_x + Z_\pi)^2}{g_m^2 Z_\pi^2} \tag{9.29}$$

In the above expression, the input impedance relevant for noise is denoted as Z_π, indicating a π-model for the small-signal equivalent circuit of the device. For high gain BJTs, the general expression can be simplified and separated to obtain the voltage and current noise terms.

The input referred voltage noise is given by:

$$S^{IN}_v = 4kTR_B + \left(2qI_B + K_B \cdot \frac{I_B^\gamma}{f^\alpha}\right)R_B^2 + 2qI_C \frac{[|R_B + Z_\pi|^2]}{g_m^2 |Z_\pi|^2} \tag{9.30}$$

The input referred current noise is given by:

$$S_i^{IN} = 2qI_B + K_B \cdot \frac{I_B^{\gamma}}{f^{\alpha}} + \left(2qI_C + K_C \cdot \frac{I_C^{\gamma}}{f^{\alpha}}\right) \cdot \frac{1}{g_m^2 |Z_{\pi}|^2} \tag{9.31}$$

The input referred voltage and current noise expressions can be used to completely characterize the noise properties of an individual device. The circuit noise performance is obtained by deriving the noise parameters from the above expressions, as described in the next section. Note that all types of bipolar devices (heterojunction bipolar transistors, etc.) can be treated in a manner similar to that given here. The overall frequency dependence of noise in BJTs is discussed, together with that of MOSFETs, at the end of the next section.

Field Effect Transistors

Figure 9.3 shows a schematic of an n-channel MOSFET with a conventional device layout. The macroscopic noise sources in a MOSFET are thermal noise from resistive regions (especially from the gate resistance), shot noise from gate leakage current, dynamic channel thermal noise from drain current, and flicker noise from drain and gate currents. These noise sources can be summarized as:

$$S_{R_x}^{t} = 4kTR_x \tag{9.32a}$$

$$S_{I_G}^{s} = 2qI_G \tag{9.32b}$$

$$S_{I_D}^{t} = \beta \cdot 4kTg_{\mu} \tag{9.32c}$$

$$S_{I_G}^{f} = K_G \cdot \frac{I_G^{\gamma}}{f^{\alpha}} \tag{9.32d}$$

$$S_{I_D}^{f} = K_D \cdot \frac{I_D^{\gamma}}{f^{\alpha}} \tag{9.32e}$$

Note that, except for the channel thermal noise of the drain current, the intrinsic noise sources in a MOSFET are very similar to that of a BJT. The pre-factor β to the channel thermal noise was introduced

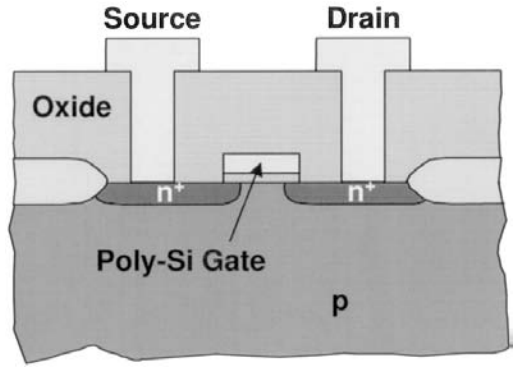


FIGURE 9.3 Schematic cross-section of an n-channel MOSFET in a conventional device layout.

to account for the specific operation of FETs in general. It was shown to have a value of 2/3 for MOSFETs and a value between 1/3 and 2/3 for JFETs. For sub-micron MOSFETs, high field effects are known to increase its value to well above 2/3. In a manner similar to that used for BJTs, one can express the noise at the input by taking into account the transconductance of the device, and then separate into voltage and current contributions.

The total input referred noise is given by:

$$S_{tot}^{IN} = S_{R_x}^{t} + (S_{I_G}^{s} + S_{I_G}^{f}) \cdot R_x^2 + (S_{I_D}^{t} + S_{I_D}^{f}) \cdot \frac{(R_x + Z_\pi)^2}{g_m^2 Z_\pi^2} \tag{9.33}$$

The input referred voltage noise is given by :

$$S_v^{IN} = 4kTR_G + \left(2qI_G + K_G \cdot \frac{I_G^\gamma}{f^\alpha}\right) \cdot R_G^2 + \left(\beta \cdot 4kTg_m + K_D \cdot \frac{I_D^\gamma}{f^\alpha}\right) \frac{[|R_G + Z_\pi|^2]}{g_m^2 |Z_\pi|^2} \tag{9.34}$$

The input referred current noise is given by :

$$S_i^{IN} = 2qI_G + K_G \cdot \frac{I_G^\gamma}{f^\alpha} + \left(\beta \cdot 4kTg_m + K_D \cdot \frac{I_D^\gamma}{f^\alpha}\right) \frac{1}{g_m^2 |Z_\pi|^2} \tag{9.35}$$

The overall frequency behavior of the noise in FETs is similar to that of BJTs; however, the contributions from the device parameters are different in each case specific to the device properties. In the above expressions, the flicker noise becomes predominant below a "corner" frequency f_{c1}; whereas above a second "corner" frequency f_{c2}, the noise increases with frequency. Figure 9.4 illustrates this behavior schematically, where the high-frequency increase of noise is referred to as being due to impedance coupling. In this region, the thermal or shot noise from one region of the device is coupled into other regions due to the reactive components and gives rise to frequency dependence. Figure 9.5 also illustrates the overall behavior of noise figure vs. frequency measured on devices from different technologies. The

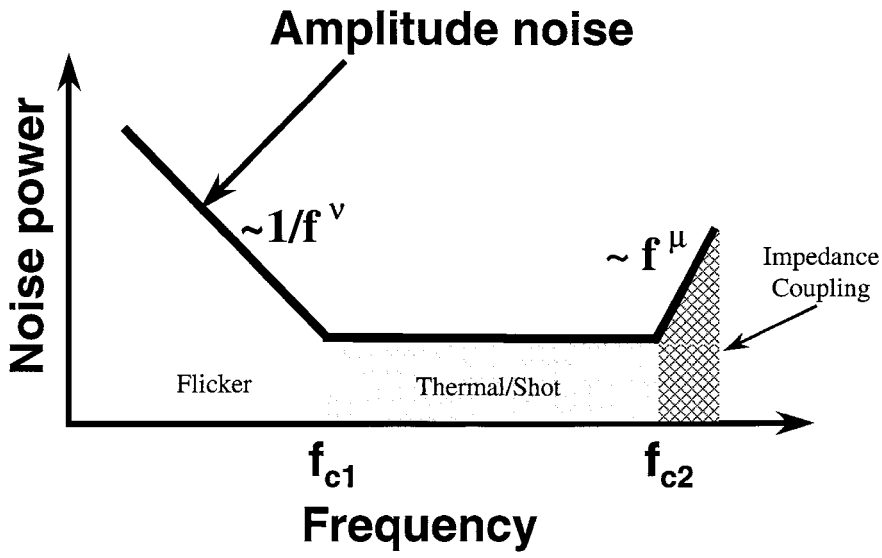


FIGURE 9.4 Schematic of amplitude noise as function of frequency, indicating the three distinct regions of noise (flicker, thermal/shot, and impedance coupling).

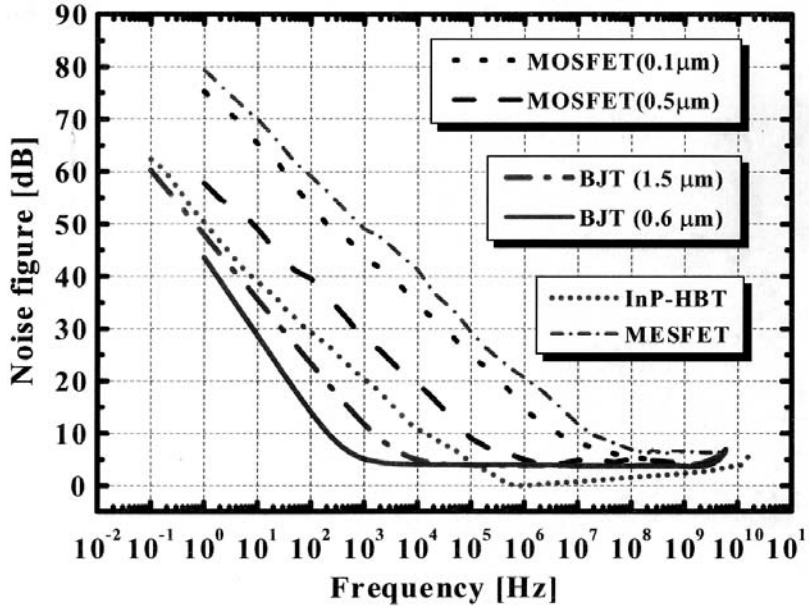


FIGURE 9.5 Measured noise figure vs. frequency for various device technologies. The plot indicates the relative magnitude of the three noise regions shown schematically in Fig. 9.4. The plot also presents the wide range of noise behavior found in device technologies.

bias-dependent behavior of noise in the low-frequency flicker noise regime is illustrated in Fig. 9.6, where measured output current noise power at 10 Hz is plotted as function of drain current for n-channel MOSFETs, with different gate lengths as given in the legend of the figure. Note that although all devices exhibit power law behavior, the exponent increases with decreasing gate length.

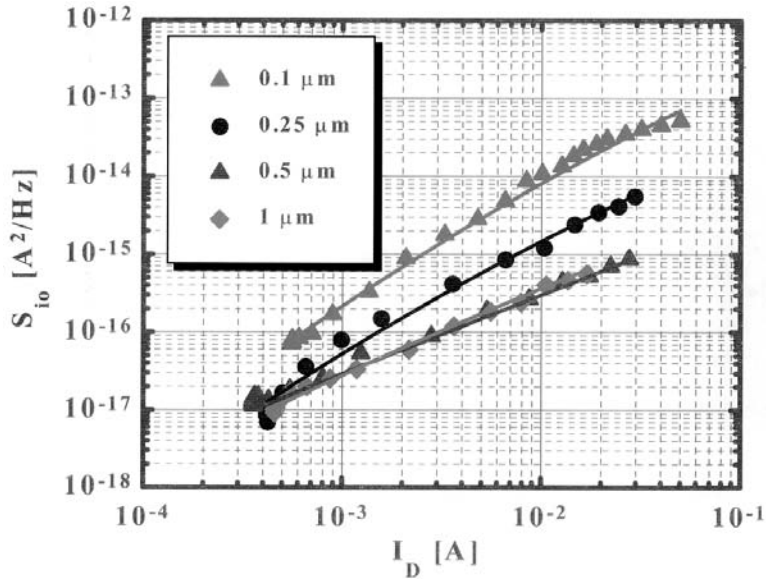


FIGURE 9.6 Output current noise spectral density vs. drain current for a set of n-channel MOSFETs with equal gate widths and different lengths, as given in the legend. Note that all devices exhibit power-law bias dependence, but the exponent increases with decreasing gate length.

9.4 Chip Noise

The device noise mechanisms discussed in Section 9.3 give rise to noise at the chip level in a number of ways. This section outlines the principal noise mechanisms encountered in amplifiers, oscillators, as well as timing jitter and interconnect noise in VLSI circuits.

Amplifier Noise

Amplifier noise is a small-signal phenomenon and can thus be treated by a linear approximation. It is directly determined by the noise and gain of the devices used in the design. The noise figure of an amplifier can be computed from the noise parameters of the active devices used in the circuit. The noise parameters are a set of four quantities that characterize the noise of any 2-port (e.g., active device, amplifier, etc.) completely. They are derived from the input referred voltage and current noise sources of the 2-port in the following manner. For a noise voltage v_n and a noise current i_n at the input of a 2-port, the corresponding noise correlation matrix is given by:

$$\hat{C} = \frac{1}{2B} \cdot \begin{bmatrix} \langle v_n^2 \rangle & \langle v_n i_n^* \rangle \\ \langle v_n^* i_n \rangle & \langle i_n^2 \rangle \end{bmatrix} \tag{9.36}$$

Here, B denotes the frequency bandwidth. The voltage and current noise sources used in the expression above are related to the noise spectral densities derived in the previous sections as follows:

$$S_v^{IN} = \frac{\langle v_n^2 \rangle}{B} \tag{9.37a}$$

$$S_i^{IN} = \frac{\langle i_n^2 \rangle}{B} \tag{9.37b}$$

The noise parameters are the minimum noise figure F_{min}, the complex optimum source impedance Z_{opt}, and the noise conductance g_n. They are expressed in terms of the input referred noise voltage and current as follows:

$$F_{min} = 1 + \frac{[v_n + Z_{opt} \cdot i_n]^2}{4kTBZ_{opt}} \tag{9.38a}$$

$$Z_{opt} = \sqrt{\frac{\langle v_n^2 \rangle}{\langle i_n^2 \rangle}} \tag{9.38b}$$

$$g_n = \frac{\langle i_n^2 \rangle}{4kTB} \tag{9.38c}$$

The correlation matrix expressed in terms of the four noise parameters is given by:

$$\hat{C} = 2kT \cdot \begin{bmatrix} g_n |Z_{opt}|^2 & \left[\frac{(F_{min} - 1)}{2} - g_n Z_{opt}\right] \\ \left[\frac{(F_{min} - 1)}{2} - g_n Z_{opt}^*\right] & g_n \end{bmatrix} \tag{9.39}$$

The noise parameters are directly measurable quantities for an active device or circuit and, thus, the noise correlation matrix can be evaluated quantitatively. The overall noise figure of a linear 2-port depends not only on its four noise parameters, but also on the source impedance as given by:

$$F = F_{min} + \frac{g_n}{R_s} \cdot |Z_s - Z_{opt}|^2 \tag{9.40}$$

Here, R_s is the real part of the source impedance Z_s. For a cascade of amplifiers with individual noise figures F_i and gain G_i ($i = 1, 2, \ldots$), the total noise figure is given by:

$$F = F_1 + \frac{(F_2 - 1)}{G_1} + \frac{(F_3 - 1)}{G_1 G_2} + \ldots \tag{9.41}$$

The above expression shows that the primary contribution to the total noise figure of any amplifier comprised of several stages is from the initial stages. A quantity that takes into account both noise figure and gain of an amplifier is the noise measure given by:

$$M = \frac{F - 1}{1 - (1/G)} \tag{9.42}$$

The noise measure can be used as a figure-of-merit for the overall noise–gain performance of a linear amplifier for analog applications. As described by Eq. 9.41, the primary contribution to amplifier noise can be attributed to the noise of the input active device. Hence, an experimental study of noise in devices provides valuable information for optimum design of low noise amplifiers.

For the purpose of illustrating the main features of amplifier noise, we show measured RF noise behavior of an active device. Figure 9.7 shows the frequency dependence of minimum noise figure, input

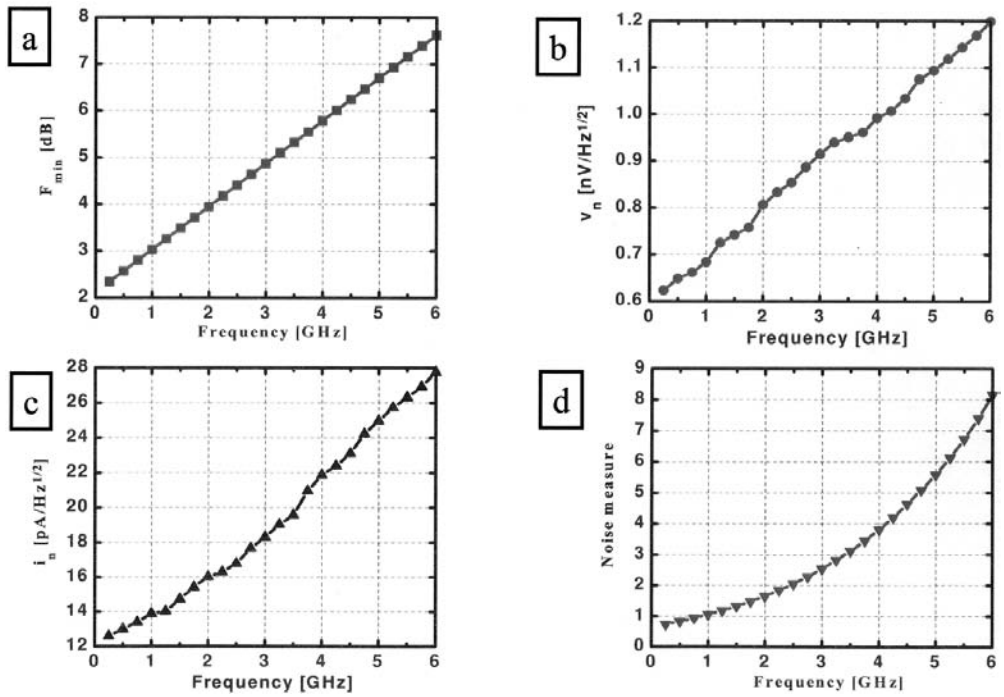


FIGURE 9.7 Plots of minimum noise figure, input referred voltage noise, input referred current noise, and noise measure vs. frequency for an npn BJT with an emitter width of 0.5 μm technology. The plots show the increase of noise with frequency in the impedance coupling regime.

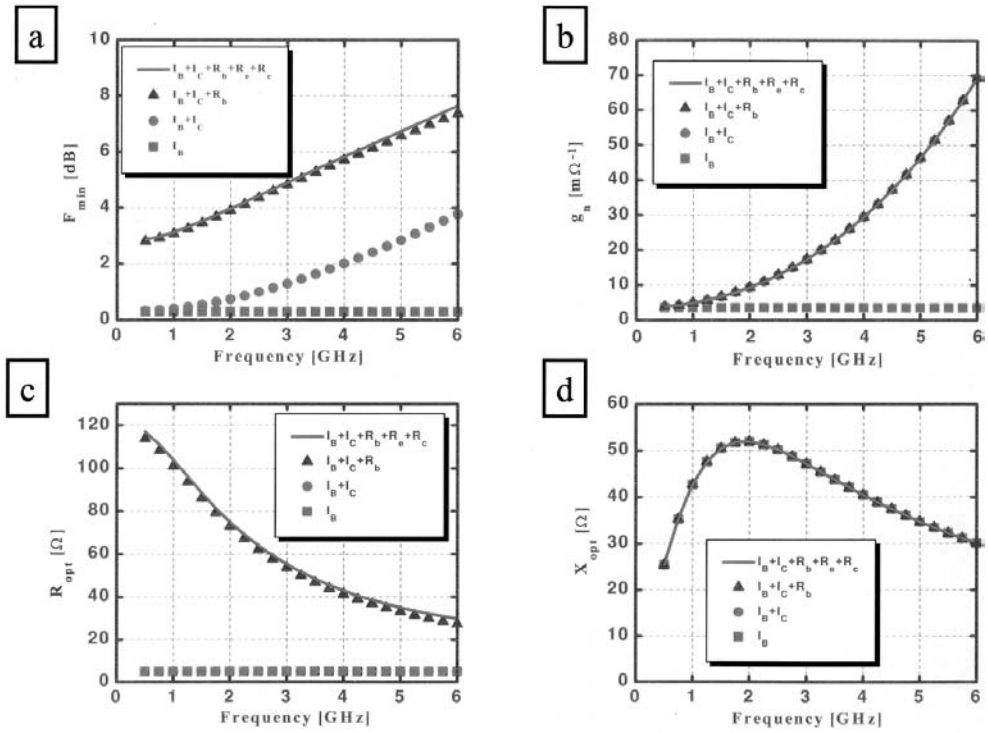


FIGURE 9.8 Plots of the four RF noise parameters vs. frequency obtained from fitting measured data on an npn BJT to a small-signal model of the device. The different curves in each plot indicate the individual contributions from the various regions of the device, as discussed in the text.

referred voltage noise, input referred current noise, and noise measure for a BJT. Note that in the frequency range shown in the figure, all quantities increase with frequency corresponding to the impedance coupling regime. The voltage noise is a measure of the thermal noise of the input resistance, and the current noise is a measure of the input referred shot noise. In order to optimize the device technology for low noise applications, it is important to study the relative contributions from the various regions of an active device to the total noise power. This is done most conveniently by extracting a small-signal noise model of the device from measured data. After successively removing the noise contributions from various regions, the resulting noise parameters can be plotted and compared. Figure 9.8 shows the results of such noise modeling of a BJT. The four noise parameters are plotted vs. frequency, with the individual contributions from the various regions of the device illustrated in the figure. Here, it is seen that the major contribution to the minimum noise figure is the base resistance, followed by the base–collector shot noise at higher frequencies. The noise conductance is primarily determined by the base–collector shot noise. The optimum source resistance is given by the thermal noise of the base resistance. Finally, the optimum source reactance is not explicitly determined by any of the noise sources, but by the input impedance of the device.

Oscillator Noise

The major noise source in oscillators is phase noise caused by mixing of device noise (flicker, shot, thermal) with the carrier frequency due to the nonlinear operation of the circuits. In general, noise in oscillators is a large-signal phenomenon and, hence, linearization techniques have limited applicability. Several approaches have been published for calculating the phase noise. The simplest is that of Leeson, which estimates the phase noise of oscillators due to nonlinear mixing of device flicker noise, and finite bandwidth of the oscillator. More advanced approaches include both analytical and numerical studies,

as given in the references. Here, we briefly outline the fundamental features of noise in a negative conductance LCR oscillator driven by an external current source (e.g., an active device). In the presence of small-signal noise, the output voltage of the oscillator with an intrinsic amplitude v_0 and frequency ω_0 is given by:

$$v(t) = v_0 \cdot [1 + a(t)] \cdot \cos\{\omega_0 t - \varphi(t)\} \tag{9.43}$$

The amplitude fluctuations a(t) and phase fluctuations $\phi(t)$ can be evaluated to first order by just taking into account the mixing of the noise with the free oscillations. The spectral density of the amplitude fluctuations (AM noise) is given by:

$$S_a(\Delta\omega) = \frac{S_I(\omega)}{2v_0^2(G_L - G_0)^2 \cdot \left[1 + Q_0^2\left(\frac{\Delta\omega}{\omega_0}\right)^2\right]} \tag{9.44}$$

Here, G_L is the loss conductance, G_0 the lowest-order term of an expansion of the nonlinear negative conductance, and Q_0 the Q-factor of the oscillator. The frequency offset from the fundamental oscillator is denoted by $\Delta\omega$. The spectral density of the phase fluctuations (PM noise) is given by:

$$S_\varphi(\Delta\omega) = \frac{2S_I(\omega)}{v_0^2(G_L - G_0)^2 \cdot Q_0^2\left(\frac{\Delta\omega}{\omega_0}\right)^2} \tag{9.45}$$

Both amplitude and phase noise exhibit a $\propto 1/\Delta\omega^2$ dependence as a function of frequency offset from the oscillation frequency. Note that the phase noise is always larger than the amplitude noise and is therefore of most relevance to chip design. Moreover, a frequency-dependent current noise $S_I(\omega) \propto 1/\omega$ (flicker noise of device) gives rise to a $\propto 1/\Delta\omega^3$ dependence of phase noise. Figure 9.9 shows schematically a simplified case of oscillator noise with two typically encountered types of phase noise frequency dependencies. For small offset frequencies, the phase noise is caused by up-conversion of the device flicker noise. At larger offset frequencies, the phase noise shows a behavior depending on the value of the flicker noise corner frequency with respect to that of the oscillator bandwidth.

Timing Jitter

In digital circuits, the timing of pulses exhibits a random fluctuation as a function of time, which is referred to as *jitter*. Jitter is fundamentally caused by the noise of individual components of the circuit (active devices, resistors, interconnect delays, etc.) and is of primary concern to the design of low-noise oscillators. As signal levels decrease and clock frequencies increase, jitter becomes a serious noise phenomenon in VLSI circuits. Jitter can be expressed in terms of fluctuations of the fundamental period of oscillation. The distribution of these fluctuations has a standard deviation that is used as a measure of jitter:

$$\sigma_T = \frac{\sqrt{\langle T^2\rangle - \langle T\rangle^2}}{\langle T\rangle} \tag{9.46}$$

For example, the thermal noise of collector load resistors in a differential pair stage of a ring oscillator is shown to result in jitter given by:

$$\sigma^t_{R_C} = \sqrt{2kTC_C} \cdot f(I_E, R_C) \tag{9.47}$$

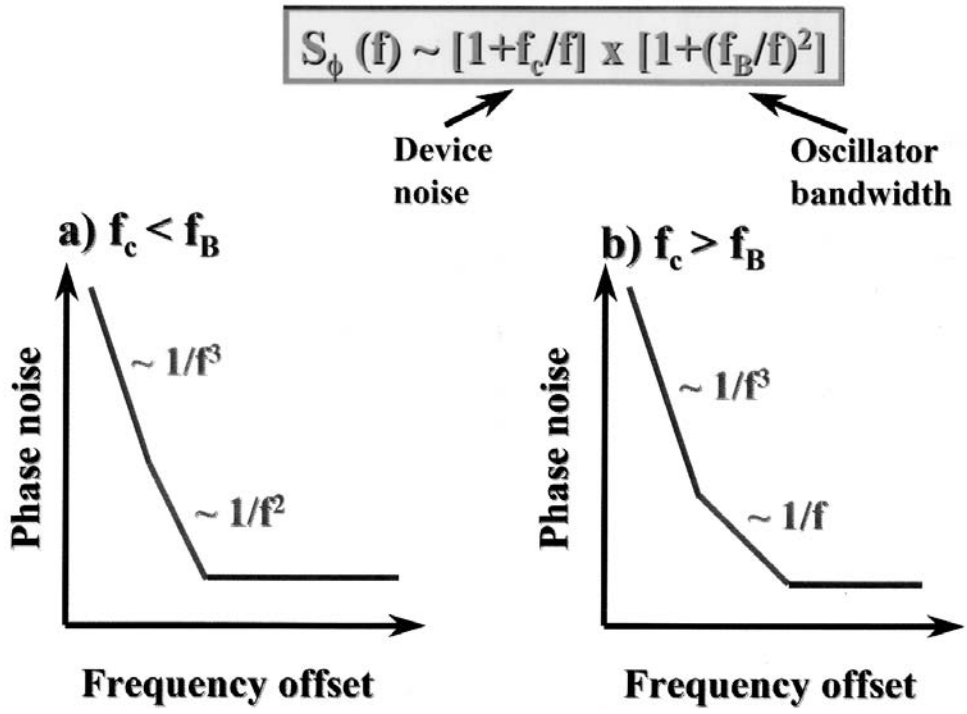


FIGURE 9.9 Schematic of frequency dependence of phase noise vs. offset from the fundamental frequency. In a simplified model, the phase noise can be treated as a product of two terms resulting from device noise and oscillator bandwidth. The two cases shown relate the flicker noise corner frequency f_c to the oscillator bandwidth f_B.

Note that the above expression contains the voltage noise fluctuations from the collector resistance R_C and shunt capacitance C_C. The thermal noise of transistor input resistances (e.g., base or gate resistance) is shown to give rise to a jitter of the form:

$$\sigma^t_{R_X} = \sqrt{4kTR_X} \cdot f(I_E, C_C) \tag{9.48}$$

Note that here the thermal noise of the input resistance R_x directly affects the jitter of the circuit. Similarly, other noise sources at the oscillator input will modulate the fundamental frequency and contribute proportionally to jitter.

Interconnect Noise

Interconnects in VLSI chips are a source of noise referred to as crosstalk, which originates from capacitive and inductive coupling of signals between transmission lines. A quantitative treatment of crosstalk requires an appropriate modeling of the impedance of the transmission lines. At sufficiently high frequencies, the wavelength of the propagating electromagnetic wave becomes comparable to the length scales of the transmission lines. In this regime, the transmission needs to be treated as a distributed entity. The electrical properties of a distributed transmission line can be characterized by the characteristic impedance Z_0 and the propagation coefficient γ. For a "quasi-TEM" mode of propagation along the line, these quantities are given by:

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}} \tag{9.49a}$$

$$\gamma = \sqrt{(R + j\omega L)(G + j\omega C)} \tag{9.49b}$$

Here, the transmission line is characterized by a lateral resistance per unit length R, a lateral inductance per unit length L, a perpendicular conductance per unit length G, and a perpendicular capacitance per unit length C. The magnitude of the crosstalk between two lines is determined by their mutual capacitance C_m and mutual inductance L_m. Since the capacitive noise is proportional to $C_m \cdot dV/dt$, and the inductive noise is proportional to $L_m \cdot dI/dt$, the crosstalk can be treated as being proportional to V_{DD}/τ and proportional to I_{sat}/τ, where τ is a pulse rise or fall time.

Another more fundamental source of noise in VLSI interconnects is the thermal noise of transmission lines. It can be shown that by treating the interconnect as a distributed transmission line, the thermal noise can be expressed as:

$$S_v^t = 4kTR \cdot f(\gamma, l) \tag{9.50}$$

In the above expression, the thermal noise due to the resistance R of the interconnect is modified by a function of only the propagation coefficient γ and the total length l of the interconnect.

9.5 Future Trends

Scaling

The scaling of silicon technology with decreasing minimum feature size is associated with changes in active device parameters as well as changes in passive components and interconnect dimensions. The effect of technology scaling on noise in VLSI chips can be estimated by considering the dependence of the noise of the individual components of the chip on their scaled properties. It turns out that general expressions for technology scaling of noise cannot be formulated, since device dimensions are chosen specific to the chip design. However, one can observe certain trends for active devices. It is found that in MOSFETs, the flicker noise scaling is primarily determined by changes in the gate-oxide quality (trap density in the oxide) and roughness of the oxide–channel interface (carrier mobility in the channel). At RF frequencies, the noise is dependent on device dimensions (base resistance of BJT, gate resistance of MOSFET, etc.) and on small-signal properties (base-collector capacitance, gate-channel capacitance, etc.). Although the specific choice of device dimensions primarily determines its noise, the trend of increasing cutoff frequencies results in improvement of RF noise at a given frequency due to decreased device parasitics. However, the thermal noise of interconnects increases as the technology is scaled to smaller dimensions, due to the associated decrease of cross-sectional area.

Processing

Process development in VLSI technologies is intimately related to device performance. As described above, the RF noise in BJTs is significantly determined by thermal noise in the base resistance and by shot noise from the base–collector junction. These contributions can be minimized by utilizing super-self-aligned device layouts, as shown schematically in Fig. 9.10. In this case, the extrinsic base region is significantly reduced to decrease the base thermal noise. The effect of the shot noise from the base–collector junction is minimized by reducing the base–collector capacitance that acts as a feedback capacitance and couples the collector current shot noise into the base. In MOSFETs, a reduction of the gate resistance is achieved through the use of higher conducting gate stacks (Fig. 9.11) or a multi-finger layout. Moreover, the coupling of channel thermal noise into the gate region can be minimized by reducing the gate–channel capacitance. In general, processing has a strong influence on the flicker noise of semiconductor devices. In BJTs, the passivation of the base–emitter junction region improves flicker noise substantially. In MOSFETs, the quality of the gate-oxide and its interface with the channel

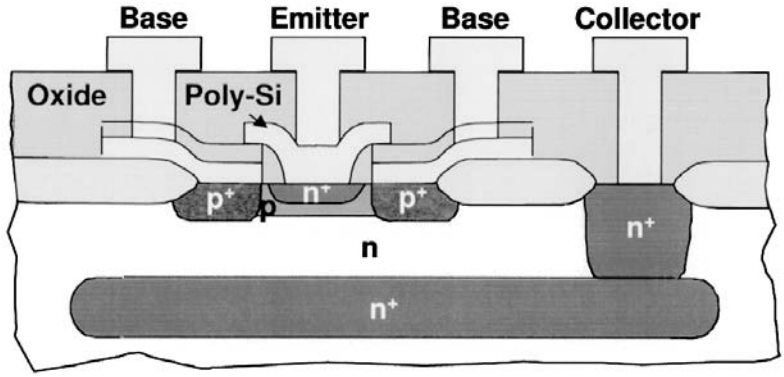


FIGURE 9.10 Schematic of a super-self-aligned BJT layout used to reduce base thermal noise and the effect of base–collector shot noise.

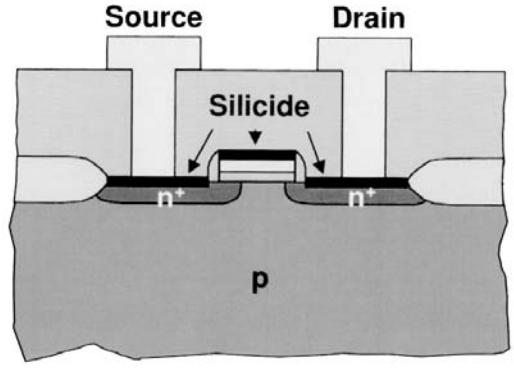


FIGURE 9.11 Schematic of a silicide-gate MOSFET used to reduce the thermal noise from the gate resistance.

affects flicker noise and can be improved through annealing treatments. One of the approaches being studied presently for reducing RF noise in MOSFETs is the use of silicon-on-insulator substrates instead of conducting substrate material, in order to reduce parasitic elements of the device.

Non-equilibrium Transport

In deep-submicron devices at high electric fields, the assumption of thermal equilibrium between carriers and lattice is not valid. Here, we briefly indicate the type of corrections required for estimating the noise in non-equilibrium transport.

The diffusion coefficient in non-equilibrium transport can be modified by approximating it to be a simple frequency-dependent quantity of the form:

$$D(\omega) = \frac{D(0)}{1 + (\omega\tau)^2} \tag{9.51}$$

The resulting thermal noise is then given by:

$$S_I(\omega) = \frac{S_I(0)}{1 + (\omega\tau)^2} \tag{9.52}$$

In the above expression, the quantity τ is the mean scattering time of the carriers. A further effect of non-equilibrium transport are velocity-velocity correlations, which modify the macroscopic values of the noise spectral densities and need to be considered in detail.

The general expression for shot noise in semiconductors needs to include finite carrier transit times and is given by:

$$S_I(\omega) = 2qI \cdot \left[\frac{\langle N^2\rangle - \langle N\rangle^2}{\langle N\rangle}\right] \cdot f(\omega, \tau, \tau_T) \tag{9.53}$$

Here, N is the number of carriers, τ the mean scattering time, and τ_T the carrier transit time. We can consider several limiting cases of the above expression. For sufficiently low frequencies and small transit times, such that $\omega << 1/\tau << 1/\tau_T$, the shot noise is given by:

$$S_I(\omega) = 2qI \cdot \left[\frac{\langle N^2\rangle - \langle N\rangle^2}{\langle N\rangle}\right] \tag{9.54}$$

The above expression is the full shot noise in a semiconductor sample where all the carriers are traversing the sample along the field direction. For a Poisson distribution function of carrier number, the variance of the fluctuations is equal to the mean of the distribution, and then the simple expression 2qI is recovered for the shot noise. For sufficiently small scattering times, such that $\tau << \tau_T$, the shot noise is given by:

$$S_I(\omega) = 2qI \cdot \left[\frac{\langle N^2\rangle - \langle N\rangle^2}{\langle N\rangle}\right] \cdot \left\{\frac{2\tau}{\tau_T} \cdot \frac{1}{1 + (\omega\tau)^2}\right\} \tag{9.55}$$

The above expression indicates that the shot effect is reduced by the randomizing due to scattering of carriers. At high frequencies, the shot noise also shows a decrease due to effective acceleration of carriers along the field direction.

9.6 Conclusions

In this chapter, the issue of noise in VLSI technologies was treated by introducing the microscopic noise mechanisms from fundamental carrier transport in semiconductors. The noise properties of active semiconductor devices and passive components were given in a general form and certain approximations relevant for most applications were outlined. The noise at the chip level was presented in terms of specific circuits where noise-critical applications are realized. Finally, trends in future VLSI technologies and their implications for noise were briefly mentioned.

Acknowledgments

Collaborations with the following organizations within Lucent Technologies are gratefully acknowledged: VLSI Technology Integration Department in Orlando, FL; Analog Design Methodology Group in Reading, PA; Compact Modeling and Measurements Group in Cedar Crest, PA. In addition, we gratefully acknowledge the many helpful discussions with colleagues from the Physical Research, Silicon Research, and Wireless Research Laboratories at Bell Laboratories.

References

Section 9.1

1. Meindl, J. D., "Ultra-Large Scale Integration," *IEEE Trans. Elec. Dev.*. ED-31, 1555, 1984.

2. van der Ziel, A. and Amberiadis, K., "Noise in VLSI," *VLSI Electronics,* Vol. 7, Academic Press, 1984, 261ff.
3. van der Ziel, A., "Noise in VLSI," *VLSI Handbook,* Academic Press, 1985, 603ff.
4. Tsividis, Y., *Mixed Analog-Digital VLSI Devices and Technology*, Mc-Graw Hill, 1995.
5. Martin, S., Archer, V., Boulin, D., Frei, M., Ng, K., and Yan, R.-H., "Device Noise in Silicon RF Technologies," *Bell Labs Technical Journal,* 2(3), 30, 1997.

Section 9.2

6. van der Ziel, A., *Noise in Measurements*, Wiley, 1976.
7. Gupta, M. S. (ed.), *Electrical Noise: Fundamentals & Sources*, IEEE Press, 1977.
8. Buckingham, M. J., "Noise in Electronic Devices and Systems," Wiley (1983).
9. Ferry, D. K. and Grondin, R. O., *Physics of Submicron Devices*, Plenum Press, 1991.

Section 9.3

10. Motchenbacher, C. D. and Connelly, J. A., *Low Noise Electronic System Design*, Wiley, 1993.
11. Ng, K., *Complete Guide to Semiconductor Devices*, Mc-Graw Hill, 1995.
12. Martin, S., Booth, R., Chyan, Y.-F., Frei, M., Goldthorp, D., Lee, K.-H., Moinian, S., Ng, K., and Subramaniam, P., "Modeling of Correlated Noise in RF Bipolar Devices," *IEEE MTT-S Digest,* MTT-Symposium, 1998, 941.
13. See also Refs. 6, 7, and 8.

Section 9.4

14. Hillbrand and Russer, "An Efficient Method for Computer Aided Noise Analysis of Linear Amplifier Networks," *IEEE Trans. Circuits and Systems,* CAS-23, 235, 1976.
15. Abidi, A. and Meyer, R.G., "Noise in Relaxation Oscillators," *IEEE J. Solid-State Circuits,* SC-18, 794, 1983.
16. Brews, J. R., "Electrical Modeling of Interconnections," *Submicron Integrated Circuits* (Ed. R. K. Watts), Wiley, 1989.
17. Vendelin, G. D., Pavio, A. M., and Rohde, U. L., *Microwave Circuit Design*, Wiley, 1990.
18. Gray, P. R. and Meyer, R. G., *Analog Integrated Circuits*, Wiley, 1993.
19. Engberg, J. and Larsen, T., *Noise Theory of Linear and Nonlinear Circuits*, Wiley, 1995.
20. Verghese, N. K., Schmerbeck, T. J., and Allstot, D. J., *Simulation Techniques and Solutions for Mixed-Signal Coupling in Integrated Circuits*, Kluwer, 1995.
21. Schaeffer, D. K., and Lee, T. H., "A 1.5 V, 1.5 GHz CMOS Low Noise Amplifier," *IEEE J. Solid-State Circuits,* 32, 745, 1997.
22. McNeill, J. A., "Jitter in Ring Oscillators," *IEEE J Solid-State Circuits,* 32, 370, 1997.
23. Restle, P. J., Jenkins, K. A., Deutsch, A., and Cook, P. W., "Measurement and Modeling of On-Chip Transmission Line Effects in a 400 MHz Microprocessor," *IEEE J. Solid-State Circuits,* 33, 662, 1998.
24. Demir, A., Mehrotra, A., andRoychowdhury, J. , "Phase Noise in Oscillators : A Unifying Theory and Numerical Methods for Characterization," *IEEE Trans. Circuits and Systems-1: Fundamental Theory and Application*, to be published, 1999.
25. See also Refs. 6 and 8.

Section 9.5

26. Simeon, E. and Claeys, C., "The Low-Frequency Noise Behavior of Silicon-on-Insulator Technologies," *Solid-State Electronics*, 39, 949, 1996.
27. Abou-Allam, E. and Manku, T., "A Small-Signal MOSFET Model for Radio Frequency IC Applications," *IEEE Trans. CAD of IC and Systems,* 437, 1997.
28. Tin, S. F., Osman, A. A., and Mayaram, K., "Comments on "A Small-Signal MOSFET Model for Radio Frequency IC Applications," *IEEE Trans. CAD of IC and Systems,* 17, 372, 1998.
29. See also Refs. 7 and 9.

Hesketh, P.J. "Micromachining"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

10 Micromachining

Peter J. Hesketh
The Georgia Institute of Technology

10.1 Introduction

There has been a tremendous growth in activity over the past seven years in exploring the use of micromachining for the fabrication of novel microstructures, microsensors, and microdevices and also their integration with electronic circuits. Specific application areas have developed to such an extent that there are specialist meetings on the following topics: sensors and actuators,[1] microelectromechanical system (MEMS),[2] microchemical analysis systems (μTAS),[3] optical-MEMS,[4] MEMS-electronics,[5] chemical sensors,[7] and microstructures and microfabricated systems.[7] Early examples of MEMS devices and innovations are reviewed in Peterson's[8] classic paper. Up until recently, the MEMS micromachining processes were developed outside the realm of a CMOS line although the advantages of IC fabrication processes and the economies of batch fabrication have been used for production of microdevices at low

cost and high volume. The tremendous successes of microfabricated silicon pressure sensors used for blood pressure monitoring and automotive air intake manifold pressure sensing, ink-jet printer heads, and the air bag accelerometer sensors, and most recently projection overhead display systems, demonstrate the tremendous success of this technology. There are several important differences between MEMS processing and IC processing that make this an exciting and rapidly evolving field:

- Wider range of materials
- Wider range of fabrication processes utilized
- Use of three-dimensional structures
- Material properties are not fully characterized
- Interdisciplinary expertise necessary for successful technology implementation
- CAD tools are not yet fully developed for integrated thermal/mechanical, magnetic, optical, and electronic design

In keeping with the general philosophy of this volume, I will emphasize the fundamental principles and discuss CMOS-compatible processing methods. Selected examples of MEMS devices will be given to illustrate some of exciting applications of this technology. The reader is referred to the comprehensive survey by Göpel et al.[9] and the vast diversity of micromachining and micromanufacturing methods described in recent books by Kovacs,[10] Madou,[11] and Sze.[12] In addition, reference materials include a collection of classic papers by Trimmer,[13] texts on sensors by Middlehoek and Audet,[14] Ristic,[15] Gardener,[16] and bio and chemical sensors texts by Janata,[17] Madou and Morrison,[18] Moseley et al.,[19] and Wilson et al.[20]

10.2 Micromachining Processes

There are a wide range of MEMS processing methods for 2-D and 3-D structures, many of which are incompatible with CMOS fabrication. Micromachining is the process of forming such structures by processes of chemical etching, physical machining, and layering of materials by a diverse range of processes, as listed in Table 10.1. A range of processes for material removal has been developed, including chemical etching,[21] laser ablation,[22] precison machining,[23] focused ion beam etching,[24] ultrasonic drilling and electrodischarge machining,[25] and others. Historically, surface and bulk micromachining have developed independently to produce novel structures and microdevices in two different ranges of dimensional scales. A great deal of work has addressed hybrid approaches that combine bulk and surface processes in novel ways, which will be discussed in the next section. A major obstacle to the further development of MEMS devices is the compatibility of micromachining processes with CMOS circuit processing if integration of electronics is desired. A recent review of micromachining methods for a range of materials of interest is given by Williams and Muller.[26]

10.3 Bulk Micromachining of Silicon

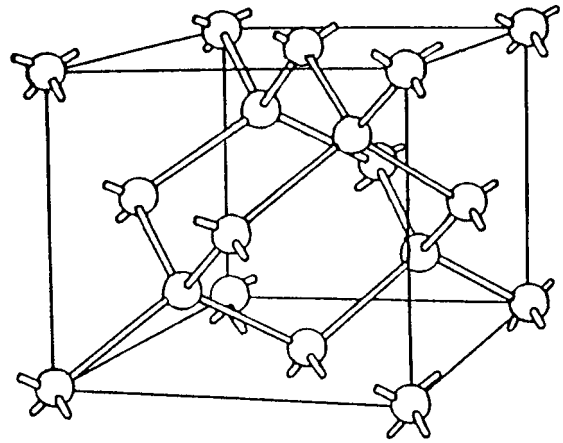
Bulk machining involves etching the bulk single-crystal silicon wafer with respect to a patterned insoluble masking layer that defines the structure on the wafer surface. These methods are either isotropic-independent of direction, or anisotropic-crystal plane selective. Silicon has a diamond lattice as shown in Fig. 10.1(a). The dimensions of the structure in the bulk of the wafer are defined by the crystal plane locations. In particular, the etch rate of the (111) planes are slower than other crystalline directions by typically 35 to 50 times.[27] An SiO_2 or Si_3N_4 mask is used in on etching solution, as shown in Fig. 10.1(b).

Anisotropic etching which is a function of the crystal planes of the Si substrate, occurs in strong bases. The (111) close-packed silicon lattice plane etches slower than other directions of the crystal. Examples of commonly used formulations are given in Table 10.2. This process has been extremely successful for the fabrication of diaphragms for pressure sensors and other devices, aligning the [110] flat of a (100) wafer to a mask opening a rectangular opening which is produced. Etching into the

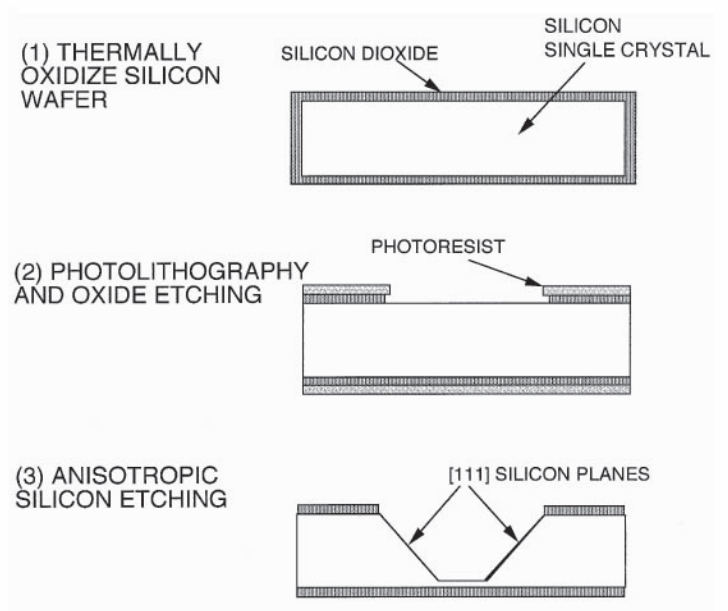
TABLE 10.1 MEMS Processing Technologies

Process	Physical Dimension Range/Aspect Ratio	Materials	Etch Stop Techniques	Through-put	Cost	Ref.
		Subtractive processes				
Bulk micromachining	μm-cm/1:400	Single-crystal silicon, GaAs glass etching	Dopant-selective electrochemical Buried layer	High	Low	9–11, 27–49
Reactive ion etch	μm-mm/1:100	Wide range of materials	Buried layer	Low	High	62–64
Laser ablation	1-100 μm/1:50	Various	Timed	Low	High	22
Electrodischarge machining	2 μm-mm/*	Si, metals	Timed	Low	Med	25
Precision mechanical cutting	nm-cm/*	PMMA	Tool position	Low	High	23
Focussed ion beam machining	nm-μm	various	Timed	Low	High	24
Chemical etching	μm/1:10	Metals, semiconductors, insulators	Timed	High	Low	21,26,50
Ultrasonic machining	25 μm-mm/*	Glass, ceramic, semiconductor, metals	Tool position	Moderate	Moderate	—
		Additive processes				
Physical vapor depositon	Wide range of materials	Electron beam or thermal evaporation/sputtering	—	Moderate	High	26
Chemical vapor deposition	Surface micromachining	LPCVD of polysilicon/PSG or sputtered aluminum/photoresist	Selectivity of sacrificial etch to sacrificial layer to structural layer	High	Moderate	67–71,
Laser-assisted CVD	nm-μm	Various	—	Low	High	—
Molecular beam epitaxy	nm	Semiconductors	—	Low	Very High	—
LIGA	μm-cm	PMMA	—	Low	High	51,101
Electroplating into a mold:	μm-mm	Cu, Ag, Au, Fe, permalloy	—	High	Low	
	μm-mm/1:10	Polyimide	—	High	Moderate	96,98–100
	μm-mm	SU-8	—	High	Low	97
	μm-mm	Thick photoresist	—	High	Low	—

* function of total geometry.



(a)

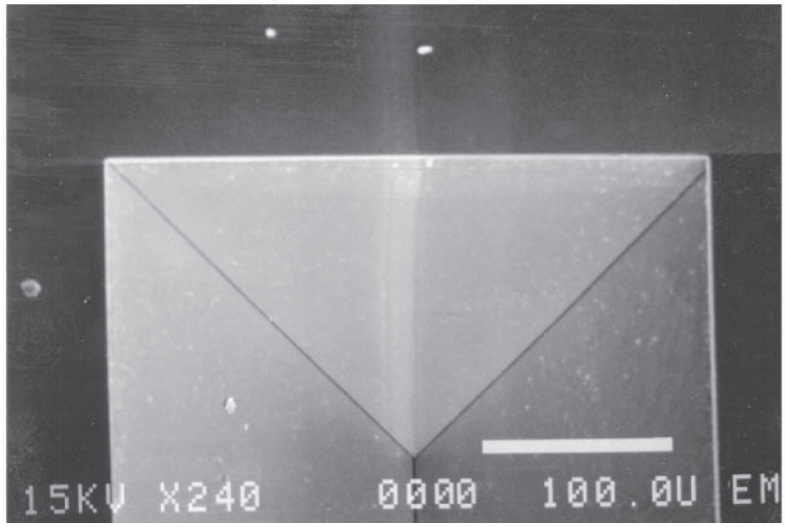


(b)

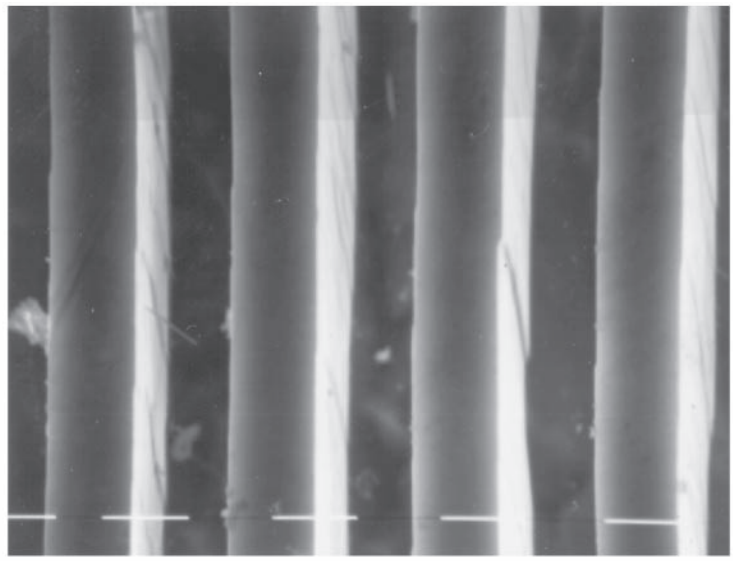
FIGURE 10.1(a-b) (a) Diamond lattice of silicon with principle axis indicated; (b) schematic cross-section of bulk micromachining processing steps to form a cavity in a (100) silicon wafer.

bulk of the silicon crystal the dimensions of the resulting structure is defined by the slow etching {111} planes (see Fig. 10.1c), which are parallel and perpendicular to the [011] flat on a (100) silicon wafer. Alternatively, on a (110) silicon wafer, a slot is produced, as shown in Fig. 10.1(d). Note here that the shape is bounded by the {111} crystal planes in the vertical direction and the {100} planes or {311} planes at the base of the groove. There is insufficient space here to describe the mechanism of etch chemistry, and the reader is referred to the work of Seidel,[28] Kendall,[29] Palik et al.,[30] Hesketh et al.,[31] and Allongue et al.,[32] the recent review given by Kovacs et al.,[33] and a recent workshop of Wet Chemical Etching of Silicon.[34]

KOH is perhaps the most widely used bulk micromachining wet chemical etchant, although it is a strong ionic contaminant to the CMOS process. Despite these difficulties, it has been demonstrated in post- and pre-processing formalisms with stringent chemical cleaning. The etch has been charac-



(c)



(d)

FIGURE 10.1(c-d) (c) Electron micrograph of a cavity etched in (100) silicon wafer dimensional marker is 100 μm; (d) slots etched in (110) silicon wafer, dimensional marker is 10 μm.

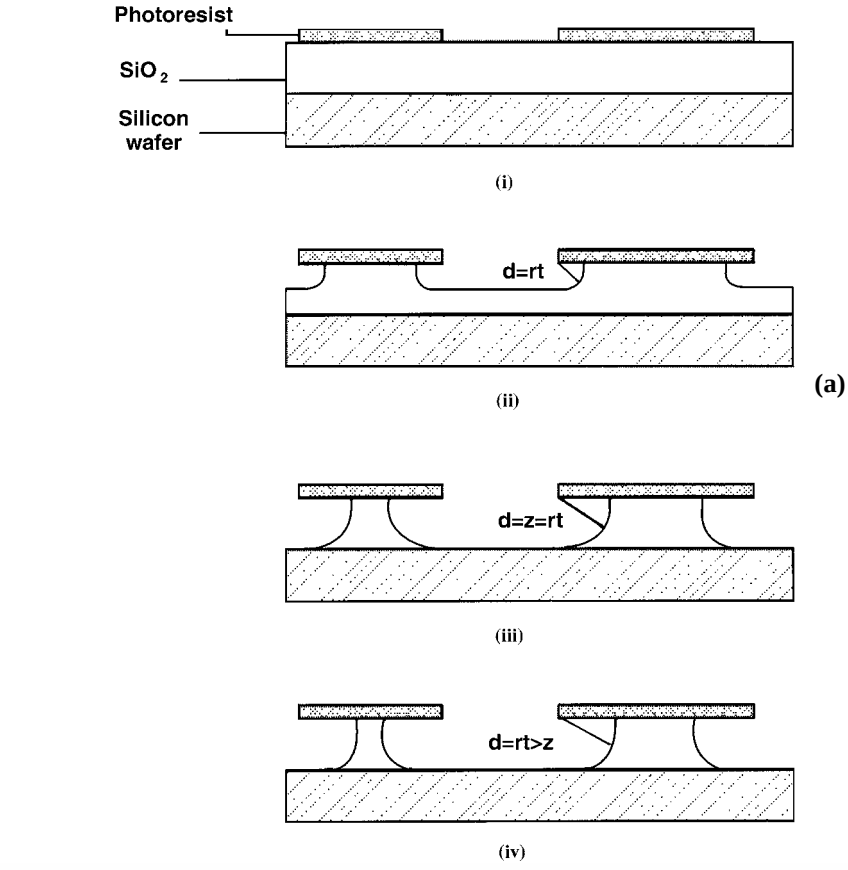
terized extensively.[35–37] Table 10.2 lists the plane selectivity of useful concentrations that in additon to the surface roughness, are a function of the solution composition. However, roughness appears to be related to hydrogen bubble release from the surface, and work by Bressers et al.[38] has demonstrated that the addition of ferricyanide ions reduces hillock formation. Note that selectivity to oxide masking layers is not as high as CsOH and TMAH. Alcohol can be added to the etch to improve the surface finish and uniformity.[39]

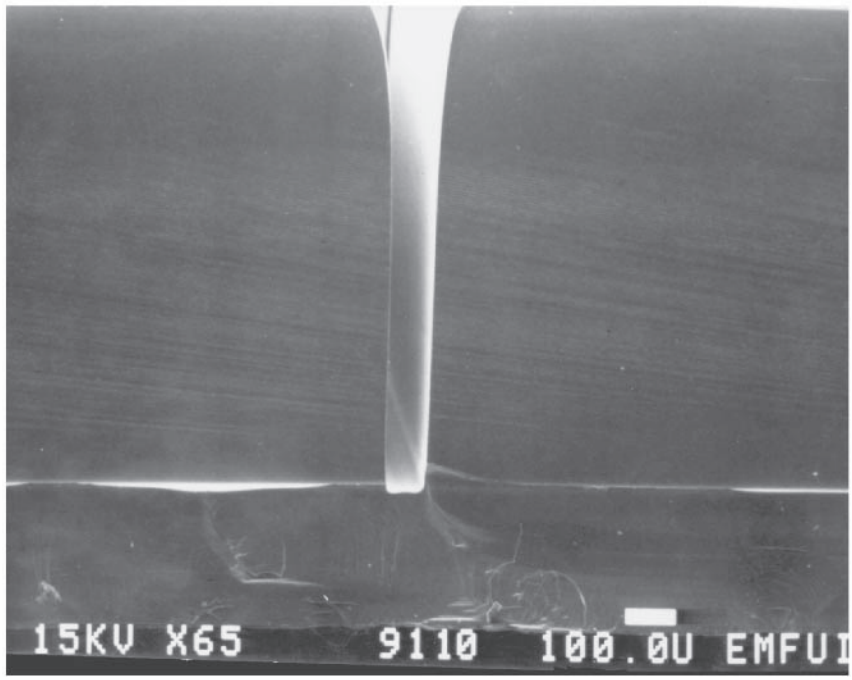
CsOH etchant has been characterized,[40–42] and the results are summarized in Table 10.2. It has high selectivity to silicon dioxide and is dopant selective, producing smooth membranes at high concentrations. Surface roughness is often a key parameter in device design and is of key importance for technologically useful etches to obtain controlled surface conditions.

TABLE 10.2 Bulk Etching Solutions for Silicon

Etching Solution	Concentration (wt%)	Temp. (°C)	Etch rate of (100) plane (μm/hr)[a,b]	Anisotropy[b] $\frac{\text{Rate(110)}}{\text{Rate(100)}}$	Anisotropy[b] $\frac{\text{Rate(100)}}{\text{Rate(111)}}$	Selectivity $\frac{\text{Rate Si(100)}}{\text{Rate } SiO_2}$ (Thermal)	Selectivity $\frac{\text{Rate Si(100)}}{\text{Rate } Si_3N_4}$ (LPCVD)	Metals	Etch stop on p^+ silicon	Com.	Ref.
					Isotropic etches						
HNA	250 ml HF/ 500 ml HNO_3/ 800ml CH_3COOH	20	4–20 [function of stirring]	—	—			—	—		50
					Anisotropic etches						
KOH	45	85	55	~1.5	200	300	40,000	Au	Yes	Not CMOS compatible, inexpensive, safe, widely used	27–34, 36–38
KOH/isopropyl alcohol	26/4	80	66	~0.6	~200	High	Very high	Au	$>10\times10^{20}/cm^3$ decreases rate by 20	Not CMOS compatible, inexpensive, safe, widely used	39
CsOH	50	70	19	0.2—2.5b	50	2000	Very high	Au	Yes	Expensive material, good selectivity with SiO_2	40–42
Ethylenediamine/ pyrocatechol/water	255cc/ 45grm/ 120cc	100	66	<1	35	3500	Very high	Au, Cu Cr, Ag, Ta, Ni	$>7\times10^{19}/cm^3$ decreases rate by 50	Carcinogenic, widely used, low anisotropy	43–44
TMAH	4 20	80	54	—	25	5000	Very high	Al [Si doping of solution]	Yes	Flammable, low anisotropy	45–47, 49
TMAH/alcohol	15–25	70–90	11–40	—	15–37	5000	>21,000		Yes		48

[a] Function of temperature.
[b] Function of solution concentration.





(b)

FIGURE 10.2 (a) Schematic diagram of isotropic etching, indicating undercut of mask and definition of isotropy; (b) effect of isotropic etching on grooves sawed into silicon wafer. The etching has rounded the tops of the grooves. (Kasapbasioglu, H. et al., *J. Electrochem. Soc.*, 140, 2319, 1993. With permission.)

EDP has been demonstrated as compatible with CMOS processing. It has high selectivity to dielectic and metallization layers and contains no ionic contaminants. Etch has been characterized by Reisman et al.[43] and Finne and Klein.[44] The popular formulations are as follows: Type-S is 1000 ml ethylenediamine, 160 g pyrocatechol, 133 ml water, and 6 g pyrazine; Type-F is 1000 ml ethylenediamine, 320 g pyrocatechol, 320 ml water, and 6 g pyrazine. These solutions etch thermal oxide at about 55 Å/h at 115°C, there is no detectable attack of LPCVD silicon nitride, and the following metals — Au, Cr, Ag, and Cu — are resistant to EDP. The plane selectivity is listed in Table 10.2.

TMAH has been studied extensively due to its low ionic contamination with CMOS. The early work on etch characterization was carried out by Tabata.[45] Characteristics are listed in Table 10.2 and the plane selectivity is markedly lower than KOH; however, selectivity to SiO_2 is quite high. The solution also passivates aluminum surfaces when the pH is adjusted into a suitable range.[46] This makes it a strong candidate for CMOS post-processing; however, the etch rate is reduced with silicon doping and there are unresolved issues regarding the solution stability. Ammonium persulphonate was added to the TMAH bath to reduce surface roughness by limiting the formation of hydrogen bubbles, specifically for a 5 wt % solution, 40 g/l silicic acid, and 5 to 10 g/l ammonium persulphate.[47] Changes in the surface morphology and high index plane selectivity are observed with the addition of alcohol[48] and hillock formation has been studied.[49]

Isotropic etching in a mixtures of hydrofluoric acid (HF) and nitric acid (HNO_3) produces typically rounded profiles.[50] Figure 10.2 shows a schematic diagram of the profile produced during isotropic etching. The etching takes place in a two-step process, the first being oxidation of the silicon by the HNO_3, and the second being dissolution of the oxidized layer into a soluble H_2SiF_6 silicate. These two processes have different reaction rates and, hence, polishing occurs at low concentrations of HF where it defines the rate-limiting step in the reaction. The etchant can be stabilized by the addition of acetic acid, which helps prevent the dissociation of the nitric acid into NO_3^- and NO_2^-. The etch is dopant selective, having a lower etch rate for lightly doped region ($<10^{17}/cm^3$) of silicon relative to heavier doping regions.[21] Figure 10.2(b) shows the application of this etching mixture to the rounding of pins that have been produced by mechanical sawing of the silicon.

Etch Stop Methods

There are various methods that have been developed for membrane fabrication at a defined doped layer as shown in Fig. 10.3(a). The heavily doped etch stop occurs when the doping level of p-type silicon is greater than $2–5 \times 10^{19}/cm^3$ for all of the principle etching solutions listed in Table 10.2. It is believed that the etch mechanism is related to the availability of electrons at the surface, which is a participant in the silicon dissolution process. Thus, high p^+ doping results in electron-hole recombination. Doping and etch back has been very successful in producing capacitive pressure sensors and neural recording probes.[52] The etch rate of the heavily doped material has an n^4 dependence on the concentration of the dopant; however, there are some differences in the abruptness of this dependence as indicated in Fig. 10.3(b). This is in agreement with the four electrons per silicon atom dissolution process. The mechanism for dopant-selective behavior is discussed by Collins.[53]

Other methods are listed in Table 10.3. The electrochemical etch stop relies on the formation of an anodic oxide on the silicon, which stops the etching process. The process was first demonstrated by Waggener and has been developed for silicon microstructures by Jackson et al.[54] and Kloeck et al.[55] Figure 10.4 shows a typical experimental set-up in which an n/p junction is reverse-biased in the etching tank and protected in a Teflon holder so that only the p-type surface is exposed to the solution. Because the junction is reverse-biased, the potential on the silicon is close to the open-circuit potential and so etching continues uninhibited. However, once the n-type material is reached, the potential becomes that applied by the external circuit. Care must be taken to ensure that the reverse-bias leakage current is not high enough to allow the potential on the exposed silicon to reaching a point more positive than the passivation potential, as indicated in the figure by the operating potential for the silicon surface. The passivation potential is the potential at which oxide forms rather than dissolution

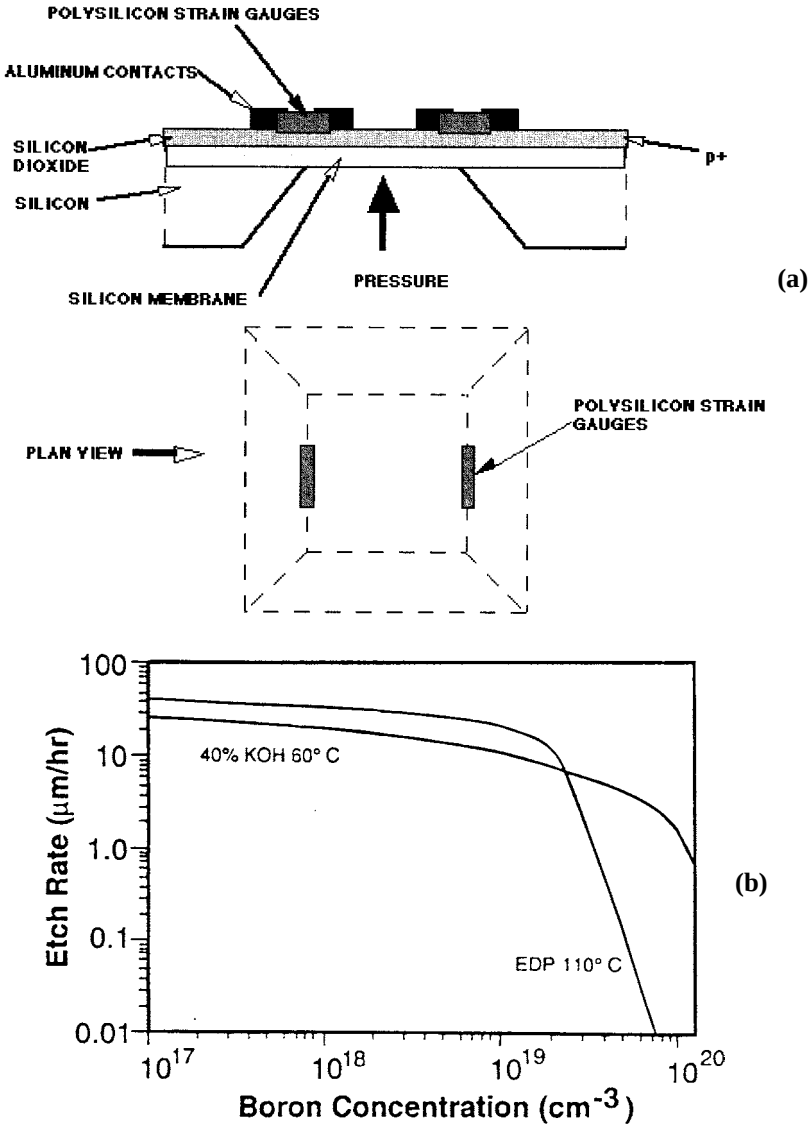


FIGURE 10.3 (a) Schematic diagram of a heavily doped layer diaphragm formation; (b) etch stop of heavily doped silicon for KOH and EDP anisotropic etching solutions. (Collins, S. D., *J. Electrochem. Soc.*, 144, 2242, 1997. With permission.)

TABLE 10.3 Silicon Membrane Fabrication Methods

Method	Thickness Range	Process	Comments	Ref.
Timed etch	5–100s μm	Observation of diaphram thickness as a function of time	Inaccurate	53
p^+ heavy doping	0.1–5 μm	Boron doping of ion implant to define location of membrane	Membrane is highly compressive state	52,53
Electrochemical etch stop	0.1–100 μm	p/n junction location of membrane	Electrical contact is necessary to wafer so special fixturing is required	53–55
Buried oxide layer	0.1–1 μm	SIMOX implant of O_2 dose above $10^{18}/cm^3$	Expensive; annealing damage from surface region after implant critical	53
Si-Si bonding			Alignment of two wafers is required	117–119

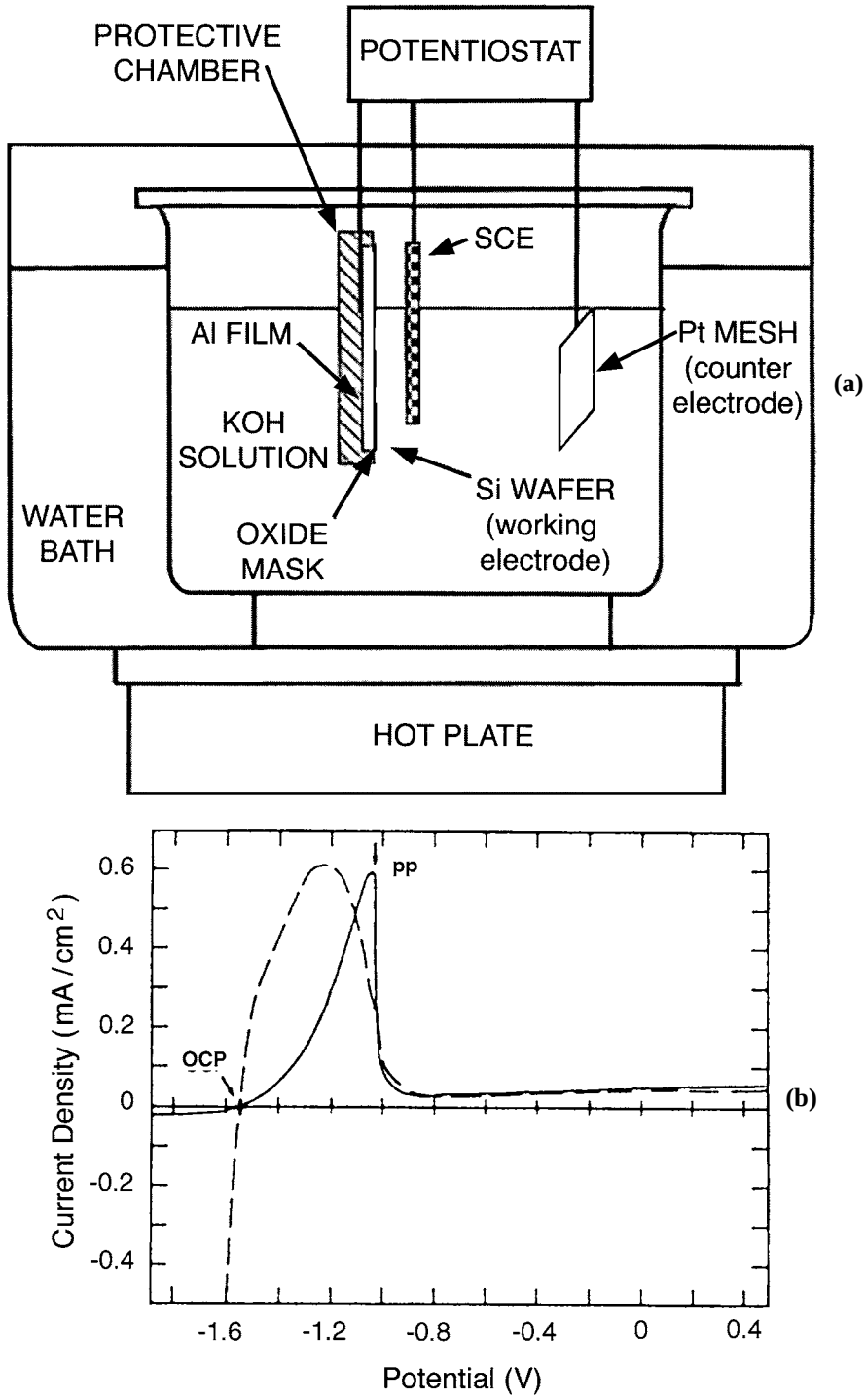


FIGURE 10.4 (a) Typical experimental set-up for electrochemical etch stop method for formation of thin silicon diaphragms (SCE is the Standard Calomel Electrode); (b) current–voltage characteristics of the p-type (solid line) and n-type (dashed line) silicon in 40 wt. % KOH solution at 60°C, where OCP is the open circuit potential and PP passivation potential. (Kloeck, B., et al., *IEEE Trans. Electron. Dev.*, 36, 663, 1989. With permission.)

takes place. The current is used as a diagnostic to indicate when the layer is removed in practical etching procedures because a current peak occurs when the oxidation reaction occurs due to the additional electrons required by the reaction. Once a passivating layer of oxide has formed, the current drops to a low level of leakage current through the oxide layer. The oxide dissolution process is slow in these solutions, and hence, if the bias is removed, after a finite time the oxide layer is removed and etching of the silicon continues.

Electrochemical Machining and Porous Silicon

Tuller and Mlcak[56] have demonstrated that anodic dissolution of silicon offers a controllable method for the fabrication of cantilever beams, membranes, and other structures. Dopant-selective etch stop can be achieved because p-type Si etches in HF solutions under anodic bias, whereas the n-type is stable. However, under strong illumination, porous silicon is formed on the n-type material. Hence, the etch rate and surface morphology are a function of the doping level, current density, and illumination level. The anodic current is a direct measure of the dissolution rate and, by defining an optically opaque insoluble mask on the silicon surface, high aspect ratio structures have been defined in the surface. Porous silicon is under investigation as a MEMS material as discussed by Schöning et al.[57] for its extremely high surface area.

Xenon Difluoride Etching

This dry gas-phase isotropic etch process relies on sublimation of solid XeF_2 at room temperature into a vapor that selectively attacks Si over SiO_2, Si_3N_4, and several metals.[58] Hoffman et al.[59] demonstrated its use for sensors and for CMOS-compatible processing. Chu et al.[60] later confirmed the minimal etch rate of Al, Cr, TiN, and SiC. Alternatively, gas-phase isotropic etching in xenon difluoride, or reactive ion etching methods may be selected. Tea et al.[61] has studied several of these methods and concluded that EDP can attach the Al bonding pads, whereas XeF_2 shows excellent compatibility. Microheaters for chemical sensors and microwave transmission lines were successfully fabricated; however, there was a lack of compatibility with catalytic metals such as Pt, Ir, and Au.

RIE Etching

Dry etching offers the advantages of controlled dimensions independent of the crystal planes in the substrate, in addition to higher dimensional control to sub-micron over wet chemical methods. The process utilized either a dc, RF, microwave, or inductively coupled energy to excite a plasma of reactive ions and accelerate them to the substrate. Depending on the ion acceleration potential utilized and the gas pressure, there are very different process characteristics. Figure 10.5 shows the different processes of (a) sputtering by physical bombardment of the surface, (b) chemical etching which is isotropic, and (c) reactive ion etching in which the ion bombardment enhances the chemical etching rate. This ion-assisted mechanism provides anisotropy, which is useful technologically for the fabrication of high aspect ratio structures. Ion milling represents the processes at high potentials where sputtering is dominant and at the other extreme at low substrate bias plasma etching where chemistry plays the dominant role. The reader is referred to excellent texts on dry etching equipment and processing methods.[62,63] Key factors in the design of dry etching processes for films are the selectivity with respect to the masking layer and with respect to the substrate material. The etch rate, uniformity, and anisotropy are other key process parameters.

Deep-RIE Etching

Deep-RIE etching was developed by the application of a novel etch chemistry and high-density plasmas created by either microwave or inductively coupled sources. Mixed gas chemistries can be utilized to provide sidewall passivation and further increase the anisotropy of the process. Selectivities with respect to a photoresist mask of 200:1 can be reached, allowing one to etch completely through a silicon wafer. Bhardwaj et al.[64] have demonstrated deep reactive ion etching by alternating between deposition of a

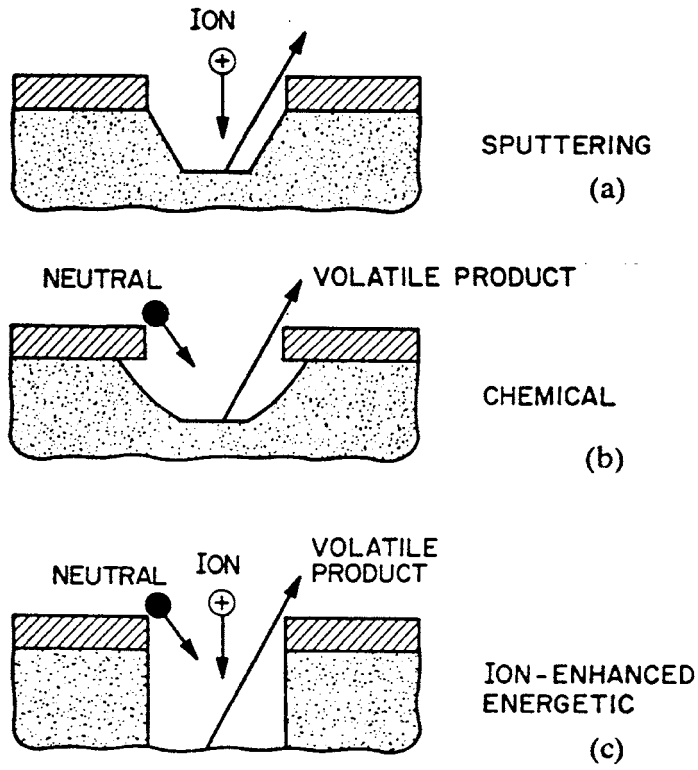


FIGURE 10.5 (a) Sputtering, (b) plasma etching, and (c) reactive ion etching.

passivating film and etch chemistry. The surface is coated with a Teflon-like material and the material remains on the sidewalls; however, ion bombardment removes it from the horizontal surfaces. Typically, a low bias voltage is utilized so that the mask erosion rate is minimized. An example of grooves etched with this process is given in Fig. 10.6.

Corner Compensation

Corner compensation of the bulk etched structure is important to maintain the shape of a micromachined structure. For example, microchambers for DNA sequencing by hybridization have been fabricated with corner compensation structures oriented in the [110] direction.[65] A great deal of work has examined the compensation of exposed corners for etching in KOH solutions under different conditions.[66] The shape of the compensation structure is shown in Fig. 10.7. There are several commercially available CAD tools for silicon bulk etching. Figure 10.7(c) is an example of a etched cantilever in CsOH solution. The CAD simulation shows the faceting high-index planes and the general features of the structure.

10.4 Surface Micromachining

Surface micromachining involves depositing thin layers of polysilicon as a structural layer and phosphosilicate glass as a sacrificial layer (see Fig. 10.8), as first demonstrated by Howe and Muller.[67] These layers are typically less than 2 μm in thickness, and etching away the PSG layer in a HF-based etching process. Undercut etch rates have been characterized, and the maximum dimensions of the thin polysilicon beams are limited due to internal stresses and the mechanical strength of the polysilicon.[68] Fine-grained polysilicon deposited at 580°C is subsequently annealed for surface micromachining to

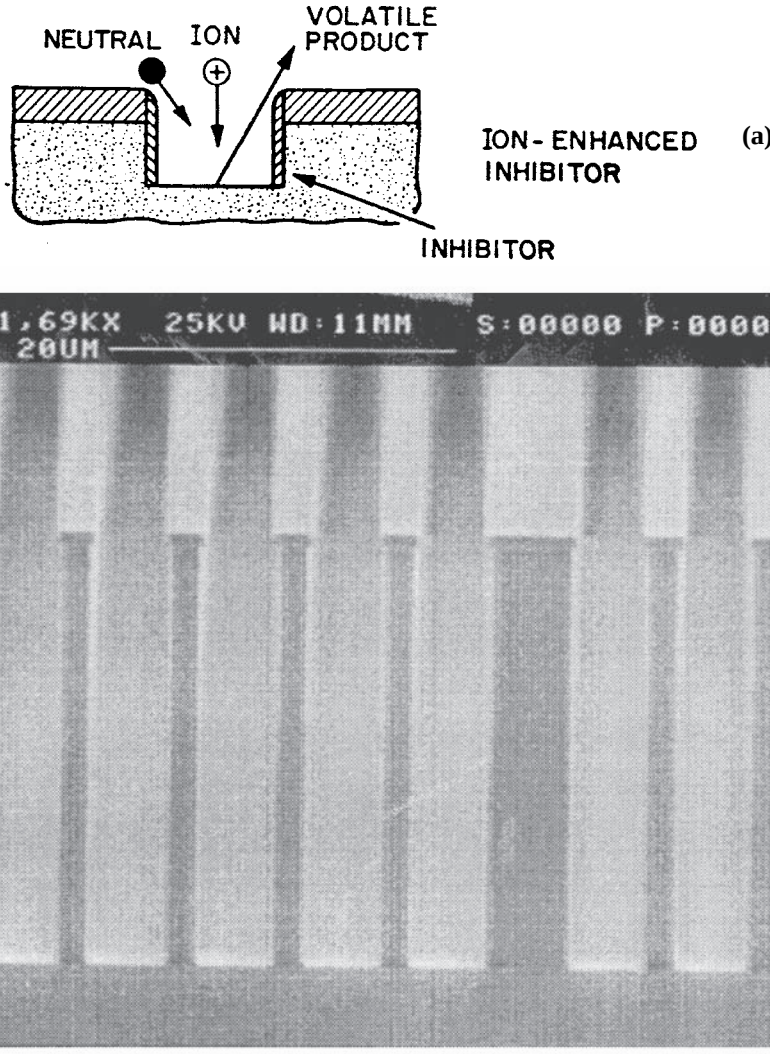
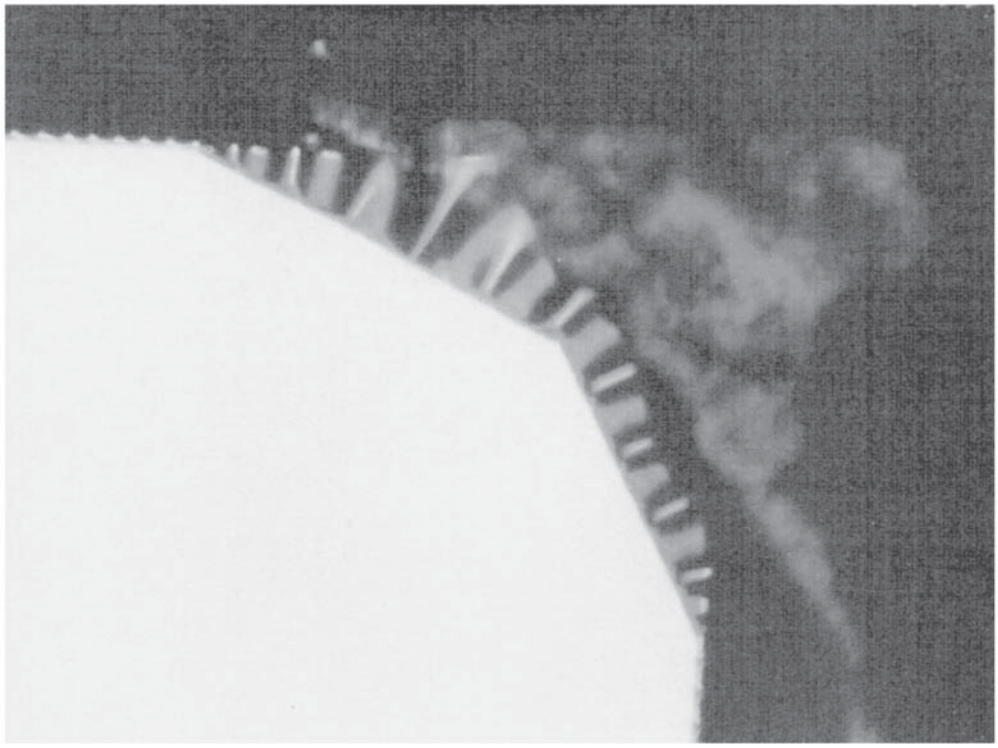
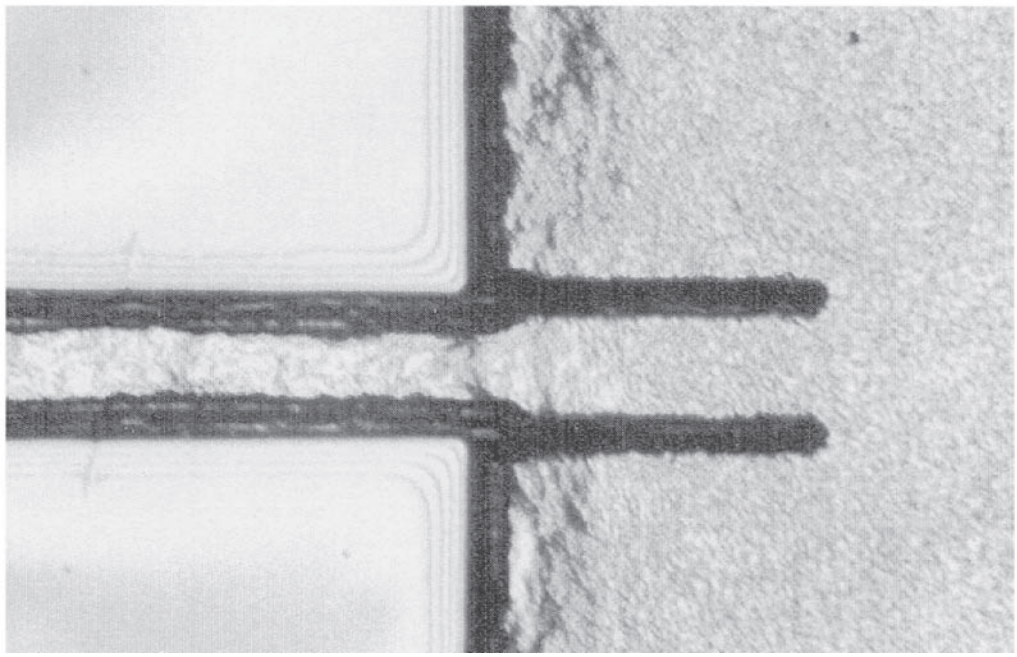


FIGURE 10.6 (a) Method of side-wall passivation during RIE etching; (b) slot produced by deep RIE etching process. (Bhardwaj, J. et al. in *Microstructures and Microfabricated Systems-III, Proceedings of the Electrochemical Society*, 97-5, 118, 1997. With permission.)

produce low stress polysilicon. Guckel et al.[69] have demonstrated that post-deposition anneal conditions are key to defining the low stress characteristics of polysilicon, as shown in Fig. 10.9. Conductive regions are defined by ion implantation. An alternative process is to carry out *in situ* doping with phosphourus and a relatively rapid deposition rate, followed by an RTA at 950°C to produce a low tensile stress film with a small stress gradient through the film.[70] Polysilicon is an extremely stable and good-quality material for producing micromechanical elements, as demonstrated by the success of these devices (examples are given in the next section). Sealing of surface micromachined structures has been developed for absolute pressure sensors by reactive sealing of small openings in the surface or by dry deposition of another layer of material like silicon nitride.[71] Surface micromachining is not limited to poly-Si and PSG. A wide range of material combinations have been studied.[15] Machining with aluminum as the structural layer and a photoresist as the sacrificial layer is discussed by Westerberg et al.[72]

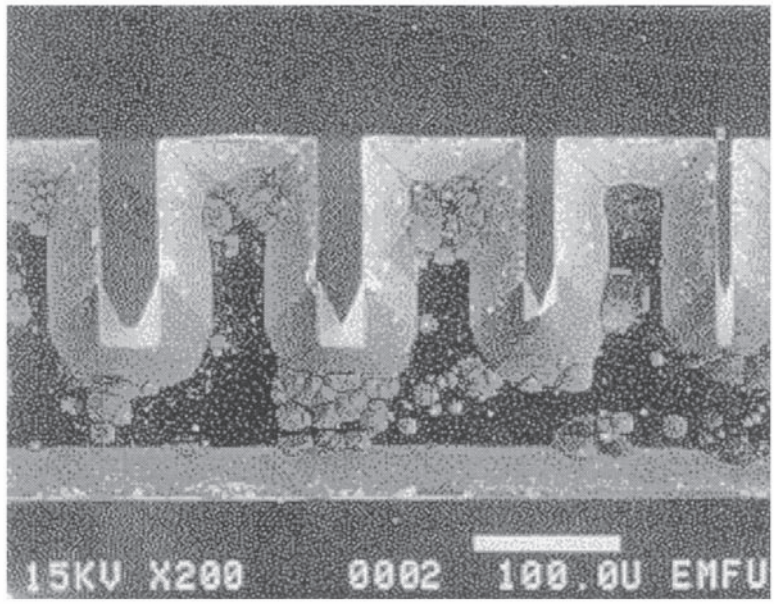


(a)

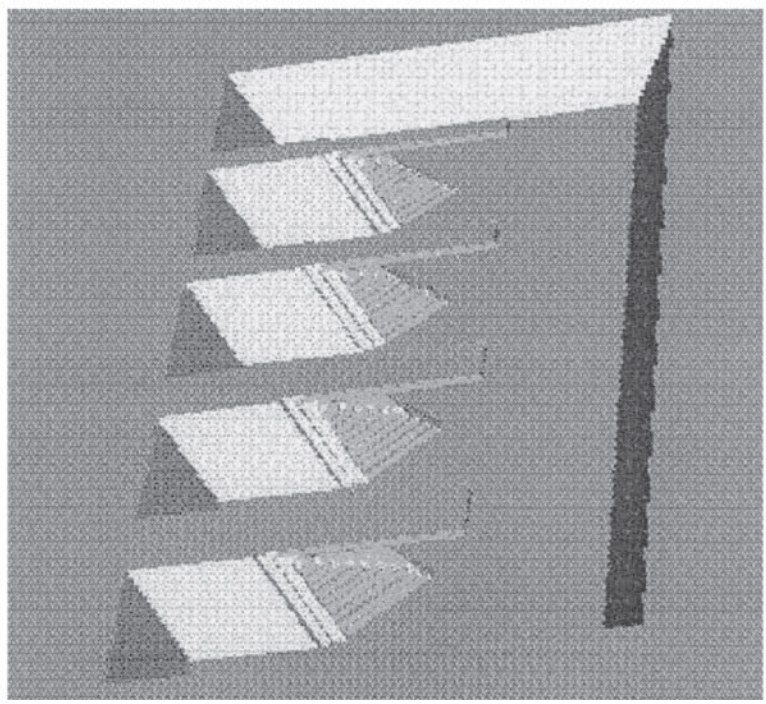


(b)

FIGURE 10.7(a-b) (a) Exposed corner etched in 50 wt% CsOH for 4 hours; (b) corner compensation structure 20-μm wide after etching for 3 hours in 49 wt% KOH.



(c)



(d)

FIGURE 10.7(c-d) (c) Etched cantilever beams in silicon dioxide by etching in CsOH solution; (d) example of CAD simulation of etched cantilever structures in silicon. (Shih, B. et al., submitted to *J. Electrochem Soc.*, 1998. With permission.)

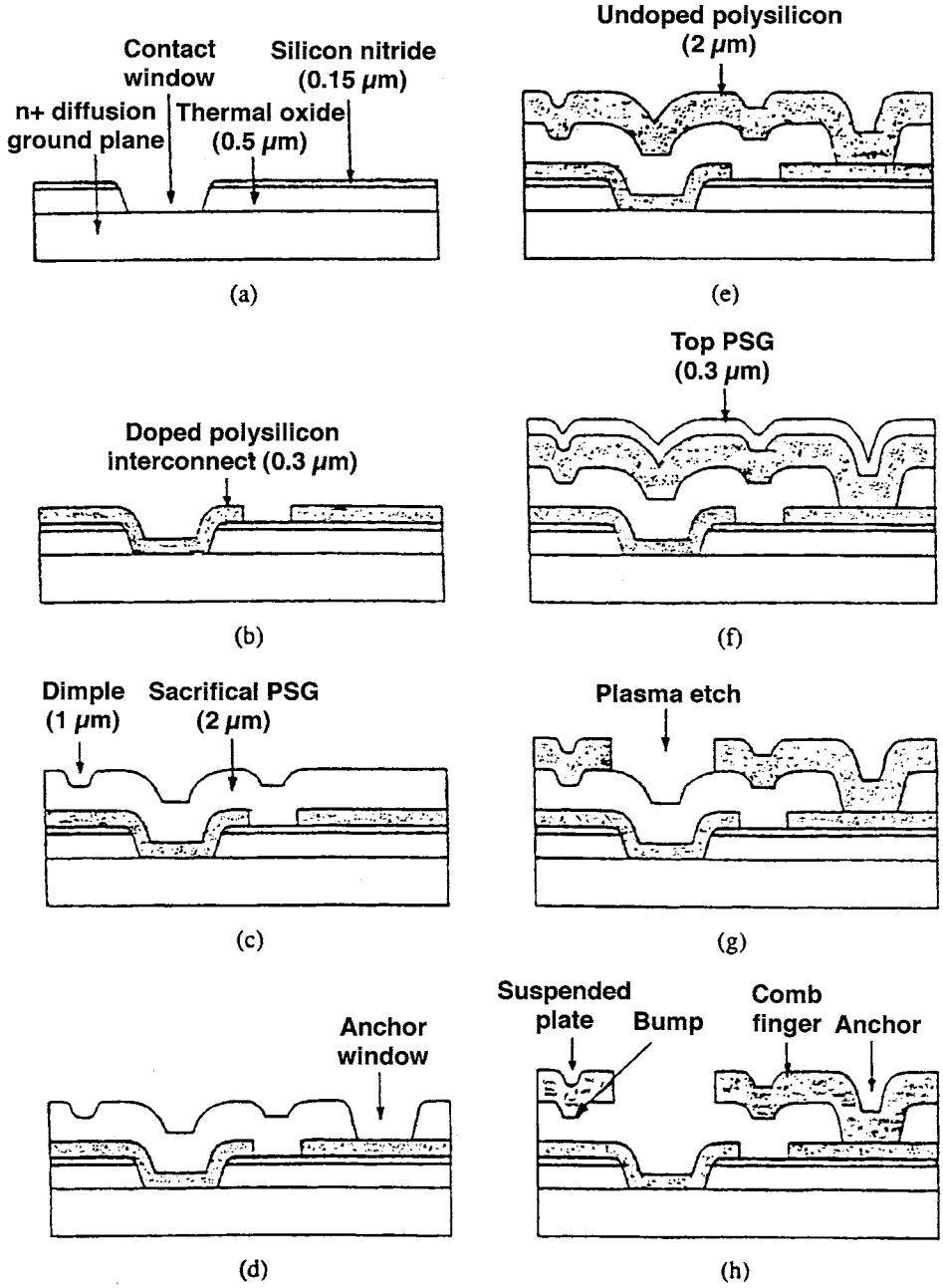


FIGURE 10.8 Cross-sectional diagram of processing steps in the surface micromachining process. (Courtesy of Tong, W. C. et al. *Sensors and Actuators*, 20, 25, 1989. With permission.)

Stiction

Stiction in surface micromachined structures is an issue that must be addressed due to the very small gaps present between surfaces. Tas et al.[73] investigated the origins of stiction in surface micromachined devices

and found that roughness plays an important role. The critical dimensions for the mechanical force to snap back a cantilever beam against the stiction force was investigated. The energy of adhesion by liquid bridging found that if the tip touches the substrate, the surface energy plus the deformation energy has a minimum for a detachment length smaller than the beam length. Stiction can occur during processing or during device operation. Various methods have been developed to avoid stiction during processing. Carbon dioxide drying is used because, below the critical temperature of the supercritical carbon dioxide, no liquid vapor interface exists that could cause contact. During device operation, the fail-safe requires that these surfaces do not make contact or an adequate release force can be applied. Surface energy is only partially successful in modifying these interactions and it is important to limit the contact area. Specifically, dimples, bumps, sidewall spacers, or increased surface roughness can define smaller contact areas.

Materials Properties of Thin Films

The properties of thin films differ from bulk material, and measurement techniques have been developed for *in situ* determination of these properties on a silicon wafer. Properties of interest are Young's modulus, Poisson ratio, residual stress, tensile strength, fracture toughness, thermal conductivity, density, and optical aborbance and optical reflectivity. Finding mechanical properties of thin films as a function of deposition conditions has been cataloged for many materials in a properties database available from Intellisense Inc.[74] Table 10.4, summarizes properties of thin-film CMOS and MEMS materials. However, this information should be used with caution given that there is considerable variation in materials properties as a function of the specific growth conditions due to variations in the microstructure and the film thickness. Low-stress polysilicon has been deposited in an epitaxy reactor in a process compatible with bipolar electronics by Gennissen et al.[75] In addition, more recent work by Wenk[76] has demonstrated thick low-stress polysilicon films grown in an epitaxial growth reactor from DCS at 1000°C for a microfabricated accelerometer and inertial gyroscope. Figure 10.9(b) shows a typical biaxial stress plot for the 3-μm polysilicon tested from the MUMPS process by Sharpe et al.[77] Other studies include those of Kahn et al.,[78] Maier-Schneider et al.,[79] and Biebl et al.[80]

Silicon Nitride

Stoichiometric silicon nitride has high tensile stress and this limits the maximum film thickness that can be fabricated on a wafer.[81] Stress relaxation occurs in silicon-rich nitride as a function of the film stoichiometry and has been characterized by Gardemoers et al.,[82] Chu et al.,[83] French et al.,[84] and Habermehl.[85] Figure 10.10(a) shows the intrinsic stress as a function of deposition conditions for silicon-rich nitride growth at several different temperatures and pressures and Fig. 10.10(b) for PECVD material.[86] Other materials have been studied, including SiC[87] and diamond like carbon.[88]

10.5 Advanced Processing

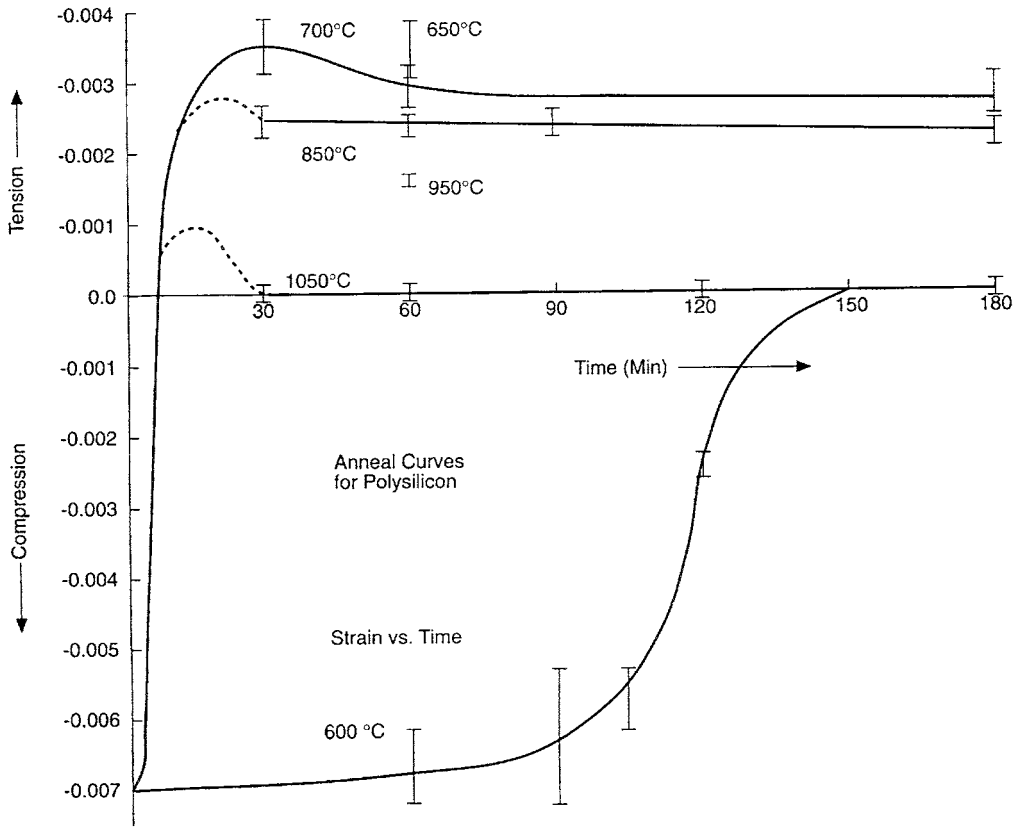
We have discussed both surface and bulk machining processes and these offer certain advantages and limitations, as contrasted by French and Sarro.[89] Due to the space limitation, there is only an opportunity to mention a few examples of mixed-technology processes that have combined aspects of surface and bulk machining with other processes like chemical mechanical polishing for planarization.

Chemical Mechanical Polishing

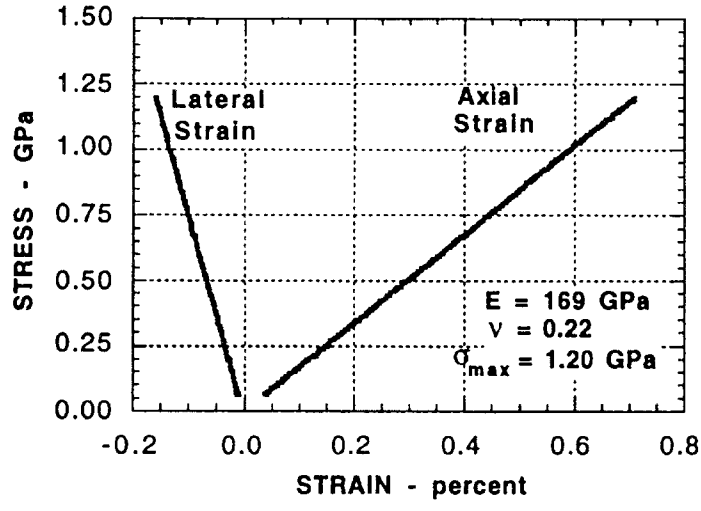
Chemical mechanical polishing is well-known in the lapping and polishing of wafers prior to fabrication. It has been recently demonstrated as a key process for achieving multilayers of polysilicon and metallization by Rogers and Sniegowski.[90] They have achieved up to five levels of polysilicon fabrication for micromechanical actuators, linkages, gears, and mechanisms. Examples of interconnected gears, locking pins, and movable elements are shown in Fig. 10.11. The key to this process is planarization of the surface between polysilicon layer deposition processes by chemical mechanical polishing.[91] In this process, planarization is

TABLE 10.4 Materials Properties of LPCVD Deposited MEMS Materials

Material	Growth Conditions	Film Thickness	Property	Value	Comments	Ref.
			Polysilicon			
MUMPS process		3 μm	Young's modulus Tensile strength	169±6.15 GPa 1.20±0.15 GPa	—	77
Thick polysilicon	1100°C, SiH_4/B_2H_6 or 610°C, Sitty	2.5-10 μm	Young's modulus Fracture toughness	150±3- GPa 2.3±0.1 MPa √m 280 MPa	undoped film	78
Thin polysilicon	565°C, SiH_4, 620°C, SiH_4, 100 mTorr	1 μm	As-deposited residual stress			
CMOS		0.33 μm	Young's modulus Tensile strength	168±7 GPa 2.11±0.10 GPa	—	80
			Young's modulus Intrinsic stress Intrinsicties	162.8±6 GPa –350±12 GPa 162.8±6 GPa	— As deposited After 1000°C anneal	79
			Silicon Nitride			
Standard process	800°C, $SiCl_2H_2lNH_3$	—	Intrinsic stress	~1.2 GPa		81
Si-rich, variable stoichiometry	800, 850°C 200, 410 mTorr $SiCl_2H_2$/ NH_3	~0.1 μm	Intrinsic stress	(See Figure 10.10)	—	82–84
Silicon-rich, variable stoichiometry	850°C 200 mTorr $SiCl_2H_2/NH_3$	0.25–0.45 μm	Young's modulus			85
PECVD				(190 GPa)		



(a)



(b)

FIGURE 10.9 (a) Stress in fine-grained polysilicon as a function of post-deposition annealing temperature and time (Guckel, H., et al., *Sensors and Actuators A*, 21, 346, 1990. With permission); (b) biaxial stress for polysilicon fabricated in the MUMPS process. (Sharpe, W. N., et al., in *Proceedings of the Tenth Annual Internations Workshop on Micro Electro Mechanical Systems, Nagoya*, Japan, January, 1997, IEEE, New Jersey, Catalog Number 97CH36021, 424. With permission.)

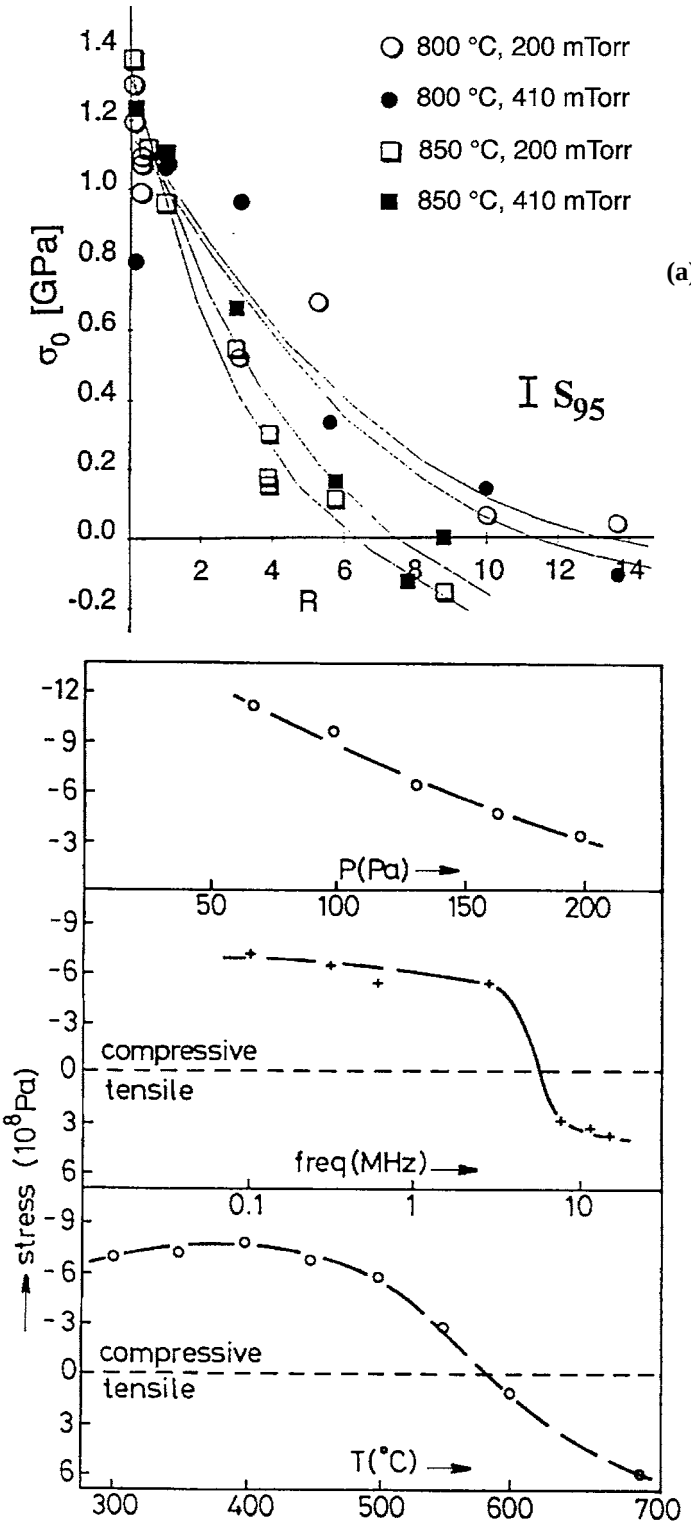


FIGURE 10.10 (a) The effect of deposition parameters on the residual stress of LPCVD deposited silicon rich silicon nitride where R is the ratio of dichlorosilone to amnoia gas flow (Gardeniers, J. G. E. et al., *J. Vac. Sci. Tech. A.*, 14, 2879, 1996. With permission.); (b) intrinsic stress of PECVD Si_3N_4 as a function of processing conditions. (Classen, W. A. P. et al., *J. Electrochem. Soc.*, 132, 893, 1985. With permission.)

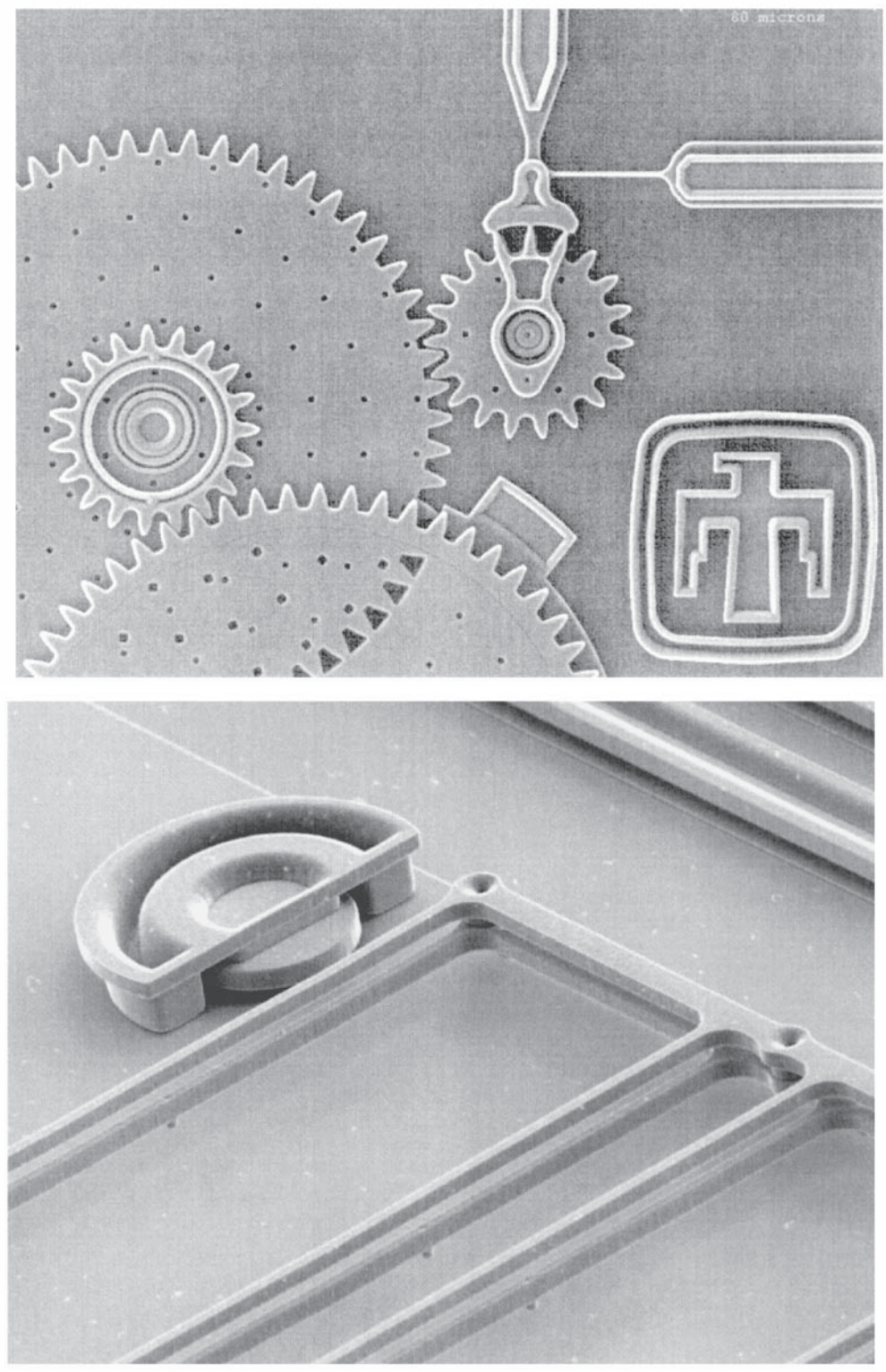
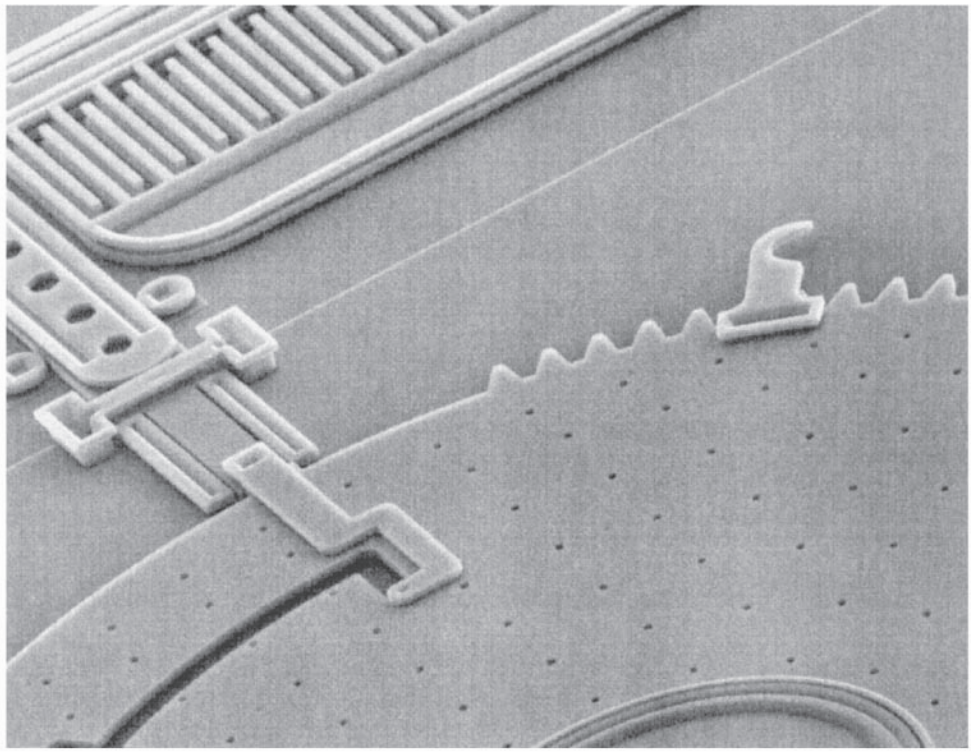
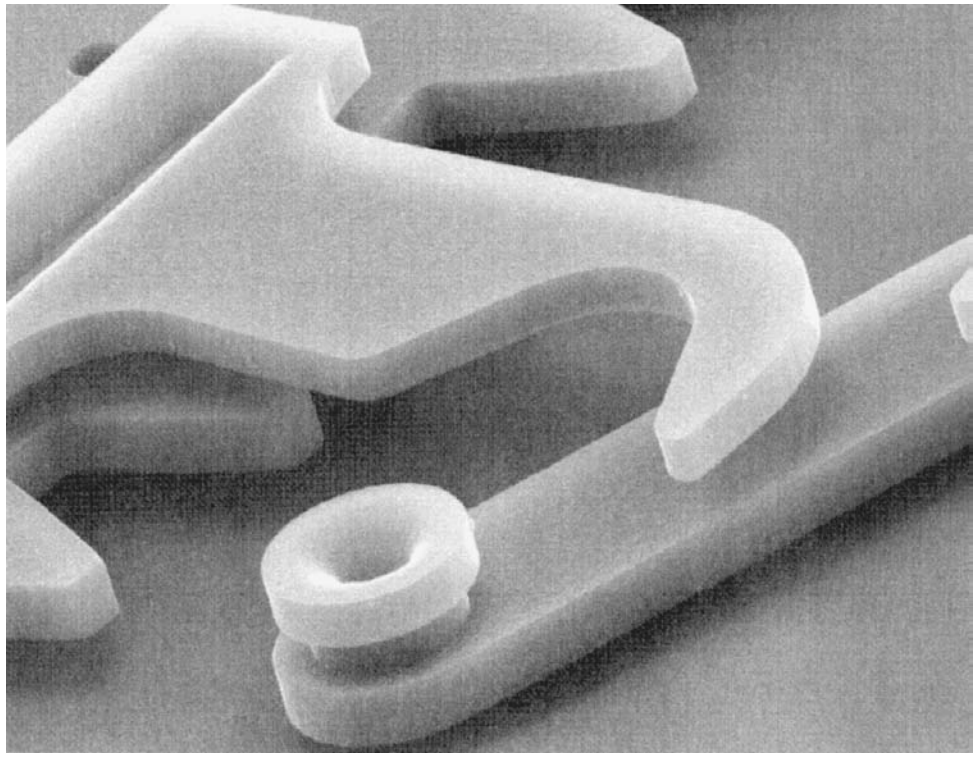


FIGURE 10.11 (a-b) SEM micrographs of decices fabricated by the SUMMIT process: (a) gears; (b) linear spring.



(c)



(d)

FIGURE 10.11 (c-d) (c) Mechanism; (d) hook. (Courtesy of Sandia National Laboratories' Intelligent Micromachine Initiative; www.mdl.sandia.gov/Micromachine. With permission.)

achieved by depositing a thick layer of silicon dioxide over the polysilicon layer and polishing back to achieve planarization.[92] The process flow is shown schematically in Fig. 10.12. The key advantages of this process are: (1) the removal of stringers without the necessity for an over-etch; (2) improvement in line width control given that topographic features are removed; (3) removal of mechanical protrusions between layers that can cause interference; and (4) the possibility to carry out integration by placing the mechanical elements in a etched well in the silicon substrate (as discussed in the next section). Finally, these processing improvements lead to higher process yield.

SCREAM processing utilizes the advantages of the properties of single-crystal silicon for producing a wide range of elegant devices.[93] This method was derived from a process to make isolated islands of submicron silicon by local oxidation. Suspended high aspect ratio structure are fabricated by SF_6 RIE step, followed by sidewall oxidation and isotropic silicon etch to undercut the structure (see Fig. 10.13). Both suspended structures and structures with metallized sides for electrostatic drives can be realized by varying the width of the structure and processes, each with high out-of-plane stiffness, as defined by the ratio of beam height to length, $[h^3/l^3]$. Examples of devices realized by this elegant process include tip-on-tip sensors, microaccelerometers, electrostatic drives, torsional actuators, microloading machines, and STM tips with integrated drive. In addition, beams filled with spin on glass isolation have been demonstrated with high thermal isolation. The planarity of large aspect ratio structures has been investigated by Saif and MacDonald.[94]

HEXSIL is an RIE-etched channel and fill process that utilizes polishing for planarization.[95] The trench with controls the material that fills the grooves defined by RIE process. After the groove is coated with an oxide layer, the body of the device is made from undoped polysilicon, followed by a doped layer, and finally, Ni is electrolytically deposited into grooves to define highly conducting regions (see Fig. 10.14 for a process overview). The different groove width provide discrimination between different materials; hence, three types of material combinations are defined. Once the sacrificial oxide is removed, the HEXIL structure is released, micro-tweezers, and other structures have been defined by this method, as shown in Fig. 10.14(b).

Electroplating into Molds

A variety of materials have been utilized for a mold into which metals can be electroplated. These include the LIGA process, thick photoresists, polyimide,[96] and SU-8.[97] Photoresists often have a limited sidewall profile; however, polyimide structures with vertical wall profiles can be processed up to 100 μm in thickness.[98] The process flow is shown in Fig. 10.15 that utilizes a standard optical mask and standard UV exposure system. A spiral inductor structure produced by Ahn et al.[99] is shown in Fig. 10.15(b). A 30-μm thick nickel-iron permalloy magnetic core is wrapped with 40-μm thick multilevel copper conductor lines. [The magnetic circuit consists of an inductor $4 \times 1 \times 0.13$ mm^3 having 33 turns, with an inductance of 0.4 μH at 1 kHz to 1 MHz.] Applications of inductive components are for microsensors, microactuators, and magentic power devices such as dc-to-dc converters. Layers of copper conductor and high-permeability NiFe are electroplated to form a high-value inductance with parallel and spiral coils.[100] The process SU-8 is a negative-acting epoxy resin that has been fabricated up to 100 μm thick layers in a standard alignment system (see example in Fig. 10.16). Multiple levels of metallization have been developed of ULSI copper plating by the metal is formed by the Damascene process with a photoresist mold. Planarization is achieved by chemical mechanical polishing, which relies on the materials (metals and dielectics) having similar abrasion rates.

LIGA Process

LIGA is an acronym from the German Lithographie, Galvanoformung, und Abformung, denoting the use of X-ray lithography from a synchrotron source, in thick PMMA layers to define structures with very high aspect ratios, followed by nickel electroplating, and subsequent replication by injection molding.[101] Because LIGA is based on X-ray radiation, it is not compatible with CMOS processing. The intensity of

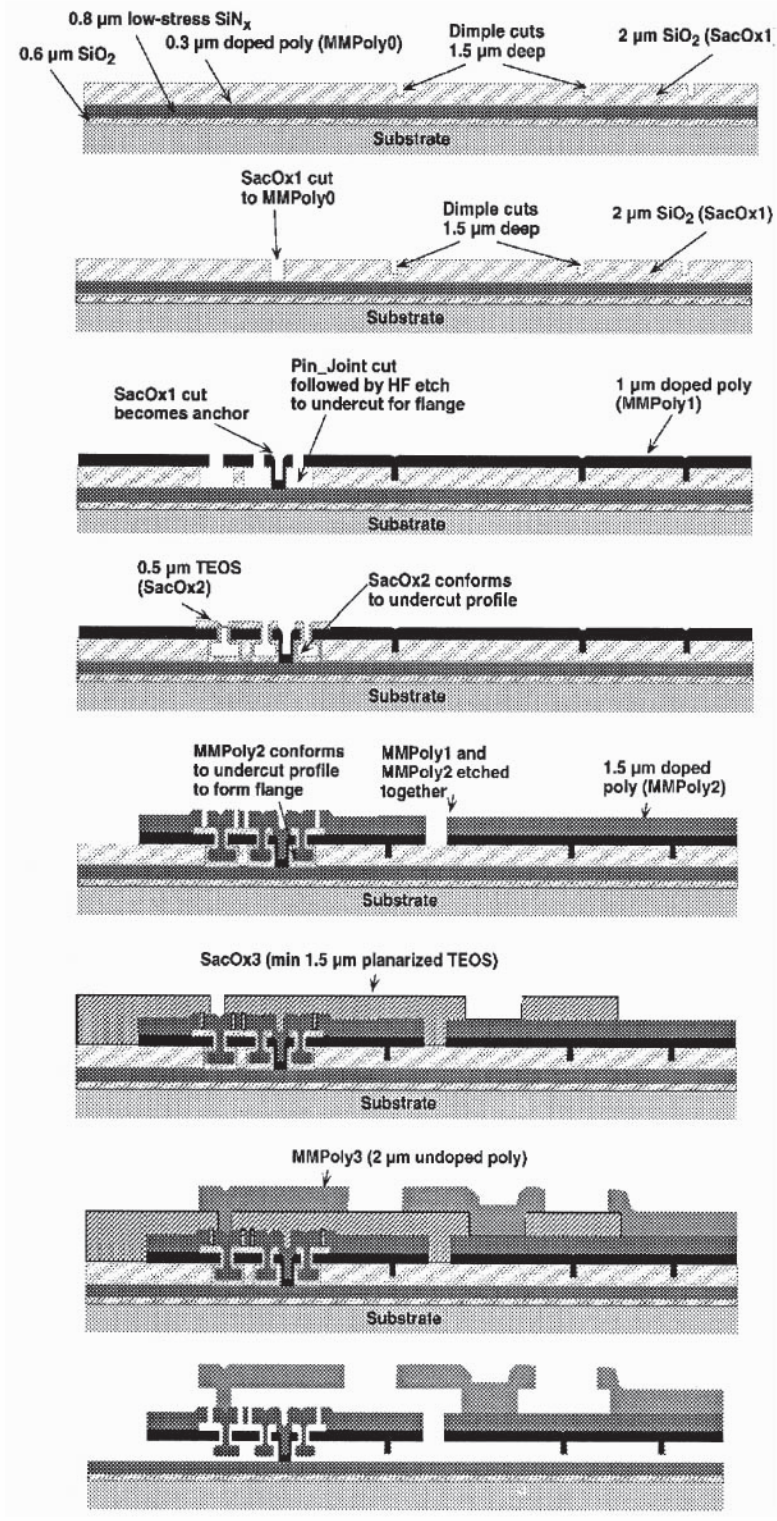


FIGURE 10.12 Schematic diagram of processing steps in the SUMMIT process for polysilicon MEMS. (Courtesy of Sandia National Laboratories' Intelligent Micromachine Initiative; www.mdl.sandia.gov/Micromachine. With permission.)

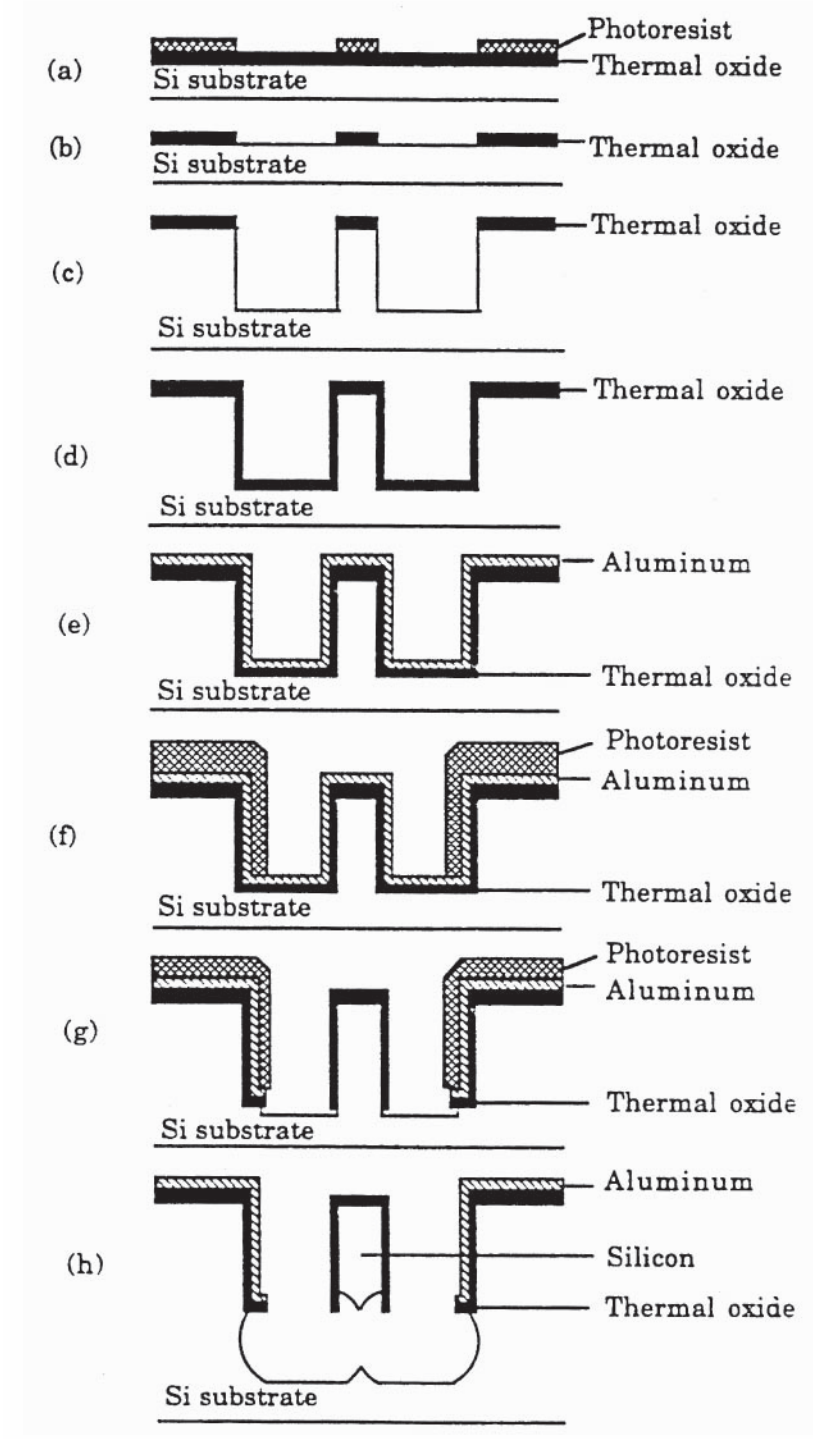
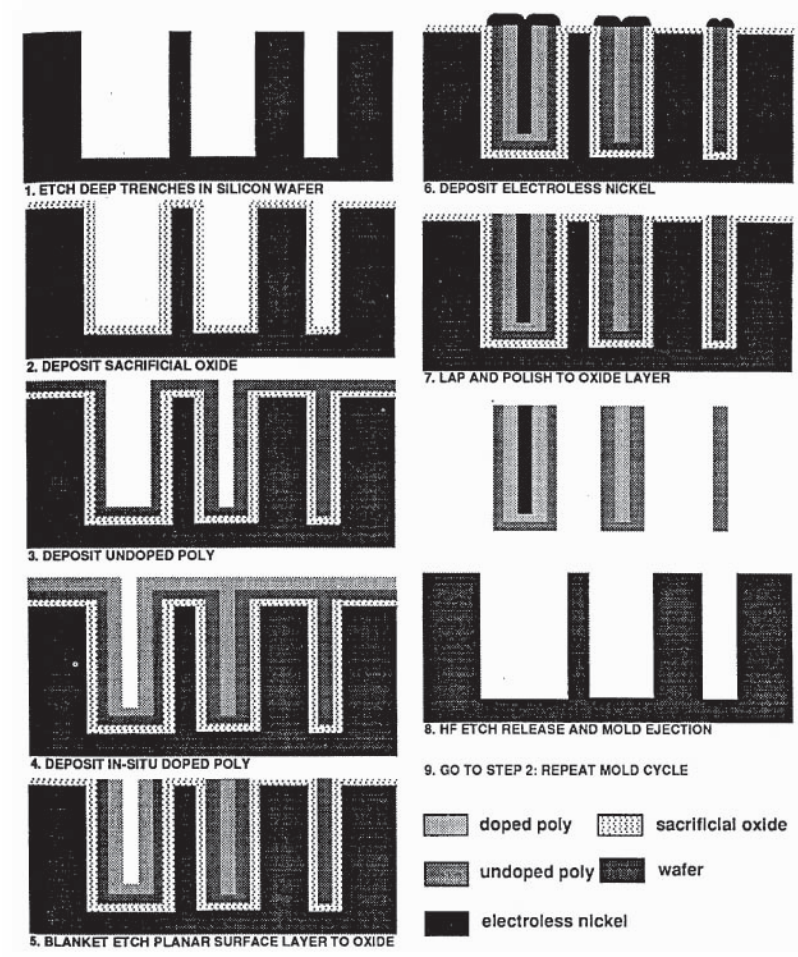
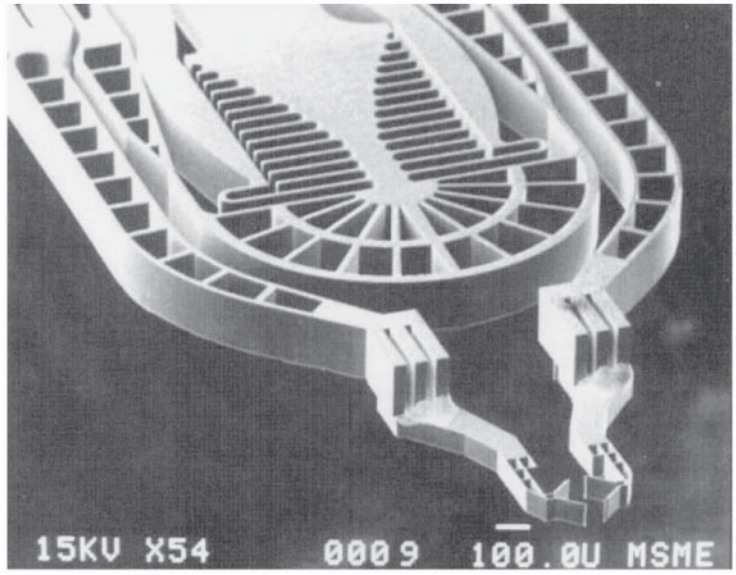


FIGURE 10.13 Cross-sectional diagram of the SCREAM process for the formation of silicon cantilevers with aluminum electrodes adjacent to each side of the beam. The undercut of the silicon sidewalls isolate the aluminum from the substrate. (McDonald, N. C., *Microelectroic Engineering*, 32, 49, 1996. With permission.)



(a)



(b)

FIGURE 10.14 (a) Cross-sectional schematic of the HEXIL process; (b) micro-tweezers produced by the HEXIL process. (Keller, C., *Microfabricated High Aspect Ratio Silicon Flexures*, ISBN 0-9666376-0-7. With permission.)

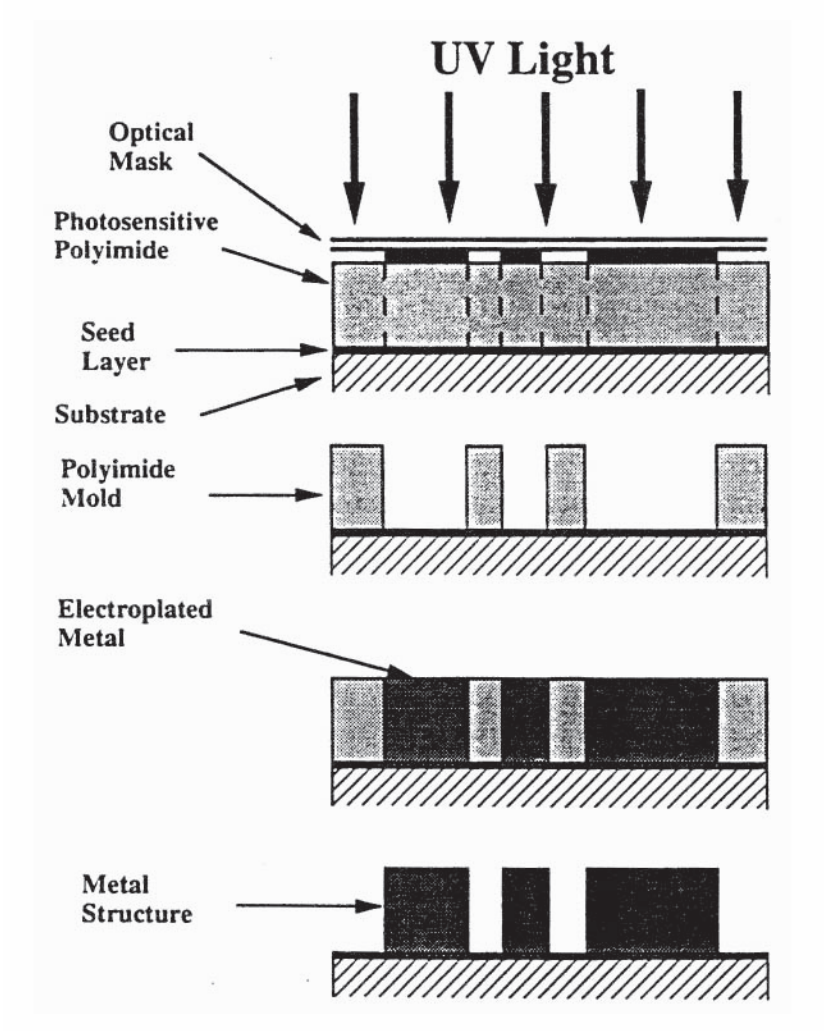


FIGURE 10.15 Schematic representation of the process sequence for fabricating metallic electroplated microstructures using photosensitive polyimide. (Frazier, A. B., et al., *Sensors and Actuators A.*, 45, 47, 1994. With permission.)

the X-rays produces damage in the dielectric components of the CMOS circuit. An excellent review of recent development in LIGA processing is given by Friedrich et al.[51] Optical gratings have been made by the LIGA process by Cox et al.[102]; specifically, infrared tunable filters driven by magnetic actuators, features 30 μm in height and a filter period of 8 to 17 μm. (See Fig. 10.17.)

GaAs Micromachining

The properties of GaAs as a MEMS material have been investigated by Hjort et al.[103] The key advantages of GaAs are that it is a direct-bandgap semiconductor also having high electron mobility, so that optoelectronic devices can be realized, and the piezoelectric properties are comparable to that of quartz. Finally, electronic devices can be operated at temperatures up to 350°C. Although the mechanical strength of the material is lower than silicon, the material can be bulk micromachined with HNO_3/HF solutions. Surface-type micromachining processes have also been developed, based on the advanced state-of-the-art in modulated material doping and selective etching of AlGaAs. Interdigitated electrostatic actuators have been realized in addition to surface acoustic wave sensors (SAW) optical sensors, and piezoelectric sensors (see later section on optical devices).

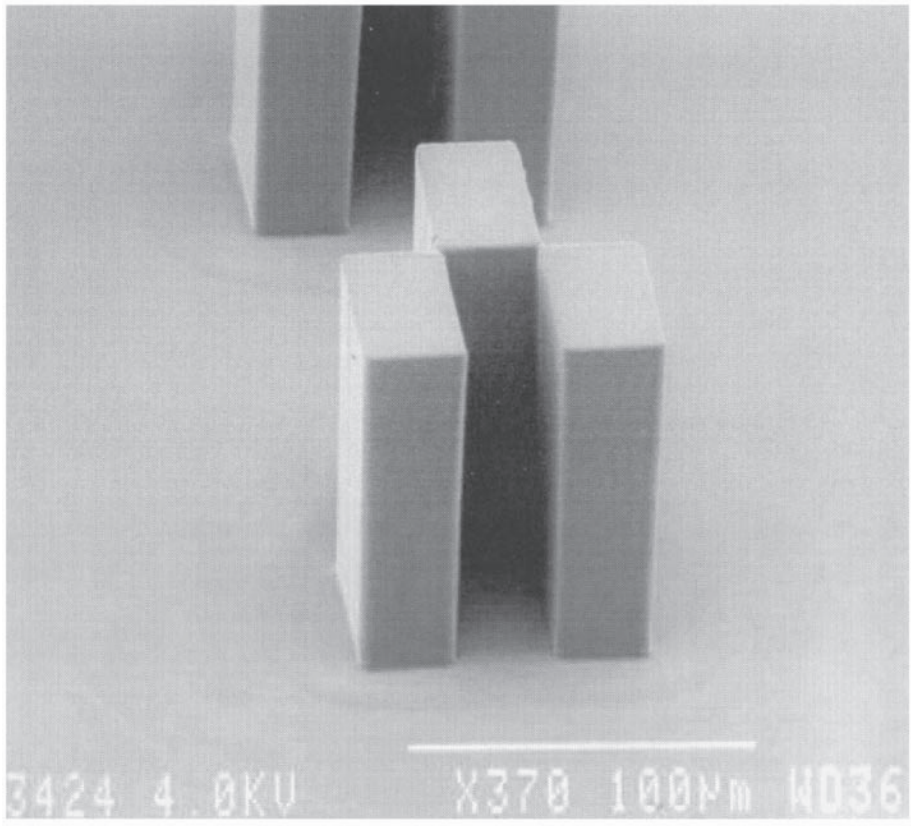


FIGURE 10.16 SEM photomicrograph of a high aspect ratio 100-micron thick SU-8 photoresist structure formed by a single spin coating and contact lithography exposure. (Photo courtesy of Electronic Visions, Phoenix, Arizona. With permission.)

10.6 CMOS and MEMS Fabrication Process Integration

One of the key challenges in micromachining processes is combining the electronic devices with the mechanical, optical, or chemical function of the MEMS device. Although most work has utilized a hybrid approach in which the MEMS device is fabricated independently of the interface electronics, there are several examples of integrated sensors and other devices. The early work of Parameswaran et al.[104] demonstrated a post-CMOS processing anisotropic etching in KOH solutions that was feasible under limited conditions. The circuit elements and the MEMS devices were defined by a standard CMOS process and then the structures were released from the substrate or thermally isolated from the silicon substrate by a post-processing step. Requirements for such an etch are that it is compatible with exposed silicon dioxide, and silicon nitride, on the chip. Other etching chemistry may be more suitable such as XeF_2 or TMAH. Another issue is ionic contamination which leads to failure of the CMOS devices through mobile ions present in the gate dielectric,[105] producing instability in threshold voltages.

There have been several approaches for combining CMOS circuits with MEMS structures, summarized as follows:

- Post-processing : protecting the CMOS circuit with a chemically resistant film(s) and carrying out the micromachining after the circuits are complete and avoiding any high-temperature steps.
- Combined processing: integration in a custom MEMS/CMOS process or utilizing the CMOS layers themselves for MEMS devices.

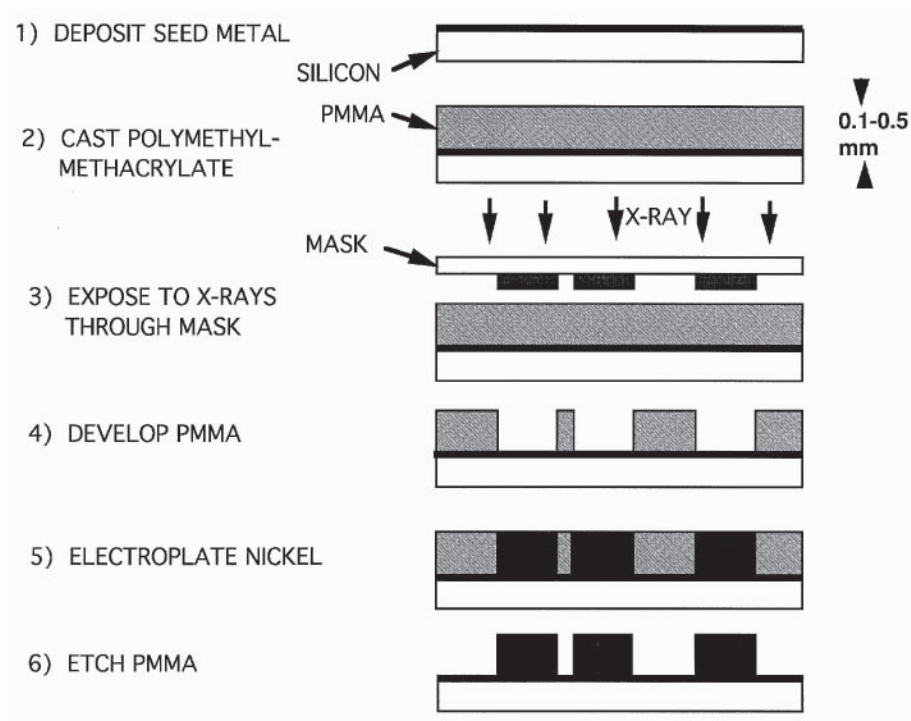


FIGURE 10.17 Schematic diagram of the LIGA process modified for MEMS.

- Pre-processing: etching wells in the wafer of a depth equal to the total height for the formation of the MEMS device. Fabrication of MEMS devices and protection with an encapsulation layer that is planar with the silicon surface. CMOS circuit fabrication then follows, and removal of the encapsulating film releases the MEMS structure.

Post-processing

Micromachined thermally isolated regions have been developed by Parameswaran et al.[106] in a CMOS-compatible process for infrared emitter arrays. Post-processing was carried out following a commercial Northern Telecom Canada Ltd. COMS3 DLM (3-μm 13 mask) process. Openings are defined in layers so that the silicon surface is exposed for the EDP etching. The active devices, both p-MOS and n-MOS, were tested after etching and there was no change in device performance.

Fedder et al.[107] have demonstrated the fabrication of intricate structures with a standard CMOS process. Structures are designed in the metal and polysilicon layers and aligned so that a post-processing dry etch step defines the dimensions and a post-processing etch undercuts the structure. Releasing it from the substrate is carried out by RIE etching (see Fig. 10.18). Electrostatic comb drives actuate microstructures that are 107 μm wide and 109 μm long show a resonance amplitude of 1 μm with an ac drive voltage of 11 V. The effective Young's modulus of the structure was found to be 63 GPa. The design rules are as follows. The scaling factor for the 0.8-μm process is $\lambda = 0.4$ μm. The minimum beam width is 3λ (1.2 μm), and the minimum gap is 3λ (1.2 μm); however, for holes, a minimum dimension of 4 μm is required for release. The CMOS circuit is protected by metal-3 to prevent etching. The metal-1 and -2 collar is inserted underneath the break in metal-3 to prevent the etch from reaching the surface and facilitating electrical interconnects to the MEMS structure.

Finally, planarization techniques have been developed by Lee et al.[108] to prepare a foundry-fabricated chip for post-processing by other low-temperature methods, including electroforming, LIGA, and reactive ion etching.

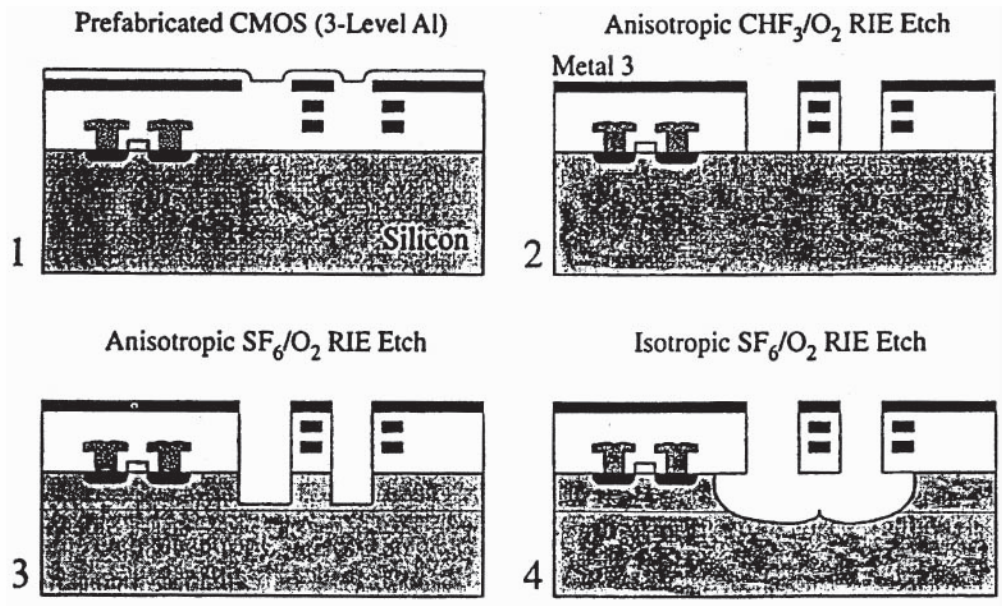


FIGURE 10.18 Example of the use of variable anisotropy dry etch on prefabricated CMOS integrated circuit using the upper level of metallization as the mask: (1) CMOS cross-section, (2) anisotropic CHF_3/O_2 RIE process, (3) anisotropic SF_6/O_2 RIE process, and (4) isotropic SF_6/O_2 RIE process. (Fedder, G. K. et al., in *Proceedings of the International Meeting on MicroElectroMechanical Systems*, IEEE, p. 13, 1996. With permission.)

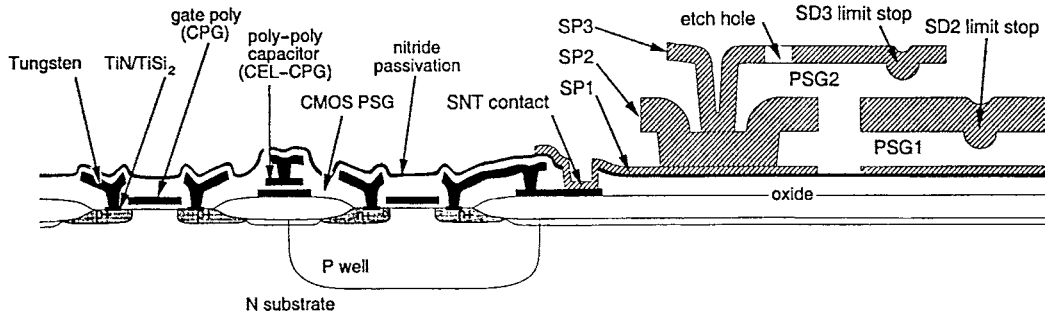
Mixed Processing

The surface micromachined accelerometer manufactured at analog devices utilizes an integrated process. The BiCMOS process is interleaved with the micromachining so that the higher-temperature steps are completed first; then the lower temperature steps and metallization complete the device fabrication.

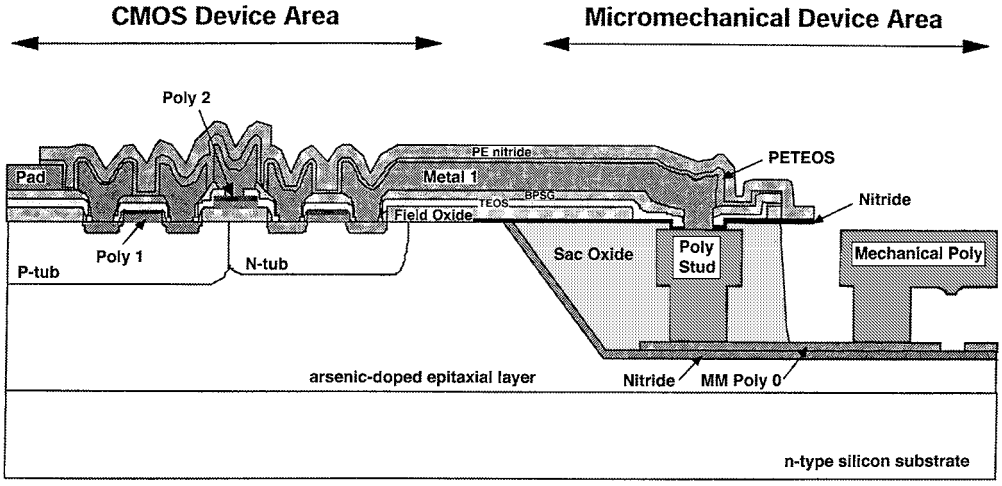
A different approach has been taken by the group at the Univesity of California at Berkeley, using high-temperature metallization of tungsten and titanium silicide and TiN barrier layers to replace the aluminum.[109] A double polysilicon single metal, n-well CMOS technology is fabricated first, encapsulated with PSG and low-stress nitirde, as shown schematically in Fig. 10.19.

Pre-processing

Smith et al.[110] pioneered the pre-processing of fabricating the MEMS device in a buried well and then encapsulation with a protective passivation film, which is later be removed once the CMOS process is complete. The key to this process is the use of CMP to planarize the surface after MEMS device fabrication and before the CMOS fabrication is begun. The release etch must be highly selective to materials in the MEMS structure and not damage the CMOS outer layers of material (see Fig. 10.19). This process has been very successful in fabricating structures such as pressure sensors, electronic oscillators, microaccelerometers, and gyroscopes. Also, Gianchandani et al.[111] have demonstrated pre-processing integration of thick polysilicon microstructures with a CMOS process. Silicon-to-silicon bonding has been utilized for sensor integration with a pre-etched sealed cavity process,[112] shown in Fig. 10.19(b). The thin membrane formed undergoes plastic deformation, and as a result, the proper design of the cavity geometry is critical to control the gas pressures during bonding. Pressure sensors, accelerometers, and gas flow shear stress sensors have been demonstrated with and without integrated electronics. Lowering the bonding temperature is of key interest to allow more widespread use of this bonding method — because of its incompatibility with many materials and processes.



(a)



(b)

FIGURE 10.19 (a) Schematic cross-section of the modular integration of CMOS and microstructures technology using post-IC MEMS fabrication with tungsten interconnect technology; (b) a cross-sectional schematic of the subsurface, embedded MEMS integrated technology. (Sniegowski, J. J., in *Microstructures and Microfabricated Systems-IV*, Vol. 98-14, Ed: P.J. Hesketh, H. Hughes, and W. E. Bailey, The Electrochemical Society, Pennington, New Jersey, 1998. With permission.)

10.7 Wafer Bonding

Anodic bonding was first demonstrated by Wallis and Pemerantz[113] and can be carried out between a range of glass/metal sealing combinations. The key to obtaining a reliable to bond is to minimize the stress; thus, the glass must be selected very carefully to match the expansion coefficient with the silicon. For example, the expansion coefficient of silicon is a function of temperature and that of Pyrex glass is much less so (Fig. 10.20). Sodium ions that become mobile at elevated temperatures (>300°C) produce a depletion layer at the silicon/glass interface.[114] The resulting electrostatic attraction between these two charge layers brings the surfaces into intimate contact, and a chemical bonding takes place that is irreversible and strong. This bond is hermetic and is widely used in microsensor technology for pressure sensors and other devices, such as fluid pumps and valves.[115] The process compatibility with CMOS is

TABLE 10.5 Wafer Bonding Techniques

Bonding Technique	Materials	Surface Treatment	Process	Time	Bond Strength/Comments	Ref.
Anodic bonding	Silicon/7740 Pyrex glass	Clean	350–450°C ~500-1000V	~1-10 min	1-3 MPa[a]/uniform reliable hermetic bond formed	113,116
Silicon-silicon	Si-Si SiO_2-Si and SiO_2/SiO_2	Hydrophobic Hydrophilic	500–1100	Hrs	Difficult to avoid voids unless processed at higher temperatures	117,118
Borosilicate glass	$Si/Si0_2$ and Si_3N_4		450	30 min	—	120,122
Eutectic	Si-Au-SiO_2	Clean and oxide-free	~350	—	148 MPa[b]/Nonuniform bonding area	119
Solder	SiO_2-Pb/Sn/Ag-SiO_2	Needs solder flux	250–400	min	Large difference in thermal expansion coefficient can lead to mechanical fracture	119
Glass frit	SiO_2-glass Ag mixture-SiO_2	Clean	~350	<hr	Difficult to form thin layers	121

under investigation.[116] The circuit is protected from the large electrostatic fields by shorting the gate regions together with a polysilicon strip. After bonding, regions were opened up in this area to facilitate etching. In addition, cavities were drilled ultrasonically in the Pyrex wafer to reduce the electric field over the active circuits.

The key advantage of silicon-silicon direct bonding is that the same material is used so there are minimal thermal stresses after bonding. The wafers must be flat, scrupulously clean, and in prime condition to achieve a reliable bond. First, the wafers are chemically cleaned and surface-activated in a nitric acid solution. Then the wafers are bonded at room temperature in a special jig that has been demonstrated to improve the bonding yield, as shown in Fig. 10.21.[117] A subsequent anneal step increases the bonding strength through a chemical reaction that grows a very thin silicon dioxide layer at the interface (Fig. 10.21(b)). The wafers can be inspected for voids utilizing an infrared microscope or an ultrasonic microscope. High yield has been achieved and the community that develop silicon-on-insulator technology have published conference proceedings on these methods.[118]

Other bonding methods are listed in Table 10.6; these include eutectic,[119] low-temperature glass,[120] glass frit,[121] and borosilicate glass.[122] Materials are selected to minimize the stresses in the bond by selecting a match in the thermal expansion coefficients, or a compliant layer is utilized at the interface.

10.8 Optical MEMS

There is great interest in taking advantage of MEMS devices for the manipulation of optical beams for which they are naturally suited, because of the ease of batch fabrication into large arrays, the small forces required, and the high speed of operation. Examples of devices that have been demonstrated include chopper beam steering, diffraction gratings, optical scanners, Fabre-Perot interferometer, sensors, and spectrometers on a chip. This demonstrates the high speed of operation that is achievable with MEMS technology. A recent excellent review of MEMS devices is given by Wu[123] and an earlier paper by Motamedi.[124]

Components

In the first generation of microfabricated optical components, hybrid assembly into optical devices was carried out rather than integrated functionality. Various optical components were reviewed by Motamedi et al.,[125] including diffractive optical components integrated with infrared detectors, refractive micro-optics,

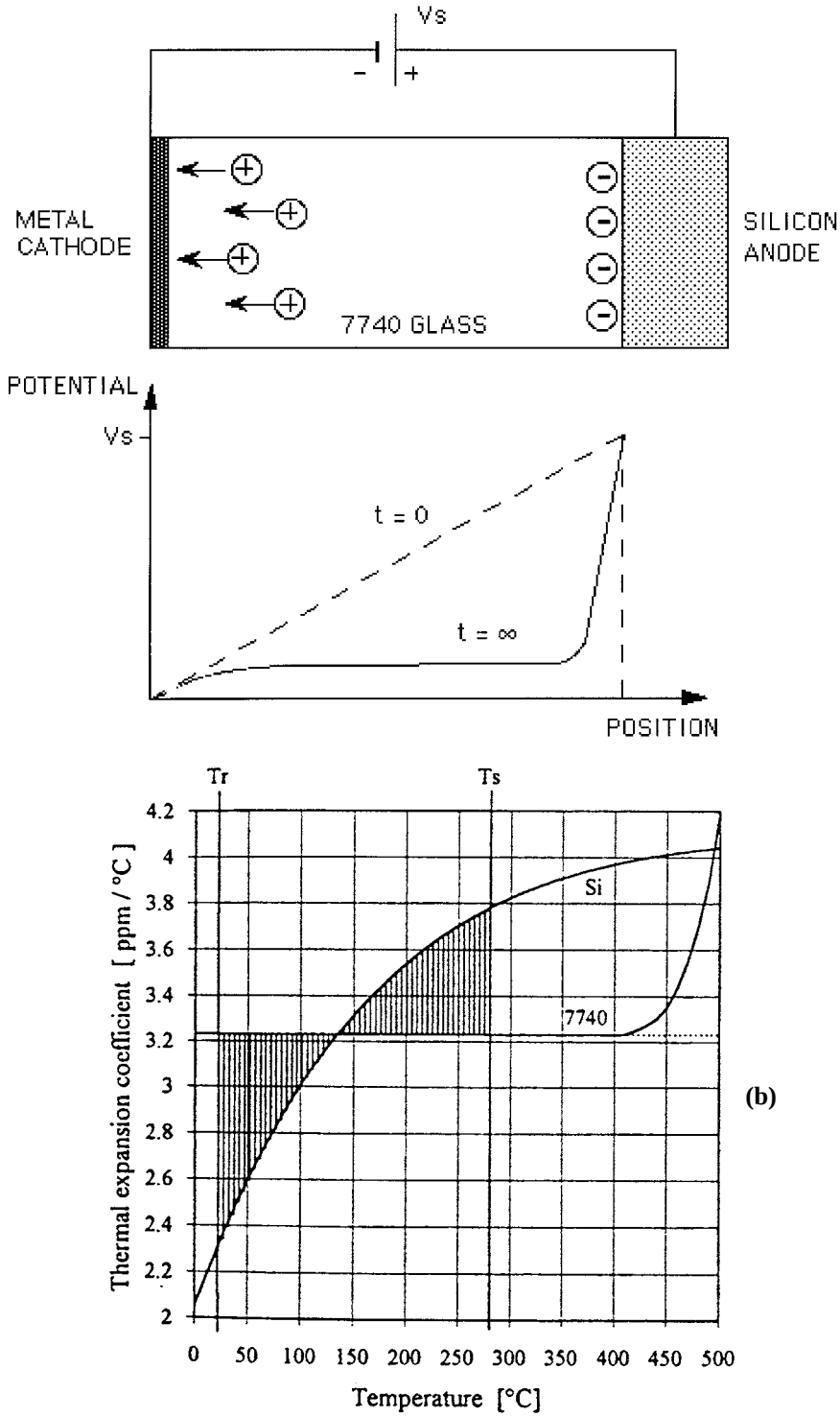


FIGURE 10.20 (a) Schematic diagram showing anodic bonding process and potential distribution; (b) thermal expansion coefficient of Si and Pyrex glass as a function of temperature. (Peeters, E., Process Development for 3D Silicon Microstructures with Application to Mechanical Sensors Design, Ph.D. thesis, Catholic University of Louvain, Belgium, 1994. With permission.)

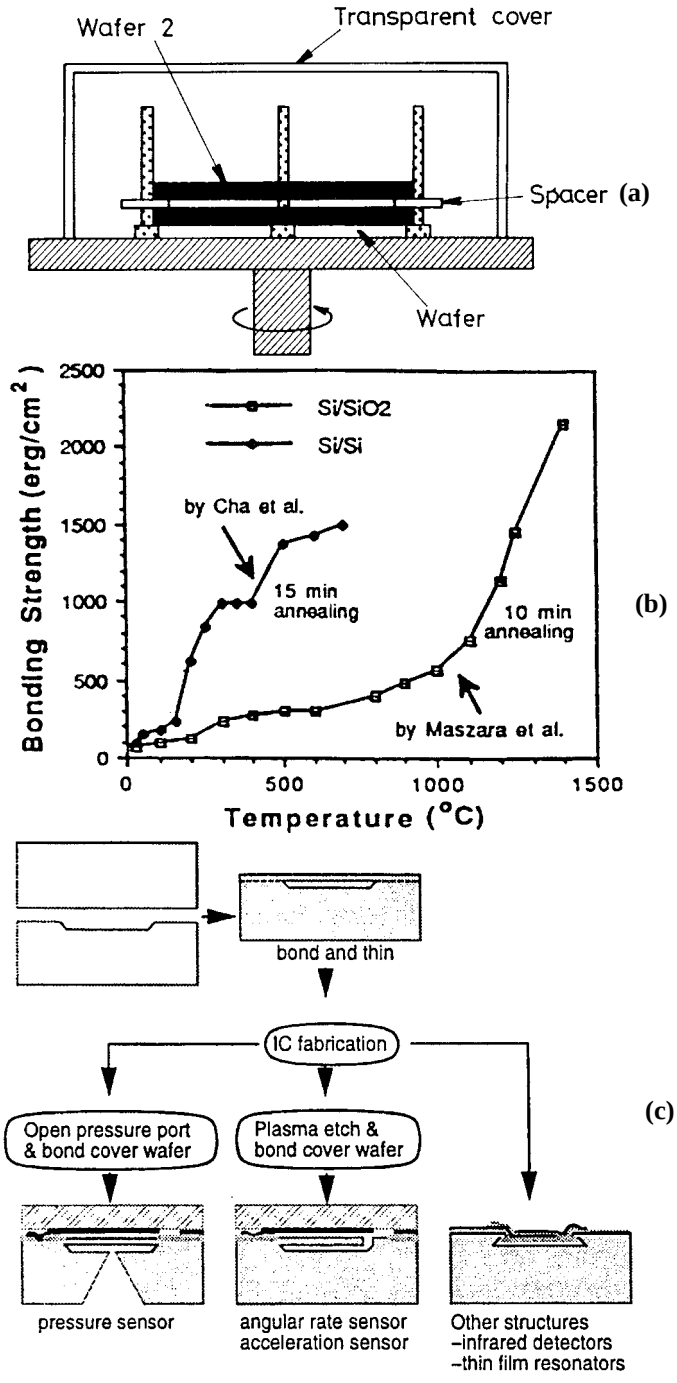


FIGURE 10.21 (a) Schematic diagram of set-up for direct silicon-to-silicon wafer bonding process (Cha, G. et al., in *Proc. First Int. Symp. Semicond. Wafer Bonding Sci. Tech. Appl.*, Eds, Gösele, U. et al., *The Electromechanical Society*, Pennington, NJ, p. 249, 1992 and Masgara, W. P. et al., *J. Appl. Phys.*, 64, 4943, 1989. With permission.); (b) bond strength versus anneal temperature for silicon-silicon direct bond (Mitani, K. and Gösele, U.M., *J. Electron. Mat.*, 21, 669, 1992. With permission.); (c) method for formation of silicon diaphragm by silicon to silicon bonding. (Parameswaran L. et al., in *Meeting Abstracts of the 194th Meeting of the Electrochemical Society*, Boston, Nov. 1-6th, Abst #1144, 1998. With permission.)

and microlenses. However, the integration of optical components into a micro-optical bench offers several advantages over guided wave approaches; in particular, high spatial bandwidth, independent optical routing 3-D interconnects, and optical signal processing. Moving the individual optical elements out of the plane has greatly expanded the utility of this method.[126] Micro-electrostatic deformable mirrors have also been demonstrated as an effective method for reducing aberrations.[127]

Tunable semiconductor laser diodes have been demonstrated by Uenishi et al.[128] with anisotropically etched (110) silicon substrates and hybrid assembly. A cantilever 1.7 mm long and 8 μm wide defines a (111) surface reflecting mirror normal to the substrate provided modulation from 856 to 853 nm, with a drive of 12.5 V. The etching conditions in KOH solutions were optimized for minimum surface roughness of the (111) surface.

External mirrors for edge emitting lasers can also be produced in GaAs by micromachining. Larson et al.[129] have demonstrated a Fabry-Perot mirror interferometer by combining a GaAs-AlAs vertical cavity laser VCSEL with a suspended movable top mirror membrane. The bottom mirror is a 12.5 period GaAs/AlAs distributed Bragg reflector of 640 Å/775 Å thickness with a center wavelength of 920 nm. The GaAs laser had a center wavelength of 950 nm and a cavity of 2580 Å thick GaAs layer. The top electrode was 2230 Å SiN_xH_y with a nominal 200-Å Au reflector. The air gap thickness is modulated around 3λ/4 with electrostatic means to provide a 40-nm tuning range with an 18-V drive. Figure 10.22(b) shows the reflectance spectra from the device.

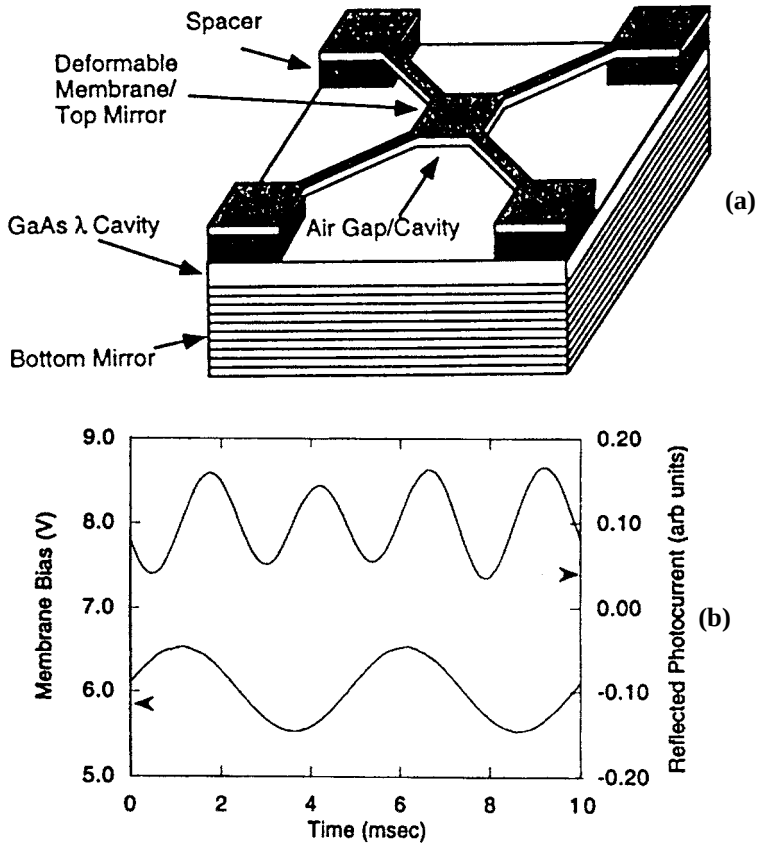


FIGURE 10.22 (a) Schematic diagram of coupled-cavity microinterferometer; (b) membrane voltage and reflected photocurrent traces for the device acting as an intensity modulator for an active wavelength of 933 nm. (Larson, M. C. et al., *IEEE Phot. Tech. Lett.*, 7, 382, 1995. With permission.)

Fabry-Perot tuning was also applied to a photodiode detector by Wu et al.[123] A DBR mirror is defined on top of a movable cantilever. A 30-nm tuning range was achieved with a 7-V drive and 17-dB extinction ratio. The DBR comprised a top reflector of 18 pairs n-doped $Al_{0.6}Ga_{0.4}As$-$Al_{0.1}Ga_{0.9}$ As and a fixed portion was two pairs p^+-doped $Al_{0.6}Ga_{0.4}As$-$Al_{0.1}Ga_{0.9}As$. The bottom DBR was 13 pairs of n-doped AlAs-GaAs mirror grown onto an n^+-doped GaAs substrate.

Modulators and Active Gratings

Micromechanical optical modulators offer certain advantages over traditional means, specifically, maintenance of wavelength coherence, reliability, and temperature insensitivity. Goossen et al.[130] have developed a silicon mechanical antireflectance switch based on a silicon nitride membrane. The thickness of the nitride membrane can be defined precisely so that an antireflection condition occurs when $\lambda/4$ film is brought into contact with no gap. The air gap defines a second coupled cavity at $m\lambda/4$. High transmission is achieved for m even and reflection for m odd. Contrast ratios of 24 dB were obtained with maximum response times of 250 ns.

Sene et al.[131] developed a grating light modulator based on a surface micromachined polysilicon layer. Each polysilicon beam in the array is individually addressed and deflected electrostatically. When the grating is moved, it diffracts the optical beam from the zero order to the +/- first order (see Fig. 10.23). The design incorporates drive plates at the edges of the grating so that the grating lines are not part of the electrostatic actuation. Two anti-sag support lines keep the grating lines parallel during actuation. There are also 0.75-μm deep dimples in the upper electrode to prevent stiction. Thermally actuated beams are used to assemble these structures (see Section 10.9).

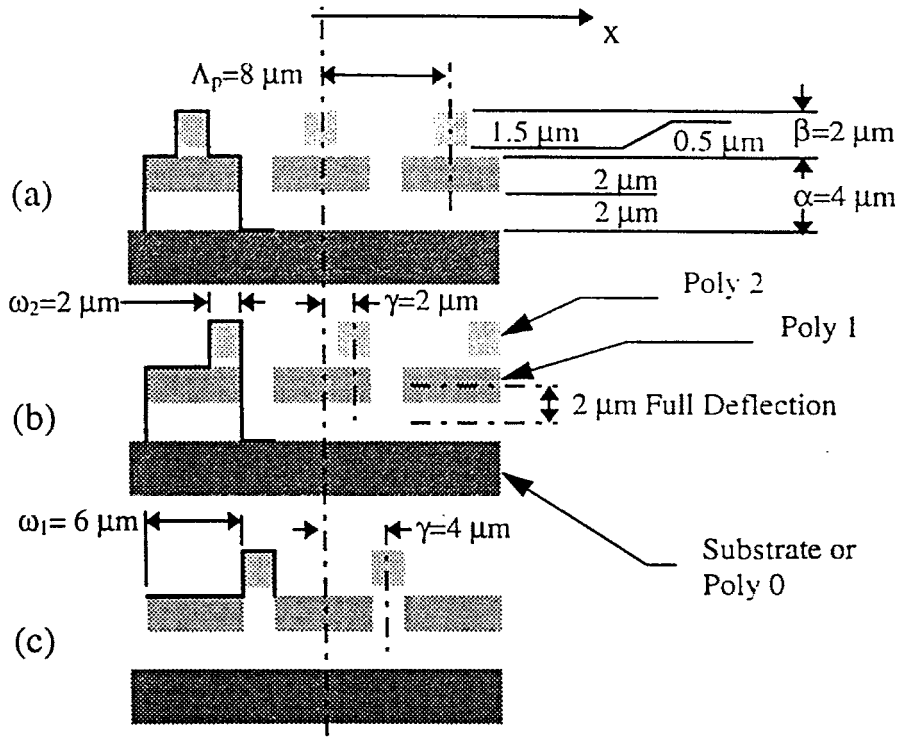


FIGURE 10.23 Schematic representation of variable diffraction grating. The value of γ will depend on the position of the upper, poly-2 grating and the three positions span a full period as indicated by the solid lines, for the following values of γ: (a) $\gamma = 0$, (b) $\gamma = 2$ μm, and (c) $\gamma = 4$ μm. (Sene, D. E. et al., in *Proceedings of the Ninth Annual International Workshop on Micro Electro Mechanical Systems*, San Diego, CA, February 1996, 222. With permission.)

Burns and Bright[132] have developed microelectromechanical variable blaze gratings operated by adjusting the blaze angle of each slat so that specular reflection of the incident light matches a particular grating diffraction order. Figure 10.24 shows a grating element that was fabricated with polysilicon using the MUMPS process surface micromachining available at MCNC.[45] Both electrostatic and thermal actuators were studied. Light beams of diameters greater than 1 mm and power levels of 1 W have been directed. Measurements with 20 mW HeNe (632.8 nm) produced diffraction efficiency in the far field of 55%, which agreed with model results. Devices with gold metallization demonstrate improved reflectivity; however, they are not fully compatible with CMOS processing.

Scanning Mirrors

Miller et al.[133] fabricated a magnetically actuated scanner with a 30-turn coil on an 11-μm thick permalloy layer. The external magnetic field provided deflection while the coil provided fine control and/or fast motion. Asada et al.[134] fabricated optical scanners with bulk micromachining and a magnetic drive. The electroplated copper used photoresist mold with period of 50 μm. Coils formed on the *x*-axis and *y*-axis plate of the Pyrex glass plate. Spring constants were evaluated for the *x*- and *y*-axes at 6.48×10^{-4} Nm and 12.8×10^{-4} Nm, respectively, and resonant frequencies of 380 Hz and 1450 Hz, respectively. Judy and Muller[135] demonstrated a torsional mirror scanner moved with a magnetic field. They electroplated a nickel mirror 450-μm square on a polysilicon flexure over a 10-turn coil. With a current of 500 mA and field of ~5 kA/m, the mirror moved more than 45°. Micromachined electromagnetic scanning mirrors have also been fabricated and demonstrated by Miller and Tai.[136] One advantage of magnetic actuators is that both attractive and repulsive forces can be generated. The mirrors are permalloy coated ($Ni_{90}Fe_{10}$) and formed on a silicon substrate with a 20-μm thick epitaxial layer for etch stop. Copper coils are electroplated into a photoresist mold. The mirror is shown schematically in Fig. 10.25(a), and the deflection as a function of the external magnetic field in Fig. 10.25(b). Utilizing these mirrors, holographic data storage has been demonstrated. Scanners are widely used in printers, display devices, graphic storage systems, and bar code readers.

Kiang et al.[137] have developed polysilicon hinged structures for scanners which utilized an electrostatic drive. The 200 × 250-μm mirror was rotated 12° with a drive voltage of 20 V, and the device had a resonant frequency of 3 kHz.

Fischer et al.[138] have utilized electrostatic means for mirror deflection with integrated p-well CMOS drive circuits. Two types of torsional mirrors, comprising a polysilicon layer with double-beam suspension and a reflecting area of 75×41 μm^2, have been realized. The mirrors were integrated by post-processing a layer of polysilicon at 630°C, implanting with phosphorus, dose 5×10^{15}/cm^2, and annealing at 900°C, resulting in a resistivity of 100Ω/sq. The electronics included a demultiplexer circuit for addressing the mirrors and a drive circuit producing 30 V for the electrostatic deflection. Bühler et al.[139] have also developed an electrostatically driven mirror made of aluminum in arrays for low-cost applications. The CMOS-compatible process consisted of modifying the second metal layer deposited process into two successive passes. The first, of 1.1 μm, established a thick metal layer for the mirror plate and the second, of 0.3 μm, a thin metal layer for the hinges. Deposition was carried out by sputtering at 250°C for improved step coverage; however, a roughness of ~53 nm resulted. Smooth reflecting surface room-temperature depositions were preferred, resulting in roughness ~12.5 nm. The mirrors were released by sacrificial aluminum and oxide etching. They were deflected with a drive voltage of 11 V for a pixel area of 30×40 μm^2.

Ikeda et al.[140] fabricated a scanning system that had a two-dimensional array with integrated photodetectors and piezoresistors. A bulk piezoelectric actuator moves the silicon nitride bridges with a torsional spring scanning angle of 40° and 30° bending and twisting and a resonant frequency of 577 Hz in bending and 637 Hz in torsional motion.

Huang et al.[141] have demonstrated piezoelectrically actuated ZnO cantilevers for application in projection displays, as shown in Fig. 10.26. One of the key advantages of piezoelectrically controlled motion is that the displacement is linearly proportional to the applied voltage. Although sputtered ZnO films have a lower piezoelectric coefficient than PZT, the fabrication process is compatible with CMOS

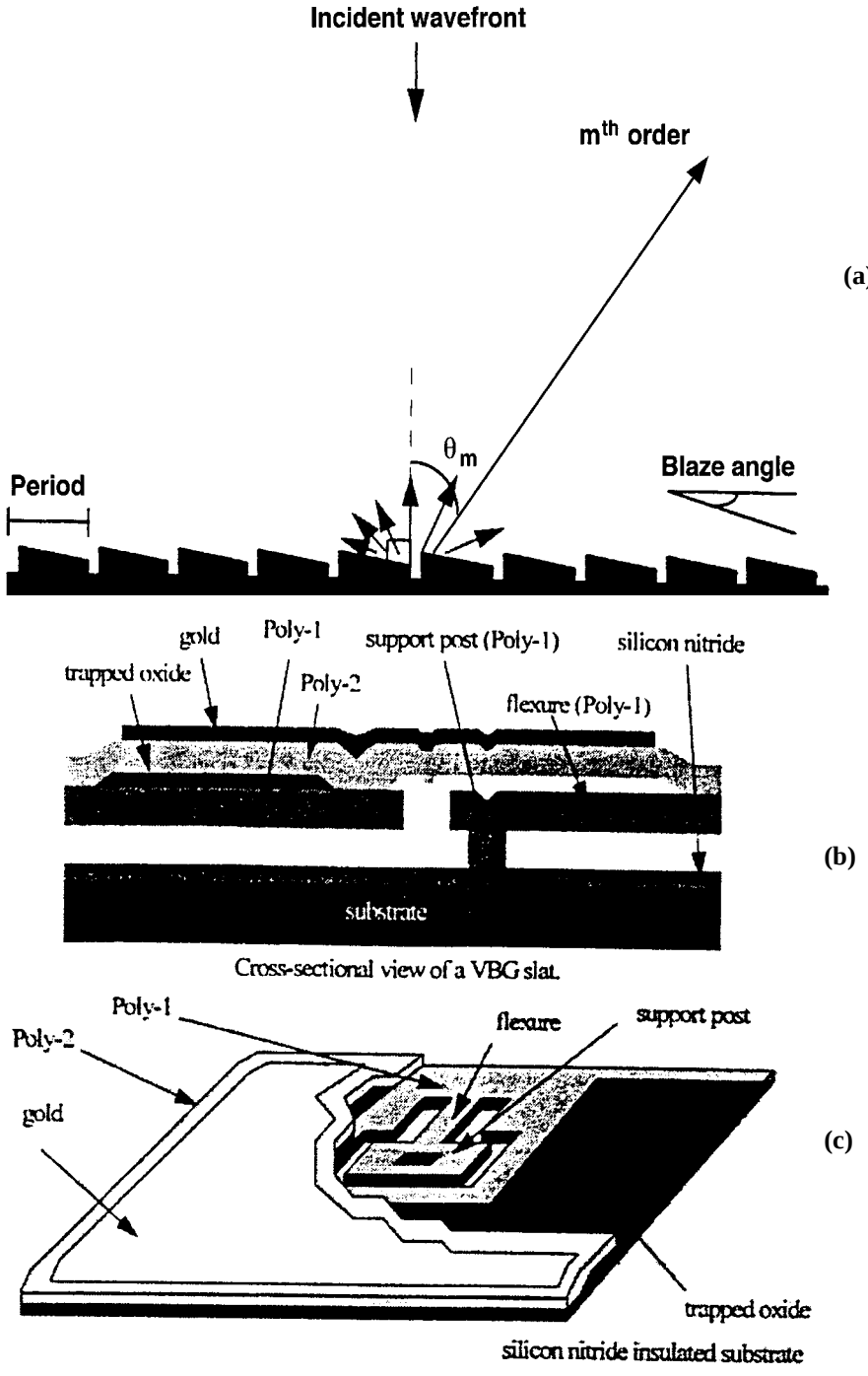


FIGURE 10.24 (a) Schematic diagram of reflective blazed grating illuminated at normal incidence; (b) cross-sectional view of the slat support posts and flexure used in the electrostatically actuated variable blaze grating; (c) the embossing present in the cross-section view of the gold layer. (Burns, D. B. and Bright, V. M., *Sensors and Actuators A*, 64, 7, 1998. With permission.)

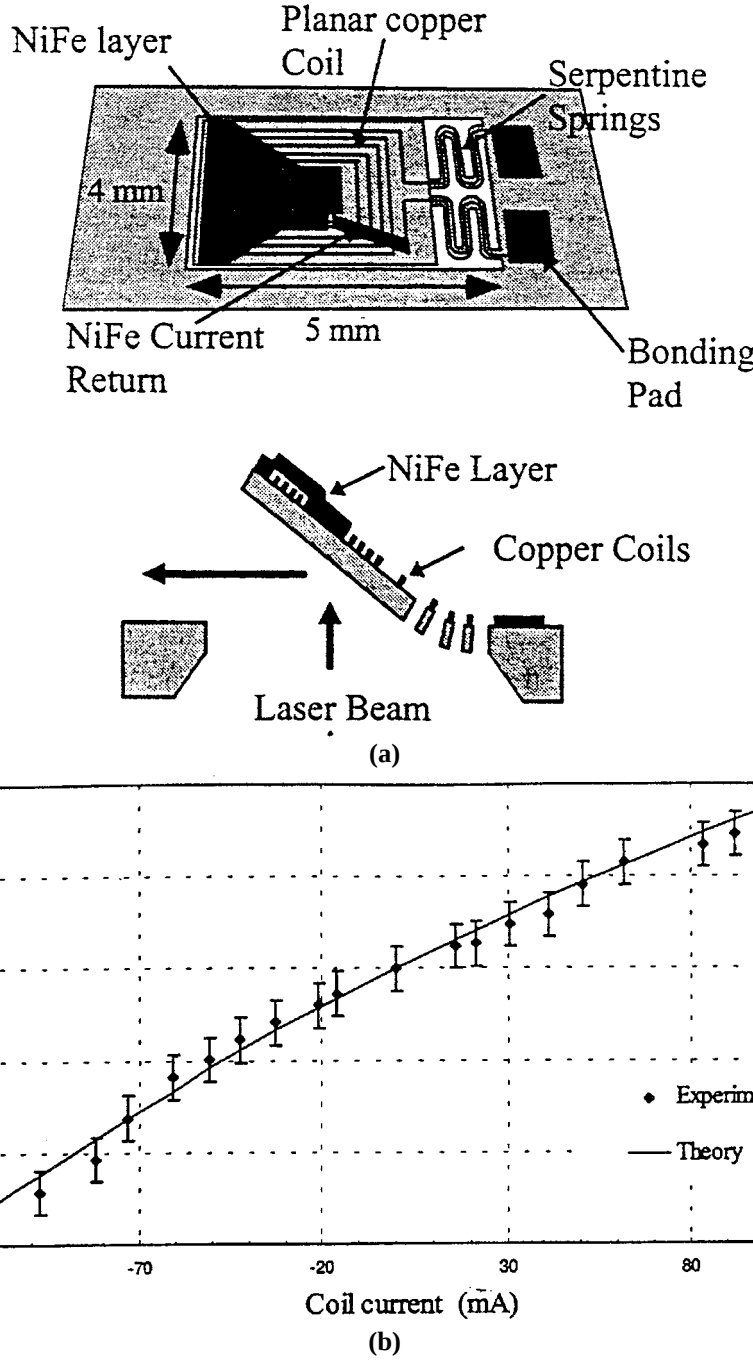


FIGURE 10.25 (a) Schematic diagram and cross-section of the deflected magnetic micromirror; (b) change in deflection angle from bias position for a variable coil current with external field of 994 Oe. (Miller, R. and Tai, Y.-C., *Opt. Eng.*, 36, 1399, 1997. With permission.)

processing. Calculations show that for a beam length of 150 μm, the tip deflection is 0.06°/V or 0.12 μm/V. The ZnO is fabricated with a sacrificial spin-on-glass process, the upper and lower electrodes formed from aluminum. The release step involves a HF vapor etch at low concentration to avoid attack of the Al and ZnO thin film. Tip displacements were measured with a laser interferometer and, in order

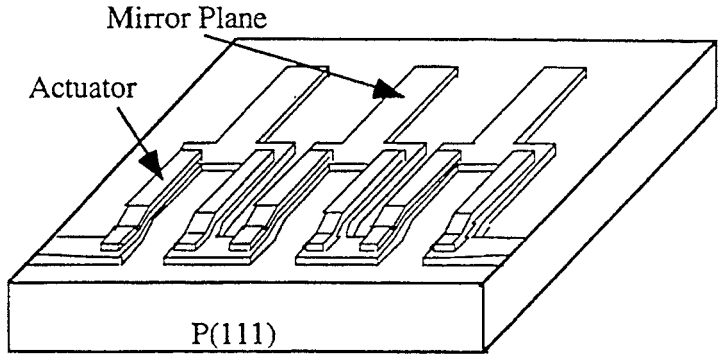


FIGURE 10.26 Perspective view of the ZnO-driven cantilever array. (Huang, Y. et al., in *Digest of Technical Papers, Solid-State Sensor and Actuator Workshop*, Hilton Head, South Carolina, June 1996, 191. With permission.)

to distinguish between any thermal contribution to the measured deflections, a drive waveform of unbiased square wave was selected. The frequency response was over 80 Hz with a 1 μm air gap and limited to about 10 Hz with a 0.5-μm air gap, indicating the dominance of squeeze film damping. The thermal deflection was bout 2 to 3 orders of magnitude less than the piezoelectric response.

Spectrometer on a Chip

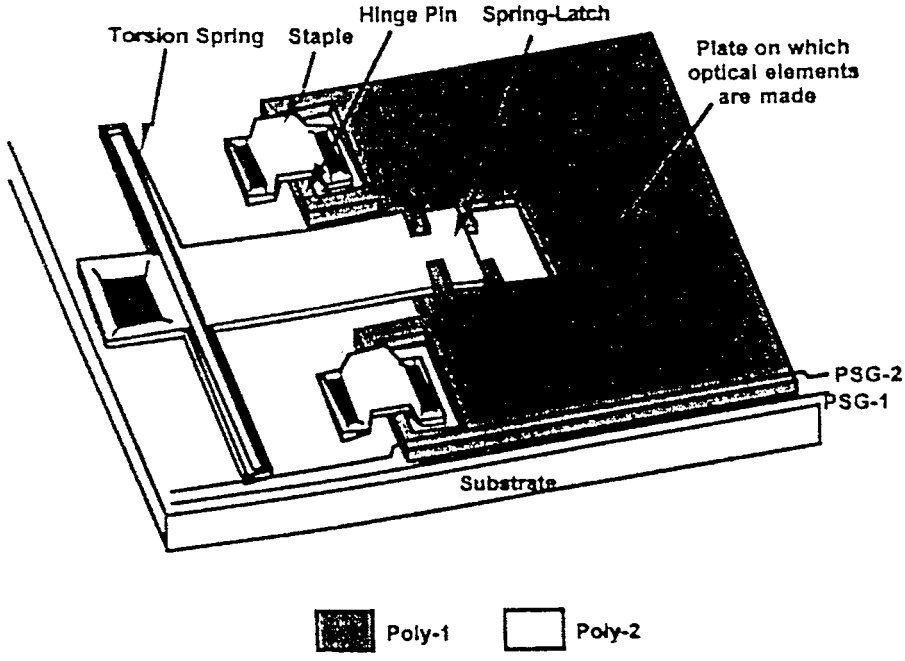
Surface micromachining has been demonstrated by Lin et al.[142] for out-of-plane assembly of optical elements. The hinge mechanism allows the plate to be moved to a vertical position and locked into place with a spring latch (see Fig. 10.27(a)). Micro-Fresnel lenses with a 280-μm diameter and an optical axis 254 μm above the plane of the silicon have been realized. The slide latch precisely defines the angle of the element (see Fig. 10.27(b)). It has a 'V' shape 2 μm wide in the center. In addition, rotating structures can be realized, such as a rotating mirror. Surface micromachined free space optical components have been demonstrated by Lee et al.[143] for the collimation and routing of optical beams. Micrograting with 5-μm pitch are fabricated on flip-up structures metallized with a thin layer of aluminum. A diffraction pattern was imaged with a CCD camera and beams directed to a second grating into the zero-order beam. Recent progress in micro-optical systems is reviewed by Bright et al.,[144] including mirrors, Fresnel lenses, gratings, and larger systems. All of these structures were fabricated by surface micromachining in the MCMC processes[145] using electrostatically actuated gold surfaced mirror.

Micromechanical Displays

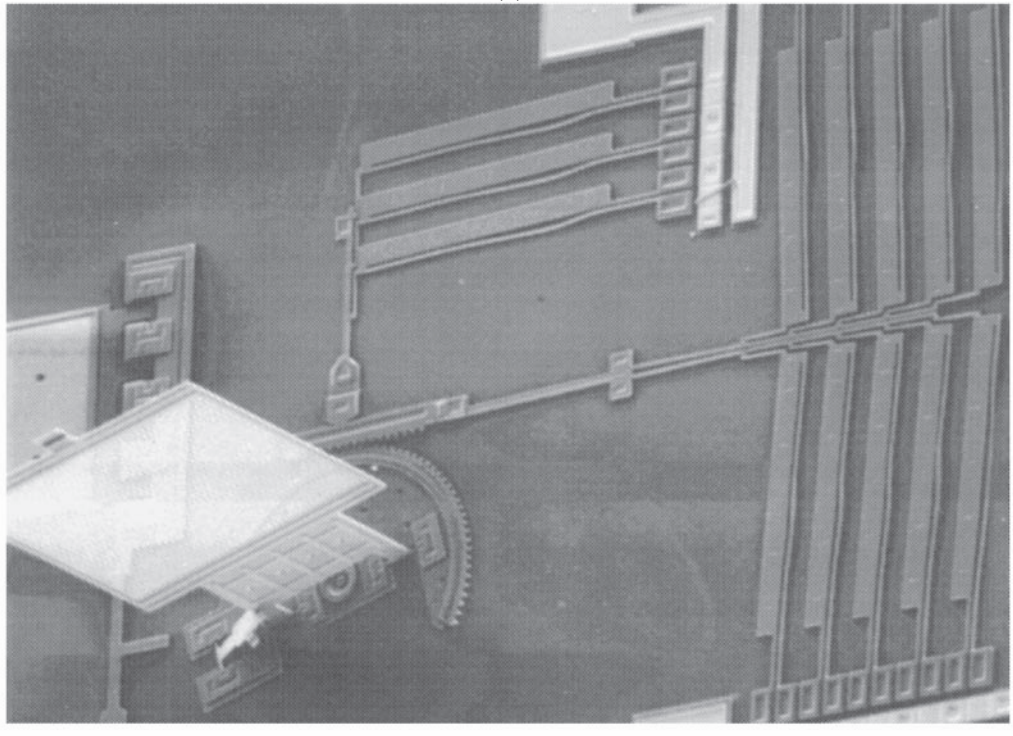
Miniature display systems have been commercialized by Texas Instruments in projection television systems.[146] They offer the advantage of cold operation and high contrast (>100:1) compared to the cathode-ray tube (CRT). A CMOS-like process over a CMOS memory element defined aluminum mirrors, each $16 \times 16\ \mu m^2$ in area, that can reflect light in two directions. The hinges and support structure are positioned under the reflector element. The display element can be moved by up to +/-10° and at speeds up to 10 μs, which make them suitable for standard rate NTSC video. Figure 10.28 shows the structure of one element in the 124×124 elements. The underlying memory cell operates on 5 V. Eight-bit pulse width modulation of the mirror state produces a gray scale or color image. The fabrication process is compatible with CMOS processing; however, to date, only the display has been fabricated on chip, and hybrid packaging is used for the drive electronics.

Optical Disk Pick-up Head

The technology for fabrication of optical pick-up heads is currently limiting the track access speed and maximum data rate. Ukita et al.[147] developed a flying optical head consisting of a laser diode and a



(a)



(b)

FIGURE 10.27 (a) Schematic diagram of the three-dimensional micro-optics element. After release etch, the micro-optical plate can be rotated out of the substrate plane and locked by the spring latches (Bright, V. M. et al. *IEICE Trans. Electron.*, E80-C, 206, 1997. With permission.); (b) SEM micrograph of the micro-Fresnel lens in the micro-XYZ stage integrated with eight scratch drive actuators. (Lee, S. S. et al., *Appl. Phys. Lett.*, 67, 2135, 1995. With permission.)

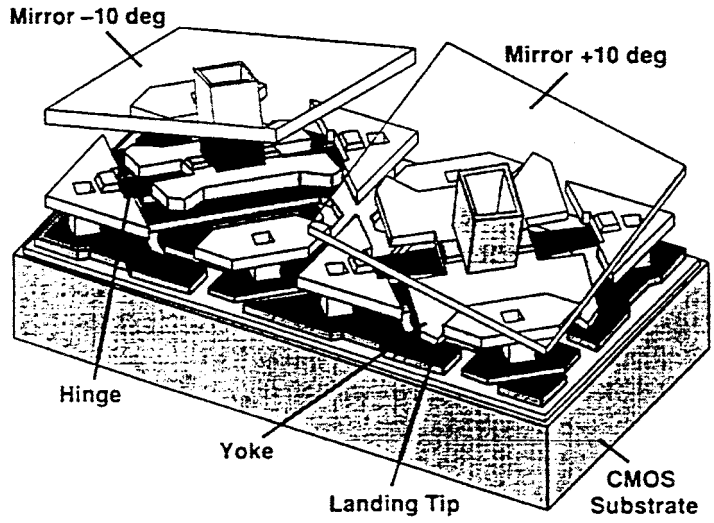


FIGURE 10.28 A two digital micro-mirror device pixels mirrors are shown as transparent for clarity in the diagram. (Hornbeck, L. J, in *Symposium Micromachining and Microfabrication, Proceedings of SPIE,* Vol. 2783, Austin, TX, 1996, 2. With permission.)

photodiode. The recording medium was a phase change material SbTeGe on a 130-mm diameter glass substrate. The read/write head was mounted on a gold electrode on the slider. Light reflected from the medium was fed back into the active region, with head-to-disk spacing of typically <1 μm. The 1.3-μm wavelength InGaAsP laser diode has a spot diameter constrained by the ridged waveguide shape. The reflection of the recording medium was reduced by an antireflection coating of 0.24-μm SiN_xH_y. Recording was achieved by producing a change in the reflectivity of the phase change medium, based on crystalline to amorphous states of the film, typically at a power of 20 mW and data rates of up to 1 MHz. Reading is achieved at lower power and higher rates. A single chip which integrates a photodetector, several Fresnel lenses, and a semiconductor laser has been demonstrated by Lin et al.[148] Surface micromachining allows the integration of a 45° reflector, Fresnel lens, rotary beam splitter, and photodetector on a chip. A 980-nm laser source was attached to the surface with an optical axis 245 μm above the silicon surface. The beam splitter has a 20-nm gold layer and the reflectors and mirrors have a thicker layer of gold. The focusing lens, with a NA of 0.17, results in a spot with FWHM of 6.1 μm in the *x*-direction and 2.6 μm in *y*-direction. The advantages of this system are small size, light weight, and potentially low-cost integration of actuation on chip to achieve track-to-track alignment.

10.9 Actuators for MEMS Optics

A variety of actuation principles have been demonstrated that are compatible with optical elements on chip. A limited number of these are compatible with CMOS processing. Further developments in integration of miniature optical components with functional optical MEMS devices is expected in the near future.[149]

Electrostatic Comb Drive Actuators

The electrostatic force generated between two conductive plates provides a compact, efficient actuation principle. These actuators are often limited to small displacements, and the force depends on the capacitance of the electrode. Interdigitated comb drives provide increased force and extended linear range, compared to parallel plate design. Recently, a large displacement actuator has been designed at Sandia National Laboratories using multiple stages of gearing (see Fig. 10.29). In general, high voltages of ~100 V

are required to drive electrostatic actuators. Circuits have been designed to work in conjunction with micromachining processes to integrate the actuators and drive circuit on the same chip.[150]

Linear Microvibromotor

The linear microvibromotor is based on impact momentum to produce small displacements. Each impact from the comb drive produces a step of typically 0.27 μm. Although this is an impulsive drive, the standard deviation between steps is 0.17 μm. A maximum speed to 1 mm/s has been demonstrated and used for a slide-tilt mirror and alignment of beams in a fiber coupler.[151]

Scratch Drive

The scratch drive is based on applying pulses to a plate and allowing the successive bending and release to produce lateral motion of the bushing to move out.[152] During release, the none-symetric functional forces produce an incremental motion DX. Microactuators and XYZ stages have been developed for a micro-optical bench.[153] A comb drive is used to drive the torsional *z* actuator with displacements up to 140 μm. Figure 10.29(b) is a schematic diagram of the microactuator stage. The lower 45° mirror is moved to achieve lateral adjustment of the beam. The translation stages are defined in the first (poly-1) layer, and the second (poly-2) layer defines the optical elements. The scratch drive actuator is particularly suited for this application because of the high forces and small step size (~10 nm) at moderate drive voltages of 87 V. Two-dimensional optical beam scanning has been demonstrated of several mm in the far field at a distance of 14 cm utilizing a HeNe laser source. A micro-Fresnel lens has been integrated into the actuator with eight scratch drive actuators.

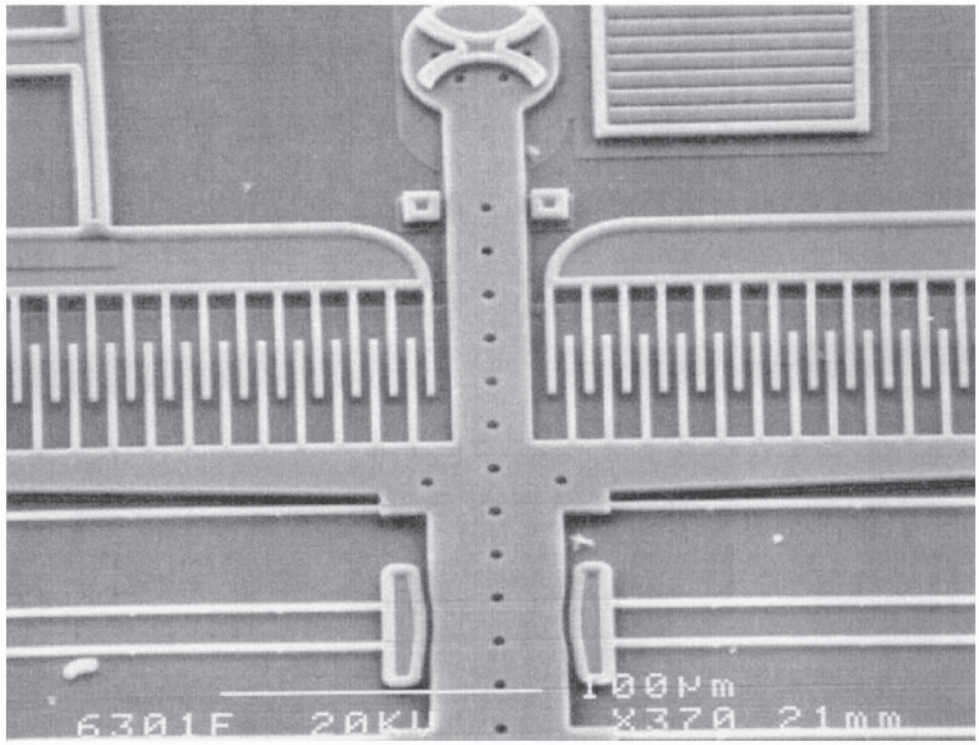
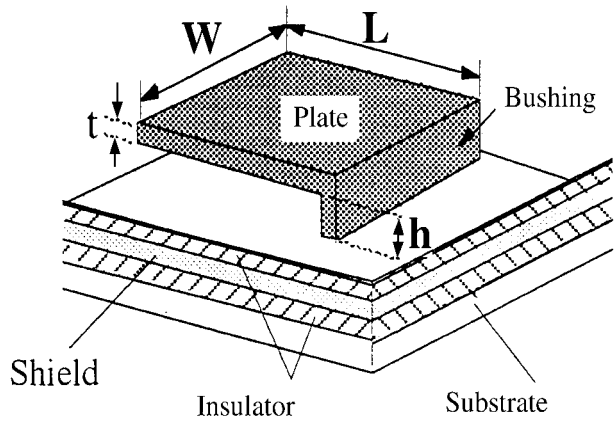
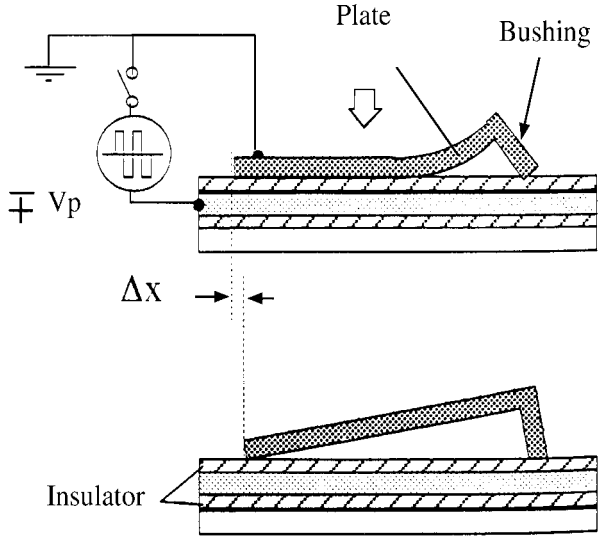


FIGURE 10.29(a) Electron micrograph of an interdigitated electrostatic drive. (Courtesy of Sandia National Laboratories' Intelligent Micromachine Initiative; www.mdl.sandia.gov/Micromachine. With permission.)



(b)



(c)

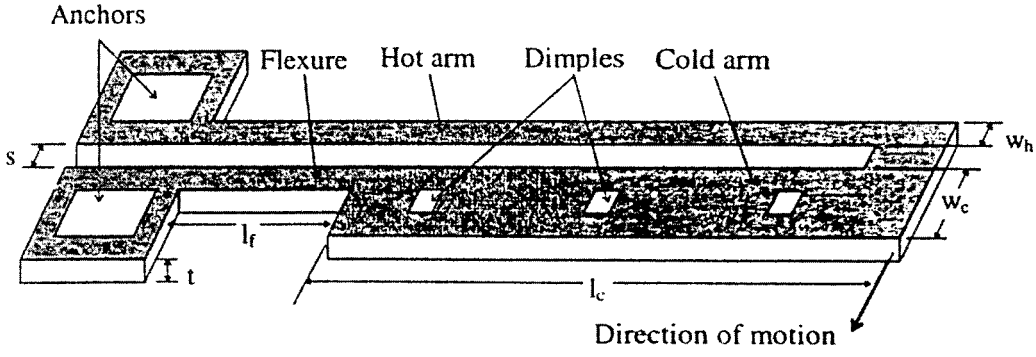


FIGURE 10.29(b-c) (b) Scratch drive actuator. (Fukuta, Y. et al., in *Proceedings of the 10th Annual International Workshop on Micro Electro Mechanical Systems*, Nagoya, Japan, IEEE, New Jersey, 1997, 477. With permission.); (c) schematic diagram of the lateral thermal actuator. Typical dimensions are given in the text. (Comtois, J. H., and Bright, V. M., in *Digest of Technical Papers, Solid-State Sensor and Actuator Workshop*, Hilton Head, South Carolina, June 1996, 152. With permission.)

Thermal Actuator

Thermal actuator arrays for positioning surface micromachined elements have been demonstrated, as well as automated assembly of polysilicon mirrors and other elements with thermal actuators.[154–156] The thermal actuator was designed for vertical and horizontal motions. The horizontal actuator is shown schematically in Fig. 10.29(c). These structures were fabricated with MUMPS processes.[145] It comprises a hot and a cold arm of polysilicon. Initially, the components are on the surface of the substrate; however when current is passed, one side becomes hotter than the other. The deflection of up to 16 μm is produced at moderate power levels of 16 mW and forces of 7 μN. In a second mode of operation if the actuator is heated above that for maximum deflection the hot arm becomes shorter than before and a negative deflection results with power off condition (see Fig. 10.29). The vertical actuator consists of two polysilicon beams separated by a 0.75-μm air gap. The lower arm is wider than the top one. When current is applied, the upper arm becomes hotter, providing higher electrical and higher thermal resistance, and thus a higher temperature driving the arm downward toward the substrate. Back-bending of the vertical actuator is particularly useful for clamping components in automatic assembly operations without the continuous application of current. These actuators are suitable for forming arrays of devices; designs of linear motors have been described. A self-engaging locking mechanism is also described which takes advantage of a tether from the upper polysilicon layer interacting with a key hole on a movable plate. When the movable plate is rotated out of the plane of the wafer, the tether slides into the wide section of the opening, which is wider than the tether. For example, the assembly of a polysilicon mirror 104 × 108 μm is achieved with two vertical actuators and a linear motor. Bending of the actuators is achieved with current levels of 4.2 mA at a voltage of ~14 V and the linear motor is operated with 24 mA at 5.5 V.[156]

Magnetically Driven Actuators

Magnetically driven micromirrors have been developed by Shen et al.[157] and one key advantage of magnetic drive is that actuation can be achieved in two directions by reversing the current flow, unlike electrostatic drive. A CMOS-compatible process was utilized to fabricate a suspended plate approximately 200 × 200 μm on a side. The deflection of the plate was ±1.5° with a single suspension; however, with multiple suspension arms, up to ±27° was observed at a drive current of 20 mA. Other work on magnetically driven mirrors was discussed in the previous section.

10.10 Electronics

For electronic applications of MEMS, the compatibility of the micromachining processes with IC processing is key for integration with active electronic components. There has been considerable work on fabrication of passive components by micromachining, specifically capacitors, inductors, and microwave transmission lines, and other components. The key advantages for passive component integration are ease of manufacturability for the higher frequency range of 100 to 1000 GHz where characteristic dimensions are mm to sub-mm range compatible with micromachining. This offers the opportunity to fabricate components and packaging in an integrated approach. Applications include test instruments, communications systems, radar, and others.

RF and Microwave Passive Components

Large suspended inductors were fabricated by Chang et al.[158] to demonstrate the integration of such components with electronic circuits. A 100-nH spiral with a self-resonance at 3 GHz integrated with a balanced cascade tuned amplifier with gain of 14 dB centered at 770 MHz, implemented with standard digital 2-μm CMOS process. The amplifier had a noise figure of 6 dB and a power dissipation of 7 mW, operated from a 3-V supply.

A 1-GHz CMOS RF front end has been demonstrated by Rofougaran et al.[159] The application is for direct conversion wireless receiver, or zero RF, and frequency shift-keying receivers. The building blocks

TABLE 10.6 Microrelays

Application	Fabrication Process	Drive	Contact On-Resistance	Maximum Current	Off- Resistance/ Breakdown voltage	Switching Time	Insertion Loss	Ref.
			Electrostatic					
Automated test equipment	Bulk micromachining and anodic bonding	<100V	<3 Ω	—	—	<20 μs		172
Switching	CMOS compatible	1-10V with DC bias of 30-54 V	—	—	—	<1 ms		176
RF to microwave	Surface micromachining	28V at >50 nA	~0.22 Ω	200 mA	—	—	0.1 db at 4 GHz	173
RF to microwave	Surface micromachining on GaAs	~30 V	—	—	—	—	0.3 dB at 20 GHz	174
Switching	Electroplated metal films	24 V	0.05 Ω (initial)	5 mA (single contact); 150 mA (multiple contacts)	>100 V	—	—	170–171
Small-signal RF	Surface micromachining	20–100 V [10 μW]	10–80 Ω	1 mA	—	2.6– 20μs	-	175
			Thermal					
Switching	MUMPS	7-12 V	2.4 Ω	80 mA	—	—	—	178
RF impedance matching	Surface micromachining in polysilicon	12 mW	2.1–35.6 Ω	>1mA	$>10^{12}$ Ω 400V	<0.5 ms	—	177
			Magnetic					
Electrical control circuits	Polyimide mold and electroplated metals	180 mA (33 mW)	0.022 Ω	1.2 A	—	0.5-2.5 ms	—	179

consist of low-noise RF amplifier, down-conversion mixer, and the contact pad is modified to reduce electric noise by connecting metal layer-1 to RF ground and metal-2 to RF input. The MEMS aspects are in the tuned amplifier's 50-nH spiral inductor, which would normally self-resonate at 700 MHz due to capacitive coupling to the substrate through the 1-μm field oxide, so that the use of standard CMOS inductors is generally limited to 5 to 10 nH. In this work, a gas-phase isotropic etch provides removal of the substrate from under the coil, allowing 50 nH to be realized.

López-Villegas et al.[160] have studied integrated RF passive components fully integrated with CMOS processing. Spiral inductors of 10 to 20 turns with 10-μm line width were characterized over the frequency range 50 MHz to 40 GHz. The influence of the material was key to defining the self-resonance of the structure, typically 1 to 6 GHz. Interdigitated capacitors were also fabricated with values of 0.5 to 1.35 pF and had self-resonances of 6 to 7 GHz.

Microwave Waveguides

Rectangular waveguides have been fabricated by McGrath et al.[161] by a bulk micromachining process and characterized over the frequency range 75 to 110 GHz. Slots are formed in a (110) silicon wafer, which were subsequently coated with 250 Å Cr and 5000 Å Au to form a plating base for a further 3 μm of electroplated Au. Losses measured in a 2.5-cm guide were comparable to commercial waveguides at about 0.024 dB/m. Active and passive components could be integrated into these structures.

Circuit components have also been fabricated by bulk micromachining with the added advantage of an integrated package by Franklin-Drayton et al.[162,163] and Katehi et al.[164] A series open-end tuning stub and a stepped impedance low-pass filter were realized for the frequency range 10 to 40 GHz. The method of design is based upon a quasi-static model utilizing TEM or quasi-TEM approximation, following this with a finite difference time-domain technique to evaluate the performance. The mesh was carefully selected to reduce truncation errors and grid dispersion errors, typically less than 1/20 of the shortest wavelength. Electrical conductors are assumed perfect conductors and, at dielectric interfaces, the average of the two permitivities was taken. Metallization comprised Ti/Au/Ti with subsequent electroplating to a final thickness of 3 μm Au. Alignment between cavities and waveguide structures was achieved via windows etched through the wafer thickness. Figure 10.30(a) shows a five-section, stepped impedance low-pass filter in which the 100- and 20-ohm impedance steps are formed by conductor widths of 20 μm and 380 μm, with slot widths of 210 μm and 30 μm, respectively. Figure 10.30(b) shows the integrated packaging topology and micrographs of the fabricated structures.

Microwave transmission lines have also been fabricated by Milanovíc et al.[165] with a commercial CMOS process with post-processing micromachining. The transmission lines were designed to operate in TEM mode

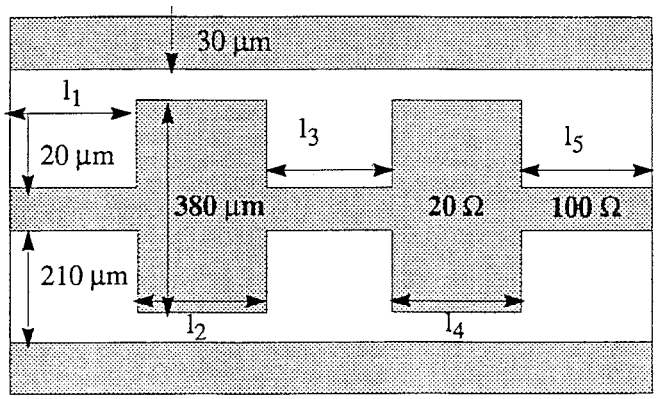


FIGURE 10.30(a) Integrated packaging. (Drayton, R. F. et al., *IEEE Trans. Microwave Theory and Tech.*, 46, 900, 1998. With permission.)

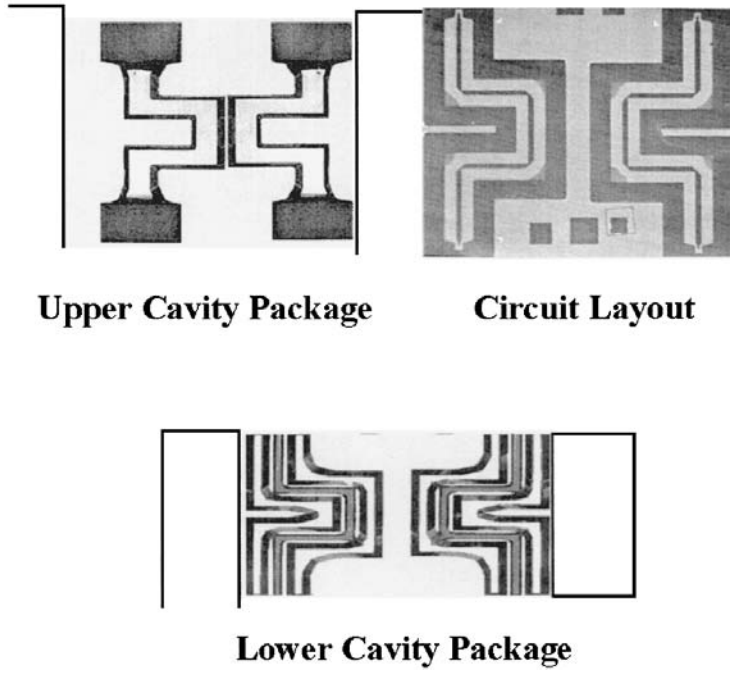


FIGURE 10.30(b) Microfabricated two-stage coupler. Chip measures 7.6 × 10 cm. Upper cavity shows probe windows as dark region. Conductor is dark region in circuit layout. (Franklin-Drayton, R. et al., *The International Journal of Microcircuits and Electronic Packaging*, 18, 19, 1995. With permission.)

with 50 and 120 Ω nominal characteristic impedance with standard layout tools. The post-processing etch was used to remove the silicon from underneath the conductive aluminum transmission lines to lower the losses. Figure 10.30(c) shows the simplified layout of the co-planar waveguides where the open areas are shaded in black. The open areas are first etched with a xenon difluoride, followed by anisotropic chemical etching with EDP. The cavities connect beneath the aluminum conductors, but enough material remains for mechanical stability. A fully formed trench also lowers electromagnetic coupling to the substrate. Measurements for test chip with three different lengths, 0.8 to 3.7 mm, with open and short stubs were made between 1 and 40 GHz. Insertion loss was calculated based on transmission line measurements, as shown in Fig. 10.30(d).

Tuning Fork Oscillator

A tuning fork-based oscillator has been fabricated by surface micromachining in polysilicon with integrated electronics by Roessig et al.[166] The device has an output frequency of 1.0022 MHz and exhibits a noise floor of –88 dBc/Hz at a distance of 500 Hz from the carrier. Previous surface micromachined oscillators used resistors to detect motion — which limits the noise floor. In this device, the capacitive detection is employed and, hence, the large impedance at the sensing node introduces a smaller input current noise than the resistive method. Figure 10.31 shows the integrated device with a double-ended tuning fork with tines 2 μm wide, 60 μm long, and 2 μm thick. These are fabricated in an etched well prior to the CMOS circuit, as discussed in the Section 10.6.

Thermal Devices

A great deal of progress has been made in the integration of thermal sensors, infrared sensors, and gas flow sensors with on-chip CMOS electronics. Baltes et al.[167] describe the fabrication and operation of a thermoelectric air-flow sensor and an infrared sensor, in addition to measurements of the thermophysical properties of material components of CMOS electronics.

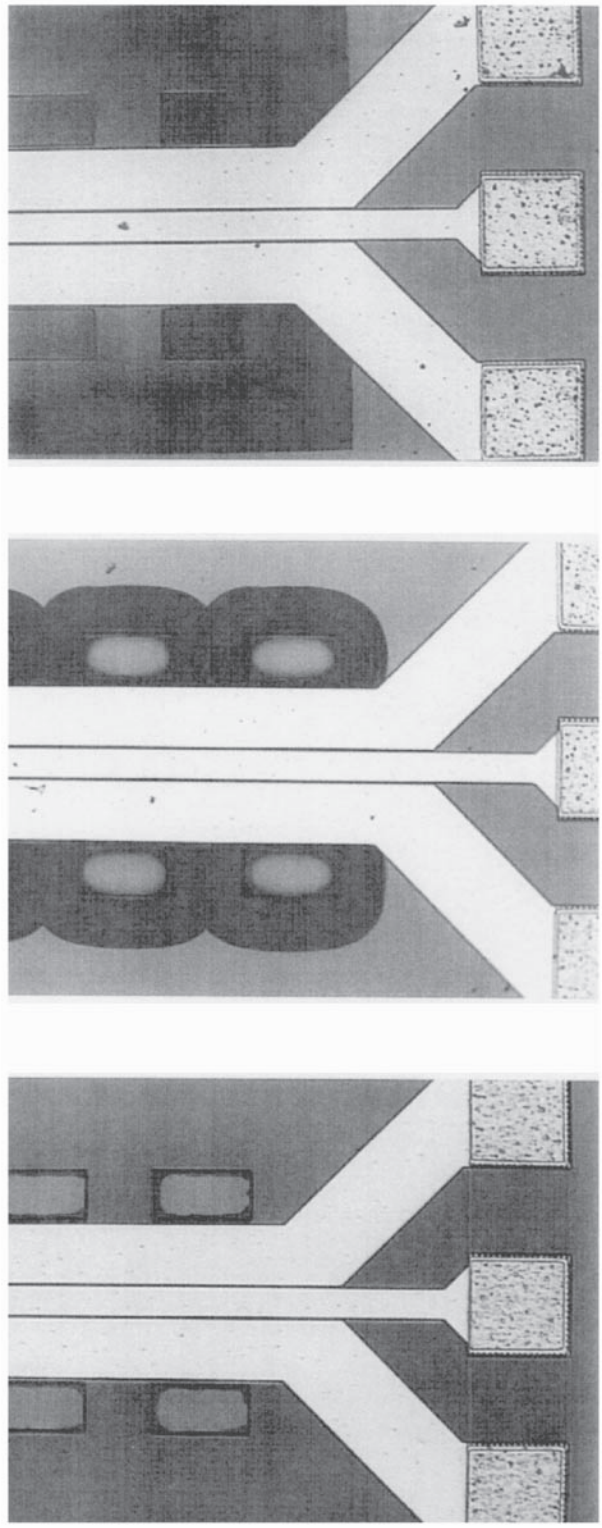


FIGURE 10.30(c) SEM micrograph of the 50 Ω transmission lines, of width 130 μm. (a) after CMOS fabrication; (b) after isotropic etch; and (c) after combined etch. (Milanovic, V. et al., *IEEE Trans. Microwave Theory and Techniq.*, 45, 630 1997. With permission.)

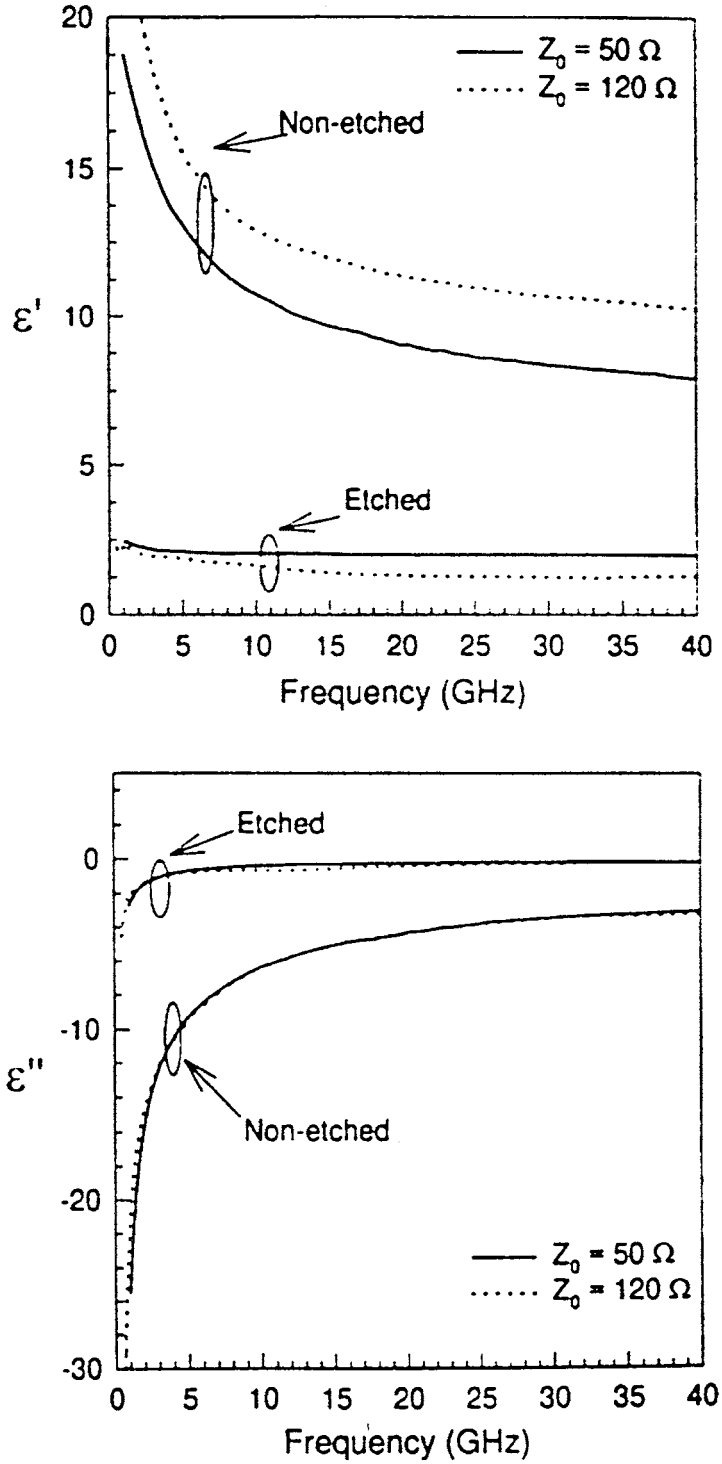


FIGURE 10.30(d) Measured effective dielectric constant of transmission lines before and after etching. (Milanovic, V. et al., *IEEE Trans. Microwave Theory and Techniq.*, 45, 630 1997. With permission.)

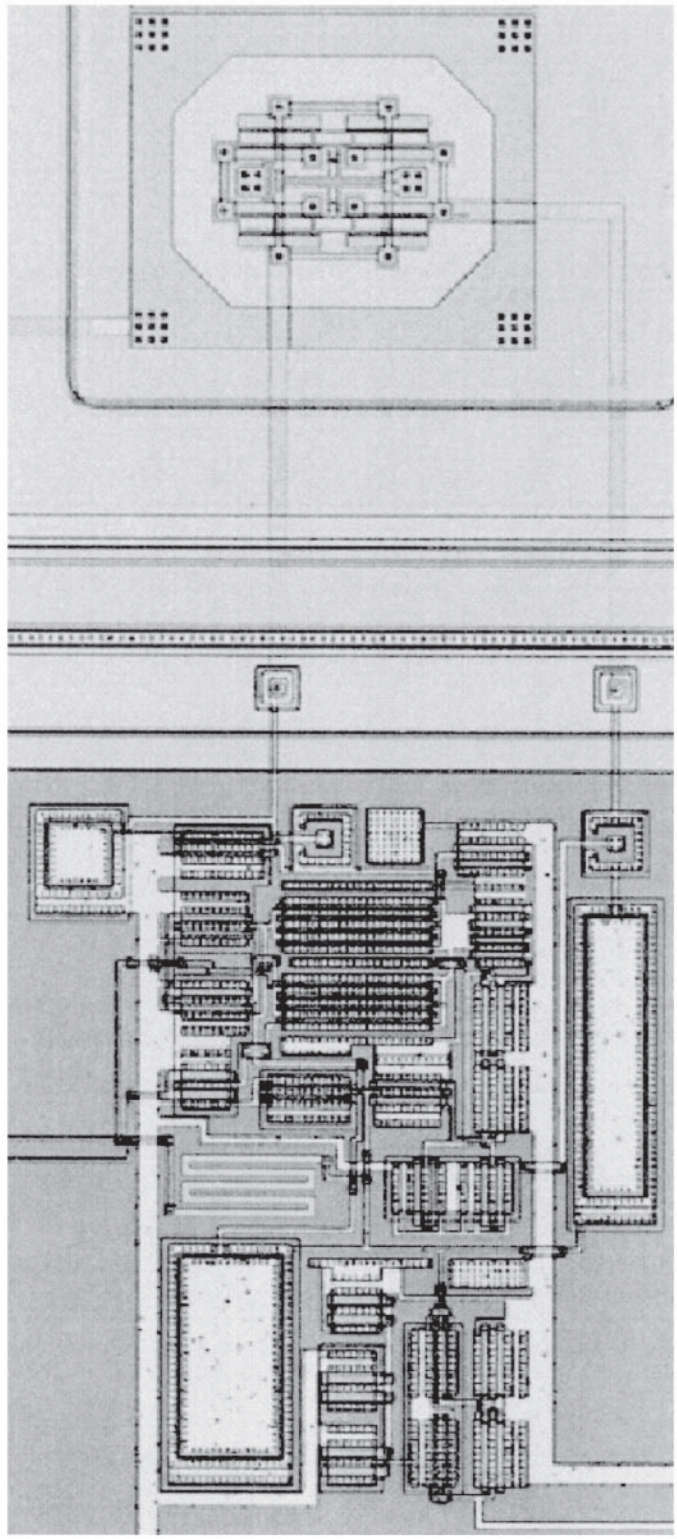


FIGURE 10.31 Tuning fork oscillator integrated with CMOS electronics. (Courtesy of Sandia National Laboratories' Intelligent Micromachine Initiative; www.mdl.sandia.gov/Micromachine and T. Roessig U.C. Berkeley. With permission.)

Bandgap Voltage Reference

Reay et al.[168] have demonstrated thermally isolated bandgap voltage reference temperature sensors. The reference has a 5300°C/W thermal resistance isolation from the substrate silicon, a 2.5-ms thermal time constant, and uses 1.5 mW at 25°C ambient temperature. This regulation achieved a reduction of the temperature coefficient from 400 to 9 ppm/°C for an ambient temperature range of 0 to 80°C. A schematic of the circuit is shown in Fig. 10.32(a). The servo-amplifier adjusts the reference voltage until the currents in the two branches are equal and thus generate the bandgap voltage.

RMS Converter

Measurement of true RMS voltage is complicated by the fact that the waveform shape is important and the peak value is only utilized the waveform shape must be known. Commercial true RMS meters utilize thermal heat to evaluate the power in the signal with specialized components. Klassen et al.[169] has developed a CMOS fabrication process in which a suspended, thermally isolated platform is utilized for this purpose. On the platform, a resistive heater and diode-connected vertical BJT are formed; temperature sensitivity of 2 mV/K. The beams, 85 to 225 μm in length, are defined in an oxide-nitride diaphragm and undercut with a bulk-silicon etch in TMAH, as shown in Fig. 10.32(b). The beams had a maximum thermal resistance of 37,000 K/W in air. The cascade CMOS operational amplifier, followed by a source follower to provide up to 50 mA of current for the heating element, operated from a 5-V supply. The quiescent power consumption of the amplifier was 950 μW and the –3-dB frequency was 415 MHz. With a sinusoidal input signal at 1 kHz, the measured dynamic range of the system was from 2.4 mV to 1.1 V r.m.s. (53 dB). Measurements of nonlinearity were 4% compared to a Hewlett-Packard HP3478A, which had a specified nonlinearity of 0.2% for low-frequency inputs.

Microrelays

An important illustrative example of MEMS process integration in which electronic and mechanical function are combined is the microrelay. There has been considerable interest in relays and switches for high-impedance isolation of circuit components, and for RF and microwave switching. There is insufficient space in this chapter to give a comprehensive overview of these activities; however, Table 10.6 summarizes some of the work that has been directed toward the success of these microdevices — grouped by actuation method. These devices typically have lifetimes of greater than 10^6 cycles. Zavracky et al.[170] and McGruer et al.[171] have built electrostatic relays with multiple contacts to increase the maximum current-carrying capacity. Micrographs of the electroplated thick film of the relay are shown in Fig. 10.33. Other electrostatic designs demonstrate low power consumption, [172–175] and CMOS-compatible microrelays have been demonstrated by Grétillat et al.[176] Novel latching a surface micromachined devices have been demonstrated and an example[177] is shown in Fig. 10.34(a).[178] Alternative actuation schemes are thermal and magnetic. Finite element modeling of the actuator and the magnetic circuit has been carried out by Taylor et al.[179] to provide improved design methods for these devices (see Fig. 10.34(b)). The thickness of the permalloy layer must be large enough to avoid saturation of the magnetic field. Minimum switching current and optimum coil spacing for operation at under 100 mA were achieved in these devices. The hold force is high — to provide low contact resistance. Contact resistance is a critical issue and has been studied in macroscopic relays by Holm[180] and Schimkat[181] with forces in the μN range.

Integrated Ink-Jet Printers

Integration of fluidic elements with a drive circuit on a single chip have been demonstrated by Krause et al.[182] Figure 10.35 shows the integrated structure. The fluid cavity is bulk micromachined from the back of the wafer, while the (2 μm-process) CMOS electronics are fabricated on the front side. The devices were fabricated on 4 in. dia. p-type 30-50 Ω-cm, 400 μm thick (110) silicon to achieve high aspect ratio slots. The heating elements for bubble formation are integrated into the structure with 30 μJ required to eject each ink droplet. The heaters have a possible output of 1 GW/m^2, but in this

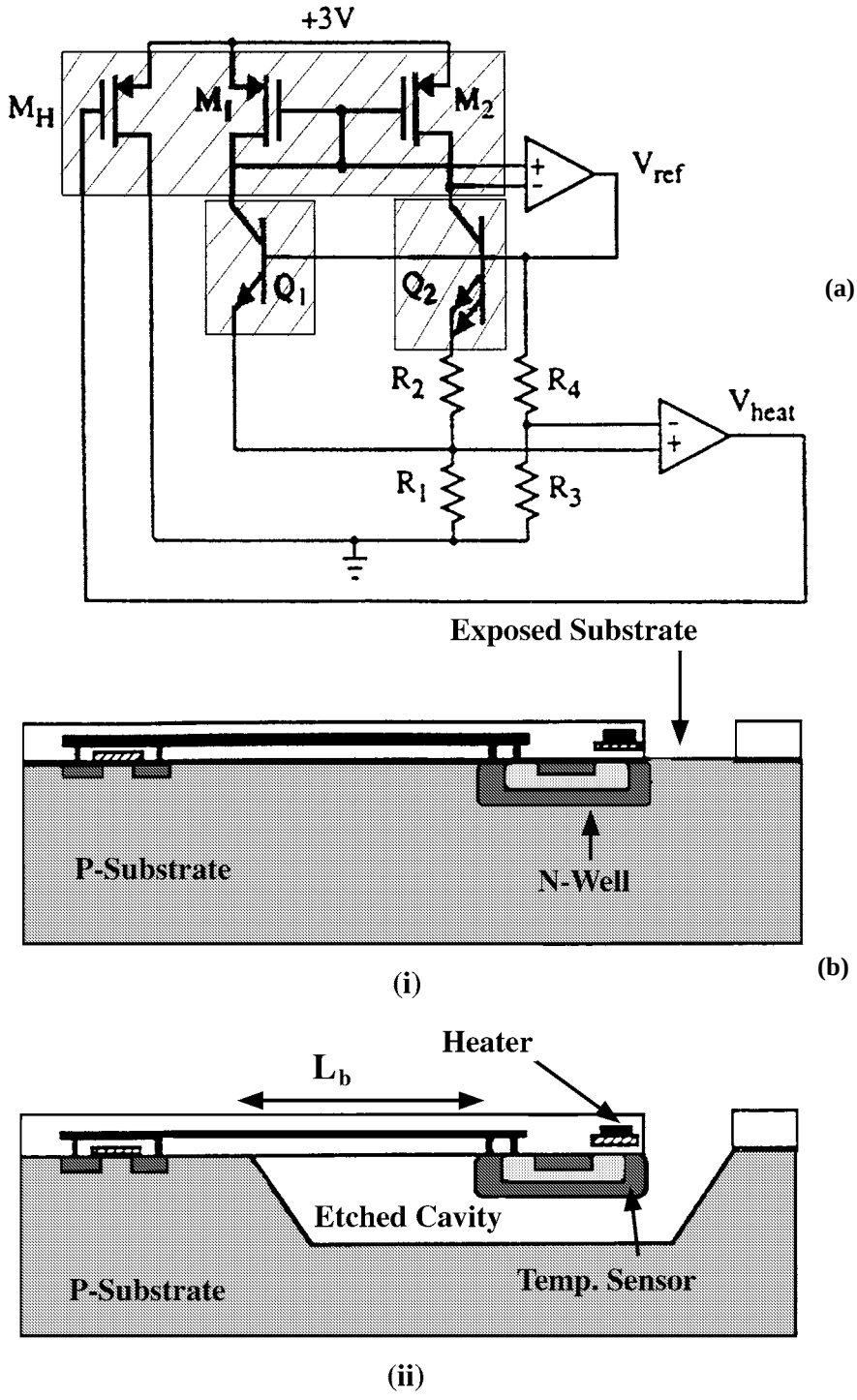
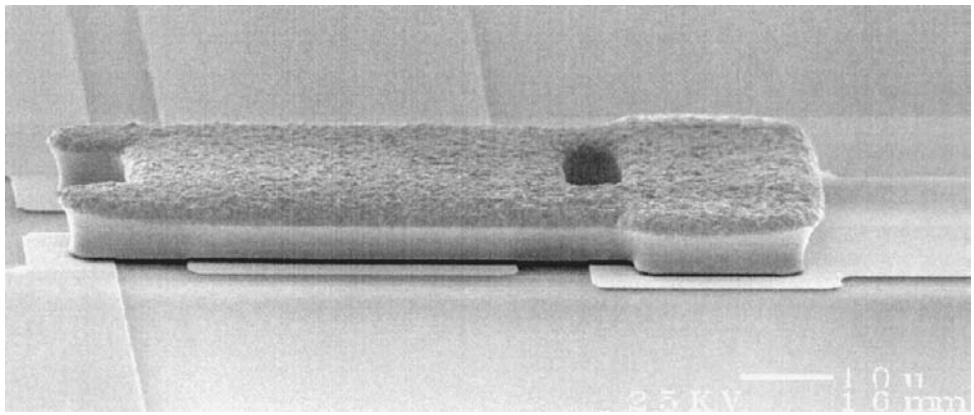
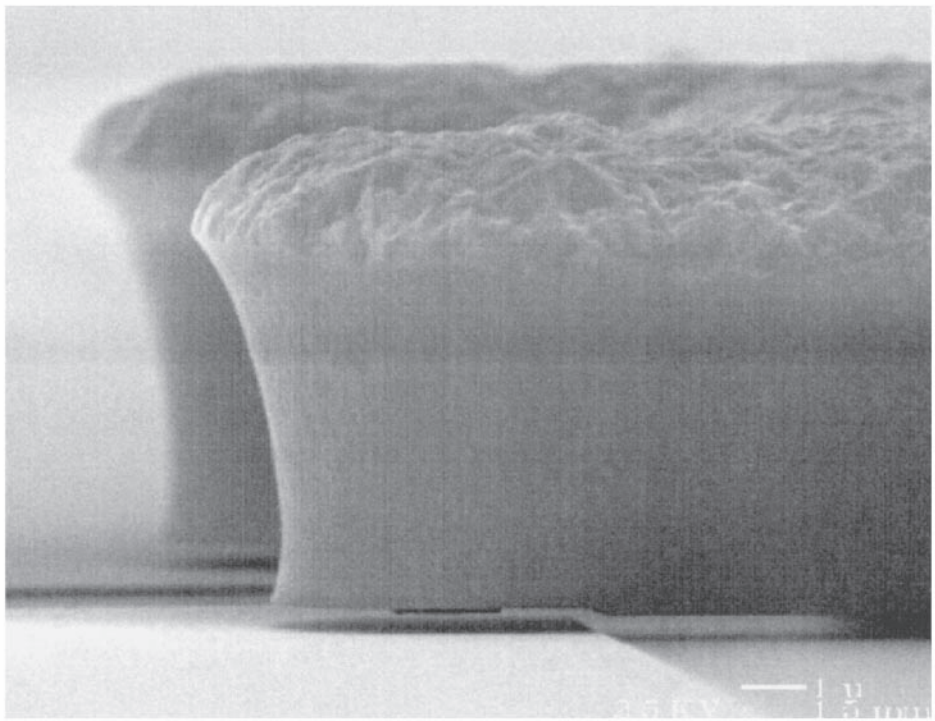


FIGURE 10.32 (a) Schematic diagram of a complete bandgap reference showing the PMOS heating transistors and thermal control loop. The shaded regions are separated thermally isolated n-wells; (b) cross-sectional view of a thermoelement for an RMS converter, at different stages in the fabrication process: (i) upon completion of the CMOS process, and (ii) after the post processing step of etching in TMAH. (Reay, R. J., et al., *IEEE J. Solid-State Circuits*, 30,1374, 1995. With permission.)



(a)



(b)

FIGURE 10.33 (a) Micrograph of an electrostatically actuated gold metal microrelay; (b) close up view of the contact area. (McGruer, N. E. et al., in *Digest of Technical Papers, Solid-State Sensor and Actuator Workshop*, Hilton Head, South Carolina, June 1998, 132. With permission.)

application, generate 7.5 W in a 50-nozzle array firing at 5 kHz. The chip is covered with an electroplated layer, shielding the device from mechanical, chemical, and electric damage comprising 4-μm nickel and 1 μm gold on a Ti/Cu adhesion layer.

10.11 Chemical Sensors

There has been great success in developing chemical sensors, however, one of the key stumbling blocks is the compatibility of chemically sensitive layers with IC processes, thus limiting the possibilities for

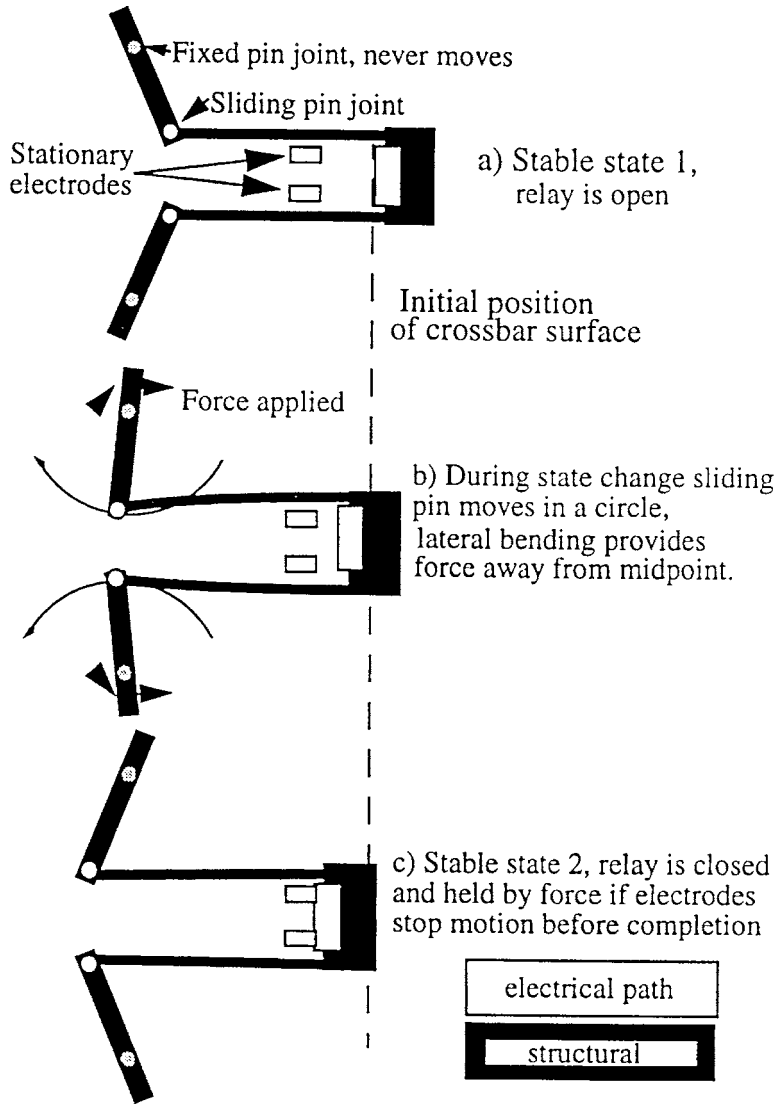


FIGURE 10.34(a) Bistable action in the relay frame hold the device in the open or closed state without actuation: (Kruglick, E. J. J. and Pister, K. S. J., in *Digest of Technical Papers, IEEE Solid-State Sensor and Actuator Workship*, Hilton Head, SC, 1998, 333. With permission.)

process integration. When one considers it the processing complexity required to integrate the electronics and chemically sensitive layers on chip is justified, there are a number critical issues:

- Is the cost per packaged functional sensor lowered by integration?
- Does the application requires integration (i.e., is small size essential?)?
- Is the sensing function improved by integration?

An excellent review of recent work on chemical sensors has been published by Janata et al.[183] Most work has focused on hybrid designs in which the electronics and chemical sensor arrays are fabricated separately and then interconnected. The work of Madou et al.[184] in a blood gas sensor for the measurement of pH, CO_2, and O_2 *in vivo* is an example of this approach. Here, the device was fabricated by bulk micromachining on a thin silicon piece approximately 350 μm wide and bonded to an associated interface circuit chip that was made at an IC foundry.

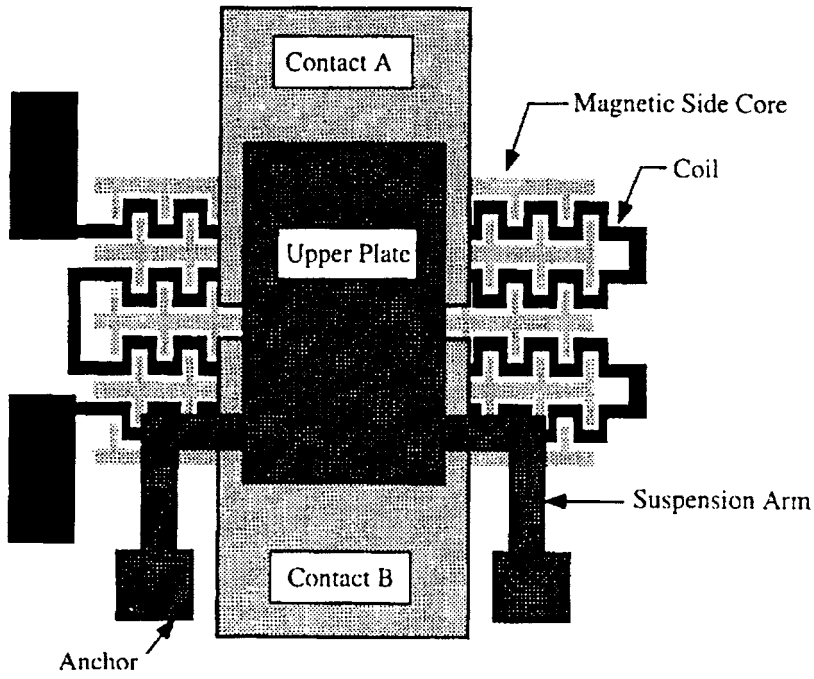


FIGURE 10.34(b) Schematic top view of the magnetic microrelay, illustrating the relative positions of the upper moveable plate, contact, side cores and coil. The conductor width in the coil is 80 μm. (Taylor, W. P. et al., *J. Microelectro. Mech. Syst.*, 7, 181, 1998. With permission.)

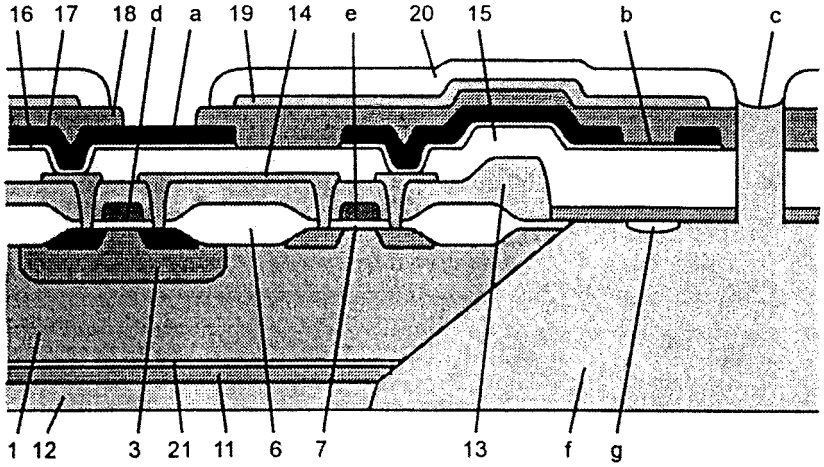


FIGURE 10.35 Structure of the backshooter microsystem ink-jet print head (not to scale) illustration: 1- substrate, 6 - field oxide, 7 - gate oxide, 11 - etch stop layer, 12 -PECVD SiO_2, 13- BSG, 14 - first Al layer, 15 - undoped silica glass, 16 - heater layer, 17 - second Al layer, 18 - thermal throttle layer PECVD Si_3N_4, 19 - galvanic adhesive layer Ti/Cu, 20 - carrier layer Ni/ASu, 21 - thermal SiO_2. Elements a - bond pad (Al), b - heating element, c - nozzle, d - p-MOS transistor, e - NMOS transistor, f - ink-jet chamber, g - vapor bubble formed. (Krause, P., et al., *Proceedings of Transducers 95*, Stolkholm, Sweden, 1995. With permission.)

ISFET

The chemically selective FET developed by Janata[185] and Bergveld[186] demonstrates specific analyte selectivity based on an FET structure with the gate coated with a chemically sensitive layer exposed to the solution. The sensing mechanism is based on a variety of surface-specific interactions.[187] These devices may be configured as gas-sensitive devices, for hydrogen detection,[188] ion-selective devices,[189] enzyme FETs, and most recently, suspended gate structures. Figure 10.36 shows a schematic diagram of an ISFET. The important characteristic of these sensors is that the gate potential and, hence, the channel threshold voltage are defined by the potential applied at the reference electrode and the interfacial potentials. This potential is related to the activities of participating ions rather than their concentrations. Hydrogen ion sensitivity is intrinsic to the dielectric material coating the gate. Bousse[190] has developed an OH site-binding theory to account for the pH dependence of the FET for oxide and nitride gates. The most stable gate dielectric choices are TaO_2 and Al_2O_3. Advanced concepts for back-side contact FETs are reviewed by Cane et al.[191] One of the advantages of ISFET technology is that it can be made compatible with CMOS processing. An integrated CHEMFET, demonstrated by Domanský et al.[192] is capable of measurement of work function and bulk resistance changes. Here, a carbon black/organic polymer composite film for the detection of solvents covers the gate, as shown in Fig. 10.36(b).

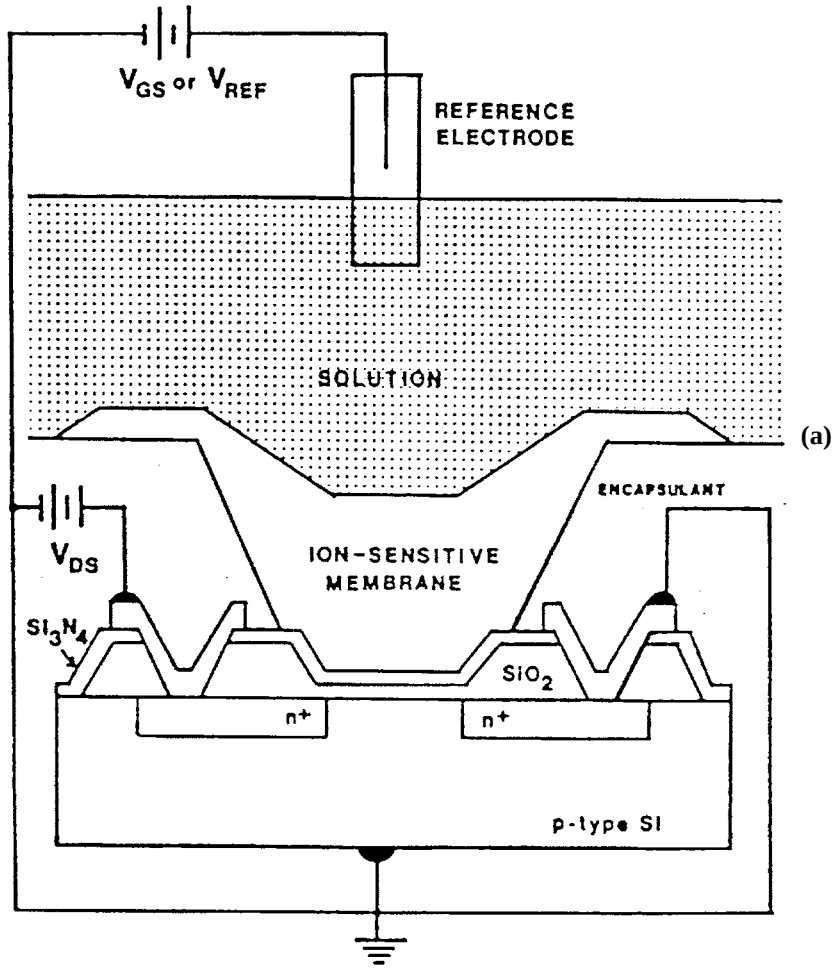


FIGURE 10.36(a) Schematic diagram of an ISFET. (Janata, J., in *Solid State Chemical Sensors*, J. Janata and R. J. Huber, Eds., Academic Press, New York, 1985. With permission.)

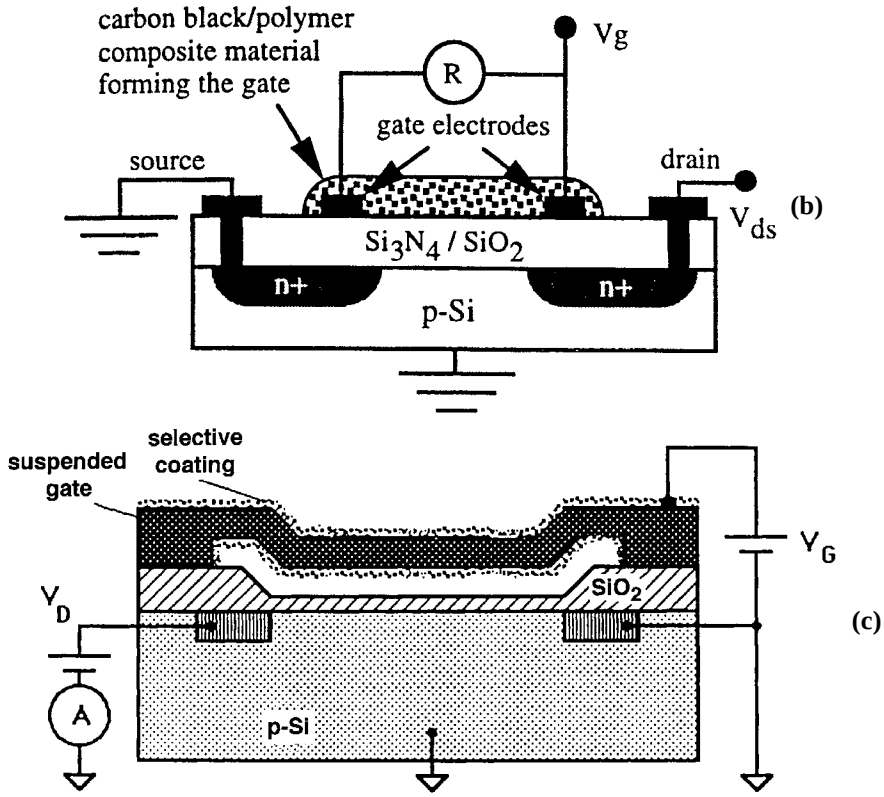
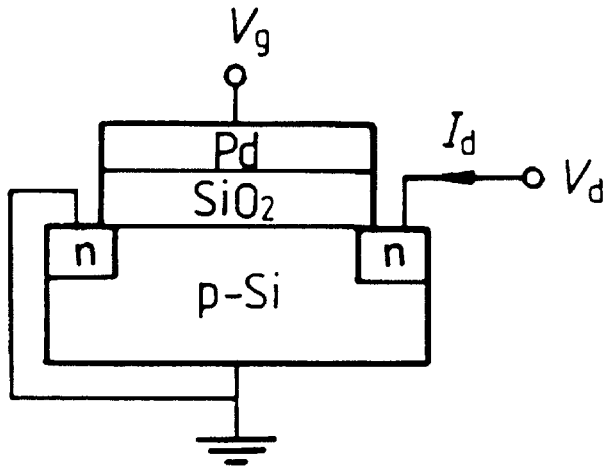


FIGURE 10.36(b-c) (b) Carbon black impregnated gate FET. (Domanský, K. et al., in *Digest of Technical Paper, IEEE Solid-State Sensors and Actuators Workshop,* Hilton Head, NC, 1998, 187. With permission.); and (c) suspended gate FET. (Mosowicz, M., and Janata, J., in *Chemical Sensor Technology,* T. Seiyama, Ed., Elsevier, New York, 1988. With permission.)

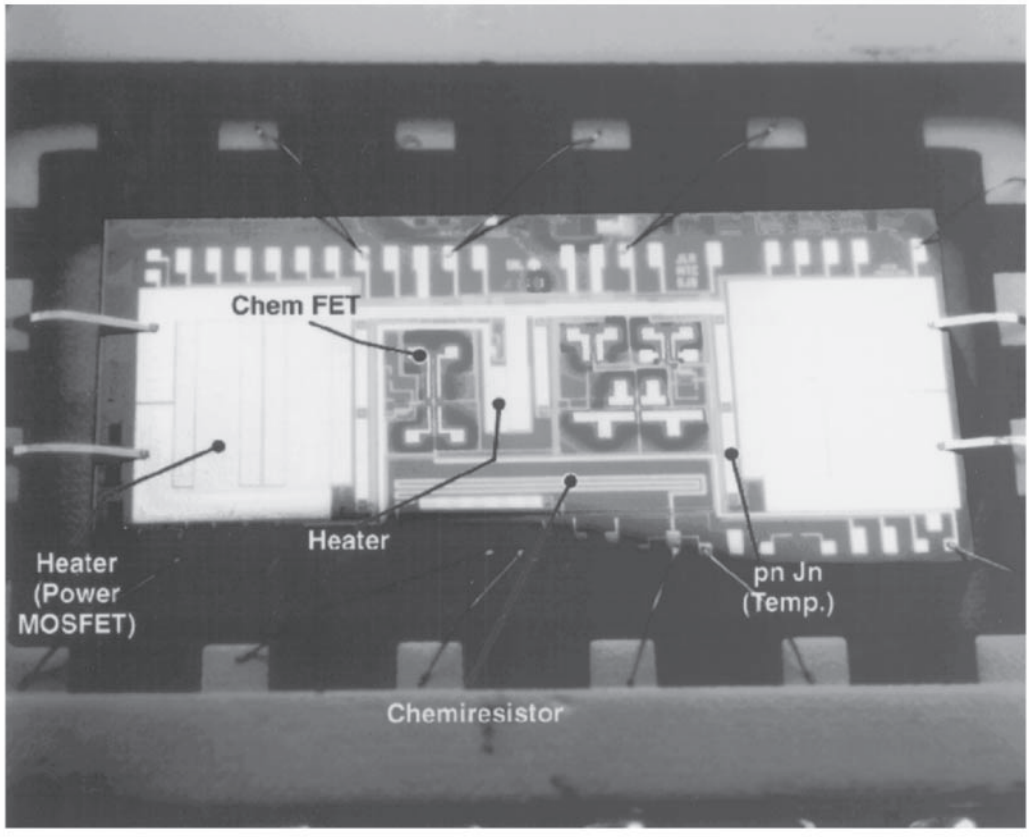
Chemically sensitive layers are, in general, not process compatible with CMOS circuit fabrication. Approaches that have been made in this area include fabrication of the electronics first, followed by deposition of the chemically sensitive membranes while the CMOS circuit is covered with a passivation coating.

Hydrogen Sensor

The Pd gate MOS hydrogen sensitive FET was invented by Lundström et al.[188] The device is shown schematically in Fig. 10.37(a). Upon exposure to hydrogen, dipoles are created at the SiO_2/Pd interface producing a shift in the threshold voltage. A hydrogen chemical sensor has been successfully integrated with electronics components at Sandia National Laboratories (Rodriguez et al.[193]). This sensor utilizes two Pd/Ni layers, one as a chemiresistor and the second as the gate of an FET. Figure 10.37(b) shows a picture of the sensor with integrated heaters and temperature sensors. The FET shown in Fig. 10.37(b) is more sensitive at low concentration ranges and has a logarithmic response, however, the conductimetric sensor has a square-root dependence on the hydrogen concentration. The sensors are operated at elevated temperature of approximately 100°C to increase the reaction kinetics and improve reversibility. Typical response data is shown in Fig. 10.37(c) with a 1% hydrogen concentration, resulting in a response time of a few seconds. The sensor combination has an exceptionally wide dynamic range of six orders of magnitude, and response time was within 5 seconds. Heating is achieved through two power transistors and temperature monitoring with an array of nine p/n junction diodes. Typical die size is 270 × 120 mils. Devices have been demonstrated with stabilities of over 60 days and show reversible behavior. Sensors are being commercialized for detection of hydrogen in aerospace applications. Advanced versions of this sensor have also been produced with fully

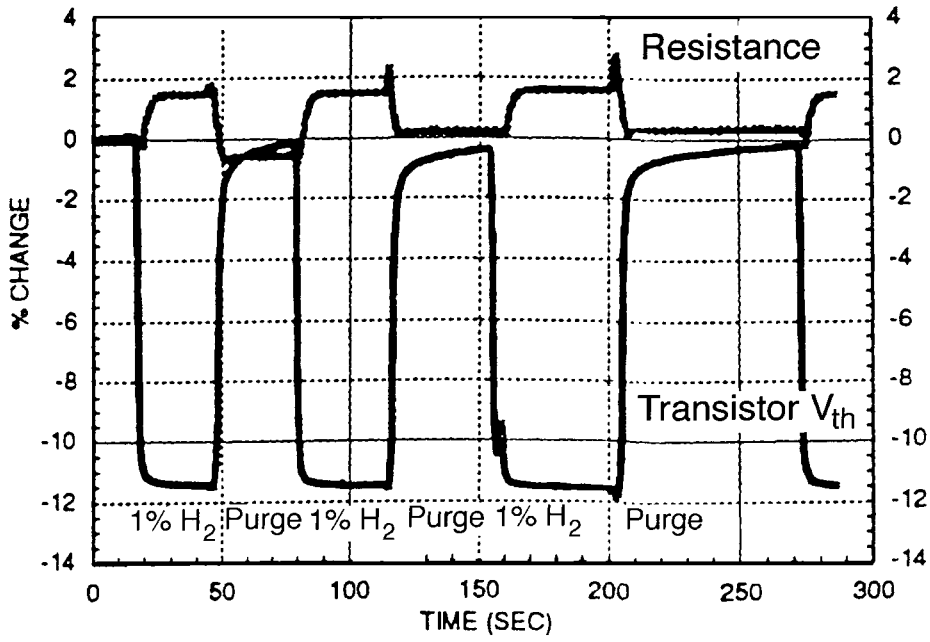


(a)



(b)

FIGURE 10.37(a-b) (a) Pd-gate FET. (Lundström, I., and Svensson, C., in *Solid State Chemical Sensors*, J. Janata and R. J. Huber eds., Academic Press, New York, 1985. With permission.); (b) Photograph of the robust hydrogen sensor with integrated temperature sensors, Pd gate FET, chemiresistor, and heater elements. (Rodriguez, J. L. et al., *IEDM Tech. Digest*, IEEE, San Francisco, CA, Dec. 1992, 521–524. With permission.)



(c)

FIGURE 10.37(c) Response of sensor to hydrogen. (Rodriguez, J. L. et al., *IEDM Tech. Digest*, IEEE, San Francisco, CA, Dec. 1992, 521–524. With permission.)

integrated op-amps and control electronics, including analog capacitors, high-value polysilicon resistors, current mirrors, and operational amplifiers.

Gas Sensors

Microhotplates have been developed by Suehle et al.[194] for tin oxide chemical sensors. These devices are conductimetric sensors for reducing gases and operate at elevated temperatures, typically ~350°C. A suspended sandwich structure of CVD oxides, encapsulating a polysilicon heater, and integrated with an aluminum layer to provide thermal diffusion, is shown in Fig. 10.38. Post-processing was carried out after a standard CMOS process by EDP etching with added aluminum hydroxide to ensure passivation of any exposed Al conductors. Heating current (mA) was provided to the polysilicon layer and temperature sensing from van der Pauw aluminum layer with a temperature coefficient of resistance typically 0.003667/°C. The hotplates were effectively thermally isolated, showing efficiency of 8°C/mW in air, thermal time constant of 0.6 ms, and maximum operating temperature of 500°C. SnO_2 was deposited onto the hotplate by reactive sputter deposition in ultrahigh vacuum; and by heating the platform during deposition, selective control of the local material properties was achieved (such as grain size and the conductivity). After deposition, annealing was also carried out selectively *in situ* on the hotplates. The selectivity of these devices can be further modified by addition of catalytic metals such as Pt, Pd, or Ir. Semancik and Cavicchi[195] have demonstrated kinetic sensing on microhotplates by modulation of the sensor temperature to enhance analyte discrimination. Microhotplates were also fabricated with tungsten metallization so they could operate up to 800°C. The response of a Pt-doped SnO_2 sensor operating at 130°C to CO gas is shown in Fig. 10.38(b). The stability of high-temperature micromachined TiO_x gas sensors has been investigated by Patel et al.[196] for measurements of hydrogen and propylene in the presence of oxygen. The temperature played a key role in defining the sensor response to hydrogen at 100°C and propylene when above 350°C.

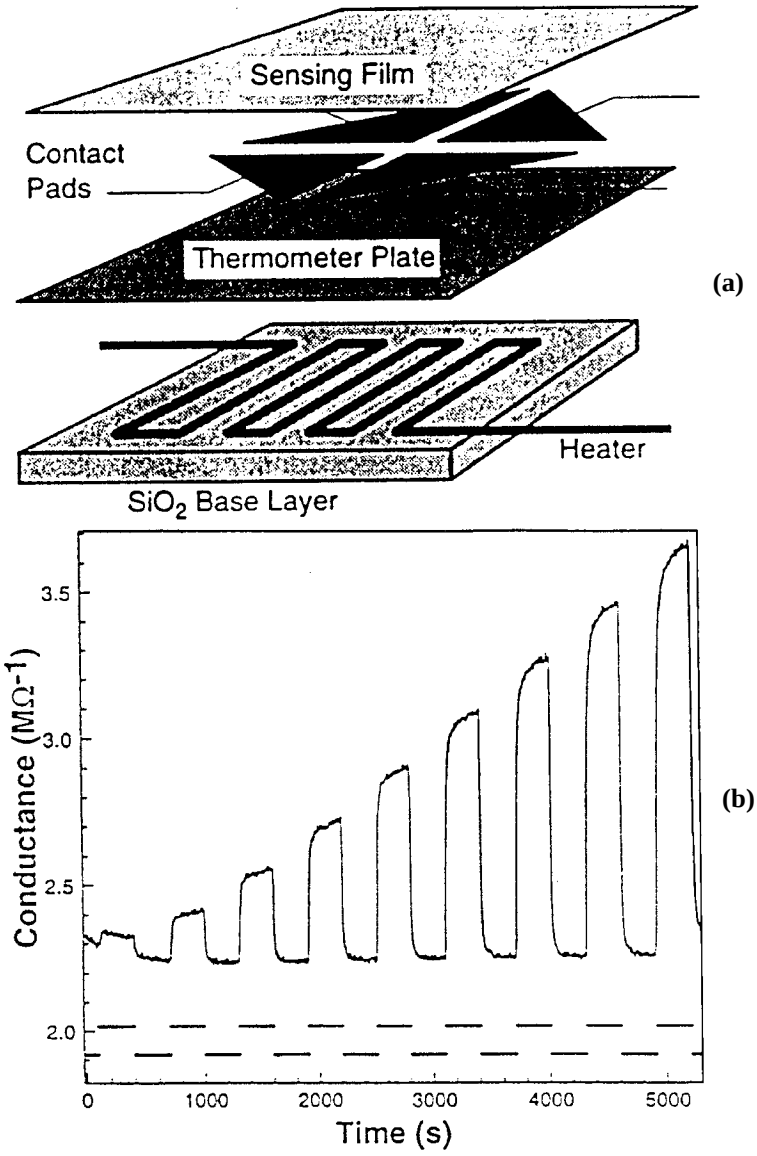


FIGURE 10.38 (a) Schematic diagram of a single micro-hot plate and functional cross-section of component parts; (b) static response at 130°C of a Pt/SnO_2 microsensor to on/off CO exposures, into dry air, of increasing concentrations from 5 to 45 ppm. (Suehle, J. S. et al., *IEEE Elec. Dev. Lett.*, 14, 118 1993. With permission.)

Artifical Nose

Microfabrication technology lends itself to the construction of arrays of sensors with differing chemical selectivities. Capacitive-based gas sensors having selectivity to different classes of chemical species[197] along with pattern recognition[198] have been demonstrated as a viable scheme for the realization of the electronic nose. The polymer coatings produce characteristic dielectric constant, mass, or conductivity changes when the analyte is adsorbed. This work on chemiresistor arrays has also been integrated with CMOS interface circuits for applications in food quality and odor identification.

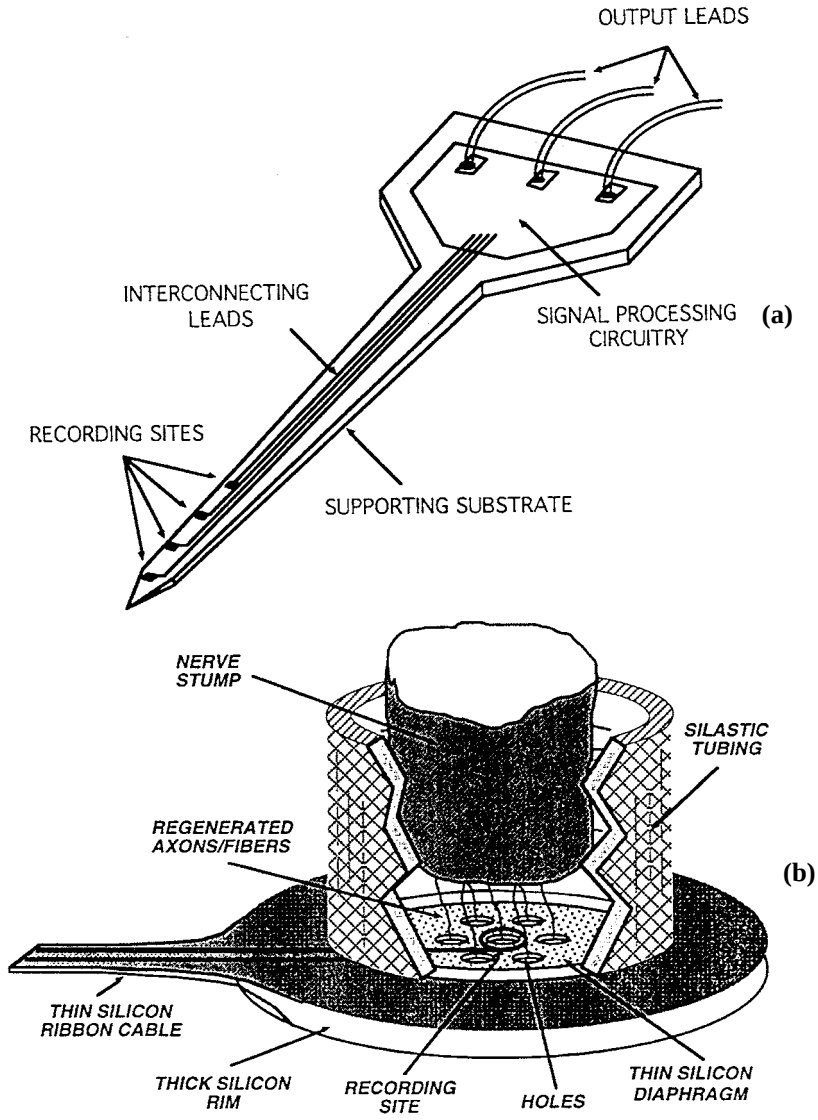


FIGURE 10.39 (a) Schematic diagram of boron doped etch stopped neural probe. (Najafi, K., *Handbook of Microlithogrpahy, Micromachining, and Microfabrication, Vol. 2: Micromachining and Microfabrication*, Ed. P. Rai-Choudhury, SPIE, Washington, 1997, 517. With permission.); (b) schematic diagram of the neural interface structure. (Akin, T. and Najafi, K., *IEEE Trans. Biomed. Eng.*, 41, 305, 1994. With permission.)

Neural Probes

Najafi[199] has reviewed his extensive work on neural probes with integrated electronics. The early design involved four masks and had a high yield. A boron diffusion defined the thickness of the structure. Three-dimensional multielectrode systems were later developed to improve electrode positioning. Each array of neural probes is inserted into a silicon machined substrate and electrical connections are made between the probe and support chip by electroplating nickel. On the chip, preamplifiers are followed by analog multiplexers and a broad-band output buffer to drive the external data line. Later developments included a NMOS and CMOS integrated circuit with ten recording sites of gold electrodes on 100-μm centers. The circuit operated with a 5-V supply and consumes 5 mW. A 32-electrode version also has an integrated multiplexing for 32-to-8 switching array. Preamp specifications were 150 to 300 V/V, –3 dB bandwidth

100 Hz to 10 kHz. Akin and Najafi[200] have developed novel sieve structures for stimulation electrodes. They include a silicon ribbon cable that allows connections with minimal mechanical hindrance of the implant while maintaining electrical connections. Neural probes for recording brain activity have also been fabricated by Kewley et al.,[201] with integration of the buffer electronics with the probe tip electrode. The advantage of a dry-etch process is a small, well-defined tip radius of 0.25 μm in this case. He integrated 18-channel preamplifiers in a MOSIS 2-μm, low-noise analog process, each having a total gain of 150 V/V. Probe tips of iridium were fabricated with PECVD layers of silicon nitride low-stress material over the electrodes, achieving a parasitic capacitance of 20 pF and an electrode capacitance of 40 pF; stable, low-leakage current of less than 0.25 pA at a 5-V bias in buffered saline solutions, in addition to maintaining a well-adhered film necessary to maintain the tip electrode integrity.

References

1. Biannual meetings in June: (1) International Meeting on Transducers; (2) Sensors and Actuators Workshop held at Hilton Head, South Carolina.

2a. *Institute of Electrical and Electronic Engineers (IEEE) MEMS Conference* every year in Jan/Feb.

2b. Annual Meeting of the American Society of Mechanical Engineers (ASME).

3. *Proceedings of the Micro Total Analysis Systems Workshop*; most recent meeting held in Banff, Canada, Oct. 13-16th, 1998, published by Kluwer, 1998.
4. The Society for Photo and Instrumentation Engineers (SPIE), Bellingham, WA.
5. IEEE, IEDM Meeting, New York.
6. International Meeting on Chemical Sensors.
7. Fall and Spring Symposia at Meetings of The Electrochemical Society, Pennington, New Jersey.
8. Peterson, K. E., Silicon as a mechanical material, *Proceedings of the IEEE*, 70, 420, 1982.
9. Göepel, W. et al., *Sensors a Comprehensive Survey, Fundamentals and General Aspects*, John Wiley & Sons, New York, 1989.
10. Kovacs, G. T. A., *Micromachined Transducers Sourcebook*, McGraw-Hill, New York, 1998.
11. Madou, M. J., *Fundamentals of Microfabrication*, CRC Press, Boca Raton, FL, 1997.
12. Sze, S. M., *Semiconductor Sensors*, John Wiley & Sons, Sommerset, NJ, 1994.
13. Trimmer, W. S., *Micromechanics and MEMS: Classic and Seminal Papers to 1990*, IEEE Press, New York, 1997.
14. Middelhoek, S. and Audet, S. A., *Silicon Sensors*, Academic Press, Boston, MA, 1989.
15. Ristic, L. J., *Sensor Technology and Devices*, Artech House, London, 1994.
16. Gardner, J. W., *Microsensors: Principles and Applications*, John Wiley & Sons, Chichester, West Sussex, U.K., 1994.
17. Janata, J., *Principles of Chemical Sensors*, Plenum Press, New York, 1989.
18. Madou, J. M. and Morrison, J. R., *Solid-State Chemical Sensors*, Plenum Press, New York, 1989.
19. Moseley, P. T., Norris, J., and Williams, D. E., *Techniques and Mechanisms in Gas Sensing*, Adam Higler, New York, 1991.
20. Turner, A. P. F., Karube, I., and Wilson, G., *Biosensors Fundamentals and Applications*, Oxford University Press, New York, 1987.
21. Vossen, J. L. and Kern, W., *Thin Film Processes*, Academic Press, 1978.
22. Brannon, J., *Eximer Laser Ablation and Etching, AVS Monograph Series*, M-10, American Vacuum Society, New York, 1993.
23. Friedrich, C. R., Warrington, R., Bacher, W., Bauer, W., Coane, P. J., Göttert, J., Hanemann, T., Haußelt, J., Heckele, M., Knitter, R., Mohr, J., Piotter, V., Ritzhaupt-Kleissl, H.-J., and Ruprecht, R., High Aspect Ratio Processing, in *Handbook of Microlithogrpahy, Micromachining, and Microfabrication*, Vol. 2: Micromachining and Microfabrication, Ed. P. Rai-Choudhury, SPIE, Washington, 1997, 299.
24. Stewart, D. K. and Casey, J. D., Focused ion beams for micromachining and microchemistry, in *Handbook of Microlithography, Micromachining and Microfabrication*, Vol. 2: Micromachining and Microfabrication, Editor P. Rai-Choudhury, SPIE, 1997, 153.

25. Allen, D. M., *The Principles and Practice of Photochemical Machining and Photoetching*, Adam Hilger, Bristol, 1986.
26. Williams, K. R. and Muller, R. S., Etch rates for micromachining processing, *J. Microelectromech. Syst.*, 5, 256, 1996.
27. Kendall, D. L. and Shoultz, R. A., Wet chemical etching of Silicon and SiO_2 and ten challenges for micromachiners, in *Handbook of Microlithogrpahy, Micromachining, and Microfabrication*, vol. 2: *Micromachining and Microfabrication*, Ed. P. Rai-Choudhury, SPIE, Washington, 1997, 41.
28. Seidel, H., Cspregi, L., Heuberger, A., and Baumgartel, H., *J. Electrochem. Soc.*, 137, 3612, 1990; and Seidel, H., Cspregi, L., Heuberger, A., and Baumgartel, H., *J. Electrochem. Soc.*, 137, 3626, 1990.
29. Kendall, D. L., Vertical etching of silicon at a very high spect ratios, Ann. *Rev. Mater. Sci.*, 9, 373, 1979.
30. Palik, E. D., Glembocki, O. J., and Heard, J. I., Study of bias-dependent etching of Si in aqueous KOH, *J. Electrochem. Soc.*, 134, 404, 1987.
31. Hesketh, P. J., Ju, C., Gowda, S., Zanoria, E., and Danyluk, S., A surface free energy model of silicon anisotropic etching, *J. Electrochem. Soc.*, 140, 1080, 1993.
32. Allongue, P. V., Costa-Kiedling, and Gerishcher, H., Etching of silicon in NaOH solutions, *J. Electrochem. Soc.*, 134, 404, 1987.
33. Kovacs, G. T. A., Mauluf, N. I., and Petersen, K. E., Bulk micromachining of silicon, *Proc. IEEE*, 86, 1536, 1998.
34. *Proceedings of Workshop of Physical Chemistry of Wet Chemical Etching of Silicon*, Holten, The Netherlands, May, 1998.
35. Clark, L. D. and Edell, D. L., *Proceedings IEEE Micro Robots and Teleoperators Workshop*, Hyannis, MA, 1987.
36. Bean, K., Anisotropic etshing of silicon, *IEEE Trans. Electron. Dev.*, 25, 1185, 1978.
37. Glembocki, O. J., Palik, E. D., de Guel, G. R., and Kendall, D. L., Hydration model for the molarity dependence of the etch rate of si in aqueous alkali hydroxides, *J. Electrochem. Soc.*, 138, 1055, 1991.
38. Bressers, P. M. M. C., Kelly, J. J., Gardeniers, J. G. E., and Elwenspoek, M., Surface morphology of p-type (100) silicon etched in aqueous alkaline solutions, *J. Electrochem. Soc.*, 143, 1744, 1996.
39. Price, J. B., Anisotropic etching of silicon with KOH-H_2O-isopropyl alcohol, in *Semiconductor Silicon*, Eds. H. R. Huff and R. R. Burgess, Softbound Proceedings of the ECS, 1973, 339.
40. Clark, J. D., L., Lund, J. L., and Edell, D. J., Cesium hydroxide [CsOH]: a useful etchant for micromachining silicon, in *Technical Digest of Papers, Solid-State Sensor and Actuator Workshop*, Hilton Head, South Carolina, 1988, 5.
41. Ip Yam, J. D., Santiago-Aviles, J. J., and Zemel, J. N., An investigation of the anisotropic etchng of (100) silicon using cesium hydroxide, *Sensors and Actuators A,* 29, 121, 1991.
42. Ju, C. and Hesketh, P. J., Measurements of the anisotropic etching of silicon in aqueous cesium hydroxide, *Sensors and Actuators A,* 33, 191, 1992.
43. Reisman, A., Berkenbilt, M., Chan, A. A., Kaufman, F. B., and Green, D. C., The controlled etching of silicon in catalyzed ethylene-diamine-pyrocatechol-water solutions, *J. Electrochem. Soc.*, 126, 1406, 1979.
44. Finne, R. M. and Klein, D. L., A water-amine complexing agent system for etching in silicon, *J. Electrochem. Soc.*, 114, 965, 1967.
45. Tabata, O., Asahi, R., Funabashi, H., Shimaoka, K., and Sugiyama, S., Anisotropic etching of silicon in TMAH solutions, *Sensors and Actuators A,* 34, 51, 1992.
46. Pandy, A., Landsberger, L., Nikpour, B., Paranjape, M., and Kahrizi, M., Experimental investigation of high Si/Al selectivity during anisotropic etching in tetra-methyl ammonium hydroxide, *J. Vacuum Sci. Tech. A*, 16, 868, 1998.
47. Klaassen, E. H., Reay, R. J., Storment, C., Audy, J., Henry, P., Brokaw, A. P., and Kovacs, G. T. A., Micromachined thermally isolated circuits, in *Digest of Technical Papers, Solid-State Sensors and Actuators Worskhop,* Hilton Head, South Carolina, June, 1996, pg. 127.

48. Merlos, A., Acco, M., Bao, M. H., Bausells, J., and Esteve, J., TMAH/IPA anisotropic etching characteristics, *Sensors and Actuators A,* 37-38, 737, 1993.
49. Landsberger, L. M., Naseh, S., Kahrizi, M., and Paranjape, M., On hillocks generated during anisotropic etching of Si in TMAH, *J. Microelectromechanical Syst.*, 5, 106, 1996.
50. Schwartz, B. and Robbins, H. R., Chemical etching of silicon. III. A temperature study I the acid system, *J. Electrochem. Soc.*, 108, 365 1961.
51. Friedrich, C. R. et al., High aspect ratio processing, in *Handbook of Microlithography, Micromaching and Microfabrication*, vol. 2: Micromaching and Microfabrication, Rai-Chowdhury, P., Ed., SPIE Optical Engineering Press, Bellingham, WA, 1997, Chap. 6.
52. Najafi, K., Wise, K. D., and Mochizuki, T., A high-yield IC-compatible multichannel recording array, *IEEE Trans. on Elec. Dev.*, 32, 1206, 1985.
53. Collins, S. D., Etch stop techniques for micromachining, *J. Electrochem. Soc.*, 144, 2242, 1997.
54. Jackson, T., N., Tischler, M. A., and Wise, K. D., An electrochemical p-n junction etch stop for the formation of silicon microstructures, *IEEE Electron. Dev. Lett.*, 2, 44, 1981.
55. Kloeck, B., Collins, S. D., de Rooij, N. F., and Smith, R. L., Study of electrochemical etch-stop for high-precision thickness control of silicon membranes, *IEEE Trans. Electron. Dev.*, 36, 663, 1989.
56. Tuller, H. L., and Mlcak, R., Photo-asisted silicon micromachining: oportunities for chemical sensing, *Sensors and Actuators B,* 35, 255, 1996.
57. Schöning, M. J., Ronkel, F., Crott, M., Thust, M., Schultze, J. W., Kordos, P., and Lüth, H., Miniaturization of potentiometric sensors using porous silicon microtechnology, *Electrochimica Acta*, 42, 3185, 1997.
58. Winters, H. F. and Coburn, J. W., The etching of silicon with XeF_2 vapor, *Appl. Phys. Lett.*, 34, 70, 1979.
59. Hoffman, E., Warneke, B., Kruglick, E., Weingold, J., and Pister, K. S. J., 3D structures with piezoresistive sensors in standard CMOS, in *Proceedings of the IEEE International Meeting on Micro Electro Mechanical Systems*, Amsterdam, The Netherlands, Jan. 29-Feb. 2, 1995, 288.
60. Chu, P. B., Chen, J. T., Yeh, R., Lin, G., Huang, J. C. P., Warneke, B. A., and Pister, K. S. J., Controlled pulse-etching with xenon difluride, in *Proceedings of Transducers '97, Int. Conf. Solid-State Sensors and Actuators*, Chicago, IL, June 16-19, 1997, 665.
61. Tea, N. H., Milanovic, V., Zincke, C. A., Suehle, J. S., Gaitan, M., Zaghloul, M. E., and Geist, J., Hybrid postprocessing etching for CMOS-compatible MEMS., *J. Micromechanical Systems*, 6, 363, 1997.
62. Manos, D. M. and Flamm, D. L., *Plasma Etching: An Introduction*, Academic Press, New York, 1989.
63. Sugawara, M., *Plasma Etching*, Oxford Science Publications, New York, 1998.
64. Bhardwaj, J., Ashraf, H., and McQuarrie, A., Dry silicon etching for MEMS, in *Microstructures and Microfabricated Systems-III, Proceedings of the Electrochemical Society*, vol. 97-5, 118, 1999.
65. Shih, B. St.Clair, L., Hesketh, P. J., Naylor, D. L., and Yershov, G. M., Corner compensation for CsOH micromachining of a silicon fluidic chamber, submitted to *J. Electrochem Soc.*, 1999.
66. Offereins, H. L., Sandmaier, H., Marusczyk, K., Kühl, K., and Plettner, A., Compensating corner under-cutting of (100) silicon in KOH, *Sensors and Materials*, 3, 127, 1992.
67. Howe, R. T. and Muller, R. S., Polycrystalline and amorphous silicon micromechanical beams: annealing and mechanical properties, *Sensors and Actuators*, 4, 447, 1983.
68. Maier-Schneider, D., Maibach, J., Obermeier, E., and Schneider, D., Variations in Young's modulus and intrinsic stress of LPCVD-polysilicon due to high-temperature annealing, *J. Micromech. Microeng.*, 5, 121, 1995.
69. Guckel, H., Sniegowski, J. J., Christenson, T. R., and Raissi, F., The application of fine-grained, tensile polysilicon to mechanically resonsnt transducers, *Sensors and Actuators A,* 21, 346, 1990.
70. Krulevitch, P. A., Micromechanical Investigations of Silicon and Ni-Ti-Cu Thin Films, Ph.D. thesis, University of California Berkeley, 1994.
71. Guckel, H. and Burns, D. W., Planar processed polysilicon sealed cavities for pressure transducer arrays, in *Proceedings of the IEEE International Electron Devices Meeting*, San Francisco, CA, Dec. 9-12th, 1984, 223.

72. Westberg, D., Paul, O., Anderson, G. I., and Baltes, H., Surface micromachining by sacrificial aluminum etching, *J. Micromech. Microeng.*, 6, 376, 1996.
73. Tas, N., Sonnenberg, T., Jansen, H., Legtenberg, R., and Elwenspoek, M., Stiction in surface micromachining, *J. Micromech. Microeng.*, 6, 385, 1996.
74. Intellisense Inc., Cambridge, MA.
75. Gennissen, P. T. J., Bartek, M., French, P. J., and Sarro, P. M., Bipolar-compatible epitaxial poly for smart sensors: stress minimization and applications, *Sensors and Actuators*, 62, 636, 1997.
76. Wenk, B., Thick polysilicon based surface micromachining, in *Microstructures and Microfabricated Systems -IV, Proceedings of the Electrochemical Society*, vol. 98-14, 12, 1998.
77. Sharpe, W. N., Yuan, B., Vaidyanathan, R., and Edwars, R. L., Measurements of Young's modulus, Poisson's ratio, and tensile strength of polysilicon, *Proceedings of the Tenth Annual Internations Workshop on Micro Electro Mechanical Systems*, Nagoya, Japan, January 1997, IEEE, NJ, Catalog Number 97CH36021, 424.
78. Kahn, H., Stemmer, S., Nandakumar, K., Heuer, A. H., Mullen, P. L. Ballarini, R., and Huff, M. A., Mechanical properties of thick, surface micromachined polysilicon films, in Proceedings *of the Ninth Annual International Workshop of Micro Electo Mechanical Systems*, San Diego, CA, February 1996, IEEE, NJ, Cat. Number 96CH35856, 343.
79. Maier-Schneider, D., Maibach, J., Obermeier, E., and Schneider, D., Variations in Young's modulus and intrinsic stress of LPCVD-polysioicon due to high-temperature annealing, *J. Micromech. Microeng.*, 5, 121, 1995.
80. Biebl, M. and Philipsborn, Fracture strength of doped and undoped polysilicon, in Digest of Technical Papers, The *8th International Conference on Solid-State Sensors and Actuators, and Eurosensors IX*, Stockholm, Sweden, June, 1995, 72
81. Adams, A. C., *Dielectric and Polysilicon Film Deposition*, Chapter 6, in VLSI Technology, Editor S. M. Sze, McGraw-Hill, New York, 1983, 93.
82. Gardeniers. J. G. E., Tilmans, H. A. C., and Visser, C. C. G., LPCVD silicon-rich silicon nitride films for applications in micromechanics, studeied with statistical experimental design, *J. Vac. Sci. Tech. A.*, 14, 3879, 1996.
83. Chou, B. C. S., Shie, J.-S., and Chen, C.-N., Fabrication of low-stress dielectric thin-film for microsensor applications, *IEEE Electron Device Letters*, 18, 599, 1997.
84. French, P. J., Sarro, P. M., Mallée, R., Fakkeldij, E. J. M., and Wolffenbuttel, Optimization of a low-stress silicon nitirde process for surface-micromachining applications, *Sensors and Actuators A,* 58, 149, 1997.
85. Habermehl, S., Stress relaxation in Si-rich silicon nitride thin films, *J. Appl. Phys.*, 83, 4672, 1998.
86. Classen, W. A. P. et al., Influence of deposition temperature, gas pressure, gas phase composition, and RF frequency on composition and mechanical stress of plasma silicon nitride layers, *J. Electrochem. Soc.*, 132, 893, 1985.
87. Sarro, P. M., deBoer, C. R., Korkmaz, E., and Laros, J. M. W., Low-stress PECVD SiC thin films for IC-compatible microstructres, *Sensors and Actuators A,* 67, 175, 1998.
88. Pan, L. S. and Kania, D. R., *Diamond: Electronic Properties and Applications*, Kluwer Academic Pub., 1995.
89. French, P. J. and Sarro, P. M., Surface versus bulk micromachining: the contest for suitable applications, *J. Micromech. Microeng.*, 8, 45, 1998.
90. Rogesrs, M. S. and Sniegowski, J. J., 5-level polysilicon surface micromachine technology: application to complex mechanical systems, in *Digest of Technical Papers Solid-State Sensor and Actuator Workshop*, Hilton Head, SC, June 1998, 144.
91. Sniegowski, J. J., Miller, S. L., LaVigne, G. F., Rogers, M. S., and McWhorter, P. J., Monolithic geared-mechanisms driven by a polysilicon surface-micromachined on-chip electrostatic microengine, in *Digest of Technical Papers, Solid-State Sensors and Actuators Workshop,* Hilton Head, SC, June, 1996, 178,

92. Sniegowski, J. J., Chemical mechanical polishing: an enabling fabrication process for surface micromachining technologies, in *Microstructures and Microfabricated Systems-IV*, Ed., P. J. Hesketh, H. Hughes, and W. E. Bailey, Proceedings of the Electrochemical Society, vol. 98-14, 1, 1998.
93. McDonald, N. C., SCREAM MicroElectroMechanical Systems, *Microelectroic Engineering*, 32, 49, 1996.
94. Saif, M. T. A. and MacDonald, N. C., Planarity of large MEMS, *J. Microelectromech. Syst.*, 5, 79, 1996.
95. Keller, C. G. and Howe, R. T., Nickel-filled hexsil thermally actuated tweezers, in *Proceedings of Transducers*, 95.
96. Soane, D. S. and Martynenko, Z., *Polymers in Microelectronics Fundamentals and Applications*, Elsevier, New York, 1989.
97. Lorenz, H., Despont, M., Fahrni, N., LaBianca, N., Renaud, P., and Vettiger, P., SU-8: a low-cost negative resist for MEMS, *J. Micromech. and Microeng.*, 7, 121, 1997.
98. Frazier, A. B., Ahn, C. H., and Allen, M. G., Development of micromachined devices using polyimide-based processes, *Sensors and Actuators A*, 45, 47, 1994.
99. Ahn, C. H., Kim, Y. J., and Allen, M. G., A fully integrated planar toroidal inductor with a micromachinined nickel-iron magnetic bar, *IEEE Trans. Comp. Packag. & Manuf. Tech. A*, 17, 3, 1994.
100. Ahn, C. H. and Allen, M. G., A planar micromachined spiral inductor for integrated magnetic microactuator applications, *J. Micromech. and Microeng.*, 3, 37, 1993.
101. Becker, E. W. et al., Fabrication of microstructures with high aspect ratios and great structureal heights by synchrotron radiation lithography, galvanoforming, and plastic molding (LIGA Process), *Microelectronic Eng.*, 4, 35, 1986.
102. Cox, J. A., Zook, J. D., Ohnstein, T., and Dobson, D. C., Optical performance of high-aspect LIGA gratings, *Opt. Eng.*, 36, 1367, 1997.
103. Hjort, K., Söderkvist, J., and Schweits, J.-A., Gallium arsenide as a mechanical material, *J. Micromech. & Microeng.*, 4, 1, 1994.
104. Parameswaran, M., Baltes, H. P., Ristic, L.J., Dhaded, A. C., and Robinson, A. M., A new approach for the fabrication of micromechanical structures, *Sensors and Actuators*, 19, 289, 1989.
105. Nicollian, E. H. and Brews, J. R., *MOS (Metal Oxide Semiconductor) Physics and Technology*, Addision-Wesley, Reading, MA, 1990.
106. Parameswaran, M., Robinson, A. M., Blackburn, D. L., Gaitan, M., and Geist, J., Micromachined thermal radiation emitter from a commercial CMOS process, *IEEE Elec. Dev. Lett.*, 12, 57, 1991.
107. Fedder, G. K., Santhanam, S., Reed, M. L., Eaggle, S. C., Guillou, D. F., Lu, M. S.-C., and Carley, L. R., Laminated high-aspect-ratio microstructures in a conventional CMOS process, in *Proceedings of the Ninth Annual International Workshop of Micro Electo Mechanical Systems*, San Diego, CA, February, 1996, IEEE, NJ, Cat. Number 96CH35856, 13.
108. Lee, J.-B., English, J., Ahn, C.-H., and Allen, M. G., Planarization techniques for vertically integrated metallic MEMS on silicon foundry circuits, *J. Micromach. Microeng.*, 7, 44, 1997.
109. Bustillo, J. M., Fedder, G. K., Nguyen, C. T.-C., and Howe, R. T., Process technology for the modular integration of CMOS and polysilicon microstructures, *Microsystem. Technol.*, 1, 30, 1994.
110. Smith, J., Montague, S., Sneigowski, J., Murry, J., and McWhorter, P., Embedded micromechanical devices for the monolithic integration of MEMS with CMOS, *IEDM Tech. Digest '95*, 1995, 609.
111. Gianchandani, Y. B., Kim, H., Shinn, M., Ha, B., Lee, B., Najafi, K., and Song, C., A MEMS-first fabrication process for integrating CMOS circuits with polysilicon microstructures, in *Proceedings IEEE the Eleventh Annual International Workshop on Micro Electro Mechanical Systems*, Heidelberg, Germany, January 1998, 257
112. Parameswaran, L., Hsu, C. H., and Schmidt, M. A., Integrated sensor technology, in *Meeting Abstracts of the 194th Meeting of the Electrochemical Society*, Boston, Nov. 1-6, Abst # 1144, 1998.
113. Wallis, G. and Pomerantz, Field assisted glass-metal sealing, *J. Appl. Phys.*, 40, 3946, 1969.

114. Albaugh, K. B. and Cade, E., Mechanism of anodic bonding of silicon to pyrex glass, *Digest of Technical Papers, IEEE Solid-State Sensors and Actuators Workshop,* Hilton Head Island, SC, 1988, 109.
115. Cunneen, J., Lin, Y.-C., Caraffini, S., Boyd, J. G., Hesketh, P. J., Lunte, S. M., and Wilson, G. S., A positive displacement micropump for microdialysis, *Mechatronics Journal*, 8, 561, 1998.
116. van der Groen, S., Rosmeulen, M., Baert, K., Jansen, P., and Deferm, L., Substrate bonding techniques for CMOS processed wafers, *J. Micromech. Microeng.*, 7, 108, 1997.
117. Shimbo, M. et al., Silicon-to-silicon direct bonding method, *J. Appl. Phys.*, 60, 2987, 1986.
118. Tong, Q. Y. and Gösele, U., *Semiconductor Wafer Bonding*, John Wiley & Sons, 1998.
119. Knecht, T. A., Bonding techniques for solid-state pressure sensors, *Proc. Transducers '87*, Tokyo, 95-98.
120. Legtenberg, R., Bouwstra, S., and Elwenspoek, M., Low-temperature glass bonding for sensor applications using boron oxide thin films, *J. Micromech. Microeng.*, 1, 157-160, 1991.
121. Nguyen, M. N., Low stress silver-glass die attach material, *IEEE Trans. Comp. Hybrid, Manuf. Tech.,* 13, 478, 1990.
122. Hanneborg, A., Nese, M., and Ohlckers, P., Silicon-to-silicon anodic bonding with a borosilicate glass layer, *J. Micromech. Microeng.*, 1, 139, 1991.
123. Wu, M. C., Micromachining for optical and optoelectronic systems, *Proceedings of the IEEE*, 85, 1833, 1997.
124. Motamedi, M. E., Micro-opto-electro-mechanical systems, *Optical Engineering*, 33, 3505, 1994.
125. Motamedi, M. E., Wu, M. C., and Pister, K. S. J., Micro-opto-elecro-mechanical devices and on-chip optical processing, *Opt. Eng.*, 36, 1282, 1997.
126. Pister, K. S. J., Judy, M. W., Burgett, S. R., and Fearing, R. S., Microfabricated hinges, *Sensors and Actuators A,* 33, 249, 1992.
127. Roggeman, M. C., Bright, V. M., Welsh, B. M., Hick, S. R., Roberts, P. C., Cowan, W. D., and Comtois, J. H., Use of micro-electro-mechanical deformable mirrors to control aberrations in optical systems: theoretical and experimental results, *Opt. Eng.*, 36, 1326 1997.
128. Usenishi, Y., Tsugari, M., and Mehregany, M., Micro-opto-mechanical devices fabricated by anisotropic etching of (110) silicon, *J. Micromech. Microeng*, 5, 305, 1995.
129. Larson, M. C., Pezeshki, B., and Harris, J. S., Vertical coupled-cavity microinterferometer on GaAs with deformable-membrane top mirror, *IEEE Phot. Tech. Lett.*, 7, 382, 1995.
130. Gossen, K. W., Walker, J. A., and Arney, S. C., Silicon modulator based on mechanically-active anti-reflection layer with 1 Mbit/sec capability for fiber-in-the-loop applications, *IEEE Photon. Tech. Lett.*, 6, 1119, 1994.
131. Sene, D. E., Grantham, J. W., Bright, V. M., and Comtois, J. H., Development and characterization of micro-mechanical gratings for optical modulation, in *Proceedings of the Ninth Annual International Workshop on Micro Electro Mechanical Systems*, San Diego, CA, Februray 1996, 222.
132. Burns, D. B. and Bright, V. M., Development of microelectromechanical variable blaze gratings, *Sensors and Actuators A,* 64, 7, 1998.
133. Miller, R., A., Burr, G. W., Tai, Y.-C., and Psaltis, D., Electromagnetic MEMS scanning mirrors for holographic data storage, in *Digest of Technical Papers, Solid-State Sensor and Actuator Workshop*, Hilton Head, South Carolina, June 1996, 183.
134. Asada, N., Matsuki, H., Minami, K., and Esashi, M., Silicon micromachined two-dimensional galvano optical scanner, *IEEE Trans. on Mag.*, 30, 4647, 1994.
135. Judy, J. W. and Muller, R. S., Batch-fabricated, addressable, magnetically actuated microstructures, in *Digest of Technical Papers, Solid State Sensors and Actuators Workshop,* Hilton Head, SC, June 1996, 187.
136. Miller, R. and Tai, Y.-C., Micromachined electromagnetic scanning mirrors, *Opt. Eng.*, 36, 1399, 1997.
137. Kiang, M. H., Solgaard, O., Muller, R. S., and Lau, K. Y., Surface-micromachined electrostatic-comb drive scanning micromirrors for barcode scanners, in *Proceedings of the Ninth Annual International Workshop on Micro Electro Mechanical Systems*, San Diego, CA, February 1996, 192.

138. Fischer, M., Nägele, M., Eichner, D., Schöllhorn, C., and Strobel, R., Integration of surface-micromachined polysilicon mirrors and a standard CMOS process, *Sensors and Actuators A,* 52,140, 1996.
139. Bühler, J., Steiner, F.-P., Hauert, R., and Baltes, H., Linear array of complementary metal oxide semiconductor double-pass metal micromirrors, *Opt. Eng.*, 35, 1391, 1997.
140. Ikeda, M., Goto, H., Sakata, M., Wakabayashi, S., Imanaka, K., Takeuchi, M., and Yada, T., Two dimensional silicon micromachined noptical scanner intetgrated with photo detector and piezoresistor, in *Digest of Technical Papers, The 8th International Conference on Solid-State Sensors and Actuators, and Eurosensors IX*, Stockholm, Sweden, 1995, 293.
141. Huang, Y., Zhang, H., Kim, E. S., Kim, G. K., and Joen Y. B., Piezoelectrically actuated microcantilever for actuated mirror array application, in *Digest of Technical Papers, Solid-State Sensor and Actuator Workshop*, Hilton Head, South Carolina, June 1996, 191.
142. Lin, L. Y., Lee, S. S., Pister, K. S. J., and Wu, M. C., Micro-machined three-dimensional micro-optics for integrated free-space optical systems, *IEEE Photon. Tech. Lett.*, 6, 1445, 1994.
143. Lee, S. S., Lin, L. Y., and Wu, M. C., Surface-micromachined free-space micro-optical systems containing three-dimensional microgratings, *Appl. Phys. Lett.*, 67, 2135, 1995.
144. Bright, V. M., Comtois, J. H., Reid, J. R., and Sene, D. E., Surface micromachined micro-opti-electro-mechanical systems, *IEICE Trans. Elec.*, E80, 206, 1997.
145. Markus, K. W. and Koester, D. A., Multi-user MEMS process (MUMPS) introduction and design rules, MCNC Electronics Tech. Div., 3201 Cornwallis Road, Research Triangle Park, NC, Oct. 1994.
146. Hornbeck, L. J., Digital light processing™ and MEMS: refelecting the digital dispaly needs of the networked society, in *Symposium Micromachining and Microfabrication*, Proc. SPIE vol. 2783, Austin, TX, 1996, 2.
147. Ukita, H., Sugiyama, Y., Nakada, H., and Katagiri, Y., Read/wite performance and reliability of a flying optical head using a monolithically integrated LD-PD, *Appl. Optic.*, 30, 3770, 1991.
148. Lin, L. Y., Shen, J. L., Lee, S. S., and Wu, M. C., Realization of novel monolithic free-space optical disk pickup heads by surface micromachining, *Optics Letters*, 21, 155, 1996.
149. Ukita, H., Micromechanical photonics, *Optical Review*, 4, 623, 1997.
150. Maluf, N. I., Reay, R. J., and Kovacs, G. T. A., High-voltage devices and circuits fabricated using foundary CMOS for use with electrostatic MEM actuators, *Sensors and Actuators A*, 52, 187, 1996.
151. Daneman, M. J, Tien, N. C., Solgaard, O., Lau, K. Y., and Muller R. S., Linear vibromotor-actuated micromachined microreflector for integrated optical systems, in *Digest of Technical Papers, Solid-State Sensor and Actuator Workshop*, Hilton Head, SC, June 1996, 109.
152. Fukuta, Y., Collard, D., Akiyama, T., Yang, E. H., and Fujita, H., Microactuated self-assembling of 3D polysilicon structures with reshaping technology, in *Proceedings of the 10th Annual International Workshop on Micro Electro Mechanical Systems*, Nagoya, Japan (IEEE, NJ, 1997), 477.
153. Lin, L. Y., Shen, J. L., Lee, S. S., Su, G. D., and Wu, M. C., Microactuated micro-xyz stages for free-space micro-optical bench, in *Proceedings of the 10th Annual International Workshop on Micro Electro Mechanical Systems*, Nagoya, Japan, 1997, 43.
154. Comtois, J. H. and Bright, V. M., Surface micromachined polysilicon thermal acutaor arrays and applications, in *Digest of Technical Papers, Solid-State Sensor and Actuator Workshop*, Hilton Head, SC, June 1996, 152.
155. Reid, J. R., Bright, V. M., and Comtois, J. H., Arrays of thermal micro-actuators coupled to micro-optical components, *Proc. of SPIE*, 2865, 74, 1996.
156. Reid, J. R, Bright, V. M., and Butler, J. T., Automated assembly of flip-up mirocmirrors, *Sensors & Actuators A*, 66, 292, 1998.
157. Shen, B. et al., Contilever micromachined structures in CMOS technology with magnetic actuation, *Sensors and Materials*, 9, 347, 1997.
158. Chang, J. Y.-C., Abidi, A. A., and Gaitan, M., Large suspended inductors on silicon and their use in a 2-μm MOS RF amplfier, *IEEE Electron Device Letters,* 14, 246, 1983.
159. Rofougaran, A., Chang, J. Y. C., Rofougaran, M., and Abidi, A., A GHz CMOS RF front-end IC for a direct-conversion wireless receiver, *IEEE Journal of Solid-State Circuits*, 31, 880, 1996.

160. López-Villegas, J. M., Samitier, J., Bausells, J., Merlos, A., Cané, and Knöchel, R., Study of integrated RF passive components performed using CMOS and Si micromachining technologies, *J. Micromachine and Microeng.*, 7, 162, 1997.
161. McGrath, W. R., Walker, C, Yap, M., and Tai, Y.-C., Silicon micromachined waveguides for millimeter-wave and submillimeter-wave frequencies, *IEEE Microwave and Guided Wave Letters*, 3, 61, 1993.
162. Franklin-Drayton, R., Dib, N. I., and Katehi, L. P. B., Design of micromachined high frequency circuit components, *International Journal of Microcircuits and Electronic Packaging*, 18, 19, 1995.
163. Drayton, R. F., Henderson, R. M., and Katehi, L. P. B., Monolithic packaging concepts for high isolation in circuits and antennas, *IEEE Trans. Microwave Theory and Tech.*, 46, 900, 1998.
164. Katehi, L. P. B., Rebeiz, G. M., Weller, T. M., Drayton R. F., Cheng, H.-J., and Whitaker, J. F., Micromachined circuits for millimeter- and sub-millimeter-wave applications, *IEEE Antennas and Propagation Magazine*, 35, 9, 1993.
165. Milanovic, V., Gaitan, M., Bowen, E. D., and Zaghloul, M. E., Micromachined microwave transmission lines in CMOS technology, *IEEE Trans. Microwave Theory and Techniques*, 45, 630 1997.
166. Roessig, T. A., Howe, R. T., Pisano, A. P., and Smith, J. H., Surface-micromachined 1Mhz oscillator with low-noise pierce configuration, in *Digest of Technical Papers, IEEE Solid-State Sensors and Actuators Workshop,* Hilton Head Island, SC, 1988, 328.
167. Baltes, H., Paul, O., and Brand, O., Micromachined thermally based CMOS microsensors, *Proceedings of the IEEE*, 86, 1660, 1998.
168. Reay, R. J., Klaassen, E. K., and Kovacs, G. T. A., A micromachined low-power temperature-regulated bandgap voltage reference, *IEEE J. Solid-State Circuits*, 30,1374, 1995.
169. Klaassen, E. H., Reay, R. J., and Kovacs, G. T. A., Diode-based thermal r.m.s. converter with on-chip circuitry fabricated using CMOS technology, *Sensors and Actuators A*, 52, 33, 1996.
170. Zavracky, P. M., Majumder, S., and McGruer, N. E., Micromechanical switches fabricated using nickel surface micromachining, *J. Microelectromechanical Systems*, 6, 3, 1997.
171. McGruer, N. E., Zavracky, P. M., Majumder, S., Morrison, R., and Adams, G. G., Electrostatically actuated microswitches; scaling properties, in *Digest of Technical Papers, IEEE Solid-State Sensors and Actuators Workshop,* Hilton Head Island, SC, 1988, 132.
172. Drake, J., Jerman, H., Lutze, B., and Stuber, M., An electrostatically actuated micro-relay, in *Proceedings of Transducers '95, Int. Conf. Solid-State Sensors and Actuators*, 1995, 384 .
173. Yao, J. J., and Chang, M. Frank, A surface micromachined miniature switch for telecommunications applications with signal frequencies from DC up to 4 Ghz, in *Proceedings of Transducers '95, Int. Conf. Solid-State Sensors and Actuators*, 1995, 384.
174. Randall, J. N., Goldsmith, C., Denniston, D., and Lin, T.-H., Fabrication of micromechanical switches for routing radio frequency signals, *J. Vac. Sci. Tech. B*, 14, 3692, 1996.
175. Schiele, I., Huber, J., Hillerich, B., and Kozlowski, F., Surface-micromachined electrostatic microrelay, *Sensors and Actuators A*, 66, 345, 1998.
176. Grétillat, M.-A., Thiebaud, P., Linder, C., and de Rooij, N. F., Integrated circuit compatible electrostatic polysilicon microrelays, *J. Micromech. and Microeng.*, 5, 156, 1995.
177. Sun, X., Q., Farmer, K. R., and Carr, W. N., A bistable microrelay based on two-segment multimorph cantilever actuators, in *Proceedings IEEE the Eleventh Annual International Workshop on Micro Electro Mechanical Systems*, Heidelberg, Germany, January 1998, 154.
178. Kruglick, E. J. J. and Pister, K. S. J., Bistable MEMS relays and contact characterization, *Digest of Technical Papers, IEEE Solid-State Sensors and Actuators Workshop,* Hilton Head Island, SC, 1988, 333.
179. Taylor, W. P., Brand, O., and Allen, M. G., Fully integrated magnetically actuated micromachined relays, *J. Microelectro. Mech. Syst.*, 7, 181, 1998.
180. Holm, R., *Electric Contacts Handbook*, Springer-Verlag, Berlin, 1958.
181. Schimkat, J., Contact materials for microrelays, in *Proceedings IEEE the Eleventh Annual International Workshop on Micro Electro Mechanical Systems*, Heidelberg, Germany, January 1998, 190.

182. Krause, P., Obermeier, E., and Wehl, W., Backshooter — a new smart micromachined single-chip ink jet printhead, in *Proceedings of Transducers '95, Int. Conf. Solid-State Sensors and Actuators*, 1995.
183. Janata, J., Josowicz, Vaný sek, P., and DeVaney, D. M., Chemical Sensors, *Anal. Chem.*, 70, 179R, 1998.
184. Madou, M., Otagawa, T., Joseph, J., Hesketh, P., and Saaman, A., *Catheter-Based Micromachined Electrochemical Sensors, SPIE Optics, Electro-Optics, and Laser Applications in Science and Engineering*, Los Angeles, CA, January 1989.
185. Janata, J., Chemically sensitive field-effect transistors, in *Solid State Chemical Sensors*, J. Janata and R. J. Huber, Eds., Academic Press, New York, 1985.
186. Bergveld, P., Development of an ion-sensitive solid-state devices for neurophysioligical measurements, *IEEE Trans Biomed. Eng.*, 17, 70, 1970.
187. Janata, J., Chemical selectivity of field-effect transistors, *Sensors and Actuators*, 12, 121, 1987.
188. Lundström, I. and Svensson, C., Gas-sensitive metal gate semiconductor devices, in *Solid State Chemical Sensors*, J. Jananta and R. J. Huber, Eds., Academic Press, New Yrok, 1985.
189. Sibald, A., Recent advances in field-effect chemical microsensors, *J. Molecular Electronics*, 2, 51, 1986.
190. Bousse, L. and Bergveld, P., The role of buried OH-sites in the response mechanism of inorganic-gate pH-sensitive ISFETs, *Sensors and Actuators*, 6, 65, 1984.
191. Cané, C., Grácia, I., and Merlos, A., Microtechnologies for pH ISFET chemical sensors, *Microelectronics Journal*, 28, 389, 1997.
192. Domanský, K., Zapf, V. S., Grate, J. W., Ricco, A. J., Yelton, W. G., and Janata, J., Integrated chemiresistor and work function microsensor array with cargon black/polymer composite materials, in *Digest of Technical Papers, IEEE Solid-State Sensors and Actuators Workshop,* Hilton Head Island, SC, 1988, 187.
193. Rodriguez, J. L., Hughes, R. C., Corbett, W. T., and McWhorter, P. J., Robust, wide range hydrogen sensor, *IEDM Tech. Digest*, IEEE, Cat # 92CH3211-0, San Francisco, CA, 1992, 521.
194. Suehle, J. S., Cavicchi, R. E., Gaitan, M., and Semancik, S., Tin oxide gas sensor fabricated using CMOS micro-hotplates and in-situ processing, *IEEE Elec. Dev. Lett.*, 14, 118 1993.
195. Semancik, S., and Cavicchi, R., Kinetically controlled chemical sensing using micromachined structures, *Acc. Chem. Res.*, 31 279, 1998.
196. Patel, S. V., Wise, K. D., Gland, J. L., Zanini-Fisher, M., and Schwank, J. W., Characteristics of silicon-micromachined gas sensors based on Pt/TiO_x thin films, *Sensors and Actuators B,* 42, 205, 1997.
197. Cornila, C., Hierlemann, A., Lenggenhager, R., Malcovati, P., Baltes, H., Noetzel, G., Weimar, U., and Göpel, W., Capacitive sensors in CMOS technology with polymer coating, *Sensors and Actuators B,* 24, 357, 1995.
198. Hierlemann, A., Schwiezer-Berberich, M., Weimar, U., Kraus, G., Pfau, A., and Göpel, W., Pattern recognition and multicomponent analysis, in *Sensors Update*, H. Baltes, W. Göpel, and J. Hesse, Eds., VCH Pub., Weinheim, Germany, 1996.
199. Najafi, K., Micromachined systems for neurophysiological appplications, in *Handbook of Microlithogrpahy, Micromachining, and Microfabrication*, vol. 2: Micromachining and Microfabrication, Ed., P. Rai-Choudhury, SPIE, Washington, 1997, 517.
200. Akin, T., and Najafi, K., A micromachined silicon sieve electrode for nerve regeneration applications, *IEEE Trans. Biomed. Eng.*, 41, 305, 1994.
201. Kewley, D. T., Hills, M. D., Borkholder, D. A., Opris, I. E., Maluf, N. I., Storment, C. W., Bower, J. M., and Kovacs, G. T. A., Plasma-etched neural probes, *Sensors and Actuators A,* 58, 27, 1985.

Khandelwal, P., Shenai, K. "Microelectronics Packaging"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

11

Microelectronics Packaging

Krishna Shenai
Pankaj Khandelwal
University of Illinois at Chicago

11.1 Introduction

Packaging of electronic circuits is the science and the art of establishing interconnections and a suitable operating environment for predominantly electrical circuits. It supplies the chips with wires to distribute signals and power, remove the heat generated by the circuits, and provides them with physical support and environmental protection. It plays an important role in determining the performance, cost, and reliability of the system. With the decrease in feature size and increase in the scale of integration, the delay in on-chip circuitry is now smaller than that introduced by package. Thus, the ideal package would be one that is compact, and should supply the chips with a required number of signal and power connections, which have minute capacitance, inductance, and resistance. The package should remove the heat generated by the circuits. Its thermal properties should match well with semiconductor chip to avoid stress-induced cracks and failures. The package should be reliable, and it should cost much less than the chips it carries[1] (See Table 11.1).

TABLE 11.1 Electronic Packaging Requirements

Speed	Size
• Large bandwidth	• Compact size
• Short inter-chip propagation delay	
Thermal & Mechanical	Test & Reliability
• High heat removal rate	• Easy to test
• A good match between the thermal coefficients of the dice and the chip carrier	• Easy to modify • Highly reliable • Low cost
Pin Count & Wireability	Noise
• Large I/O count per chip	• Low noise coupling among wires
• Large I/O between the first and second level package	• Good-quality transmission line • Good power distribution

11.2 Packaging Hierarchy

The semiconductor chip is encapsulated into a package, which constitutes the first level of packaging. A printed circuit board is usually employed because the total circuit and bit count required might exceed that available on a single first-level package. Further, there may be components that cannot be readily integrated on a chip or first-level package, such as capacitors, high power resistors, inductors, etc. Therefore, as a general rule, several levels of packaging will be present. They are often referred to as a packaging hierarchy. The number of levels within a hierarchy may vary, depending on the degree of integration and the totality of packaging needs[2] (see Fig. 11.1).

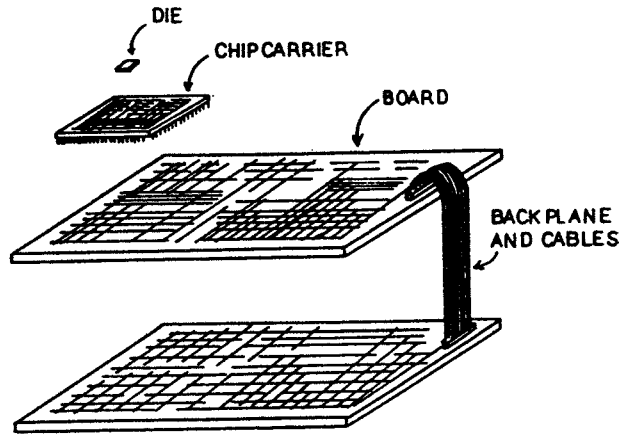


FIGURE 11.1 Packaging hierarchy of a hypothetical digital computer.

In the past, the packaging hierarchy contained more levels. Dies were mounted on individual chip carriers, which were placed on a printed circuit board. Cards then plugged into a larger board, and the boards were cabled into a gate. Finally, the gates were connected to assemble the computer. Today, higher levels of integration make many levels of packaging unnecessary, and this improves the performance, cost, and reliability of the computers. Ideally, all circuitry one day may be placed on a single piece of semiconductor. Thus, packaging evolution reflects the integrated circuits progress.[3,4]

11.3 Package Parameters

A successful package design will satisfy all given application requirements at an acceptable design, manufacturing, and operating expense. As a rule, application requirements prescribe the number of logic

circuits and/or bits of storage that must be packaged, interconnected, supplied with electric power, kept within a proper temperature range, mechanically supported, and protected against the environment. Thus. IC packages are designed to accomplish the following three basic functions[5]:

- Enclose the chip within a protective envelope to protect it from the external environment
- Provide electrical connection from chip to circuit board
- Dissipate heat generated by the chip by establishing a thermal path from a semiconductor junction to the external environment

To execute these functions, package designers start with a fundamental concept and, using principles of engineering, material science, and processing technology, create a design that encompasses:

1. Low lead capacitance and inductance
2. Safe stress levels
3. Material compatibility
4. Low thermal resistance
5. Seal integrity
6. High reliability
7. Ease of manufacture
8. Low cost

Success in performing the functions outlined depends on the package design configuration, the choice of encapsulating materials, and the operating conditions.[6,7] Package design is driven by performance, cost, reliability, and manufacturing considerations. Conflicts between these multiple criteria are common. The design process involves many tradeoff analyses and the optimization of conflicting requirements.

While designing the package for an application, the following parameters are considered.

Number of Terminals

The total number of terminals at packaging interfaces is a major cost factor. Signal interconnections and terminals constitute the majority of conducting elements. Other conductors supply power and provide ground or other reference voltages.

The number of terminals supporting a group of circuits is strongly dependent on the function of this group. The smallest pinout can be obtained with memory ICs because the stream of data can be limited to a single bit. Exactly the opposite is the case with groups of logic circuits which result from a random partitioning of a computer. The pinout requirement is one of the key driving parameters for all levels of packaging: chips, chip carriers, cards, modules, cables, and cable connectors.

Electrical Design Considerations

Electrical performance at the IC package level is of great importance for microwave designs and has gained considerable attention recently for silicon digital devices due to ever-increasing speed of today's circuits and their potentially reduced noise margins.[8] As a signal propagates through the package, it is degraded due to reflections and line resistance. Controlling the resistance and the inductance associated with the power and ground distribution paths to combat ground bounce and the simultaneous switching noise has now become essential. Controlling the impedance environment of the signal distribution path in the package to mitigate the reflection-related noise is becoming important. Reflections, in addition, cause an increase in the transition time, and may split the signal into two or more pulses with the potential of causing erroneous switching in the subsequent circuit and thus malfunction of the system. Controlling the capacitive coupling between signal traces in the signal distribution path to reduce crosstalk is gaining importance. Increased speed of the devices demands that package bandwidth be increased to reduce undue distortion of the signal. All these criteria are related through geometric variables, such as conductor cross-section and length, dielectric thickness, and the dielectric constant of the packaging body. These problems are usually handled with transmission line theory.[9]

Thermal Design Considerations

The thermal design objective is to keep the operating junction temperature of a silicon chip low enough to prevent triggering the temperature-activated failure mechanisms. Thus, the package should provide a good medium for heat transfer from junction to the ambient/heat sink. It is generally recommended to keep the junction temperature below 150°C to ensure proper electrical performance and to contain the propensity to fail.[10,11]

Thermal expansion caused by heating up the packaging structure is not uniform — it varies in accordance with the temperature gradient at any point in time and with the mismatches in the thermal coefficient of expansion. Mechanical stresses result from these differences and are one of the contributors to the finite lifetime and the failure rate of any packaging structure.[12]

In a simplistic heat transfer model of a packaged chip, the heat is transferred from the chip to the surface of the package by conduction, and from the package surface to the ambient by convection and radiation.[13,14] Typically, the temperature difference between the case and ambient is small, and hence radiation can be neglected. This model also neglects conduction heat transfer out of the package terminals, which can become significant. A multilayer example, which models the heat transfer from a region in the silicon device to the ambient, is shown in Fig. 11.2. The total thermal resistance from the junction to the ambient is given by:

$$R_{\theta ja} = R_{\theta jc} + R_{\theta cs} + R_{\theta sa} \tag{11.1}$$

The resulting junction temperature, assuming a power dissipation of P_d, is

$$T_j = P_d(R_{\theta jc} + R_{\theta cs} + R_{\theta sa}) + T_a \tag{11.2}$$

in analogy with electric circuits. If there are parallel paths for heat flow, the thermal resistances are combined in exactly the same manner as electrical resistors in parallel.

$R_{\theta cs}$, the conductive thermal resistance, is mainly a function of package materials and geometry. With the higher power requirements, one must consider the temperature dependence of materials selected in design. T_j depends on package geometry, package orientation in the application, and the conditions of the ambient in the operating environment. The heat sink is responsible for getting rid of the heat of the environment by convection and radiation. Because of all the many heat transfer modes occurring in a finned heat sink, the accurate way to obtain the exact thermal resistance of the heat sink would be to measure it. However, most heat sink manufactures today provide information about their extrusions concerning the thermal resistance per unit length.

Reliability

The package should have good thermomechanical performance for better reliability. A variety of materials of widely differing coefficients of thermal expansion (CTEs) are joined to create interfaces. These interfaces are subject to relatively high process temperatures and undergo many temperature cycles in their useful life as the device is powered on and off. As a result, residual stresses are created in the interfaces. These stresses cause reliability problems in the packages.[15,16]

Testability

Implicit in reliability considerations is the assumption of a flawless product function after its initial assembly — a zero defect manufacturing. Although feasible in principle, it is rarely practiced because of the high costs and possible loss of competitive edge due to conservative dimensions, tolerances, materials, and process choices. So, several tests are employed to assess the reliability of the packages.[17,18]

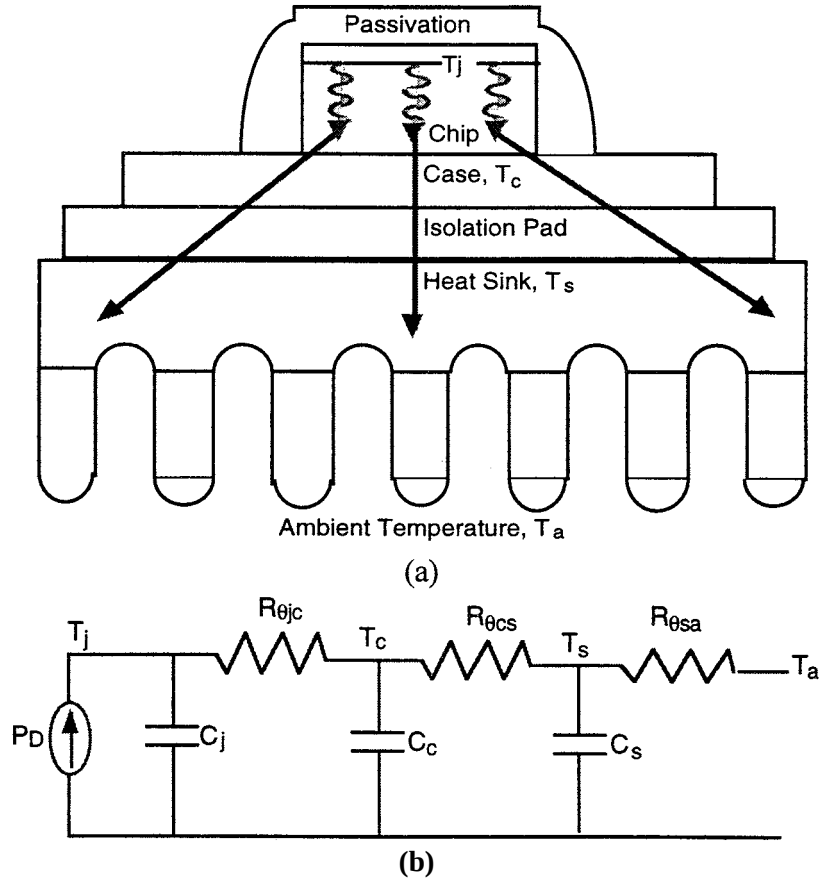


FIGURE 11.2 Steady-state heat flow and thermal resistance in a multilayer structure (a) path of heat flow; (b) equivalent electrical circuit based on thermal resistance.

11.4 Packaging Substrates

An IC package falls into two basic categories. In the first, a single-layer type, the package is constructed around the IC chip on a lead frame. In the second, a multilayer type, the IC chip is assembled into a prefabricated package.

In a single-layer technology, the IC chip is first mechanically bonded to a lead frame, and then electrically interconnected with fine wires from the chip bond pads to the corresponding lead-frame fingers. The final package is then constructed around the lead-frame subassembly. Two single-layer technologies are used in the industry: molded plastic and glass-sealed pressed ceramic.

Plastic Packaging

Plastic is a generic term for a host of man-made organic polymers. Polymer materials are relatively porous structures, which may allow absorption or transport of water molecules and ions.[19] The aluminum metallization is susceptible to rapid corrosion in the presence of moisture, contaminants, and electric fields. So, plastic packages are not very reliable. Impurities from the plastic or other materials in the construction of the package can cause threshold shifts or act as catalysts in metal corrosion. Fillers can also affect reliability and thermal performance of the plastic package.

Ceramic Packaging

Pressed ceramic technology packages are used mainly for economically encapsulating ICs and semiconductor devices requiring hermetic seals. Hermeticity means that the package must pass both gross and fine leak tests and also exclude environmental contaminants and moisture for a long period of time. Further, any contaminant present before sealing must be removed to an acceptable level before or during the sealing process.[20] Silicon carbide (SiC), aluminum nitride, beryllia (BeO), and alumina (Al_2O_3) are some of the ceramics used in electronic packaging. In comparison with other ceramics, SiC has a thermal expansion coefficient closer to silicon, and as a result less stress is generated between the *dice* and the substrate during temperature cycling. In addition, it has a very high thermal conductivity. These two properties make SiC a good packaging substrate and a good heat sink that can be bonded directly to silicon *dice* with little stress generation at elevated temperatures. Its high dielectric constant, however, makes it undesirable as a substrate to carry interconnections. Alumina and BeO have properties similar to SiC.[21]

TABLE 11.2A Thermal and Electrical Properties of Materials Used in Packaging

Metals	Coefficient of Thermal Expansion (CTE) (10^{-6} K^{-1})	Thermal Conductivity (W/cm-K)	Specific Electrical Resistance $10^{-6}\Omega$-cm
Aluminum	23	2.3	2.8
Silver	19	4.3	1.6
Copper	17	4.0	1.7
Molybdenum	5	1.4	5.3
Tungsten	4.6	1.7	5.3

Insulating Substrates	Coefficient of Thermal Expansion (CTE) (10^{-6} K^{-1})	Thermal Conductivity (W/cm-K)	Dielectric Constant
Alumina (Al_2O_3)	6.0	0.3	9.5
Beryllia (BeO)	6.0	2.0	6.7
Silicon carbide (SiC)	3.7	2.2	42
Silicon dioxide (SiO_2)	0.5	0.01	3.9

Semiconductors	Coefficient of Thermal Expansion (CTE) (10^{-6} K^{-1})	Thermal Conductivity (W/cm-K)	Dielectric Constant
Silicon	2.5	1.5	11.8
Germanium	5.7	0.7	16.0
Gallium arsenide	5.8	0.5	10.9

TABLE 11.2B Some Properties of Ceramic Packaging Materials

Property	BeO	AlN	Al_2O_3 (96%)	Al_2O_3 (99.5%)
Density (g/cm^3)	2.85	3.28	3.75	3.8
CTE (ppm/K)	6.3	4.3	7.1	7.1
TC (W/cm-K)	285	180	21	25.1
Dielectric const.	6.7	10	9.4	10.2
Loss tangent	0.0001	0.0005	0.0001	0.0001

11.5 Package Types

IC packages have been developed over time to meet the requirements of high speed and density. The history of IC package development has been the continuous battle to miniaturize.[22,23] Figure 11.3 illustrates the size and weight reduction of IC package over time.

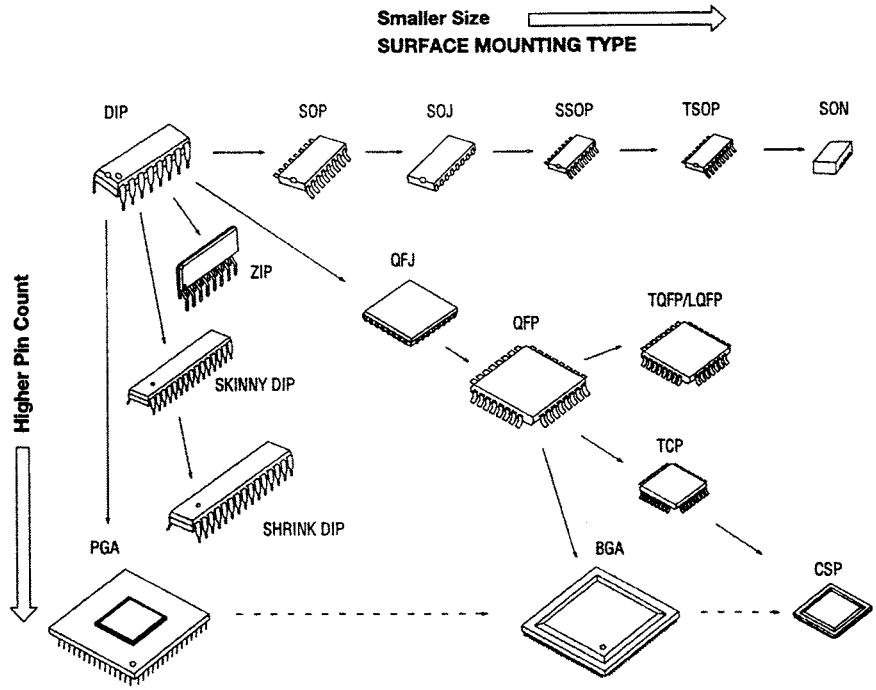


FIGURE 11.3 Packaging trends.

Several packages can be classified as follows.

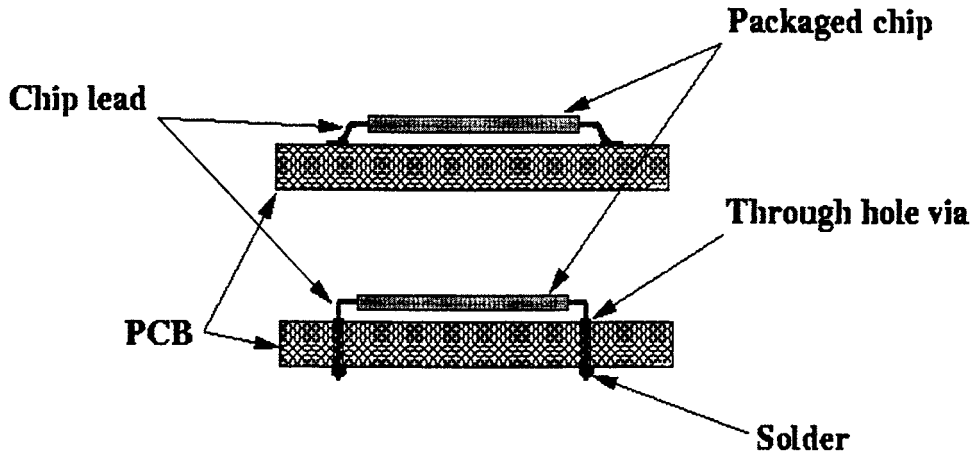
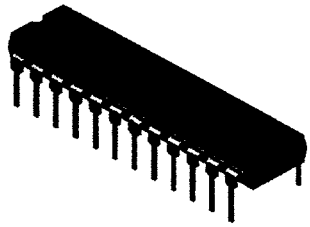


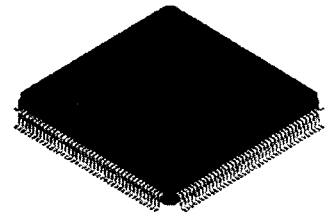
FIGURE 11.4 A generic schematic diagram showing the difference between the surface-mount technology (upper) and through hole mounting (lower).

Through Hole Packages

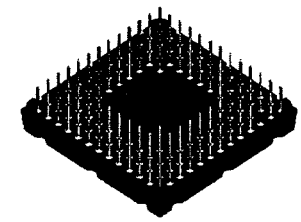
Through-the-board hole mounting technology uses precision holes drilled through the board and plated with copper. This copper plating forms the connections between separate layers. These layers consist of thin copper sheets stacked together and insulated by epoxy fiberglass. There are no dedicated via structures to make connections between wiring levels; through holes serve that purpose. Through holes form a sturdy support for the chip carrier and resist thermal and mechanical stresses caused by the variations in the expansions of components at raised temperatures. Different types (see Fig. 11.5) of through hole packages can be further classified as:



(a) Dual in-line package (DIP).



(b) Quad flat package (QFD).



(c) Pin grid array package (PGA).

FIGURE 11.5 Different through mount packages.

Dual-in-Line Packages (DIPs)

A dual-in-line package is a rectangular package with two rows of pins in its two sides. Here, first the die is bonded on the lead frame and in the next step, chip I/O and power/ground pads are wire-bonded to the lead frame, and the package is molded in plastic. DIPs are the workhorse of the high-volume and general-purpose logic products.

Quad Flat Packages (QFPs)

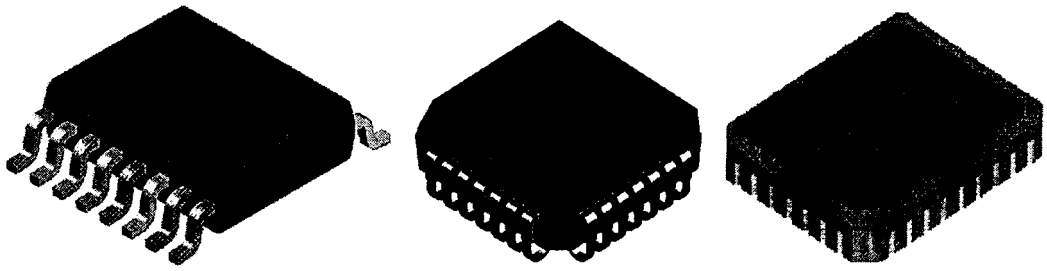
With the advances in VLSI technology, the lower available pin counts of the rectangular DIP became a limiting factor. With pins spaced 2.4 mm apart on only two sides of the package, the physical size of the DIP has become too great. On the other hand, the physical size of an unpackaged microelectronic circuit (bare die) has been reduced to a few millimeters. As a result, the DIP package has become up to 50 times larger than the bare die size itself, thus defeating the objective of shrinking the size of the integrated circuits. So, one solution is to provide pins all around. In QFPs, pins are provided on all four sides. Thin QFPs are developed to reduce the weight of the package.

Pin Grid Arrays (PGA)

A pin grid array has leads on its entire bottom surface rather than only at its periphery. This way it can offer a much larger pin count. It has cavity-up and cavity-down versions. In a cavity-down version, a die is mounted on the same side as the pins facing toward the PC board, and a heat sink can be mounted on its backside to improve the heat flow. When the cavity and the pins are on the same side, the total number of pins is reduced because the area occupied by the cavity is not available for brazed pins. The mounting and wire bonding of the dice are also more difficult because of the existence of the pins next to the cavity. High pin count and larger power dissipation capability of PGAs make them attractive for different types of packaging.

Surface-Mounted Packages

Surface mounting solves many of the shortcomings of through-the-board mounting. In this technology, a chip carrier is soldered to the pads on the surface of a board without requiring any through holes. The smaller component sizes, lack of through holes, and the possibility of mounting chips on both sides of the PC board improve the board density. This reduces package parasitic capacitances and inductances associated with the package pins and board wiring. Various types of surface-mount packages are available on the market and can be divided into the following categories (see Fig. 11.6):



(a) Small outline package (SOP). **(b) Plastic-leaded chip carriers (PLCC).** **(c) Leadless ceramic chip carriers (LCCC).**

FIGURE 11.6 Different surface-mount packages.

Small-Outline Packages (SOPs)

The small-outline package has gull-wing shaped leads. It requires less pin spacing than through-hole-mounted DIPs and PGAs. SOP packages usually have small lead counts and are used for discrete, analog, and SSI/MSI logic parts.

Plastic-leaded Chip Carriers (PLCCs)

Plastic-leaded chip carriers, such as gull-wing and J-leaded chip carriers, offer higher pin counts than SOP. J-leaded chip carriers pack denser and are more suitable for automation than gull-wing leaded carriers because their leads do not extend beyond the package.

Leadless Ceramic Chip Carriers (LCCCs)

Leadless ceramic chip carriers take advantage of multilayer ceramic technology. The conductors are left exposed around the package periphery to provide contacts for surface mounting. *Dice* in leadless chip carriers are mounted in cavity-down position, and the back side of the chip faces away from the board, providing a good heat removal path. The ceramic substrate also has a high thermal conductivity. LCCCs are hermetically sealed.

Flip-Chip Packages

The length of the electrical connections between the chip and the substrate can be minimized by placing solder bumps on the *dice*, flipping the chips over, aligning them with the contacts pads on the substrate, and reflowing the solder balls in a furnace to establish the bonding between the chips and the package. This method provides electrical connections with minute parasitic inductance and capacitance. In addition, contact pads are distributed over the entire chip surface. This saves silicon area, increases the maximum I/O and power/ground terminals available with a given die size, and provides more efficiently routed signal and power/ground interconnections on the chips.[24] (see Fig. 11.7.)

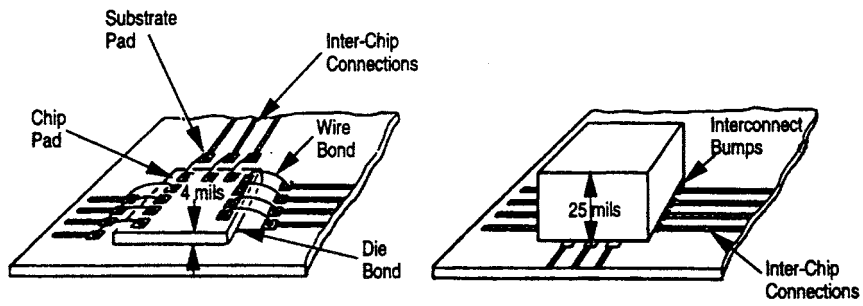


FIGURE 11.7 Flip-chip package and its interconnections.

Chip Size Packages (CSPs)

To combine the advantages of both packaged chip and bare chip in one solution, a variety of CSPs have been developed.[25,26] CSPs can be divided into two categories: the fan-in type and the fan-out type.

Fan-in type CSPs are suitable for memory applications that have relatively low pin counts. This type is further divided into two types, depending on the location of bonding pads on the chip surface; these are the center pad type and the peripheral pad type. This type of CSP keeps all the solder bumps within the chip area by arranging bumps in area array format on the chip surface.

The fan-out CSPs are used mainly for logic applications: because of the die size to pin count ratio, the solder bumps cannot be designed within the chip area.

Multi-Chip Modules (MCMs)

In a multi-chip module, several chips are supported on a single package. Most multi-chip packages are made of ceramic. By eliminating one level of packaging, the inductance and capacitance of the electrical connections among the *dice* are reduced. Usually, the *dice* are mounted on a multilayer ceramic substrate via solder bumps, and the ceramic substrate offers a dense interconnection network.[27,28] (See Fig. 11.8.) There are several advantages of multi-chip modules over single-chip carriers. The multi-chip module minimizes the chip-to-chip spacing and reduces the inductive and capacitive discontinuities between the chips mounted on the substrate by replacing the die-bump-interconnect-bump-die path. In addition, narrower and shorter wires on the ceramic substrate have much less capacitance and inductance than the PC board interconnections.

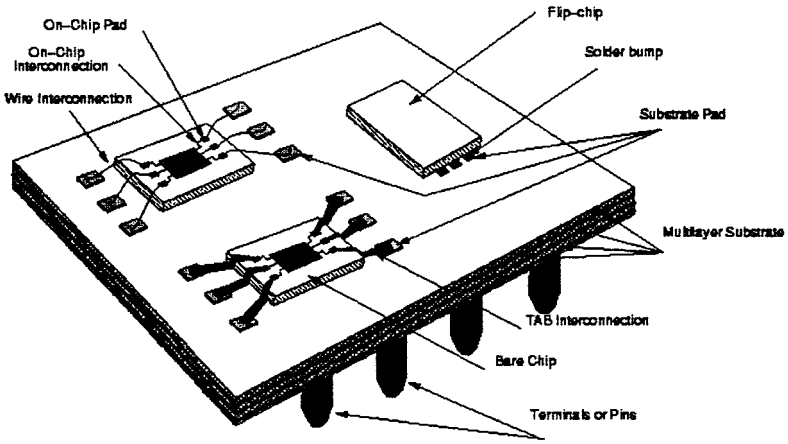


FIGURE 11.8 A generic schematic diagram of an MCM, showing how bare *dice* are interconnected to an MCM substrate using different interconnection technologies.

3-D VLSI Packaging

The driving forces behind the development of three-dimensional packaging technology are similar to the multi-chip module technology, although the requirements for the 3-D technology are more aggressive. These requirements include the need for significant size and weight reductions, higher performance, small delay, higher reliability, and potentially reduced power consumption.

Silicon-on-Silicon Hybrid

A silicon substrate can also be used as an interconnection medium to hold multi-chips as an alternative to ceramic substrates. This is called silicon-on-silicon packaging or, sometimes, hybrid wafer-scale

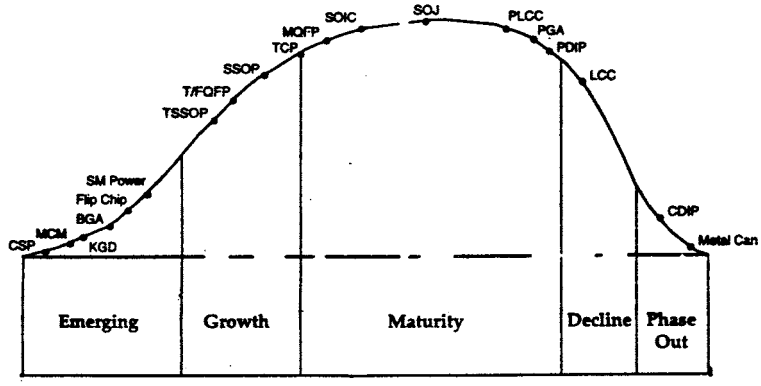


FIGURE 11.9 IC packaging life cycle.

integration. Thin-film interconnections are fabricated on a wafer and separately processed, and test dice are mounted on this silicon substrate via wire bonding, TAB, or solder bumps. Using this technique, chips fabricated in different technologies can be placed on the same hybrid package. The silicon substrate can also potentially contain active devices that serve as chip-to-chip drivers, bus and I/O multiplexers, and built-in test circuitry.[29]

11.6 Hermetic Packages

In hermetic packages, the packaged cavity is sealed in such a way that all external chemical species are permanently prevented from entering into it. In practice, however, a finite rate of leakage occurs through diffusion and permeation. Moisture is the principal cause of device failures. Moisture by itself does not cause electronic problems when trapped in an electronic package, because it is a poor electrical conductor. However, water can dissolve salts and other polar molecules to form an electrolyte, which together with the metal conductors and the potential difference between them, can create leakage paths as well as corrosion problems. Moisture is contributed mainly by the sealing ambient, the absorbed and dissolved water from the sealing materials, lid and the substrate and the leakage of external moisture through the seal. No material is truly hermetic to moisture. The permeability to moisture of glasses, ceramics, and metals, however, is very low and is orders of magnitude lower than for any plastic material. Hence, the only true hermetic packages are those made of metals, ceramics, and glasses. The common feature of hermetic packages is the use of a lid or a cap to seal in the semiconductor device mounted on a suitable substrate. The leads entering the package also need to be hermetically sealed.

11.7 Die Attachment Techniques

To provide electrical connections between the chip pads and package, different bonding techniques are used. These can be classified as follows.

Wire Bonding

Wire bonding (see Fig. 11.10) is a method used to connect a fine wire between an on-chip pad and a substrate pad. This substrate may simply be the ceramic base of a package or another chip. The common materials used are gold and aluminum. The main advantage of wire bonding technology is its low cost; but it cannot provide large I/O counts, and it needs large bond pads to make connections. The connections have relatively poor electrical performance.

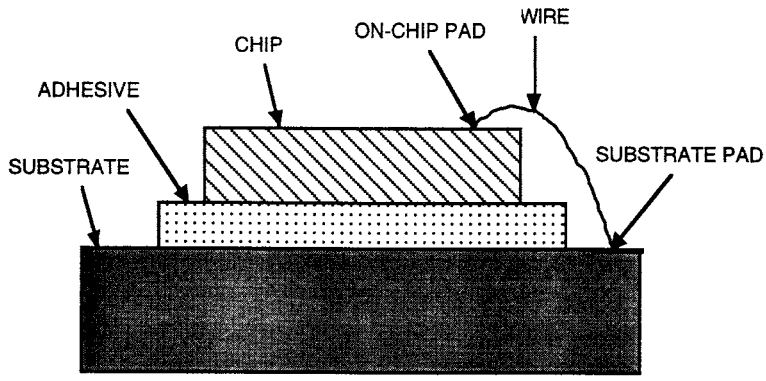


FIGURE 11.10 Wire bonding assembly.

Tape-Automated Bonding

In tape-automated bonding (TAB) technology, a chip with its attached metal films is placed on a multilayer polymer tape. The interconnections are patterned on a multilayer polymer tape. The tape is positioned above the "bare die" so that the metal tracks (on the polymer tape) correspond to the bonding sites on the die (Fig. 11.11). TAB technology provides several advantages over wire bonding technology. It requires a smaller bonding pad, smaller on-chip bonding pitch, and a decrease in the quantity of gold used for bonding.[30] It has better electrical performance, lower labor costs, higher I/O counts and lighter weight, greater densities, and the chip can be attached in a face-up or face-down configuration. TAB technology includes time and cost of designing and fabricating the tape and the capital expense of the TAB bonding equipment. In addition, each die must have its own tape patterned for its bonding configuration. Thus, TAB technology has typically been limited to high-volume applications.

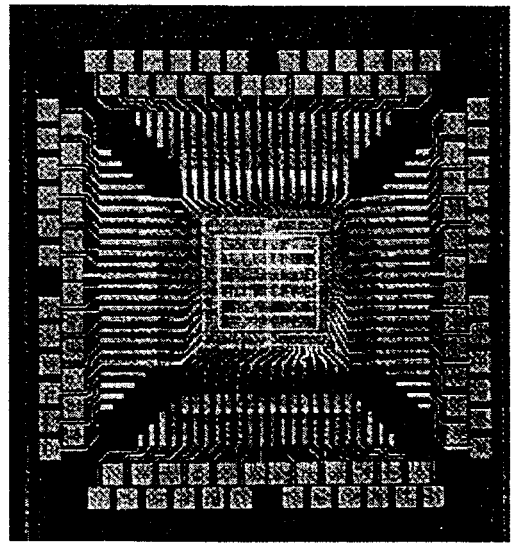


FIGURE 11.11 Tape-automated bonded die.

Solder Bump Bonding

Solder bumps are small spheres of solder (solder balls) that are bonded to contact areas or pads of semiconductor devices and subsequently used for face-down bonding. The length of the electrical

connections between the chip and the substrate can be minimized by placing solder bumps on the die, flipping the die over, aligning the solder bumps with the contact pads on the substrate, and re-flowing the solder balls in a furnace to establish the bonding between the die and the substrate[31] (Fig. 11.12). This technology provides electrical connections with minute parasitic inductances and capacitances. In addition, the contact pads are distributed over the entire chip surface rather than being confined to the periphery. As a result, the silicon area is used more efficiently, the maximum number of interconnects is increased, and signal interconnections are shortened. But this technique results in poor thermal conduction, difficult inspection of the solder bumps, and possible thermal expansion mismatch between the semiconductor chips and the substrate.

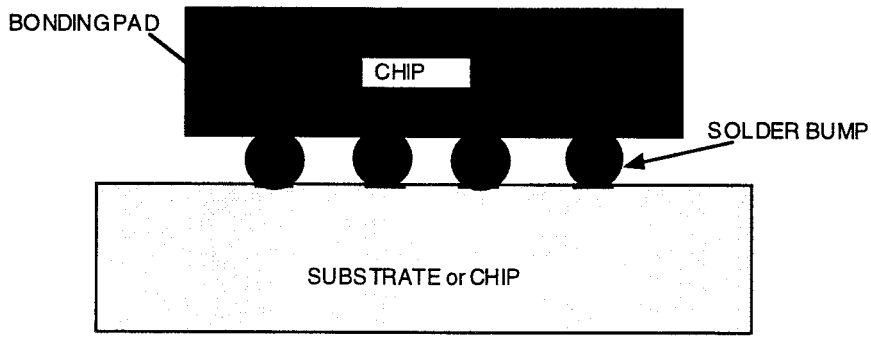


FIGURE 11.12 Flip-chip method using solder bumps.

11.8 Package Parasitics

Typically, the electrical interconnection of a chip in a package consists of chip-to-substrate interconnect, metal runs on the substrate, and finally, pins from the package. Associated with these are the electrical resistance, inductance and capacitance — referred to as package parasitics. The electrical parasitics are determined by the physical parameters such as interconnect width, thickness, length, spacing, and resistivity; by the thickness of the dielectric; and by the dielectric constant.

Resistance refers to both dc and ac. The dc resistance of an interconnect is a property of its cross-sectional area, length, and material resistivity. In addition, the ac resistance depends on the frequency of the signal and is higher than the dc resistance because of the skin effect. Resistance in the power distribution path results in attenuation of input signals to the device and output signals from the device. This has the effect of increasing the path delay.

Capacitance of an interconnect is a property of its area, the thickness of the dielectric separating it from the reference potential, and the dielectric constant of the dielectric. It is convenient to consider this as two parts: capacitance with respect to ground, and capacitance with respect to other interconnections. The capacitance with respect to ground is referred to as the *load capacitance*. This is seen as part of the load by the output driver and thus can slow down the rise time of the driver. Interlead capacitance couples the voltage change on the active interconnect to the quiet interconnect.[32] This is referred to as *crosstalk*.

Inductance can be defined only if the complete current path is known. In the context of component packages, the inductance of an interconnect should be understood as part of a complete current loop. Thus, if the placement of the package in the system alters the current path in the package, the package inductance will vary. Total inductance consists of self-inductance and mutual inductance. Mutual inductance between two interconnects generates a voltage in one when there is current change in the other. Inductive effects are the leading concern in the design of power distribution path in high-performance packages. They are manifested as “ground bounce” noise and “simultaneous switching” noise.

11.9 Package Modeling

As the complexity of devices increases, design and development efforts for packages become comparable to design and development efforts for chips. Many package design concepts must be simulated to assess their associated performance parameters.[33,34] Computer-aided design software and test chips are becoming indispensable design tools. Computer-aided design tools are extensively used to analyze the thermal, thermomechanical, mechanical, and electrical parameters of packages[35]; for example, electrical modeling extracts an equivalent electrical circuit that describes the physical structure of the package and, hence, the equivalent electrical circuit of the package can be used in circuit simulation programs to evaluate the overall performance of a packaged circuit. Until now, the equivalent electrical circuit incorporated only lumped electrical parameters; but as frequency of operation of the circuits is increasing, the distributed model of the package needs to be developed for high-frequency simulations.[36]

11.10 Packaging in Wireless Applications

Wireless applications typically involve RF, high-frequency digital, and mixed-mode circuits. Wireless packaging requires minimal electrical parasitic effects that need to be well-characterized.

In wireless applications, the trend is to integrate multiple modules on a single chip.[37] So, the thermal management of the whole chip becomes crucial. The IC package must have good thermal properties. Metal as a material shows optimal properties concerning thermal conductivity, electromagnetic shielding, mechanical and thermal stability. For thermal expansion, best match to semiconductor and ceramic material can be achieved with molybdenum, tungsten, or special composites like kovar. Ceramic materials are applied, both as parts of the package as well as for subsystem carrying RF transmission lines. To this end, and to provide electromagnetic shielding, these materials partly have to be metallized. Aluminum nitride, beryllia, aluminum silicon carbide, and CVD diamond show best thermal conductivity and are therefore applied in high-power applications,[38,39] while alumina is well known for standard microwave applications.[40]

Integration of passive components is a major challenge in wireless packages. More and more efforts are being made to integrate passive components, power devices on a chip with the other mixed signal circuits.[41] The size of the package becomes an issue. Micromachining technology provides a way to make miniature packages that conform to RF circuits, while providing physical and electrical shielding. Conformal packages made by applying micromachining technology provide the capability to isolate individual circuit elements and improve circuit performance by eliminating the radiation and cross-coupling between the adjacent circuits.[42,43]

At high frequencies, interconnections need to be carefully designed. Microstrip interconnects, coplanar waveguide are mostly used for microwave packaging.[44] Flip-chip packaging has tremendous potential for future RF packaging.[45]

11.11 Future Trends

Packaging is the bridge between silicon and electronics system. Packaging design and fabrication are increasingly important to system applications. Consideration of factors affecting waveform integrity for both power and signal (i.e., timing, cross-talk, and ground bounce) will affect device layout on the chip, chip layout on the package, and interconnect.

Conventional surface mount packages will dominate in the region of low pin count and low clock frequency. Ball-grid array packages and chip-scale packages will be used for medium pin counts. Technically bare chip solutions can cover the whole area, but have a reliability versus cost tradeoff. Bare chip solutions could be very competitive with packaged solutions, as they can accomplish very high density and very good electrical performance.

Packaging needs are driven as much by market application requirements as by silicon technology. Cost drives technology tradeoffs for all market segments. As the complexity of package technology continues to increase, new materials will be needed to meet design and performance challenges. Significant engineering development will be needed for power increases at each technology generation.

An integrated design environment of physical, electrical, thermal, thermo-mechanical, chip, package, and system design needs to be evolved. Most of these integrated solutions will provide modeling and simulation capabilities that will be embodied in packaging computer aided design systems. Design tools are required to manage the complexity of packaging that is being pushed to its performance limits.

References

1. Tummalam, R. R. and Klopfenstein, A. G., *Microelectronics Packaging Handbook*, Van Nostrand Reinhold, New York, 1989.
2. Bakoglu, H. B., *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, New York, 1990.
3. Tummala, R. R., "Electronic Packaging in the 1990s — A Perspective from America," *IEEE Trans. Components, Hybrids, Manuf. Technol.*, vol. 14, no. 2, pp. 262, June 1991.
4. Wessely, H., "Electronic Packaging in the 1990s — A Perspective from Europe," *IEEE Trans. Components, Hybrids, Manuf. Technol.*, vol. 14, no. 2, pp. 272, June 1991.
5. Mones, A. H. and Spielberger, R. K., "Interconnecting and Packaging VLSI Chips," *Solid State Technology*, pp. 119-122, Jan. 1984.
6. Kakei, M., "Low Stress Molding Compounds for VLSI Devices," *Nikkei Microdevices*, 1984.
7. Lyman, J., "VLSI Packages are Presenting Diversified Mix," *Electronics*, pp. 67-73 Sept. 1984.
8. Bakoglu, H. B., "Packaging for High-Speed System," *IEEE International Solid-State Circuits Conference (ISSCC'98)*, pp. 100-101, San Francisco, Feb. 1988.
9. Kaupp, H. R., "Characteristics of Microstrip Transmission Line," *IEEE Trans. on Computers*, EC-16, pp. 185, April 1967.
10. Trivedi, M. and Shenai, K., "Framework for Power Package Design and Optimization," *Intl. Workshop on Integrated Power Packaging (IWIPP'98)*, 1998, Chicago, IL.
11. Khandelwal, P., Trivedi, M., and Shenai, K., "Thermal Issues in LDMOSFET's Packages," *European Solid-State Device Research Conference (ESSDRC)*, 1998.
12. Manchester K. and Bird, K. , "Thermal Resistance: A Reliability Consideration," *IEEE Trans. Components, Hybrids, Manuf. Technol.,* vol. 31, no. 4, pp. 550, Dec. 1980.
13. Chu, R. C., Hwang, U. P., and Simons, R. E., "Conduction Cooling for LSI Packages, A One-Dimensional Approach," *IBM Journal of Research and Development*, vol. 26, no. 1, pp. 45-54, Jan. 1982.
14. Mohan, N., Undeland, T. M., and Robbins, W. P., *Power Electronics: Converters, Applications, and Design*, John Wiley & Sons, 1996.
15. Fukuzawa, I., "Moisture Resistance Degradation of Plastic LSIs by Reflow Soldering," *Proc. International Reliability Physics Symposium*, pp. 192, 1985.
16. Lau, J. H., *Solid Joint Reliability: Theory and Applications*, Van Nostrand Reinhold, New York, 1991.
17. Gallace, L. J. and Rosenfield, M., "Reliability of Plastic Encapsulated Integrated Circuit in Moisture Environments," *RCA Review*, vol. 45, no. 2, pp. 95-111, June 1984.
18. Lau, J. H., "Thermal Stress Analysis of SMT PQFP Packages and Interconnections," *J. Electronic Packaging, Trans. of ASME*, vol. 2, pp. 111, March 1989.
19. Kawai, S., "Structure Design of Plastic IC Packages," *Proc. SEMI Tech. Symposium*, pp. 349, Nov. 1988.
20. White, M. L., "Attaining Low Moisture Levels in Hermetic Packages," *Proc. 20th International Reliability Physics Symposium*, pp. 253, 1982.
21. Sepulveda, J. L. and Siglianu, R., "BeO Packages House High-Power Components," *Microwaves & RF*, pp. 111-124, March 1998.
22. Fehr, G., Long, J., and Tippetts, A., "New Generation of High Pin Count Packages," *Proc. IEEE Custom Integrated Circuits Conference (CICC'85)*, pp. 46-49, 1985.

23. Sudo, T., "Considerations of Package Design for High Speed and High Pin Count CMOS Devices," *Proc. 39th Electronic Components and Technology Conference*, 1989.
24. Midford, T. A., Wooldridge, J. J., and Sturdivant, R. L., "The Evolution of Packages for Monolithic Microwave and Millimeter Wave Circuits," *IEEE Trans. on Antennas and Propagation*, vol. 43, no. 9, pp. 983, Sept. 1995.
25. Yamaji, Y., Juso, H., Ohara, Y., Matsune, Y., Miyata, K., Sota, Y., Narai, A., and Kimura, T., "Development of Highly Reliable CSP," *Proc. 47th Electronic Components and Technology Conference*, pp. 1022-1028, 1997.
26. Su, L. S., Louis, M., and Reber, C., "Cost Analysis of Chip Scale Packaging," *Proc. 21st IEEE/CPMT Intl. Electronics Manufacturing Technology Symposium*, pp. 216-223, 1997.
27. Lyman, J., "Multichip Modules Aim at Next Generation VLSI," *Electronic Design*, pp. 33-34, March 1989.
28. Barlett, C. J., Segelken, J. M., and Teneketgen, N. A., "Multichip Packaging Design for VLSI Based Systems," *IEEE Trans. Components, Hybrids, Manuf. Technol.,* vol. 12, no. 4, pp. 647-653, Dec. 1987.
29. Spielberger, R. K., Huang, C. D., Nunne, W. H., Mones, A. H., Fett, D. C., and Hampton, F. L., "Silicon-on-Silicon Packaging," *IEEE Trans. Components, Hybrids, Manuf. Technol.,* vol. 7, no. 2, pp. 193-196, June 1984.
30. Andrews, W., "High Density Gate Arrays Tax Utility, Packaging, and Testing," *Computer Design*, pp. 43-47, Aug. 1988.
31. Fujimoto, H., "Bonding of Ultrafine Terminal-Pitch LSI by Micron Bump Bonding Method," *Proc. IMC*, pp. 115, 1992.
32. Dang, R. L. M. and Shigyo, N., "Coupling Capacitance for Two-Dimensional Wires," *IEEE Electron Device Letters*, vol. EDL-2, pp. 196-197, Aug. 1981.
33. Jackson, R. W., "A Circuit Topology for Microwave Modeling of Plastic Surface Mount Packages," *IEEE Trans. on Microwave Theory and Techniques*, vol. 44, pp. 1140-1146, 1997.
34. Gupta, R., Allstot, D. J. , and Meixner, R., "Parasitic-Aware Design and Optimization of CMOS RF Integrated Circuits," *IEEE MTT-S International Symposium*, vol. 3, pp. 1867-1870, 1998.
35. Perugupalli, P., Xu, Y., and Shenai, K., "Measurement of Thermal and Packaging Limitations in LDMOSFETs for RFIC Applications," *IEEE Instrumentation and Measurement Technology Conference (IMTC),* pp. 160-164, May 1998.
37. Raid, S. M., Su, W., Salma, I., Riad, A. E., Rachlin, M. , Baker, W., and Perdue, J., "Plastic Packaging Modeling and Characterization at RF/Microwave Frequencies," *Proc. of 3rd International Symposium on Advanced Packaging Materials*, pp. 147, Mar. 1997.
38. Bugeau, J. L. , Heitkamp, K. M., and Kellerman, D., "Aluminum Silicon Carbide for High Performance Microwave Packages," *IEEE MTT-S International Symposium*, vol. 3, pp. 1575-1578, 1995.
39. Gomes-Casseres, M., and Fabis, P. M., "Thermally Enhanced Plastic Package Using Diamond for Microwave Applications," *IEEE MTT-S International Symposium*, vol. 1, pp. 227-230, 1996.
40. Kemerley, R. T., "Emerging Microwave Packaging Technologies," *IEEE 3rd Tropical Meeting on Electrical Performance of Electronic Packaging*, pp. 139-242, 1994.
41. Smith, C. R., "Power Module Techmology for the 21st Century," *Proc. of the IEEE National Aerospace and Electronics Conference*, vol. 1, pp. 106-113, 1995.
42. Drayton, R. F. and Katehi, L.P.B., "Micromachined Conformal Packages for Microwave and Millimeter-Wave Applications," *IEEE MTT-S International Symposium*, vol. 3, pp. 1387-1390, 1995.
43. Drayton, R. F., Henderson, R. M., and Katehi, L.P.B., "Advanced Monolithic Packaging Concepts for High Performance Circuits and Antennas," *IEEE MTT-S International Symposium*, vol. 3, pp. 1615-1618, 1996.
44. Wein, D. S., "Advanced Ceramic Packaging for Microwave and Millimeter Wave Applications," *IEEE Trans. on Antennas and Propogation*, vol. 43, pp. 940-948, Sept. 1995.
45. Krems, T., Haydll, W. H., Massler, H., and Rudiger, J., "Advances of Flip Chip Technology in Millimeter-Wave Packaging," *IEEE MTT-S International Symposium*, vol. 2, pp. 987-990, 1997.

Choma Jr., J., et al. "Multichip Module Technologies"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

12

Multichip Module Technologies

Victor Boyadzhyan
Jet Propulsion Laboratory

John Choma, Jr.
University of Southern California

12.1 Introduction

From the pioneering days to its current renaissance, the electronics industry has become the largest and most pervasive manufacturing industry in the developed world. Electronic products have the hallmark of innovation, creativity, and cost competitiveness in the world market place. The way the electronics are packaged, in particular, has progressed rapidly in response to customers' demands in general for diverse functions, cost, performances, and robustness of different products. For practicing engineers, there is a need to access the current state of knowledge in design and manufacturing tradeoffs.

Thus arises a need for electronics technology-based knowledge to optimize critical electronic design parameters such as speed, density, and temperature, resulting in performance well beyond PC board design capabilities. By removing discrete component packages and using more densely packed interconnects, electronic circuit speeds increase. The design challenge is to select the appropriate packaging technology, and to manage any resulting thermal problems.

The expanding market for high-density electronic circuit layouts calls for multi-chip modules (MCMs) to be able to meet the requirements of fine track and gap dimensions in signal layers, the retention of accurately defined geometry in multilayers, and high conductivity to minimize losses. Multi-chip module technologies fill this gap very nicely. This chapter provides engineers/scientists with an overview of existing MCM technologies and briefly explains similarities and differences of existing MCM technologies. The text is reinforced with practical pictorial examples, omitting extensive development of theory and details of proofs.

The simplest definition of a multi-chip module (MCM) is that of a single electronic package containing more than one integrated circuit (IC) die.[1] An MCM combines high-performance ICs with a custom-designed common substrate structure that provides mechanical support for the chips and multiple layers of conductors to interconnect them.

One advantage of this arrangement is that it takes better advantage of the performance of the ICs than it does interconnecting individually packaged ICs because the interconnect length is much shorter. The really unique feature of MCMs is the complex substrate structure that is fabricated using multilayer ceramics, polymers, silicon, metals, glass ceramics, laminates, etc. Thus, MCMs are not really new. They have been in existence since the first multi-chip hybrid circuit was fabricated. Conventional PWBs utilizing chip-on-board (COB), a technique where ICs are mounted and wire-bonded directly to the board, have also existed for some time. However, if packaging efficiency (also called silicon density), defined as the percentage of area on an interconnecting substrate that is occupied by silicon ICs, is the guideline used to define an MCM, then many hybrid and COB structures with less than 30% silicon density do not qualify as MCMs. In combination with packaging efficiency, a minimum of four conductive layers and 100 I/O leads has also been suggested as criteria for MCM classification.[1]

A formal definition of MCMs has been established by the Institute for Interconnecting and Packaging Electronic Circuits (IPC). They defined three primary categories of MCMs: MCM-L, MCM-C, and MCM-D.

It is important to note that these are simple definitions. Consequently, many IC packaging schemes, which technically do not meet the criteria of any of the three simple definitions, may incorrectly be referred to as MCMs. However, when these simple definitions are combined with the concept of packaging efficiency, chip population, and I/O density, there is less confusion about what really constitutes an MCM. The fundamental (or basic) intent of MCM technology is to provide an extremely dense conductor matrix for the interconnection of bare IC chips. Consequently, some companies have designated their MCM products as high-density interconnect (HDI) modules.

12.2 Multi-Chip Module Technologies

From the above definitions, it should be obvious that MCM-Cs are descended from classical hybrid technology, and MCM-Ls are essentially highly sophisticated printed circuit boards, a technology that has been around for over 40 years. On the other hand, MCM-Ds are the result of manufacturing technologies that draw heavily from the semiconductor industry.

MCM-L

Modules constructed of plastic laminate-based dielectrics and copper conductors utilizing advanced forms of printed wiring board (PWB) technologies to form the interconnects and vias are commonly called "laminate MCMs," or MCM-Ls.[2]

Advantages	
Economic	Ability to fabricate circuits on large panels with a multiplicity of identical patterns. Reduces manufacturing cost. Quick response to volume orders.

Disadvantages	
Technological	More limited in interconnect density relative to advanced MCM-C and MCM-D technologies. Copper slugs and cut-outs are used in MCM-Ls for direct heat transfer. This degrades interconnection density.

MCM-L development has involved evolutionary technological advances to shrink the dimensions of interconnect lines and vias. From a cost perspective, it is desirable to use conventional PWB technologies for MCM-L fabrication. This is becoming more difficult as the need for multi-chip modules with higher interconnect density continues.

As MCM technologies are being considered for high-volume consumer products applications, a focus on containing the cost of high-density MCM-Ls is becoming critical.

The most usefull charateristic in assessing the relative potential of MCM-L technology is interconnection density,[3,4] which is given by:

$$\text{Packaging efficiency (\%)} = \text{Silicon chip area/Package area} \tag{12.1}$$

The above formula measures how much of the surface of the board can be used for chip mounting pads versus how much must be avoided because of interconnect traces and holes/pads.

MCM-C

These are modules constructed on co-fired ceramic or glass-ceramic substrates using thick-film (screen printing) technologies to form the conductor patterns using fireable metals. The term "co-fired" implies that the conductors and ceramic are heated at the same time. These are also called thick-film MCMs.

Ceramic technology for MCMs can be divided into four major categories

- Thick-film hybrid process
- High-temperature co-fired alumina process (HTCC)
- Low-temperature co-fired ceramic/glass based process (LTCC)
- High T_c aluminum nitride co-fired substrate (AIN)

Thick-film hybrid technology produces by the successive deposition of conductors, dielectric, and/or resistor patterns onto a base substrate.[5] The thick-film material, in the form of a paste, is screenprinted onto the underlying layer, then dried and fired. The metallurgy chosen for a particular hybrid construction depends on a number of factors, including cost sensitivity, conductivity requirements, solderability, wire bondability, and more. A comparative summary of typical ceramic interconnect properties is compiled in Table 12.1.

TABLE 12.1 A Comparative Summary of Typical Ceramic Interconnect Properties

Item	Thick Film	HTCC	LTCC
Line width (μm)	125	100	100
Via diameter (μm)	250	125	175
Ave. No. conductor layers	1–6	1–75	1–75
Conductor res. (mohm/sq)	2–100	8–12	3–20
ε (dielectric)	5–9	9–10	5–8
CTE	4–7.5	6	3–8
T_c Dielectric (W/mC)	2	15–20	1–2
Relative cost (low volume)	Medium	High	High
Tooling costs	Low	High	High
Capital outlay	Low	High	Medium

12.3 Materials for HTCC Aluminum Packages

Metal conductors of tungsten and molybdenum are used for compatibility in co-firing to temperatures of 1600°C. Materials compatibility during co-firing dictates that raw materials of alumina with glass used for densification and any conductor metal powders (W, Mo) must be designed to closely match onset, rate, and volume shrinkage; promote adhesion; and minimize thermal expansion mismatch between conductor and dielectric.

Processing HTCC Ceramics

The raw materials used in fabrication of aluminum substrates include aluminum oxide, glass, binder, plasticizer, and solvent. Materials specifications are used to control alumina particle size, surface area, impurity, and agglomeration. Glass frit is controlled through composition, glass transition and softening point, particle size, and surface area. Molecular weight, group chemistry, and viscosity controls are used for the binder and plasticizer.

Metal Powder and Paste

A thick-film paste often uses metal powder, glass powder, organic resins, and solvents. Compositions are varied to control screening properties, metal shrinkage, and conductivity. Paste fabrication begins with batch mixing, dispersion, and deagglomeration, which are completed on a three-roll mill.

Thick-Film Metallization

The greensheet is cast, dried, stripped from the carrier film, and blanked into defect-free sheets, typically 200 mm^2. The greensheet is then processed through punching, screening, and inspection operations.

HTCC in Summary

- Electrical performance characteristics include 50-ohm impedance, low conductor resistance, ability to integrate passive components such as capacitors and inductors, the ability to achieve high wiring density (ease of increasing the number of wiring at low cost), the ability to support high-speed simultaneous switching drivers, and the ease of supporting multiple reference voltages.
- Inherent thermal performance characteristics superior to MCM-L and MCM-D.
- Time-demonstrated reliability.

12.4 LTCC Substrates

The use of glass and glass-ceramics in electronic packaging goes back to the invention of semiconductors. Glasses are used for sealing T-O type packages and CERDIPs, as crossover and inter-level dielectrics in hybrid substrates. The success of co-fired alumina substrates spurred the development of the multilayer glass-ceramic substrates. These advantages derive from the higher electrical conductivity lines of copper, silver, or gold; the lower dielectric constant of the glass ceramic; and the closer CTE match of the substrate to silicon.

Two approaches have been used to obtain glass-ceramic compositions suitable for fabricating self-supporting substrates.[6–8] In the first approach, fine powder of a suitable glass-composition is used that has the ability to sinter well in the glassy state and simultaneously crystallize to become a glass-ceramic. More commonly, mixtures of ceramic powders are used, such as alumna and a suitable glass in nearly equal proportions, to obtain densely sintered substrates.[9,10] Because many glass and ceramic powders can be used to obtain densely sintered glass-ceramic, the actual choice is often made on the basis of other desirable attributes in the resulting glass-ceramic — such as low dielectric constant for lowering the signal propagation delay and coefficient of thermal expansion (CTE) closely matched to the CTE of

silicon to improve the reliability of solder interconnections. Unlike the case of a crystallizing glass, the mixed glass and ceramic approach allows for a much wider choice of materials.

12.5 Aluminum Nitride

Aluminum nitride products are used in a variety of commercial and military applications.

Thermal management with solutions such as AlN can provide superior cooling to ensure reliable system operation. AlN packages typically offer a thermal conductivity of 150 to 200 W/mK, a level which can be compared with many metals or other high thermal conductive materials such as berillia (BeO) or silicon carbide (SiC). AlN has a thermal coefficient of expansion of 4.4 ppm, which is better matched to silicon than to alumina or plastics. Table 12.2 provides a comparison of AlN properties.

TABLE 12.2 AlN Properties Comparison

Item	ALN	HTCC	LTCC	BeO	Si	Cu
Thermal conductivity (W/mK)	175	25	2	260	150	394
Density (g/cm)		3.3	3.9	2.6	2.8	8.9
Dielectric constant (Mhz)	8.9	9.5	5.0	6.7		
Dissipation factor	0.0004	0.0004	0.0002	0.0004		
Bending strength (MaP)	320	420	210	220		

Source: From Ref. 2.

12.6 Materials for Multi-layered AlN Packages

Aluminum nitride is a synthetic compound manufactured by two processes: the carbothermal reduction of alumina (Eq. 12.2) and/or direct nitridation of aluminum metal (Eq. 12.3):

$$Al_2O_3 + 3C + N_2 \rightarrow 2AlN + 3CO \tag{12.2}$$

$$2Al + N_2 \rightarrow 2AlN \tag{12.3}$$

MCM-D

Modules are formed by the deposition of thin-film metals and dielectrics, which may be polymers or inorganic dielectrics. These are commonly called thin-film MCMs.

Here, the focus will be on materials to fabricate the high-density MCM-D interconnect. The materials of construction can be categorized as the thin-film dielectric, the substrate, and the conductor metallization.

12.7 Thin-Film Dielectrics

Dielectrics for the thin-film packaging are polymeric and inorganic. Here, we will try to be brief and informative about those categories. Thin-film packages have evolved to a much greater extent with polymeric materials. The capability offered by polymers include a lower dielectric constant, the ability to form thicker layers with higher speeds, and lower cost of deposition. Polymer dielectrics have been used as insulating layers in recent microelectronics packaging.

12.8 Carrier Substrates

The thin-film substrate must have a flat and polished surface in order to build upon. The substrate should be inert to the process chemicals, gas atmospheres, and temperatures used during the fabrication of the interconnect. Mechanical properties are particularly important because the substrate must be strong

enough to withstand handling, thermal cycling, and shock. The substrate must also meet certain CTE constraints because it is in contact with very large silicon chips on one side and with the package on the other side.[11,12] Thermal conductivity is another important aspect when heat-generating, closely spaced chips need that heat conducting medium. It is informative to state that high-density, large-area processing has generated interest in glass as a carrier material.

Metallic substrates have been used to optimize the thermal and mechanical requirements while minimizing substrate raw material and processing costs. Metallic sandwiches such as Cu/Mo/Cu can be tailored to control CTE and thermal properties. 5%Cu/Mo/5%Cu is reported to have a thermal conductivity (TC) of 135 W/mK, a CTE of 5.1 ppm, an as-manufactured surface finish of 0.813 μm, and a camber of 0.0005 in/in.

12.9 Conductor Metallization

In MCM, fabrication materials chosen will depend on the design, electrical requirements, and process chosen to fabricate the MCM. It is important to note that the most critical requirements for conductor metallization are conductivity and reliability.

Aluminum

Aluminum is a low-cost material that has adequate conductivity and can be deposited and patterned by typical IC techniques. It is resistant to oxidation. It can be sputtered or evaporated, but cannot be electroplated.

Copper

Copper has significant conductivity over aluminum and is more electromigration-resistant. It can be deposited by sputtering, evaporation, electroplating, or electroless plating. Copper rapidly oxidizes, forming a variety of oxides that can have poor adhesion to polymer dielectrics and copper itself.

Gold

Gold is used in thin-film structures to minimize via contact resistance problems caused by oxidation of Cu and Al. Gold can be deposited by sputtering, evaporation, electroplating, or electroless plating. Cost is high with excellent resistivity characteristic. Adhesion is poor and so it requires a layer (50 to 200 nm) of Ti or Ti/W.

12.10 Choosing Substrate Technologies and Assembly Techniques

The MCM designer has the freedom of choosing/identifying substrate and assembly technologies[13–15] from many sources.[16,17] If you are about to start a design and are looking for guidelines, finding a good source of information could be Internet access. In addition to designing to meet specific performance requirements, it is also necessary to consider ease of manufacture. For example, Maxtek publishes a set of design guidelines, that inform the MCM designer of preferred assembly materials and processes, process flows, and layout guidelines.

Substrate Technologies	Assembly Techniques
Chip-on-board	Surface mount
Chip-on-flex	Chip-and-wire
Thick-film ceramic	Mixed technology
Cofired ceramic	Special module
Thin-film ceramic	

Under Substrate Technologies and Assembly techniques it will be very informative to look at some pictorial examples, a primary source of which is Maxtek. The pictorial examples, followed by a brief description of technology or assembly technique shown, will serve as a quick guideline for someone who would like to get a feel of what technologies are viable in the MCM technology domain. Here, it is important to mention that a lot of current space electronic flight projects use MCM technologies for their final deliverables. Project Cassini, for example, used MCM hybrids in telecommunication subassemblies. On this note, take a look at some of the examples of MCM technologies currently on the market.

Chip-on-Board

Chip-on-board substrate technology (Fig. 12.1) has low set-up and production costs and utilizes Rigid FR-406, GETEK, BT Epoxy, or other resin boards.[3] Assembly techniques used are direct die attach/wire bonding techniques, combined with surface-mount technologies.

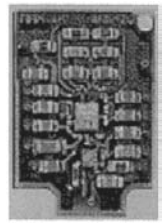


FIGURE 12.1 Chip-on-board.

Chip-on-Flex

Chip-on-Flex substrate technologies[18,19] (Fig. 12.2) are excellent for high-frequency, space-constrained circuit implementation. In creating this particular technology, the manufacturer needs Kapton or an equivalent flex-circuit base material with board stiffeners. Here, the die can be wire-bonded, with "glob-top" protection, while other discretes can be surface mounted. Integral inductors can be incorporated (e.g., for load matching).

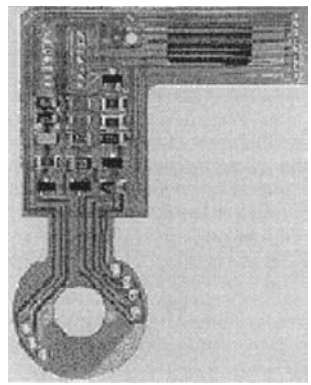


FIGURE 12.2 Chip-on-flex.

Thick-Film Ceramic

This technology is the most versatile technology, with low-to-medium production costs. The 1-GHz attenuator above demonstrates the versatility of thick film on ceramic substrate technology (Fig. 12.3),

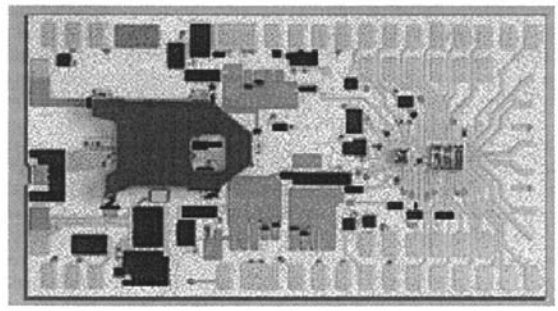


FIGURE 12.3 Thick-film ceramic.

which utilizes both standard and custom ICs, printed resistors and capacitors actively trimmed to 0.25% with extremely stable capacitors formed between top plate and ground plane on the other side of substrate. Thick-film thermistor here senses overheating resistor and protects the remainder of the circuit.

Co-fired Ceramic

Co-fired ceramic MCM substrate technologies (low- or high-temperature co-fired multilayer ceramic) (Fig. 12.4) are particularly suited for high-density digital arrays. Despite its high set-up and tooling costs, up to 14 co-fired ceramic layers are available from this particular manufacturer. In this technology, many package styles are available, including DIP, pin-grid array, and flat pack.

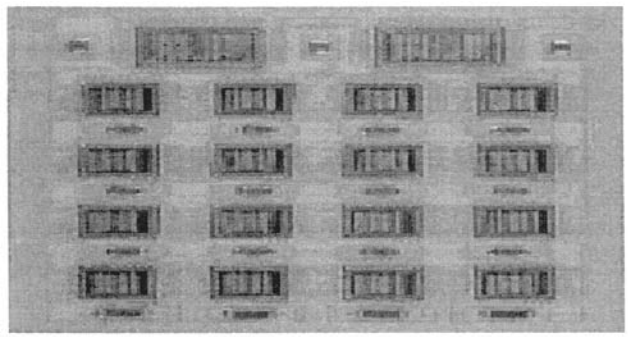


FIGURE 12.4 Co-fired ceramic.

Thin-Film Ceramic

For thin-film ceramic technologies (see Fig. 12.5), here is the outlined technology features include:

- High-performance MCMs, offset by high set-up and tooling costs
- Alumina or BeO (as shown here) substrate
- Both sides of substrate can be used, the back side typically being a ground plane, with access through plated-through holes
- Chip-and-wire assembly with epoxied capacitors
- Many packaging styles available, including Kovar or ceramic package and lid
- Primarily used for high-frequency circuits requiring thin-film inductors or controlled impedance lines

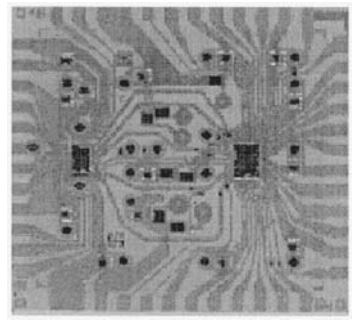


FIGURE 12.5 Thin-film ceramic.

For quick reference, some specifications and MCM technology information are summarized in Tables 12.3 and 12.4.

TABLE 12.3 Thick-Film Specifications

Printed Component	Sheet Resistivity per square	Typical Values	Comments
Conductor, Gold	3–5 mohm		Lowest resistivity, 5-mil min. line/space
Etched thick-film	3–5 mohm		2 mil min. line/space
Conductor, Pd-Ag	<50 mohm		Lowest cost, solderable, 10-mil min. line/space
Resistor	3 ohm–1 Mohm		<±100 ppm/°C, laser trimmable to ±0.25%
Capacitor		10, 50, 100, 1500	Untrimmed ±30%, may be actively trimmed dielectric constant
Inductor		4–100 nH	Untrimmed ±10%

Source: Ref. 19.

TABLE 12.4 MCM Substrate Technologies

Technology	Material	Line/Space (mils min.)	Dielectric Constant	Integral R	Integral C	Integral L
Chip-on-board	Glass-epoxy (FR-4), polyimides	4	4.3-5.0 4.0-4.6	Y	Y	
Chip-on-flex	Kapton or equiv. with stiffeners	3 inner, 5 outer	3.2-3.9	Y	Y	
Thick-film ceramic	Alumina	5	9.26.3	Y	Y	Y
	BeO	5		Y	Y	
Multilayer ceramic	Lo-fire alumina	5	5.8-9.0	Y	Y	
	Hi-fire alumina	5	9.0-9.6	Y	Y	
Thin-film ceramic	Alumina	1	9.9	Y	Y	Y
	BeO	1	6.3			
Etched thick-film	Alumina	2	9.2	Y	Y	Y

Other thick-film components are available, such as thermistors and spark gaps. In many applications, both sides of a substrate can be used for printed components.

12.11 Assembly Techniques

Surface-Mount Assembly

The surface-mount assembly technique can be categorized under lowest-cost, fastest turnaround assembly method, using pre-packaged components soldered to glass-epoxy board, flex circuit, or thick-film ceramic substrate. Many package styles are available, such as SIP, DIP. Pins may be attached as in-line leads, 90° leads, etc.[19]

Chip-and-Wire Assembly

In order to minimize interconnect electronic circuit parasitics, a high-density layout is one of the assembly techniques recommended as a solution. The highlight of this technique is epoxy attachment/wire bonding of integrated circuits and components (e.g., capacitors) to a glass-epoxy board (chip-on-board). Of course, another way is attachment to a thick- or thin-film ceramic substrate. Currently, many package styles are available and can be listed as follows[19,20]:

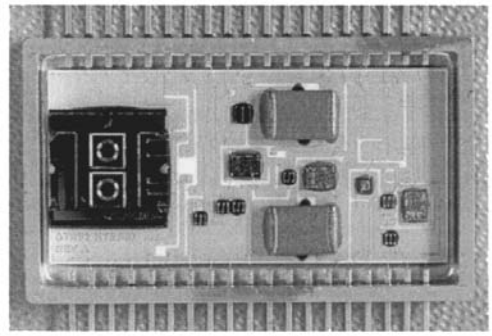


FIGURE 12.6 JPL STRV-2 project: tunneling accelerometer.

- Epoxy seal, using a B-stage ceramic lid epoxies to the substrate (quasi-hermetic seal)
- Encapsulation, in which bare semiconductor die are covered with a "glob top" (low-cost seal)
- Metal (typically Kovar) package with Kovar lid either welded or soldered to the package (hermetic seal)
- Leads are typically plug-in type, SIP, DIP, PGA, etc.

Mixed Technologies

Another category of assembly technique recognized as "mixed technologies" combines chip-and-wire with surface-mount assembly techniques on a single substrate, which may be a glass-epoxy board or ceramic substrate. Heat sink/heat spreaders are available in a variety of materials.

The Maxtek module shown in Fig. 12.7 includes a 4-layer, 20-mil-thick glass-epoxy board mounted to a beryllium copper heat spreader.

Selectively gold plated for wire bonding pads and pads at each end for use with elastomeric connectors,

- Three methods of IC die attach
- Epoxied directly to glass-epoxy board
- Epoxied directly to the BeCu heat spreader through a cutout in the board
- Epoxied to the head spreader, through a cutout, via a thermally conductive submount, to electrically isolate the die from the heat spreader

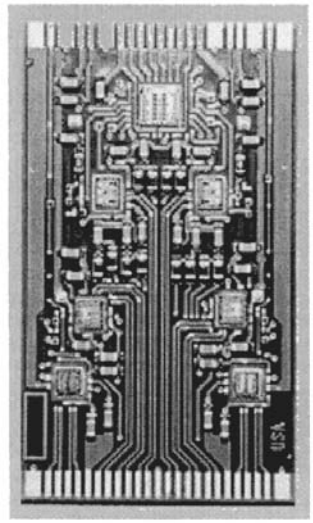


FIGURE 12.7 Example of a mixed-technology assembly technique.

- Solder-mounted resistors and capacitors
- 50-ohm differential signal lines
- IC die may be "glob topped" or covered in either a ceramic or plastic lid for protection

Special Modules

Under special modules we can emphasize and highlight technologies of complex assemblies of mixed-technology substrates, flexible circuits, and/or electromechanical components. (See Fig. 12.8.) Complex assemblies of mixed-technology substrates often utilize a B-stage epoxy lid or glob top over chip-and-wire circuitry. These technologies enable designers to provide an integrated solution complex system problems like in a module shown on page 22 which is CRT Driver System capable of displaying 4 million pixels with 1.5-ns rise and fall times. The circuit incorporates a thin-film MCM connected by a special high-frequency connector to an FR-4 board with thick-film ceramic low-inductance load resistor and flex circuit with an integral impedance-matching inductor, connecting directly to the CRT.[19,20]

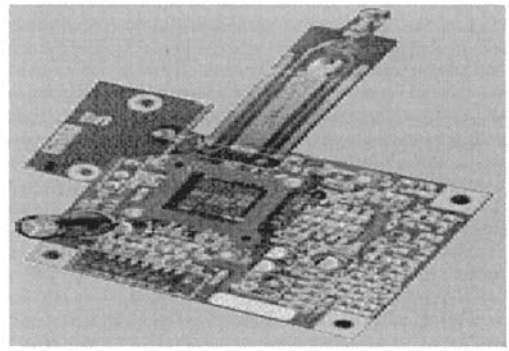


FIGURE 12.8 Example of a special module.

The design engineer of an MCM chip should work with customer to partition the circuit and optimize the design to be implemented in MCM technology. Application of technologies for placement, routing, via minimization, tree searching, and layer estimation will be important to assess at this point. The general function, purpose, and configuration of the active elements, interconnects, and assembly technology should also be assessed, along with key materials and critical properties, representative manufacturing-process flows, potential failure mechanisms, qualification procedures, and design for testability.

Note that two concepts must be carefully examined for successful MCM production. An MCM design is initiated by selecting appropriate technologies from the many options available. The basic choices are for substrate technology and assembly techniques. Design trade-offs are analyzed, and a preliminary specification is completed. Following circuit simulation, prototypes are produced and tested. When the application requires it, an ASIC can be designed to be included in the MCM.

12.12 Summary

In summary, it is customary to give an answer to the fundamental question: What multi-chip modules do for you?… and here is the answer …

MCMs optimize critical design parameters such as speed, density, and temperature, resulting in performance well beyond PC board design capabilities. By removing discrete component packages and using more densely packed interconnects, circuit speeds increase. The design challenge is to select the appropriate packaging technology, and to manage any resulting thermal problems.[20]

MCM technologies found their way and are utilized in the wireless, fiber, and instrumentation markets; space and military programs; and in the real world, they stand in the forefront of best merchant-market technology .

References

1. W. D. Brown, *ELEG 5273 Electronic Packaging*, University of Arkansas.
2. P. E. Garrou and I. Turlik, *Multichip Module Technology Handbook*, McGraw-Hill, 1998.
3. J. H. Reche, "High Density Multichip Interconnect for Advanced Packaging," *Proc. NEPCON West*, 1989, 1308
4. N. G. Koopman, T. C. Reiley, and P. A. Totta, "Chip and Package Interconnections," in *Microelectronics Packaging Handbook*, Van Nostrand Reinhold, New York, 1989, 361.
5. D. Suranayana et al., "Flip Chip Solder Bump Fatigue Life Enhanced by Polymer Encapsulation," *Proc. 40th ECTC*, 1990, 338
6. J. G. Aday, T. G. Tessier, H. Crews, and J. Rasul, "A Comperative Analysis of High Density PWB Technologies," *Proc. Int. MCM Conference*, Denver, 1996, 239.
7. J. G. Aday, T. G. Tessier, and H. Crews, "Selecting Flip Chip on Board Compatable High High Density PWB Technologies," *Int. J. Microcircuits and Electronic Packaging*, vol. 18, No. 4, 1995, 319.
8. Y. Tsukada, S. Tsuchida and Y. Mashimoto, "Surface Laminar Circuitry Packaging," *Proc. 42nd ECTC*, 1992, 22.
9. M. Moser and T. G. Tessier, "High Density PCBs for Enhanced SMT and Bare Chip Assembly Applications," *Proc. Int. MCM Conference*, Denver, 1995, 543.
10. E. Enomoto, M. Assai, Y. Sakaguchi, and C. Ohashi, " High Density Printed Wiring Boards Using Advanced Fully Additive Processing," *Proc. IPC*, 1989, 1.
11. C. Sullivan, R. Funer, R. Rust, and M. Witty, "Low Cost MCM-L for Vehicle Application," *Proc. Int. MCM Conf.*, Denver, 1996, 142.
12. W. Schmidt, "A Revolutionary Answer to Today's and Future Interconnect Challange," *Proc. of Printed Circui World Convention VI*, San Francisco, CA, 1994, T12-1.
13. D. P. Seraphim, D. E. Barr, W. T. Chen, G. P. Schmitt, and R. R. Tummala, "Printed Circuit Board Packaging," *in Microelectronics Packaging Handbook*, Van Nostrand Reinhold, New York, 1989, 853.

14. M. J. Begay, and R. Cantwell, "MCM-L Cost Model and Application Case Study," *Proc. Int. MCM Conference*, Denver, 1994, 332.
15. Compositech Ltd., 120 Ricefield Lane, Hauppauge, NY 11788-2071.
16. H. Holden, "Comparing Costs for Various PWB Build Up Technologies," *Proc. Int. MCM conference*, Denver, 1996, 15.
17. Diekman J. and Mirhej, M., "Nonwoven Aramid Papers: A New PWB Reinforcement Technology," *Proc.IEPS*, 1990, 123.
18. J. Fjelstad, T. DiStefano, and K. Karavakis, "Multilayer Flexiable Circuits with Area Arrray Interconnections for High Performance Electronics," *Proc. 2nd Int. FLEXCON*, 1995, 110.
19. Maxtek, Web site reference: http://www.maxtek.com.
20. James J. Licari and L. R. Enlow, *Hybrid Microcircuit Technology Handbook*, July 1988, 25.

Kizilyalli, I., et al. "Channel Hot Electron Degradation-Delay in MOS Transistors Due to Deuterium Anneal"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

13

Channel Hot Electron Degradation-Delay in MOS Transistors Due to Deuterium Anneal

Isik C. Kizilyalli
Lucent Bell Laboratories

Karl Hess
University of Illinois at Urbana-Champaign

Joseph W. Lyding
University of Illinois at Urbana-Champaign

13.1 Introduction

Hydrogen-related degradation by hot electrons in MOS transistors has been long known and is well documented.[1] It has recently been discovered that the degradation exhibits a giant isotope effect if hydrogen is substituted by deuterium.[2–4] The isotope effect can delay the channel hot-electron degradation by factors of 10 to 100 and, with the current definition of lifetime, even much beyond that. It therefore must be an effect different to the known kinetic isotope effect and the standard changes in reaction velocity of a factor of three or so when hydrogen is substituted by deuterium.

Deuterium is a stable and abundantly available isotope of hydrogen. It is contained at a level larger than 10^{-4} in all natural water sources. Its mass is roughly twice that of hydrogen, while all its electronic energy levels and the related chemistry are identical to that of hydrogen.

The difference in channel hot-electron degradation and lifetime must therefore be due to the mass difference. There are several possible explanations of the giant isotope effect in degradation.[4] The most probable explanation at low supply voltages ($V_{DD} < 3.3$ V) is the one first advanced in Ref. 5. This explanation concerns the dynamics of hydrogen (deuterium) desorption from an Si-H (Si-D) bond at the silicon–silicon dioxide interface under extreme non-equilibrium conditions — which is the central cause of the particular degradation discussed here. There are other forms of degradation known that show no, or a much lesser, isotope effect and are not discussed here. According to Ref 5, the heated electrons (with a typical energy of several electron volts) excite local vibrations of the Si–H bond. These vibrations have a very long lifetime of the order of nanoseconds or longer because the vibrational energy of the Si–H is mismatched to the bulk vibrations in both silicon and silicon dioxide. Therefore, a high probability exists that other hot electrons will collide with the Si–H and cause further vibrational excitation until, finally, desorption is accomplished. One vibrational mode

of Si–D, on the other hand, virtually matches a bulk silicon vibrational energy. Therefore, the local Si–D vibrations are short lived and it is much less likely that Si–D is excited so much that deuterium will desorb. The basic science of these processes has meanwhile been investigated (e.g., by scanning tunneling microscopy) and is consistent with the above description.[4] Similar desorption processes are known from photochemistry, where the energy is provided by photons, and the new aspect here is only the energy supply by the channel electrons and the extent of the isotope effect due to the special interface properties. While these considerations apply only under extreme nonequilibrium conditions, there are also in-equilibrium isotopic differences between the Si–H and Si–D bond, again due to their vibrational differences.

The question that appears under the initial equilibrium conditions, before the degradation occurs, is the following. Since many processing steps in MOS technology introduce hydrogen in one form or another, and since hydrogen densities in the silicon dioxide will be of the order of 10^{18} cm^{-3} whether desired or not, how can the hydrogen be effectively replaced by deuterium? An elementary proof[6] shows that the equilibrium population of the silicon bond by H or D also depends on the vibrational properties. Because of the higher vibrational energy of some of the Si-H vibrational modes (compared to deuterium), hydrogen is less likely to saturate the bond. In fact, if H and D are present at the same density at the interface, then (around the usual anneal temperature of 425°C) the deuterium is about 10 times more likely to populate the silicon bond than hydrogen.[6]

From these facts, it is evident that a relatively simple substitution of hydrogen by deuterium can lead to very beneficial delays of hot-electron degradation. In the following, we first describe the necessity to anneal the silicon–silicon dioxide interface with H or D; then we describe the advantages of D for hot electron degradation and the introduction of D by post-metal anneal procedures. Finally, we describe confirmations and extensions to more complex introductions of D.

13.2 Post-Metal Forming Gas Anneals in Integrated Circuits

Low-temperature post-metal anneals (350–450°C) in hydrogen ambients have been successfully used in MOS fabrication technologies to passivate the silicon dangling bonds and consequently to reduce the Si/SiO_2 interface trap charge density.[7–11] This treatment is imperative from a fabrication standpoint since silicon dangling bonds at the Si/SiO_2 interface are electrically active and lead to the reduction of channel conductance and also result in deviations from the ideal capacitance–voltage characteristics. Electron spin resonance (ESR) measurements performed in conjunction with deep-level transient spectroscopy (DLTS) and capacitance–voltage (C-V) measurements have elucidated the role of hydrogen in this defect annihilation process. The passivation process is described by the equation

$$P_b + H_2 \rightarrow P_bH + H \tag{13.1}$$

where P_bH is the passivated dangling bond. These measurements indicate that for the oxides grown on Si-<111>, the density of the interface trap states in the middle of the forbidden gap decreases from 10^{11}–10^{12} cm^{-2} eV^{-1} to about 10^{10} cm^{-2} eV^{-1} after the post-metal anneal process step. The Si-<100>/SiO_2 material system, which is technologically more significant, exhibits the same qualitative behavior. The necessity of post-metallization anneal processing for CMOS technologies is demonstrated in Figs. 13.1 and 13.2 where the measured NMOS threshold voltage (V_{th}) and transconductance (g sub m) distributions of a wafer annealed in forming gas (10% H_2) is compared with an untreated wafer. The high mean value and variation of the threshold voltage and reduced channel mobility across the untreated wafer is a clear indication of the unacceptable levels of interface trap density for CMOS circuit operation and stability. As described above, MOS transistors under bias can degrade as a result of channel hot (large kinetic energy) carriers (electrons and holes) stimulating the desorption of the hydrogen that is passivating the dangling bonds at the Si/SiO_2 interface. These concerns are exacerbated with the ever-ongoing scaling efforts for high-performance transistors and added dielectric/metallization (and hence plasma process damage) layers in integrated circuits.

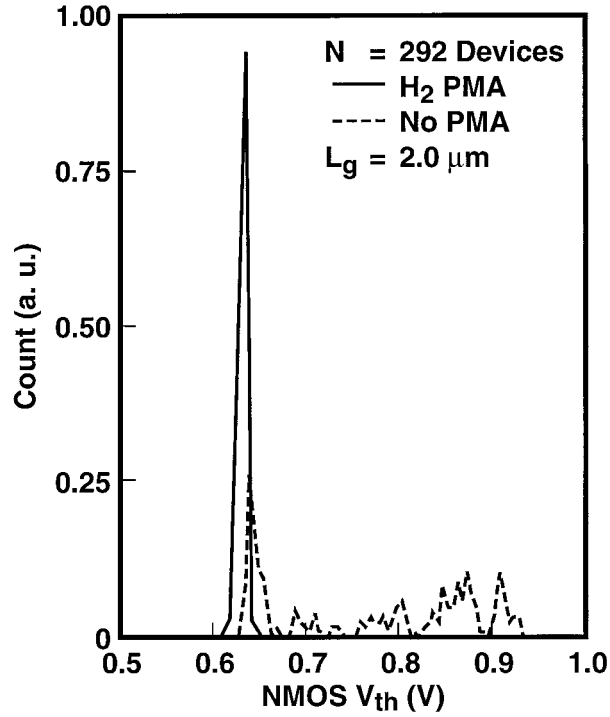


FIGURE 13.1 Histogram demonstrating the effects of hydrogen anneals on the threshold voltage V_{th} of NMOS transistors.

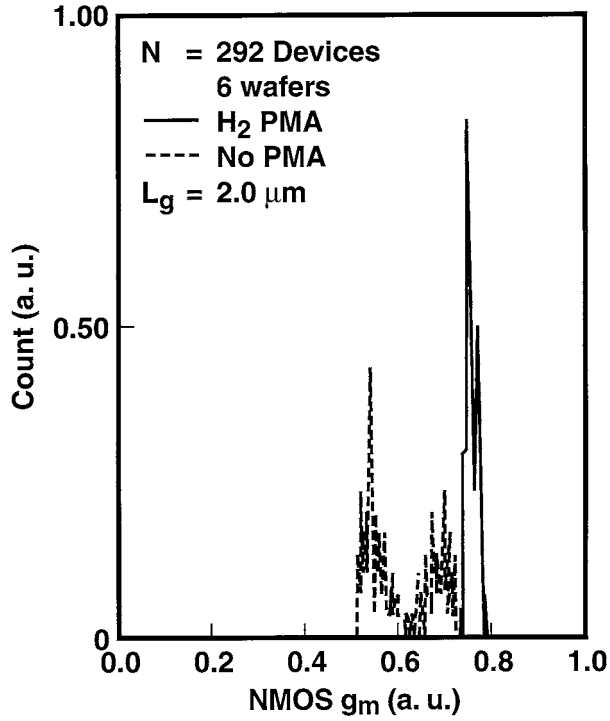


FIGURE 13.2 Same as Fig. 13.1 but for the transconductance g_m.

13.3 Impact of Hot Electron Effects on CMOS Development

The current industry practice for evaluating the intrinsic hot carrier related reliability of a MOS transistor involves two distinct accelerated electrical stress test methodologies (including certain intermediate methodologies). Although it is unlikely that the physical mechanisms responsible for gate oxide wear-out are identical in each of these cases, hydrogen appears to play some role in both modes of degradation. In the first stress configuration, the Si/SiO_2 system is degraded via large electrical fields across the gate oxide (e.g., $|V_G| >> V_D$).[12] Threshold voltage shifts in MOS capacitor structures are observed. The damage induced in this degradation mode is due to both charge trapping within the oxide and the creation of Si/SiO_2 interface trap states. Our initial studies have not identified a clear isotope effect for this mode of stress test. In this article, only the second stress configuration, namely the *channel hot carrier aging of NMOS transistors,* where we have discovered the large isotope effect, is discussed. In this mode of accelerated stress, threshold voltage instability and channel transconductance degradation in MOS transistors[13] is induced with the aid of hot carriers (electrons and holes with large kinetic energy) using the source-drain electric field. That is, the Si/SiO_2 interface is degraded by hot carriers that are traversing the device while they gain kinetic energy from the source-drain electric field. The device is biased to maximize the substrate current (e.g., $V_G \approx 1/2V_D$). The transistor aging is accelerated by stressing the device using a drain voltage (V_D) which is much larger than the intended operating voltage. The hot carrier lifetime of the transistor is estimated by extrapolating the accelerated stress test results to operating voltage (peak substrate current specification) conditions. DC, AC, or pulsed DC waveforms are most commonly used. This mode of accelerated stress tests performed on NMOS transistors typically results in localized oxide damage, which has been identified as Si/SiO_2 interface trap states.[14,15] The asymmetry of the current–voltage characteristics under source-drain reversal indicates that the damaged region is located near the drain end of the transistor where the electric fields are the largest due to the the pinch-off effect. Moreover, it has been suggested that the generation of the interface trap states is due to hot carrier-stimulated hydrogen desorption and depassivation of the silicon dangling bonds. Channel hot-carrier degradation in MOS transistors manifests itself in the form of threshold voltage (V_{th}) instability, transconductance ($g_m = dI_{DS}/dV_{GS}$) degradation, and a change in the subthreshold slope ($S_I = d \ln I_{DS}/dV_{GS}$ at $V_{GS} < V_{th}$) over time.

Various technological advances have been made to address the problem of MOS transistor degradation due to channel hot carriers. Most significant and lasting progress in fabrication technology to alleviate the channel hot carrier problem has been the development of lightly doped source-drain (LDD) and gate sidewall spacer processes.[15–17] The lightly doped drain region is used to reduce the strength of the electric field at the gate-end of the drain. Such advances have been integrated in all submicron CMOS technologies at the cost of added process complexity and intricate device design.[18] Unfortunately, processing requirements for good short channel behavior and high performance, namely, shallow source-drain junctions, reduced overlap capacitance, and low source-access resistance, are all at odds with hot carrier immune device design.

A reasonable argument (now it appears that this was merely wishful thinking) had been that the hot carrier degradation effect can be scaled away by reducing the supply voltage (constant field scaling). However, this does not appear to be the case since device feature size scaling accompanies supply voltage scaling in VLSI CMOS technologies to achieve improved performance and increased packing density. Gate oxide thickness is also reduced to maintain the device current density at low supply voltage operation. Takeda et al. have observed device degradation in a no-LDD NMOS transistor structure with gate lengths of 0.3 μm at 2.5 V.[19] Chung et al. observed hot carrier degradation for a 0.15-μm gate length transistor (with no LDD regions) at 1.8 V.[20] Current state-of-the-art high-performance 1.5- and 1.8-V sub-0.18-μm CMOS technology with good quality gate oxide exhibits hot carrier degradation effects and requires precise drain engineering.[21,22] Circuit solutions to alleviate hot carrier degradation effects in transistors, although proposed, involve further circuit design and layout complications.[23] Hot carrier-induced transistor degradation will continue to be a major roadblock for satisfying market demand for high-performance CMOS circuits such as the Lucent-DSP shown in Fig. 13.3 with transistor gate lengths phase-shifted down to 0.1 μm and operated at a large range of supply voltages (1.0 to 1.8 V).

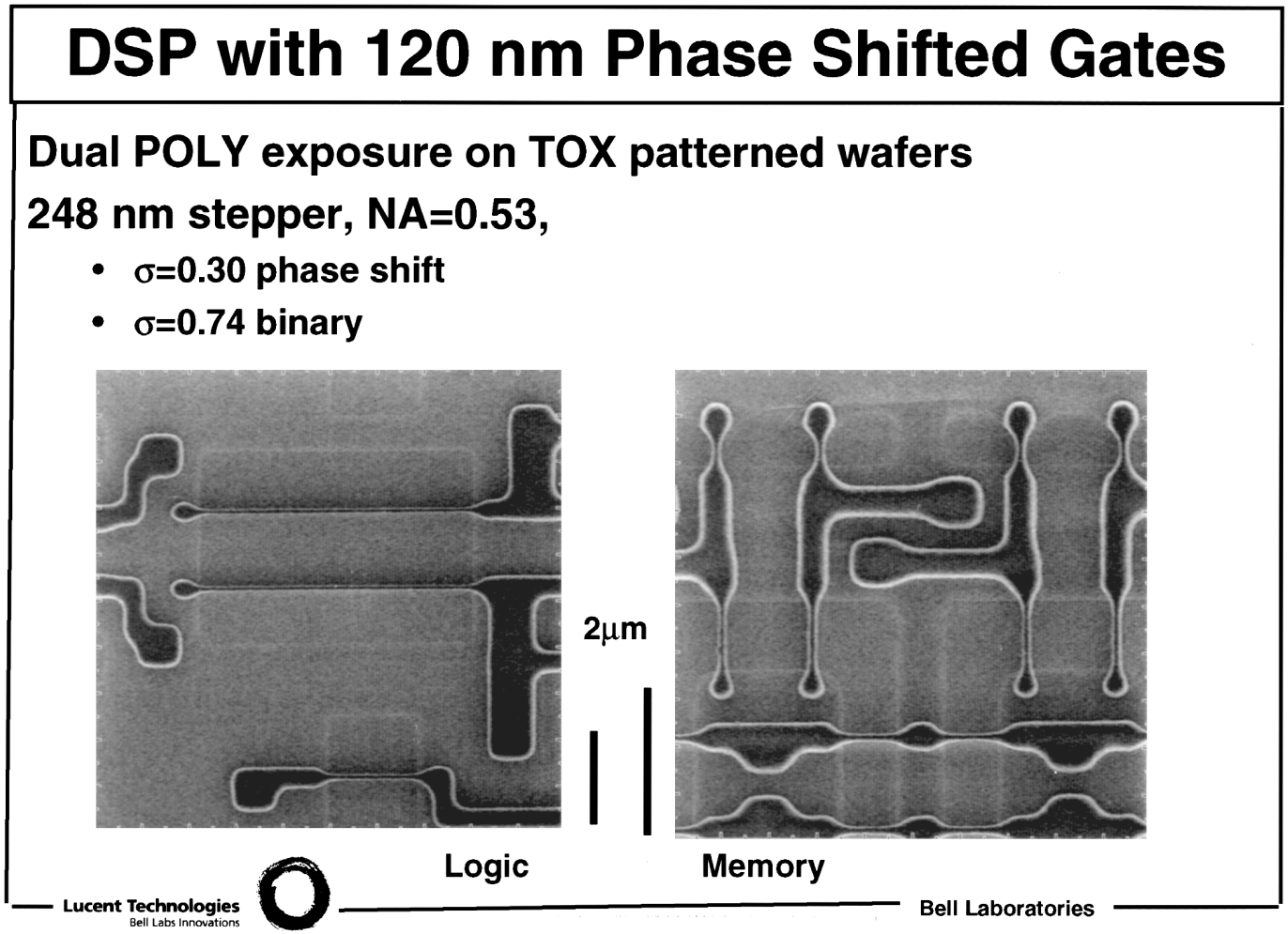


FIGURE 13.3 High-performance CMOS circuit with 0.1-μm gate length operated in a range of 1.0 V to 1.8 V of supply voltages.

13.4 The Hydrogen/Deuterium Isotope Effect and CMOS Manufacturing

As noted above, the major reliability limitation for the miniaturization CMOS technologies is the NMOS transistor hot carrier lifetime. A breakthrough in integrated circuit manufacturing is needed where reliable CMOS scaling is enabled by eliminating the undesirable effects of channel hot carriers. The giant hydrogen/deuterium isotope effect that has been discovered and reported by Lyding, Hess, and Kizilyalli[2,3] in NMOS transistors is the extremely significant increase in time-dependent channel hot carrier transistor (reliability) lifetime in devices that have been annealed with deuterium instead of hydrogen, as shown in Figs. 13.4 and 13.5. In the fabrication sequence, deuterium is introduced, instead of hydrogen, to the Si/SiO_2 interface via a low-temperature (400–450°C) post-metallization anneal process. Figure 13.6 shows the transfer characteristics of uncapped NMOS and PMOS transistors annealed in deuterium and hydrogen ambients at 400°C and 1 hour. Prior to hot electron stress, transistors annealed in either ambient are electrically identical. This results in indistinguishable device function prior to hot carrier stress. Indistinguishable device function prior to hot carrier stress is also demonstrated in two experiments shown in Table 13.1. This is an expected result since the chemistry of deuterium and hydrogen is virtually identical. These results prove that deuterium and hydrogen are equally effective in reducing the interface trap charge density which results in equivalent device function. Deuterium can be substituted for hydrogen (at least in post-metal anneal processes) in semiconductor manufacturing.

Although devices annealed either in deuterium or hydrogen appear to be identical in pre-stress electrical tests, they exhibit markedly different degradation dynamics. The observed improvement in the degradation rates (lifetimes) in the transistors is a result of the large difference in the desorption rates of the two isotopes, as described in the introduction and in detail in Refs. 4 and 5.

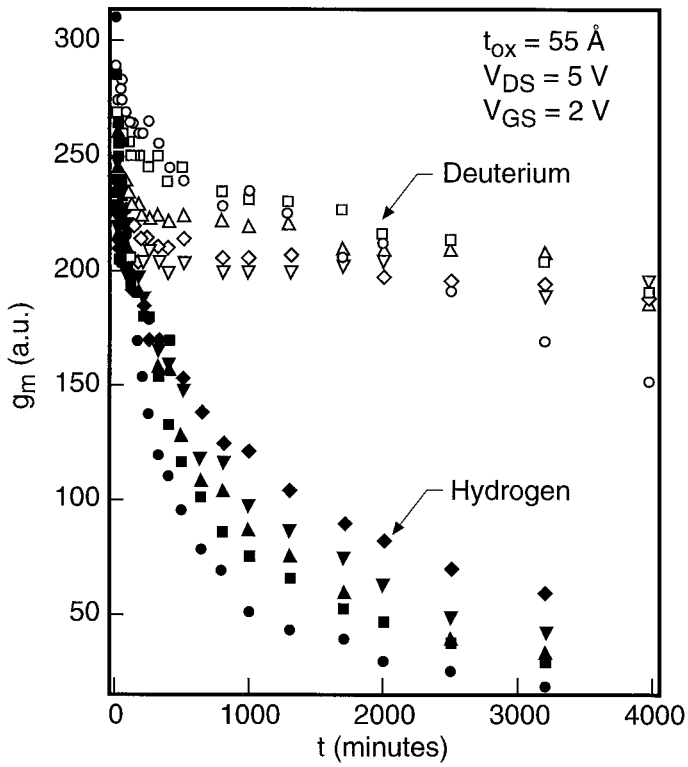


FIGURE 13.4 Transconductance channel hot-electron accelerated stress degradation with hydrogen and deuterium anneal and a significant degradation delay due to the presence of deuterium.

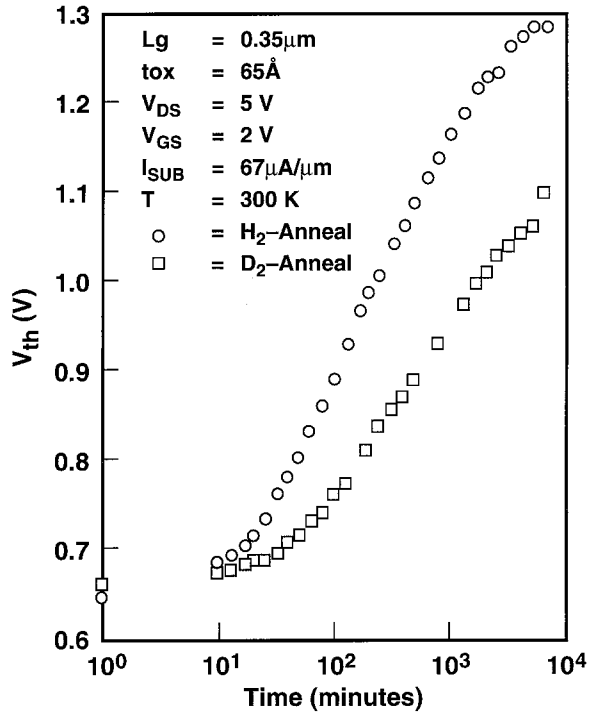


FIGURE 13.5 Same as Fig. 13.4 but for the threshold voltage.

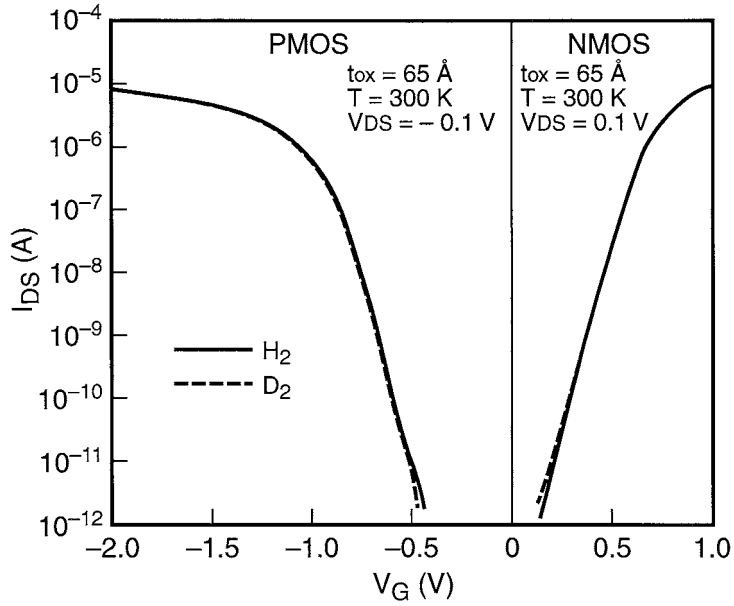


FIGURE 13.6 Subthreshold current I_{DS} as a function of gate voltage V_G for both hydrogen and deuterium anneal before degradation. No difference is shown within the experimental accuracy.

TABLE 13.1 Anneal Process and Threshold Voltage

Process	$V_{th,N}$	σ-$V_{th,N}$	$V_{th,P}$	σ-$V_{th,P}$
10%-H_2	0.51V	2.7 mV	0.983 V	2.8 mV
50%-D_2	0.51V	2.4 mV	0.986 V	4.5 mV
100%-D_2	0.51V	1.7 mV	0.987 V	1.1 mV

The large hydrogen/deuterium isotope effect in NMOS transistors has been subsequently observed and verified by other laboratories.[24–27] Studies also indicate that transistors annealed in deuterium are much more resilient against plasma process-induced damage (as quantified by Si/SiO_2 interface trap generation and gate oxide leakage).[28] Furthermore, stability of PMOS transistors;[29] hydrogenated (deuterated) amorphous silicon-based solar cells and TFTs;[30–32] hydrogen (deuterium) terminated, porous-silicon light-emitting devices;[33] and ultra-thin oxides for non-volatile memory devices[34] have been found to improve with the isotopic substitution against degradation due to light and field exposure.

For reasons outlined above, there is a strong motivation to introduce the deuterium anneal process to CMOS manufacturing. However, two further obstacles need to be removed for transfer of process to the factory floor. First, all modern CMOS technologies require a minimum of three levels of dielectric/metal interconnect process. Anneal processes that are found to be effective for improving channel hot carrier reliability in one-level of metal/dielectric structures (e.g., 400°C for 0.5 hr and 10% D_2/90% N_2) may be ineffective for multi-level metal/dielectric structures. The deuterium anneal process needs to be (and in some cases has been) optimized for multi-level interconnect. Second, the benefits of the deuterium anneal should be still evident subsequent to the final SiN cap wafer passivation process.

The test vehicle used for the experiments to surmount these challenges is a development version of Lucent Technologies 0.35 μm 3.3 V transistors with a 65 Å gate oxide, very shallow arsenic implanted MDD regions, TEOS spacers, and three dielectric (doped and undoped plasma enhanced TEOS) and metal levels (Ti/TiN/AlCuSi/TiN).[35] The deuterium (5–100% D_2) anneal process was performed after the third layer of metal had been patterned. The deuterium anneal temperatures vary between 400 to 450°C and anneal times of 1/2 to 5 hours are considered. Accelerated hot carrier DC stress experiments are performed on NMOS transistors at peak substrate current conditions. In Fig. 13.7, the time-dependent deviation of V_{th} is shown with the deuterium anneal conditions as a parameter. Table 13.2 summarizes the results of all stress experiments (V_{DS} = 5 V and V_{GS} = 2 V), assuming a degradation criteria of δV_{th} = 200 mV. The degradation dynamics of S_i and $I_{D,SAT}$ are plotted in Figs. 13.8 and 13.9. Degradation in the transistor I–V characteristics is accompanied by an increase in the interface trap density (D_{it}) as extracted (Fig. 13.10) from charge-pump current measurements. Figure 13.11 shows hot electron degradation lifetime versus substrate current with 20% g_m (transconductance) degradation as the lifetime criteria. For a peak substrate current specification of I_{SUB} = 2000 nA/μm, the extrapolated lifetime for the hydrogen and deuterium annealed device is 0.06 years and 4 years, respectively. The hot carrier lifetime (reliability) of transistors increases continuously and dramatically with increases in deuterium anneal times and temperatures. Negligible improvement is observed for short (1/2 hour) and low concentration (<10% D_2) anneal conditions. The 400°C, 2-hour process results in a four-fold improvement in lifetime over the standard hydrogen process, while the 450°C, 3-hour deuterium anneal process yields nearly a factor of 50 improvement. Hence, whenever the deuterium content in the wafer is increased, large corresponding improvements in transistor lifetimes are measured. For longer (450°C, and 5-hour) anneals, the lifetime improvement asymptotically reaches a factor of about 80 to 100 over the standard process, corresponding to similar findings in basic experiments using scanning tunneling microscopy.[4] In Fig. 13.12, it is demonstrated that the benefits of the deuterium anneal are still observed even if the post-metal anneal is followed by an SiN caps process. Similar findings for higher levels of metallization were made by other groups (see Ref. 27). For ultimate stability against further processing, and to avoid the complicated diffusion through many layers of metallization, it may be necessary to introduce the deuterium in a layer close to the device that can act as a reservoir and is activated in any temperature increase (anneal). A convincing proof of this possibility has been given.[36]

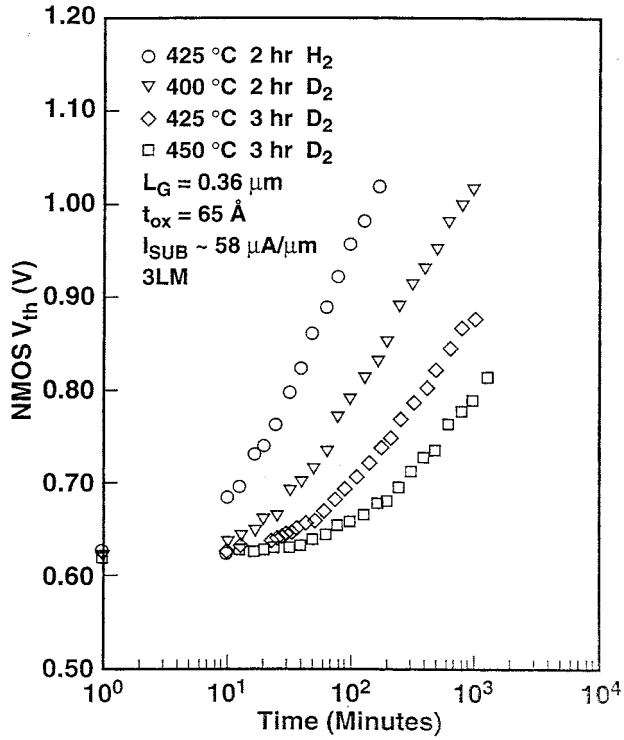


FIGURE 13.7 Degradation improvements by deuterium anneal after three levels of metallization for various temperatures and times. Only one hydrogen curve is shown; annealing with hydrogen gives identical results over wide ranges of temperature and time.

TABLE 13.2 Relative Hot Carrier Reliability (Lifetime) Improvement and D_2 Anneals

Temperature	Time	Ambient	N_2 Pre-anneal	Lifetime
425°C	2 hr	10%H_2/90%N_2	No	1.0
400°C	1/2 hr	5%D_2/95%N_2	No	1.0
400°C	1 hr	5%D_2/95%N_2	No	1.0
400°C	1 hr	10%D_2/90%N_2	No	1.0
400°C	2 hr	100%D_2	Yes	3.8
400°C	3 hr	100%D_2	No	5.5
425°C	3 hr	100%D_2	No	12.5
450°C	2 hr	100%D_2	Yes	36.0
450°C	3 hr	20%D_2/80%N_2	No	37.5
450°C	3 hr	100%D_2	No	62.5
450°C	5 hr	100%D_2	No	80.0

Figure 13.13 shows a hydrogen and deuterium profile as measured by surface ion mass spectroscopy for a successfully treated transistor. A deuterium peak concentration at the interface is a typical necessity for successful anneal. The absolute concentrations may vary according to experimental conditions.

These experiments prove that one can substitute deuterium for hydrogen in a CMOS manufacturing process with no penalty, yet with a 50 to 100-fold improvement in channel hot carrier lifetime. For completeness, other transistor structures with varying design considerations have been evaluated and summarized elsewhere.[37,38] Transistor parameters that were explored include: (1) gate oxide thickness t_{ox} = 50–115 Å, (2) LDD implant species arsenic and phosphorus and (3) gate stack structure of n^+-polysilicon and polycide (n^+-polysilicon/WSi), and (4) an experimental 0.25-μm 3.3-V CMOS[39] process

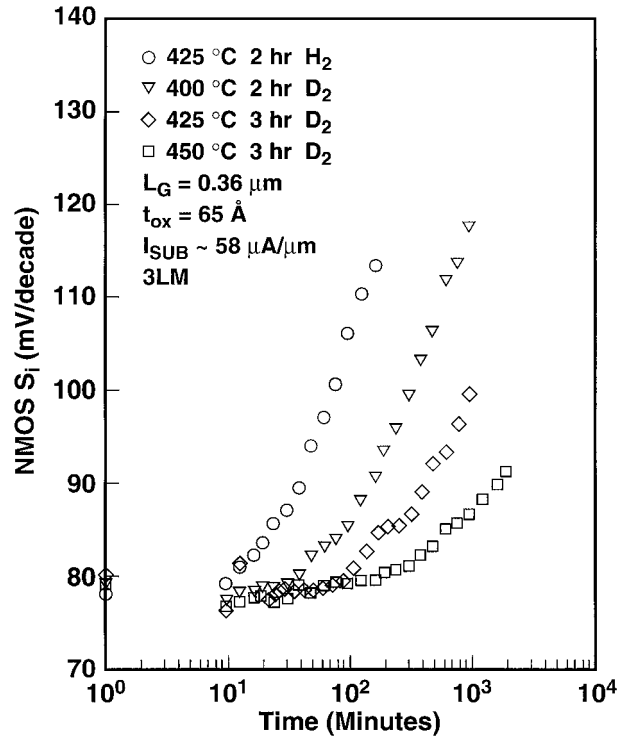


FIGURE 13.8 Degradation dynamics as in Fig. 13.7 but for S_I.

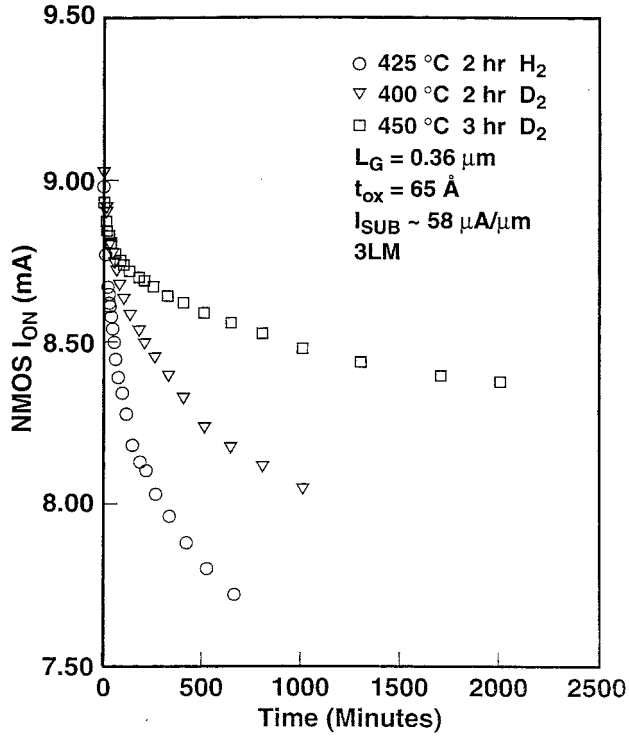


FIGURE 13.9 Degradation dynamics as in Fig. 13.7 but for I_{ON}.

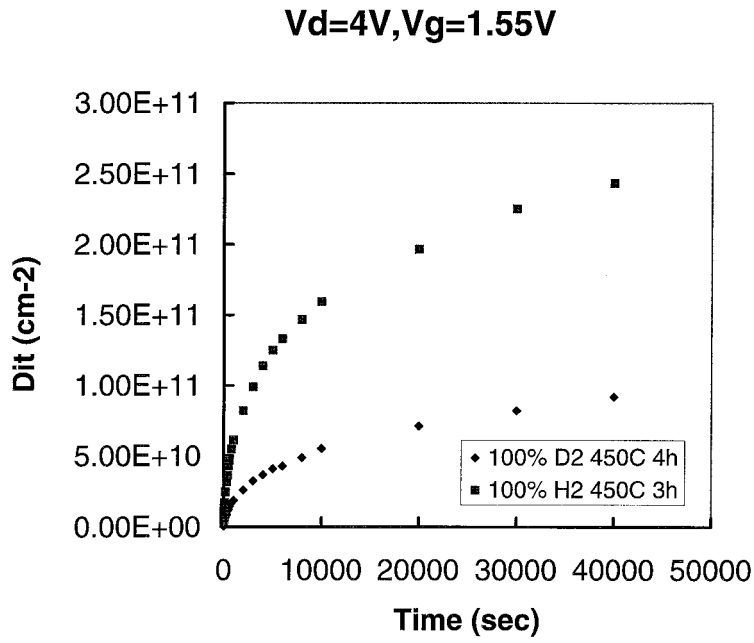


FIGURE 13.10 Interface trap density D_{it} extracted from charge-pump current measurements vs. stress time with hydrogen and deuterium anneal as indicated.

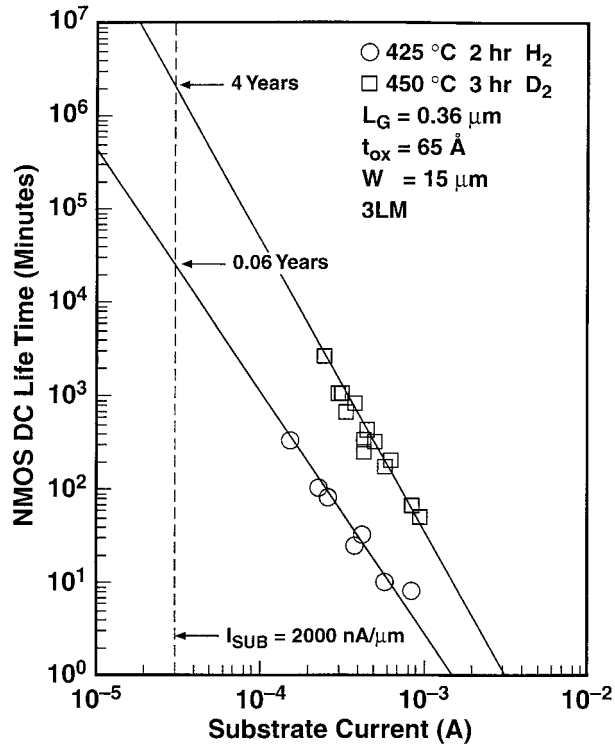


FIGURE 13.11 Channel hot electron lifetime (20% g_m degradation as lifetime limit) vs. substrate current for both hydrogen and deuterium anneal with extrapolations to actual operating conditions (dashed vertical line).

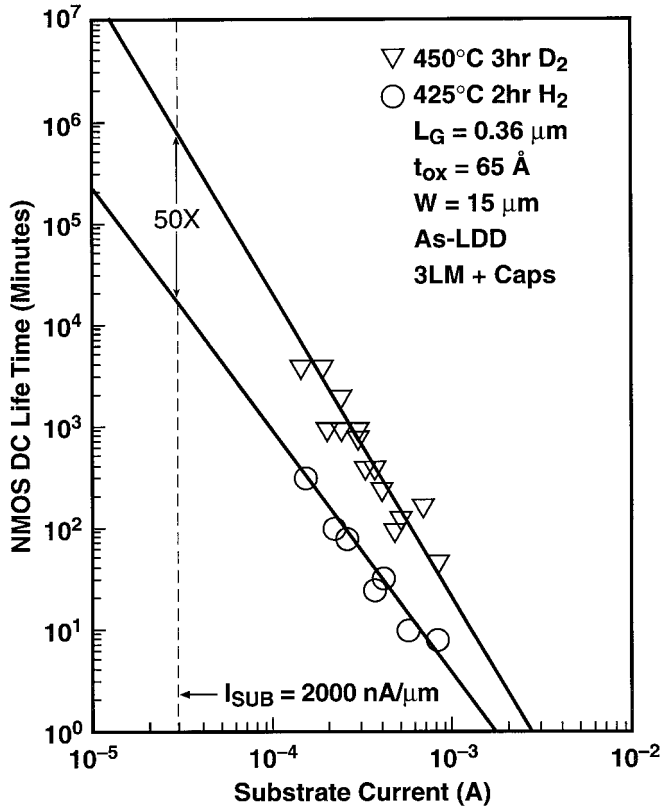


FIGURE 13.12 As in Fig. 13.11 but after final SiN cap process.

with 4 levels of metal. In all cases, the large isotope effect is observed. Clearly, the hydrogen/deuterium isotope effect is a general property of the semiconductor device wear-out.

It is important to correlate the observed improved hot carrier reliability to the location and quantity of deuterium in the wafer. Secondary ion mass spectroscopy (SIMS) analysis through the first interlevel oxide and silicon was performed on two uncapped (it is well known that $Si_x N_y$ is a barrier for deuterium) samples as shown in Fig. 13.13. The first wafer was annealed in forming gas (10% molecular hydrogen), while the other sample was annealed in forming gas comprising 10% molecular deuterium. A Cameca IMS-2f system with oxygen primary beam 60 μm^2 was used for analysis. $^{18}0^+$ was monitored to locate the SiO_2/Si interface. The $^2D^+$ concentration is inferred from the difference between the $^2H^+$ profiles for wafers annealed in deuterium and hydrogen. Deuterium was not detected under large areas of (200 times 200 μm^2) polysilicon in wafers that were annealed in deuterium and were not capped with Si_xN_y. This indicates the finite lateral diffusion length of deuterium in the transistor gate oxide and channel region. Deuterium is detected in the interlevel oxide at concentrations of 10^{19} cm^{-3} and was found to accumulate at Si/SiO_2 interfaces with a surface concentration of 10^{14} cm^{-2}. This SIMS study suggests that deuterium diffuses rapidly through the interlevel oxides and the gate sidewall spacers to passivate the interface states in the transistor channel region. However, the exact lateral spread (reach) of diffused deuterium in the transistor-channel and gate-oxide region is not certain.

The question still remains regarding the purity, specification limits, as well as cost for the implementing deuterium gas in semiconductor manufacturing. Deuterium is a stable isotope of hydrogen and is present as D_2O. Deuterium gas is produced by electrolysis of pure D_2O. Tritium is a radioactive isotope of hydrogen and has a lifetime of 12.3 years. Because of the nature of the electrolysis, the molar concentration of tritium in the gas will be essentially the same as the feed heavy water with the tritium

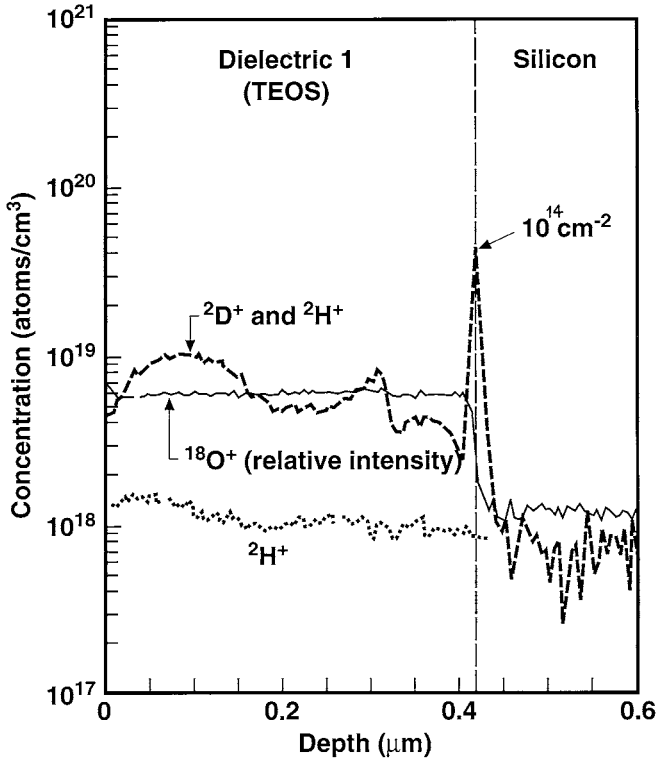


FIGURE 13.13 Typical results for hydrogen and deuterium concentrations measured after anneals by secondary ion mass spectroscopy (SIMS).

gas being in the form of DT rather than pure T_2. If the tritium content in the feed water is 50 nCi/kg, the expected concentration in the gas would be about 45 pCi/L (1.65 Bq/L). The tritium content in heavy water varies from 5 nCi/kg (virgin heavy water) to 50 Ci/kg (heavy water used in nuclear reactors). The limit of tritium in deuterium gas suitable for CMOS manufacturing is estimated as follows. When 10^{14} deuterium atoms are placed in a single chip, 3×10^{-9} Bq/L of tritium are also incorporated. This implies that one tritium decay event would occur approximately every 370 days, much below the rate of other radioactive events occurring in chip technology and operation. In addition, this is only a β-decay with usually negligible consequences. Since it is very difficult to measure tritium gas at these low concentrations, specifications should be placed on the tritium content for the heavy water used in the electrolytic production process (which is straightforward). A suggested upper limit could be 6000 nCi/kg that results in 1 tritium event/month per chip.[40] The substitution of deuterium for hydrogen adds 0.1% to the total wafer cost.

13.5 Summary

It has been demonstrated that the replacement of hydrogen by deuterium in CMOS technology can lead to significant delays in channel hot electron degradation. Increases in hot electron lifetime of a factor of 10 to 100 and beyond have been shown by simple post-metal anneals in deuterium atmosphere for several levels of metallization. Deuterium has also been proven beneficial for the reliability of other devices such as deuterated amorphous thin-film silicon devices of various kinds. Since deuterium and hydrogen have the same electronic energy levels, deuterium- and hydrogen-treated devices are indistinguishable in terms of their normal pre-stress electronic characteristics. Deuterium only delays degradation due to its higher mass and different vibrational properties.

Acknowledgments

K. H. and J. W. Lyding acknowledge financial support from the Office of Naval Research under the MURI program.

References

1. Takeda, E., Yang, C. Y., and Miura-Hamada, A., *Hot Carrier Effects in MOS Devices*, Academic Press, 1995, 73.
2. Lyding, J. W., Hess, K., and Kizilyalli, I. C., "Reduction of Hot Electron Degradation in MOS Transistors By Deuterium Processing," *Applied Physics Letters*, 68, 2526-2528, 1996.
3. Kizilyalli, I. C., Lyding, J. W., and Hess, K., "Deuterium Post Metal Annealing of MOSFETs for Improved Hot Carrier Reliability," *IEEE Electron Device Letters*, 18, 81-83, 1997.
4. Hess, K., Kizilyalli, I. C., and Lyding, J. W., "Giant Isotope Effect in Hot Electron Degradation of Metal Oxide Silicon Devices," *IEEE Transactions on Electron Devices,* 45, 406-416, 1998.
5. Vande Walle, C. G. and Jackson, W. B., "Comment on Reduction of Hot Electron Degradation in MOS Transistors by Deuterium Processing," *Appl. Phys. Lett.,* 69, 2441, 1996.
6. Ipatova, I. P., Chikalova-Luzia, O. P. , and Hess, K., "Effect of Localized Vibrations on the Si Surface Concentrations of H and D," *J. Appl. Phys.*, 83, 814-819, 1998.
7. Cheroff, G., Fang, F., and Hochberg, H., "Effect of Low Temperature Annealing on the Surface Conductivity of Si in the Si–SiO_2 — Al System," *IBM Journal*, 8, 416-421, 1964.
8. Balk, P., "Effects of Hydrogen Annealing on Silicon Surfaces," *Electrochemical Society Spring Meeting*, San Francisco, Abstract No. 109, pp. 237-240, 1965.
9. Johnson, N. M., Biegelsen, D. K. , and Moyer, M. D., "Low-Temperature Annealing and Hydrogenation of Defects at the Si-SiO_2 Interface," *J. Vac. Sci. Technolog.*, 19, 390-394, 1981.
10. Grove, A. S., *Physics and Technology of Semiconductor Devices*, New York, John Wiley & Sons, 1967.
11. Nicollian E. H., and Brews, J. R., *MOS (Metal Oxide Semiconductor) Physics and Technology*, New York, John Wiley & Sons, 1982.
12. Ning, T. H., Osburn, C. M., and Yu, H. N., "Emission Probability of Hot Electrons from Silicon to Silicon Dioxide," *J. Appl. Phys.*, 48, 286-293, 1977.
13. Abbas, S. A. and Dockerty, R. C., Hot Carrier Instability in IGFET's," *Appl. Phys. Lett.*, 27, 147-148, 1975.
14. Hu, C., Tam, S. C., Hsu, F., Ko, P., Chan, T., and Terrill, K. W., "Hot-Electron-Induced MOSFET Degradation : Model, Monitor, and Improvement," *IEEE Trans. Electron Devices*, ED-32, 375-385, 1985.
15. Doyle, B., Bourcerie, M., Marchetaux, J., and Boudou, A., "Interface State Creation and Charge Trapping in the Medium-to-High Gate Voltage Range During Hot-Carrier Stressing of N-MOS Transistors," *IEEE Trans. on Electron Devices*, ED-37, 744-754, 1990.
16. Tsang, P. J., Ogura, S., Walker, W. W., Shepard, J. F., and Critchlow, D. L., "Fabrication of High-Performance LDDFET's with Oxide Sidewall-Spacer Technology," *IEEE Trans. Electron Devices*, ED-29, 590-596, 1982.
17. Chatterjee, P. K., Shah, A. H., Lin, Y., Hunter, W. R., Walker, E. A., Rhodes, C. C., and Bruncke, W. C., "Enhanced Performance 4K×1 high-speed SRAM Using Optically Defined Submicrometer Devices in Selected Circuits," *IEEE Trans. Electron Devices*, ED-29, 700-706, 1982.
18. Chen, M. L., Leung, C. W., Cochran, W. T. , Juengling, W., Dzuiba, C. , and Yang, T., "Suppression of Hot Carrier Effects in Deep Submicrometer CMOS Technology," *IEEE Trans. Electron Devices*, ED-35, 2210-2220, 1988.
19. Takeda, E., Suzuki, N., and Hagiwara, T., "Device Performance Degradation due to Hot-Carrier Injection at Energies Below the Si-SiO_2 Energy Barrier," *IEEE International Electron Device Meeting Tech. Digest*, 396-399, 1983.

20. Chung, J. E., Jeng, M., Moon, J. E., Ko, P., and Hu, C., "Low-Voltage Hot-Electron Currents and Degradation in Deep-Submicrometer MOSFET's," *IEEE Trans. Electron Devices*, ED-37, 1651-1657, 1990.
21. Rodder, M., Aur, S., and Chen, I. C., "A Scaled 1.8 V, 0.18 μm Gate Length CMOS Technology: Device Design and Reliability Considerations," *IEEE IEDM Technical Digest*, 415-418, 1995.
22. Hargrove, M., Crowder, S., Nowak, E., Logan, R., Han, L., Ng, H., Ray, A., Sinitsky, D., Smeys, P., Guarin, F., Oberschmidt, J., Crabbe, E., Yee, D., and Su, L., "High-Performance sub-0.08 μm CMOS with Dual Gate Oxide and 9.7 ps Inverter Delay," *IEDM Tech. Digest*, 627-630, 1998.
23. Leblebici, Y. and Kang, S. M., in *Hot-carrier Reliability of MOS VLSI Circuits*, Boston, Kluwer, 1993.
24. Devine, R. A., Autran, J. L., Warren, W. L., Vanheusdan, K. L., and Rostaing, J. C., "Interfacial Hardness Enhancement in Deuterium Annealed 0.25 {mu m} Channel Metal Oxide Semiconductor Transistors," *Appl. Phys. Lett.*, 70, 2999-3001, 1997.
25. Aur, S., Grider, T., McNeil, V., Holloway, T., and Eklund, R., "Effects of Advanced Processes on Hot Carrier Reliability," *IEEE Proceedings of International Physics Symposium (IRPS)*, 1998.
26. Clark, W. F., Ference, T. G., Hook, T., Watson, K., Mittl, S., and Burnham, J., "Process Stability of Deuterium-Annealed MOSFET's," *IEEE Electron Device Lett.*, 20, 48-50, 1999.
27. Lee, J., Epstein, Y., Berti, A., Huber, J., Hess, K., and Lyding, J. W., "The Effect of Deuterium Passivation at Different Steps of CMOS Processing on Lifetime Improvements of CMOS Transistors," *IEEE Transactions on Electron Devices*, submitted.
28. Krishnan, S., Rangan, S., Hattangady, S., Xing, G., Brennan, K., Rodder, M., and Ashok, S., "Assessment of Charge-Induced Damage to Ultra-Thin Gate MOSFETs," *IEEE IEDM Technical Digest*, 445-448, 1997.
29. Li, E., Rosenbaum, E., Tao, J., and Fang, P., "CMOS Hot Carrier Lifetime Improvement from Deuterium Anneal," *56th Device Research Conference*, Santa Barbara, CA, 1998.
30. Sugiyama, S., Yang, J., and Guha, S. , "Improved Stability Against Light Exposure in Amorphous Deuterated Silicon Alloy Solar Cell," *Appl. Phys. Lett.*, 70, 378-380, 1997.
31. Wei, J. H. and Lee, S. C., "Improved Stability of Deuterated Amorphous Silicon Thin Film Transistors," *J. Appl. Phys.*, 85, 543-550, 1999.
32. Wei, J. H., Sun, M. S., and Lee, S. C., "A Possible Mechanism for Improved Light-Induced Degradation in Deuterated Amorphous-Silicon Alloy," *Appl. Phys. Lett.*, 71, 1498-1450, 1997.
33. Matsumoto, T., Masumoto, Y., Nakagawa, T., Hashimoto, M., Ueno, K., and Koshida, N. , "Electroluminescence from Deuterium Terminated Porous Silicon," *Jpn. J. Appl. Phys.*, 36, L1089-1091, 1997.
34. Kim, H. and Hwang, H., "High Quality Ultrathin Gate Oxide Prepared by Oxidation in D_2O," *Appl. Phys. Lett.*, 74, 709-710, 1999.
35. Kizilyalli, I. C., Lytle, S., Jones, B. R., Martin, E., Shive, S., Brooks, A., Thoma, M., Schanzer, R., Sniegowski, J., Wroge, D., Key, R., Kearney, J., and Stiles, K. "A Very High Performance and Manufacturable 3.3 V 0.35 μm CMOS Technology for ASICs," *IEEE Custom Integrated Circuits Conference (CICC) Tech. Digest*, pp. 31-34, 1996.
36. Ference, T. et al., *IEEE Transactions on ED46*, pp. 747-753, 1999.
37. Kizilyalli, I. C., Abeln, G., Chen, Z., Lee, J., Weber, G., Kotzias, B., Chetlur, S., Lyding, J., and Hess, K., "Improvement of Hot Carrier Reliability with Deuterium Anneals for Manufacturing Multi-Level Metal/Dielectric MOS Systems," *IEEE Electron Device Lett.*, 19, 444-446, 1998.
38. Kizilyalli, I. C., Weber, G., Chen, Z., Abeln, G., Schofield, M., Lyding, J., and Hess, K., "Multi-level metal CMOS manufacturing with deuterium for improved hot carrier reliability," IEEE IEDM Tech. Digest, 935-938, 1998.
39. Kizilyalli, I. C., Huang, R., Hwang, D., Vaidya, H., and Thoma, M., "A Merged 2.5 V and 3.3 V 0.25-μm CMOS Technology for ASICs," *IEEE CICC Technical Digest*, 159-162, 1998.
40. Machachek, R. F., Ontario Hydro (Toronto), private communication.

Comer, D.J., et al. "Bipolar Junction Transistor (BJT) Circuits"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

14

Bipolar Junction Transistor (BJT) Circuits

David J. Comer
Donald T. Comer
Brigham Young University

14.1 Introduction

The *bipolar junction transistor* (or BJT) was the workhorse of the electronics industry from the 1950s through the 1990s. This device was responsible for enabling the computer age as well as the modern era of communications. Although early systems that demonstrated the feasibility of electronic computers used the vacuum tube, the element was too unreliable for dependable, long-lasting computers. The invention of the BJT in 1947[1] and the rapid improvement in this device led to the development of highly reliable electronic computers and modern communication systems.

Integrated circuits, based on the BJT, became commercially available in the mid-1960s and further improved the dependability of the computer and other electronic systems while reducing the size and cost of the overall system. Ultimately, the microprocessor chip was developed in the early 1970s and the age of small, capable, personal computers was ushered in. While the metal-oxide-semiconductor (or MOS) device is now more prominent than the BJT in the personal computer arena, the BJT is still important in larger high-speed computers. This device also continues to be important in communication systems and power control systems.

Because of the continued improvement in BJT performance and the development of the heterojunction BJT, this device remains very important in the electronics field, even as the MOS device becomes more significant.

14.2 Physical Characteristics and Properties of the BJT

Although present BJT technology is used to make both discrete component devices as well as integrated circuit chips, the basic construction techniques are similar in both cases, with primary differences arising in size and packaging. The following description is provided for the BJT constructed as integrated circuit devices on a silicon substrate. These devices are referred to as "junction-isolated" devices.

The cross-sectional view of a BJT is shown in Fig. 14.1.[2]

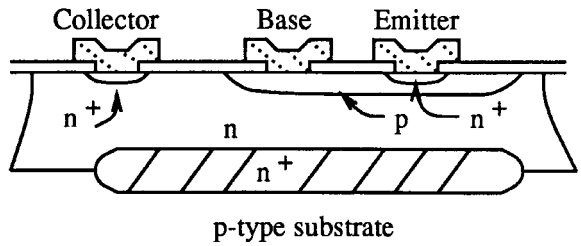


FIGURE 14.1 An integrated npn BJT.

This device can occupy a surface area of less than 1000 μm^2. There are three physical regions comprising the BJT. These are the emitter, the base, and the collector. The thickness of the base region between emitter and collector can be a small fraction of a micron, while the overall vertical dimension of a device may be a few microns.

Thousands of such devices can be fabricated within a silicon wafer. They may be interconnected on the wafer using metal deposition techniques to form a system such as a microprocessor chip or they may be separated into thousands of individual BJTs, each mounted in its own case. The photolithographic methods that make it possible to simultaneously construct thousands of BJTs have led to continually decreasing size and cost of the BJT.

Electronic devices, such as the BJT, are governed by current–voltage relationships that are typically nonlinear and rather complex. In general, it is difficult to analyze devices that obey nonlinear equations, much less develop design methods for circuits that include these devices. The basic concept of modeling an electronic device is to replace the device in the circuit with linear components that approximate the voltage–current characteristics of the device. A model can then be defined as a collection of simple components or elements used to represent a more complex electronic device. Once the device is replaced in the circuit by the model, well-known circuit analysis methods can be applied.

There are generally several different models for a given device. One may be more accurate than others, another may be simpler than others, another may model the dc voltage–current characteristics of the device, while still another may model the ac characteristics of the device.

Models are developed to be used for manual analysis or to be used by a computer. In general, the models for manual analysis are simpler and less accurate, while the computer models are more complex and more accurate. Essentially, all models for manual analysis and most models for the computer include only linear elements. Nonlinear elements are included in some computer models, but increase the computation times involved in circuit simulation over the times in simulation of linear models.

14.3 Basic Operation of the BJT

In order to understand the origin of the elements used to model the BJT, we will discuss a simplified version of the device as shown in Fig. 14.2. The device shown is an npn device that consists of a p-doped material interfacing on opposite sides to n-doped material. A pnp device can be created using an n-doped central region with p-doped interfacing regions. Since the npn type of BJT is more popular in present construction processes, the following discussion will center on this device.

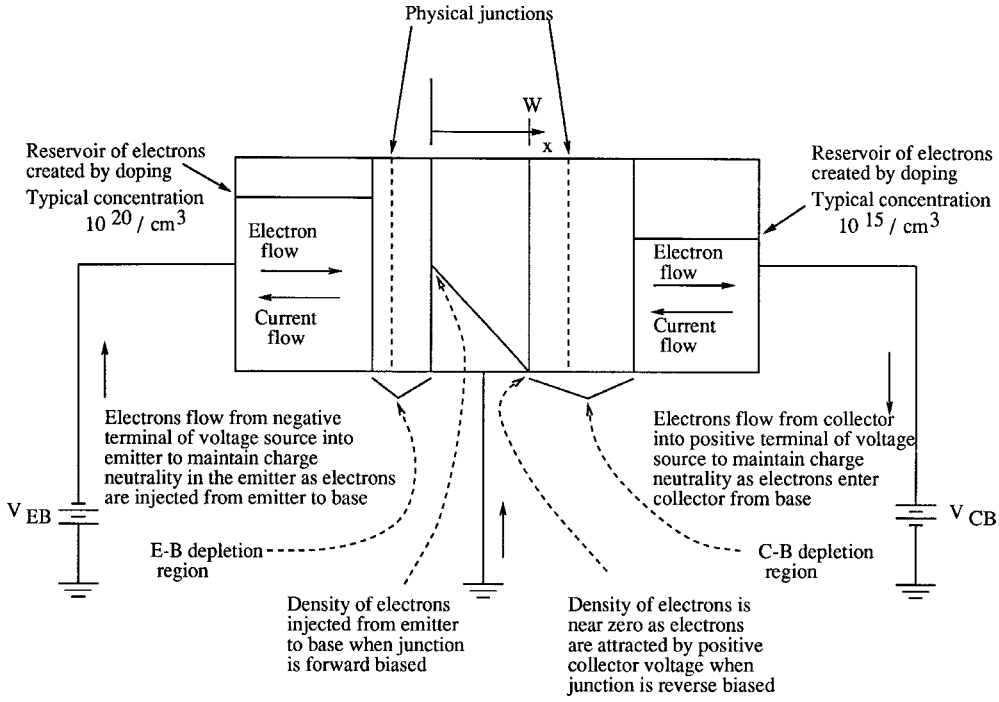


FIGURE 14.2 Distribution of electrons in the active region.

The geometry of the device implied in Fig. 14.2 is physically more like the earlier alloy transistor. This geometry is also capable of modeling the modern BJT (Fig. 14.1) as the theory applies almost equally well to both geometries. Normally, some sort of load would appear in either the collector or emitter circuit; however, this is not important to the initial discussion of BJT operation.

The circuit of Fig. 14.2 is in the active region, that is, the emitter–base junction is forward-biased, while the collector–base junction is reverse-biased. The current flow is controlled by the profile of electrons in the p-type base region. It is proportional to the slope or gradient of the free electron density in the base region. The well-known diffusion equation can be expressed as[3]:

$$I = qD_nA\frac{dn}{dx} = -\frac{qD_nAn(0)}{W} \tag{14.1}$$

where q is the electronic charge, D_n is the diffusion constant for electrons, A is the cross-sectional area of the base region, W is the width or thickness of the base region, and $n(0)$ is the density of electrons at the left edge of the base region. The negative sign reflects the fact that conventional current flow is opposite to the flow of the electrons.

The concentration of electrons at the left edge of the base region is given by:

$$n(0) = n_{bo}e^{qV_{BE}/kT} \tag{14.2}$$

where q is the charge on an electron, k is Boltzmann's constant, T is the absolute temperature, and n_{bo} is the equilibrium concentration of electrons in the base region. While n_{bo} is a small number, $n(0)$ can be large for values of applied base to emitter voltages of 0.6 to 0.7 V. At room temperature, this equation can be written as:

$$n(0) = n_{bo}e^{V_{BE}/0.026} \tag{14.3}$$

In Fig. 14.2, the voltage $V_{EB} = -V_{BE}$..

A component of hole current also flows across the base–emitter junction from base to emitter. This component is rendered negligible compared to the electron component by doping the emitter region much more heavily than the base region.

As the concentration of electrons at the left edge of the base region increases, the gradient increases and the current flow across the base region increases. The density of electrons at $x = 0$ can be controlled by the voltage applied from emitter to base. Thus, this voltage controls the current flowing through the base region. In fact, the density of electrons varies exponentially with the applied voltage from emitter to base, resulting in an exponential variation of current with voltage.

The reservoir of electrons in the emitter region is unaffected by the applied emitter-to-base voltage as this voltage drops across the emitter–base depletion region. This applied voltage lowers the junction voltage as it opposes the built-in barrier voltage of the junction. This leads to the increase in electrons flowing from emitter to base.

The electrons injected into the base region represent electrons that were originally in the emitter. As these electrons leave the emitter, they are replaced by electrons from the voltage source, V_{EB}. This current is called emitter current and its value is determined by the voltage applied to the junction. Of course, conventional current flows in the opposite direction to the electron flow.

The emitter electrons flow through the emitter, across the emitter–base depletion region, and into the base region. These electrons continue across the base region, across the collector–base depletion region, and through the collector. If no electrons were "lost" in the base region and if the hole flow from base to emitter were negligible, the current flow through the emitter would equal that through the collector. Unfortunately, there is some recombination of carriers in the base region. When electrons are injected into the base region from the emitter, space charge neutrality is upset, pulling holes into the base region from the base terminal. These holes restore space charge neutrality if they take on the same density throughout the base as the electrons. Some of these holes recombine with the free electrons in the base and the net flow of recombined holes into the base region leads to a small, but finite, value of base current. The electrons that recombine in the base region reduce the total electron flow to the collector. Because the base region is very narrow, only a small percentage of electrons traversing the base region recombine and the emitter current is reduced by a small percentage as it becomes collector current.

In a typical low-power BJT, the collector current might be $0.995I_E$. The current gain from emitter to collector, I_C/I_E, is called α and is a function of the construction process for the BJT. Using Kirchhoff's current law, the base current is found to equal the emitter current minus the collector current. This gives:

$$I_B = I_E - I_C = (1-\alpha)I_E \tag{14.4}$$

If $\alpha = 0.995$, then $I_B = 0.005I_E$. Base current is very small compared to emitter or collector current. A parameter β is defined as the ratio of collector current to base current resulting in:

$$\beta = \frac{\alpha}{1-\alpha} \tag{14.5}$$

This parameter represents the current gain from base to collector and can be quite high. For the value of α cited earlier, the value of β is 199.

14.4 Use of the BJT as an Amplifier

Figure 14.3 shows a simple configuration of a BJT amplifier. This circuit is known as the *common emitter configuration.*

A voltage source is not typically used to forward-bias the base–emitter junction in an actual circuit, but we will assume that V_{BB} is used for this purpose. A value of V_{BB} or V_{BE} near 0.6 to 0.7 V would be appropriate for this situation. The collector supply would be a large voltage, such as 12 V. We will assume that the value of V_{BB} sets the dc emitter current to a value of 1 mA for this circuit. The collector current entering the BJT will be slightly less than 1 mA, but we will ignore this difference and assume that I_C = 1 mA also. With a 4-kΩ collector resistance, a 4-V drop will appear across R_C, leading to a dc output voltage of 8 V. The distribution of electrons across the base region for the steady-sate or quiescent conditions is shown by the solid line of Fig. 14.3(a).

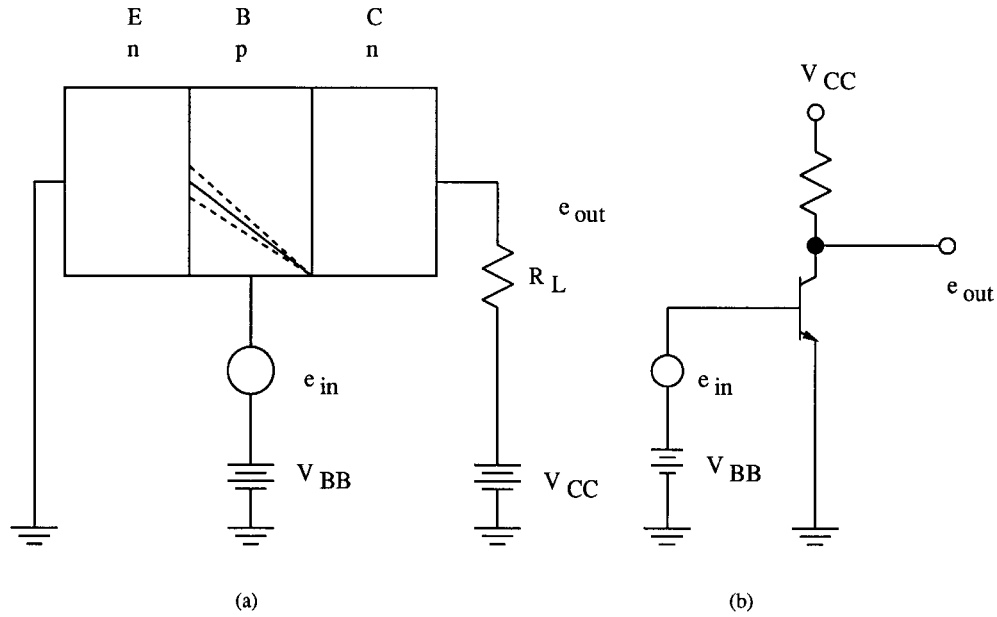


FIGURE 14.3 A BJT amplifier.

If a small ac voltage now appears in series with V_{BB}, the injected electron density at the left side of the base region will be modulated. Since this density varies exponentially with the applied voltage (see Eq. 14.2), a small ac voltage can cause considerable changes in density. The dashed lines in Fig. 14.3(a) show the distributions at the positive and negative peak voltages. The collector current may change from its quiescent level of 1 mA to a maximum of 1.1 mA as e_{in} reaches its positive peak, and to a minimum of 0.9 mA when e_{in} reaches its negative peak. The output collector voltage will drop to a minimum value of 7.6 V as the collector current peaks at 1.1 mA, and will reach a maximum voltage of 8.4 V as the collector current drops to 0.9 mA. The peak-to-peak ac output voltage is then 0.8 V. The peak-to-peak value of e_{in} to case this change might be 5 mV, giving a voltage gain of A = –0.8/0.005 = –160. The negative sign occurs because when e_{in} increases, the collector current increases, but the collector voltage decreases. This represents a phase inversion in the amplifier of Fig. 14.3.

In summary, a small change in base-to-emitter voltage causes a large change in emitter current. This current is channeled across the collector, through the load resistance, and can develop a larger incremental voltage across this resistance.

14.5 Representing the Major BJT Effects by an Electronic Model

The two major effects of the BJT in the active region are the diode characteristics of the base–emitter junction and the collector current that is proportional to the emitter current. These effects can be modeled by the circuit of Fig. 14.4.

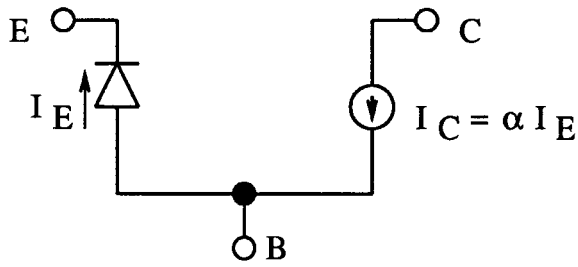


FIGURE 14.4 Large-signal model of the BJT.

The simple diode equation represents the relationship between applied emitter-to-base voltage and emitter current. This equation can be written as

$$I_E = I_1(e^{qV_{BE}/kT} - 1) \tag{14.6}$$

where q is the charge on an electron, k is Boltzmann's constant, T is the absolute temperature of the diode, and I_1 is a constant at a given temperature that depends on the doping and geometry of the emitter-base junction.

The collector current is generated by a dependent current source of value $I_C = \alpha I_E$.

A small-signal model based on the large-signal model of Fig. 14.4 is shown in Fig. 14.5. In this case, the resistance, r_d, is the dynamic resistance of the emitter-base diode and is given by:

$$r_d = \frac{kT}{qI_E} \tag{14.7}$$

where I_E is the dc emitter current.

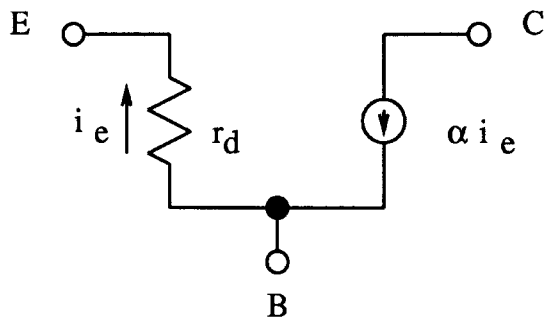


FIGURE 14.5 A small-signal model of the BJT.

14.6 Other Physical Effects in the BJT

The preceding section pertains to the basic operation of the BJT in the dc and midband frequency range. Several other effects must be included to model the BJT with more accuracy. These effects will now be described.

Ohmic Effects

The metal connections to the semiconductor regions exhibit some ohmic resistance. The emitter contact resistance and collector contact resistance is often in the ohm range and does not affect the BJT operation in most applications. The base region is very narrow and offers little area for a metal contact. Furthermore, because this region is narrow and only lightly doped compared to the emitter, the ohmic resistance of the base region itself is rather high. The total resistance between the contact and the intrinsic base region can be 100 to 200 Ω. This resistance can become significant in determining the behavior of the BJT, especially at higher frequencies.

Base-Width Modulation (Early Effect)

The widths of the depletion regions are functions of the applied voltages. The collector voltage generally exhibits the largest voltage change and, as this voltage changes, so also does the collector–base depletion region width. As the depletion layer extends further into the base region, the slope of the electron distribution in the base region becomes greater since the width of the base region is decreased. A slightly steeper slope leads to slightly more collector current. As reverse-bias decreases, the base width becomes greater and the current decreases. This effect is called *base-width modulation* and can be expressed in terms of the Early voltage,[4] V_A, by the expression:

$$I_C = \beta I_B\left(1 + \frac{V_{CE}}{V_A}\right) \tag{14.8}$$

The Early voltage will be constant for a given device and is typically in the range of 60 to 100 V.

Reactive Effects

Changing the voltages across the depletion regions results in a corresponding change in charge. This leads to an effective capacitance since

$$C = \frac{dQ}{dV} \tag{14.9}$$

This depletion region capacitance is a function of voltage applied to the junction and can be written as[4]:

$$C_{dr} = \frac{C_{Jo}}{(\phi - V_{app})^m} \tag{14.10}$$

where C_{Jo} is the junction capacitance at zero bias, ϕ is the built-in junction barrier voltage, V_{app} is the applied junction voltage, and m is a constant. For modern BJTs, m is near 0.33. The applied junction voltage has a positive sign for a forward-bias and a negative sign for a reverse-bias. The depletion region capacitance is often called the *junction capacitance*.

An increase in forward base–emitter voltage results in a higher density of electrons injected into the base region. The charge distribution in the base region changes with this voltage change, and this leads to a capacitance called the *diffusion capacitance*. This capacitance is a function of the emitter current and can be written as:

$$C_D = k_2 I_E \tag{14.11}$$

where k_2 is a constant for a given device.

14.7 More Accurate BJT Models

Figure 14.6 shows a large-signal BJT model used in some versions of the popular simulation program known as SPICE.[5] The equations for the parameters are listed in other texts[5] and will not be given here.

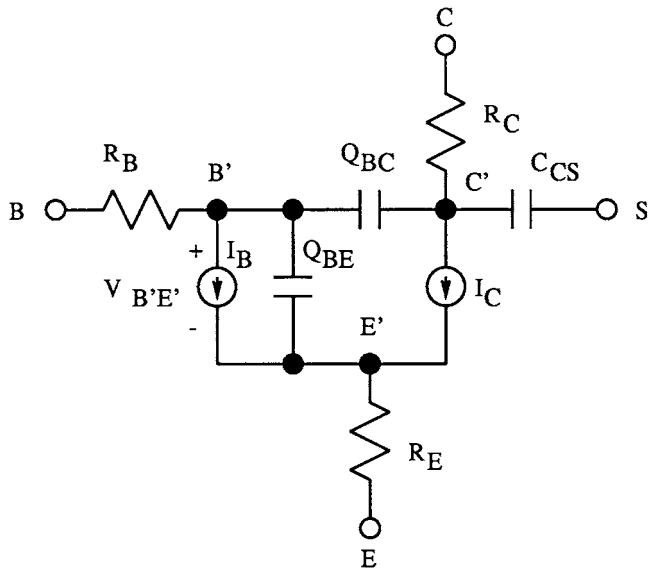


FIGURE 14.6 A more accurate large-signal model of the BJT.

Figure 14.7 shows a small-signal SPICE model[5] often called the hybrid-π equivalent circuit. The capacitance, C_π, accounts for the diffusion capacitance and the emitter–base junction capacitance. The collector–base junction capacitance is designated C_μ. The resistance, r_π, is equal to $(\beta + 1)r_d$. The transductance, g_m, is given by:

$$g_m = \frac{\alpha}{r_d} \tag{14.12}$$

The impedance, r_o, is related to the Early voltage by:

$$r_o = \frac{V_A}{I_C} \tag{14.13}$$

R_B, R_E, and R_C are the base, emitter, and collector resistances, respectively. For hand analysis, the ohmic resistances R_E and R_C are neglected along with C_{CS}, the collector-to-substrate capacitance.

14.8 Heterojunction Bipolar Junction Transistors

In an npn device, all electrons injected from emitter to base are collected by the collector, except for a small number that recombine in the base region. The holes injected from base to emitter contribute to emitter junction current, but do not contribute to collector current. This hole component of the emitter

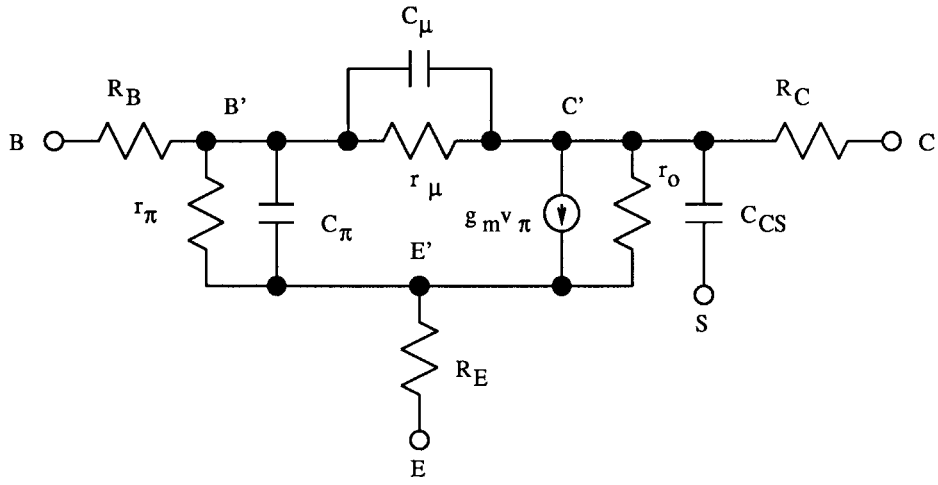


FIGURE 14.7 The hybrid-π small-signal model for the BJT.

current must be minimized to achieve a near-unity current gain from emitter to collector. As α approaches unity, the current gain from base to collector, β, becomes larger.

In order to produce high-β BJTs, the emitter region must be doped much more heavily than the base region, as explained earlier. While this approach allows the value of β to reach several hundred, it also leads to some effects that limit the frequency of operation of the BJT. The lightly doped base region causes higher values of base resistance, as well as emitter–base junction capacitance. Both of these effects are minimized in the *heterojunction BJT* (or HBJT). This device uses a different material for the base region than that used for the emitter and collector regions. One popular choice of materials is silicon for the emitter and collector regions, and a silicon/germanium material for the base region.[6] The difference in energy gap between the silicon emitter material and the silicon/germanium base material results in an asymmetric barrier to current flow across the junction. The barrier for electron injection from emitter to base is smaller than the barrier for hole injection from base to emitter. The base can then be doped more heavily than a conventional BJT to achieve lower base resistance, but the hole flow across the junction remains negligible due to the higher barrier voltage. The emitter of the HBJT can be doped more lightly to lower the junction capacitance. Large values of β are still possible in the HBJT while minimizing frequency limitations. Current gain-bandwidth figures exceeding 60 GHz have been achieved with present industrial HBJTs.

From the standpoint of analysis, the SPICE models for the HBJT are structurally identical to those of the BJT. The difference is in the parameter values.

14.9 Integrated Circuit Biasing Using Current Mirrors

Differential stages are very important in integrated circuit amplifier design. These stages require a constant dc current for proper bias. A simple bias scheme for differential BJT stages will now be discussed.

The diode-biased current sink or *current mirror* of Fig. 14.8 is a popular method of creating a constant-current bias for differential stages.

The concept of the current mirror was developed specifically for analog integrated circuit biasing and is a good example of a circuit that takes advantage of the excellent matching characteristics that are possible in integrated circuits. In the circuit of Fig. 14.8, the current I_2 is intended to be equal to or "mirror" the value of I_1. Current mirrors can be designed to serve as sinks or sources.

The general function of the current mirror is to reproduce or mirror the input or reference current to the output, while allowing the output voltage to assume any value within some specified range. The current mirror can also be designed to generate an output current that equals the input current multiplied by a scale factor *K*. The output current can be expressed as a function of input current as:

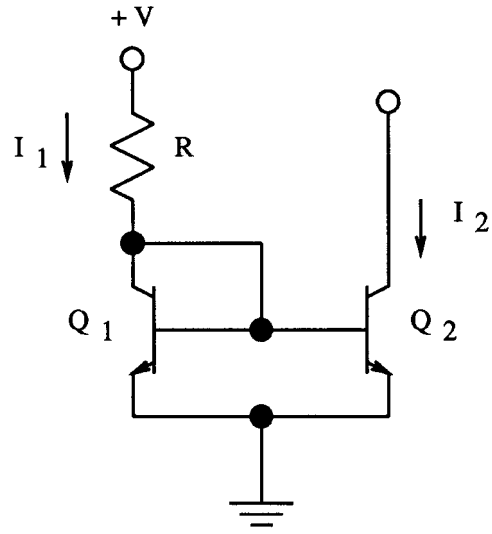


FIGURE 14.8 Current mirror bias stage.

$$I_O = KI_{IN} \tag{14.14}$$

where K can be equal to, less than, or greater than unity. This constant can be established accurately by relative device sizes and will not vary with temperature.

Figure 14.9 shows a multiple output current source where all of the output currents are referenced to the input current. Several amplifier stages can be biased with this multiple output current mirror.

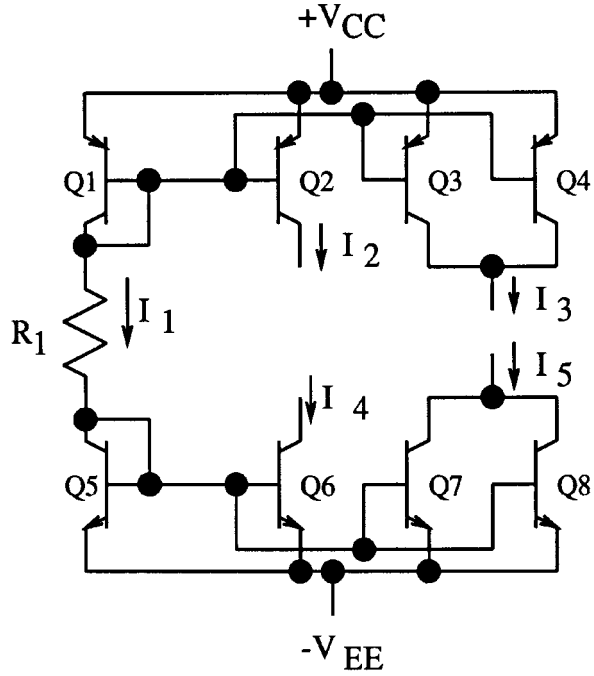


FIGURE 14.9 Multiple output current mirror.

Current Source Operating Voltage Range

Figure 14.10 shows an ideal or theoretical current sink in (a) and a practical sink in (b). The voltage at node A in the theoretical sink can be tied to any potential above or below ground without affecting the value of *I*. On the other hand, the practical circuit of Fig. 14.10(b) requires that the transistor remain in the active region to provide a current of:

$$I = \alpha \frac{V_B - V_{BE}}{R} \tag{14.15}$$

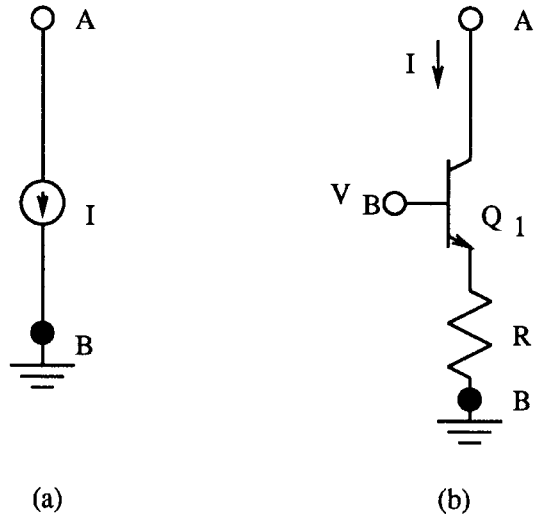


FIGURE 14.10 Current sink circuits: (a) ideal sink, (b) practical sink.

This requires that the collector voltage exceed the voltage V_B at all times. The upper limit on this voltage is determined by the breakdown voltage of the transistor. The output voltage must then satisfy:

$$V_B < V_C < (V_B + BV_{CE}) \tag{14.16}$$

where BV_{CE} is the breakdown voltage from collector to emitter of the transistor. This voltage range over which the current source operates is called the *output voltage compliance range* or the *output compliance*.

Current Mirror Analysis

The current mirror is again shown in Fig. 14.11. If devices Q_1 and Q_2 are assumed to be matched devices, we can write:

$$I_{E1} = I_{E2} = I_{EO} e^{V_{BE}/V_T} \tag{14.17}$$

where $V_T = kT/q$, $I_{EO} = AJ_{EO}$, A is the emitter area of the two devices, and J_{EO} is the current density of the emitters. The base currents for each device will also be identical and can be expressed as:

$$I_{B1} = I_{B2} = \frac{I_{EO}}{\beta + 1} e^{V_{BE}/V_T} \tag{14.18}$$

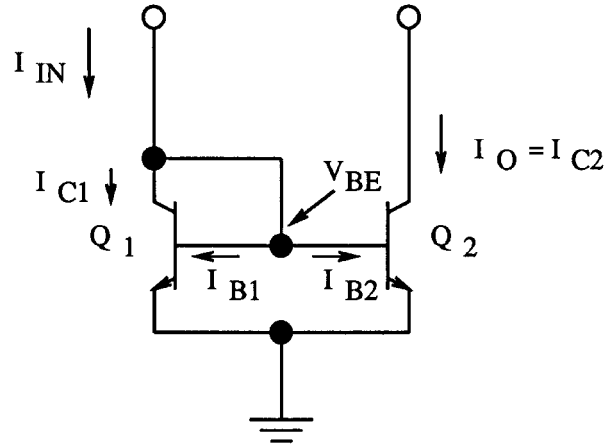


FIGURE 14.11 Circuit for current mirror analysis.

Device Q_1 operates in the active region, but near saturation by virtue of the collector–base connection. This configuration is called a *diode-connected transistor*. Since the collector-to-emitter voltage is very small, the collector current for device Q_1 is given by Eq. 14.8, assuming $V_{CE} = 0$. This gives:

$$I_{C1} = \beta I_{B1} \approx \frac{\beta}{\beta + 1} I_{EO} e^{V_{BE}/V_T} \tag{14.19}$$

The device Q_2 does not have the constraint that $V_{CE} \approx 0$ as device Q_1 has. The collector voltage for Q_2 will be determined by the external circuit that connects to this collector. Thus, the collector current for this device is:

$$I_{C2} = \beta I_{B2}\left(1 + \frac{V_{C2}}{V_A}\right) \tag{14.20}$$

where V_A is the Early voltage. In effect, the output stage has an output impedance given by Eq. 14.13. The current mirror more closely approximates a current source as the output impedance becomes larger.

If we limit the voltage V_{C2} to small values relative to the Early voltage, I_{C2} is approximately equal to I_{C1}. For integrated circuit designs, the voltage required at the output of the current mirror is generally small, making this approximation valid.

The input current to the mirror is larger than the collector current and is:

$$I_{IN} = I_{C1} + 2I_B \tag{14.21}$$

Since $I_{OUT} = I_{C2} = I_{C1} = \beta I_B$, we can write Eq. 14.21 as:

$$I_{IN} = \beta I_B + 2I_B = (\beta + 2) I_B \tag{14.22}$$

Relating I_{IN} to I_{OUT} results in:

$$I_{OUT} = \frac{\beta}{\beta + 2} I_{IN} = \frac{I_{IN}}{1 + 2/\beta} \tag{14.23}$$

For typical values of β, these two currents are essentially equal. Thus, a desired bias current, I_{OUT}, is generated by creating the desired value of I_{IN}. The current I_{IN} is normally established by connecting a resistance R_1 to a voltage source V_{CC} to set I_{IN} to:

$$I_{IN} = \frac{V_{CC} - V_{BE}}{R_1} \tag{14.24}$$

Control of collector/bias current for Q_2 is then accomplished by choosing proper values of V_{CC} and R_1.

Figure 14.12 shows a multiple-output current mirror.

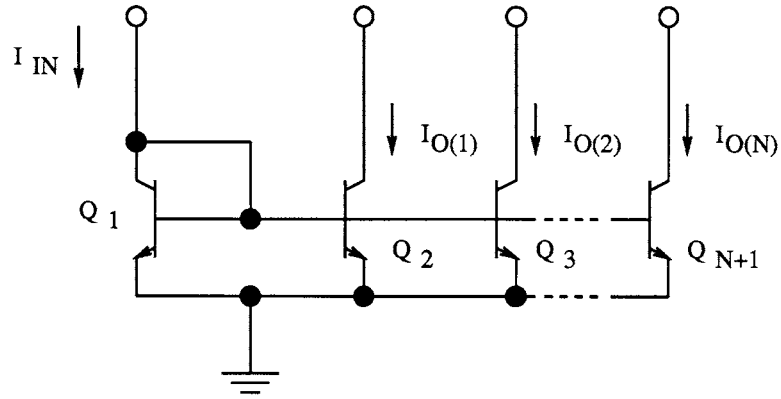


FIGURE 14.12 Multiple-output current mirror.

It can be shown that the output current for each identical device in Fig. 14.12 is:

$$I_O = \frac{I_{IN}}{1 + \dfrac{N+1}{\beta}} \tag{14.25}$$

where N is the number of output devices.

The current sinks can be turned into current sources by using pnp transistors and a power supply of opposite polarity. The output devices can also be scaled in area to make I_{OUT} be larger or smaller than I_{IN}.

Current Mirror with Reduced Error

The difference between output current in a multiple-output current mirror and the input current can become quite large if N is large. One simple method of avoiding this problem is to use an emitter follower to drive the bases of all devices in the mirror, as shown in Fig. 14.13.

The emitter follower, Q_0, has a current gain from base to collector of $\beta + 1$, reducing the difference between I_O and I_{IN} to:

$$I_{IN} - I_O = \frac{N+1}{\beta+1} I_B \tag{14.26}$$

The output current for each device is:

$$I_O = \frac{I_{IN}}{1 + \dfrac{N+1}{\beta(\beta+1)}} \tag{14.27}$$

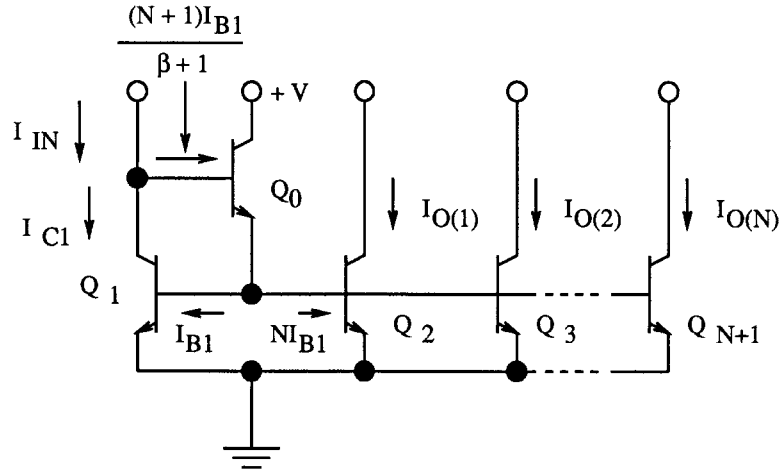


FIGURE 14.13 Improved multiple output current mirror.

The Wilson Current Mirror

In the simple current mirrors discussed, it was assumed that the collector voltage of the output stage was small compared to the Early voltage. When this is untrue, the output current will not remain constant, but will increase as output voltage (V_{CE}) increases. In other words, the output compliance range is limited with these circuits due to the finite output impedance of the BJT.

A modification of the improved output current mirror of Fig. 14.13 was proposed by Wilson[7] and is illustrated in Fig. 14.14.

The Wilson current mirror is connected such that $V_{CB2} = 0$ and $V_{BE1} = V_{BE0}$. Both Q_1 and Q_2 now operate with a near-zero collector–emitter bias although the collector of Q_0 might feed into a high-voltage point. It can be shown that the output impedance of the Wilson mirror is increased by a factor of β /2 over the simple mirror. This higher impedance translates into a higher output compliance. This circuit also reduces the difference between input and output current by means of the emitter follower stage.

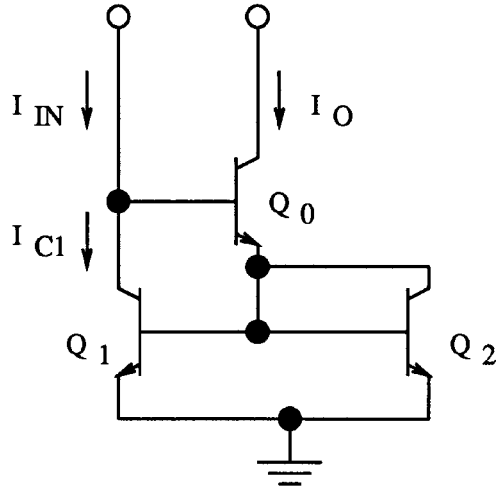


FIGURE 14.14 Wilson current mirror.

14.10 The Basic BJT Switch

In digital circuits, the BJT is used as a switch to generate one of only two possible output voltage levels, depending on the input voltage level. Each voltage level is associates with one of the binary digits, 0 or 1. Typically, the high voltage level may fall between 2.8 V and 5 V while the low voltage level may fall between 0 V and 0.8 V.

Logic circuits are based on BJT stages that are either in cutoff with both junctions reverse-biased or in a conducting mode with the emitter–base junction forward-biased. When the BJT is "on" or conducting emitter current, it can be in the active region or the saturation region. If it is in the saturation region, the collector–base region is also forward-biased.

The three possible regions of operation are summarized in Table 14.1.

TABLE 14.1 Regions of Operation

Region	Cutoff	Active	Saturation
C–B bias	Reverse	Reverse	Forward
E–B bias	Reverse	Forward	Forward

The BJT very closely approximates certain switch configurations. For example, when the switch of Fig. 14.15(a) is open, no current flows through the resistor and the output voltage is +12 V. Closing the switch causes the output voltage to drop to zero volts and a current of $12/R$ flows through the resistance. When the base voltage of the BJT of Fig. 14.15(b) is negative, the device is cut off and no collector current flows. The output voltage is +12 V, just as in the case of the open switch. If a large enough current is now driven into the base to saturate the BJT, the output voltage becomes very small, ranging from 20 mV to 500 mV, depending on the BJT used. The saturated state corresponds closely to the closed switch. During the time that the BJT switches from cutoff to saturation, the active region equivalent circuit applies. For high-speed switching of this circuit, appropriate reactive effects must be considered. For low-speed switching, these reactive effects can be neglected.

Saturation occurs in the basic switching circuit of Fig. 14.15(b) when the entire power supply voltage drops across the load resistance. No voltage, or perhaps a few tenths of volts, then appears from collector to emitter. This occurs when the base current exceeds the value

$$I_{B(sat)} = \frac{V_{CC} - V_{CE(sat)}}{\beta R_L} \tag{14.28}$$

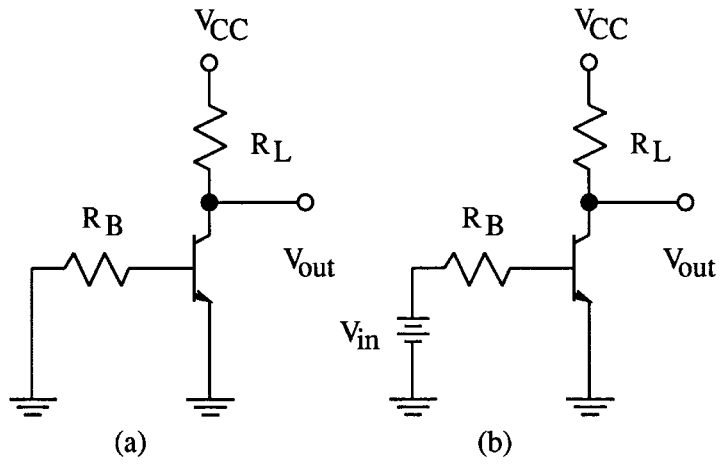


FIGURE 14.15 The BJT as a switch: (a) open switch, (b) closed switch.

When a transistor switch is driven into saturation, the collector–base junction becomes forward-biased. This situation results in the electron distribution across the base region shown in Fig. 14.16. The forward-bias of the collector–base junction leads to a non zero concentration of electrons in the base that is unnecessary to support the gradient of carriers across this region. When the input signal to the base switches to a lower level to either turn the device off or decrease the current flow, the excess charge must be removed from the base region before the current can begin to decrease.

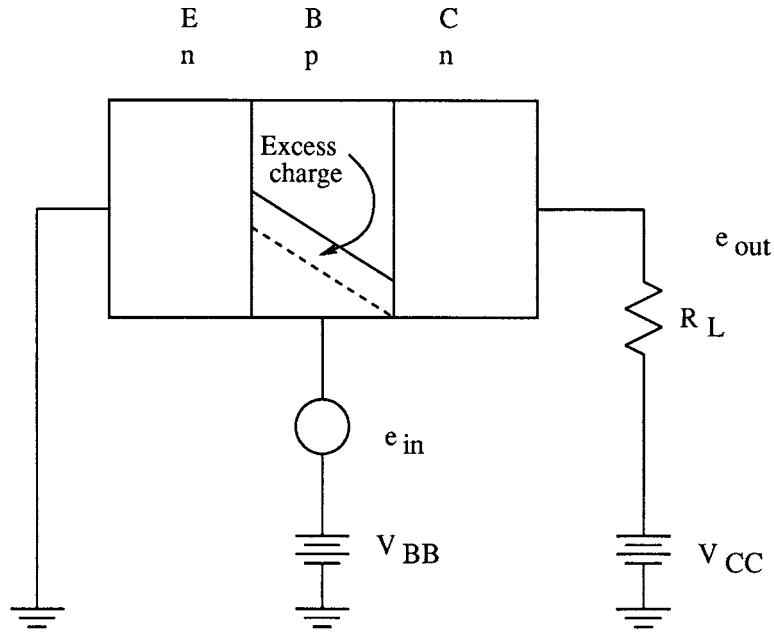


FIGURE 14.16 Electron distribution in the base region of a saturated BJT.

14.11 High-Speed BJT Switching

There are three major effects that extend switching times in a BJT:

1. The depletion-region or junction capacitances are responsible for delay time when the BJT is in the cutoff region.
2. The diffusion capacitance and the Miller-effect capacitance are responsible for the rise and fall times of the BJT as it switches through the active region.
3. The storage time constant accounts for the time taken to remove the excess charge from the base region before the BJT can switch from the saturation region to the active region.

There are other second-order effects that are generally negligible compared to the previously listed time lags.

Since the transistor is generally operating as a large-signal device, the parameters such as junction capacitance or diffusion capacitance will vary as the BJT switches. One approach to the evaluation of time constants is to calculate an average value of capacitance over the voltage swing that takes place. Not only is this method used in hand calculations, but most computer simulation programs use average values to speed calculations.

Overall Transient Response

Before discussing the individual BJT switching times, it is helpful to consider the response of a common-emitter switch to a rectangular waveform. Figure 14.17 shows a typical circuit using an npn transistor.

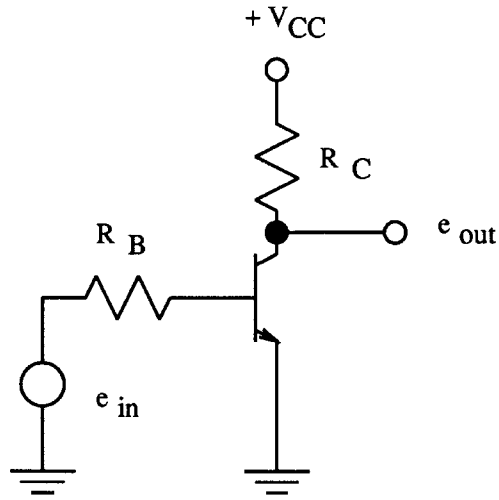


FIGURE 14.17 A simple switching circuit.

A rectangular input pulse and the corresponding output are shown in Fig. 14.18. In many switching circuits, the BJT must switch from its "off" state to saturation and later return to the "off" state. In this case, the delay time, rise time, saturation storage time, and fall time must be considered in that order to find the overall switching time.

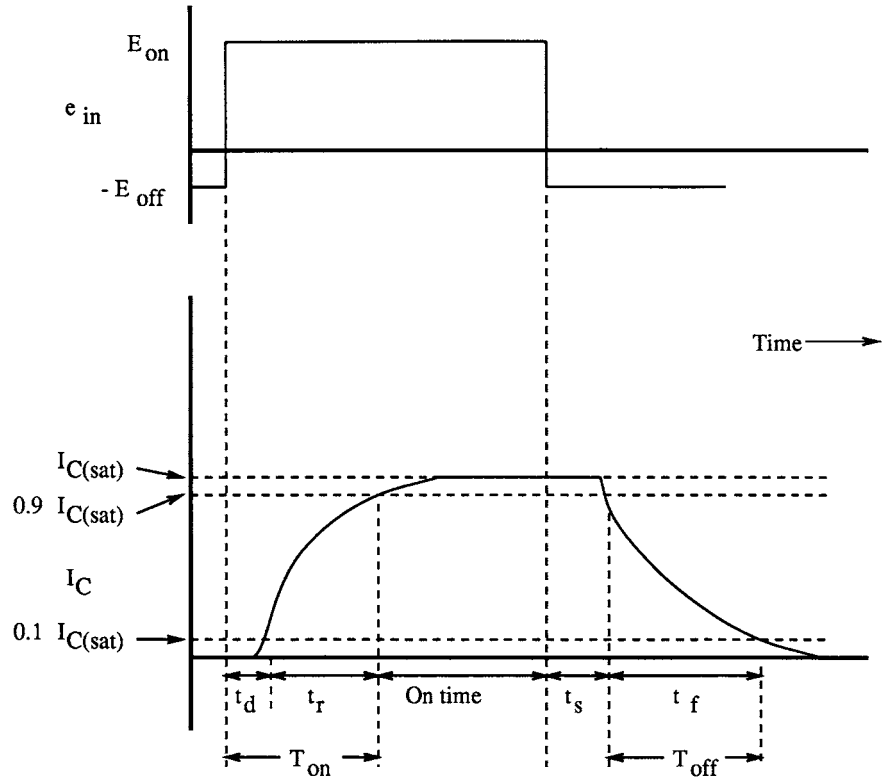


FIGURE 14.18 Input and output waveforms.

The total waveform is made up of five sections: delay time, rise time, on time, storage time, and fall time. The following list summarizes these points and serves as a guide for future reference:

t_d' = Passive delay time; time interval between application of forward base drive and start of collector-current response.
t_d = Total delay time; time interval between application of forward base drive and the point at which I_C has reached 10% of the final value.
t_r = Rise time; 10- to 90-% rise time of I_C waveform.
t_s' = Saturation storage time; time interval between removal of forward base drive and start of I_C decrease.
t_s = Total storage time; time interval between removal of forward base drive and point at which $I_C = 0.9I_{C(sat)}$.
t_f = Fall time; 90- to 10-% fall time of I_C waveform
T_{on} = Total turn-on time; time interval between application of base drive and point at which I_C has reached 90% of its final value.
T_{off} = Total turn-off time; time interval between removal of forward base drive and point at which I_C has dropped to 10% of its value during on time.

Not all applications will require evaluation of each of these switching times. For instance, if the base drive is insufficient to saturate the transistor, t_s will be zero. If the transistor never leaves the active region, the delay time will also be zero.

The factors involved in calculating the switching times are summarized in the following paragraphs.[8]

The passive delay time is found from:

$$t'_d = \tau_d \ln\left(\frac{E_{on} + E_{off}}{E_{on} - V_{BE(on)}}\right) \tag{14.29}$$

where τ_d is the product of the charging resistance and the average value of the two junction capacitances.

The active region time constant is a function of the diffusion capacitance, the collector–base junction capacitance, the transconductance, and the charging resistance. This time constant will be denoted by τ. If the transistor never enters saturation, the rise time is calculated from the well-known formula:

$$t_r = 2.2\tau \tag{14.30}$$

If the BJT is driven into saturation, the rise time is found from[8]:

$$t_r = \tau \ln\left(\frac{K - 0.1}{K - 0.9}\right) \tag{14.31}$$

where K is the overdrive factor or the ratio of forward base current drive to the value needed for saturation. The rise time for the case where K is large can be much smaller than the rise time for the nonsaturating case ($K < 1$). Unfortunately, the saturation storage time increases for large values of K.

The saturation storage time is given by:

$$t'_s = \tau_s \ln\left(\frac{I_{B1} - I_{B2}}{I_{B(sat)} - I_{B2}}\right) \tag{14.32}$$

where τ_s is the storage time constant, I_{B1} is the forward base current before switching, and I_{B2} is the current after switching and must be less than $I_{B(sat)}$. The saturation storage time can slow the overall switching time significantly. The higher speed logic gates utilize circuits that avoid the saturation region for the BJTs that make up the gate.

14.12 Simple Logic Gates

Although the resistor-transistor-logic (RTL) family has not been used since the late 1960s, it demonstrates the concept of a simple logic gate. Figure 14.19 shows a four-input RTL NOR gate.

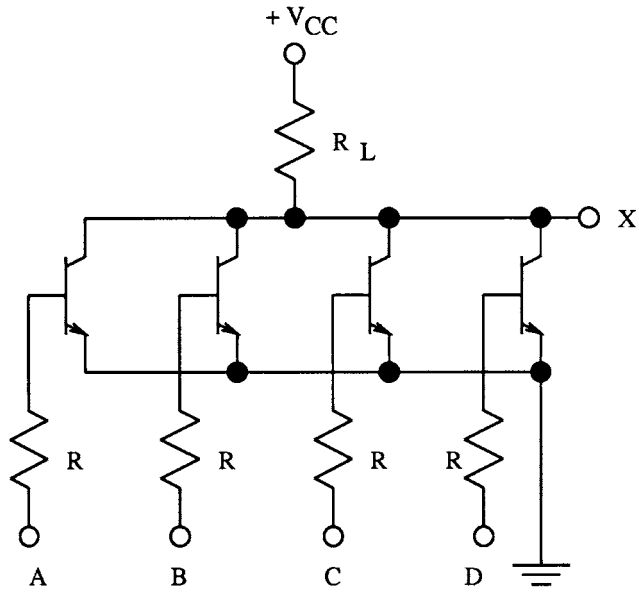


FIGURE 14.19 A four-input RTL NOR gate.

If all four inputs are at the lower voltage level (e.g., 0 V), there is no conducting path from output to ground. No voltage will drop across R_L, and the output voltage will equal V_{CC}. If any or all of the inputs moves to the higher voltage level (e.g., 4 V), any BJT with base connected to the higher voltage level will saturate, pulling the output voltage down to a few tenths of a volt. If positive logic is used, with the high voltage level corresponding to binary "1" and the low voltage level to binary "0," the gate performs the NOR function. Other logic functions can easily be constructed in the RTL family.

Over the years, the performance of logic gates has been improved by different basic configurations. RTL logic was improved by diode-transistor-logic (DTL). Then, transistor-transistor-logic (TTL) became very prominent. This family is still popular in the small-scale integration (SSI) and medium-scale integration (MSI) areas, but CMOS circuits have essentially replaced TTL in large-scale integration (LSI) and very-large-scale integration (VLSI) applications.

One popular family that is still prominent in very high-speed computer work is the emitter-coupled logic (ECL) family. While CMOS packs many more circuits into a given area than ECL, the frequency performance of ECL leads to its popularity in supercomputer applications.

14.13 Emitter-Coupled Logic

Emitter-coupled logic (ECL) was developed in the mid-1960s and remains the fastest silicon logic circuit available. Present ECL families offer propagation delays in the range of 0.2 ns.[9] The two major disadvantages of ECL are: (1) resistors which require a great deal of IC chip area, must be used in each gate, and (2) the power dissipation of an ECL gate is rather high. These two shortcomings limit the usage of ECL in VLSI systems. Instead, this family has been used for years in larger supercomputers than can afford space and power to achieve higher speeds.

The high speeds obtained with ECL are primarily based on two factors. No device in an ECL gate is ever driven into the saturation region and, thus, saturation storage time is never involved as devices switch from one state to another. The second factor is that required voltage swings are not large. Voltage excursions necessary to change an input from the low logic level to the high logic level are minimal. Although noise margins are lower than other logic families, switching times are reduced in this way.

Figure 14.20 shows an older ECL gate with two separate outputs. For positive logic, *X* is the OR output while *Y* is the NOR output.

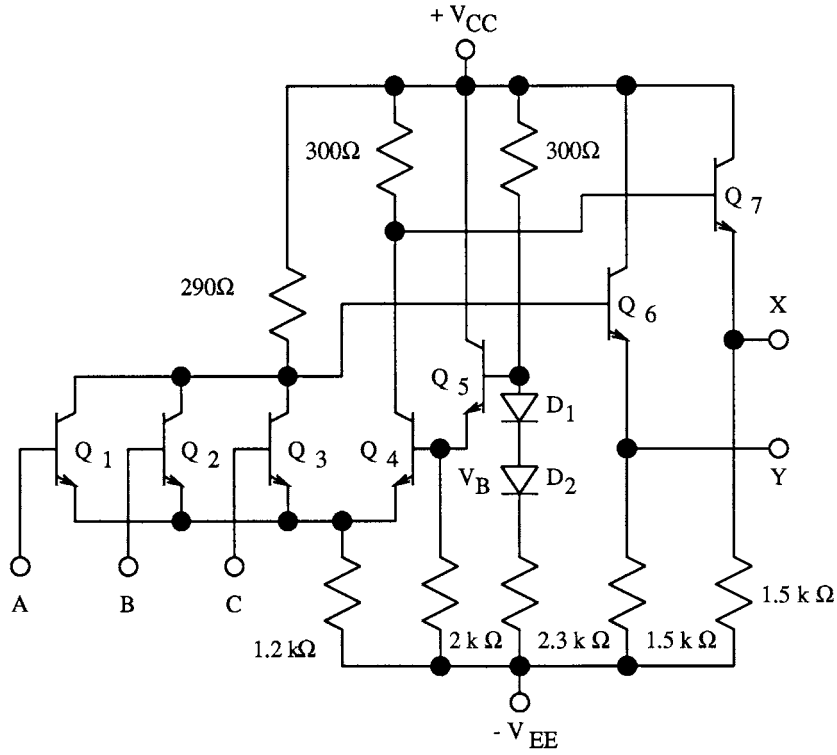


FIGURE 14.20 An ECL logic gate.

Often, the positive supply voltage is taken as 0 V and V_{EE} as –5 V due to noise considerations. The diodes and emitter follower Q_5 establish a temperature-compensated base reference for Q_4. When inputs *A*, *B*, and *C* are less than the voltage V_B, Q_4 conducts while Q_1, Q_2, and Q_3 are cut off. If any one of the inputs is switched to the 1 level, which exceeds V_B, the transistor turns on and pulls the emitter of Q_4 positive enough to cut this transistor off. Under this condition, output *Y* goes negative while *X* goes positive. The relatively large resistor common to the emitters of Q_1, Q_2, Q_3, and Q_4 prevents these transistors from saturating. In fact, with nominal logic levels of –1.9 V and –1.1 V, the current through the emitter resistance is approximately equal before and after switching takes place. Thus, only the current path changes as the circuit switches. This type of operation is sometimes called *current mode switching*. Although the output stages are emitter followers, they conduct reasonable currents for both logic level outputs and, therefore, minimize the asymmetrical output impedance problem.

In an actual ECL gate, the emitter follower load resistors are not fabricated on the chip. The newer version of the gate replaces the emitter resistance of the differential stage with a current source, and replaces the bias voltage circuit with a regulated voltage circuit.

A Closer Look at the Differential Stage

Figure 14.21 shows a simple differential stage similar to the input stage of an ECL gate.[2] Both transistors are biased by a current source, I_T, called the *tail current*. The two input signals e_1 and e_2 make up a differential input signal defined as:

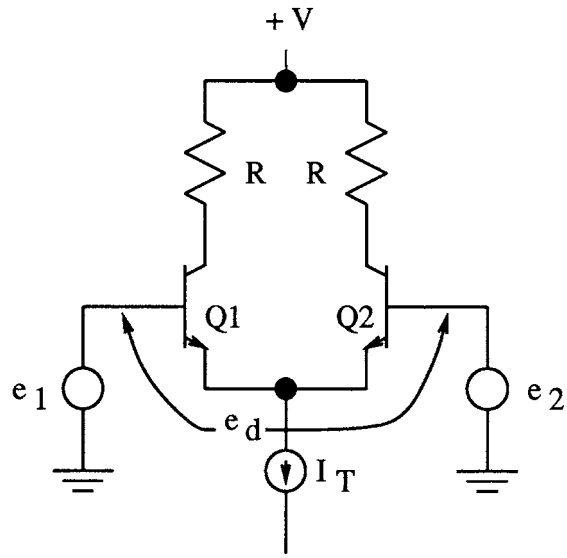


FIGURE 14.21 A simple differential stage similar to an ECL input stage.

$$e_d = e_1 - e_2 \tag{14.33}$$

This differential voltage can be expressed as the difference between the base–emitter junction voltages as:

$$e_d = V_{BE1} - V_{BE2} \tag{14.34}$$

The collector currents can be written in terms of the base–emitter voltages as:

$$I_{C1} = \alpha I_{EO} e^{V_{BE1}/V_T} \approx I_{EO} e^{V_{BE1}/V_T} \tag{14.35}$$

$$I_{C2} = \alpha I_{EO} e^{V_{BE2}/V_T} \approx I_{EO} e^{V_{BE2}/V_T} \tag{14.36}$$

where matched devices are assumed.

A differential output current can be defined as the difference of the collector currents, or

$$I_d = I_{C1} - I_{C2} \tag{14.37}$$

Since the tail current is $I_T = I_{C1} + I_{C2}$, taking the ratio of I_d to I_T gives:

$$\frac{I_d}{I_T} = \frac{I_{C1} - I_{C2}}{I_{C1} + I_{C2}} \tag{14.38}$$

Since $V_{BE1} = e_d + V_{BE2}$, we can substitute this value for V_{BE1} into Eq. 14.35 to write:

$$I_{C1} = I_{EO} e^{\left(e_d + V_{BE2}\right)/V_T} = I_{EO} e^{e_d/V_T} e^{V_{BE2}/V_T} \tag{14.39}$$

Substituting Eqs. 14.36 and 14.39 into Eq. 14.38 results in:

$$\frac{I_d}{I_T} = \frac{e^{e_d/V_T} - 1}{e^{e_d/V_T} + 1} = \tanh\frac{e_d}{2V_T} \tag{14.40}$$

or

$$I_d = I_T \tanh\frac{e_d}{2V_T} \tag{14.41}$$

This differential current is graphed in Fig. 14.22.

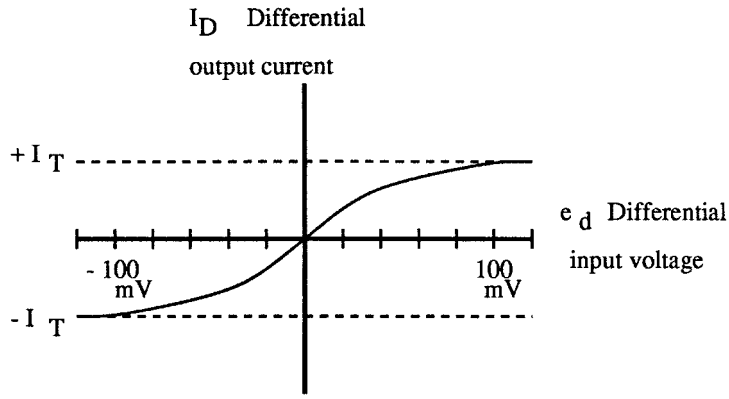


FIGURE 14.22 Differential output current as a function of differential input voltage.

When e_d is zero, the differential current is also zero, implying equal values of collector currents in the two devices. As e_d increases, so also does I_d until e_d exceeds $4V_T$, at which time I_d has reached a constant value of I_T. From the definition of differential current, this means that I_{C1} equals I_T while I_{C2} is zero. As the differential input voltage goes negative, the differential current approaches $-I_T$ as the voltage reaches $-4V_T$. In this case, $I_{C2} = I_T$ while I_{C1} goes to zero.

The implication here is that the differential stage can move from a balanced condition with $I_{C1} = I_{C2}$ to a condition of one device fully off and the other fully on with an input voltage change of around 100 mV or $4V_T$. This demonstrates that a total voltage change of about 200 mV at the input can cause an ECL gate to change states. This small voltage change contributes to smaller switching times for ECL logic.

The ability of a differential pair to convert a small change in differential base voltage to a large change in collector voltage also makes it a useful building block for analog amplifiers. In fact, a differential pair with a pnp transistor current mirror load, as illustrated in Fig. 14.23, is widely used as an input stage for integrated circuit op-amps.

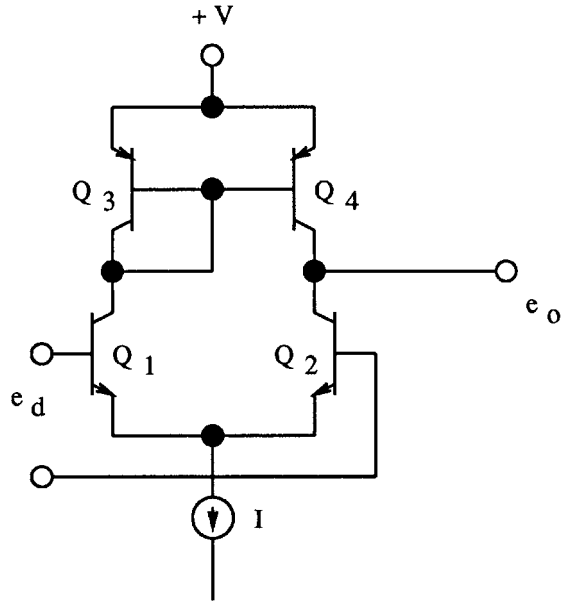


FIGURE 14.23 Differential input stage with current mirror load.

References

1. Brittain, J. E. (Ed.), *Turning Points in American Electrical History,* IEEE Press, New York, 1977, Sec. II-D.
2. Comer, D. T., *Introduction to Mixed Signal VLSI*, Array Publishing, New York, 1994, Ch. 7.
3. Sedra, A. S. and Smith, K. C., *Microelectronic Circuits, 4th ed.,* Oxford University Press, New York, 1998, Ch. 4.
4. Gray, P. R. and Meyer, R. G., *Analysis and Design of Analog Integrated Circuits, 3rd ed.*, John Wiley & Sons, Inc., New York, 1993, Ch. 1.
5. Vladimirescu, A., *The Spice Book,* John Wiley & Sons, Inc., New York, 1994, Ch. 3.
6. Streetman, B. G., *Solid State Electronic Devices, 4th ed.*, Prentice-Hall, Englewood Cliffs, NJ, 1995, Ch. 7.
7. Wilson, G. R., "A monolithic junction FET - NPN operational amplifier," *IEEE Journal of Solid State Circuits*, Vol. SC-3, pp. 341-348, Dec. 1968.
8. Comer, D. J., *Modern Electronic Circuit Design*, Addison-Wesley, Reading, MA, 1977, Ch. 8.
9. Motorola Technical Staff, *High Performance ECL Data*, Motorola, Inc., Phoenix, AZ, 1993, Ch. 3.

Lee, T.H., et al. "RF Passive IC Components"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

15

RF Passive IC Components

Thomas H. Lee
Maria del Mar Hershenson
Sunderarajan S. Mohan
Kirad Samavati
C. Patrick Yue
Stanford University

15.1 Introduction

Passive energy storage elements are widely used in radio-frequency (RF) circuits. Although their impedance behavior often can be mimicked by compact active circuitry, it remains true that passive elements offer the largest dynamic range and the lowest power consumption. Hence, the highest performance will always be obtained with passive inductors and capacitors. Unfortunately, standard integrated circuit technology has not evolved with a focus on providing good passive elements. This chapter describes the limited palette of options available, as well as means to make the most use out of what is available.

15.2 Fractal Capacitors

Of capacitors, the most commonly used are parallel-plate and MOS structures. Because of the thin gate oxides now in use, capacitors made out of MOSFETs have the highest capacitance density of any standard IC option, with a typical value of approximately 7 fF/μm^2 for a gate oxide thickness of 5 nm. A drawback, however, is that the capacitance is voltage dependent. The applied potential must be well in excess of a threshold voltage in order to remain substantially constant. The relatively low breakdown voltage (on the order of 0.5 V/nm of oxide) also imposes an unwelcome constraint on allowable signal amplitudes. An additional drawback is the effective series resistance of such structures, due to the MOS channel resistance. This resistance is particularly objectionable at radio frequencies, since the impedance of the combination may be dominated by this resistive portion.

Capacitors that are free of bias restrictions (and that have much lower series resistance) may be formed out of two (or more) layers of standard interconnect metal. Such parallel-plate capacitors are quite linear and possess high breakdown voltage, but generally offer two orders of magnitude lower capacitance density than the MOSFET structure. This inferior density is the consequence of a conscious and continuing effort by technologists to keep low the capacitance between interconnect layers. Indeed, the

vertical spacing between such layers generally does not scale from generation to generation. As a result, the disparity between MOSFET capacitance density and that of the parallel-plate structure continues to grow as technology scales.

A secondary consequence of the low density is an objectionably high capacitance between the bottom plate of the capacitor and the substrate. This bottom-plate capacitance is often a large fraction of the main capacitance. Needless to say, this level of parasitic capacitance is highly undesirable.

In many circuits, capacitors can occupy considerable area, and an area-efficient capacitor is therefore highly desirable. Recently, a high-density capacitor structure using lateral fringing and fractal geometries has been introduced.[1] It requires no additional processing steps, and so it can be built in standard digital processes. The linearity of this structure is similar to that of the conventional parallel-plate capacitor. Furthermore, the bottom-plate parasitic capacitance of the structure is small, which makes it appealing for many circuit applications. In addition, unlike conventional metal-to-metal capacitors, the density of a fractal capacitor increases with scaling.

Lateral Flux Capacitors

Figure 15.1(a) shows a lateral flux capacitor. In this capacitor, the two terminals of the device are built using a single layer of metal, unlike a vertical flux capacitor, where two different metal layers must be used. As process technologies continue to scale, lateral fringing becomes more important. The lateral spacing of the metal layers, s, shrinks with scaling, yet the thickness of the metal layers, t, and the vertical spacing of the metal layers, t_{ox}, stay relatively constant. This means that structures utilizing lateral flux enjoy a significant improvement with process scaling, unlike conventional structures that depend on vertical flux. Figure 15.1(b) shows a scaled lateral flux capacitor. It is obvious that the capacitance of the structure of Fig. 15.1(b) is larger than that of Fig. 15.1(a).

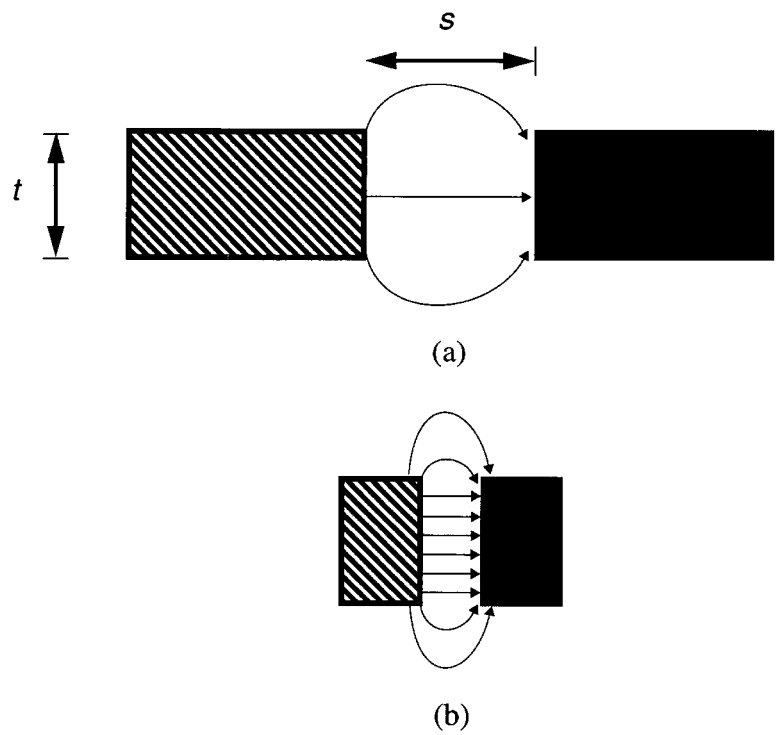


FIGURE 15.1 Effect of scaling on lateral flux capacitors: (a) before scaling and (b) after scaling.

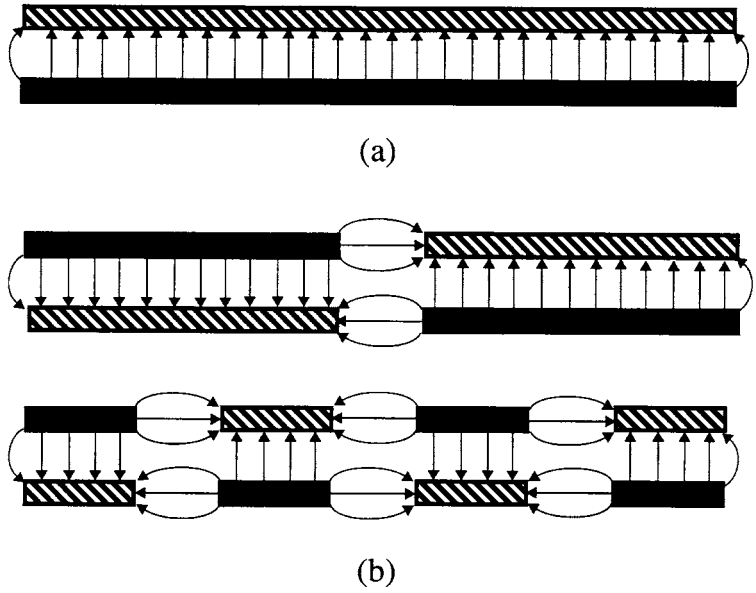


FIGURE 15.2 Vertical flux vs. lateral flux: (a) astandard parallel-plate structure, and (b) cross-connected metal layers.

Lateral flux can be used to increase the total capacitance obtained in a given area. Figure 15.2(a) is a standard parallel-plate capacitor. In Fig. 15.2(b), the plates are broken into cross-connected sections.[2] As can be seen, a higher capacitance density can be achieved by using lateral flux as well as vertical flux. To emphasize that the metal layers are cross connected, the two terminals of the capacitors in Fig. 15.2(b) are identified with two different shadings. The idea can be extended to multiple metal layers as well.

Figure 15.3 shows the ratio of metal thickness to minimum lateral spacing, *t/s*, vs. channel length for various technologies.[3–5] The trend suggests that lateral flux will have a crucial role in the design of capacitors in future technologies.

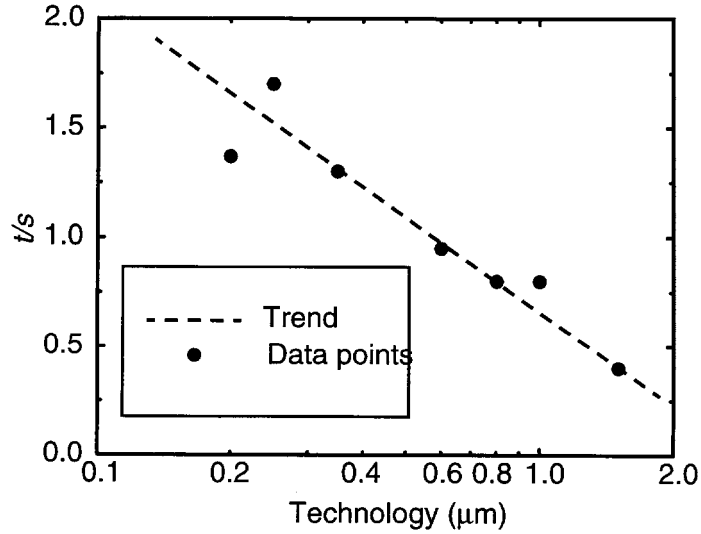


FIGURE 15.3 Ratio of metal thickness to horizontal metal spacing vs. technology (channel length).

The increase in capacitance due to fringing is proportional to the periphery of the structure; therefore, structures with large periphery per unit area are desirable. Methods for increasing this periphery are the subject of the following sections.

Fractals

A fractal is a mathematical abstract.[6] Some fractals are visualizations of mathematical formulas, while others are the result of the repeated application of an algorithm, or a *rule*, to a *seed*. Many natural phenomena can be described by fractals. Examples include the shapes of mountain ranges, clouds, coastlines, etc.

Some ideal fractals have finite area but infinite perimeter. The concept can be better understood with the help of an example. *Koch islands* are a family of fractals first introduced as a crude model for the shape of a coastline. The construction of a Koch curve begins with an *initiator*, as shown in the example of Fig. 15.4(a). A square is a simple initiator with $M = 4$ sides. The construction continues by replacing each segment of the initiator with a curve called a *generator*, an example of which is shown in Fig. 15.4(b) that has $N = 8$ segments. The size of each segment of the generator is $r = 1/4$ of the initiator. By recursively replacing each segment of the resulting curve with the generator, a fractal border is formed. The first step of this process is depicted in Fig. 15.4(c). The total area occupied remains constant throughout the succession of stages because of the particular shape of the generator. A more complicated Koch island can be seen in Fig. 15.5. The associated initiator of this fractal has four sides and its generator has 32 segments. It can be noted that the curve is self similar, that is, each section of it looks like the entire fractal. As we zoom in on Fig. 15.5, more detail becomes visible, and this is the essence of a fractal.

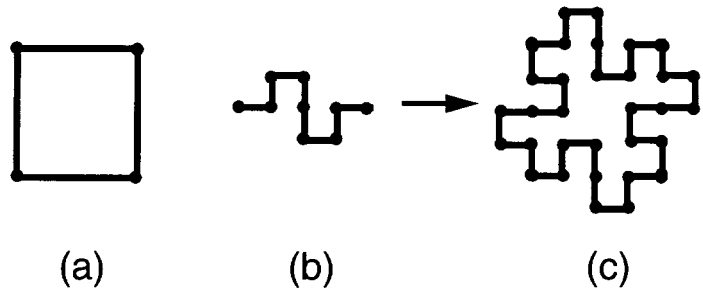


FIGURE 15.4 Construction of a Koch curve: (a) an initiator, (b) a generator, and (c) first step of the process.

Fractal dimension, *D*, is a mathematical concept that is a measure of the complexity of a fractal. The dimension of a flat curve is a number between 1 and 2, which is given by

$$D = \frac{\log(N)}{\log\left(\frac{1}{r}\right)} \tag{15.1}$$

where N is the number of segments of the generator and r is the ratio of the generator segment size to the initiator segment size. The dimension of a fractal curve is not restricted to integer values, hence the term "fractal." In particular, it exceeds 1, which is the intuitive dimension of curves. A curve that has a high degree of complexity, or *D*, fills out a two-dimensional flat surface more efficiently. The fractal in Fig. 15.4(c) has a dimension of 1.5, whereas for the border line of Fig.15.5, $D = 1.667$.

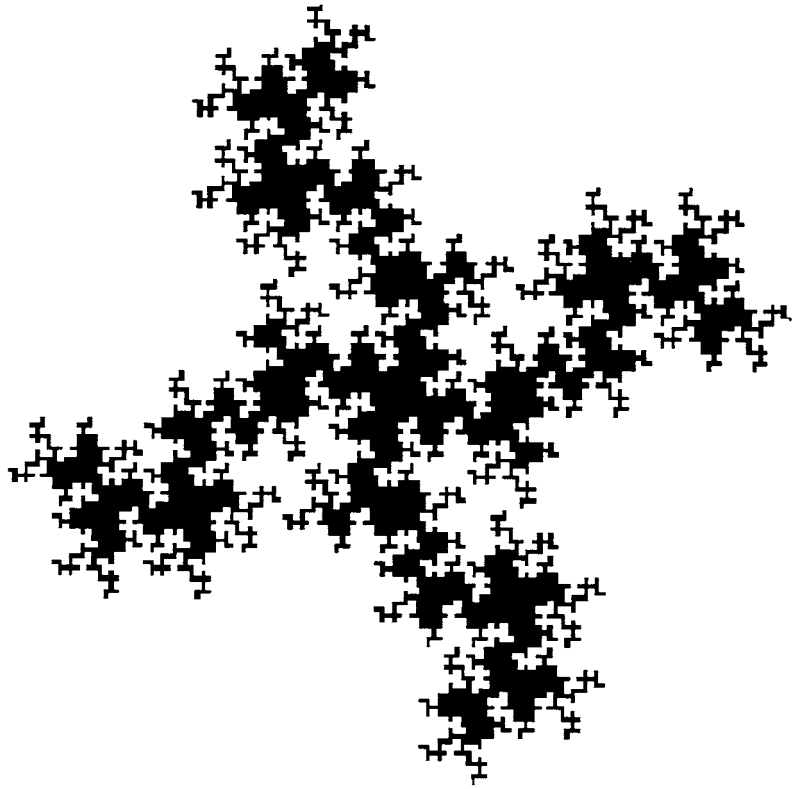


FIGURE 15.5 A Koch island with $M = 4$, $N = 32$, and $r = 1/8$.

For the general case where the initiator has M sides, the periphery of the initiator is proportional to the square root of the area:

$$P_0 = k \cdot \sqrt{A} \tag{15.2}$$

where k is a proportionality constant that depends on the geometry of the initiator. For example, for a square initiator, $k = 4$; and for an equilateral triangle, $k = 2 \cdot \sqrt[4]{27}$. After n successive applications of the generation rule, the total periphery is

$$P = k\sqrt{A} \cdot (Nr)^n \tag{15.3}$$

and the minimum feature size (the resolution) is

$$l = \frac{k\sqrt{A}}{M} \cdot r^n \tag{15.4}$$

Eliminating n from Eqs. 15.3 and 15.4 and combining the result with Eq. 15.1, we have

$$P = \frac{k^D}{M^{D-1}} \cdot \frac{(\sqrt{A})^D}{l^{D-1}} \tag{15.5}$$

Equation 15.5 demonstrates the dependence of the periphery on parameters such as the area and the resolution of the fractal border. It can be seen from Eq. 15.5 that as l tend toward zero, the periphery goes to infinity; therefore, it is possible to generate fractal structures with very large perimeters in any given area. However, the total periphery of a fractal curve is limited by the attainable resolution in practical realizations.

Fractal Capacitor Structures

The final shape of a fractal can be tailored to almost any form. The flexibility arises from the fact that a wide variety of geometries can be used as the initiator and generator. It is also possible to use different generators during each step. This is an advantage for integrated circuits where flexibility in the shape of the layout is desired.

Figure 15.6 is a three-dimensional representation of a fractal capacitor. This capacitor uses only one metal layer with a fractal border. For a better visualization of the overall picture, the terminals of this square-shaped capacitor have been identified using two different shadings. As was discussed before, multiple cross-connected metal layers may be used to improve capacitance density further.

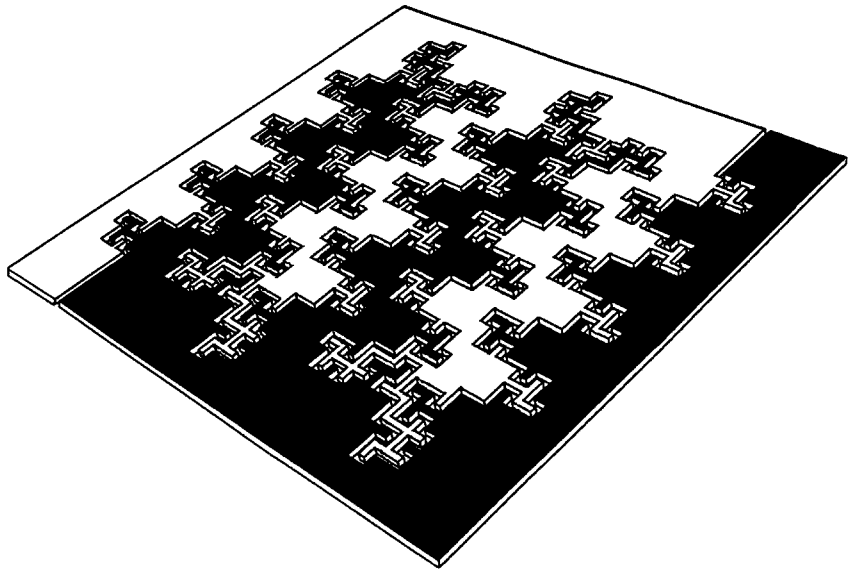


FIGURE 15.6 3-D representation of a fractal capacitor using a single metal layer.

One advantage of using lateral flux capacitors in general, and fractal capacitors in particular, is the reduction of the bottom-plate capacitance. This reduction is due to two reasons. First, the higher density of the fractal capacitor (compared to a standard parallel-plate structure) results in a smaller area. Second, some of the field lines originating from one of the bottom plates terminate on the adjacent plate, instead of the substrate, which further reduces the bottom-plate capacitance as shown in Fig. 15.7. Because of this property, some portion of the parasitic bottom-plate capacitor is converted into the more useful plate-to-plate capacitance.

The capacitance per unit area of a fractal structure depends on the dimension of the fractal. To improve the density of the layout, fractals with large dimensions should be used. The concept of fractal dimension is demonstrated in Fig. 15.8. The structure in Fig. 15.8(a) has a lower dimension compared to the one in Fig. 15.8(b), so the density (capacitance per unit area) of the latter is higher.

To demonstrate the dependence of capacitance density on dimension and lateral spacing of the metal layers, a first-order electromagnetic simulation was performed on two families of fractal

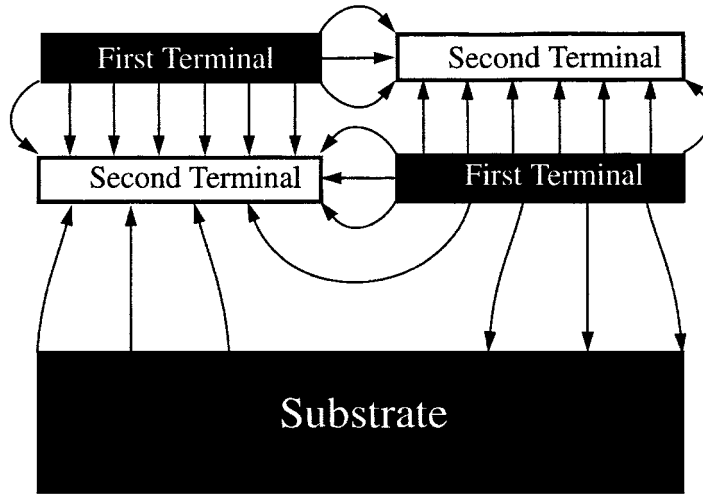


FIGURE 15.7 Reduction of the bottom-plate parasitic capacitance.

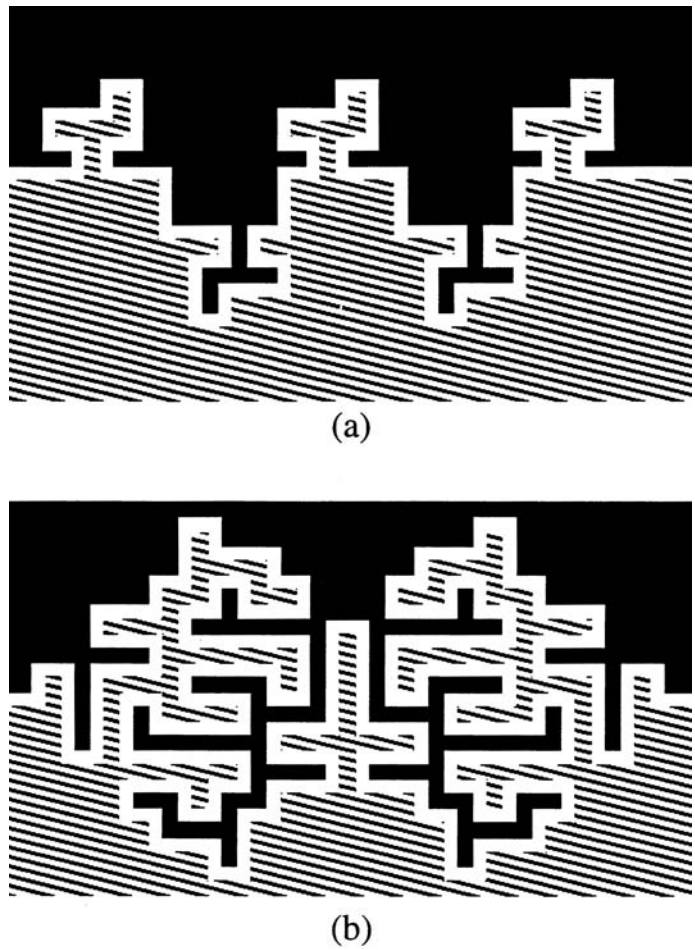


FIGURE 15.8 Fractal dimension of (a) is smaller than (b).

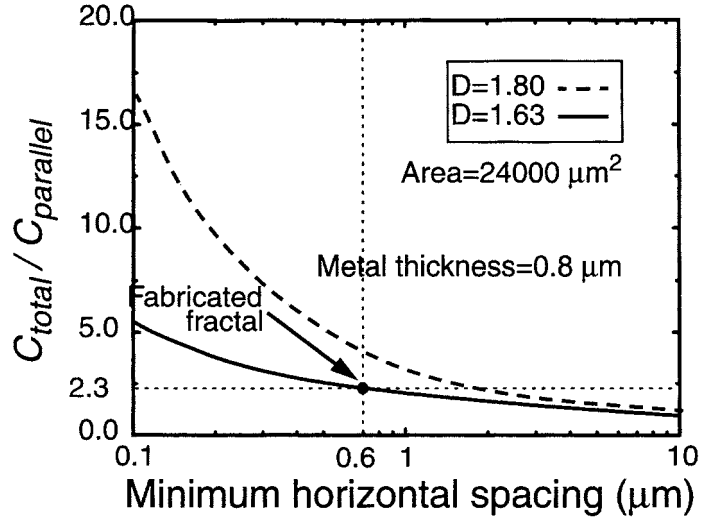


FIGURE 15.9 Boost factor vs. lateral spacing.

structures. In Fig. 15.9, the boost factor is plotted vs. horizontal spacing of the metal layers. The *boost factor* is defined as the ratio of the total capacitance of the fractal structure to the capacitance of a standard parallel-plate structure with the same area. The solid line corresponds to a family of fractals with a moderate fractal dimension of 1.63, while the dashed line represents another family of fractals with $D = 1.80$, which is a relatively large value for the dimension. In this first-order simulation, it is assumed that the vertical spacing and the thickness of the metal layers are kept constant at a 0.8-μm level. As can be seen in Fig. 15.9, the amount of boost is a strong function of the fractal dimension as well as scaling.

In addition to the capacitance density, the quality factor, *Q*, is important in RF applications. Here, the degradation in quality factor is minimal because the fractal structure automatically limits the length of the thin metal sections to a few microns, keeping the series resistance reasonably small. For applications that require low series resistance, lower dimension fractals may be used. Fractals thus add one more degree of freedom to the design of capacitors, allowing the capacitance density to be traded for a lower series resistance.

In current IC technologies, there is usually tighter control over the lateral spacing of metal layers compared to the vertical thickness of the oxide layers, from wafer to wafer and across the same wafer. Lateral flux capacitors shift the burden of matching away from oxide thickness to lithography. Therefore, by using lateral flux, matching characteristics can improve. Furthermore, the pseudo-random nature of the structure can also compensate, to some extent, the effects of non-uniformity of the etching process. To achieve accurate ratio matching, multiple copies of a unit cell should be used, as is standard practice in high-precision analog circuit design.

Another simple way of increasing capacitance density is to use an interdigitated capacitor depicted in Fig. 15.10.[2,7] One disadvantage of such a structure compared to fractals is its inherent parasitic inductance. Most of the fractal geometries randomize the direction of the current flow and thus reduce the effective series inductance; whereas for interdigitated capacitors, the current flow is in the same direction for all the parallel stubs. In addition, fractals usually have lots of rough edges that accumulate electrostatic energy more efficiently compared to interdigitated capacitors, causing a boost in capacitance (generally of the order of 15%). Furthermore, interdigitated structures are more vulnerable to non-uniformity of the etching process. However, the relative simplicity of the interdigitated capacitor does make it useful in some applications.

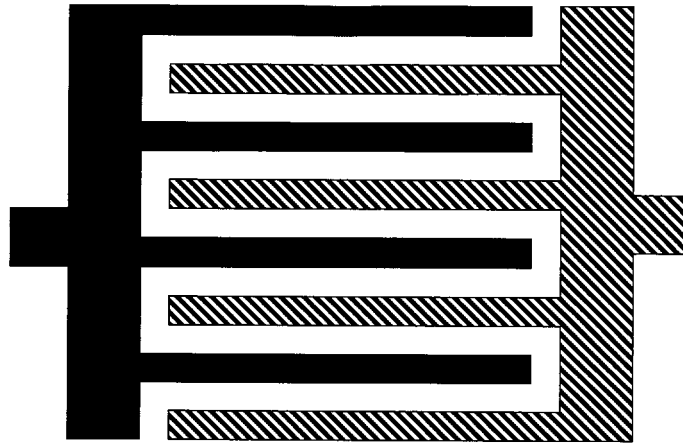


FIGURE 15.10 An interdigitated capacitor.

The woven structure shown in Fig. 15.11 may also be used to achieve high capacitance density. The vertical lines are in metal-2 and horizontal lines are in metal-1. The two terminals of the capacitor are identified using different shades. Compared to an interdigitated capacitor, a woven structure has much less inherent series inductance. The current flowing in different directions results in a higher self-resonant frequency. In addition, the series resistance contributed by vias is smaller than that of an interdigitated capacitor, because cross-connecting the metal layers can be done with greater ease. However, the capacitance density of a woven structure is smaller compared to an interdigitated capacitor with the same metal pitch, because the capacitance contributed by the vertical fields is smaller.

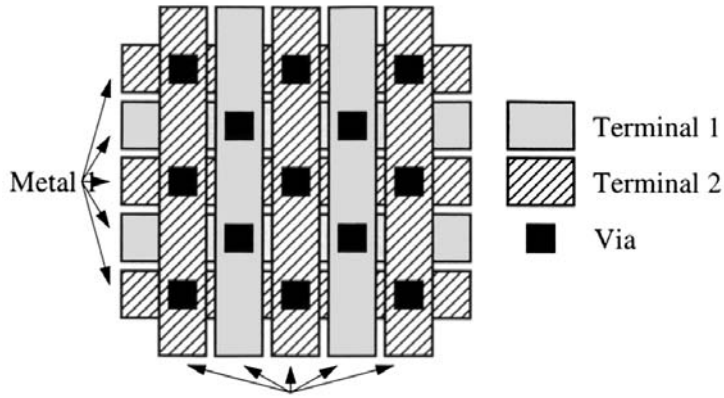


FIGURE 15.11 A woven structure.

15.3 Spiral Inductors

More than is so with capacitors, on-chip inductor options are particularly limited and unsatisfactory. Nevertheless, it is possible to build practical spiral inductors with values up to perhaps 20 nH and with Q values of approximately 10. For silicon-based RF ICs, Q degrades at high frequencies due to energy dissipation in the semiconducting substrate.[8] Additionally, noise coupling via the substrate at GHz frequencies has been reported.[9] As inductors occupy substantial chip area, they can potentially be the

source and receptor of detrimental noise coupling. Furthermore, the physical phenomena underlying the substrate effects are complicated to characterize. Therefore, decoupling the inductor from the substrate can enhance the overall performance by increasing Q, improving isolation, and simplifying modeling.

Some approaches have been proposed to address the substrate issues; however, they are accompanied by drawbacks. Some[10] have suggested the use of high-resistivity (150 to 200 Ω-cm) silicon substrates to mimic the low-loss semi-insulating GaAs substrate, but this is rarely a practical option. Another approach selectively removes the substrate by etching a pit under the inductor.[11] However, the etch adds extra processing cost and is not readily available. Moreover, it raises reliability concerns such as packaging yield and long-term mechanical stability. For low-cost integration of inductors, the solution to substrate problems should avoid increasing process complexity.

In this section, we present the *patterned ground shield* (PGS),[23] which is compatible with standard silicon technologies, and which reduces the unwanted substrate effects. The great improvement provided by the PGS reduces the disparity in quality between spiral inductors made in silicon and GaAs IC technologies.

Understanding Substrate Effects

To understand why the PGS should be effective, consider first the physical model of an ordinary inductor on silicon, with one port and the substrate grounded, as shown in Fig. 15.12.[8] An on-chip inductor is physically a three-port element including the substrate. The one-port connection shown in Fig. 15.12 avoids unnecessary complexity in the following discussion and at the same time preserves the inductor characteristics. In the model, the series branch consists of L_s, R_s, and C_s. L_s represents the spiral inductance, which can be computed using the Greenhouse method[12] or well-approximated by simple analytical formulas to be presented later. R_s is the metal series resistance whose behavior at RF is governed by the eddy current effect. This resistance accounts for the energy loss due to the skin effect in the spiral interconnect structure as well as the induced eddy current in any conductive media close to the inductor. The series feedforward capacitance, C_s, accounts for the capacitance due to the overlaps between the spiral and the center-tap underpass.[13] The effect of the inter-turn fringing capacitance is usually small because the adjacent turns are almost at equal potentials, and therefore it is neglected in this model. The overlap capacitance is more significant because of the relatively large potential difference between the spiral and the center-tap underpass. The parasitics in the shunt branch are modeled by C_{ox}, C_{Si}, and R_{Si}. C_{ox} represents the oxide capacitance between the spiral and the substrate. The silicon substrate capacitance and resistance are modeled by C_{Si} and R_{Si}, respectively.[14,15] The element R_{Si} accounts for the energy dissipation in the silicon substrate.

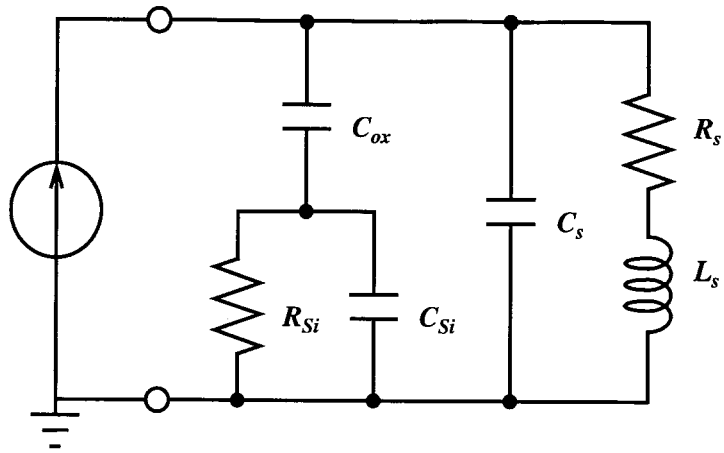


FIGURE 15.12 Lumped physical model of a spiral inductor on silicon.

Expressions for the model element values are as follows:

$$R_s = \frac{\rho l}{\delta w\left(1 - e^{-\frac{t}{\delta}}\right)} \tag{15.6}$$

$$C_s = nw^2 \cdot \frac{\varepsilon_{ox}}{t_{oxM1-M2}} \tag{15.7}$$

$$C_{ox} = \frac{\varepsilon_{ox}}{2t_{ox}} \cdot l \cdot w \tag{15.8}$$

$$C_{Si} = \frac{1}{2} \cdot l \cdot w \cdot C_{sub} \tag{15.9}$$

$$R_{Si} = \frac{2}{l \cdot w \cdot G_{sub}} \tag{15.10}$$

where ρ is the DC resistivity of the spiral; t is the overall length of the spiral windings; w is the line width; δ is the skin depth; n is the number of crossovers between the spiral and center-tap (and thus $n = N - 1$, where N is the number of turns); $t_{oxM1-M2}$ is the oxide thickness between the spiral and substrate; C_{sub} is the substrate capacitance per unit area; and G_{sub} is the substrate conductance per unit area. In general, one treats C_{sub} and G_{sub} as fitting parameters.

Exploration with the model reveals that the substrate loss stems primarily from the penetration of the electric field into the lossy silicon substrate. As the potential drop in the semiconductor (i.e., across R_{Si} in Fig. 15.12) increases with frequency, the energy dissipation in the substrate becomes more severe. It can be seen that increasing R_p to infinity reduces the substrate loss. It can be shown that R_p approaches infinity as R_{Si} goes either to zero or infinity. This observation implies that Q can be improved by making the silicon substrate *either* a perfect insulator or a perfect conductor. Using high-resistivity silicon (or etching it away) is equivalent to making the substrate an open circuit. In the absence of the freedom to do so, the next best option is to convert the substrate into a better conductor. The approach is to insert a ground plane to block the inductor electric field from entering the silicon. In effect, this ground plane becomes a pseudo-substrate with the desired characteristics.

The ground shield cannot be a solid conductor, however, because image currents would be induced in it. These image currents tend to cancel the magnetic field of the inductor proper, decreasing the inductance. To solve this problem, the ground shield is patterned with slots orthogonal to the spiral as illustrated in Fig. 15.13. The slots act as an open circuit to cut off the path of the induced loop current. The slots should be sufficiently narrow such that the vertical electric field cannot leak through the patterned ground shield into the underlying silicon substrate. With the slots etched away, the ground strips serve as the termination for the electric field. The ground strips are merged together around the four outer edges of the spiral. The separation between the merged area and the edges is not critical. However, it is crucial that the merged area not form a closed ring around the spiral since it can potentially support unwanted loop current. The shield should be strapped with the top layer metal to provide a low-impedance path to ground. The general rule is to prevent negative mutual coupling while minimizing the impedance to ground.

The shield resistance is another critical design parameter. The purpose of the patterned ground shield is to provide a good short to ground for the electric field. Since the finite shield resistance contributes to energy loss of the inductor, it must be kept small. Specifically, by keeping the shield resistance small compared to the reactance of the oxide capacitance, the voltage drop that can develop across the shield resistance is very small. As a result, the energy loss due to the shield resistance is insignificant compared to other losses. A typical on-chip spiral inductor has parasitic oxide capacitance between 0.25 and 1 pF,

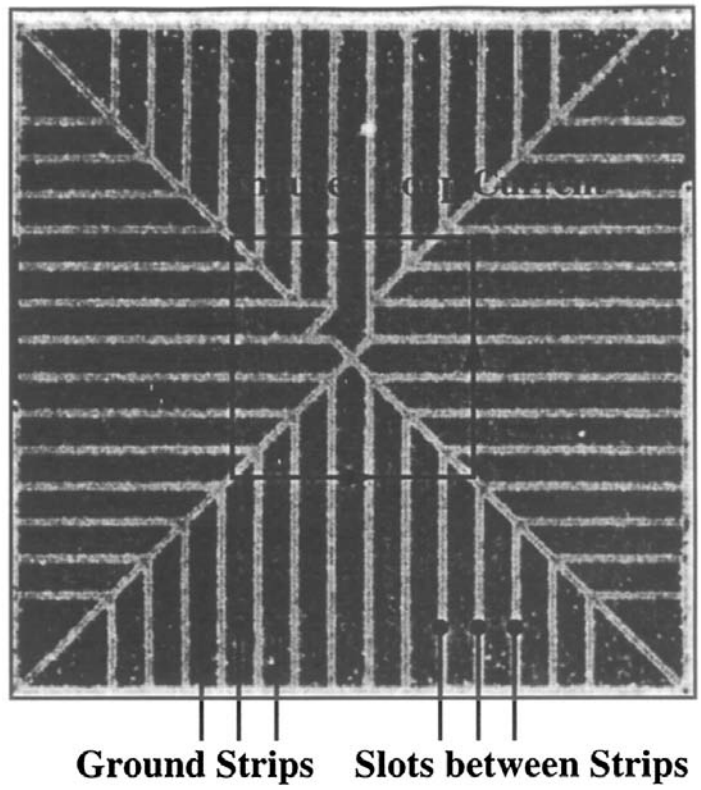


FIGURE 15.13 A close-up photo of the patterned ground shield.

depending on the size and the oxide thickness. The corresponding reactance due to the oxide capacitance at 1 to 2 GHz is of the order of 100 Ω, and hence a shield resistance of a few ohms is sufficiently small not to cause any noticeable loss.

With the PGS, one can expect typical improvements in Q ranging from 10 to 33%, in the frequency range of 1 to 2 GHz. Note that the inclusion of the ground shields increases C_p, which causes a fast roll-off in Q above the peak-Q frequency and a reduction in the self-resonant frequency. This modest improvement in inductor Q is certainly welcome, but is hardly spectacular by itself. However, a more dramatic improvement is evident when evaluating inductor-capacitor resonant circuits. Such LC tank circuits can absorb the parasitic capacitance of the ground shield. Since the energy stored in such parasitic elements is now part of the circuit, the overall circuit Q is greatly increased. Improvements of factors of approximately two are not unusual, so that tank circuits realized with PGS inductors possess roughly the same Q as those built in GaAs technologies.

As stated earlier, substrate noise coupling can be an issue of great concern owing to the relatively large size of typical inductors. Shielding by the PGS improves isolation by 25 dB or more at GHz frequencies. It should be noted that, as with any other isolation structure (such as a guard ring), the efficacy of the PGS is highly dependent on the integrity of the ground connection. One must often make a tradeoff between the desired isolation level and the chip area that is required to provide a low-impedance ground connection.

Simple, Accurate Expressions for Planar Spiral Inductances

In the previous section, a physically based model for planar spiral inductors was offered, and reference was made to the Greenhouse method as a means for computing the inductance value. This method uses

as computational atoms the self- and mutual inductances of parallel current strips. It is relatively straightforward to apply, and yields accurate results. Nevertheless, simpler analytic formulas are generally preferred for design since important insights are usually more readily obtained.

As a specific example, square spirals are popular mainly because of their ease of layout. Other polygonal spirals have also been used to improve performance by more closely approximating a circular spiral. However, a quantitative evaluation of possible improvements is cumbersome without analytical formulas for inductance.

Among alternative shapes, hexagonal and octagonal inductors are used widely. Figures 15.14 through 15.16 and show the layout for square, hexagonal, and octagonal inductors, respectively. For a given shape, an inductor is completely specified by the number of turns n, the turn width w, the turn spacing s, and any one of the following: the outer diameter d_{out}, the inner diameter d_{in}, the average diameter $d_{avg} = 0.5(d_{out} + d_{in})$, or the fill ratio, defined as $\rho = (d_{out} - d_{in})/(d_{out} + d_{in})$. The thickness of the inductor has only a very small effect on inductance and will therefore be ignored here.

We now present three approximate expressions for the inductance of square, hexagonal, and octagonal planar inductors. The first approximation is based on a modification of an expression developed by Wheeler[16]; the second is derived from electromagnetic principles by approximating the sides of the spirals as current sheets; and the third is a monomial expression derived from fitting to a large database of inductors (whose exact inductance values are obtained from a 3-D electromagnetic field solver). All three expressions are accurate, with typical errors of 2 to 3%, and very simple, and are therefore excellent candidates for use in design and optimization.

Modified Wheeler Formula

Wheeler[16] presented several formulas for planar spiral inductors, which were intended for discrete inductors. A simple modification of the original Wheeler formula allows us to obtain an expression that is valid for planar spiral integrated inductors:

$$L_{mw} = K_1 \mu_0 \frac{n^2 d_{avg}}{1 + K_2 \rho} \tag{15.11}$$

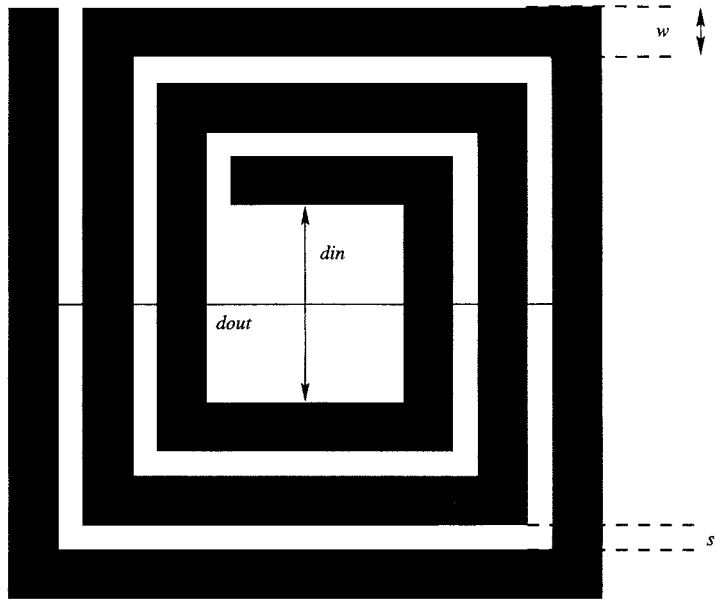


FIGURE 15.14 Square inductor.

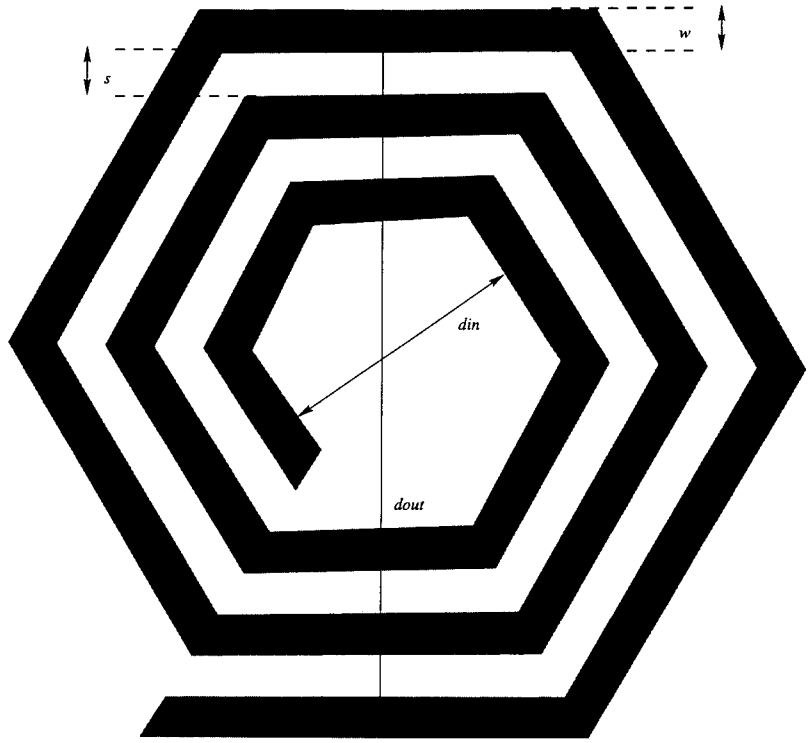


FIGURE 15.15 Hexagonal inductor.

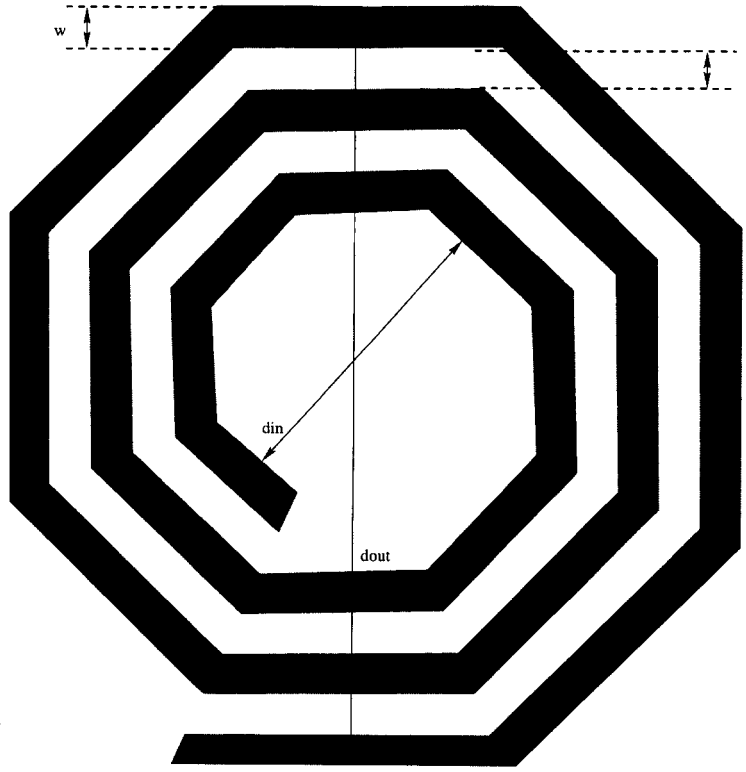


FIGURE 15.16 Octagonal inductor.

where ρ is the fill ratio defined previously. The coefficients K_1 and K_2 are layout dependent and are shown in Table 15.1

TABLE 15.1 Coefficients for Modified Wheeler Formula

Layout	K_1	K_2
Square	2.34	2.75
Hexagonal	2.33	3.82
Octagonal	2.25	3.55

The fill factor ρ represents how hollow the inductor is: for small ρ, we have a hollow inductor ($d_{out} \cong d_{in}$), and for a large ρ we have a filled inductor ($d_{out} >> d_{in}$). Two inductors with the same average diameter but different fill ratios will, of course, have different inductance values; the filled one has a smaller inductance because its inner turns are closer to the center of the spiral, and so contribute less positive mutual inductance and more negative mutual inductance. Some degree of hollowness is generally desired since the innermost turns contribute little overall inductance, but significant resistance.

Expression Based on Current Sheet Approximation

Another simple and accurate expression for the inductance of a planar spiral can be obtained by approximating the sides of the spirals by symmetrical current sheets of equivalent current densities.[17] For example, in the case of the square, we obtain four identical current sheets: the current sheets on opposite sides are parallel to one another, whereas the adjacent ones are orthogonal. Using symmetry and the fact that sheets with orthogonal current sheets have zero mutual inductance, the computation of the inductance is now reduced to evaluating the self-inductance of one sheet and the mutual inductance between opposite current sheets. These self- and mutual inductances are evaluated using the concepts of geometric mean distance (GMD) and arithmetic mean distance (AMD).[17,18] The resulting expression is:

$$L_{gmd} = \frac{\mu n^2 d_{avg}}{\pi}(c_1(\log c_2/\rho) + c_3\rho) \tag{15.12}$$

where the coefficients c_i are layout dependent and are shown in Table 15.2

TABLE 15.2 Coefficients for Current-Sheet Inductance Formula

Layout	c_1	c_2	c_3
Square	2.00	2.00	0.54
Hexagonal	1.83	1.71	0.45
Octagonal	1.87	1.68	0.60

A detailed derivation of these formulas can be found in Ref. 19. Since this formula is based on a current sheet approximation, its accuracy worsens as the ratio *s/w* becomes large. In practice, this is not a problem since practical integrated spiral inductors are built with $s < w$. The reason is that a smaller spacing improves the inter-winding magnetic coupling and reduces the area consumed by the spiral. A large spacing is only desired to reduce the inter-winding capacitance. This is rarely a concern as this capacitance is always dwarfed by the under-pass capacitance.[8]

Data-Fitted Monomial Expression

Our final expression is based on a data-fitting technique, in which a population of thousands of inductors are simulated with an electromagnetic field solver. The inductors span the entire range of values of relevance to RF circuits. A monomial expression is then fitted to the data, which ultimately yields:

$$L_{mon} = \beta d_{avg}^{\alpha 1} w^{\alpha 2} d_{avg}^{\alpha 3} n^{\alpha 4} s^{\alpha 5} \tag{15.13}$$

where the coefficients β and α_i are layout dependent, and given in Table 15.3.

TABLE 15.3 Coefficients for Monomial Inductance Formula

Layout	b	α_1	α_2	α_3	α_4	α_5
Square	1.66×10^{-3}	-1.33	-0.13	2.50	1.83	-0.022
Hexagonal	1.33×10^{-3}	-1.46	-0.16	2.67	1.80	-0.030
Octagonal	1.34×10^{-3}	-1.35	-0.15	2.56	1.77	-0.032

Of course, it is also possible to use other data-fitting techniques; for example, one which minimizes the maximum error of the fit, or one in which the coefficients must satisfy given inequalities or bounds. The monomial expression is useful since, like the other expressions, it is very accurate and very simple. Its real value, however, is that it can be used for the optimal design of inductors and circuits containing inductors, using geometric programming, which is a type of optimization method that requires monomial models.[20,21]

Figure 15.17 shows the absolute error distributions of these expressions. The plots show that typical errors are in the 1 to 2% range, and most of the errors are below 3%. These expressions for inductance, while quite simple, are thus sufficiently accurate that field solvers are rarely necessary.

These expressions can be included in a physical, scalable lumped-circuit model for spiral inductors where, in addition to providing design insight, they allow efficient optimization schemes to be employed.

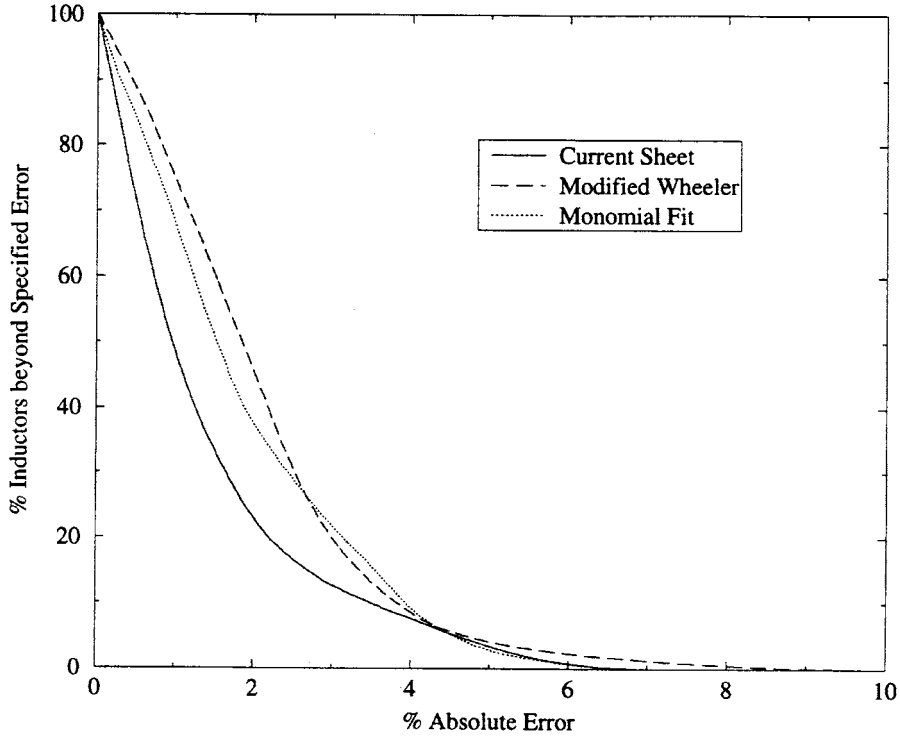


FIGURE 15.17 Error distribution for three formulas, compared to field solver simulations.

15.4 On-Chip Transformers

Transformers are important elements in RF circuits for impedance conversion, impedance matching, and bandwidth enhancement. Here, we present an analytical model for monolithic transformers that is suitable for circuit simulation and design optimization. We also provide simple expressions for calculating the mutual coupling coefficient (k).

We first discuss different on-chip transformers and their advantages and disadvantages. We then present an analytical model along with expressions for the elements in it and the mutual coupling coefficient.

Monolithic Transformer Realizations

Figures 15.18 through 15.23 illustrate common configurations of monolithic transformers. The different realizations offer varying tradeoffs among the self-inductance and series resistance of each port, the mutual coupling coefficient, the port-to-port and port-to-substrate capacitances, resonant frequencies, symmetry, and area. The models and coupling expressions allow these trade-offs to be systematically explored, thereby permitting transformers to be customized for a variety of circuit design requirements.

The characteristics desired of a transformer are application dependent. Transformers can be configured as three or four-terminal devices. They may be used for narrowband or broadband applications. For example, in single-sided to differential conversion, the transformer might be used as a four-terminal narrowband device. In this case, a high mutual coupling coefficient and high self-inductance are desired, along with low series resistance. On the other hand, for bandwidth extension applications, the transformer is used as a broadband three-terminal device. In this case, a small mutual coupling coefficient and high series resistance are acceptable, while all capacitances need to be minimized.[22]

The tapped transformer (Fig. 15.18) is best suited for three-port applications. It permits a variety of tapping ratios to be realized. This transformer relies only on lateral magnetic coupling. All windings can

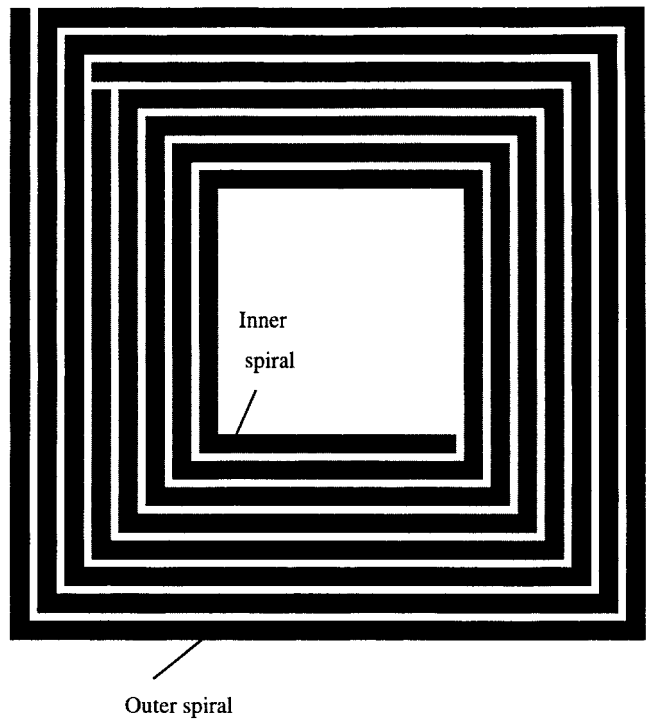


FIGURE 15.18 Tapped transformer.

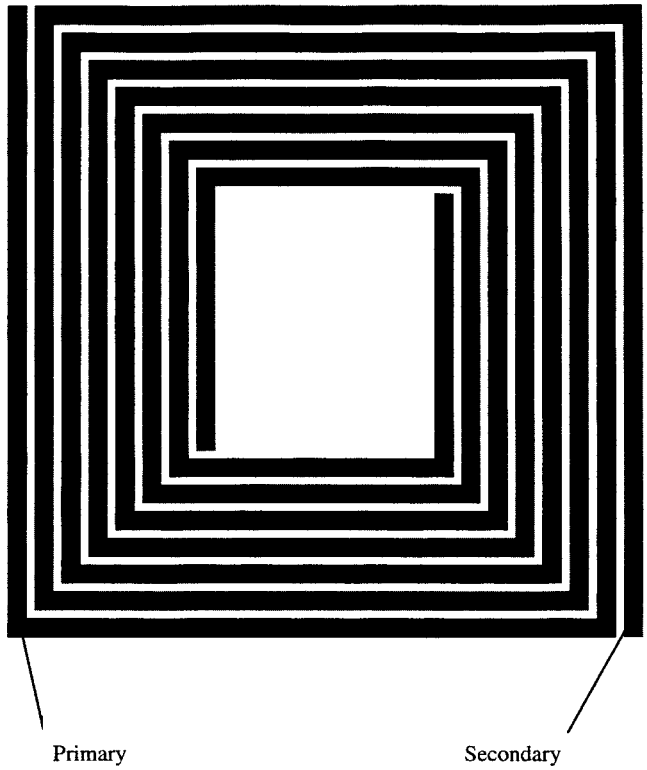


FIGURE 15.19 Interleaved transformer.

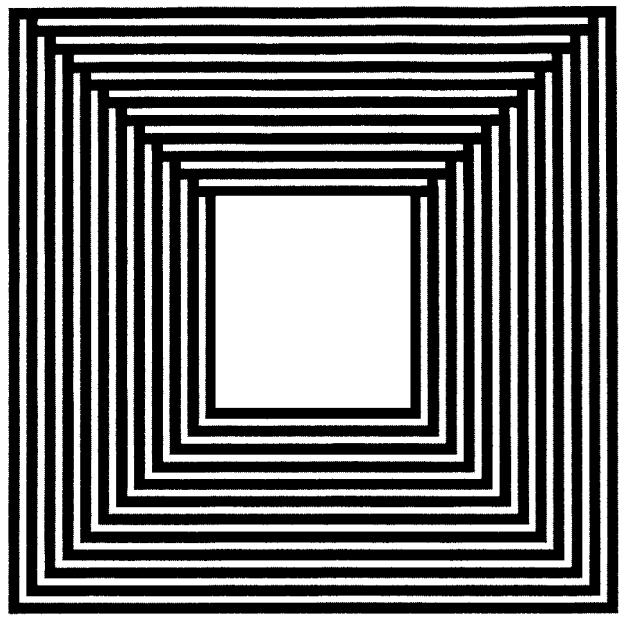


FIGURE 15.20 Stacked transformer with top spiral overlapping the bottom one.

be implemented with the top metal layer, thereby minimizing port-to-substrate capacitances. Since the two inductors occupy separate regions, the self-inductance is maximized while the port-to-port capacitance is minimized. Unfortunately, this spatial separation also leads to low mutual coupling ($k = 0.3$–0.5).

The interleaved transformer (Fig. 15.19) is best suited for four-port applications that demand symmetry. Once again, capacitances can be minimized by implementing the spirals with top level metal so that high resonant frequencies may be realized. The interleaving of the two inductances permit moderate coupling ($k = 0.7$) to be achieved at the cost of reduced self-inductance. This coupling may be increased at the cost of higher series resistance by reducing the turn width (w) and spacing (s).

The stacked transformer (Fig. 15.20) uses multiple metal layers and exploits both vertical and lateral magnetic coupling to provide the best area efficiency, the highest self-inductance, and highest coupling ($k = 0.9$). This configuration is suitable for both three- and four-terminal configurations. The main drawback is the high port-to-port capacitance, or equivalently a low self-resonant frequency. In some cases, such as narrowband impedance transformers, this capacitance may be incorporated as part of the resonant circuit. Also, in multi-level processes, the capacitance can be reduced by increasing the oxide thickness between spirals. For example, in a five-metal process, 50 to 70% reductions in port-to-port capacitance can be achieved by implementing the spirals on layers five and three instead of five and four. The increased vertical separation will reduce k by less than 5%. One can also trade off reduced coupling for reduced capacitance by displacing the centers of the stacked inductors (Figs. 15.21 and 15.22)

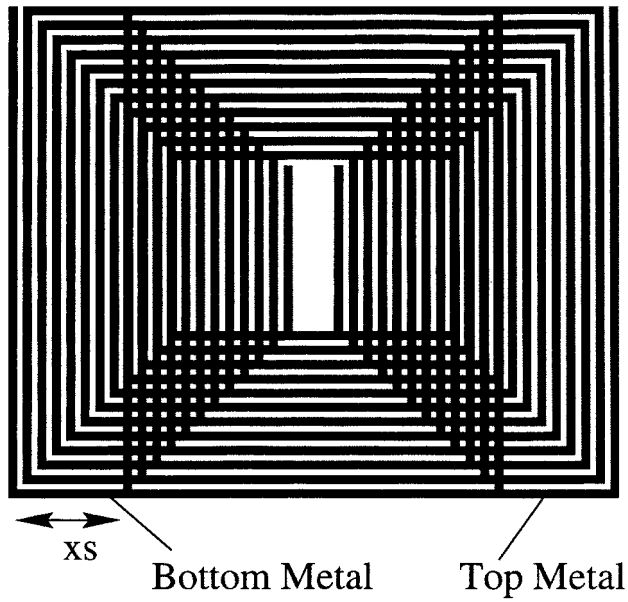


FIGURE 15.21 Stacked transformer with top and bottom spirals laterally shifted.

Analytical Transformer Models

Figures 15.23 and 15.24 present the circuit models for tapped and stacked transformers, respectively. The corresponding element values for the tapped transformer model are given by the following equations (subscript o refers to the outer spiral, i to the inner spiral, and T to the whole spiral):

$$L_T = \frac{9.375\mu_0 n_T^2 A D_T^2}{11 OD_T - 7 A D_T} \tag{15.14}$$

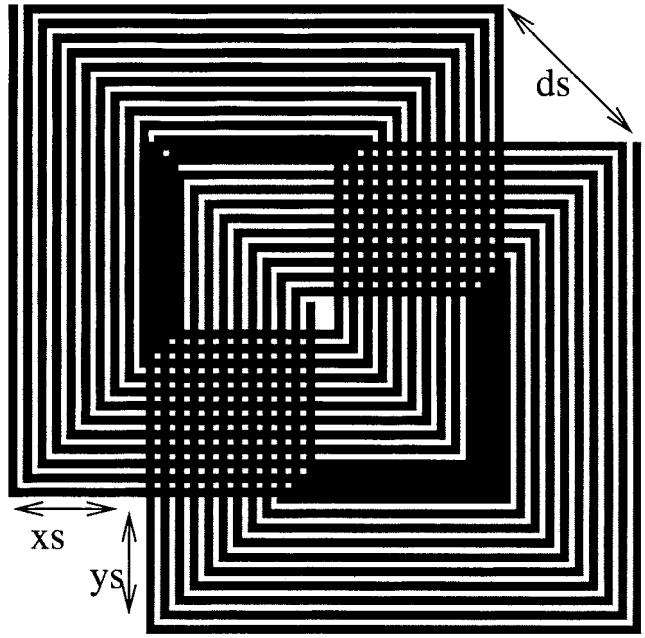


FIGURE 15.22 Stacked transformer with top and bottom spirals diagonally shifted.

$$L_o = \frac{9.375\mu_0 n_o^2 AD_o^2}{11OD_o - 7AD_o} \tag{15.15}$$

$$L_i = \frac{9.375\mu_0 n_i^2 AD_i^2}{11OD_i - 7AD_i} \tag{15.16}$$

$$M = \frac{L_T - L_o - L_i}{2\sqrt{L_o L_i}} \tag{15.17}$$

$$R_{so} = \frac{\rho l_o}{\delta w\left(1 - e^{-\frac{t}{\delta}}\right)} \tag{15.18}$$

$$R_{si} = \frac{\rho l_i}{\delta w\left(1 - e^{-\frac{t}{\delta}}\right)} \tag{15.19}$$

$$C_{ovo} = \frac{\varepsilon_{ox}}{t_{ox,\,t-b}} \cdot (n_o - 1)w^2 \tag{15.20}$$

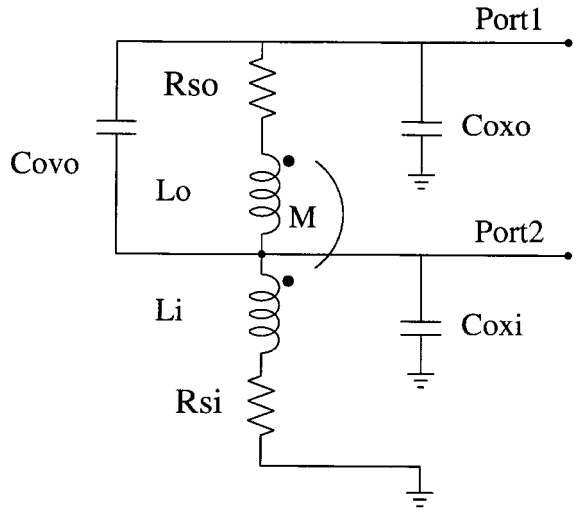


FIGURE 15.23 Tapped transformer model.

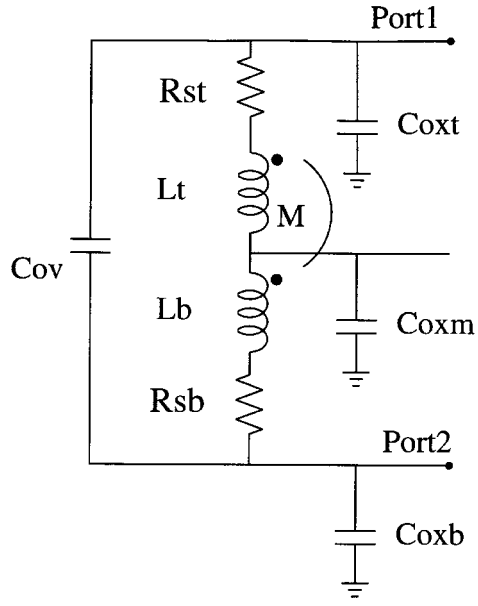


FIGURE 15.24 Stacked transformer model.

$$C_{oxo} = \frac{\varepsilon_{ox}}{2t_{ox}} \cdot l_o w \tag{15.21}$$

$$C_{oxi} = \frac{\varepsilon_{ox}}{2t_{ox}} \cdot (l_o + l_i) w \tag{15.22}$$

where ρ is the DC metal resistivity; δ is the skin depth; $t_{ox,t-b}$ is the oxide thickness from top level metal to bottom metal; n is the number of turns; *OD*, *AD*, and *ID* are the outer, average, and inner

diameters, respectively; l is the length of the spiral; w is the turn width; t is the metal thickness; and A is the area.

Expressions for the stacked transformer model are as follows (subscript t refers to the top spiral and b to the bottom spiral):

$$L_t = \frac{9.375\mu_0 n^2 AD^2}{11OD_T - 7AD_T} \tag{15.23}$$

$$L_b = L_t \tag{15.24}$$

$$k = 0.9 - \frac{d_s}{AD} \tag{15.25}$$

$$M = k\sqrt{L_t L_b} \tag{15.26}$$

$$R_{st} = \frac{\rho_t l}{\delta_t w\left(1 - e^{-\frac{t_t}{\delta_t}}\right)} \tag{15.27}$$

$$R_{sb} = \frac{\rho_b l}{\delta_b w\left(1 - e^{-\frac{t_b}{\delta_b}}\right)} \tag{15.28}$$

$$C_{ov} = \frac{\varepsilon_{ox}}{2t_{ox,\,t-b}} \cdot l \cdot w \cdot \frac{A_{ov}}{A} \tag{15.29}$$

$$C_{oxt} = \frac{\varepsilon_{ox}}{2t_{oxt}} \cdot l \cdot w \cdot \frac{A - A_{ov}}{A} \tag{15.30}$$

$$C_{oxb} = \frac{\varepsilon_{ox}}{2t_{ox}} \cdot l \cdot w \tag{15.31}$$

$$C_{oxm} = C_{oxt} + C_{oxb} \tag{15.32}$$

where t_{oxt} is the oxide thickness from top metal to the substrate; t_{oxb} is the oxide thickness from bottom metal to substrate; k is the coupling coefficient; A_{ov} is the overlap area of the two spirals; and d_s is the center-to-center spiral distance.

The expressions for the series resistances (R_{so}, R_{si}, R_{st}, and R_{sb}), the port-substrate capacitances (C_{oxo}, C_{oxi}, C_{oxt}, C_{oxb}, and C_{oxm}) and the crossover capacitances (C_{ovo}, C_{ovi}, and C_{ov}) are taken from Ref. 8. Note that the model accounts for the increase in series resistance with frequency due to skin effect. Patterned ground shields (PGS) are placed beneath the transformers to isolate them from resistive and capacitive coupling to the substrate.[23] As a result, the substrate parasitics can be neglected.

The inductance expressions in the foregoing are based on the modified Wheeler formula discussed earlier.[24] This formula does not take into account the variation in inductance due to conductor thickness and frequency. However, in practical inductor and transformer realizations, the thickness is small compared to the lateral dimensions of the coil and has only a small impact on the inductance. For typical conductor thickness variations (0.5 to 2.0 μm), the change in inductance is within a few percent for practical inductor geometries. The inductance also changes with frequency due to changes in current distribution within the conductor. However, over the useful frequency range of a spiral, this variation is negligible.[23] When compared to field solver simulations, the inductance expression exhibits a maximum error of 8% over a broad design space (outer diameter OD varying from 100 to 480 μm, L varying from 0.5 to 100 nH, w varying from 2 μm to $0.3OD$, s varying from 2 μm to w, and inner diameter ID varying from 0.2 to $0.8OD$).

For the tapped transformer, the mutual inductance is determined by first calculating the inductance of the whole spiral (L_T), the inductance of the outer spiral (L_o), the inductance of the inner spiral (L_i), and then using the expression $M = (L_T - L_o - L_i)/2$. For the stacked transformer, the spirals have identical lateral geometries and therefore identical inductances. In this case, the mutual inductance is determined by first calculating the inductance of one spiral (L_T), the coupling coefficient (k) and then using the expression $M = kL_T$. In this last case the coupling coefficient is given by $k = 0.9 - d_s/(AD)$ for $d_s < 0.7AD$, where d_s is the center-to-center spiral distance and AD is the average diameter of the spirals. As d_s increases beyond $0.7AD$, the mutual coupling coefficient becomes harder to model. Eventually, k crosses zero and reaches a minimum value of approximately –0.1 at $d_s = AD$. As d_s increases further, k asymptotically approaches zero. At $d_s = 2AD$, $k = -0.02$, indicating that the magnetic coupling between closely spaced spirals is negligible.

The self-inductances, series resistances, and mutual inductances are independent of whether a transformer is used as a three- or four-terminal device. The only elements that require recomputation are the port-to-port and port-to-substrate capacitances. This situation is analogous to that of a spiral inductor being used as a single- or dual-terminal device.

As with the inductance formulas, the transformer models obviate the need for full field solutions in all but very rare instances, allowing rapid design and optimization.

References

1. Samavati, H. et al., "Fractal capacitors," *1998 IEEE ISSCC Dig. of Tech. Papers*, Feb. 1998.
2. Akcasu, O. E., "High capacitance structures in a semiconductor device," U.S. Patent 5 208 725, May 1993.
3. Bohr, M., "Interconnect scaling — The real limiter to high performance VLSI," *Intl. Electron Devices Meeting Tech. Digest*, pp. 241-244, 1995.
4. Bohr, M. et al., "A high performance 0.25 μm logic technology optimized for 1.8V operation," *Intl. Electron Devices Meeting Tech. Digest*, pp. 847-850, 1996.
5. Venkatesan, S. et al., "A high performance 1.8V, 0.20 μm CMOS technology with copper metallization," *Intl. Electron Devices Meeting Tech. Digest*, pp. 769-772, 1997.
6. Mandelbrot, B. B., *The Fractal Geometry of Natur*e, W. H. Freeman, New York, 1983.
7. Pettenpaul, E. et al., "Models of lumped elements on GaAs up to 18 GHz," *IEEE Transactions on Microwave Theory and Techniques*, vol. 36, no. 2, pp. 294-304, Feb. 1988.
8. Yue, C. P., Ryu, C., Lau, J., Lee, T. H., and Wong, S. S., "A physical model for planar spiral inductors on silicon," *International Electron Devices Meeting Technical Digest*, pp. 155-158, Dec. 1996.

9. Pfost, M., Rein, H.-M., and Holzwarth, T., "Modeling substrate effects in the design of high speed Si-bipolar IC's," *IEEE J. Solid-State Circuits*, vol. 31, no. 10, pp. 1493-1501, Oct. 1996.
10. Ashby, K. B., Koullias, I. A., Finley, W. C., Bastek, J. J., and Moinian, S., "High Q inductors for wireless applications in a complementary silicon bipolar process," *IEEE J. Solid-State Circuits*, vol. 31, no. 1, pp. 4-9, Jan. 1996.
11. Chang, J. Y.-C., Abidi, A. A., and Gaitan, M., "Large suspended inductors on silicon and their use in a 2-μm CMOS RF amplifier," *IEEE Electron Device Letters*, vol. 14, no. 5, pp. 246-248, May 1993.
12. Greenhouse, H. M., "Design of planar rectangular microelectronic inductors," *IEEE Transactions on Parts, Hybrids, and Packing*, vol. PHP-10, no. 2, pp. 101-109, June 1974.
13. Wiemer, L. and Jansen, R. H.,"Determination of coupling capacitance of underpasses, air bridges and crossings in MICs and MMICs," *Electronics Letters*, vol. 23, no. 7, pp. 344-346, Mar. 1987.
14. Ho, I. T. and Mullick, S. K., "Analysis of transmission lines on integrated-circuit chips," *IEEE J. Solid-State Circuits*, vol. SC-2, no. 4, pp. 201-208, Dec. 1967.
15. Hasegawa, H., Furukawa, M., and Yanai, H., "Properties of microstrip line on Si-SiO_2 system," *IEEE Transactions on Microwave Theory and Techniques*, vol. MTT-19, no. 11, pp. 869-881, Nov. 1971.
16. Wheeler, H. A., "Simple inductance formulas for radio coils," *Proc. of the IRE*, vol. 16, no. 10, pp. 1398-1400, October 1928.
17. Rosa, E. B., "Calculation of the self-inductances of single-layer coils," *Bull. Bureau of Standards*, vol. 2, no. 2, pp. 161-187, 1906.
18. Maxwell, J. C., *A Treatise on Electricity and Magnetism*, Dover, 3rd ed., 1967.
19. Mohan, S. S., "Formulas for planar spiral inductances," *Tech. Rep., IC Laboratory*, Stanford University, Aug. 1998, http://www-smirc.stanford.edu.
20. Boyd, S. and Vandenberghe, L., "Introduction to convex optimization with engineering applications," Course Notes, 1997, http://www-leland.stanford.edu/class/ee364/.
21. Hershenson, M., Boyd, S. P., and Lee, T. H., "GPCAD: A tool for CMOS op-amp synthesis," in *Digest of Technical Papers, IEEE International Conference on Computer-Aided Design*, Nov. 1998.
22. Lee, T. H., *The Design of CMOS Radio-Frequency Integrated Circuits*, Cambridge University Press, Cambridge, 1998.
23. Yue, C. P. et al., "On-chip spiral inductors with patterned ground shields for Si-based RF ICs," *IEEE J. Solid-State Circuits*, vol. 33, pp. 743-752, May 1998.
24. Wheeler, H. A., "Simple inductance formulas for radio coils," *Proc. of the IRE*, vol. 16, no. 10, pp. 1398-1400, Oct. 1928.

Rollins, J.G. "Analog Circuit Simulation"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

16

Analog Circuit Simulation

J. Gregory Rollins
Antrim Design Systems

16.1 Introduction

Analog circuit simulation usually means simulation analog circuits or very detailed simulation of digital circuits. The most widely known and used circuit simulation program is SPICE (simulation program with integrated circuit emphasis) of which it is estimated that there are over 100,000 copies in use. SPICE was first written at the University of California at Berkeley in 1975, and was based on the combined work of many reasearchers over a number of years. Research in the area of circuit simulation continues at many universities and industrial sites. Commercial versions of SPICE or related programs are available on a wide variety of computing platforms, from small personal computers to large mainframes. A list of some commercial simulator vendors can be found in the Appendix. The focus of this chapter is the simulators and the theory behind them. Examples are also given to illustrate their use.

16.2 Purpose of Simulation

Computer-aided simulation is a powerful aid during the design or analysis of VLSI circuits. Here, the main emphasis will be on analog circuits; however, the same simulation techniques may be applied to digital circuits, which are, after are, composed of analog circuits. The main limitation will be the size of these circuits because the techniques presented here provide a very detailed analysis of the circuit in question and, therefore, would be too costly in terms of computer resources to analyze a large digital system. However, some of the techniques used to analyze digital systems (like iterated timing analysis or relaxation methods) are closely related to the methods used in SPICE.

It is possible to simulate almost any type of circuit SPICE. The programs have built-in elements for resistors, capacitors, inductors, dependent and independent voltage and current sources, diodes, MOSFETs, JFETs, BJTs, transmission lines, and transformers. Commercial versions have libraries of standard

components which have all necessary parameters prefitted to typical specifications. These libraries include items such as discrete transistors, op-amps, phase-locked loops, voltage regulators, logic integrated circuits, and saturating transformer cores. Versions are also available which allow the inclusion of digital models (mixed mode simulation) or behavioral models which allow the easy modeling of mathematical equations and relations.

Computer-aided circuit simulation is now considered an essential step in the design of modern integrated circuits. Without simulation, the number of "trial runs" necessary to produce a working IC would greatly increase the cost of the IC and the critical time to market. Simulation provides other advantages, including:

- The ability to measure "inaccessible" voltages and currents which are buried inside a tiny chip or inside a single transistor.
- No loading problems are associated with placing a voltmeter or oscilloscope in the middle of the circuit, measuring difficult one-shot waveforms or probing a microscopic die.
- Mathematically ideal elements are available. Creating an ideal voltage or current source is trivial with a simulator, but impossible in the laboratory. In addition, all component values are exact and no parasitic elements exist.
- It is easy to change the values of components or the configuration of the circuit.

Unfortunately, computer-aided simulation has it own set of problems, including:

- Real circuits are distributed systems, not the "lumped element models" which are assumed by simulators. Real circuits, therefore, have resistive, capacitive, and inductive parasitic elements present in addition to the intended components. In high-speed circuits, these parasitic elements can be the dominant performance-limiting elements in the circuit, and they must be painstakingly modeled.
- Suitable predefined numerical models have not yet been developed for certain types of devices or electrical phenomena. The software user may be required, therefore, to create his or her own models out of other models which are available in the simulator. (An example is the solid-state thyristor, which may be created from an npn and pnp bipolar transistor).
- The numerical methods used may place constraints on the form of the model equations used. In addition, convergence difficulties can arise, making the simulators difficult to use.
- There are small errors associated with the solution of the equations and other errors in fitting the non-linear models to the trsnsistors which make up the circuit.

16.3 Netlists

Before simulating, a circuit must be coded into a netlist. Figure 16.1 shows the circuit for a simple differential pair. Circuit nodes are formed wherever two or more elements meet. This particular circuit has seven nodes, which are numbered zero to six. The ground or datum node is traditionally numbered as zero. The circuit elements (or branches) connect the nodes.

The netlist provides a description of the topography of a circuit and is simply a list of the branches (or elements) that make up the circuit. Typically, the elements may be entered in any order and each has a unique name, a list of nodes, and either a value or model identifier. For the differential amplifier of Fig. 16.1, the netlist is shown in Fig. 16.2.

The first line gives the title of the circuit (and is required in many simulators). The next three lines define the three voltage sources. The letter V at the beginning tells SPICE that this is a voltage source element. The list of nodes (two in this case) is next followed by the value in volts. The syntax for the resistor is similar to that of the voltage source; the starting letter R in the names of the resistors tells SPICE that these are resistors. SPICE also understands that the abbreviation "k" after a value means 1000. For the two transistors Q1 and Q2, the starting letter Q indicates a bipolar transistor. Q1 and Q2 each

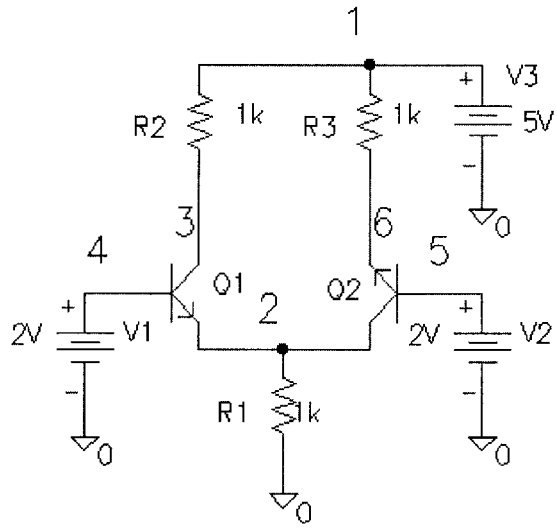


FIGURE 16.1 Circuit for differential pair.

```
Differential pair circut
V1 4 0 2V
V2 5 0 2V
V3 1 0 5V
R1 2 0 1k
R2 3 1 1K
R3 6 1 1K
Q1 3 4 2 m2n2222
Q2 6 5 2 mq2n2222
.model m2n2222 NPN IS=1e-12 BF=100 BR=5 TF=100pS
```

FIGURE 16.2 Netlist for differential pair.

have three nodes and in SPICE, the convention for their ordering is collector, base, emitter. So, for Q1, the collector is connected to node 3, the base to node 4, and the emitter to node 2. The final entry "m2n2222" is a reference to the model for the bipolar transistor (note that both Q1 and Q2 reference the same model). The ".model" statement at the end of the listing defines this model. The model type is npn (for an npn bipolar junction transistor), and a list of "parameter = value" entries follow. These entries define the numerical values of constants in the mathematical models which are used for the bipolar transistor. (Models will be discused in more detail later on.) Most commercial circuit simulation packages come with "schematic capture" software that allows the user to draw the circuit by placing and connecting the elements with the mouse

16.4 Formulation of the Circuit Equations

In SPICE, the circuits are represented by a system of ordinary differential equations. These equations are then solved using several different numerical techniques. The equations are constructed using Kirchoff's voltage and current laws. The first system of equations pertains to the currents flowing into each node. One equation is written for each node in the circuit (except for the ground node), so the following equation is really a system of N equations for the N nodes in the circuit. The subscript *i* denotes the node index.

$$0 = F_i(V) = G_i(V) + \frac{\partial Q_i(V)}{\partial t} + W_i(t) \tag{16.1}$$

V is an N-dimensional vector that represents the voltages at the nodes. Q is another vector which represents the electrical charge (in Coulombs) at each node. The term W represents any independent current sources that may be attached to the nodes and has units of amperes. The function $G(V)$ represents the currents that flow into the nodes as a result of the voltages V. If the equations are formulated properly, a system of N equations in N unknowns results.

For example, for the circuit of Fig. 16.3 which has two nodes, we need to write two equations. At Node 1:

$$0 = (V_1 - V_2)/R_1 + \frac{d(C_1 V_1)}{dt} + I_1 \tag{16.2}$$

We can identify $G(V)$ as $(V_1 - V_2)/R$, the term $Q(V)$ is $C_1 V_1$ and $W(t)$ is simply I_1. Likewise at Node 2:

$$0 = (V_2 - V_1)/R_1 + V_2/R_2 + gmV_1 \tag{16.3}$$

In this example, G and Q are simple linear terms; however, in general, they can be non-linear functions of the voltage vector V.

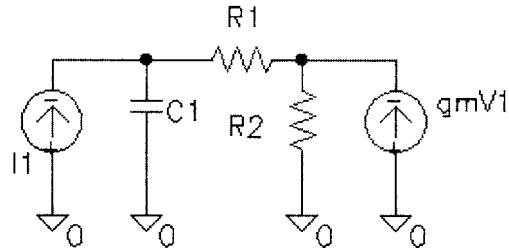


FIGURE 16.3 Example circuit for nodal analysis.

16.5 Modified Nodal Analysis

Normal nodal analysis that uses only Kirchoff's current law, cannot be used to represent ideal voltage sources or inductors. This is so because the branch current in these elements cannot be expressed as a function of the branch voltage. To resolve this problem, KVL is used to write a loop equation around each inductor or voltage source. Consider Fig. 16.4 for an example of this procedure. The unknowns to be solved for are the voltage V_1 at Node 1, V_2 the voltage at Node 2, V_3 the voltage at Node 3, the current flowing through voltage source V_1 which we shall call I_x and the current flowing in the inductor $L1$ which we shall call I_1 The system of equations is:

$$\begin{aligned} 0 &= V_1/R_1 + I_x \\ 0 &= V_2/R_2 - I_z + I_l \\ 0 &= V_3/R_3 - I_l \\ 0 &= V_1 - V_x + V_2 \\ 0 &= V_2 + \frac{d(L_1 I_1)}{dt} - V_3 \end{aligned} \tag{16.4}$$

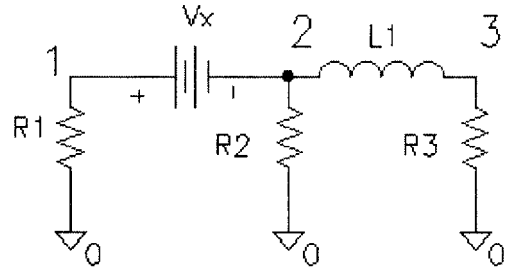


FIGURE 16.4 Circuit for modified nodal analysis.

The use of modified nodal analysis does have the disadvantage of requiring that an additional equation be included for each inductor or voltage source, but has the advantage that ideal voltage sources can be used. The total number of equations to be solved is therefore the number of nodes plus the number of voltages sources and inductors.

16.6 Active Device Models

VLSI circuits contain active devices like transistors or diodes which act as amplifiers. These devices are normally described by a complicated set of non-linear equations. We shall consider a simple model for the bipolar transistor — the Ebers-Moll model. This model is one of the first developed, and while it is too simple for practical application, it is useful for discussion.

A schematic of the Ebers-Moll model is shown in Fig. 16.5. The model contains three non-linear voltage-dependent current sources I_c, I_{bf}, and I_{br} and two non-linear capacitances C_{be} and C_{bc}. The current flowing in the three current sources are given by the following equations:

$$I_c = I_s(\exp(V_{be}/V_t)) - \exp(V_{ce}/V_t) \tag{16.5}$$

$$I_{bf} = \frac{I_s}{B_f}(\exp(V_{be}/V_t) - 1) \tag{16.6}$$

$$I_{br} = \frac{I_s}{B_r}(\exp(V_{bc}/V_t) - 1) \tag{16.7}$$

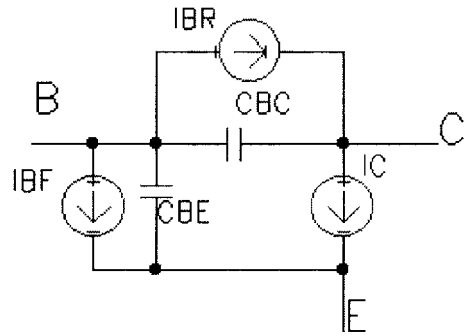


FIGURE 16.5 The Ebers-Moll model for the bipolar transistor.

The voltages V_{be} and V_{bc} are the voltages between base and emitter and the base and collector, respectively. I_s, B_f and B_r are three user-defined parameters which govern the DC operation of the BJT. V_t is the "thermal voltage" or kT/q, which has the numerical value of approximately 0.26 volts at room temperature. Observe that in the normal forward active mode, where $V_{be} > 0$ and $V_{ce} < 0$, I_{br} and the second term in I_c vanish and the current gain of the BJT, which is defined as I_c/I_b becomes numerically equal to B_f. Likewise, in the reverse mode where $V_{ce} > 0$ and $V_{be} < 0$, the reverse gain (I_e/I_b) is equal to B_r.

The two capacitances in Fig. 16.5 contribute charge to the emitter, base, and collector, and this charge is given by the following equations:

$$Q_{be} = \tau_f I_s(\exp V_{be}/V_t - 1) + C_{je}\int_0^{V_{be}} (1 - V/V_{je})^{-m_e} \tag{16.8}$$

$$Q_{bc} = \tau_r I_s(\exp V_{bc}/V_t - 1) + C_{jc}\int_0^{V_{bc}} (1 - V/V_{jc})^{-m_c} \tag{16.9}$$

Q_{be} contributes positive charge to the base and negative charge to the emitter. Q_{bc} contributes positive charge to the base and negative charge to the collector. The first term in each charge expression is due to charge injected into the base from the emitter for Q_{be} and from the collector into the base for Q_{bc}. Observe that the exponential terms in the charge terms are identical to the term in I_c. This is so because the injected charge is proportional to the current flowing into the transistor. The terms τ_f and τ_r are the forward and reverse transit times, respectively, and correspond to the amount of time it takes the electrons (or holes) to cross the base. The second term in the charge expression (the term with the integral) corresponds to the charge in the depletion region of the base–emitter junction for Q_{be} and in the base–collector junction for Q_{bc}. Recall that the depletion width in a pn junction is a function of the applied voltage. The terms V_{je} and V_{jc} are the "built-in" potentials with units of volts for the base–emitter and base–collector junctions. The terms m_c and m_e are the grading coefficients for the two junctions and are related to how rapidly the material changes from n-type to p-type across the junction.

This "simple" model has eleven constants I_s, B_f, B_r, C_{je}, C_{jc}, M_e, M_c, V_{je}, V_{jc}, T_f, and T_r which must be specified by the user. Typically, these constants would be extracted from measured I-V and C-V data taken from real transistors using a fitting or optimization procedure (typically a non-linear least-squares fitting method is needed). The Ebers-Moll model has a number of shortcomings which are addressed in newer models like Gummel-Poon, Mextram, and VBIC. The Gummel-Poon model has over 40 parameters that must be adjusted to get a good fit to data in all regions of operation.

Models for MOS devices are even more complicated than the bipolar models. Modeling the MOSFET is more difficult than the bipolar transistor because it is often necessary to use a different equation for each of the four regions of operation (off, subthreshold, linear, saturation) and the drain current and capacitance are functions of three voltages (V_{ds}, V_{bs}, and V_{gs}) rather than just two (V_{be} and V_{ce}) as in the case of the BJT. If a Newton-Raphson solver is to be used, the I-V characteristics and capacitances must be continuous and it is best if their first derivatives are continuous as well. Furthermore, MOS models contain the width (W) and length (L) of the MOSFET channel as parameters; and for the best utility the model should remain accurate for many values of W and L. This property is referred to as "scalability."

Over the years, literally hundreds of different MOS models have been developed. However, for modern VLSI devices, only three or four are commonly used today. These are the SPICE Level-3 MOS model, the HSPICE Level-28 model (which is a proprietary model developed by Meta Software), the public domain BSIM3 model develeped at UC Berkeley, and MOS9 developed at Phillips. These models are supported by many of the "silicon foundries," that is, parameters for the models are provided to chip designers by the foundries. BSIM3 has been observed to provide a good fit to measured data and its I-V curves to be smooth and continuous (thereby resulting in good simulator convergence). The

main drawback of BSIM3 is that it has over 100 parameters which are related in intricate ways, making extraction of the parameter set a difficult process.

A process known as "binning" is used to provide greater accuracy. When binning is used, a different set of model parameters is used for each range of the channel length and width (L and W). An example of this is shown in Fig. 16.6. For a given type of MOSFET, 12 complete sets of model parameters are extracted and each is valid for a given range. For example, in Fig. 16.6, the set represented by the number "11" would only be valid for channel lengths between 0.8 and 2.0 microns and for channel widths between 0.5 and 0.8 microns. Thus, for a typical BSIM3 model with about 60 parameters, $12 \times 60 = 720$ parameters would need to be extracted in all and this just for one type of device.

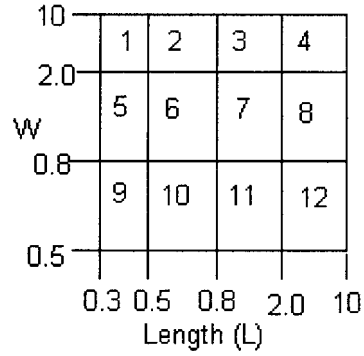


FIGURE 16.6 Binning of MOS parameters.

Many commercial simulators contain other types of models besides the traditional R, L, C, MOS, and BJT devices. Some simulators contain "behavioral" models which are useful for systems design or integration tasks; examples of these are integrators, multipliers, summation, and LaPlace operator blocks. Some simulators are provided with libraries of prefitted models for commercially available operational amplifiers, logic chips, and discrete devices. Some programs allow "mixed-mode" simulation, which is a combination of logic simulation (which normally allows only a few discrete voltage states) and analog circuit simulation.

16.7 Types of Analysis

For analog circuits, there are three commonly used methods of analysis, these being DC, AC, and transient analysis. DC analysis is used to examine the steady-state operation of a circuit; that is, what the circuit voltages and currents would be if all inputs were held constant for an infinite time. AC analysis (or sinusoidal steady state) examines circuit performance in the frequency domain using phasor analysis. Transient analysis is performed in the time domain and is the most powerful and computationally intensive of the three. For special applications, other methods of analysis are available such as the Harminic-Balance method, which is useful for detailed analysis of non-linear effects in circuits excited by purely perodic signals (like mixers and RF amplifiers).

DC (Steady-State) Analysis

DC analysis calculates the steady-state response of a circuit (with all inductors shorted and capacitors removed). DC analysis is used to determine the operating point (Q-point) of a circuit, power consumption, regulation and output voltage of power supplies, transfer functions, noise margin and fanout in logic gates, and many other types of analysis. In addition, a DC solution must be calculated to find the starting point for AC and transient analysis.

To calculate the DC solution, we need to solve Kirchoff's equations formulated earlier. Unfortunately, since the circuit elements will be non-linear in most cases, a system of transcendental equations will normally result and it is impossible to solve this system analytically. The method which has met with the most success is Newton's method or one of its derivatives.

Newton's Method

Newton's method is actually quite simple. We need is to solve the system of equations $F(X) = 0$ for X, where both F and X are vectors of dimension N. (F is the system of equations from modified nodal analysis, and X is the vector of voltages and current that we are solving for.) Newton's method states that given an initial guess for X^i, we can obtain a better guess X^{i+1} from the equation:

$$X^{i+1} = X^i - [J(X^i)]^{-1} F(X^i) \tag{16.10}$$

Note that all terms on the right side of the equation are functions only of the vector X^i. The term $J(X)$ is a N × N square matrix of partial derivatives of F, called the Jacobian. Each term in J is given by:

$$J_{i,j} = \frac{\partial F_i(X)}{\partial X_j} \tag{16.11}$$

We assemble the Jacobian matrix for the circuit at the same time that we assemble the circuit equations. Analytic derivatives are used in most simulators.

The –1 in Eq. 16.10 indicates that we need to invert the Jacobian matrix before multiplying by the vector F. Of course, we do not need to actually invert J to solve the problem; we only need to solve the linear problem $F = YJ$ for the vector Y and then calculate $X^{i+1} = X^i - Y$. A direct method such as the LU decomposition is usually employed to solve the linear system.

For the small circuit of Fig. 16.3, analyzed in steady state (without the capacitor), the Jacobian entries are:

$$\begin{aligned} J_{1,1} &= 1/R_1 & J_{1,2} &= -1/R_1 \\ J_{2,1} &= 1/R_1 + gm & J_{2,2} &= 1/R_1 + 1/R_2 \end{aligned} \tag{16.12}$$

For a passive circuit (i.e., a circuit without gain), the Jacobian will be symmetric and for any row, the diagonal entry will be greater than the sum of all the other entries.

Newton's method converges quadratically, provided that the initial guess X^i is sufficiently close to the true solution. Quadratically implies that if the distance between X^i and the true solution is d, then the distance between X^{i+1} and the true solution will be d^2. Of course, we are assuming that d is small to start with. Still, programs like SPICE may require 50 or more iterations to achieve convergence. The reason for this is that, often times, the initial guess is poor and quadratic convergence is not obtained until the last few iterations. There are additional complications like the fact that the model equations can become invalid for certain voltages. For example, the BJT model will "explode" if a junction is forward-biased by more than 1 V or so since : $\exp(1/Vt) = 5e16$. Special limiting or damping methods must be used to keep the voltages and currents to within reasonable limits.

Example Simulation

Most circuit simulators allow the user to ramp one or more voltage sources and plot the voltage at any node or the current in certain branches. Returning to the differential pair of Fig.16.1, we can perform a DC analysis by simply adding a .DC statement (see Fig. 16.7). A plot of the differential output voltage (between the two collectors) and the voltage at the two emitters is shown in Fig. 16.8. Observe that the output voltage is zero when the differential pair is "balanced" with 2.0 V on both inputs. The output saturates at both high and low values for V_1, illustrating the non-linear nature of the analysis. This

```
Stead state analysis of differential pair.
V1 4 0 2V
V2 5 0 2V
V3 1 0 5V
R1 2 0 1k
R2 3 1 1K
R3 6 1 1K
Q1 3 4 2 m2n2222
Q2 6 5 2 m2n2222
.model m2n2222 NPN IS=1e-12 BF=100 BR=5 TF=100pS
.dc V1  1.0  3.0  0.01
```

FIGURE 16.7 Input file for DC sweep of V_1.

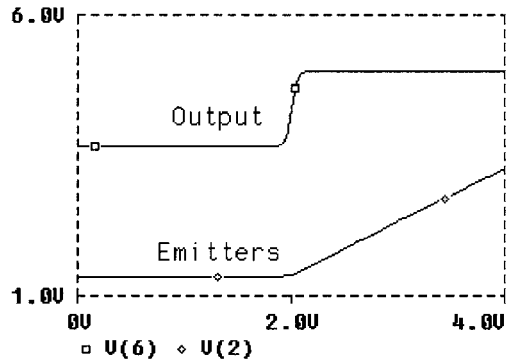


FIGURE 16.8 Output from DC analysis.

simulation was run using the PSPICE package from MicroSim Corporation. The simulation run is a few seconds on a 486 type PC.

AC Analysis

AC analysis is performed in the frequency domain under the assumption that all signals are represented as a DC component V_{dc} plus a small sinusoidal component V_{ac}.

$$V = V_{dc} + V_{ac}\exp(j\omega t) \tag{16.13}$$

Here, $j = \sqrt{-1}$, ω is the radial frequency ($2\pi f$), and V_{ac} is a complex number. Expanding (1) about the DC bias point V_{dc} (also referred to as the Q point), we obtain:

$$F(V) = F(V_{dc}) + W_{dc} + W_{ac} + \frac{\partial G(V_{dc})}{\partial V_{dc}} V_{ac} + \frac{\partial}{\partial t}\left(\frac{\partial Q(V_{dc})}{\partial V_{dc}}\right) V_{ac} + \alpha V_{ac}^2 \Lambda \tag{16.14}$$

The series has an infinite number of terms; however, we assume that if V_{ac} is sufficiently small, all terms above first order can be neglected. The first two terms on the right-hand side are the DC solution and, when taken together, yield zero. The third term W_{ac} is the vector of independent AC current sources which drive the circuit. The partial derivative in the fourth term is the Jacobian element, and the derivative of Q in parentheses is the capacitance at the node. When we substitute the exponential into Eq. 16.14, each term will have an exponential term that can be canceled. The result of all these simplifications is the familiar result:

$$0 = W_{ac} + JV_{ac} + j\omega CV_{ac} \tag{16.15}$$

This equation contains only linear terms which are equal to the partial derivatives of the original problem evaluated at the Q point. Therefore, before we can solve the AC problem, we must calculate the DC bias point. Rearranging terms slightly, we obtain:

$$V_{ac} = -(J + j\omega C)^{-1} W_{ac} \tag{16.16}$$

The solution at a given frequency can be obtained from a single matrix inversion. The matrix, however, is complex but normally the complex terms share a sparsity pattern similar to the real terms. It is normally possible (in FORTRAN and C++) to create a suitable linear solver by taking the linear solver which is used to calculate the DC solution and substituting "complex" variables for "real" variables. Since there is no non-linear iteration, there are no convergence problems and AC analysis is straightforward and fool-proof.

The same type of analysis can be applied to the equations for modified nodal analysis. The unknowns will of course be currents and the driving sources voltage sources.

$$I_{ac} = -(J + j\omega L)^{-1} E_{ac} \tag{16.17}$$

The only things that must be remembered with AC analysis are:

1. The AC solution is sensitive to the Q point, so if an amplifier is biased near its saturated DC output level, the AC gain will be smaller than if the amplifier were biased near the center of its range.
2. This is a linear analysis and therefore "clipping" and slew rate effects are not modeled. For example, if a 1-V AC signal is applied to the input of a small signal amplifier with a gain of 100 and a power supply voltage of 5 V, AC analysis will predict an output voltage of 100 V. This is of course impossible since the output voltage cannot exceed the power supply voltage of 5 V. If you want to include these effects, use transient analysis.

AC Analysis Example

In the following example, we will analyze the differential pair using AC analysis to determine its frequency response. To perform this analysis in SPICE, we need only specify which sources are the AC driving sources (by adding the magnitude of the AC signal at the end) and specify the frequency range on the .AC statement (see Fig. 16.9). SPICE lets the user specify the range as linear or "decade," indicating that we desire a logarithmic frequency scale. The first number is the number of frequency points per decade. The second number is the starting frequency, and the third number is the ending frequency.

```
AC analysis of differential pair.
V1 4 0 2V   AC 1
V2 5 0 2V
V3 1 0 5V
R1 2 0 1k
R2 3 1 1K
R3 6 1 1K
Q1 3 4 2 m2n2222
Q2 6 5 2 mq2n2222
.model m2n2222 NPN IS=1e-12 BF=100 BR=5 TF=100pS
.AC DEC 10 1e3  1e9
```

FIGURE 16.9 Input file for AC analysis.

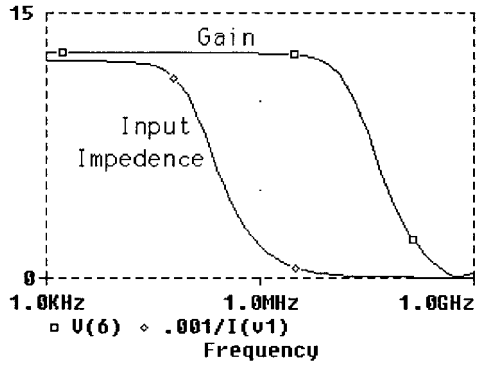


FIGURE 16.10 Gain and input impedance calculated by AC analysis.

Figure 16.10 shows the results of the analysis. The gain begins to roll off at about 30 MHz due to the parasitic capacitances within the transistor models. The input impedance (which is plotted in kΩ) begins to roll off at a much lower frequency. The reduction in input impedance is due to the increasing current that flows in the base-emitter capacitance as the current increases. SPICE does not have a method of calculating input impedance, so we have calculated it as $Z = V_{in}/I(V_{in})$, where $V_{in} = 1.0$, using the post-processing capability of PSPICE. This analysis took about 2 seconds on a 486 type PC.

Noise Analysis

Noise is a problem primarily in circuits that are designed for the amplification of small signals like the RF and IF amplifiers of a receiver. Noise is the result of random fluctuations in the currents which flow in the circuit and is generated in every circuit element. In circuit simulation, noise analysis, is an extension of AC analysis. During noise analysis, it is assumed that every circuit element contributes some small noise component either as a voltage V_n in series with the element or as a current I_n across the element. Since the noise sources are small in comparison to the DC signal levels, AC small signal analysis is an applicable analysis method.

Different models have been developed for the noise sources. In a resistor, thermal noise is the most important component. Thermal noise is due to the random motion of the electrons:

$$I_n^2 = \frac{4kT\Delta f}{R} \tag{16.18}$$

where T is the temperature, k is Boltzman's constant, and Δf is the bandwidth of the circuit. In a semiconductor diode, shot noise is important. Shot noise is related to the probability that an electron will surmount the semiconductor barrier energy and be transported across the junction:

$$I_n^2 = 2qI_d\Delta f \tag{17.19}$$

There are other types of noise that occur in diodes and transistors; examples are flicker and popcorn noise. Noise sources, in general, are frequency dependent.

Noise signals will be amplified or attenuated as they pass through different circuits. Normally, noise is referenced to some output point called the "summing node." This would normally be the output of the amplifier where we would actually measure the noise. We can call the gain between the summing node and the current flowing in an element j in the circuit $A_j(f)$. Here, f is the analysis frequency since the gain will normally be frequency dependent.

Noise signals are random and uncorrelated to each other so their magnitudes must be root-mean-squared summed rather than simply summed. Summing all noise sources in a circuit yields:

$$I_n(f) = \sqrt{\sum_j A_j^2(f) I_j^2(f)} \tag{16.20}$$

It is also common to reference noise back to the amplifier input and this is easily calculated by dividing the above expression by the amplifier gain. Specifying noise analysis in SPICE is simple. All the user needs to do is add a statement specifying the summing node and the input source. Spice then calculates the noise at each as a function of frequency

```
.noise v([6]) V1
```
(16.21)

See Fig. 16.11 for example output. Many circuit simulators will also list the noise contributions of each element as part of the output. This is particularly helpful in locating the source of noise problems.

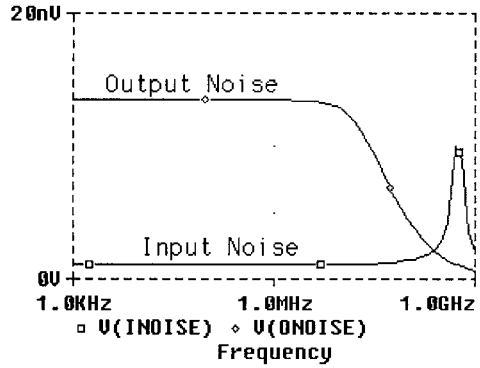


FIGURE 16.11 Noise referenced to output and input.

Transient Analysis

Transient analysis is the most powerful analysis capability because the transient response of a circuit is so difficult to calculate analytically. Transient analysis can be used for many types of analysis, such as switching speed, distortion, and checking the operation of circuits like logic gates, oscillators, phase-locked loops, or switching power supplies. Transient analysis is also the most CPU intensive and can require 100 or 1000 times the CPU time of DC or AC analysis.

Numerical Method

In transient analysis, time is discretized into intervals called *time steps*. Typically, the time steps are of unequal length, with the smallest steps being taken during intervals where the circuit voltages and currents are changing most rapidly. The following procedure is used to discretize the time-dependent terms in Eq. 16.1.

Time derivatives are replaced by difference operators, the simplest of which is the forward difference operator:

$$\frac{dQ(t_k)}{dt} = \frac{Q(t_{k+1}) - Q(t_k)}{h} \tag{16.22}$$

where h is the time step given by $h = t_{k+1} - t_k$. We can easily solve for the charge $Q(t_{k+1})$ at the next time point:

$$Q(t_{k+1}) = Q(t_k) - h(G_i(V(t_k)) + W_i(t_k)) \tag{16.23}$$

using only values from past time points. This means that it would be possible to solve the system simply by plugging in the updated values for V each time. This can be done without any matrix assembly or inversion and is very nice. (Note for simple linear capacitors, $V = Q/C$ at each node, so it is easy to get V back from Q.) However, this approach is undesirable for circuit simulation for two reasons. (1) The charge Q, which is a "state variable" of the system, is not a convenient choice since some nodes may not have capacitors (or inductors) attached, in which case they will not have Q values. (2) It turns out that forward (or explicit) time discretization methods like this one are unstable for "stiff" systems, and most circuit problems result in "stiff systems." The term "stiff system" refers to a system that has greatly varying time constants.

To overcome the stiffness problem, we must use implicit time discretization methods which, in essence, means that the G and W terms in the above equations must be evaluated at t_{k+1}. Since G is non-linear, we will need to use Newton's method once again.

The most popular implicit method is the trapezoidal method. The trapezoidal method has the advantage of only requiring information from one past time point and, furthermore, has the smallest error of any method requiring one past time point. The trapezoidal method states that if I is the current in a capacitor, then:

$$I(t_{k+1}) = \frac{dQ}{dt} = 2\frac{Q(V(t_{k+1})) - Q(V(t_k))}{h} - I(t_k) \tag{16.24}$$

Therefore, we need only substitute the above equation into Eq. (16.1) to solve the transient problem. Observe that we are solving for the voltages $V(t_{k+1})$, and all terms involving t_k are constant and will not be included in the Jacobian matrix. An equivalent electrical model for the capacitor is shown in Fig. 16.12. Therefore, the solution of the transient problem is in effect a series of DC solutions where the values of some of the elements depend on voltages from the previous time points.

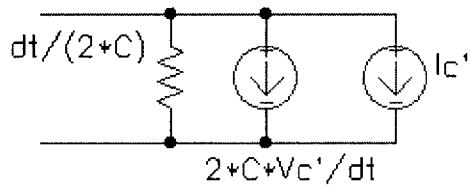


FIGURE 16.12 Electrical model for a capacitor; the two current sources are independent sources. The prime (') indicates values from a preceding time point.

All modern circuit simulators feature automatic time step control. This feature selects small time steps during intervals where changes are occurring rapidly and large time steps in intervals where there is little change. The most commonly used method of time step selection is based on the local truncation error (LTE) for each time step. For the trapezoidal rule, the LTE is given by:

$$\varepsilon = \frac{h^3}{12}\frac{d^3x}{dt^3}(\xi) \tag{16.25}$$

and represents the maximum error introduced by the trapezoidal method at each time step. If the error (ε) is larger than some preset value, the step size is reduced. If the error is smaller, then the step size is increased. In addition, most simulators select time points so that they coincide with the edges of pulse-type waveforms.

Transient Analysis Examples

As a simple example, we return to the differential pair and apply a sine wave differentially to the input. The amplitude (2 V p-p) is selected to drive the amplifier into saturation. In addition, we make the frequency (50 MHz) high enough to see phase shift effects. The output signal is therefore clipped due to the non-linearities and shifted in phase due to the capacitive elements in the transistor models (see Fig. 16.13). The first cycle shows extra distortion since it takes time for the "zero-state" response to die out. This simulation, using PSPICE, runs in about one second on a 486 type computer.

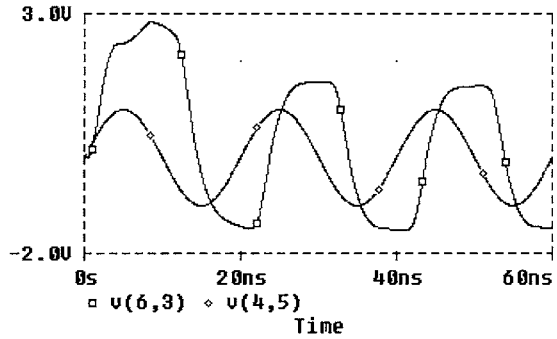


FIGURE 16.13 Transient response V(6,3) of differential amplifier to sinusoidal input at V(4,5).

16.8 Verilog-A

Verilog-A is a new language designed for simulation of analog circuits at various levels. Mathematical equations can be entered directly as well as normal SPICE-type circuit elements.

Groups of equations and elements can be combined into reusable "modules" that are similar to subcircuits. Special functions are also provided for converting analog signals into digital equivalents, and vice versa. Systems-type elements such as LaPlace operators, integrators, and differentators are also provided. This makes it possible to perform new types of modeling which were not possible in simulators like SPICE:

- Equations can be used to construct new models for electrical devices (for example, the Ebers-Moll model described earlier could be easily implemented).
- Behavioral models for complex circuits like op-amps, comparitors, phase detectors, etc. can be constructed. These models can capture the key behavior of a circuit and yet be simulated in a small fraction of the time it takes to simulate at the circuit level.
- Special interface elements make it possible to connect an analog block to a digital simulator, making mixed-mode simulation possible.Verilog-A is related to and compatible with the popular Verilog-D modeling language for digital circuits.

As an example, consider a phase-locked loop circuit which is designed as an 50X frequency multiplier. A block diagram for the PLL is shown in Fig. 16.14 and the Verilog-A input listing is shown in Figs. 16.15 and 16.16.

Simulation of this system at the circuit level is very time consuming due to the extreme difference in frequencies. The phase detector operates at a low frequency of 1.0 MHz, while the VCO operates at close to 50 MHz. However, we need to simulate enough complete cycles at the phase detector output to verify that the circuit correctly locks onto the reference signal.

The circuit is broken up into five blocks or modules: The "top module," VCO, divider, phase detector, and loop filter. The VCO has a simple linear dependence of frequency on the VCO input voltage and produces a sinusoidal output voltage. The VCO frequency is calculated by the simple expression freq = center + gain * ($V_{in} - V_{min}$). Center and gain are parameters which can be passed in when the VCO is

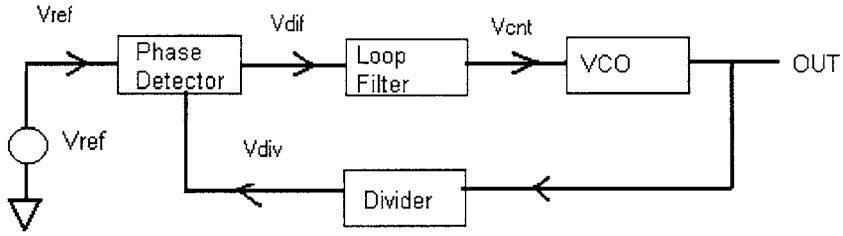


FIGURE 16.14 Block diagram of phase-locked loop.

created within the top module by the special syntax "`#(.gain(2e7), .center(3e7)`" in the top module. If the parameters are not specified when the module is created, then the default values specified within the module are used instead. The special V() operator is used to obtain the voltage at a node (in this case `V(in)` and `V(Vdd)`). The sinusoidal output is created using the SIN and IDT operators. SIN calculates the sin of its argument. I_{dt} calculates the integral of its argument with respect to time. The amplitude of the output is taken from the V_{dd} input, thus making it easy to integrate the VCO block with others. Given that V_{dd} = 5 volts, gain = 2e7 Hz/V center = 3e7 Hz, and in = 1.8, the final expression for the VCO output is:

$$V_{out} = 2.5 + 2.5\sin\left(3e7 + 2e7\int 2\pi(V_{in} - 1.8)dt\right) \tag{16.26}$$

```
`include "electrical.h"
`timescale 100ns / 10ns

module top;                          // Top level Module
        electrical Vdd, Vref, Vdif, Vcnt, Vout, Vdiv;
        pd #(.gain(15u), .rout (2e6), dir(-1)) p1(Vdif, Vdiv, Vref, Vdd);
        filter f1(Vdif, Vcnt);
        vco #(.gain(2e7), .center(3e7)) v1(Vout, Vcntm Vdd);
        divide #(.count(50)) d1(Vdiv, Vout, Vdd);
        analog begin
                V(Vdd) <+ 5.0;
                V(Vref) <+ 2.5+2.5*sin(6.238*1e6*$realtime);
        end
endmodule

module pd (out,vco,ref,vdd);        // Phase detector Module.
        inout out,vco,ref,vdd;
        electrical out,vco,ref,vdd;
        parameter real gain=15u, rout=5e6;     // gain  & output impedance
        parameter integer dir=1;   // 1,-1 pos or neg edge trigger
        integer state;
        analog begin
                @(cross((V(ref) - V(vdd)/2),dir)) state = state - 1;
                @(cross((V(vco) - V(vdd)/2),dir)) state = state + 1;
                if (state > 1) state = 1;
                if (state < -1) state = -1;
                if (state != 0)   I(out) <+ transition(state * gain , 0, 0, 0);
               I(out) <+ V(out)/rout;
        end
endmodule
```

FIGURE 16.15 Part one of Verilog-A listing for PLL.

```
module divide (out,in, vdd);      // Divider module....
        inout out,in;
        electrical out,in;
        parameter real count=4; // divide by this.
    integer n,state;
        analog begin
                @(cross((V(in) - V(vdd)/2.0),1))  n = n + 1;
                if (n >= count/2) begin
                        if (state == 0) state = 1;
                        else state = 0;
                        n = 0;
                end
                V(out) <+ transition(state*5 , 0, 0, 0);
        end
endmodule

module vco (vout, vin, vdd);        // VCO module
        inout vin, vout, vdd;
        electrical vin, vout, vdd;
        parameter real gain = 5e6,  center = 40e6,  vmin=1.8;
        real freq,vinp;
        analog begin
            vinp = V(vin);
            if (vinp < vm)  vinp = vmin;
            freq = center +  gain*(vinp-vmin);
            V(vout) <+ V(vdd)/2.0*sin(6.28318531*idt(freq,0)) + V(vdd)/2.0;
        end
endmodule

'language SPICE
.SUBCKT filter Vin Vout
Rf2 Vin Vout 100
Rfilter Vin VC2 200000
C1 VC2 0 58p
C2 Vin 0 5p
.ends
'endlanguage
```

FIGURE 16.16 Part two of Verilog-A PLL listing.

The phase detector functions as a charge pump which drives current into or out of the loop filter, depending on the phase difference between its two inputs. The `@cross(V1,dir)` function becomes true whenever signal V_1 crosses zero in the direction specified by `dir`. This either increments or decrements the variable STATE. The "transition" function is used to convert the STATE signal, which is essentially digital and changes abruptly, into a smoothly changing analog signal which can be applied to the rest of the circuit. The "<+" (or contribution) operator adds the current specified by the equation on the right to the node on the left. Therefore, the phase detector block forces current into the output node whenever the VCO signal leads the reference signal and forcing current out of the output node whenever the reference leads the vco signal. The phase detector also has an output resistance which is specified by parameter ROUT.

The loop filter is a simple SPICE subcircuit composed of two resistors and one capacitor. Of course, this subcircuit could contain other types of elements as well and can even contain other Verilog-A modules. The divider block simply counts zero crossings and, when the count reaches the preset divisor, the output of the divider is toggeled from 0 to 1, or vice versa. The transition function is used to ensure that a smooth, continuous analog output is generated by the divider.

This PLL was simulated using AMS from Antrim Design Systems. The results of the simulations are shown in Fig. 16.17. The top of Fig. 16.17 shows the output from the loop filter (V_{cnt}). After a few cycles, the PLL has locked onto the reference signal. The DC value of the loop filter output is approximately 2.8 V. Referring back to the VCO model, this gives an output frequency of 2e7*(2.8 – 1.8) + 3e7 = 50 MHz,

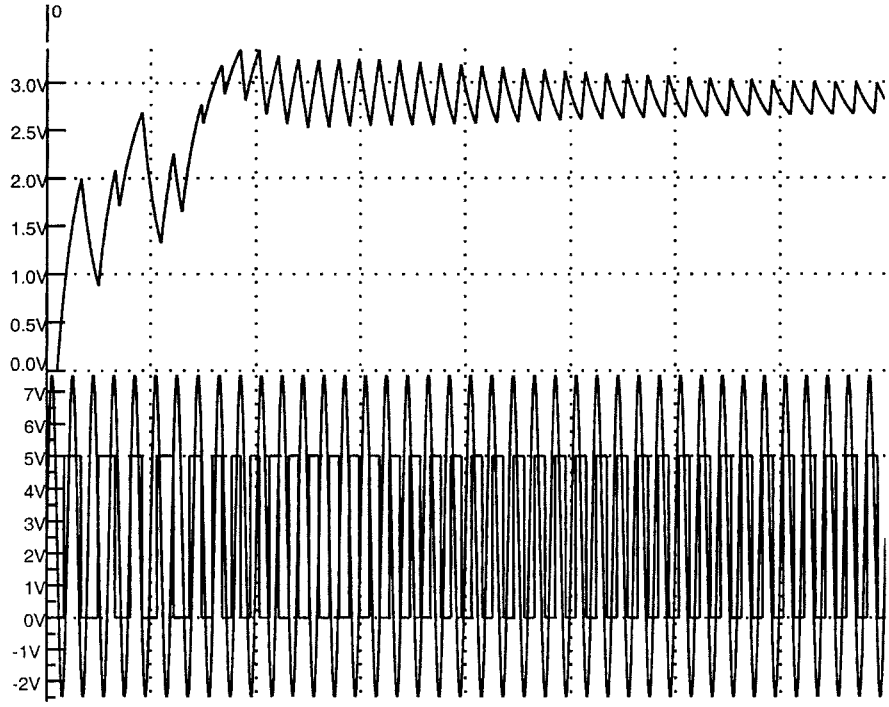


FIGURE 16.17 Loop filter output (top) V_{ref} and divider output (bottom) from PLL simulation.

which is as expected. The lower portion of Fig. 16.17 shows the divider output (V_{div}) and the reference signal (V_{ref}). It can be seen that the two signals are locked in phase. Figure 16.18 shows the VCO output and the divider output. As expected, the VCO frequency is 50 times the divider frequency.

The behavioral models used in this example are extremely simple ones. Typically, more complex models must be used to accurately simulate the operation of an actual PLL. A better model might include effects such as the non-linear dependence of the VCO frequency on the input voltage, the effects on signals introduced through power supply lines, delays in the divider and phase detector, and finite signal rise and fall times. These models can be built up from measurements, or transistor-level simulation of the underlying blocks (a process known as *characterization*). Of course, during the simulation, any of the behavioral blocks could be replaced by detailed transistor level models or complex Verilog-D digital models.

Another Verilog-A example is shown in Fig. 16.19. Here, the Ebers-Moll model developed earlier is implemented as a module. This module can then be used in a circuit in much the same way as the normal built-in models. Verilog-A takes care of calculating all the derivatives needed to form the Jacobian matrix. The "parameter" entries can be used in the same way as the parameters on a SPICE.MODEL statement. Observe the special "ddt" operator. This operator is used to take the time derivative of its argument. In this case, the time derivative of the charge (a current) is calculated and summed in with the other DC components. The "$limexp" operation is a special limited exponential operator designed to give better convergence when modeling pn junctions. Of course, this module could be expanded and additional features could be added.

16.9 Fast Simulation Methods

As circuits get larger, simulation times become larger. In addition, as integrated circuit feature sizes shrink, second-order effects become more important and many circuit designers would like to be able to simulate

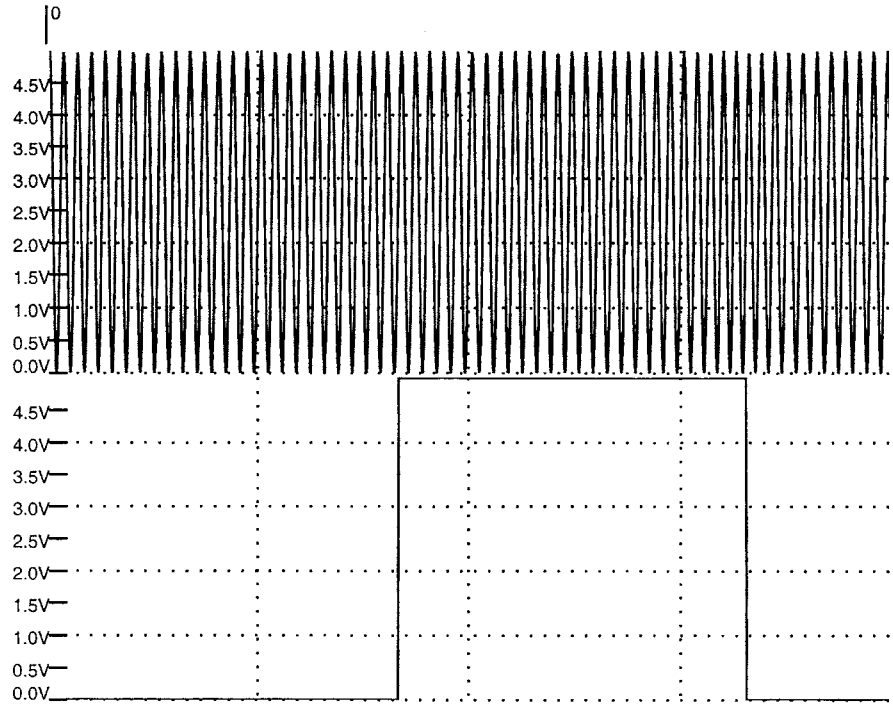


FIGURE 16.18 VCO output and divder output from PLL simulation.

large digital systems at the transistor level (requiring 10,000 to 100,000 nodes). Numerical studies in early versions of SPICE showed that the linear solution time could be reduced to 26% for relatively small circuits with careful coding. The remainder is used during the assembly of the matrix, primarily for model evaluation. The same studies found that the CPU time for the matrix solution was proportional to $n^{1.24}$, where n is the number of nodes. The matrix assembly time on the other hand should increase linearly with node count. Circuits have since grown much bigger, but the models (particularly for MOS devices) have also become more complicated.

Matrix assembly time can be reduced by a number of methods. One method is to simplify the models; however, accuracy will be lost as well. A better way is to precompute the charge and current characteristics for the complicated models and store them into tables. During simulation, the actual current and charges can be found from table lookup and interpolation, which can be done quickly and efficiently. However, there are some problems:

1. To assure convergence of Newton's method, both the charge and current functions and their derivatives must be continuous. This rules out most simple interpolation schemes and means that something like a cubic spline must be used.
2. The tables can become large. A MOS device has four terminals, which means that all tables will be functions of three independent variables. In addition, the MOSFET requires four separate tables (Id, Qg, Qd, Qb). If we are lucky, we can account for simple parameteric variations (like channel width) by a simple multiplying factor. However, if there are more complex dependencies as is the case with channel length, oxide thickness, temperature, or device type, we will need one complete set of tables for each device.

If the voltages applied to an element do not change from the past iteration to the present iteration, then there is no need to recompute the element currents, charges, and their derivatives. This method is referred to as taking advantage of latency and can result in large CPU time savings in logic circuits, particularly if coupled with a method which only refractors part of the Jacobian matrix. The tricky part

```
module mybjt (C,B,E);
        inout C,B,E;
        electrical C,B,E;
        parameter real is=1e-16, bf=100, br=10, tf=1n, tr=10n, cje=1p cjc=1p,
                vje=0.75, vjc=0.75, mje=0.33 mjc=0.33;
        real ibc, ibe, qbc, qbe, vbe, vbc, cc, cb, x, y;
        analog begin
                vbe=V(B,E);
                vbc=V(B,C);
                ibe = is*($limexp(vbe/$vt)-1);
                ibc = is*($limexp(vbc/$vt)-1);
                cb = ibe/bf + ibc/br;
                cc = ibe – ibc – ibc/br
                if(vbe < 0) begin
                        x=1-vbe/vje;
                        y=exp(-mje*ln(x));
                        qbe = vje*cje*(1-x*y)/(1.0-mje)+tf*ibe;
                end
                else qbe = cje*(vbe+0.5*mje*vbe*vbe/vje) + tf*ibe;
                if(vbc < 0) begin
                        x=1-vbc/vjc;
                        y=exp(-mjc*ln(x));
                        qbc = vjc*cjc*(1-x*y)/(1.0-mjc) + tr*ibc;
                end
                else qbc = cjc*(vbc+0.5*mjc*vbc*vbc/vjc) + tr*ibc;
                ibe = ddt(qbe);
                ibc = ddt(qbc);
                I(C,E) <+ cc - ibc;
                I(B,E) <+ cb + ibe + ibc;
        end
endmodule
```

FIGURE 16.19 Verilog-A implimentation of the Ebers-Moll model.

is knowing when the changes in voltage can be ignored. Consider, for example, the input to a high-gain op-amp, here ignoring a microvolt change at the input could result in a large error at the output. Use of sophisticated latency-determining methods could also cut into the savings.

Another set of methods are the waveform relaxation techniques which increase efficiency by temporarily ignoring couplings between nodes. The simplest version of the method is as follows. Consider a circuit with n nodes which requires m time points for its solution. The circuit can be represented by the vector equation:

$$F_i(V(t)) + \frac{dQ_i(V(t))}{dt} = 0 \tag{16.27}$$

Using trapezoidal time integration gives a new function:

$$W_i(V(k)) = F_i(V(k)) + F_i(V(k-1)) + 2[Q_i(V(k)) - Q_i(V(k-1))]/dt = 0 \tag{16.28}$$

We need to find the $V(k)$ which makes W zero for all k time points at all i nodes. The normal method solves for all n nodes simultaneously at each time point before advancing k. Waveform relaxation solves for all m time points at a single node (calculates the waveform at that node) before advancing to the next node. An outer loop is used to assure that all the individual nodal waveforms are consistent with each other.

Waveform relaxation is extremely efficient as long as the number of outer loops is small. The number of iterations will be small if the equations are solved in the correct order; that is, starting on nodes which are signal sources and following the direction of signal propagation through the circuit. This way, the

waveform at node $i + 1$ will depend strongly on the waveform at node i, but the waveform at node i will depend weakly on the signal at node $i + 1$. The method is particularly effective if signal propagation is unidirectional, as is sometimes the case in logic circuits. During practical implementation, the total simulation interval is divided into several subintervals and the subintervals are solved sequentially. This reduces the total number of time points which must be stored. Variants of the method solve small numbers of tightly coupled nodes as a group; such a group might include all the nodes in a TTL gate or in a small feedback loop. Large feedback loops can be handled by making the simulation time for each subinterval less than the time required for a signal to propagate around the loop.

The efficiency of this method can be further improved using different time steps at different nodes, yielding a multi-rate method. This way, during a given interval, small time steps are used at active nodes while large steps are used at inactive nodes (taking advantage of latency).

16.10 Commercially Available Simulators

The simulations in this chapter were performed with the evaluation version of PSPICE from Microsim and AMS from Antrim design systems. The following vendors market circuit simulation software. The different programs have strengths in different areas and most vendors allow you to try their software in-house for an "evaluation period" before you buy.

SPICE2-SPICE3	University of California Berkeley, CA
AMS	Antrim Design Systems, Scotts Valley, CA., www.antrim.com
PSPICE	Orcad Corporation, Irvine, CA, www.orcad.com
HSPICE	Avant! Coproration, Fremont, CA, www.avanticorp.com
ISPICE	Intusoft, SanPedro, CA, www.intusoft.com
SABER	Analogy, Beaverton, OR, www.analogy.com
SPECTRE	Cadence Design Systems, San Jose, CA, www.cadence.com
TIMEMILL	Synopsys Corporation, Sunnyvale, CA, www,synopsys.com
ACCUSIM II	Mentor Graphics, Wilsonville, OR, www.mentorg.com

References

On general circuit simulation:

1. J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, New York, Van Nostrand Reinhold, 1983.
2. A. E. Ruehli (Editor), *Circuit Analysis, Simulation and Design, Part 1*, Elsevier Science Publishers, B.V. North Holland, 1981.
3. L. Nagel, SPICE2: A Computer Program to Simulate Semiconductor Circuits, Ph.D. thesis, University of California, Berkeley, 1975.
4. K. Kundert, *The Designers Guide to SPICE and SPECTRE*, Kluwer Academic Publishers, 1995.
5. P. W. Tuinenga, *SPICE, A Guide to Circuit Simulation and Analysis Using PSPICE*, Prentice-Hall, Englewood Cliffs, NJ, 1988.
6. J. A. Connelley and P. Choi, *Macromodeling with SPICE*, Prentice-Hall, Engelwood Cliffs, NJ, 1992.
7. P. Gray and R. Meyer, *Analysis and Design of Analog Integrated Circuits*, Wiley, New York, 1977.
8. A. Vladimiresch, *The SPICE Book*, Wiley, New York, 1994.

On modern techniques:

9. A. E. Ruehli (Editor), *Circuit Analysis, Simulation and Design, Part 2*, Elsevier Science Publishers, B.V. North Holland, 1981.

On device models:

10. P. Antognetti and G. Massobrio, *Semiconductor Modeling with SPICE2*, McGraw-Hill, New York, 2nd ed., 1993.
11. D. Foty, *MOSFET Modeling with SPICE*, Prentice-Hall, Englewood Cliffs, NJ, 1997.

12. BSIM3 Users Guide, www-device.EECS.Berkeley.EDU/~bsim3.
13. MOS-9 models, www-us2.semiconductors.philips.com/Philils_Models.

On Verilog-A

14. D. Fitzpatrick, *Analog Behavior Modeling with the Verilog-A Language*, Kluwer-Academic Publishers, 1997.

Nakhla, M.S., Achar, R. "Interconnect Modeling and Simulation"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

17

Interconnect Modeling and Simulation

Michel S. Nakhla
Ramachandra Achar
Carleton University

17.1 Introduction

With the rapid developments in VLSI technology, design, and CAD techniques, at both the chip and package level, the central processor cycle times are reaching the vicinity of 1 ns and communication switches are being designed to transmit data that have bit rates faster than 1 Gb/s. The ever-increasing quest for high-speed applications is placing higher demands on interconnect performance and highlights the previously negligible effects of interconnects (Fig. 17.1), such as ringing, signal delay, distortion, reflections, and crosstalk.[1–33] In addition, the trend in the VLSI industry toward miniature designs, low power consumption, and increased integration of analog circuits with digital blocks has further complicated the issue of signal integrity analysis. Fig. 17.2 describes the effect of scaling of chip on the global interconnect delay. As seen, the global interconnect delay grows as a cubic power of the scaling factor.[1] It is predicted that interconnects will be responsible for nearly 70 to 80% of the signal delay in high-speed systems.

Thousands of engineers, intent on the best design possible, use SPICE[7] on a daily basis for analog simulation and general-purpose circuit analysis. However, the high-speed interconnect problems are not always handled appropriately by the present levels of SPICE. If not considered during the design stage, these interconnect effects can cause logic glitches that render a fabricated digital circuit inoperable, or they can distort an analog signal such that it fails to meet specifications. Since extra iterations in the

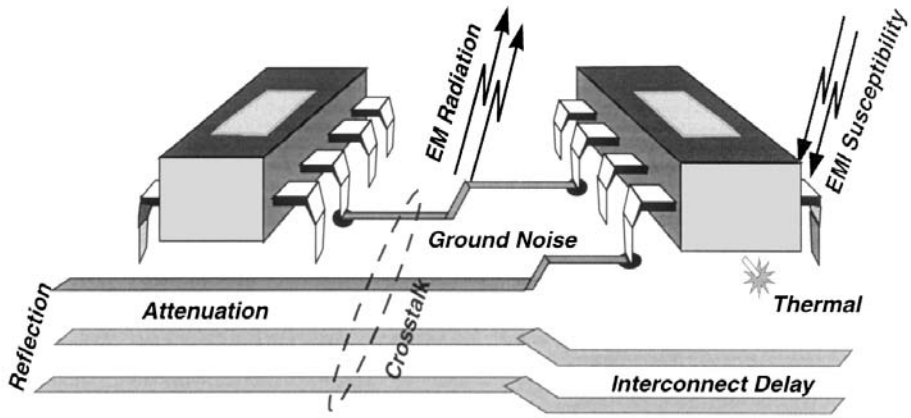


FIGURE 17.1 High-speed interconnect effects.

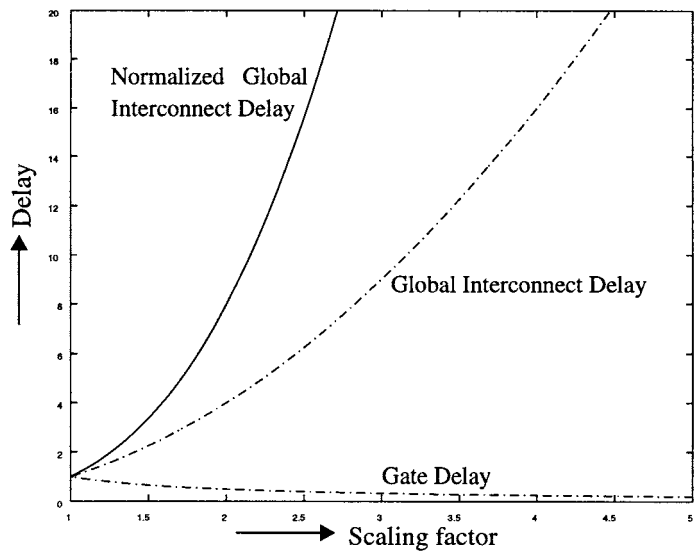


FIGURE 17.2 Impact of scaling on signal delay in high-speed systems.

design cycle are costly, accurate prediction of these effects is a necessity in high-speed designs. Hence, it becomes extremely important for designers to simulate the entire design along with interconnect subcircuits as efficiently as possible while retaining the accuracy of simulation.[12,26–65]

What Is High Speed?

Speaking on a broader perspective, a "high-speed interconnect" is the one in which the time taken by the propagating signal to travel between its end points cannot be neglected. An obvious factor that influences this definition is the physical extent of the interconnect — the longer the interconnect, the more time the signal takes to travel between its end points. Smoothness of signal propagation suffers once the line becomes long enough for the signal's rise/fall times to roughly match its propagation time through the line. Then, the interconnect electrically isolates the driver from the receivers, which no longer

function directly as loads to the driver. Instead, within the time of the signal's transition between its high and low voltage levels, the impedance of the interconnect becomes the load for the driver and also the input impedance to the receivers.[1–6] This leads to various transmission line effects, such as reflections, overshoot, undershoot, and crosstalk, and modeling of these needs the blending of EM and circuit theory.

Alternatively, the term "high-speed" can be defined in terms of the frequency content of the signal. At low frequencies, an ordinary wire (i.e., an interconnect) will effectively short two connected circuits. However, this is not the case at higher frequencies. The same wire, which is so effective at lower frequencies for connection purposes, has too much inductive/capacitive effect to function as a short at higher frequencies. Faster clock speeds and sharper slew rates tend to add more and more high-frequency contents.

An important criterion used for classifying interconnects is the *electrical length* of an interconnect. An interconnect is considered to be "electrically short," if at the highest operating frequency of interest, the interconnect length is physically shorter than one-tenth of the wavelength (i.e., length of the interconnect/$\lambda < 0.1$, $\lambda = v/f$). Otherwise the interconnect is referred as electrically long.[1,8] In most digital applications, the desired highest operating frequency (which corresponds to the minimum wavelength) of interest is governed by the rise/fall time of the propagating signal. For example, the energy spectrum of a trapezoidal pulse is spread over an infinite frequency range; however, most of the signal energy is concentrated near the low-frequency region and decreases rapidly with increase in frequency (this is illustrated in Fig.17.3 for two different instances of rise times: 1 ns and 0.1 ns). Hence, ignoring the high-frequency components of the spectrum above a maximum frequency, f_{max}, will not seriously alter the overall signal shape. Consequently, for all practical purposes, the width of the spectrum can be assumed to be finite. In other words, the signal energy of interest is assumed to be contained in the major lobe of the spectrum, and f_{max} can be defined as coresponding to 3-dB bandwidth point[2,3,25]

$$f_{max} = \frac{0.35}{t_r} \tag{17.1}$$

where t_r is the rise/fall time of the signal. This implies that, for example, for a rise time of 0.1 ns, the maximum frequency of interest is approximately 3 GHz or the minimum wave-length of interest is 10 cm. In some cases, the limit can be more conservatively set as[25]

$$f_{max} = \frac{1}{t_r} \tag{17.2}$$

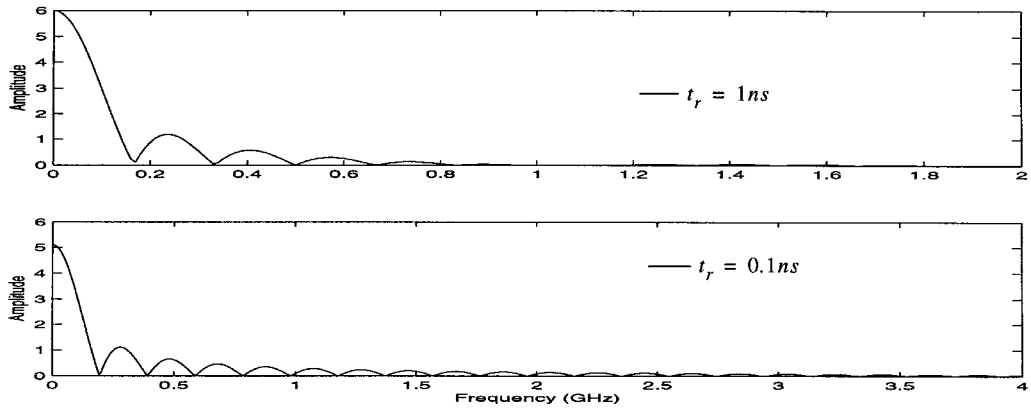


FIGURE 17.3 Frequency spectrum of trapezoidal pulse t_{pw} = 5 ns.

In summary, the primary factors with regard to high-speed signal distortion effects that should be considered are interconnect length, cross-sectional dimensions, signal slew rate, and clock speed. Other factors that also should be considered are logic levels, dielectric material, and conductor resistance. Electrically short interconnects can be represented by lumped models, whereas electrically long interconnects need distributed or full-wave models.

17.2 Interconnect Models

High-speed system designers are driven by the motivation to have signals with higher clock and slew rates while at the same time to innovate on reducing the wiring cross-section as well as packing the lines together. Reducing the wiring dimensions results in appreciably resistive lines. In addition, all these interconnections may have non-uniform cross-sections caused by discontinuities such as connectors, vias, wire bonds, flip-chip solder balls, redistribution leads, and orthogonal lines. Interconnections can be from various levels of design hierarchy (Fig. 17.4) such as on-chip, packaging structures, multi-chip modules, printed circuit boards, and backplanes. On a broader perspective, interconnection technology can be classified into five categories, as shown in Table 17.1,[6] namely, on-chip wiring, thin-film wiring, ceramic carriers, thin-film wiring, printed circuit boards, and shielded cables. For the categories shown in Table 17.1, the wavelength is of the order of 1 to 10 cm. The propagated signal rise times are in the range 100 to 1000 ps. Hence, the line lengths are either comparable or much longer than the signal wavelengths. Depending on the operating frequency, signal rise times, and the nature of the structure, the interconnects can be modeled as lumped (RC or RLC), distributed (frequency-independent/dependent RLCG parameters, lossy, coupled), full-wave models, or measured linear subnetworks.

Lumped Models

In the past, interconnect models have been generally restricted to RC tree models. RC trees are RC circuits with capacitors from all nodes to ground, no floating capacitors, no resistors to ground. The signal delay

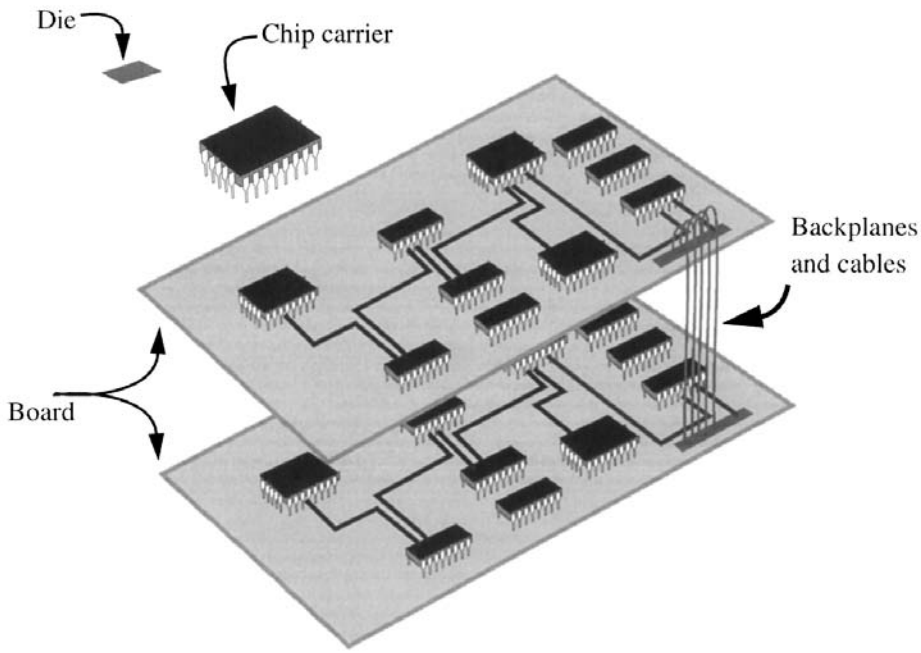


FIGURE 17.4 Interconnect hierarchy.

TABLE 17.1 Interconnect Technologies

Interconnection Type	Line Width (μm)	Line Thickness (μm)	Line Resistance (ohm/cm)	Maximum Length (cm)
On-chip	0.5–2	0.7–2	100–1000	0.3–1.5
Thin-film	10–25	5–8	1.25–4	20–45
Ceramic	75–100	16–25	0.4–0.7	20–50
Printed circuit board	60–100	30–50	0.06–0.08	40–70
Shielded cables	100–450	35–450	0.0013–0.033	150–500

through RC trees were often estimated using a form of the *Elmore delay*,[8,34] which provided a dominant time constant approximation for monotonic step responses.

Elmore Delay

There are many definitions of delay, given the actual transient response. Elmore delay is defined as the time at which the output transient rises to 50% of its final value. Elmore's expression approximates the mid-point of the *monotonic step response* waveform by the mean of the impulse response as

$$T_D = \int_0^\infty t\dot{v}dt \tag{17.3}$$

Since $v(t)$ is monotonic, its first derivative (the impulse response) will have the form of a probability density function. The mean of the distribution of the first derivative is a good approximation for the 50% point of the transient portion of $v(t)$. For an RC tree, Elmore's expression can be applied since step responses for these circuits are always monotonic.

However, with increasing signal speeds, and in diverse technologies such as bipolar, BiCMOS, or MCMs, RC tree models are no longer adequate. In bipolar circuits, lumped-element interconnect models may require the use of inductors or grounded resistors, which are not compatible with RC trees. Even for MOS circuits operating at higher frequencies, the effects of coupling capacitances may need to be included in the delay estimate.

RLC circuits with non-equilibrium initial conditions may have responses that are non-monotonic. This typically results in visible signal ringing in the waveform. A single time constant approximation with Elmore delay is not generally sufficient for such circuits. Usually, lumped interconnect circuits extracted from layouts contain large number of nodes, which make the simulation highly CPU intensive. Figure 17.5 shows a general lumped model where *r*, *l*, *c*, and *g* correspond to the resistance, inductance, capacitance, and conductance of the interconnect, respectively.

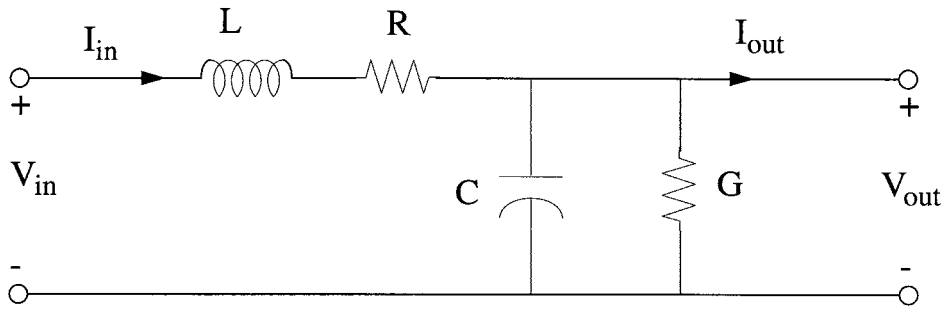


FIGURE 17.5 Lumped-component model.

Distributed Transmission Line Models

At relatively higher signal speeds, the electrical length of interconnects becomes a significant fraction of the operating wavelength, giving rise to signal-distorting effects that do not exist at lower frequencies. Consequently, the conventional lumped impedance interconnect models become inadequate, and transmission line models based on quasi-TEM assumptions are needed. The *TEM* (*Transverse Electromagnetic Mode*) approximation represents the ideal case, where both E and H fields are perpendicular to the direction of propagation and it is valid under the condition that the line cross-section is much smaller than the wavelength. However, in practical wiring configurations, the structure has all the inhomogeneities mentioned previously. Such effects give rise to E or H fields in the direction of propagation. If the line cross-section or the extent of these non-uniformities remain a small fraction of the wavelength in the frequency range of interest, the solution to Maxwell's equations are given by the so-called quasi-TEM modes and are characterized by distributed R, L, C, G per unit length parameters (Fig. 17.6).

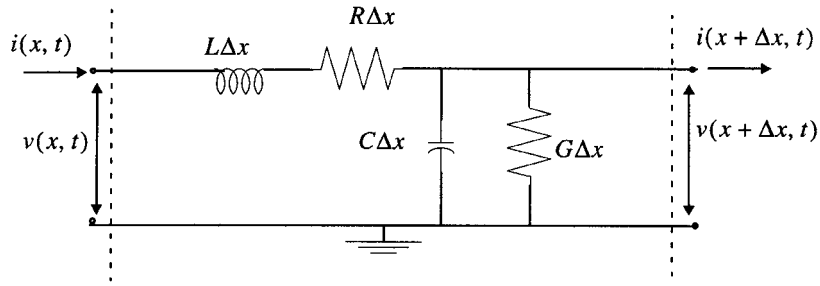


FIGURE 17.6 One seqment of distributed transmission line model.

The basic quasi-TEM model is the simple "delay" line or lossless line ($R = G = 0$). In this case, a signal traveling along a line has the same amplitude at all points, but is shifted in time with a propagation delay per unit length (τ) given by

$$\tau = \sqrt{LC} = \sqrt{\varepsilon\mu} \tag{17.4}$$

where ε and μ are the dielectric permitivity and permeability of the medium, respectively. The characteristic impedance for the lossless case is given by

$$Z = \sqrt{L/C} \tag{17.5}$$

More complicated models include per-unit-length loss (either in the direction of traveling wave or due to dielectric substrate loss) or coupling between adjacent transmission lines, where the coupling may be resistive, inductive, capacitive, or a combination of these.[8] In such cases, the propagation constants (γ) and the characteristic impedances (Z) are given by

$$\gamma = \sqrt{(R + jwL)(G + jwC)} \tag{17.6}$$

$$Z = \sqrt{\frac{R + jwL}{G + jwC}} \tag{17.7}$$

Distributed Models with Frequency-Dependent Parameters

As the operating frequency increases, the per unit length parameters of the transmission line can vary. This is mainly due to varying current distribution in the conductor and ground plane caused

by the induced electric field. This phenomenon can be categorized as follows: skin, edge, and proximity effects.[8,20,56]

Edge and Proximity Effects

The *edge* and *proximity effects* influence the interconnect parameters in the low to medium frequency region. The edge effect causes the current to concentrate near the sharp edges of the conductor, thus raising the resistance. It affects both the signal and ground conductors, but is more pronounced on signal conductors. The proximity effect causes the current to concentrate in the sections of ground plane that are close to the signal conductor. This modifies the magnetic field between the two conductors, which in turn reduces the inductance per unit length. It also raises the resistance per unit length as more current is crowded in the ground plane under the conductor. The proximity effect appears at medium frequencies. While both effects need to be accounted for, the proximity effect seems to have more significance, especially in its effect on the inductance.

Skin Effect

The *skin effect* causes the current to concentrate in a thin layer at the conductor surface. It is pronounced mostly at high frequencies on both the signal and ground conductors. The current distribution falls off exponentially as we approach the interior of the conductor. The average depth of current penetration is a function of frequency and is known as *skin depth*, which is given by

$$\delta = \left(\frac{2\rho}{w\mu}\right)^{1/2} \tag{17.8}$$

where w is frequency (rad/s), ρ is the volume resistivity, and μ is the magnetic permeability. This results in the resistance being proportional to the square root of the frequency at very high operating frequencies. The magnetic fields inside the conductors are also reduced due to skin effect. This reduces the internal inductance and therefore the total inductance. At even higher frequencies, as the internal inductance approaches zero, the edge, and proximity effects being fully pronounced, the inductance becomes essentially constant.

Typical Behavior of *R* and *L*

The frequency plots of R and L of the microstrip in Fig. 17.7 are shown in Fig. 17.8 and Fig. 17.9. These plots present a typical behavior of R and L in general. L starts off as essentially constant until the edge and proximity effects get into effect. The edge effect causes the resistance to increase, and the current crowding under signal conductor (due to the proximity effect) causes the inductance to decrease and resistance to increase. As we go higher in frequency, the edge and proximity effects become fully pronounced and will cause little additional change in R and L, but the skin effect becomes significant. Initially, the inductance is reduced due to skin effect because of the reduction of magnetic fields inside the conductors; but as the contribution of those magnetic fields to the overall inductance becomes insignificant, L becomes essentially constant at very high frequency. The resistance, on the other hand, becomes a direct function of the skin depth and therefore varies with the square root of the frequency.

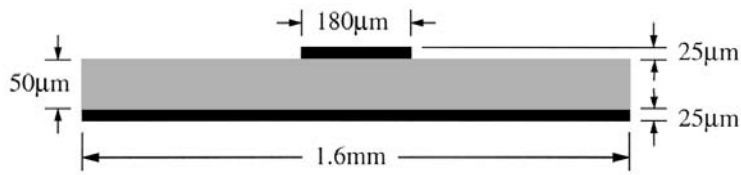


FIGURE 17.7 An interconnect over a ground plane.

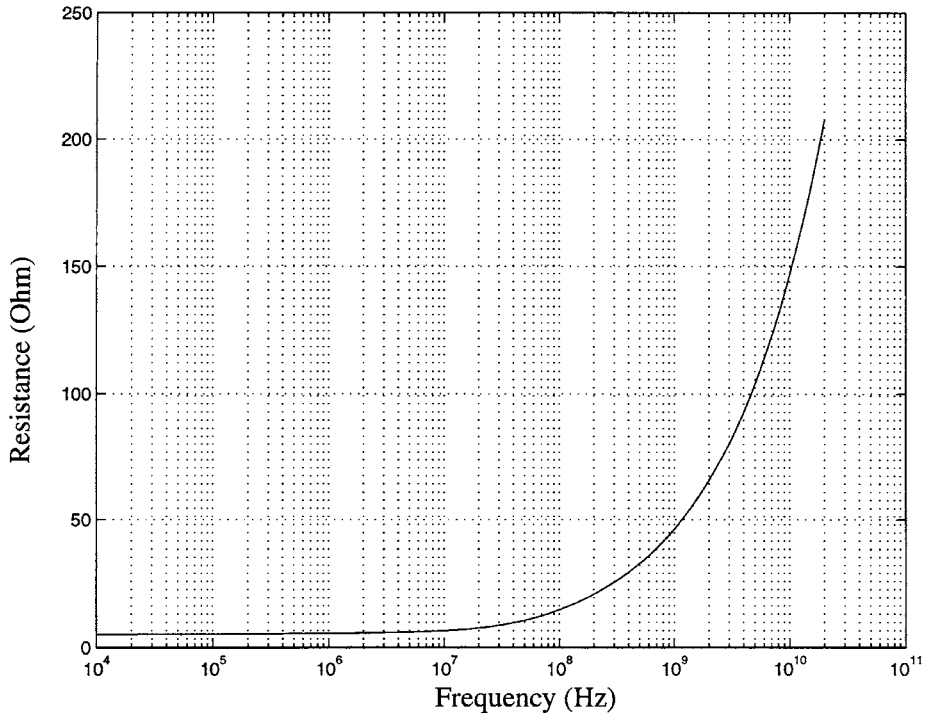


FIGURE 17.8 Frequency-dependent resistance.

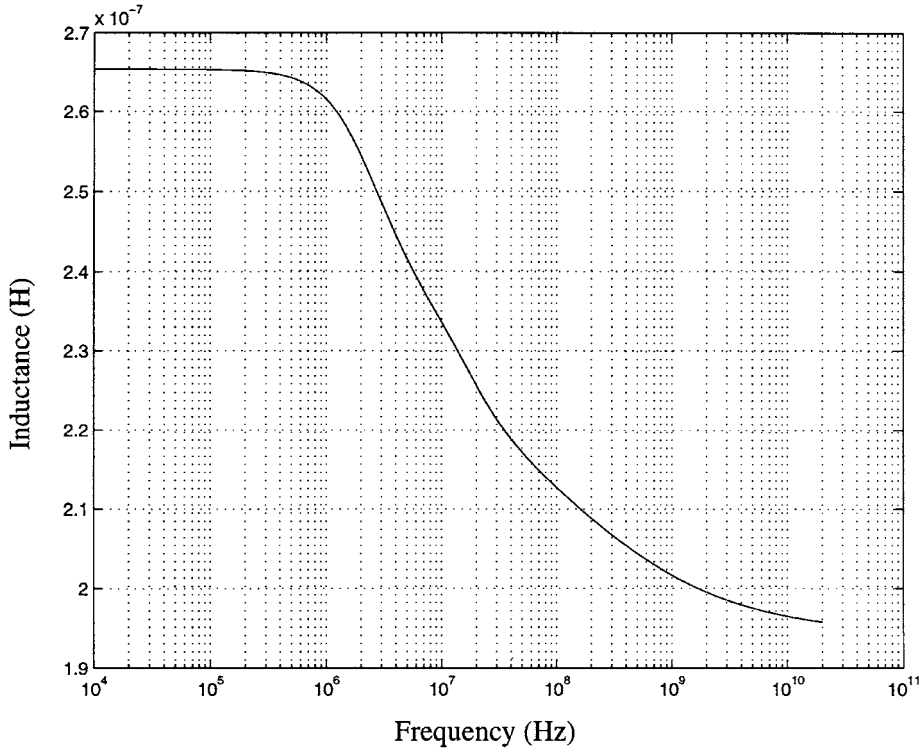


FIGURE 17.9 Frequency-dependent inductance.

Full-Wave Models

At further subnanosecond rise times, the line cross-section or the non-uniformities become a significant fraction of the wavelength and, under these conditions, the field components in the direction of propagation can no longer be neglected (Fig. 17.10). Consequently, even the distributed models based on quasi-TEM approximations become inaccurate in describing the interconnect performance.[12–19] In such situations, full-wave models, which take into account all possible field components and satisfies all boundary conditions, are required to give an accurate estimation of high-frequency effects.

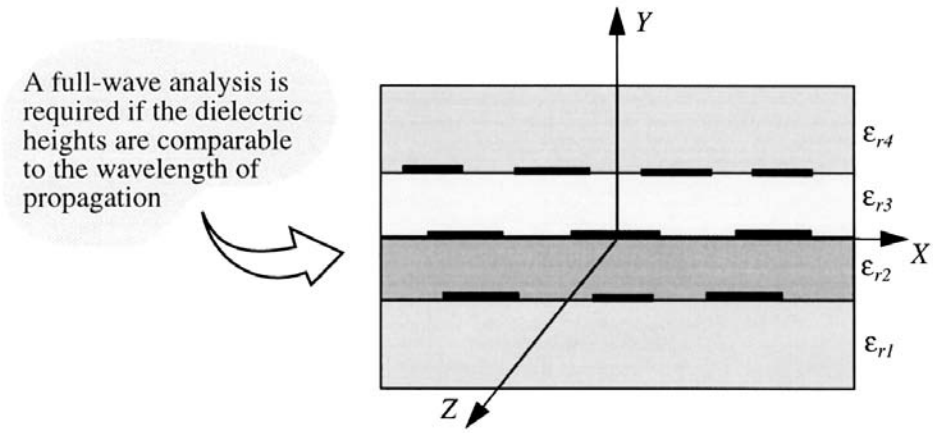


FIGURE 17.10 Cross-sectional view of a multiconductor dispersive system.

The information that is obtained through a full-wave analysis is in terms of field parameters such as propagation constant, characteristic impedance, etc. A typical behavior of the modal propagation constant and characteristic impedances obtained using the full-wave spectral domain method for the structure shown in Fig. 17.11 is given in Fig. 17.12. The deviation suffered by the quasi-TEM models with respect to full-wave results is illustrated through a simple test circuit shown in Fig. 17.13. As seen from Fig. 17.14 for the structure under consideration, quasi-TEM results deviated from full-wave results as early as 400 MHz. The differences in the modeling schemes with respect to the transient responses are illustrated in Fig. 17.15 and Fig. 17.16. *In general, an increase in the dielectric thickness causes quasi-TEM models to become inaccurate at relatively lower frequencies.* This implies that a full-wave analysis becomes necessary as we move up in the integration hierarchy, from the chip to PCB/system level. It is found that depending on the interconnect structure, when the cross-sectional dimensions approach 1/40 to 1/10 of the effective

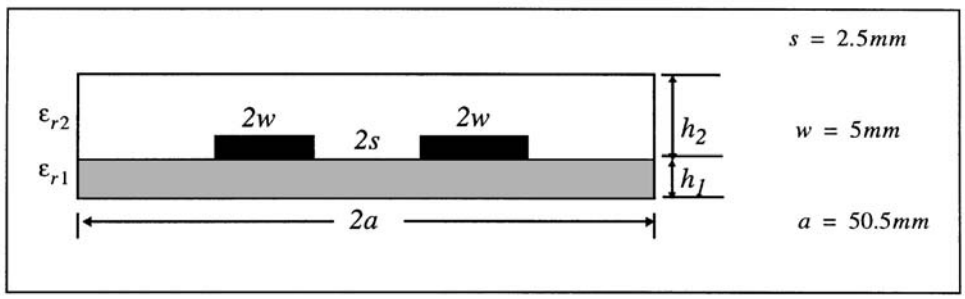


FIGURE 17.11 A coupled interconnect structure.

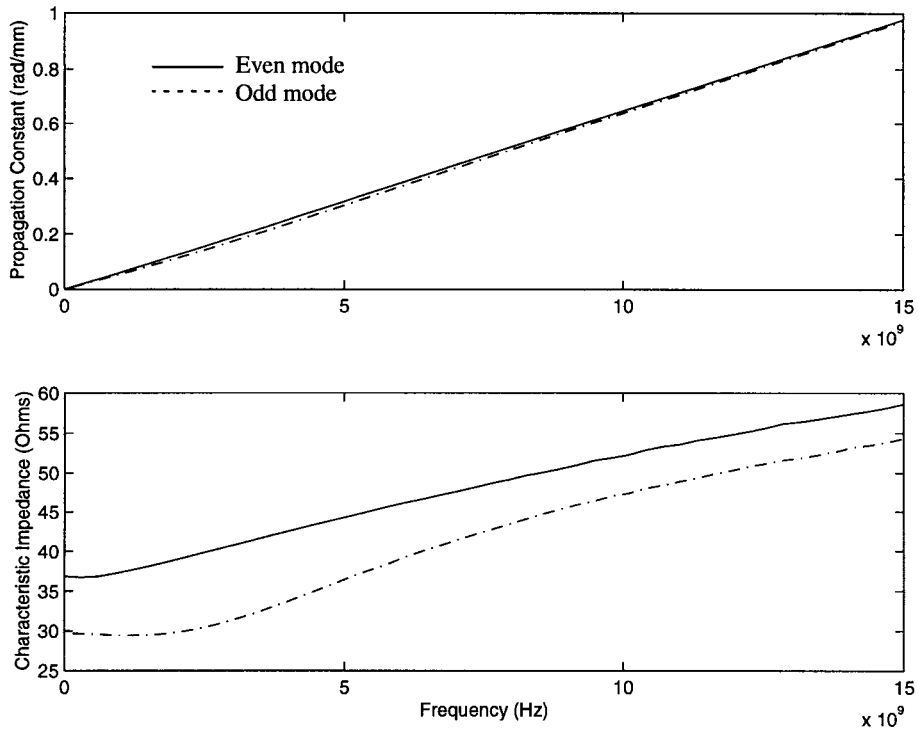


FIGURE 17.12 Modal propagation constans and charactersitic impedances.

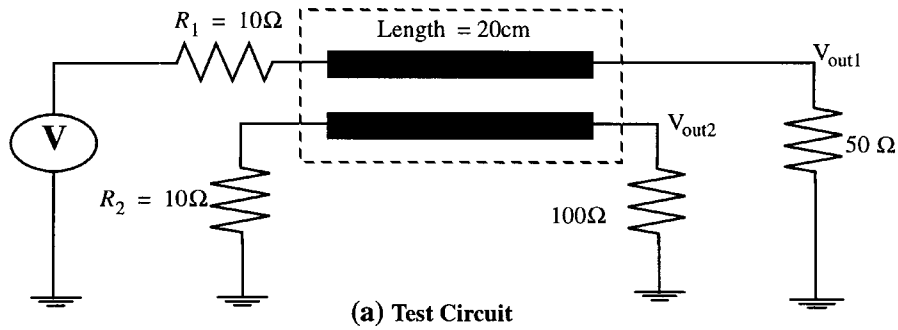


FIGURE 17.13 Test circuit for simulation of interconnect structure in Fig. 18.13.

wavelength, quasi-TEM approximation deviates considerably from full-wave results. (The effective wavelength of a wave propagating in a dielectric medium at a certain frequency is given by

$$\lambda_e = \frac{velocity\ of\ the\ wave}{frequency} = \frac{1/(\sqrt{\mu_r E_{effective}})}{f} \tag{17.9}$$

Also, a reduction in the separation width between adjacent conductors makes the full-wave analysis more essential, especially for crosstalk evaluation. The same is true with an increase in the dielectric constant values.

However, circuit simulation of full-wave models is highly involved. A circuit simulator requires the information in terms of currents, voltages, and circuit impedances. This demands a generalized method to combine modal results into circuit simulators in terms of a full-wave stencil. Another important issue involved here is the cost of a full-wave analysis associated with each interconnect subnetwork at each

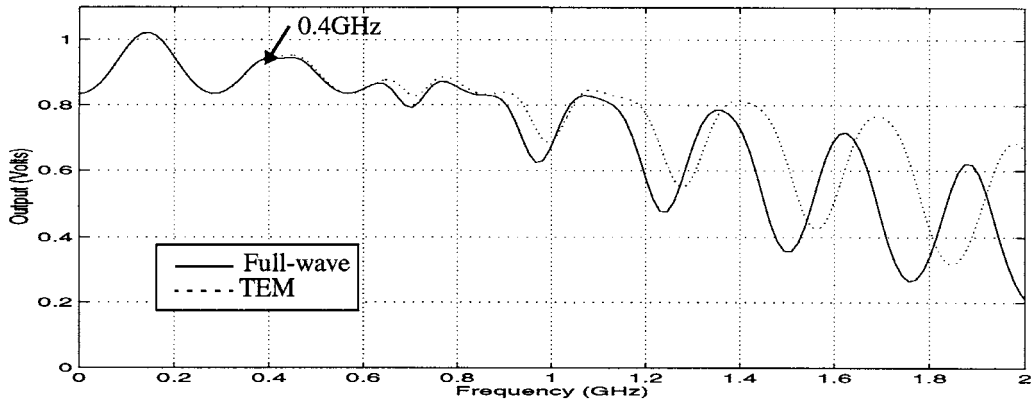


FIGURE 17.14 Comparison of full-wave and quasi-TEM frequency responses.

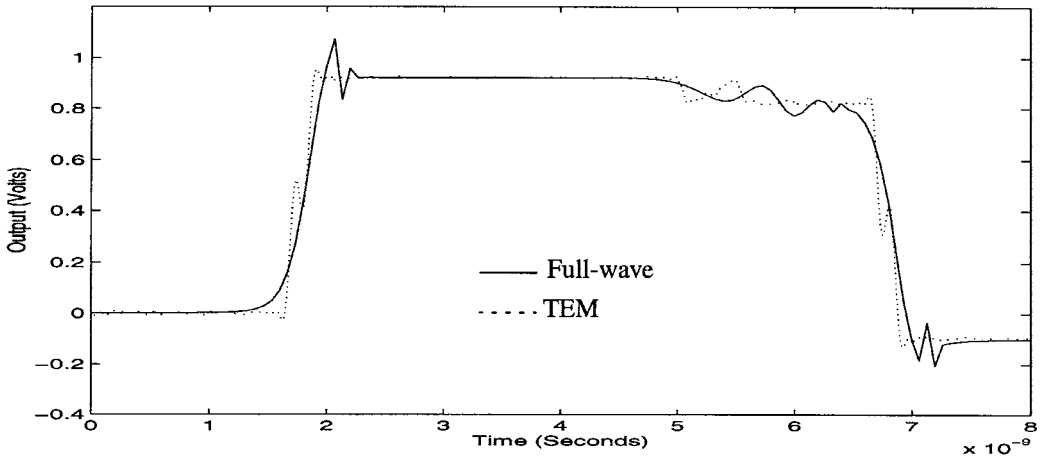


FIGURE 17.15 Comparison of full-wave and quasi-TEM transient responses at node V_{out1}.

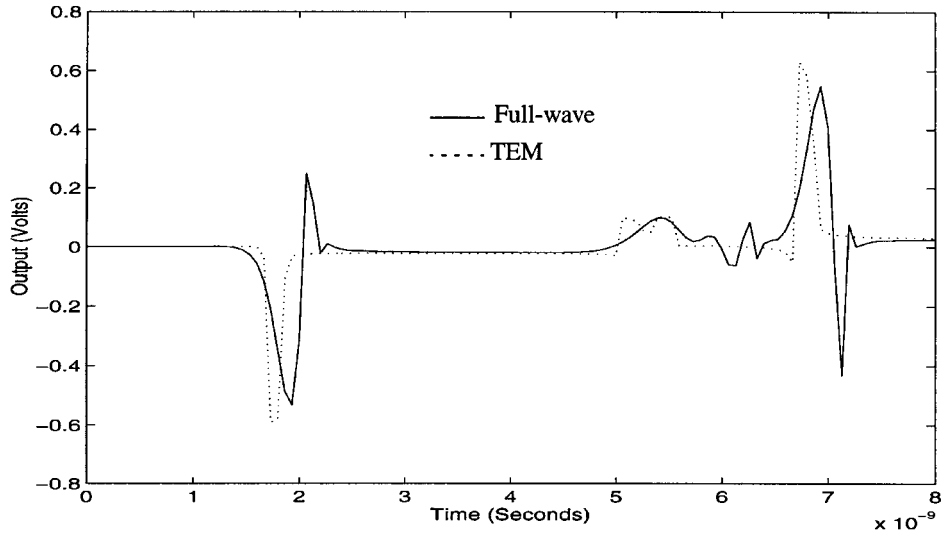


FIGURE 17.16 Comparison of full-wave and quasi-TEM crosstalk responses (node V_{out2}).

frequency point of analysis. For typical high-speed interconnect circuits which need thousands of frequency point solutions to get accurate responses, it would become prohibitively CPU-expensive to simulate because of the combined cost involved (i.e., evaluation of a full-wave model to obtain modal parameters and computation of circuit response at each point).

Measured Subnetworks

In practice, it may not be possible to obtain accurate analytical models for interconnects because of the geometric inhomogeneity and associated discontinuities. To handle such situations, modeling techniques based on measured data have been proposed in the literature.[22,24,59–65] In general, the behavior of high-speed interconnects can easily be represented by measured frequency-dependent scattering parameters or time-domain terminal measurements. Time-domain data could be obtained by time-domain reflectometry (TDR) measurements[59] or electromagnetic techniques.[23,24] One important factor to note here is that the subnetwork can be characterized by large sets of time-domain data, making the system overdetermined. This kind of measurement data in large number of sets is essential in a practical environment as the measurements are usually contaminated by noise, and the use of only a single set of measurements may lead to inaccurate results. Including all available sets of measurements in the simulation helps to reduce the impact of noise on network responses. However, handling such overdetermined situations in circuit simulation is a tedious and a computationally expensive process.[22]

EMI Subnetworks

Electrically long interconnects function as spurious antennas to pick up emissions from other nearby electronic systems. This makes susceptibility to emissions a major concern to current system designers of high-frequency products. Hence, the availability of interconnect simulation tools, including the effect of incident fields, is becoming an important design requirement. In addition, analysis of radiations from interconnects is also becoming increasingly important in high-furnace designs.[9,57,58]

17.3 Distributed Transmission Line Equations

Transmission line characteristics are in general described by Telegrapher's equations. Consider the multiconductor transmission line (MTL) system shown in Fig. 17.17. Telegrapher's equations for such a structure are derived by discretizing the lines into infinitesimal sections of length Δx and assuming uniform per-unit length parameters of resistance (R), inductance (L), conductance (G), and capacitance (C). Each section then includes a resistance $R\Delta x$, inductance $L\Delta x$, conductance $G\Delta x$, and capacitance $C\Delta x$ (Fig. 17.6). Using Kirchoff's current and voltage laws, one can write

$$v(x + \Delta x, t) = v(x, t) - R\Delta x i(x, t) - L\Delta x \frac{\partial}{\partial t} i(x, t) \tag{17.10}$$

or

$$\frac{v(x + \Delta x, t) - v(x, t)}{\Delta x} = -Ri(x, t) - L\frac{\partial}{\partial t} i(x, t) \tag{17.11}$$

Taking the limit $\Delta x \to 0$, one gets

$$\frac{\partial}{\partial x} v(x, t) = -Ri(x, t) - L\frac{\partial}{\partial t} i(x, t) \tag{17.12}$$

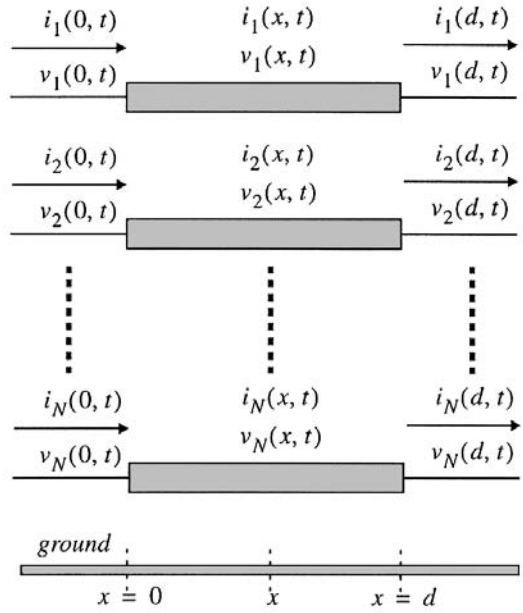


FIGURE 17.17 Multiconductor transmission line system.

Similarly for $i(x, t)$, one can write

$$\frac{\partial}{\partial x} i(x, t) = -Gv(x, t) - C\frac{\partial}{\partial t} v(x, t) \tag{17.13}$$

The equations for a single line also hold good for multiple coupled lines, with a modification that per-unit-length parameters now become matrices (R, L, G, and C) and voltage–current variables become vectors represented by V and I, respectively. Noting this and taking Laplace transform of Eqs. 17.12 and 17.13, one can write

$$\frac{\partial}{\partial x} \boldsymbol{V}(x, s) = -(\boldsymbol{R} + s\boldsymbol{L})\boldsymbol{I}(x, s) \tag{17.14}$$

$$\frac{\partial}{\partial x} \boldsymbol{I}(x, s) = -(\boldsymbol{G} + s\boldsymbol{C})\boldsymbol{V}(x, s) \tag{17.15}$$

Equations 17.14 and 17.15 can be written as

$$\frac{\partial}{\partial x} \boldsymbol{V}(x, s) = -\boldsymbol{Z}_P \boldsymbol{I}(x, s) \tag{17.16}$$

$$\frac{\partial}{\partial x} \boldsymbol{I}(x, s) = -\boldsymbol{Y}_P \boldsymbol{V}(x, s) \tag{17.17}$$

where $\boldsymbol{Z}_P$ and $\boldsymbol{Y}_P$ represent the impedance and admittance matrices, given by

$$\boldsymbol{Z}_P = \boldsymbol{R} + s\boldsymbol{L}; \qquad \boldsymbol{Y}_P = \boldsymbol{G} + s\boldsymbol{C} \tag{17.18}$$

Eigenvalue-Based Transmission Line Stencil

The two differential equations given in Eqs. 17.16 and 17.17 can be combined into a set of wave equations as

$$\frac{\partial}{\partial x^2}\boldsymbol{V}(x, s) = \boldsymbol{Z}_P\boldsymbol{Y}_P\boldsymbol{V}_P(x, s) \tag{17.19}$$

$$\frac{\partial}{\partial x^2}\boldsymbol{I}(x, s) = \boldsymbol{Y}_P\boldsymbol{Z}_P\boldsymbol{I}_P(x, s) \tag{17.20}$$

which will have a solution of the form

$$\boldsymbol{V}_m(x, s) = \boldsymbol{V}_m(0, s)e^{\pm\gamma_m(s)x} \tag{17.21}$$

$$\boldsymbol{I}_m(x, s) = \boldsymbol{I}_m(0, s)e^{\pm\gamma_m(s)x} \tag{17.22}$$

where $\gamma_m(s)$ is the complex propagation constant. Substituting the solution forms of Eqs. 17.21 and 17.22 into wave equation 17.19 yields

$$det\{\gamma_m^2\boldsymbol{U} - \boldsymbol{Z}_P\boldsymbol{Y}_P\}\,\boldsymbol{V}_m(0) = 0 \tag{17.23}$$

where $\boldsymbol{U}$ is an identity matrix. $\boldsymbol{V}_m(0)$ will have nontrivial solutions if γ_m^2 satisfies the eigenvalue problem given by

$$det\{\gamma_m^2\boldsymbol{U} - \boldsymbol{Z}_P\boldsymbol{Y}_P\} = 0 \tag{17.24}$$

For inhomogeneous dielectrics, there exist in general m distinct eigenvalues where $m = 1, 2, \ldots, N$. Each eigenvalue has its corresponding eigenvector S_m. Let Γ be a diagonal matrix whose elements are the complex propagation constants $\{\gamma_1, \gamma_2, \ldots, \gamma_N\}$. Let S_v be a matrix with eigenvectors S_m placed in respective columns. The transmission line stencil can now be derived after little manipulations as

$$\boldsymbol{P}\boldsymbol{V}(s) + \boldsymbol{Q}\boldsymbol{I}(s) = 0 \tag{17.25}$$

where

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{S}_V\boldsymbol{E}_1\boldsymbol{S}_v^{-1} & -\boldsymbol{U} \\ \boldsymbol{S}_i\boldsymbol{E}_2\boldsymbol{S}_v^{-1} & \boldsymbol{0} \end{bmatrix}; \qquad \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{S}_v\boldsymbol{E}_1\boldsymbol{S}_v^{-1} & \boldsymbol{0} \\ \boldsymbol{S}_i\boldsymbol{E}_2\boldsymbol{S}_i^{-1} & -\boldsymbol{U} \end{bmatrix} \tag{17.26}$$

$$\boldsymbol{E}_1 = diag\left\{\frac{e^{(-\gamma_m d)} + e^{(\gamma_m d)}}{2}\right\}; \qquad \boldsymbol{E}_2 = diag\left\{\frac{e^{(-\gamma_m d)} - e^{(\gamma_m d)}}{2}\right\} \tag{17.27}$$

where d is length of the line and $m = 1, 2, \ldots, N$. S_i is computed as $\boldsymbol{S}_i = \boldsymbol{Z}_P^{-1}\boldsymbol{S}_v\boldsymbol{\Gamma}$. V and I represent the Laplace-domain terminal voltage and current vectors of multiconductor transmission line, given by

$$\boldsymbol{V}(s) = \begin{bmatrix} \boldsymbol{V}(0) \\ \boldsymbol{V}(d) \end{bmatrix}; \qquad \boldsymbol{I}(s) = \begin{bmatrix} \boldsymbol{I}(0) \\ \boldsymbol{I}(d) \end{bmatrix} \tag{17.28}$$

The MTL stencil described by Eq. 17.25 is widely used in moment matching techniques (MMTs) for transmission line analysis.[38] Another form of MTL stencil is also quite popular and it has the matrix exponential form,[44] which is explained below.

Matrix Exponential Stencil

Equations 17.14 and 17.15 can be written in the hybrid form as

$$\frac{\partial}{\partial x}\begin{bmatrix} \boldsymbol{V}(x, s) \\ \boldsymbol{I}(x, s) \end{bmatrix} = (\boldsymbol{D} + s\boldsymbol{E})\begin{bmatrix} \boldsymbol{V}(x, s) \\ \boldsymbol{I}(x, s) \end{bmatrix} \tag{17.30}$$

where

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{0} & -\boldsymbol{R} \\ -\boldsymbol{G} & \boldsymbol{0} \end{bmatrix}; \qquad \boldsymbol{E} = \begin{bmatrix} \boldsymbol{0} & -\boldsymbol{L} \\ -\boldsymbol{C} & \boldsymbol{0} \end{bmatrix} \tag{17.30}$$

Hybrid transmission line stencil in the exponential form can be written as

$$\begin{bmatrix} \boldsymbol{V}(d, s) \\ \boldsymbol{I}(d, s) \end{bmatrix} = e^{(\boldsymbol{D} + s\boldsymbol{E})d}\begin{bmatrix} \boldsymbol{V}(0, s) \\ \boldsymbol{I}(0, s) \end{bmatrix} \tag{17.31}$$

Parameters $\boldsymbol{P}$ and $\boldsymbol{Q}$ of the transmission line in Eq. 17.26 can also be computed making use of Eq. 17.31 as follows: Define $\boldsymbol{T}(s)$ as

$$\boldsymbol{T}(s) = \begin{bmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} \\ \boldsymbol{T}_{21} & \boldsymbol{T}_{22} \end{bmatrix} = e^{(\boldsymbol{D} + s\boldsymbol{E})d} \tag{17.32}$$

Using Eq. 17.32 and rewriting Eq. 17.31 in the form of Eq. 17.26, we get

$$\boldsymbol{P} = \begin{bmatrix} -\boldsymbol{T}_{11} & \boldsymbol{U} \\ -\boldsymbol{T}_{21} & 0 \end{bmatrix} \qquad \boldsymbol{Q} = \begin{bmatrix} -\boldsymbol{T}_{12} & 0 \\ -\boldsymbol{T}_{22} & \boldsymbol{U} \end{bmatrix} \tag{17.33}$$

Distributed vs. Lumped: Number of Lumped Segments Required

It is often of practical interest to switch between distributed models and lumped representations. In this case, it is necessary to know approximately how many lumped segments are required to approximate a distributed model. For the purpose of illustration, consider LC segments, which can be viewed as low-pass filters. For a reasonable approximation, this filter must pass at least some multiples of the highest frequency content f_{max} of the propagating signal (say, ten times, $f_0 \geq 10f_{max}$). In order to relate these,[2,3] we make use of the 3-dB passband of the LC filter given by

$$f_0 = \frac{1}{\pi\sqrt{LdCd}} = \frac{1}{\pi\tau d} \tag{17.34}$$

where d is the length of the line. From Eq. 17.1, we have $f_{max} = 0.35/t_r$ and using Eq. 17.34, we can express the relation $f_0 \geq 10f_{max}$ in terms of the delay of the line and the rise time as

$$\frac{1}{\pi\tau d} \geq 10 \times 0.35 / t_r \tag{17.35}$$

or

$$t_r \geq 3.5(\pi\tau d) \approx 10\tau d \tag{17.36}$$

In other words, delay allowed per segment is $0.1t_r$. Hence, the total number of segments (N) required is given by

$$N = \frac{10\tau d}{t_r} \tag{17.37}$$

In the case of RLC segments, in addition to satisfying Eq. 17.36, the series resistance of each segment must also be accounted for. The series resistance *Rd* representing the ohmic drop should not lead to impedance mismatch, which can result in excessive reflection within the segment.[2,3]

Example

Consider a digital signal with rise time of 0.2 ns propagating on a lossless wire of length 10 cm, with a per unit delay of 70.7 ns. This can be represented by a distributed model with per unit length parameters of L = 5 nH/cm and C = 1 pF/cm. If the same circuit were to be represented by lumped segments, one needs $N = ((10 \times 70.7e^{-12} \times 10)/(0.2e^{-9})) = 35$ sections. It is to be noted that using more sections does not clean up ripples completely, but helps reduce the first overshoot (Gibb's phenomenon). Ripples are reduced when some loss is taken into account.

17.4 Interconnect Simulation Issues

As pointed out earlier, simulation of interconnects is associated with two major bottlenecks: mixed frequency/time problem and the CPU expense.

Mixed Frequency/Time Problem

The major difficulty in simulating these high-frequency models lies in the fact that distributed/full-wave elements, while formulated in terms of partial differential equations, are best described in the frequency domain. Non-linear terminations, on the other hand, can only be given in the time domain. These simultaneous formulations cannot be handled by a traditional ordinary differential equation solver such as SPICE.[8,38]

CPU Expense

In general, interconnect networks consist of hundreds or thousands of components, such as resistors, capacitors, inductors, transmission lines, and other levels of interconnect models. At the terminations, there generally exist some nonlinear elements such as drivers and receivers. If only lumped RLC models are considered, ordinary differential equation solvers such as SPICE may be used for simulation purposes. However, the CPU cost may be large, owing to the fact that SPICE is mainly a non-linear simulator and it does not handle large RLC networks efficiently.

Background on Circuit Simulation

Prior to introducing interconnect simulation algorithms, it would be useful to have a glimpse at the basic circuit simulation techniques. Conventional circuit simulators are based on the simultaneous solution of linear equations which are obtained by applying Kirchoff's current law (KCL) to each node in the network. Either for frequency- or time-domain analysis, the first step is to set up the *modified nodal analysis matrix (MNA)*.[68] For example, consider the small circuit in Fig. 17.18. Its MNA equations are

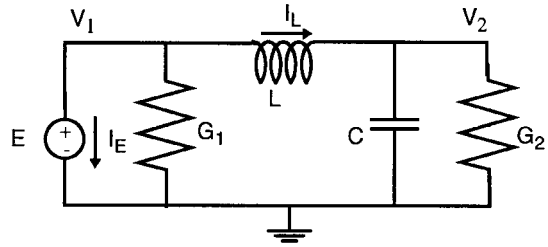


FIGURE 17.18 Example circuit for MNA formulation.

$$\left(\begin{bmatrix} G_1 & 0 & 1 & 1 \\ 0 & G_2 & -1 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} + s \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & C & 0 & 0 \\ 0 & 0 & -L & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right) \begin{bmatrix} V_1 \\ V_2 \\ I_L \\ I_E \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ E \end{bmatrix} \tag{17.38}$$

The above equation, representing a simple two-node circuit, has the same form as any other MNA matrix representing a large linear lumped network. Hence, MNA equations in general for lumped linear networks can be written as

$$(\boldsymbol{G} + s\boldsymbol{W})\boldsymbol{X}(s) - \boldsymbol{b} = \boldsymbol{0} \tag{17.39}$$

or, in the time domain, it can be written as

$$\boldsymbol{G}\boldsymbol{x}(t) + \boldsymbol{W}\dot{\boldsymbol{x}}(t) - \boldsymbol{b}(t) = \boldsymbol{0} \tag{17.40}$$

Discussion of CPU Cost in Conventional Simulation Techniques

Frequency-domain simulation is conventionally done by solving Eq. 17.39 at each frequency point through LU decomposition and forward-backward substitution. For time-domain simulation, linear multi-step techniques[69] are used. The most common of these integration formulas is the trapezoidal rule, which gives the following difference formula

$$\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \Delta t \left(\frac{\dot{\boldsymbol{x}}(t) + \dot{\boldsymbol{x}}(t + \Delta t)}{2} \right) \tag{17.41}$$

$$\left(\boldsymbol{G} + \frac{2}{\Delta t}\boldsymbol{W} \right) \boldsymbol{x}(t + \Delta t) = \left(\frac{2}{\Delta t}\boldsymbol{W} - \boldsymbol{G} \right) \boldsymbol{x}(t) + (\boldsymbol{b}(t) + \boldsymbol{b}(t + \Delta t)) \tag{17.42}$$

Note that the trapezoidal rule is an integration formula of order of 2, which is the highest possible order that could ensure absolute stability.[69] Due to such relatively low order, simulators are forced to use small step sizes to ensure the accuracy during transient simulation. Transient response computation requires the LU decomposition of Eq. 17.42 at every time step. It gets further complicated in the presence of nonlinear elements, in which case the Eq. 17.40 gets modified as

$$\boldsymbol{G}\boldsymbol{x}(t) + \boldsymbol{W}\dot{\boldsymbol{x}}(t) + \phi(\boldsymbol{x}(t)) - \boldsymbol{b}(t) = \boldsymbol{0} \tag{17.43}$$

where $\phi(\boldsymbol{x}(t))$ is a non-linear function of $\boldsymbol{x}$. The difference equation based on the trapezoidal rule therefore becomes

$$\left(\boldsymbol{G} + \frac{2}{\Delta t}\boldsymbol{W}\right)\boldsymbol{x}(t + \Delta t) + \phi(\boldsymbol{x}(t + \Delta t)) = \left(\frac{2}{\Delta t}\boldsymbol{W} - \boldsymbol{G}\right)\boldsymbol{x}(t) + (\boldsymbol{b}(t) + \boldsymbol{b}(t + \Delta t)) - \phi(\boldsymbol{x}(t)) \quad (17.44)$$

In the case of nonlinear elements, Newton iterations are used to solve Eq. 17.44, which requires two to three LU decompositions at each time step. This causes (note that W and G matrices for interconnect networks are usually very large) the CPU cost of a time-domain analysis to be very expensive. This led to the development of model-reduction algorithms, which effects a reduction in the order of the linear subnetwork before performing a non-linear analysis so as to yield fast simulation results.

Circuit Equations in the Presence of Distributed Elements

Now consider the general case in which the network ϕ also contains arbitrary linear subnetworks along with lumped components. The arbitrary linear subnetworks may contain lossy coupled transmission lines and also measured subnetworks. Let there be N_t lossy coupled transmission line sets, with n_a coupled conductors in the linear subnetwork a. Without loss of generality, the modified nodal admittance (MNA)[68,69] matrix for the network ϕ with an impulse input excitation can be formulated as

$$\boldsymbol{C}\frac{\partial}{\partial t}\boldsymbol{v}(t) + \boldsymbol{G}\boldsymbol{v}(t) + \sum_{a=1} \boldsymbol{D}_a \boldsymbol{i}_a(t) - \boldsymbol{b}\delta(t) = 0, \qquad t \in [0, T] \quad (17.45)$$

where:

- $\boldsymbol{D}_a = [d_{i,j}] \in \{0, 1\}$, where $i \in \{1, 2, \ldots, N_\phi\}$, $j \in \{1, 2, \ldots, 2n_a\}$ with a maximum of one non-zero in each row or column, is a selector matrix that maps $\boldsymbol{i}_a(t) \in \Re^{2n_a}$, the vector of terminal currents entering the transmission line subnetwork a, into the node space $\Re^N\phi$ of network ϕ.
- N_ϕ is the total number of variables in the MNA formulation.

Using the transmission line stencil given by Eq. 17.25 and taking the Laplace transform of Eq. 17.45 assuming zero initial conditions, one obtains

$$\begin{bmatrix} \boldsymbol{G} + s\boldsymbol{C} & \boldsymbol{D}_1 & \ldots & \boldsymbol{D}_{N_t} \\ \boldsymbol{P}_1\boldsymbol{D}_1^t & \boldsymbol{Q}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \ldots & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{P}_{N_t}\boldsymbol{D}_{N_t}^t & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Q}_{N_t} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}(s) \\ \boldsymbol{I}_1(s) \\ \ldots \\ \boldsymbol{I}_{N_t}(s) \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \\ \ldots \\ \boldsymbol{0} \end{bmatrix} \quad (17.46)$$

or

$$\boldsymbol{Y}(s)\boldsymbol{X}(s) = \boldsymbol{E} \quad (17.47)$$

$$\boldsymbol{X}(s) = [\boldsymbol{V}^t(s)\boldsymbol{I}_1^t(s), \boldsymbol{I}_2^t(s), \ldots, \boldsymbol{I}_{N_t}^t(s)]^t \quad (17.48)$$

$$\boldsymbol{E} = [\boldsymbol{b}^t\boldsymbol{0}^t \ldots \boldsymbol{0}^t] \quad (17.49)$$

Example

To illustrate the above formulation, consider the network shown in Fig. 17.19, which is a modified version of the previous example with introduction of a coupled transmission line. The network can be described by Eq. 17.46 as

$$\boldsymbol{G} + s\boldsymbol{C} = \begin{bmatrix} sC_1 + G_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & G_3 & 0 & -G_3 & 0 \\ 0 & 0 & 0 & sC_2 & 0 & 0 \\ 0 & 0 & -G_3 & 0 & G_3 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}; \qquad \boldsymbol{D}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{17.50}$$

$$\boldsymbol{V}(s) = [V_1 \; V_2 \; V_3 \; V_4 \; V_5 \; I_e]^t \tag{17.51}$$

$$\boldsymbol{I}_1(s) = [I_a \; I_b \; I_c \; I_d]^t \tag{17.52}$$

$$\boldsymbol{b} = [0 \; 0 \; 0 \; 0 \; 0 e]^t \tag{17.53}$$

Conventional simulation methods obtain the frequency response of a circuit by solving Eq. 17.46 using LU decomposition and forward-backward substitution at various frequency points (usually thousands of points are required to get an accurate response over a desired frequency range). However, moment-matching techniques such as AWE extract the poles and residues of Eq. 17.46 using one LU decomposition only. The transfer function of the network and its frequent response can then be deduced from poles and residues. In addition to speeding up the simulation, MMTs provide a convenient way to handle mixed frequency/time simulation problem through macromodeling.

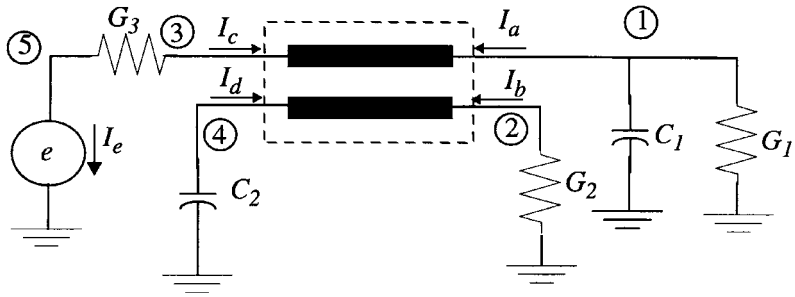


FIGURE 17.19 An example circuit containing transmission line components.

17.5 Interconnect Simulation Techniques

The main objective behind the interconnect simulation algorithms is to address the mixed frequency/time problem as well as ability to handle large linear circuits without too much CPU expense. There have been several algorithms proposed for this purpose, and they are discussed below.

Method of Characteristics

The method of characteristics transforms partial differential equations of a transmission line into ordinary differential equations.[28,29] Extensions to this method to allow it to handle lossy transmission

lines can also be found in the literature; for example, the iterative waveform relaxation techniques (IWR).[28,29] The method of characteristics is still used as one of the most practical techniques for simulation of lossless lines.

An analytical solution for Eqs. 17.21 and 17.22 was derived in Ref. 28 for two conductor lines which provides the Y parameters of the two-port transmission line network

$$\boldsymbol{I} = \boldsymbol{YV}; \qquad \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \frac{1}{Z_0(1-e^{-2\gamma l})} \begin{bmatrix} 1+e^{-2\gamma l} & -2e^{-\gamma l} \\ -2e^{-\gamma l} & 1+e^{-2\gamma l} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{17.54}$$

where γ is the propagation constant, and Z_0 is the characteristic impedance. V_1 and I_1 are the terminal voltage and current at the near end of the line, V_2 and I_2 are the terminal voltage and current at the far end of the line. The Y parameters of the transmission line are complex functions of s, and in most cases cannot be directly transformed into an ordinary differential equation in the time domain. The method of characteristics succeeded in doing such a transformation, but only for lossless transmission lines. Although this method was originally developed in the time domain using what was referred to as characteristic curves (hence, the name), a short alternative derivation in the frequency domain will be presented here. The frequency-domain approach gives more insight as to the limitations of this technique and possible solutions to those limitations.

By rearranging the terms in Eq. 17.54, we can write

$$\begin{aligned} V_1 &= Z_0 I_1 + W_{c1} \\ V_2 &= Z_0 I_2 + W_{c2} \end{aligned} \tag{17.55}$$

where W_{c1} and W_{c2} are defined as

$$\begin{aligned} W_{c1} &= e^{-\gamma l}[2V_2 - e^{-\gamma l}(Z_0 I_1 + V_1)] \\ W_{c2} &= e^{-\gamma l}[2V_1 - e^{-\gamma l}(Z_0 I_2 + V_2)] \end{aligned} \tag{17.56}$$

Using Eqs. 17.55 and 17.56, a recursive relation for W_{c1} and W_{c2} can be obtained as

$$\begin{aligned} W_{c1} &= e^{-\gamma l}[2V_2 - W_{c2}] \\ W_{c2} &= e^{-\gamma l}[2V_1 - W_{c1}] \end{aligned} \tag{17.57}$$

A lumped model of the transmission line can then be deduced from Eqs. 17.55 and 17.56, as shown in Fig. 17.20. If the lines were lossless (in which case the propagation constant is purely imaginary; $\gamma = j\beta$), the frequency-domain expression (Eq. 17.57) can be analytically converted into time domain using inverse Laplace transform as

$$\begin{aligned} w_{c1}(t+\tau) &= 2v_2(t) - w_{c2}(t) \\ w_{c2}(t+\tau) &= 2v_1(t) - w_{c1}(t) \end{aligned} \tag{17.58}$$

where $e^{-j\beta}$ is replaced by a time shift (or delay). Each transmission line can therefore be modeled by two impedances and two voltage-controlled voltage sources with time delay. While this transmission line model is in the time domain and can be easily linked to time-domain circuit simulation, the time shift affects the stability of the integration formula and causes the step size to significantly decrease, therefore increasing the CPU time. For lossy lines, the propagation constant is not purely imaginary and can therefore not be replaced by a pure delay. In that case, analytical expressions for w_{c1} and w_{c2} cannot be

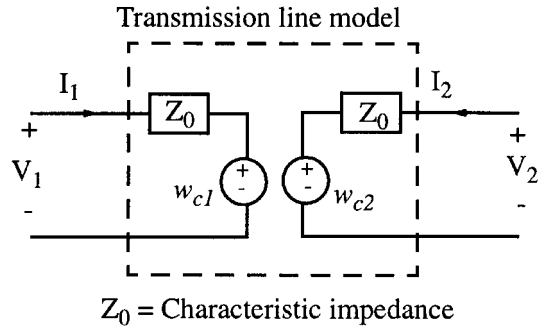


FIGURE 17.20 Macromodel using method of characteristics.

found in the time domain, although some numerical techniques were proposed (e.g., the iterative waveform relaxation techniques (IWR)).[28]

Recently, there have been several publications based on approximating the frequency characteristics of transmission-line equations using rational polynomials.[27] Analytical techniques to directly convert partial differential equations into time-domain macromodels based on Padé rational approximations of exponential matrices have also been reported.[26]

Moment-Matching Techniques

Interconnect networks generally tend to have large number of poles, spread over a wide frequency range. Although the majority of these poles would normally have very little effect on simulation results, they make the simulation CPU extensive by forcing the simulator to take smaller step sizes.

Dominant Poles

Dominant poles are those that are close to the imaginary axis and significantly influence the time as well as the frequency characteristics of the system. The moment-matching techniques (MMTs)[34–67] capitalize on the fact that, irrespective of the presence of large number of poles in a system, only the dominant poles are sufficient to accurately characterize a given system. This effect is demonstrated in Fig. 17.21, where it is clear that pole P_2 will have little effect on the final transient result.

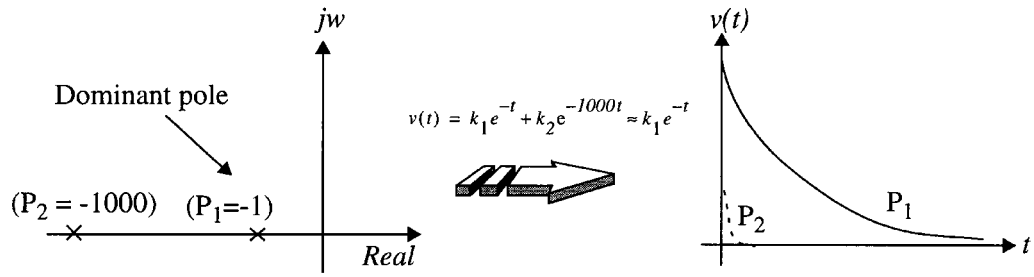


FIGURE 17.21 Summary of the steps involved in the MMT algorithm.

A brief mathematical description of the underlying concepts of moment-matching techniques is given below. Consider a single input/single output system and let $H(s)$ be the transfer function. $H(s)$ can be represented in a rational form as

$$H(s) = \frac{P(s)}{Q(s)} \tag{17.59}$$

where $P(s)$ and $Q(s)$ are polynomials in s. Equivalently, Eq. 17.59 can be written as

$$H(s) = c + \sum_{i=0}^{N_p} \frac{k_i}{s - p_i} \tag{17.60}$$

where P_i and k_i are the ith pole-residue pair, N_p is the total number of system poles, and c is the direct coupling constant. Next, the time-domain impulse response can be computed in a closed form using inverse Laplace transform as

$$h(t) = c\delta t + \sum_{i=0}^{N_p} k_i e^{p_i t} \tag{17.61}$$

In the case of large networks, N_p, the total number of poles can be of the order of thousands. Generating all the N_p poles will be highly CPU intensive even for a small network; and for large networks, it is completely impractical. MMTs address the above issue by deriving a reduced-order approximation $\hat{H}(s)$ in terms of dominant poles, instead of trying to compute all the poles of a system. Assuming that only L dominant poles were extracted which give a reasonably good approximation to the original system, Eq. 17.59 and the corresponding approximate frequency and time responses can be written as

$$H(s) \approx \hat{H}(s) = \frac{\hat{P}(s)}{\hat{Q}(s)} = \hat{c} + \sum_{i=0}^{L} \frac{\hat{k}_i}{s - \hat{p}_i} \tag{17.62}$$

$$h(t) \approx \hat{h}(t) = \hat{c}\delta t + \sum_{i=0}^{L} \hat{k}_i e^{\hat{p}_i t} \tag{17.63}$$

MMTs are based on the Padé approximation, and the steps involved in generating a reduced-order model using Padé approximation are outlined below.

Padé Approximations

Consider a system-transfer function $H(s)$ which is approximated by a rational function $\hat{H}(s)$ as

$$H(s) \approx \hat{H}(s) = \frac{a_0 + a_1 s + a_2 s^L + \ldots + a_L s^L}{1 + b_1 s + \ldots + b_M s^M} = \frac{P_L(s)}{Q_M(s)} \tag{17.64}$$

where $a_0, \ldots, a_L, b_1, \ldots, b_M$ are the unknowns (total of $L + M$ variables). Next, expanding $H(s)$ using Taylor series coefficients M_i (moments), we have

$$H(s) \approx \hat{H}(s) = m_0 + m_1 s + m_2 s^2 + \ldots + m_n s^n = \sum_{i=0}^{n} s^i m_i \tag{17.65}$$

The series in Eq. 17.65 is then matched to the lower-order rational polynomial given by Eq. 17.64; hence, the name moment-matching techniques (MMTs), also known as Padé approximation). There are $L + M$ unknowns and hence we need to match only the first $L + M$ moments as

$$\frac{a_0 + a_1 s + a_2 s^L + \ldots + a_L s^L}{1 + b_1 s + \ldots + b_M s^M} = m_0 + m_1 s + m_2 s^2 + \ldots + m_{(L+M)} s^{L+M} \tag{17.66}$$

Cross-multiplying and equating the powers of s starting from s^{L+1} to s^{L+M} on both sides of Eq. 17.66, we can evaluate the denominator polynomial coefficients as

$$\begin{bmatrix} m_{L-M+1} & m_{L-M+2} & \ldots & m_L \\ m_{L-M+2} & \ldots & \ldots & m_{L+1} \\ \ldots & \ldots & \ldots & \ldots \\ m_L & m_{L+1} & \ldots & m_{L+M-1} \end{bmatrix} \begin{bmatrix} b_M \\ b_{M-2} \\ \ldots \\ b_1 \end{bmatrix} = - \begin{bmatrix} m_{L+1} \\ m_{L+2} \\ \ldots \\ m_{L+M} \end{bmatrix} \tag{17.67}$$

The numerator coefficients can then be found by equating the remaining powers of s, starting from s^0 to s^L as

$$\begin{aligned} a_0 &= m_0 \\ a_1 &= m_1 + b_1 m_0 \\ &\ldots \\ a_L &= m_L + \sum_{i=1}^{min(L,M)} b_i m_{L-i} \end{aligned} \tag{17.68}$$

Equations 17.67 and 17.68 yield an approximate transfer function in terms of rational polynomials. Alternatively, an equivalent pole-residue model can be found as follows. Poles p_i are obtained by applying a root-solving algorithm on denominator polynomial $\hat{Q}(s)$. In order to obtain k_i, expand the approximate transfer function given by Eq. 17.62 using Maclaurin series as

$$\hat{H}(s) = \hat{c} - \sum_{n=0}^{\infty} s^n \left(\sum_{i=0}^{\tilde{} } \frac{\hat{k}_i}{\hat{p}_i^{n+1}} \right) \tag{17.69}$$

Comparing $\hat{H}(s)$ from Eqs. 17.65 and 17.69, we note that

$$\begin{aligned} m_0 &= \hat{c} - \sum_{i=0}^{L} \frac{\hat{k}_i}{\hat{p}_i} \\ &\ldots \quad (0 < i < 2L) \\ m_i &= -\sum_{i=0}^{L} \frac{\hat{k}_i}{\hat{p}_i^{i+1}} \end{aligned} \tag{17.70}$$

Residues can be evaluated by writing the equations in Eq. 17.70 in a matrix form as

$$\begin{bmatrix} \hat{p}_1^{-1} & \hat{p}_2^{-1} & & \hat{p}_L^{-1} & -1 \\ \hat{p}_1^{-2} & \hat{p}_2^{-2} & & \hat{p}_L^{-2} & 0 \\ & & \ldots & & \\ & & \ldots & & \\ \hat{p}_1^{-L-1} & \hat{p}_2^{-L-1} & & \hat{p}_L^{-L-1} & 0 \end{bmatrix} \begin{bmatrix} \hat{k}_1 \\ \hat{k}_2 \\ \ldots \\ \hat{k}_L \\ \hat{c} \end{bmatrix} = - \begin{bmatrix} M_0 \\ M_1 \\ \ldots \\ M_{L-1} \\ M_L \end{bmatrix} \tag{17.71}$$

In the above equations, $\hat{c}$ represents the direct coupling between input and output. There are more exact ways to compute $\hat{c}$ which are not detailed here; interested readers can refer to Ref. 35.

Computation of Moments

Having outlined the concept of MMTs, to proceed further, we need to evaluate the moments of the system. Consider the circuit equation represented by Eq. 17.46 and expand the response vector $\boldsymbol{X}(s)$ using Taylor series as

$$\boldsymbol{X}(s) = \boldsymbol{M_0} + \boldsymbol{M_1}s + \boldsymbol{M_2}s^2 + \ldots + \boldsymbol{M_n}s^n = \sum_{i=0}^{n} s^i \boldsymbol{M_i} \tag{17.72}$$

where $\boldsymbol{M_i}$ represents the ith moment-vector. For the purpose of illustration, consider the simple case of network with only lumped models. In this case, the network can be represented in the form Eq. 17.39 as

$$[\boldsymbol{G} + s\boldsymbol{C}][\boldsymbol{X}(s)] = [\boldsymbol{E}] \tag{17.73}$$

or

$$[\boldsymbol{G} + s\boldsymbol{C}][\boldsymbol{M_0} + \boldsymbol{M_1}s + \boldsymbol{M_2}s^2 + \ldots] = [\boldsymbol{E}] \tag{17.74}$$

Equating powers of s on both sides of Eq. 17.74, we obtain the following relationships

$$\begin{aligned} \boldsymbol{G}\boldsymbol{M_0} &= \boldsymbol{E} \\ \boldsymbol{G}\boldsymbol{M_i} &= -\boldsymbol{C}\boldsymbol{M_{i-1}} \qquad i > 0 \end{aligned} \tag{17.75}$$

The above equations give a closed form relationship for the computation of moments. The moments of a particular output node of interest (which are represented by m_i in Eqs. 17.65 to 17.71), are picked from moment-vectors $\boldsymbol{M_i}$. As seen, Eq. 17.75 requires only one LU decomposition and few forward-backward substitutions during the recursive computation of higher-order moments. Since the major cost involved in circuit simulation is due to LU decomposition, MMTs yield very high speed advantage (100 to 1000 times) compared to conventional simulators.

Generalized Computation of Moments

In the case of networks containing transmission lines and measured subnetworks, moment computation is not straightforward. A generalized relation for recursive computation of higher-order moments at an expansion point $s = \alpha$ can be derived[44] using Eq. 17.46 as:

$$\begin{aligned} [\Psi(\alpha)]\boldsymbol{M}_0 &= E \\ [\Psi(\alpha)]\boldsymbol{M}_n &= -\sum_{r=1}^{n} \frac{(\Psi^{(r)}\big|_{s=\alpha})\boldsymbol{M}_{n-r}}{r!} \end{aligned} \tag{17.76}$$

where the superscript r denotes the rth derivative at $s = \alpha$. It can be seen that the coefficient on the left-hand side does not change during higher-order moment computation and, hence, only one LU decomposition would suffice. It is also noted that the lumped networks are a special case of Eq. 18.76 (where $\Psi^{(r)} = 0$ for $r \geq 2$, in which case Eq. 17.76 reduces to the form given by Eq. 17.75).

Having obtained a recursive relationship Eq. 17.76 for higher-order moments, in order to proceed further, we need the derivatives of (Ψ). The derivatives $\boldsymbol{A_d}^{(r)}$ and $\boldsymbol{B_d}^{(r)}$ corresponding to transmission lines can be computed using the matrix exponential-based method.[34,38] Efficient techniques for computation of moments of transmission lines with frequency-dependent parameters,[56] full-wave,[12] and measured subnetworks[63,65] can also be found in the literature.

Computation of Time-Domain Macromodel

Once a pole-residue model describing the interconnect network is obtained, a time-domain realization in the form of state-space equations can be obtained as[39–41,75,76]:

$$\begin{aligned} \frac{d}{dt}[\boldsymbol{z}_\pi(t)] - [\boldsymbol{A}_\pi][\boldsymbol{z}_\pi(t)] - [\boldsymbol{B}_\pi][\boldsymbol{i}_\pi(t)] &= \mathbf{0} \\ [\boldsymbol{v}_\pi(t)] - [\boldsymbol{C}_\pi][\boldsymbol{z}_\pi(t)] + [\boldsymbol{D}_\pi][\boldsymbol{i}_\pi(t)] &= \mathbf{0} \end{aligned} \tag{17.77}$$

where i_π and v_π are the vectors of terminal currents and voltages of the linear subnetwork π. Using standard non-linear solvers or any of the general-purpose circuit simulators, the unified set of differential equations represented by Eq. 17.77 can be solved to yield unified transient solutions for the entire non-linear circuit consisting of high-frequency interconnect subnetworks. For those simulators (such as HSPICE) that do not directly accept the differential equations as input, the macromodel represented by Eq. 17.77 can be converted to an equivalent subcircuit, and is described in the next section. Figure 17.22 summarizes the computational steps involved in the MMT algorithm.

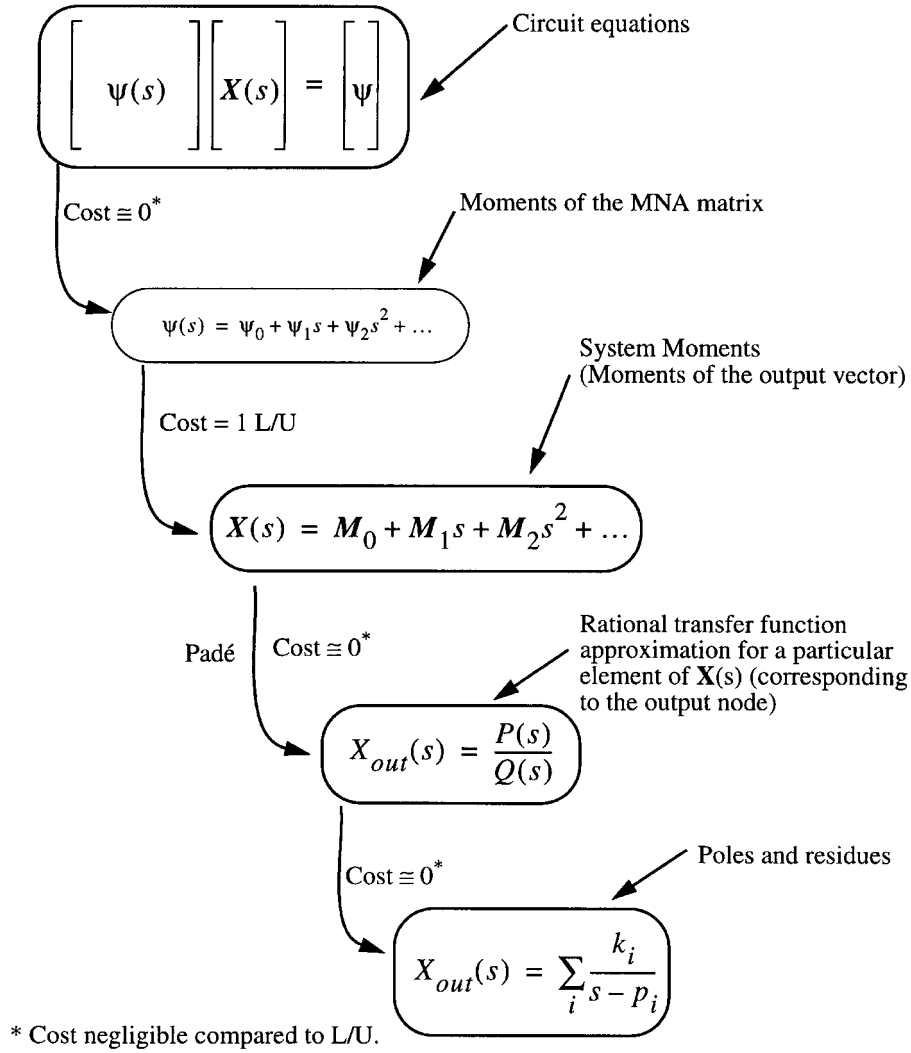


FIGURE 17.22 Summary of the steps involved in the MMT algorithm.

Limitations of Single-Expansion MMT Algorithms

Obtaining a lower-order approximation of the network transfer function using a single Padé expansion is commonly referred as *asymptotic waveform evaluation (AWE)* in the literature. However, due to the inherent limitations of Padé approximants, MMTs based on single expansion often give inaccurate results. The following is a list of those properties that have the most impact on MMTs.

- The matrix in Eq. 17.67 (which is known as Toeplitz matrix) is ill-conditioned if its size is large. This implies that we can only expect to detect six to eight accurate poles from a single expansion.
- Padé often produces unstable poles on the right-hand side of the complex plane.
- Padé accuracy deteriorates as we move away from the expansion point.
- Padé provides no estimates for error bounds.

Generally, AWE gives accurate results for RC networks, but often fails for non-RC networks. This is due to the fact that the poles of an RC network are all on the real axis and therefore an expansion at the origin could clearly determine which ones are dominant and which ones are not. However, in the case of general RLC networks, it is possible to have some of the dominant poles outside the radius of convergence of the Padé expansion. In systems containing distributed elements, the number of dominant poles will be significantly higher, and it is very difficult to capture all with a single Padé expansion.

In addition, there is no guarantee that the reduced-model obtained as above is passive. Passivity implies that a network cannot generate more energy than it absorbs, and no passive termination of the network will cause the system to go unstable.[70–73] The loss of passivity can be a serious problem because transient simulations of reduced networks may encounter artificial oscillations.

Recent Advances in Moment-Matching Techniques

In order to address the above difficulties, recent research in the circuit simulation area has focused on arriving at compact, accurate, as well as passive macromodels for high-speed interconnects. The problem of accuracy is addressed using multi-point expansion techniques such as *complex frequency hopping (CFH)*.[44–46] Also, in order to enhance the accuracy range of an approximation at a given expansion and to reduce the number of hops, several techniques based on Krylov-space formulations are developed.[47–55] Also, efficient schemes based on congruent transformation for preservation of passivity during the reduction of interconnect networks is available in the literature.[47–55] For further readings, interested readers can look at the recent proceedings of ICCAD, DAC, or IEEE transactions on computer-aided design of ICs (T-CAD), circuits and systems (T-CAS), and microwave theory and techniques (T-MTT).

Acknowledgments

The authors wish to thank and acknowledge the help and contributions provided by Roni Khazaka of Carlton University while preparing the section about frequency-dependent parameters.

References

1. H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, Reading, MA, 1990.
2. H. W. Johnson and M. Graham, *High-speed Digital Design*, Prentice-Hall, Englewood Cliffs, NJ, 1993.
3. R. K. Poon, *Computer Circuits Electrical Design*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
4. W. W. M. Dai (Guest Editor), "Special issue on simulation, modeling, and electrical design of high-speed and high-density interconnects," *IEEE Transactions on Circuits and Systems*, vol. 39, no. 11, pp. 857-982, Nov. 1992.
5. M. Nakhla, R. Achar , and R. Khazaka, *Modeling and Simulation of High-Speed VLSI Interconnects*, Chapter IV: Circuits And Systems In The Information Age, IEEE Publishers, NJ, pp. 187-215, 1997.

6. A. Deustsch, "Electrical characteristics of interconnections for high-performance systems," *Proceedings of the IEEE,* vol. 86, no. 2, pp. 315-355, Feb. 1998.
7. T. L. Quarles, *The SPICE3 Implementation Guide*, Technical Report, ERL-M89/44, University of California, Berkeley, 1989.
8. C. Paul, *Analysis of Multiconductor Transmission Lines*, John Wiley & Sons, New York, 1994.
9. C. Paul, *Introduction to Electromagnetic Compatibility*, John Wiley & Sons, New York, 92.
10. S. Gao, A. Yang, and S. Kang, "Modeling and simulation of interconnection delays and cross talks in high-speed integrated circuits," *IEEE Trans. on Circuits and Systems*, pp. 1-9, Jan. 90.
11. H. Hasegawa and S. Seki, "Analysis of interconnection delay on very high-speed LSI/VLSI chips using a microstrip line model," *IEEE Trans. Electron Devices*, pp. 1954-1960, Dec. 1984.
12. R. Achar, M. Nakhla, and Q. J. Zhang, "Full-wave analysis of high-speed interconnects using complex frequency hopping," *IEEE Trans. on Computer-Aided Design*, pp. 997-1016, Oct. 98.
13. T. Itoh and R. Mittra, "Spectral domain approach for calculating the dispersion characteristics of microstrip lines," *IEEE Trans. Microwave Theory Tech.*, pp. 496-499, Feb. 1973.
14. D. Mirshekar-Syahkal, *Spectral Domain Method for Microwave Integrated Circuits*, Joinery & Sons, Inc., 1990.
15. R. H. Jansen, "Spectral Domain Approach for microwave integrated circuits," *IEEE Trans. Microwave Theory Tech.,* vol. MTT-33, pp. 1043-1056, Feb. 1985.
16. R. Wang, and O. Wing, "A circuit model of a system of VLSI interconnects for time response computation," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-39, pp. 688-693, April 1991.
17. M. A. Kolbehdari, M. Srinivasan, M. Nakhla, Q. J. Zhang, and R. Achar, "Simultaneous time and frequency domain solution of EM problems using finite element and CFH techniques," *IEEE Trans on Microwave Theory and Techniques*, vol. 44, pp. 1526-1534, Sept. 1996.
18. A. E. Ruehli, "Equivalent circuit models for three dimensional multiconductor systems," *IEEE Trans. Microwave Theory Tech.*, vol. 22, no. 3, pp. 216-224, Mar. 1974.
19. A. E. Ruehli and H. Heeb, "Circuit models for three dimensional geometries including dielectrics," *IEEE Trans. Microwave Theory Tech.*, pp. 1507-1516, Mar. 1992.
20. A. R. Djordjevié and T. K. Sarkar, "Closed-form formulas for frequency-dependent resistance and inductance per unit length of microstrip and strip transmission lines," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-42, pp. 241-248, Feb. 1994.
21. A. R. Djordjevié, R. F. Harrington, T. K. Sarkar, and M. Bazdar, *Matrix Parameters for Multiconductor Transmission Lines: Software and Users Manual*, Reteach House, Boston, 1989.
22. R. Achar and M. Nakhla, "Efficient transient simulation of embedded subnetworks characterized by S-parameters in the presence of nonlinear elements," *IEEE Transactions on Microwave Theory and Techniques*, vol. 46, pp. 2356-2363, Dec. 1998.
23. Y. Tsuei, A. C. Cangellaris, and J. L. Prince, "Rigorous electromagnetic modeling of chip-to-package (first-level) interconnections," *IEEE Trans. Components Hybrids Manufacturing Technology*, vol. 16, no. 8, pp. 876-883, Aug. 1993.
24. M. Picket-May, A. Taflove, and J. Baron, "FD-TD modeling of digital signal propagation in 3-D circuits with passive and active loads," *IEEE Trans. Microwave Theory Tech.*, vol. 42, no. 8, pp. 1514-1523, Aug. 1994.
25. T. Dhane and D. D. Zutter, "Selection of lumped element models for coupled lossy transmission lines," *IEEE Trans. Computer-Aided Design*, vol. 11, July 1992.
26. X. Li, M. Nakhla, and R. Achar, "A universal closed-loop high-speed interconnect model for general purpose circuit simulators," *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Monterey, CA, pp. 66-69, June, 1998.
27. M. Celik and A. C. Cangellaris, "Simulation of dispersive multiconductor transmission lines by Padé approximation via Lanczos process," *IEEE Trans. MTT*, pp. 2525-2535, Dec. 96.
28. F. Y. Chang, "The generalized method of characteristics for waveform relaxation analysis of lossy coupled transmission lines," *IEEE Trans. Microwave Theory Tech.*, vol. 37, pp. 2028-2038, Dec. 1989.

29. N. Orhanovic, P. Wang, and V. K. Tripathi, "Generalized method of characteristics for time domain simulation of multiconductor lossy transmission lines," *Proceedings. IEEE Symposium on Circuits and Systems*, May 1990.
30. E. C. Chang and S. M. Kang, "Computationally efficient simulation of a lossy transmission line with skin effect by using numerical inversion of Laplace transform," *IEEE Transactions on Circuits and Systems*, vol. 39, pp. 861-868, July 1992.
31. R. Griffith and M. Nakhla, "Mixed frequency/time domain analysis on nonlinear circuits," *IEEE Trans. Computer-Aided Design*, vol. 10, no. 8, pp. 1032-1043, Aug. 1992.
32. R. Wang and O. Wing, "Transient analysis of dispersive VLSI interconnects terminated in nonlinear loads," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 1258-1277, Oct. 1992.
33. S. Lin and E. S. Kuh, "Transient simulation of lossy interconnects based on the recursive convolution formulation," *IEEE Trans on Circuits and Systems,* vol. 39, no. 11, pp. 879-892.
34. L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 352-366, Apr. 1990.
35. E. Chiprout and M. Nakhla, *Asymptotic Waveform Evaluation and Moment Matching for Interconnect Analysis*, Kluwer Academic Publishers, Boston, 1993.
36. S. Kumashiro, R. A. Rohrer, and A. J. Strojwas, "Asymptotic waveform evaluation for transient analysis of 3-D interconnect structures," *IEEE Trans. Computer-Aided Design*, vol. 12, no. 7, pp. 988-996, 1993.
37. X. Huang, "Padé Approximation of linear(ized) circuit responses," Ph.D. dissertation, Carnegie Mellon Univ., Nov. 1990.
38. T. Tang and M. Nakhla, "Analysis of high-speed VLSI interconnect using asymptotic waveform evaluation technique," *IEEE Trans. Computer-Aided Design*, pp. 2107-2116, Mar. 92.
39. D. Xie and M. Nakhla, "Delay and crosstalk simulation of high speed VLSI interconnects with nonlinear terminations," *IEEE Trans. Computer-Aided Design*, pp. 1798-1811, Nov. 1993.
40. R. Achar and M. Nakhla, *Minimum Realization of Reduced-Order Models of High-Speed Interconnect Macromodels*, Chapter: *Signal Propagation on Interconnects*, Kluwer Academic Publishers, Boston, 1998.
41. R, Achar and M. Nakhla, "A novel technique for minimum-order macromodel synthesis of high-speed interconnect subnetworks," *Proc. EEE International Symposium on Circuits and Systems (ISCAS)*, Monterey, CA, pp. 70-73, June 1998.
42. R. Griffith, E. Chiprout, Q. J. Zhang, and M. Nakhla, "A CAD framework for simulation and optimization of high-speed VLSI interconnections," *IEEE Transactions on Circuits and Systems*, vol. 39, no. 1, pp. 893-906.
43. J. E. Bracken, V. Raghavan, and R. A. Rohrer, "Interconnect simulation with asymptotic waveform evaluation (AWE)," *IEEE Transactions on Circuits and Systems*, pp. 869-878, Nov. 1992.
44. E. Chiprout and M. Nakhla, "Analysis of interconnect networks using complex frequency hopping," *IEEE Trans. Computer-Aided Design*, vol. 14, pp.186-199, Feb. 1995.
45. R. Sanaie, E. Chiprout, M. Nakhla, and Q. J. Zhang, "A fast method for frequency and time domain simulation of high-speed VLSI interconnects," *IEEE Trans. Microwave Theory Tech.*, vol. 42, no. 12, pp. 2562-2571, Dec. 1994.
46. R. Achar, M. Nakhla and E. Chiprout, "Block CFH: A model-reduction technique for distributed interconnect networks," *Proc. IEEE European Conference on Circuits Theory and Design (ECCTD)*, pp. 396-401, Sept. 1997, Budapest, Hungary.
47. P. Feldmann and R. W. Freund, "Efficient linear circuit analysis by Padé via Lanczos process," *IEEE Transactions on. Computer-Aided Design*, vol. 14, pp. 639-649, May 1995.
48. P. Feldmann and R. W. Freund, "Reduced order modeling of large linear subcircuits via a block Lanczos algorithm," in *Proc. Design Automation Conf.*, pp. 474-479, June 1995.
49. M. Silveria, M. Kamon, I. Elfadel, and J. White, "A coordinate-transformed Arnoldi algorithm for generating guaranteed stable reduced-order models of arbitrary RLC circuits," in *Proc. IEEE ICCAD*, Nov. 96.

50. M. Chou and J. White, "Efficient reduced order modeling for the transient simulation of three dimensional interconnect," in *Proc. ICCAD*, pp. 40-44, Nov. 1995.
51. K. J. Kerns and A. T. Yang, "Preservation of passivity during RLC network reduction via split congruence transformations," *IEEE Trans on. Computer-Aided Design*, pp. 582-591, July 1998.
52. A. Odabasioglu, M. Celik and L. T. Pillage, "PRIMA: Passive Reduced-Order Interconnect Macromodeling Algorithm," *Proceedings IEEE ICCAD,* pp. 58-65, Nov. 1997.
53. I. M. Elfadel and D. D. Ling, "A block rational Arnoldi algorithm for multiport passive model-order reduction of multiport RLC networks," *Proc. of ICCAD-97*, pp. 66-71, Nov. 1997.
54. Q. Yu, J. M. L. Wang, and E. S. Kuh, "Multipoint moment-matching model for multiport distributed interconnect networks," *Proc. of ICCAD-98*, pp. 85-90, Nov. 1998.
55. P. K. Gunupudi, M. Nakhla, and R. Achar, "Efficient simulation of high-speed interconnects using Krylov-space techniques," *Proceedings IEEE 7th EPEP*, New York, pp. 292-296, Oct. 1998.
56. R. Khazaka, E. Chiprout, M. Nakhla, and Q. J. Zhang, "Analysis of high-speed interconnects with frequency dependent parameters," *Proc. Intl. Symp. EMC,* pp.203-208, Zurich, March 1995.
57. R. Khazaka and M. Nakhla, "Analysis of high-speed interconnects in the presence of electromagnetic interference," *IEEE Trans. MTT*, vol. 46, pp. 940-947, July 1998.
58. I. Erdin, R. Khazaka, and M. Nakhla, "Simulation of high-speed interconnects in the presence of incident field," *IEEE Trans. MTT*, vol. 46, pp. 2251-2257, Dec. 1998.
59. S. D. Corey and A. T. Yang, "Interconnect characterization using time-domain reflectometry," *IEEE Trans. Microwave Theory Tech.*, vol. 43, pp. 2151-2156, Sep. 95.
60. B. J. Cooke, J. L. Prince, and A. C. Cangellaris "S-parameter analysis of multiconductor integrated circuit interconnect systems," *IEEE Trans. Computer-Aided Design*, pp. 353-360, Mar. 1992.
61. J. E. Schutt-Aine and R. Mittra, "Scattering parameter transient analysis of transmission lines loaded with nonlinear terminations," *IEEE Trans. Microwave Theory Tech.*, pp. 529-536, 1988.
62. P. C. Cherry and M. F. Iskander, "FDTD analysis of high frequency electronic interconnection effects," *IEEE Trans. Microwave Theory Tech.*, vol. 43, no. 10, pp. 2445-2451, Oct. 1995.
63. M. Celik, A. C. Cangellaris, and A. Deutsch, "A new moment generation technique for interconnects characterized by measured or calculated S-parameters," *IEEE Intl. Microwave Symposium Digest*, pp. 196-201, June 1996.
64. W. T. Beyene and J. E. Schutt-Aine, "Efficient trenasient simulation of high-speed interconnects characterized by sampled data," *IEEE Transactions on CPMT*, Part B, vol. 21, pp. 105-113, Feb. 1998.
65. G. Zheng, Q. J. Zhang, M. Nakhla, and R. Achar "An efficient approach for simulation of measured subnetworks with complex frequency hopping," *Proceedings IEEE/ACM In. Conf. Computer Aided Design*, San Jose, CA, pp. 23-26, Nov. 1996.
66. G. A. Baker Jr., *Essential of Padé Approximants*, Academic, New York, 1975.
67. J. H. McCabe, "A formal extension of the Padé table to include two point Padé quotients," *J. Inst. Math. Applic.*, vol. 15, pp. 363-372, 1975.
68. C. W. Ho, A. E. Ruehli, and P. A. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits and Systems*, vol. CAS-22, pp. 504-509, June 1975.
69. J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand Reinhold, New York, 1983.
70. L. Weinberg, *Network Analysis and Synthesis*, McGraw-Hill, New York, 1962.
71. E. Kuh and R. Rohrer, *Theory of Active Linear Networks*, Holden-day, San Francisco, CA, 67.
72. E. A. Guillemin, *Synthesis of Passive Networks*, John Wiley & Sons, New York, 1957.
73. M. E. V. Valkenburg, *Introduction to Modern Network Synthesis*, John Wiley & Sons, New York, 1960.
74. J. W. Demmel, *Applied Numerical Linear Algebra,* SIAM Publishers, Philadelphia, PA, 1997.
75. T. Kailath, *Linear Systems*. Prentice-Hall, Toronto, 1980.
76. C. T. Chen, *Linear System Theory and Design*. Holt, Rinehart and Winston, New York, 1984

Pedram, M. "Power Simulation and Estimation in VLSI Circuits"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

18

Power Simulation and Estimation in VLSI Circuits

Massoud Pedram
University of Southern California

18.1 Introduction

In the past, the major concerns of the designer were area, speed, and cost. Power consideration was typically of secondary importance. In recent years, however, this has begun to change and, increasingly, power is being given comparable weight to other design considerations. Several factors have contributed to this trend. Perhaps the primary driving factor has been the remarkable success and growth of the class of personal computing devices (portable desktops, audio- and video-based multimedia products) and wireless communications systems (personal digital assistants and personal communicators), which demand high-speed computation and complex functionality with low power consumption. There also exists a strong pressure for producers of high-end products to reduce their power consumption to reduce the packaging and cooling costs and improve product reliability.

When the target is a low-power application, the search for the optimal solution must include, at each level of abstraction, a "design improvement loop." In such a loop, a power analyzer/estimator ranks the various design, synthesis, and optimization options and thus helps in selecting the one that is potentially more effective from the power standpoint. Obviously, collecting the feedback on the impact of the different choices on a level-by-level basis, instead of just at the very end of the flow (i.e., at the gate level), enables a shorter development time. On the other hand, this paradigm requires the availability of power simulators and estimators, as well as synthesis and optimization tools, that provide accurate and reliable results at various levels of abstraction.

It was pointed out that the availability of level-by-level power analysis and estimation tools that are able to provide fast and accurate results is key for increasing the effectiveness of automatic design frameworks organized. We start this chapter, with a concise description of techniques for software-level estimation (Section 18.2). We then move to the behavioral level (Section 18.3), where we discuss existing power estimation approaches that rely on information-theoretic, complexity-based, and synthesis-based models. Finally, we focus our attention on designs described at the RT-level (Section 18.4). This is the area where most of the research activity on power modeling and estimation has been concentrated in recent years; we cover two of the most investigated classes of methods: namely, regression-based models and sampling-based models. Finally, we move to the gate-level power estimation (Section 18.5), where we discuss existing dynamic power estimation approaches that rely on statistical sampling and probabilistic compaction, as well as probability-based signal propagation schemes.

This chapter may be supplemented with other surveys on the topic, including Refs. 1 through 4.

18.2 Software-Level Power Estimation

The first task in the estimation of power consumption of a digital system is to identify the typical application programs that will be executed on the system. A non-trivial application program consumes millions of machine cycles, making it nearly impossible to perform power estimation using the complete program at, say, the RT-level. Most of the reported results are based on *power macro-modeling*, an estimation approach which is extensively used for behavioral and RT-level estimation (see Sections 18.3 and 18.4).

In Ref. 5, the power cost of a CPU module is characterized by estimating the average capacitance that would switch when the given CPU module is activated. In Ref. 6, the switching activities on (address, instruction, and data) buses are used to estimate the power consumption of the microprocessor. In Ref. 7, based on actual current measurements of some processors, Tiwari et al. present the following instruction-level power model:

$$Energy_p = \sum_i (BC_i N_i) + \sum_{i,j} (SC_{i,j} N_{i,j}) + \sum_k OC_k \qquad (18.1)$$

where $Energy_p$ is the total energy dissipation of the program which is divided into three parts. The first part is the summation of the base energy cost of each instruction (BC_i is the base energy cost and N_i is the number of times instruction i is executed). The second part accounts for the circuit state ($SC_{i,j}$ is the energy cost when instruction i is followed by j during the program execution). Finally, the third part accounts for energy contribution OC_k of other instruction effects such as stalls and cache misses during the program execution.

In Ref. 8, Hsieh et al. present a new approach, called *profile-driven program synthesis*, to perform RT-level power estimation for high-performance CPUs. Instead of using a macro-modeling equation to model the energy dissipation of a microprocessor in such a way that the resulting instruction trace behaves (in terms of performance and power dissipation) much the same as the original trace. The new instruction trace is however much shorter than the original one, and can hence be simulated on an RT-level description of the target microprocessor to provide the lower dissipation results quickly.

Specifically, this approach consists of the following steps:

1. Perform architectural simulation of the target microprocessor under the instruction trace of typical application programs.
2. Extract a *characteristic profile*, including parameters such as the instruction mix, instruction/data cache miss rates, branch prediction miss rate, pipeline stalls, etc., for the microprocessor.
3. Use mixed-integer linear programming and heuristic rules to gradually transform a generic program template into a fully functional program.
4. Perform RT-level simulation of the target microprocessor under the instruction trace of the new synthesized program.

Notice that the performance of the architectural simulator in gate-vectors/second is roughly 3 to 4 orders of magnitude higher than that of an RT-level simulator.

This approach has been applied to the Intel Pentium processor (which is a super-scalar pipelined CPU with 8-KB 2-way set-associative data, instruction and data caches, branch prediction, and dual instruction pipeline) demonstrating 3 to 5 orders of magnitude reduction in the RT-level simulation time with negligible estimation error.

18.3 Behavioral-Level Power Estimation

Conversely from some of the RT-level methods that will be discussed in Section 18.4, estimation techniques at the behavioral level cannot rely on information about the gate-level structure of the design components, and hence, must resort to abstract notions of physical capacitance and switching activity to predict power dissipation in the design.

Information-Theoretic Models

Information theoretic approaches for high-level power estimation[9,10] depend on information-theoretic measures of activity (e.g., entropy) to obtain quick power estimates.

Entropy characterizes the randomness or uncertainty of a sequence of applied vectors and thus it is intuitively related to switching activity; that is, if the signal switching is high, it is likely that the bit sequence is random, resulting in high entropy. Suppose the sequence contains t distinct vectors and let p_i denote the occurrence probability of any vector v in the sequence. Obviously, $\sum_{i=1}^{t} p_i = 1$. The entropy of the sequence is given by:

$$h = -\sum_{i=1}^{t} p_i \log p_i \tag{18.2}$$

where log x denotes the base 2 logarithm of x. The entropy achieves its maximum value of log t when $p_i = 1/t$. For an n-bit vector, $t \leq 2^n$. This makes the computation of the exact entropy very expensive. Assuming that the individual bits in the vector are independent, then we can write:

$$h = -\sum_{i=1}^{n} (q_i \log q_i + (1 - q_i)\log(1 - q_i)) \tag{18.3}$$

where q_i denotes the signal probability of bit i in the vector sequence. Note that this equation is only an upperbound on the exact entropy, since the bits may be dependent. This upperbound expression is, however, the one that is used for power estimation purposes. Furthermore, in Ref. 9, it has been shown that, under the temporal independence assumption, the average switching activity of a bit is upper-bounded by one half of its entropy. The power dissipation in the circuit can be approximated as:

$$Power = 0.5 V^2 f C_{tot} E_{avg} \tag{18.4}$$

where C_{tot} is the total capacitance of the logic module (including gate and interconnect capacitances) and E_{avg} is the average activity of each line in the circuit which is, in turn, approximated by one half of its average entropy, h_{avg}. The average line entropy is computed by abstracting information obtained from a gate-level implementation. In Ref. 10, it is assumed that the word-level entropy per logic level reduces quadratically from circuit inputs to circuit outputs, whereas in Ref. 9, it is assumed that the bit-level entropy from one logic level to next decreases in an exponential manner. Based on these assumptions, two different computational models are obtained.

In Ref. 9, Marculescu et al. derive a closed-form expression for the average line entropy for the case of a linear gate distribution; that is, when the number of nodes scales linearly between the number of circuit inputs, n, and circuit outputs, m. The expression for h_{avg} is given by:

$$h_{avg} = \frac{2nh_{in}}{(n+m)\ln\frac{h_{in}}{h_{out}}}\left(1 - \frac{m}{n}\frac{h_{out}}{h_{in}} - \frac{\left(1-\frac{m}{n}\right)\left(1-\frac{h_{out}}{h_{in}}\right)}{\ln\frac{h_{in}}{h_{out}}}\right) \tag{18.5}$$

where h_{in} and h_{out} denote the average bit-level entropies of circuit inputs and outputs, respectively. h_{in} is extracted from the given input sequence, whereas h_{out} is calculated from a quick functional simulation of the circuit under the given input sequence or by empirical entropy propagation techniques for precharacterized library modules. In Ref. 10, Nemani and Najm propose the following expression for h_{avg}:

$$h_{avg} = \frac{2}{3(n+m)}(H_{in} + H_{out}) \tag{18.6}$$

where H_{in} and H_{out} denote the average sectional (word-level) entropies of circuit inputs and outputs, respectively. The sectional entropy measures H_{in} and H_{out} may be obtained by monitoring the input and output signal values during a high-level simulation of the circuit. In practice, however, they are approximated as the summation of individual bit-level entropies, h_{in} and h_{out}.

If the circuit structure is given, the total module capacitance is calculated by traversing the circuit netlist and summing up the gate loadings. Wire capacitances are estimated using statistical wire load models. Otherwise, C_{tot} is estimated by quick mapping (e.g., mapping to 3-input universal gates) or by information theoretic models that relate the gate complexity of a design to the difference of its input and output entropies. One such model proposed by Cheng and Agrawal,[11] for example, estimates C_{tot} as:

$$C_{tot} = \frac{m}{n}2^n h_{out} \tag{18.7}$$

This estimate tends to be too pessimistic when n is large; hence, Ferrandi et al.[12] present a new total capacitance estimate based on the number N of nodes (i.e., 2-to-1 multiplexers) in the *Ordered Binary Decision Diagrams* (OBDD)[13] representation of the logic circuit as follows:

$$C_{tot} = \alpha\frac{m}{n}Nh_{out} + \beta \tag{18.8}$$

The coefficients of the model are obtained empirically by doing linear regression analysis on the total capacitance values for a large number of synthesized circuits.

Entropic models for the controller circuitry are proposed by Tyagi,[14] where three entropic lower bounds on the average Hamming distance (bit changes) with state set S and with T states are provided. The tightest lower bound derived in this chapter for a sparse finite state machine (FSM) (i.e., $t \le 2.23T^{1.72}/\sqrt{\log T}$, where t is the total number of transitions with non-zero steady-state probability) is the following:

$$\sum_{s_i, s_j \in S} p_{i,j}H(s_i, s_j) \ge h(p_{i,j}) - 1.52\log T - 2.16 + 0.5\log(\log T) \tag{18.9}$$

where $p_{i,j}$ is the steady-state transition probability from s_i to s_j, $H(s_i, s_j)$ is the Hamming distance between the two states, and $h(p_{i,j})$ is the entropy of the probability distribution $p_{i,j}$. Notice that the lower bound is valid regardless of the state encoding used.

In Ref. 15, using a Markov chain model for the behavior of the states of the FSM, the authors derive theoretical lower and upper bounds for the average Hamming distance on the state lines which are valid irrespective of the state encoding used in the final implementation. Experimental results obtained for the mcnc'91 benchmark suite show that these bounds are tighter than the bounds reported in Ref. 14.

Complexity-Based Models

These models relate the circuit power dissipation to some notion of *circuit complexity*. Example parameters that influence the circuit complexity include the number and the type of arithmetic and/or Boolean operations in the behavioral description, the number of states and/or transitions in a controller description, and the number of cubes (literals) in a minimum sum-of-products (factored-form) expression of a Boolean function.

Most of the proposed complexity-based models rely on the assumption that the complexity of a circuit can be estimated by the number of "equivalent gates." This information may be generated on-the-fly using analytical predictor functions, or retrieved from a pre-characterized high-level design library. An example of this technique is the *chip estimation system*,[16] which uses the following expression for the average power dissipation of a logic module:

$$Power = fN(Energy_{gate} + 0.5V^2 C_{load})E_{gate} \tag{18.10}$$

where f is the clock frequency, N is the gate equivalent count for the component, $Energy_{gate}$ is the average internal consumption for an equivalent gate (in includes parasitic capacitance contributions as well as short-circuit currents) per logic transition, C_{load} is the average capacitive load for an equivalent gate (it includes fanout load capacitances and interconnect capacitances), and E_{gate} is the average output activity for an equivalent gate per cycle. C_{load} is estimated statistically, based on the average fanout count in the circuit and custom woire load models. E_{gate} is dependent on the functionality of the module. The data is precalculated and stored in the library and is independent of the implementation style (static vs. dynamic logic, clocking strategy), and the circuit context in which the module is instantiated. This is an example of an implementation-independent and data-independent power estimation model.

In Ref. 17, Nemani and Najm present a high-level estimation model for predicting the area of an optimized single-output Boolean function. The model is based on the assumption that the area complexity of a Boolean function f is related to the distribution of the sizes of the on-set and off-set of the function. For example, using the "linear measure," the area complexity of the on-set of f is written as:

$$C_1(f) = \sum_{i=1}^{N} c_i p_i \tag{18.11}$$

where the set of integers $\{c_1, c_2, \ldots, c_N\}$ consists of the distinct sizes of the essential prime implicants of the on-set and weight p_i is the probability of the set of all minterms in the on-set of f which are covered by essential primes of size c_i, but not by essential primes of any larger size. The area complexity of the off-set of f $C_0(f)$ is similarly calculated. Hence, the complexity of function f is estimated as:

$$C(f) = \frac{C_1(f) + C_0(f)}{2} \tag{18.12}$$

The authors next derive a family of regression curves (which happen to have exponential form) relating the actual area $A(f)$ of random logic functions optimized by the SIS logic optimization program (in terms of the number of gates) to the area complexity measure $C(f)$ for different output probabilities of function f. These regression equations are subsequently used for total capacitance estimation and hence high-level power estimation. The work is extended in Ref. 18 to area estimation of multiple-output Boolean functions.

A similar technique would rely on predicting the quality of results produced by electronic design automation (EDA) flows and tools. The predictor function is obtained by performing regression analysis on a large number of circuits synthesized by the tools and relating circuit-specific parameters and/or design constraints to post-synthesis power dissipation results. For example, one may be able to produce the power estimate for an unoptimized Boolean network by extracting certain structural properties of the underlying directed acyclic graph, average complexity of each node, and user-specified constraints and plugging these values into the predictor function.

Complexity-based power prediction models for controller circuitry have been proposed by Landman and Rabaey.[19] These techniques provide quick estimation of the power dissipation in a control circuit based on the knowledge of its target implementation style (i.e., pre-charged pseudo-NMOS or dynamic PLA), the number of inputs, outputs, states, etc. The estimates will have a higher degree of accuracy by introducing empirical parameters that are determined by curve-fitting and least-squared fit error analysis on real data. For example, the power model for an FSM implemented in standard cells is given by:

$$Power = 0.5 V^2 f (N_I C_I E_I + N_O C_O E_O) N_M \tag{18.13}$$

where N_I and N_O denote the number of external input plus state lines for the FSM, C_I and C_O are regression coefficients which are empirically derived from low-level simulation of previously designed standard cell controllers, E_I and E_O denote the switching activities on the external input plus state lines and external output plus state lines, and finally N_M denotes the number of minterms in an optimized cover of the FSM. Dependence on N_M indicates that this model requires a partial (perhaps symbolic) implementation of the FSM.

Synthesis-Based Models

One approach for behavioral-level power prediction is to assume some RT-level template and produce estimates based on that assumption. This approach requires the development of a *quick synthesis* capability that makes some behavioral choices (mimicking a full synthesis program). Important behavioral choices include type of I/O, memory organization, pipelining issues, synchronization scheme, bus architecture, and controller design. This is a difficult problem, especially in the presence of tight timing constraints. Fortunately, designers or the environment often provide hints on what choices should be made. After the RT-level structure is obtained, the power is estimated using any of the RT-level techniques that will be described in Section 18.4.

Relevant data statistics such as the number of operations of a given type, bus and memory accesses, and I/O operations, are captured by *static profiling* based on stochastic analysis of the behavioral description and data streams,[20,21] or *dynamic profiling* based on direct simulation of the behavior under a typical input stream.[22,23] Instruction-level or behavioral simulators are easily adapted to produce this information.

18.4 RT-Level Power Estimation

Most RT-level power estimation techniques use regression-based, switched capacitance models for circuit modules. Such techniques, which are commonly known as *power macro-modeling*, are reviewed next.

Regression-Based Models

A typical RT-level power estimation flow consists of the following steps:

1. Characterize every component in the high-level design library by simulating it under pseudo-random data and fitting a multi-variable regression curve (i.e., the power macro-model equation) to the power dissipation results using a least mean square error fit.[24]

2. Extract the variable values for the macro-model equation from either static analysis of the circuit structure and functionality, or by performing a behavioral simulation of the circuit. In the latter case, a power co-simulator linked with a standard RT-level simulator can be used to collect input data statistics for various RT-level modules in the design.
3. Evaluate the power macro-model equations for high-level design components which are found in the library by plugging the parameter values into the corresponding macro-model equations.
4. Estimate the power dissipation for random logic or interface circuitry by simulating the gate-level description of these components, or by performing probabilistic power estimation. The low-level simulation can be significantly sped up by the application of statistical sampling techniques or automata-based compaction techniques.

The macro-model for the components may be parameterized in terms of the input bit width, the internal organization/architecture of the component, and the supply voltage level. Notice that there are cases where the construction of the macro-model of step (1) can be done analytically using the information about the structure of the gate-level description of the modules, without resorting to simulation as proposed by Benini et al.[25] On the other hand, if the low-level netlist of the library components is not known (which may be the case for soft macros), step (1) can be replaced by data collection from past designs of the component followed by appropriate process technology scaling.[26] In addition, the macro-model equation in step (2) may be replaced by a table lookup with necessary interpolation equations.

In the following paragraphs, we review various power macro-model equations which exhibit different levels of accuracy vs. computation/information usage tradeoff.

The simplest power macro-model, known as the *power factor approximation* technique,[27] is a *constant type model* that uses an experimentally determined weighting factor to model the average power consumed by a given module per input change. For example, the power dissipation of an $n \times n$ bit integer multiplier can be written as:

$$Power = 0.5 V^2 n^2 C f_{activ} \tag{18.14}$$

where V is the supply voltage level, C is the capacitive regression coefficient, and f_{activ} is the activation frequency of the module (this should not be confused with the average, bit-level switching activity of multiplier inputs).

The weakness of this technique is that it does not account for the data dependency of the power dissipation. For example, if one of the inputs to the multiplier is always 1, we would expect the power dissipation to be less than when both inputs are changing randomly. In contrast, the *stochastic power analysis* technique proposed by Landman and Rabaey[28] is based on an activity-sensitive macro-model, called the *dual bit type model*, which maintains that switching activities of high-order bits depend on the temporal correlation of data, whereas lower-order bits behave randomly. The module is thus completely characterized by its capacitance models in the sign and white noise bit regions. The macro-model equation form is then given by:

$$Power = 0.5 V^2 f \left(n_u C_u E_u + n_s \sum_{xy = ++}^{--} C_{xy} E_{xy} \right) \tag{18.15}$$

where C_u and E_u represent the capacitance coefficient and the mean activity of the unsigned bits of the input sequence, while C_{xy} and E_{xy} denote the capacitance coefficient and the transition probability for the sign change xy in the input stream. n_u and n_s represent the number of unsigned and sign bits in the input patterns, respectively. Note that E_u, E_{xy}, and the boundary between sign and noise bits are determined based on the applied signal statistics collected from simulation runs. Expanding this direction, one can use a *bitwise data model* as follows:

$$Power = 0.5V^2 f \sum_{i=1}^{n} C_i E_i \tag{18.16}$$

where n is the number of inputs for the module in question, C_i is the (regression) capacitance for input pin i, and E_i is the switching activity for the ith pin of the module. This equation can produce more accurate results by including, for example, spatio-temporal correlation coefficients among the circuit inputs. This will, however, significantly increase the number of variables in the macro-model equation, and thus the equation evaluation overhead.

Accuracy may be improved (especially for components with deep logic nesting, such as multipliers) by power macro-modeling with respect to both the average input and output activities (the *input-output data model*); that is:

$$Power = 0.5V^2 f(C_I E_I + C_O E_O) \tag{18.17}$$

where C_I and C_O represent the capacitance coefficients for the mean activities of the input and output bits, respectively. The dual bit type model or the bitwise data model may be combined with the input-output data model to create a more accurate, but more expensive, macro-model form. Recently, in Ref. 29, the authors presented a 3-D table, power macro-modeling technique which captures the dependence of power dissipation in a combinational logic circuit on the average input signal probability, the average switching activity of the input lines, and the average (zero-delay) switching activity of the output lines. The latter parameter is obtained from a fast functional simulation of the circuit. The paper also presents an automatic macro-model construction procedure based on random sampling principles. Note that the equation form and variables used for every module are the same (although the regression coefficients are different).

A parametric power model is described by Liu and Svensson,[30] where the power dissipation of the various components of a typical processor architecture, including on-chip memory, busses, local and global interconnect lines, H-tree clock net, off-chip drivers, random logic, and data path, are expressed as a function of a set of parameters related to the implementation style and internal architecture of these components. For example, consider a typical on-chip memory (a storage array of 6-transistor memory cells) which consists of four parts: the memory cells, the row decoder, the column selection, the read/write circuits. The power model for a cell array of 2^{n-k} rows and 2^k columns in turn consists of expressions for: (1) the power consumed by 2^k memory cells on a row during one pre-charge or one evaluation; (2) the power consumed by the row decoder; (3) the power needed for driving the selected row; (4) the power consumed by the column select part; and (5) the power dissipated in the sense amplifier and the readout inverter. For instance, the memory cell power (expression 1 in above) is given by:

$$Power_{memcell} = 0.5VV_{swing}2^k(C_{int} + 2^{n-k}C_{tr}) \tag{18.18}$$

where V_{swing} is the voltage swing on the bit/$\overline{bit}$ line (which may be different for read versus write), C_{int} gives the wiring-related row capacitance per memory cell, and $2^{n-k}C_{tr}$ gives the total drain capacitances on the bit/$\overline{bit}$ line. Notice that during the read time, every memory cell on the selected row drives exactly bit or $\overline{bit}$.

A salient feature of the above macro-model techniques that that they only provide information about average power consumption over a relatively large number of clock cycles. The above techniques, which are suitable for estimating the average-power dissipation, are referred to as *cumulative* power macro-models. In some applications, however, estimation of average power only is not sufficient. Examples are circuit reliability analysis (maximum current limits, heat dissipation and temperature gradient calculation, latch-up conditions), noise analysis (resistive voltage drop and inductive bounce on power and ground lines), and design optimization (power/ground net topology design, number and placement of decoupling capacitors, buffer insertion, etc.). In these cases, *cycle-accurate* (*pattern-accurate*) power estimates are required.

Mehta et al.[31] propose a clustering approach for pattern-accurate power estimation. This approach relies on the assumption that closely related input transitions have similar power dissipation. Hence, each input pattern is first mapped into a cluster, and then a table lookup is performed to obtain the corresponding power estimates from pre-calculated and stored power characterization data for the cluster. The weakness of this approach is that, for efficiency reasons, the number of clusters has to be relatively small, which would introduce errors into the estimation result. In addition, the assumption that closely related patterns (e.g., patterns with short Hamming distance) result in similar power distribution may be quite inaccurate, especially when the *mode-changing bits* are involved; that is, when a bit change may cause a dramatic change in the module behavior.

Addressing these problems, Wu et al.[32] describe an automatic procedure for cycle-accurate macro-model generation based on statistical sampling for the *training set* design and regression analysis combined with appropriate statistical tests (i.e., the F^* test) for macro-model variable selection and coefficient calculation. The test identifies the most (least) power-critical variable to add to (delete from) the set of selected variables. The statistical framework enables prediction of the power value and the confidence level for the predicted power value. This work is extended by Qiu et al.[33] to capture "important" first-order temporal correlations of up to order three at the circuit inputs. Note that here the equation form and variables used for each module are unique to that module type. Experimental results show that power macro-models with a relatively small number of input variables (i.e., 8) predict the module power with a typical error of 5 to 10% for the average power and 10 to 20% for the cycle power.

The above-mentioned high-level power macro-modeling techniques are mostly applicable to the combinational modules (such as adders, multipliers) in the circuit. This is because the power consumption in a combinational circuit module can be fully determined from the statistical characteristics of the vector sequence applied to its primary inputs. As mentioned above, in some cases, it is beneficial to use variables related to the vectors which appear on the primary outputs to further improve the accuracy of the power macro-model for combinational modules. For sequential circuit modules, using only variables related to the external inputs and outputs is not enough to model the power consumption accurately. this is because power consumption in sequential circuits strongly depends on the state transitions, which are not visible from outside. To generate an accurate power model for the sequential module, one thus needs to include some variables related to the internal state of the module.

For complex intellectual property (IP) cores, generating a power macro-model is even more challenging because of the following:

- The IP core may have internal state variables; it is therefore very difficult to generate an accurate power model for the IP core if we have access to information about its primary inputs and primary outputs only. Therefore, it is desirable to construct a power macro-model for the IP core that includes variables related to not only its primary inputs/outputs, but also to its internal state variables.
- At the system level, IP cores are usually specified behaviorally through input/output behavioral modeling. Power modeling however requires information about the internal states of the IP core. Therefore, it is necessary to modify the IP core specification at the system level so that the power model and power evaluation tool can retrieve the information they need for power estimation and optimization.
- There may be thousands of internal state variables in the IP core. Using all of these variables will result in a power model which is too large to store in the system library and too expensive to evaluate. Furthermore, requiring information about all internal states greatly increases the complexity of IP core specification and puts undue burden on the IP vendor to reveal details about the IP core, which can otherwise be hidden from the IP user.

To solve these problems, one needs to develop an IP power model constructor which will automatically generate power macro-models for IP cores using the minimum number of internal state variables. This may be achieved using a statistical relevance testing procedure such that only the "important" variables

are retained in the power model. In this way, a power macro-model with small number of variables, yet sufficient accuracy, can be constructed.

Sampling-Based Models

RT-level power evaluation can be implemented in the form of a *power co-simulator* for standard RT-level simulators. The co-simulator responsible for collecting input statistics from the output of the behavioral simulator and producing the power value at the end. If the co-simulator is invoked by the RT-level simulator every simulation cycle to collect activity information in the circuit, it is called *census macro-modeling*.

Evaluating the macro-model equation at each cycle during the simulation is actually a census survey. The overhead of data collection and macro-model evaluation can be high. To reduce the runtime overhead, Hsieh et al.[34] use *simple random sampling* to select a sample and calculate the macro-model equation for the vector pairs in the sample only. The sample size is determined before simulation. The *sampler macro-modeling* randomly selects *n* cycles and marks those cycles. When the behavioral simulator reaches the marked cycle, the macro-modeling invokes the behavioral simulator for the current input vectors and previous input vectors for each module. The input statistics are only collected in these marked cycles. Instead of selecting only one sample of large size, we can select several samples of at least 30 units (to ensure normality of sample distribution) before the simulation. Then, the average value of sample means is the estimate of population mean. In this manner, the overhead of collecting input statistics at every cycle which is required by census macro-modeling is substantially reduced. Experimental results show that sampler macro-modeling results in an average efficiency improvement of 50X over the census macro-modeling with an average error of 1%.

The macro-model equation is developed using a training set of input vectors. The training set satisfies certain assumptions such as being pseudo-random data, speech data, etc. Hence, the macro-model may become biased, meaning that it produces very good results for the class of data which behave similarly to the training set; otherwise, it produces poor results. One way to reduce the gap between the power macro-model equation and the gate-level power estimation is to use a *regression estimator* as follows.[34] It can be shown that the plot of the gate-level power value versus a well-designed macro-model equation estimate for many functional units reveals an approximately linear relationship. Hence, the macro-model equation can be used as a predictor for the gate-level power value. In other words, the sample variance of the ratio of gate-level power to macro-model equation power tends to be much smaller than that of the gate-level power by itself. It is thus more efficient to estimate the mean value of this ratio and then use a linear regression equation to calculate the mean value of the circuit-level power. The *adaptive macro-modeling* thus invokes a gate-level simulator on a small number of cycles to improve the macro-model equation estimation accuracy. In this manner, the "bias" of the static macro-models is reduced or even eliminated. Experimental results show that the census macro-modeling incurs large error (an average of 30% for the benchmark circuits) compared to gate-level simulation. The adaptive macro-modeling however exhibits an average error of only 5%, which demonstrates the superiority of the adaptive macro-modeling technique.

18.5 Gate-Level Power Estimation

In CMOS circuits, power is mostly consumed during logic switching. This simplifies the power estimation problem to that of calculating the toggle count of each circuit node under a gate-level delay model. Several gate-level power estimation techniques have been proposed in the past. These techniques, which offer orders of magnitude speed-up compared to the conventional simulation-based techniques at the circuit level, can be classified into two classes: *dynamic* and *static*. Dynamic techniques explicitly simulate the circuit under a "typical" input vector sequence (or input stream). The potential problem with these approaches is that the estimation results strongly depend on the input vector sequence, a phenomenon often described as *pattern dependence*. The static technique do not explicitly simulate under any input

vector sequence. Instead, they rely on statistical information (such as the mean activity and correlations) about the input stream. The pattern dependence problem in these techniques is less severe as these techniques either implicitly consider all input vector sequences or perform a "smoothing" operation on the input streams to reduce dependence on any given sequence.

Statistical Sampling

Existing techniques for power estimation at the gate and circuit levels can be divided into two classes: *static* and *dynamic*. Static techniques rely on probabilistic information about the input stream (such as the mean activity of the input signals and their correlations) to estimate the internal switching activity of the circuit. While these are very efficient, their main limitation is that they cannot accurately capture factors such as slew rates, glitch generation and propagation, and DC fighting. Dynamic techniques explicitly simulate the circuit under a "typical" input stream. They can be applied at both the circuit and gate levels. Their main shortcoming is, however, that they are very slow. Moreover, their results are highly dependent on the simulated sequence. To alleviate this dependence, and thereby produce a trustworthy power estimate, the required number of simulated vectors is usually high, which further exacerbates the runtime problem. An example of direct simulation techniques is given in Ref. 35.

To address this problem, a *Monte Carlo simulation* technique was proposed.[36] This technique uses an input model based on a Markov process to generate the input stream for simulation. The simulation is performed in an iterative fashion. In each iteration, a vector sequence of fixed length (called sample) is simulated. The simulation results are monitored to calculate the mean value and variance of the samples. The iteration terminates when some stopping criterion is met (see Fig. 18.1).

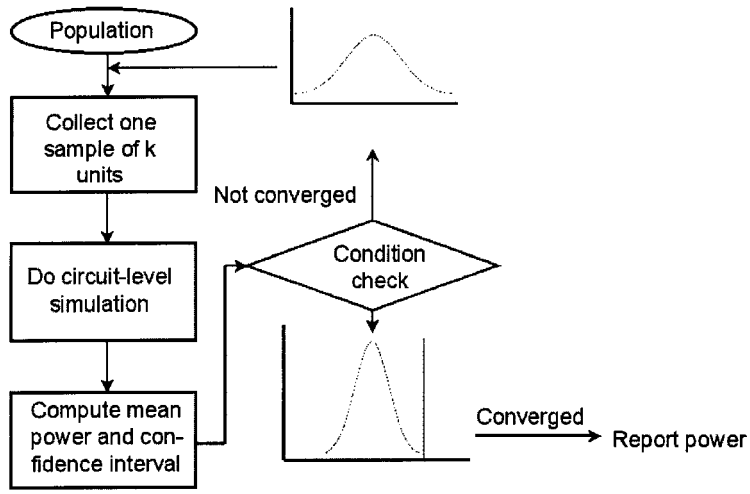


FIGURE 18.1 Monte Carlo simulation loop.

This approach suffers from three major shortcomings. First, when the vectors are regenerated for simulation, the spatial correlations among various inputs cannot be adequately captured, which may lead to inaccuracy in the power estimates. Second, the required number of samples, which directly impacts the simulation runtime, is approximately proportional to the ratio between the sample variance and the square of the sample mean value. For certain input sequences, this ratio becomes large, thus significantly increasing the simulation runtime. Finally, there is a general concern about the normality assumption of the sample distribution. Since the stopping criterion is based on such an assumption, if the sample distribution deviates significantly from the normal distribution (which may happen if the number of units per sample is small or the population distribution is ill-behaved), then the simulation may terminate

prematurely. Ill-behaved population distributions that may cause premature termination include bi-modal, multi-modal, and distributions with long or asymmetric tails.

The first concern can be addressed by developing a sampling procedure which randomly draws the units in the sample from the input population (instead of building an approximate Markov model of the population and then generating samples which conform to this approximate model). Such techniques are known as *simple random sampling*.[37] They may or may not be combined with the Monte Carlo simulation loop.

The second (and to some extent, the third) concern can be addressed by developing more efficient sampling procedures. One such procedure, called *stratified random sampling*, is introduced in Ref. 37. They key idea is to partition the population into disjoint subpopulations (called *strata*) in such a way that the power consumption characteristics within each stratum becomes more homogeneous. The samples are then taken randomly from each stratum. The units in each sample are allocated proportional to the sizes of the strata. This generally results in a significant reduction in the sampling variance (see Figs. 18.2 and 18.3).

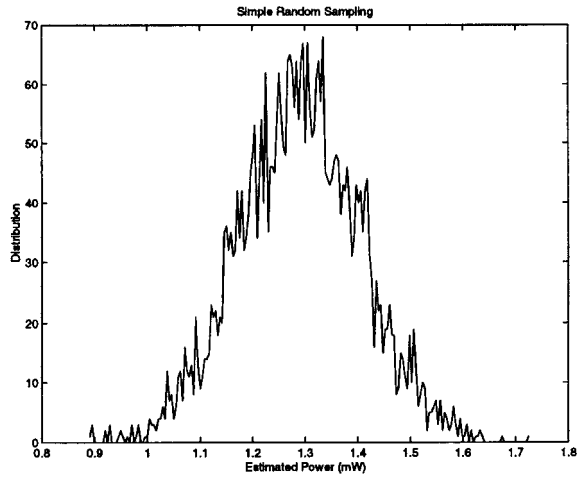


FIGURE 18.2 Sample distribution for simple random sampling.

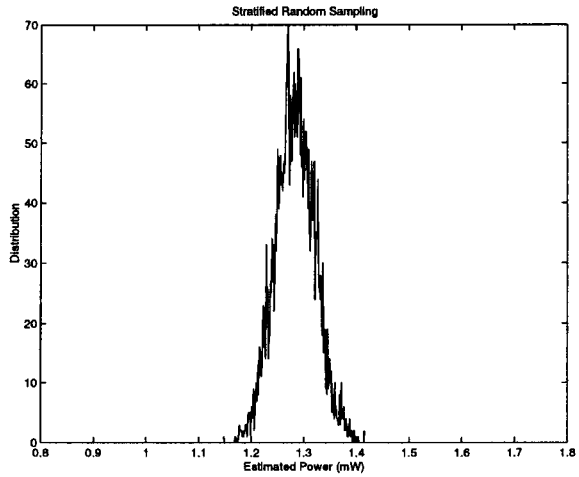


FIGURE 18.3 Sample distribution for stratified random sampling.

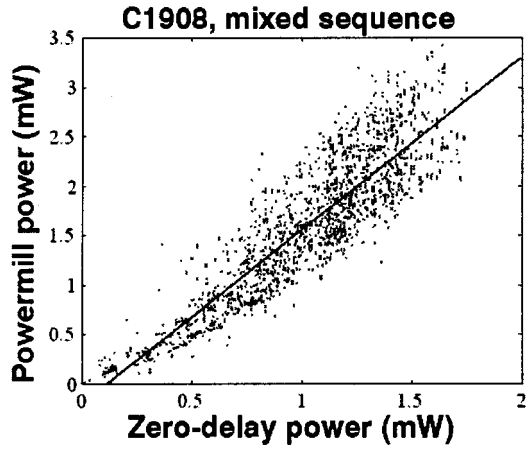


FIGURE 18.4 Scatter plot of circuit-level power vs. zero-delay power.

The stratification itself is based on a low-cost predictor (e.g., zero-delay power estimation) which needs to be evaluated for every unit in the population. The zero-delay power estimates need not produce high accuracy on a unit-by-unit basis; indeed, they should only show high statistical correlation with circuit-level power estimates (see, for example, the scatter plot for the ISCAS-85 benchmark C1809 shown in Fig. 18.4). When the population size is large, one can use a two-stage stratified sampling procedure to reduce the overhead of predictor calculation and stratification. The proposed two-stage stratified sampling technique can be easily extended to multi-stage sampling.

Compared to Ref. 36, the technique in Ref. 37 offers the following advantages: (1) it performs sampling directly on the population and thus the estimation results are unbiased; (2) experimental results on a large set of the ISCAS'85 benchmarks under non-random input sequences show a 10X improvement in the sampling efficiency; and (3) difficult population distributions can be handled more effectively.

Another efficient sampling procedure, called *linear regression estimation*, is introduced in Ref. 34. The key observation is that the sample variance of the ratio of circuit-level power to zero-delay power tends to be much smaller than that of the circuit-level power by itself. This can be easily seen by examining the scatter plots of circuit-level power versus zero-delay power for a large number of circuits under a variety of input populations.

It is thus more efficient to estimate the mean value of this ratio and then use a regression equation to calculate the mean value of the circuit-level power. Experimental results again show a 10X improvement in the sampling efficiency compared to simple random sampling.

So far, we have only considered examples of parametric sampling techniques (i.e., those whose stopping criterion is derived by assuming normality of the sample distribution). A non-parametric sampling technique (i.e., those whose stopping criterion is derived without any *a priori* assumption about the sample distribution) is presented in Ref. 38. In general, non-parametric techniques do not suffer from the premature termination problem; however, they tend to be too conservative and lead to over-sampling to achieve the specified error and confidence levels. The technique in Ref. 38, which uses the properties of "order statistics," tries to reach a good tradeoff between the estimation accuracy and the computational efficiency. More research is needed to assess the efficiency and robustness of non-parametric versus parametric sampling techniques.

For synchronous sequential circuits (e.g., finite state machines driven by a common clock), if we know the state transition graph of the FSM, we can solve the Chapman-Kolmogorov equations for the stationary state probabilities. Next, based on these probabilities, we can randomly generate a present-state vector which, along with a randomly generated external input vector, determines the next-state vector. With a

second random external input vector (generated according to the statistics of the external input sequence), a unit of the power sample can be constructed. The subsequent units are generated by randomly setting the state line vectors, followed by random generation of two external input vectors and power measurement. Unfortunately, the number of Chapman-Kolmogorov equations is exponential in the number of flip-flops, and hence this approach is not possible for large FSMs.

Existing statistical technique for the estimation of the mean power is synchronous sequential circuits (e.g., finite state machines driven by a common clock) are classified into two groups: hierarchical and flat. The hierarchical technique performs behavioral or RT-level simulation of the target circuit for all units (i.e., external input patterns) in the user-specified population, and collects the state line values corresponding to each unit. Next, it treats the FSM as a combinational circuit (i.e., it cuts the sequential feedback lines) which receives an extended input sequence and which needs to be simulated at the circuit level for accurate power estimation; the new sequence is the concatenation of the external input line and the state line values. The FSM power estimation problem is thereby transformed into a sampling problem for the resulting combinational circuit (assuming that all the flip-flops are edge-triggered). The shortcomings of this technique are that it requires RT-level simulation of the target FSM for all units of the population and that the resulting state line values must be stored for sampling in the next phase. The first problem is not critical since the RT level simulation is orders of magnitude faster than the circuit-level simulation and hence we can afford simulating the whole vector sequence at the RT level. The second problem may be of concern if the length of the input sequence is large and the computer memory is limited.

The hierarchical technique takes a somewhat more complicated form when the input population is infinite; in this case, the signal and transition probabilities of the state lines must be modeled by a Markov process which is, in turn, derived from a Markov model representation of the external input sequence. This problem has been tackled in Ref. 39, where the following two-phase procedure is proposed: (1) use sequence generation based on the given Markov model of each bit of the external input sequence and do Monte Carlo simulation to compute the signal probabilities and transition probabilities of each state line (and hence construct the corresponding Markov model for each state line); (2) cut the feedback lines in the FSM and do Monte Carlo simulation on the combinational part of the FSM using the bit-level Markov models of the external input and state lines. A major disadvantage of this technique is that in the second step, while estimating power in the combinational circuit, spatial correlations between the signal lines are ignored. This introduces large errors in the power estimates.

The flat technique[36,40] consists of two phases: warm-up period and random sampling. The purpose of the warm-up period is to perform a number of input vector simulations to achieve steady-state probability conditions on the state lines. Only then can power sampling be performed. This is because, for meaningful power sampling, the state vectors fed into the circuit have to be generated according to their stationary state probability distribution in the FSM. The problems of how to choose an initial state for each sample and how long the warm-up length should be are discussed in Ref. 40. A randomness test to dynamically decide the proper warm-up length is proposed in Ref. 41. The test is applied to the sequence of power values observed during the warm-up period (a binning technique is used to transform the sequence of power values into a binary sequence so that the test can be applied). A major concern with this technique is that the randomness test should actually be applied to the sequence of state line patterns, rather than to the sequence of power values. The former task appears to be difficult.

In practice, the warm-up period requires a large number of simulated vectors (depending on the FSM behavior and characteristics of the external input sequence). This makes the efficiency of power estimation for sequential circuits much lower than that for combinational circuits.

In addition to estimating the mean value of the power dissipation in a circuit, theory of order statistics and stratified sampling techniques have been used to estimate the maximum power dissipation[42] and the cumulative distribution function for the power dissipation.[43] This information is very useful to chip designers who are interested in reliability analysis and AC/DC noise analysis.

Probabilistic Compaction

Another approach for reducing the power simulation time is to compact the given long stream of bit vectors using probabilistic automata.[44,45] The idea in Ref. 45 is to build a *stochastic state machine* (SSM) which captures the relevant statistical properties of a given long bit stream, and then excite this machine by a small number of random inputs so that the output sequence of the machine is statistically equivalent to the initial one. The relevant statistical properties denote, for example, the signal and transition probabilities, and first-order spatio-temporal correlations among bits and across consecutive time frames. The procedure then consists of decomposing the SSM into a set of deterministic state machines, and realizing it through SSM synthesis with some auxiliary inputs. The compacted sequence is generated by uniformly random excitement of such inputs. As an example, consider the input sequence shown in Fig. 18.5; the corresponding SSM model is shown in Fig. 18.6, and the compacted sequence is shown in Fig. 18.7. Note that the average bit activities for the initial compacted sequences are 23/16 and 8/5, respectively.

```
0110101001111101O  (bit A)
1110000011011000  (bit B)
1000011010111010l  (bit C)
```

FIGURE 18.5 Initial sequence.

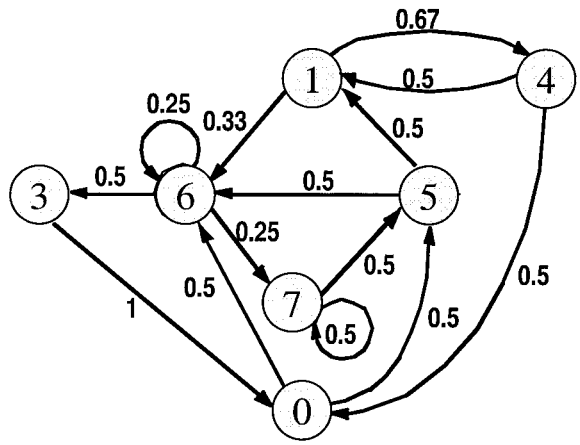


FIGURE 18.6 SSM model for the sequence.

```
001010  (bit A)
011000  (bit B)
010001  (bit C)
```

FIGURE 18.7 Compacted sequence.

The number of states in the probabilistic automaton is proportional to the number of distinct patterns in the initial vector sequence. Since this number may be large (i.e., worst-case exponential in the number of bits in each vector), one has to manage the complexity by either: (1) partitioning the n bits into b groups with a maximum size of k bits per group and then building a probabilistic

automaton for each group of bits independently; or (2) partitioning the long vector sequence into consecutive blocks of vectors such that the number of distinct vectors in each block does not exceed some user-defined parameter, say *K*. The shortcoming of the first approach is that one may generate a bit pattern (vector) that was not present in the initial sequence (since correlations across different groups of bits are ignored). This is, in turn, a problem in certain applications with forbidden input patterns (codes not used, illegal instructions, bad memory addressed, etc.). Thus, the second approach is often more desirable.

An improved algorithm for vector compaction is presented in Ref. 46. The foundation of this approach is also probabilistic in nature: it relies on adaptive (dynamic) modeling of binary input streams as first-order Markov sources of information and is applicable to both combinational and sequential circuits. The adaptive modeling technique itself (best known as *dynamic Markov chain modeling*) was recently introduced in the literature on data compression[47] as a candidate to solve various data compression problems. This original formulation is extended in Ref. 46 to manage not only correlations among adjacent bits that belong to the same input vector, but also correlations between successive patterns. The model captures completely spatial correlations and first-order temporal correlations and, conceptually, it has no inherent limitation to be further extended to capture temporal dependencies of higher orders.

A hierarchical technique for compacting large sequences of input vectors is presented in Ref. 48. The distinctive feature of this approach is that it introduces hierarchical Markov chain modeling as a flexible framework for capturing not only complex spatio-temporal correlations, but also dynamic changes in the sequence characteristics such as different input modes. The hierarchical Markov model is used to structure the input space into a hierarchy of macro- and micro-states: at the first level in the hierarchy, there is a Markov chain of macro-states describing the input modes, whereas at the second level there is a Markov chain of micro-states describing the internal behavior of each input mode. The primary motivation in doing this structuring is to enable a better modeling of the different stochastic levels that are present in sequences that arise in practice.

Another important property of such models is to individualize different operating modes of a circuit or system via higher levels in the hierarchical Markov model, thus providing a high adaptability of different system operating modes (e.g., active, standby, sleep, etc.). The structure of the model itself is general and, with further extensions, can allow an arbitrary number of activations of its sub-models. Once we have constructed the hierarchy for the input sequence, starting with a macro-state, a compaction procedure with a specified compaction ratio is applied to compact the set of micro-states within that macro-state. When done with processing the current macro-state, the control returns to the higher level in the hierarchy and, based ont he conditional probabilities that characterize the Markov chain at this level, a new macro-state is entered and the process repeats.

This sequence compaction approach has been extended in Ref. 49 to handle FSMs. More precisely, it is shown that: (1) under the stationarity and ergodicity assumptions, complete capture of the characteristics of the external input sequence feeding into a target FSM is sufficient to correctly characterize the joint (external inputs plus internal states) transition probabilities; (2) if the external input sequence has order k, then a lag-k Markov chain model of this sequence will suffice to exactly model the joint transition probabilities in the target FSM; (3) if the input sequence has order 2 or higher, then modeling it as a lag-1 Markov chain cannot exactly preserve even the first-order joint transition probabilities in the target FSM. They key problem is thus to determine the order of the Markov source which models the external input sequence. In Ref. 50, based on the notion of block (metric) entropy, a technique for identify the order of a *composite* source of information (i.e., a source which emits sequences that can be piecewise modeled by Markov chains of different orders) is introduced.

Results of these approaches show 1 to 4 orders of magnitude compaction (depending on the initial length and characteristics of the input sequence) with negligible error (i.e., $\leq$5% in most cases) using PowerMill as the simulator. As a peculiar property, note that none of these approaches needs the actual circuit to compact the input sequences.

Probabilistic Simulation

Power dissipation in CMOS circuits comes from three sources: (1) the leakage current, (2) the short-circuit which is due to the DC path between the supply rails during output transitions, and (3) the charging and discharging of capacitive loads during logic changes. In a well-designed circuit, the contribution of first two sources is relatively small. Therefore, power dissipation at the gate level can be accurately approximated by only considering the charging and discharging of capacitive loads as given by:

$$Power_{gate} = \frac{1}{2T_c} V_{dd}^2 \sum_n C_n sw_n \tag{18.19}$$

where T_c is the clock period, V_{dd} is the supply voltage, the summation is performed over all gates (nodes) in the circuit, C_n is the load capacitance of node n and sw_n is the average switching activity of node n, (that is, the expected number of signal changes) per clock cycle.

In the above equation, sw_n is related to the timing relationship (or delay) among the input signals of each gate. Indeed, the output signal of a node may change more than once due to unequal signal arriving times at its inputs. The power consumed by these extra signal changes is generally referred to as the *glitch power*. If the contribution of the glitch power is relatively small, one can approximate sw_n using the zero-delay model (e.g., assuming that all logic changes propagate through the circuit instantaneously, and hence each node changes logic value at most once). The ratio of the power estimations obtained under real-delay and the zero-delay models reflects the significance of glitch power. A typical value for this ratio is 1.20. For some types of circuits, such as parity chains, adders, or multipliers, this ratio could be well above 2.0. For these types of circuits, the delay model becomes very important, that is, the delay characterization of each gate must be very accurate. Moreover, CMOS gates have an inertial delay. Only glitches with adequate strength (that is, glitch width) can overcome the gate inertia and propagate through the gate to its output. Accurate modeling of these effects requires significantly longer execution time.

A direct simulative approach to power estimation would enumerate all pairs of input vectors and compute the number of logic changes per vector pair. The "average" switching activity is then the sum over all possible vector pairs of the product of the occurrence probability and the number of the logic changes for each vector pair. This procedure will obviously be exponential in complexity. In the following, we review some important definitions that will help manage this complexity:

1. *Signal probability*: The signal probability $P_s(n)$ of a node n is defined as the average fraction of clock cycle in which the steady-state value of n is logic one under the zero-delay model.
2. *Transition probability*: The transition $P_t^{it}(n)$ of a node is defined as the average fraction of clock cycles in which the steady-state value of n undergoes a change from logic value i to logic value j under the zero-delay model. Note that $P_t^{01}(n) = P_t^{10}(n)$ and $P_t^{01}(n) + P_t^{11}(n) = P_s(n)$.
3. *Temporal independence*: Under the temporal independence assumption, the signal value of an input node n at clock cycle i is independent of its signal value at clock cycle $i - 1$.
4. *Spatial independence*: Under the spatial independence assumption, the logic value of an input node n is independent of the logic value of any other input node m.
5. *Spatiotemporal correlation*: The spatial and temporal correlation of the input signals are collectively referred as *spatiotemporal correlation*.
6. *Full signal swing*. All logic transitions are from zero to one and vice versa (i.e., no intermediate voltages are present).

The concepts of signal and transition probabilities are very useful for power estimation under the zero-delay model. Under the real-delay model, the signal and transition probabilities are mainly used to describe the input vector statistics.

Under the zero-delay model, $P_t^{01}(n) + P_t^{10}(n)$ can replace the term sw_n in the power calculation to give the power consumption for each node. With the temporal independence assumption, $P_t^{01}(n)$ can be written as:

$$P_t^{01}(n) = (1 - P_s(n))P_s(n) \tag{18.20}$$

Therefore, the simulation can be performed using one vector to obtain the signal probabilities instead of using two vectors to explicitly calculate the transition probabilities.

The early works for computing the signal probabilities in a combinational network adopt the spatial independence assumption to keep the problem manageable. In Fig. 18.8, we use signal probability calculation based on OBDDs to illustrate how the spatial independence assumption improves the efficiency. Fig. 18.8(a) shows the example circuit and Fig. 18.8(b) shows the global function of node g represented by an OBDD. With the spatial independence assumption, the signal probability $P_s(g)$ can be calculated by recursively co-factoring the global function of g, that is:

$$g = cg_c + \bar{c}g_{\bar{c}} \tag{18.21}$$

$$P_s(g) = P_s(c)P(g_c) + (1 - P_s(c))P(g_{\bar{c}}) \tag{18.22}$$

In Fig. 18.8(b), the left and right branches of the root node compute g_c and $g_{\bar{c}}$, respectively. By recursively applying these equations with respect to all OBDD variables, $P_s(g)$ can be computed efficiently by traversing each OBDD node exactly once using a dynamic programming approach. Without the spatial independence assumption, the second equation becomes invalid. In the worst case, one may need to explicitly enumerate all the disjoint cubes (paths in the OBDD representation) in function g to compute $P_s(g)$.

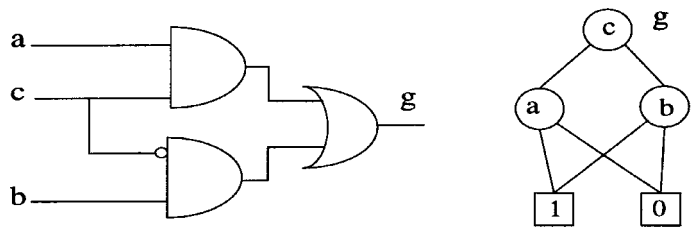


FIGURE 18.8 An OBDD example.

When we compute the signal probabilities using OBDD, the OBDDs can be either *global* or *local*. The former refers to OBDDs which are constructed in terms of the variables associated with circuit inputs, while the latter refers to OBDDs which are constructed in terms of the variables associated with some set of intermediate nodes in the fanin cone of the node in question.

We should point out that the temporal and spatial independence assumptions are only made to improve the efficiency at the cost of reducing the accuracy. These assumptions do not hold for most real-life input vectors. Therefore, the ability to do power estimation while accounting for real-life temporal and spatial correlations becomes an important criterion when comparing various estimation techniques.

The second class of switching activity estimation techniques, called *static techniques*, does not explicitly simulate under the given input vector sequence. This class can be further divided into the following groups: *exact* and *approximate*.

Exact Probabilistic Analysis

These exact techniques provide the exact switching activity estimates under the assumed delay models and specified input statistics. This is achieved by *implicitly exhaustive enumeration* of all input vectors. Unfortunately, the worst-case complexity of these approaches is exponential. Moreover, the data structures employed to perform the implicit enumeration are also exponential in the circuit input size. As a result, the feasibility of these techniques is restricted to small-size circuits.

Exact Techniques under the Zero Delay Model

In Ref. 51, each of the circuit inputs is associated with a variable name that represents the input signal probability. Given the algebraic expressions, in term of these variables, of all fanin nodes of an internal circuit node g, an algebraic expression for g can be derived based on the node function of g. The signal probability of each circuit node g is then calculated from the derived algebraic expression. A much more effective approach based on OBDDs in proposed in Ref. 52. This latter technique was illustrated with the example in Fig. 18.8. Both of these approaches assume temporal and spatial independence of circuit input signals.

Another OBDD-based approach that considers the temporal correlation of the input signals is proposed in Ref. 53. The technique uses OBDD variables of twice the number of circuit inputs. That is, for each circuit input c, two OBDD variables, c_1 and c_2, are used to represent the signal values of c at time $-\infty$ and 0 in a two-vector simulation. The computation of transition probability of each circuit node g is carried out as follows. Let g^{01} represent the Boolean function for g to produce a 0 to 1 signal change under the zero-delay model. We can write:

$$g^{01} = \bar{c}_1\bar{c}_2 g_{\bar{c}_1\bar{c}_2} + \bar{c}_1 c_2 g_{\bar{c}_1 c_2} + c_1\bar{c}_2 g_{c_1\bar{c}_2} + c_1 c_2 g_{c_1 c_2} \tag{18.23}$$

$$P_t^{01}(g) = P_t^{00}(c)P\left(g_{\bar{c}_1\bar{c}_2}\right) + P_t^{01}(c)P\left(g_{\bar{c}_1 c_2}\right) + P_t^{10}(c)P\left(g_{c_1\bar{c}_2}\right) + P_t^{11}(c)P(g_{c_1 c_2}) \tag{18.24}$$

The ordering of the OBDD variables is arranged so that c_1 and c_2 for each circuit input c are next to each other. This ordering provides an effective method to find the co-factors directly from the OBDD nodes. A more efficient method of calculating the transition probabilities without variable duplication is presented in Ref. 54.

Exact Techniques under the Real-Delay Model

An approach based on *symbolic simulation* is proposed in Ref. 55. In the symbolic simulation, the state of an internal circuit node g is described by a set of symbolic functions defined over time $(-\infty, \infty)$ in the temporal order; for example, $g_{t_0}, g_{t_1}, \cdots, g_{t_n}$ where $t_0 = -\infty$ and g_{t_i} specifies the input vectors that will produce logic one over time $[g_{t_i}, g_{t_{i+1}}]([g_{t_i}, \infty])$ if $i < n$ $(i = n)$. The state of each circuit input c is described by two single-variable symbolic functions, c_1 and c_2, defined at time $-\infty$ and 0, respectively. The computation of all symbolic functions for each gate can be performed during a topological order. This procedure is similar to event-driven simulation. That is, for each symbolic function defined at time t at any fanin of gate g, a new symbolic function at the output of g at time $t + d$ (d is the delay of the gate) is constructed based on the Boolean function of g and the states of all other fanins of g at time t. The sw_n is calculated by summing the signal probabilities of the exclusive or functions of symbolic functions that are defined at two consecutive time instances; for example,

$$sw_g = \sum_{i=0}^{n-1} P_s(g_{t_i} \oplus g_{t_{i+1}}) \tag{18.25}$$

The major disadvantage of this method is that it is computationally intractable, as the number of symbolic functions for each gate could be large. Moreover, the symbolic functions could become very complicated when one attempts to model the glitch propagation mechanism accurately and potential filter out glitches that have short width.

So far, no exact technique have been proposed for considering the spatial correlation of input signals under either the zero or the real-delay model. The difficulty lies in the fact that the spatial correlations can in general exist among every m-tuple of inputs where m ranges from 2 to the number of circuit inputs n. In the worst case, there is an exponential number of spatial correlation parameters that need to be considered compared to the $4n$ for the temporal correlation case.

Approximate Techniques

These techniques are developed as approximation techniques for the implicit enumeration approaches. They are referred as *approximate probabilistic techniques,* mainly because the probabilistic quantities such as signal and transition probabilities are explicitly (vs. implicitly in exact techniques) propagated in the networks.

Probabilistic Techniques under the Zero-Delay Model

An efficient algorithm to estimate the signal probability of each internal node using pair-wise correlation coefficients among circuit nodes is proposed in Ref. 56. This technique allows the spatial correlations between pairs of circuit input signals to be considered. A more general approach that accounts for spatiotemporal correlations is proposed in Ref. 54. The mathematical foundation of this extension is a four-state, time-homogeneous Markov chain where each state represents some assignment of binary values to two lines x and y, and each edge describes the conditional probability for going from one state to the next. The computational requirement of this extension can however be high since it is linear in the product of the number of nodes and number of paths in the OBDD representation of the Boolean function in question. A practical method using local OBDD constructions and dynamic level founding is described by the authors.

This work has been extended to handle highly correlated input streams using the notions of conditional independence and isotropy of signals.[57] Based on these notions, it is shown that the relative error in calculating the signal probability of a logic gate using pairwise correlation coefficients can be bounded from above.

Probabilistic Techniques under the Real-Delay Model

In Ref. 58 (*CREST*), the concept of *probability waveforms* is proposed to estimate the mean and variance of the current drawn by each circuit node. Although *CREST* was originally developed for switch-level simulation, the concept of probability waveforms can be easily extended to the gate level as well. A probability waveform, consists of an initial signal probability and a sequence of transition events occurring at different time instances. Associates with each transition event is the probability of the signal change. In a probability waveform, one can calculate the signal probability at any time instance t from the initial signal probability and probabilities of each transition event which occurred before t. The propagation mechanism for probability waveforms is event-driven in nature. For each transition event arrived at time t at the input of a gate g, a transition event is scheduled at the gate output at time $t + d$ (d is the delay of the gate); probability of the event is calculated based on the probability of the input events and the signal probabilities of other fanins at time t.

In Ref. 59 (*DENSIM*), the notion of *transition density* — which is the average number of transitions per second — is proposed. It can replace SW_n/T_c in power calculation equation to compute the power dissipation for each node. An efficient algorithm based on the Boolean difference operation is proposed to propagate the transition densities from circuit inputs throughout the circuit with the transition density of each node calculated based on the following formula:

$$D(y) = \sum_{i=1}^{n} P\left(\frac{\partial y}{\partial x_i}\right) D(x_i) \tag{18.26}$$

where y is the output of a node, x_i's are the inputs of y, $D(y)$ is the transition density of node y, and $(\partial y)/(\partial x_i)$ is the Boolean difference of y respect to x_i. $P((\partial y)/(\partial x_i))$ is calculated using OBDDs. The accuracy of the transition density equation can be improved by considering higher-order Boolean difference.[60]

Under the real-delay model, the correlation between gate fanins is the major source of inaccuracy for the probabilistic techniques. Moreover, because the signals may change more than once, it is very difficult to accurately model these correlations. In Ref. 61, based on an assumption on the probability distribution function of the glitch width, a conceptual low-pass filter module is proposed that improves the accuracy of *DENSIM*.

In Ref. 62 (*TPS*), the concept of probability waveforms is extended to partially account for the signal correlation among different gate fanins. The signal correlation is approximated by the steady-state correlations under the zero-delay model. To efficiently implement the techniques, the probability waveform of a node is divided into four *tagged waveforms* based on initial and final steady state. For a two-input node g, there are 16 joint tagged waveforms at the gate inputs. After the probability waveforms for all 16 joint tagged waveforms are calculated, they are combined into four tagged waveforms according to a forcing set table[62] derived from the node function of g. The OBDDs are used to calculate the probabilities of each tagged waveform and the signal correlations. This approach requires significantly less memory and runs much faster than symbolic simulation, yet achieves high accuracy (e.g., the average error in aggregate power consumption is about 10%). One interesting characteristic of this approach is that it will give an exact power estimate for the zero-delay model if the zero-delay model is assumed. Moreover, the technique can be combined with Refs. 54 and 57 to consider the spatiotemporal correlations of circuit input signals.

Both *DENSIM* and *TPS* use OBDDs to improve their efficiency; their computational work can be divided into two phases. The first phase constructs the required OBDDs and computes the signal probabilities or correlations using OBDDs; the second phase computes either the transition density or the tagged waveforms during a post-order traversal from circuit inputs to circuit outputs.

Probabilistic Techniques for Finite State Machines

The above-mentioned probabilistic methods for power estimation focus on combinational logic circuits. Accurate average switching activity estimation for FSMs is considerably more difficult that that for combinational circuits for two reasons:

- The probability of the circuit being in each of its possible states has to be calculated.
- The present-state line inputs of the FSM are strongly correlated (i.e., they are temporally correlated due to the machine behavior as represented in its State Transition Graph description and they are spatially correlated because of the given state encoding).

A first attempt at estimating switching activity in the FSMs was presented in Ref. 55. The idea is to "unroll" the next-state logic once (thus capturing the temporal correlations of present-state lines) and then perform symbolic simulation on the resulting circuit (which is hence treated as a combinational circuit). This method does not however capture the spatial correlations among present-state lines and makes the simplistic assumption that the state probabilities are uniform.

The above work was improved on in Ref. 63 as follows. For each state s_i, $1 \le i \le K$ in the STG, we associate a variable $prob(s_i)$ corresponding to the steady-state probability of the machine being in state s_i at $t = \infty$. For each edge e in the STG, we have *e.Current* signifying the state that the edge fans out from, *e.Next* signifying the state that the edge fans out to, and *e.Input* signifying the input combination corresponding to the edge. Given static probabilities for the primary inputs to the machine, we can compute $prob(Input)$, the probability of the combination *Input* occurring.[1] We can compute $prob(e.Input)$ using:

[1] Static probabilities can be computed from specified transition probabilities.

$$prob(e.Input) = prob(e.Current) \times prob(Input) \tag{18.27}$$

For each state s_i, we can write an equation:

$$prob(s_i) = \sum_{\forall e \text{ such that } e.Next = s_i} prob(e.Input) \tag{18.28}$$

Given K states, we obtain K equations, out of which any one equation can be derived from the remaining $K - 1$ equations. We have a final equation:

$$\sum_{i=1}^{K} prob(s_i) = 1 \tag{18.29}$$

This linear set of K equations can be solved to obtain the different $prob(s_i)$s. This system of equations is known as the Chapman-Kolmogorov equations for a discrete-time discrete-transition Markov process. Indeed, if the process satisfies the conditions that it has a finite number of states, its essential states form a single chain, and it contains no periodic states, then the above system of equations will have a unique solution.

The Chapman-Kolmogorov method requires the solution of a linear system of equations of size 2N, where N is the number of flip-flops in the machine. In general, this method cannot handle circuits with large numbers of flip-flops because it requires explicit consideration of each state in the circuit. One the positive side, state probabilities for some very large FSMs have been calculated using a fully implicit technique described in Ref. 64.

The authors of Ref. 63 also describe a method for approximate switching activity estimation of sequential circuits. The basic computation step is the solution of a non-linear system of equations as follows:

$$\begin{aligned} prob(ns_1) &= prob(f_1(i_1, i_2, \ldots, i_M, ps_1, ps_2, \ldots, ps_N)) \\ prob(ns_2) &= prob(f_2(i_1, i_2, \ldots, i_M, ps_1, ps_2, \ldots, ps_N)) \\ &\ldots \\ prob(ns_N) &= prob(f_N(i_1, i_2, \ldots, i_M, ps_1, ps_2, \ldots, ps_N)) \end{aligned} \tag{18.30}$$

where $prob(ns_i)$ corresponds to the probability that ns_i is a 1, and $prob(f_i(i_1, i_2, \ldots, i_M, ps_1, ps_2, \ldots, ps_N))$ corresponds to the probability that $f_i(i_1, i_2, \ldots, i_M, ps_1, ps_2, \ldots, ps_N)$ is a 1, which is of course dependent on the $prob(ps_j)$ and the $prob(i_k)$.

We are interested in the steady-state probabilities of the present and next-state lines implying that:

$$prob(ps_i) = prob(ns_i) = p_i \qquad 1 \le i \le N \tag{18.31}$$

A similar relationship was used in the Chapman-Kolmogorov equations.

The set of equations given the values of $prob(i_k)$ becomes:

$$\begin{aligned} y_1 &= p_1 - g_1(p_1, p_2, \ldots, p_N) = 0 \\ y_2 &= p_2 - g_2(p_1, p_2, \ldots, p_N) = 0 \\ &\ldots \\ y_N &= p_N - g_N(p_1, p_2, \ldots, p_N) = 0 \end{aligned} \tag{18.32}$$

where the g_i's are non-linear functions of the p_i's. We will denote the above equations as $Y(P) = 0$ or as $P = G(P)$. In general, the Boolean function f_i can be written as list of minterns over the i_k and ps_j, and the corresponding g_i function can be easily derived.

The fixed point (or zero) of this system of equations $P = G(P)$ (or $Y(P) = 0$) can be found using the Picard-Peano (or Newton-Raphson) iteration.[65] The uniqueness or the existence of the solution is not guaranteed for an arbitrary system of non-linear equations. However, since in our application we have a correspondence between the non-linear system of equations and the State Transition Graph of the sequential circuit, there will exist at least one solution to the non-linear system. Further, convergence is guaranteed under mild assumptions for our application.

Increasing the number of variables or the number of equations in the above system results in increased accuracy.[148] For a wide variety of examples, it is shown that the approximation scheme is within 1–3, but is orders of magnitude faster for large circuits. Previous sequential switching activity estimation methods exhibit significantly greater inaccuracies.

18.6 Transistor-Level Power Estimation

Transistor-level simulators provide the highest accuracy for circuit power estimation. They are capable of handling various device models, different circuit design styles, single- and multi-phase clocking methodologies, tristate drives, etc. However, they suffer from memory and execution time constraints and are not suitable for large, cell-based designs. Circuit techniques for power measurement (capacitive and short-circuit power components) using the "power meters" is described in Refs. 66 and 67. A fast and accurate circuit-level simulator based on the stepwise equivalent conductance and piecewise linear waveform approximation has been described in Ref. 68.

PowerMill[69] is a transistor-level power simulator and analyzer which applies an event-driven timing simulation algorithm (based on simplified table-driven device models, circuit partitioning and single-step nonlinear iteration) to increase the speed by two to three orders of magnitude over SPICE, while maintaining an accuracy of within 10power information (instantaneous, average, and RMS current values) as well as the total power consumption (due to capacitance currents, transient short circuit currents, and leakage currents).

18.7 Conclusion

The increased degree of automation of industrial design frameworks has produced a substantial change in the way digital ICs are developed. The design of modern systems usually starts from specifications given at a very high level of abstraction. This is because existing EDA tools are able to automatically produce low-level design implementations directly from descriptions of this type.

It is widely recognized that power consumption has become a critical issue in the development of digital systems; then, electronic designers need tools that allow them to explicitly control the power budget during the phases of the design process. This is because the power savings obtainable through automatic optimization are usually more significant than those achievable by means of technological choices (e.g., process and supply-voltage scaling).

This chapter has provided review of existing methodologies and tools for power modeling, simulation, and estimation, techniques of VLSI circuits at different design abstraction levels. A wealth of research results and a few pioneering commercial tools have appeared in the last few years. This chapter has, however, described representative contributions to this important field and has been non-exhaustive. We expect this field to remain quite active in the foreseeable future. New trends and techniques will emerge, some approaches described in this review will consolidate, while others will become obsolete; this is in view of technological and strategic changes in the world of microelectronics.

References

1. F. N. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 446-455, 1994.
2. M. Pedram, "Power Minimization in IC Design: Principles and Applications," *ACM Transactions on Design Automation of Electronic Systems*, vol. 1, no. 1, pp. 3-56, 1996.
3. J. M. Rabaey and M. Pedram, (Editors), *Low Power Design Methodologies*, Kluwer Academic Publishers, 1996.
4. J. Mermet and W. Nebel, (Editors), *Low Power Design in Deep Submicron Electronics*, Kluwer Academic Publishers, 1997.
5. T. Sato, Y. Ootaguro, M. Nagamatsu, and H. Tago, "Evaluation of Architectural-Level Power Estimation for CMOS RISC Processors," *ISLPE-95*: *IEEE International Symposium on Low Power Electronics*, San Jose, CA, pp. 44-45, Oct. 1995.
6. C.-L. Su, C.-Y. Tsui, and A. M. Despain, "Low Power Architecture Design and Compilation Techniques for High-Performance Processors," *IEEE CompCon-94*, pp. 489-498, Feb. 1994.
7. V. Tiwari, S. Malik, and A. Wolfe, "Power Analysis of Embedded Software: A First Step Towards Software Power Minimization," *IEEE Transactions on VLSI Systems*, vol. 2, no. 4, pp. 437-445, 1994.
8. C.-T. Hsieh, M. Pedram, H. Mehta, and F. Rastgar, "Profile-Driven Program Synthesis for Evaluation of System Power Dissipation," *DAC-34: ACM/IEEE Design Automation Conference*, Anaheim, CA, pp. 576-581, June 1997.
9. D. Marculescu, R. Marculescu, and M. Pedram, "Information Theoretic Measures for Power Analysis," *IEEE Transactions on CAD*, vol. 15, no. 6, pp. 599-610, 1996.
10. M. Nemani and F. Najm, "Towards a High-Level Power Estimation Capability," *IEEE Transactions on CAD*, vol. 15, no. 6, pp. 588-598, 1996.
11. K. T. Cheng and V. D. Agrawal, "An Entropy Measure for the Complexity of Multi-Output Boolean Functions," *DAC-27: ACM/IEEE Design Automation Conference*, Orlando, FL, pp. 302-305, June 1990.
12. F. Ferrandi, F. Fummi, E. Macii, M. Poncino, and D. Sciuto, "Power Estimation of Behavioral Descriptions," *DATE-98: IEEE Design Automation and Test in Europe*, Paris, France, pp. 762-766, Feb. 1998.
13. R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on CAD*, pp. 667-691, Aug. 1986.
14. A. Tyagi, "Entropic Bounds on FSM Switching," *IEEE Transactions on VLSI Systems*, vol. 5, no. 4, pp. 456-464, 1997.
15. D. Marculescu, R. Marculescu, and M. Pedram, "Theoretical bounds for switching activity analysis in finite-state machines," *ISLPED-98: ACM/IEEE International Symposium on Low Power Electronics and Design*, Monterey, CA, pp. 36-41, Aug. 1998.
16. K. Muller-Glaser, K. Kirsch, and K. Neusinger, "Estimating Essential Design Characteristics to Support Project Planning for ASIC Design Management," *ICCAD-91: IEEE/ACM International Conference on Computer Aided Design*, Santa Clara, CA, pp. 148-151, Nov. 1991.
17. M. Nemani and F. Najm, "High-Level Area Prediction for Power Estimation," *CICC-97: Custom Integrated Circuits Conference*, Santa Clara, CA, pp. 483-486, May 1997.
18. M. Nemani and F. Najm, "High-Level Area and Power Estimation for VLSI Circuits," *ICCAD-97: IEEE/ACM International Conference on Computer Aided Design*, San Jose, CA, pp. 114-119, Nov. 1997.
19. P. Landman and J. Rabaey, "Activity-Sensitive Architectural Power Analysis for the Control Path," *ISLPD-95: ACM/IEEE International Symposium on Low Power Design*, Dana Point, CA, pp. 93-98, April 1995.
20. A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing Power Using Transformations," *IEEE Transactions on CAD*, vol. 14, no. 1, pp. 12-31, 1995.
21. J. M. Chang and M. Pedram, "Module Assignment for Low Power," *EuroDAC-96: IEEE European Design Automation Conference*, pp. 376-381, Geneva, Switzerland, Sept. 1996.

22. N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-Driven Behavioral Synthesis for Low Power VLSI Systems," *IEEE Design and Test of Computers*, vol. 12, no. 3, pp. 70-84, 1995.
23. R. San Martin and J. Knight, "Optimizing Power in ASIC Behavioral Synthesis, *IEEE Design and Test of Computers*, vol. 13, no. 2, pp. 58-70, 1996.
24. L. Benini, A. Bogliolo, M. Favalli, and G. DeMicheli, "Regression Models for Behavioral Power Estimation," *PATMOS-96: International Workshop on Power and Timing Modeling, Optimization and Simulation*, pp. 176-186, Bologna, Italy, Sept. 1996.
25. L. Benini, A. Bogliolo, and G. DeMicheli, "Characterization-Free Behavioral Power Modeling," *DATE-98: IEEE Design Automation and Test in Europe*, Paris, France, pp. 767-773, Mar. 1997.
26. L. Benini, A. Bogliolo, and G. DeMicheli, "Adaptive Least Mean Square Behavioral Power Modeling," *EDTC-97: IEEE European Design and Test Conference*, Paris, France, pp. 404-410, Mar. 1997.
27. S. Powell and P. Chau, "Estimating Power Dissipation of VLSI Signal Processing Chips: The PFA Techniques," *IEEE Workshop on VLSI Signal Processing*, vol. IV, pp. 250-259, 1990.
28. P. Landman and J. Rabaey, "Power Estimation for High-Level Synthesis," *EDAC-93: IEEE European Conference on Design Automation*, Paris, France, pp. 361-366, Feb. 1993.
29. S. Gupta and F. N. Najm, "Power Macromodeling for High-Level Power Estimation," *DAC-34: ACM/IEEE Design Automation Conference*, Anaheim, CA, pp. 365-370, June 1997.
30. D. Liu and C. Svensson, "Power Consumption Estimation in CMOS VLSI Chips," *IEEE Journal of Solid State Circuits*, vol. 29, no. 6, pp. 663-670, 1994.
31. H. Mehta, R. Owens, and M. J. Irwin, "Energy Characterization Based on Clustering," *DAC-33: ACM/IEEE Design Automation Conference*, Las Vegas, NV, pp. 702-707, June 1996.
32. Q. Wu, C-S. Ding, C-T. Hsieh, and M. Pedram, "Statistical Design of Macro-Models for RT-Level Power Evaluation," *ASPDAC-2: ACM/IEEE Asia South Pacific Design Automation Conference*, Chiba, Japan, pp. 523-528, Jan. 1997.
33. Q. Qiu, Q. Wu, M. Pedram, and C-S. Ding, "Cycle-Accurate Macro-Models for RT-Level Power Analysis," *ISLPED-97: ACM/IEEE International Symposium on Low Power Electronics and Design*, Monterey, CA, pp. 125-130, Aug. 1997.
34. C-T. Hsieh, C-S. Ding, Q. Wu, and M. Pedram, "Statistical Sampling and Regression Estimation in Power Macro-Modeling," *ICCAD-96: IEEE/ACM International Conference on Computer Aided Design*, San Jose, CA, pp. 583-588, Nov. 1996.
35. C. M. Huizer, "Power Dissipation Analysis of CMOS VLSI Circuits by means of Switch-Level Simulation," *IEEE European Solid State Circuits Conference*, pp. 61-64, 1990.
36. R. Burch, F. Najm, P., and T. Trick, "A Monte Carlo Approach for Power Estimation," *IEEE Transactions on VLSI Systems*, vol. 1, no. 1, pp. 63-71, 1993.
37. C-S. Ding, C-T. Hsieh, Q. Wu, and M. Pedram, "Stratified Random Sampling for Power Estimation," *ICCAD-96: IEEE/ACM International Conference on Computer Aided Design*, San Jose, CA, pp. 577-582, Nov. 1996.
38. L-P. Yuan, C-C. Teng, and S-M. Kang, "Statistical Estimation of Average Power Dissipation in CMOS VLSI Circuits Using Nonparametric Technique," *ISLPED-96: ACM/IEEE International Symposium on Low Power Electronics and Design*, Monterey, CA, pp. 73-78, Aug. 1996.
39. F. N. Najm, S. Goel, and I. N. Hajj, "Power estimation in sequential circuits," *DAC-32: ACM/IEEE Design Automation Conference*, San Francisco, CA, pp. 635-640, June 1995.
40. T-L. Chou and K. Roy, "Statistical Estimation of Sequential Circuit Activity," *ICCAD-95: IEEE/ACM International Conference on Computer Aided Design*, San Jose, CA, pp. 34-37, Nov. 1995.
41. L.-P. Yuan, C.-C. Teng, and S.-M. Kang, "Statistical Estimation of Average Power Dissipation in Sequential Circuits," *DAC-33: ACM/IEEE Design Automation Conference*, Anaheim, CA, pp. 377-382, June 1996.
42. A. Hill, C.-C. Teng, and S. M. Kang, "Simulation-Based Maximum Power Estimation," *ISCAS-96: IEEE International Symposium on Circuits and Systems*, vol. IV, Atlanta, GA, pp. 13-16, May 1996.

43. C-S. Ding, Q. Wu, C.-T. Hsieh, and M. Pedram, "Statistical Estimation of the Cumulative Distribution Function for Power Dissipation in VLSI Circuits," *DAC-34: ACM/IEEE Design Automation Conference*, Anaheim, CA, pp. 371-376, June 1997.
44. C.-Y. Tsui, D. Marculescu, R. Marculecu, and M. Pedram, "Improving the Efficiency of Power Simulation by Input Vector Compaction," *DAC-33: ACM/IEEE Design Automation Conference*, Las Vegas, NV, pp. 165-168, June 1996.
45. D. Marculescu, R. Marculescu, and M. Pedram, "Stochastic Sequential Machine Synthesis Targeting Constrained Sequence Generation," *DAC-33: ACM/IEEE Design Automation Conference*, Las Vegas, NV, pp. 696-701, June 1996.
46. R. Marculescu, D. Marculescu, and M. Pedram, "Adaptive Models for Input Data Compaction for Power Simulators," *ASPDAC-2: ACM/IEEE Asia-Pacific Design Automation Conference*, Chiba, Japan, pp. 391-396, Jan. 1997.
47. G. V. Cormack and R. N. Horspool, "Data Compression Using Dynamic Markov Modeling," *Computer Journal*, vol. 30, no. 6, pp. 541-550, 1987.
48. R. Marculescu, D. Marculescu, and M. Pedram, "Power Estimation Using Hierarchical Markov Models," *DAC-34: ACM/IEEE Design Automation Conference*, Anaheim, CA, pp. 570-575, June 1997.
49. D. Marculescu, R. Marculescu, and M. Pedram, "Sequence Compaction for Probabilistic Analysis of Finite State Machines," *DAC-34: ACM/IEEE Design Automation Conference*, Anaheim, CA, pp. 12-15, June 1997.
50. R. Marculescu, D. Marculescu, and M. Pedram, "Composite Sequence Compaction for Finite State Machines Using Block Entropy and Higher-Order Markov Models," *ISLPED-97: ACM/IEEE International Symposium on Low Power Electronics and Design*, Monterey, CA, pp. 190-195, Aug. 1997.
51. K. P. Parker and J. McCluskey, "Probabilistic Treatment of General Combinational Networks," *IEEE Transactions on Computers*, vol. C24, pp. 668-670, June 1975.
52. S. Chakravarty, "On the Complexity of Using BDDs for the Synthesis and Analysis of Boolean Circuits," *27th Annual Allerton Conference on Communication, Control and Computing*, pp. 730-739, 1989.
53. P. Schneider and U. Schlichtmann, "Decomposition of Boolean Functions for Low Power Based on a New Power Estimation Technique," *WLPD-94: International Workshop on Low Power Design*, Napa, CA, pp. 123-128, April 1994.
54. R. Marculescu, D. Marculescu, and M. Pedram, "Switching Activity Analysis Considering Spatiotemporal Correlation," *ICCAD-94: IEEE/ACM International Conference on Computer Aided Design*, San Jose, CA, pp. 294-299, Nov., 1994.
55. A. Ghosh, S. Devadas, K. Keutzer, S. Malik, and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," *DAC-92: ACM/IEEE Design Automation Conference*, Anaheim, CA, pp. 253-259, June 1992.
56. S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco, "Testability Measures in Pseudorandom Tesing," *IEEE Transactions on CAD*, vol. 11, pp. 794-800, June 1992.
57. R. Marculescu, D. Marculescu, and M. Pedram, "Efficient Power Estimation for Highly Correlated Input Streams," *DAC-32: ACM/IEEE Design Automation Conference*, San Francisco, CA, pp. 628-634, June 1995.
58. F. Najm, R. Burch, P. Yang, and I. Hajj, "Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits," *IEEE Transactions on CAD*, vol. 9, no. 4, pp. 439-450, 1990.
59. F. Najm, "Transition Density: A New Measure of Activity in Digital Circuits," *IEEE Transactions on CAD*, vol. 12, no. 4, pp. 310-323, 1993.
60. T. L. Chou, K. Roy, and S. Prasad, "Estimation of Circuit Activity Considering Signal Correlation and Simultaneous Switching," *ICCAD-94: IEEE/ACM International Conference on Computer Aided Design*, San Jose, CA, pp. 300-303, Nov. 1994.
61. F. Najm, "Low-Pass Filter for Computing the Transition Density in Digital Circuits," *IEEE Transaction on CAD*, vol. 13, no. 9, pp. 1123-1131, Sep. 1994.

62. C.-Y. Tsui, M. Pedram, and A. M. Despain, "Efficient Estimation of Dynamic Power Dissipation Under a Real Delay Model," *ICCAD-93: IEEE/ACM International Conference on Computer Aided Design*, Santa Clara, CA, pp. 224-228, Nov. 1993.
63. C.-Y. Tsui, J. Montiero, M. Pedram, S. Devadas, A. M. Despain, and B. Lin, "Power Estimation in Sequential Logic Circuits," *IEEE Transactions on VLSI Systems*, vol. 3, no. 3, pp. 404-416, 1995.
64. G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Markovian Analysis of Large Finite State Machines," *IEEE Transactions on CAD*, vol. 15, no. 12, pp. 1479-1493, 1996.
65. H. M. Lieberstein. *A Course in Numerical Analysis*, Harper & Row Publishers, 1968.
66. S. M. Kang, "Accurate Simulation of Power Dissipation in VLSI Circuits," *IEEE J. Solid-State Circuits*, vol. 21, pp. 889-891, Oct. 1986.
67. G. Y. Yacoub and W. H. Ku, "An Enhanced Technique for Simulating Short-Circuit Power Dissipation," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 844-847, June 1989.
68. P. Buch, S. Lin, V. Nagasamy, and E. S. Kuh, Techniques for Fast Circuit Simulation Applied to Power Estimation of CMOS circuits," *SLPD-95: ACM/IEEE International Symposium on Low Power Design*, Dana Point, CA, pp. 135-138, April 1995.
69. C. X. Huang, B. Zhang, A-C. Deng, and B. Swirski, "The Design and Implementation of PowerMill," *SLPD-95: ACM/IEEE International Symposium on Low Power Design*, Dana Point, CA, pp. 105-110, April 1995.

Li, H.W., et al "CMOS Amplifier Design"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

19

CMOS Amplifier Design

Harry W. Li
University of Idaho at Moscow

R. Jacob Baker
University of Idaho at Boise

Donald C. Thelen
Analog Interfaces

19.1 Introduction

This chapter discusses the design, operation, and layout of CMOS analog amplifiers and subcircuits (current mirrors, biasing circuits, etc.). To make this discussion meaningful and clear, we need to define some important variables related to the DC operation of MOSFETs (Fig. 19.1). Figure 19.1(a) shows the simplified schematic representations of n- and p-channel MOSFETs. We say simplified because, when these symbols are used, it is *assumed* that the fourth terminal of the MOSFET (i.e., the body connection) is connected to either the lowest potential on the chip (V_{SS} or ground for the NMOS) or the highest potential (V_{DD} for the PMOS). Figure 19.1(b) shows the more general schematic representation of n- and p-channel MOSFETs. We are assuming that, although the drain and source of the MOSFETs are interchangeable, drain current flows from the top of the device to the bottom. Because of the assumed direction of current flow, the drain terminal of the n-channel is on the top of the symbol, while the drain terminal of the p-channel is on the bottom of the schematic symbol. The following are short descriptions of some important characteristics of MOSFETs that will be useful in the following discussion.

The Threshold Voltage

Loosely defined, the threshold voltage, V_{THN} or V_{THP}, is the minimum gate-to-source voltage (V_{GS} for the n-channel or V_{SG} for the p-channel) that causes a current to flow when a voltage is applied between the drain and source of the MOSFET. As shown in Fig. 19.1(c) the threshold voltage is estimated by plotting the square root of the drain current against the gate-source voltage of the MOSFET and looking at the intersection of the line tangent with this plot with the x-axis (V_{GS} for the n-channel). As seen in the figure, a current does flow below the threshold voltage of the device. This current is termed, for obvious reasons, the *subthreshold current*. The subthreshold current is characterized by

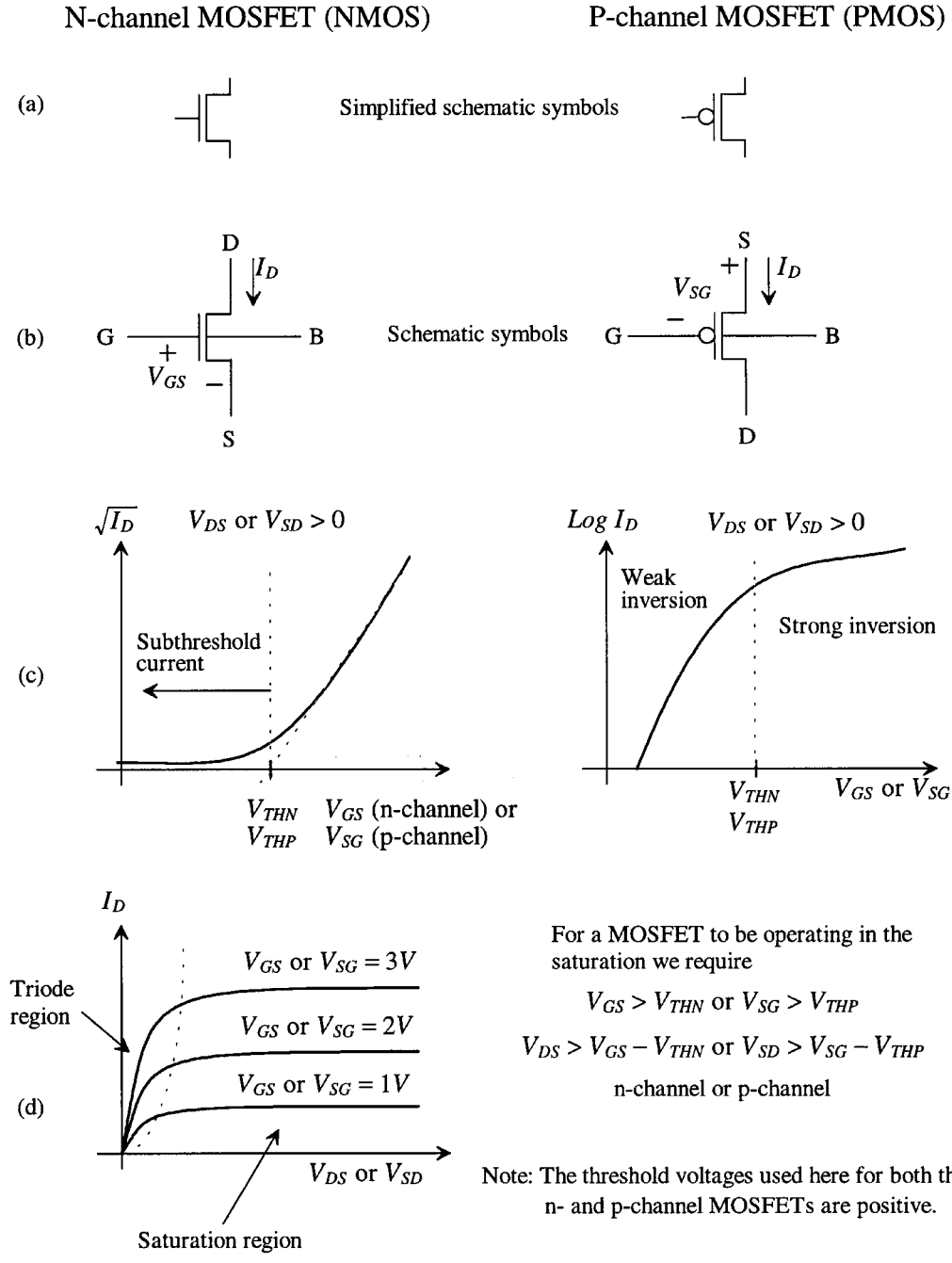


FIGURE 19.1 MOSFET device characteristics.

plotting the Log of the drain current against the gate-source voltage. The slope of the curve in the subthreshold region (sometimes also called the *weak inversion region*) is used to specify how the drain current changes with V_{GS}. A typical value for the reciprocal of the slope of this curve is 100 mV/dec. An equation relating the drain current of an n-channel MOSFET operating in the subthreshold region to V_{GS} is (assuming $V_{DS} > 100$ mV):

$$I_D = I_{D0} \cdot \frac{W}{L} \cdot \exp\left[\frac{q(V_{GS} - V_{THN})}{kT \cdot N}\right] \tag{19.1}$$

where W and L are the width and length of the MOSFET, I_{D0} is a measured constant, k is Boltzmann's constant (1.38×10^{-23} J/K), T is temperature in Kelvin, q is the electronic charge (1.609×10^{-23} C), and N is the slope parameter. Note that the slope of the Log I_D vs. V_{GS} curve in the subthreshold region is

$$\frac{\Delta \log I_D}{V_{GS}} = \frac{q \cdot \log e}{N \cdot kT} \tag{19.2}$$

The Body Effect

The threshold voltage of a MOSFET is dependent on the potential between the source of the MOSFET and its body. Consider Fig. 19.2, showing the situation when the body of an n-channel MOSFET is connected to ground and the source of the MOSFET is held V_{SB} above ground. As V_{SB} is increased (i.e., the potential on the source of the MOSFET increases relative to ground), the minimum V_{GS} needed to cause appreciable current to flow increases (V_{THN} increases as V_{SB} increases). We can relate V_{THN} to V_{SB} using the body-effect coefficient, γ, by

$$V_{THN} = V_{THN0} + \gamma\sqrt{|2\phi_F| + V_{SB}} - \sqrt{|2\phi_F|} \tag{19.3}$$

where V_{THN0} is the zero-bias threshold voltage when $V_{SB} = 0$ and ϕ_F is the surface electrostatic potential[1] with a typical value of 300 mV. An important thing to notice here is that the threshold voltage tends to change less with increasing source-to-substrate (body) potential (increasing V_{SB}).

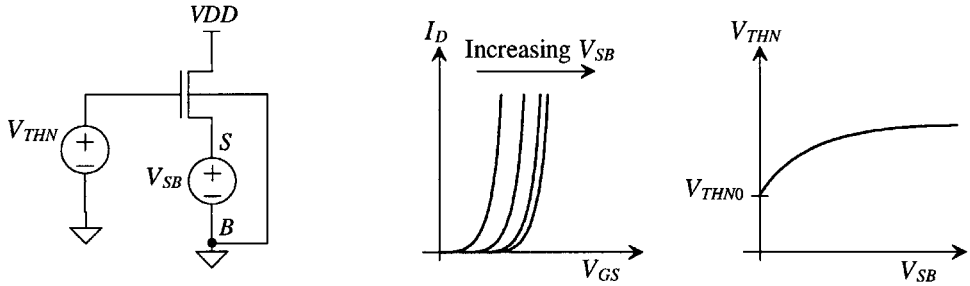


FIGURE 19.2 Illustration of the threshold voltage dependence on body-effect.

The Drain Current

In the following discussion, we will assume that the gate-source voltage of a MOSFET is greater than the threshold voltage so that a reasonably sized drain current can flow ($V_{GS} > V_{THN}$ or $V_{SG} > V_{THP}$). If this is the case, the MOSFET operates in either the triode region or the saturation region [Fig. 19.1(d)]. The drain current of a *long L* n-channel MOSFET operating in the *triode region*, is given by

$$I_D = \beta\left[(V_{GS} - V_{THN})V_{DS} - \frac{V_{DS}^2}{2}\right] \tag{19.4}$$

assuming long L with $V_{DS} \le V_{GS} - V_{THN}$. Note that the MOSFET behaves like a voltage-controlled resistor when operating in the deep triode region with a channel resistance (the resistance is measured between the drain and source of the MOSFET) approximated by

$$R_{CH} = \frac{1}{\beta(V_{GS} - V_{THN}) - \beta V_{DS}} \text{ or } \frac{1}{\beta(V_{GS} - V_{THN})} \text{ if } V_{DS} \ll V_{GS} - V_{THN} \qquad (19.5)$$

When doing analog design, it is often useful to implement a resistor whose value is dependent on a controlling voltage (a voltage-controlled resistor). When the NMOS device is operating in the saturation region, the drain current is given by

$$I_D = \beta(V_{GS} - V_{THN})^2[1 + \lambda(V_{DS} - V_{DS,\,sat})] \qquad (19.6)$$

assuming long L with $V_{DS} \geq V_{GS} - V_{THN}$, where $V_{DS,sat} = V_{GS}\ V_{THN}$.

The *transconductance parameter*, β, is given by

$$\beta = KP \cdot \frac{W}{L} = \mu_{n,\,p} C_{ox} \cdot \frac{W}{L} \qquad (19.7)$$

where $\mu_{n,p}$ is the mobility of either the electron or hole and C_{ox} is the oxide capacitance per unit area [ε_{ox}/t_{ox}, the dieletric constant of the gate oxide (35.4 aF/μm) divided by the gate oxide thickness]. Typical values for KP_n, KP_p, and C_{ox} for a 0.5-μm process are 150 μA/V^2, 50 μA/V^2, and 4 fF/μm^2, respectively. Also, an important thing to note in these equations for an n-channel MOSFET, is that V_{GS}, V_{DS}, and V_{THN} can be directly replaced with V_{SG}, V_{SD}, and V_{THP}, respectively, to obtain the operating equations for the p-channel MOSFET (keeping in mind that all quantities under normal conditions for operation are positive.) Also note that the saturation slope parameter λ (also known as the channel length/mobility modulation parameter) determines how changes in the drain-to-source voltage affect the MOSFET drain current and thus the MOSFET output resistance.

Short-Channel MOSFETs

As the channel length of a MOSFET is reduced, the electron and hole mobilities, μ_n and μ_p, start to get smaller. The mobility is simply a ratio of the electron or hole velocity to the applied electric field. Reducing the channel length increases the applied electric field while at the same time causing the velocity of the electron or hole to saturate (this velocity is labeled v_{sat}). This effect is called *mobility reduction* or *hot-carrier effects* (because the mobility also decreases with increasing temperature). The result, for a MOSFET with a short channel length L, is a reduction in drain current and a labeling of *short-channel MOSFET*. A short-channel MOSFET's current is, in general, linearly dependent on the MOSFET V_{GS} or

$$I_D = W \cdot v_{sat} \cdot C_{ox}(V_{GS} - V_{THN} - V_{DS,\,sat}) \qquad (19.8)$$

To avoid short-channel effects (and, as we shall see, increase the output resistance of the MOSFET when doing analog design), the channel length of the MOSFET is made, generally, 2 to 5 times larger than the minimum allowable L. For a 0.5-μm CMOS process, this means we make the channel length of the MOSFETs 1.0 to 2.5 μm.

MOSFET Output Resistance

An important parameter of a MOSFET in analog applications is its output resistance. Consider the portion of a MOSFET's I-V characteristics shown in Fig. 19.3(a). When the MOSFET is operating in the saturation region, the slope of I_D, because of changes in the drain current with changes in the drain-source voltage, is relatively small. If this change were zero, the drain current would be a fixed value independent of the voltage between the drain and source of the MOSFET (in other words, the MOSFET would act like an ideal current source.) Even with the small change in current with V_{DS}, we can think of the MOSFET as a current source. To model the changes in current with V_{DS}, we can simply place a resistor across the drain and source of the MOSFET [Fig. 19.3(b)]. The value of this resistor is

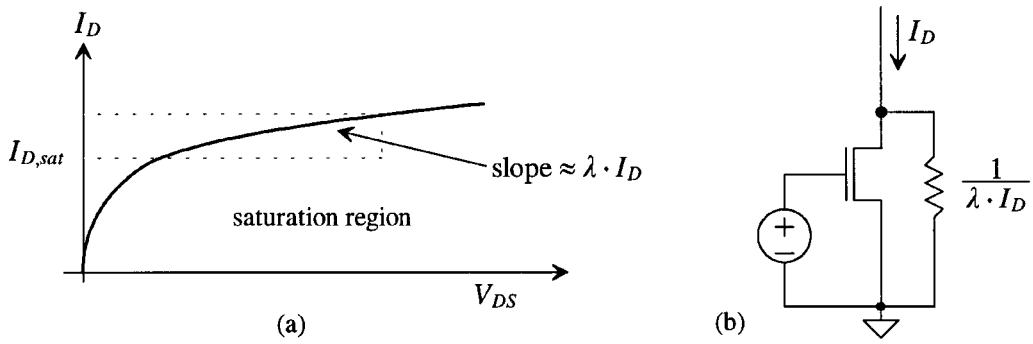


FIGURE 19.3 Output resistance of a MOSFET.

$$r_o \approx \frac{1}{\lambda I_D} \tag{19.9}$$

where λ is in the range of 0.1 to 0.01 V^{-1}.

At this point, several practical comments should be made: (1) in general, to increase λ, the channel length is made 2 to 5 times the minimum allowable channel L (this was also necessary to reduce the short-channel effects discussed above); (2) the value of λ is normally determined empirically; trying to determine λ from an equation is, in general, not too useful; (3) the output resistance of a MOSFET is a function of the MOSFET's drain current. The exact value of this current is not important when estimating the output resistance. Whether $I_{D,sat}$ (the drain current at $V_{D,sat}$) or the actual operating point current, I_D, is used is not practically important when determining r_o.

MOSFET Transconductance

It is useful to determine how a change in the gate-source voltage changes the drain current of a MOSFET operating in the saturation region. We can relate the change in drain current, i_d, to the change in gate-source voltage, v_{gs}, using the MOSFET transconductance, g_m, or

$$g_m = \frac{\Delta i_D}{\Delta v_{GS}} \Rightarrow i_d = g_m v_{gs} \tag{19.10}$$

Neglecting the output resistance of a MOSFET, we can write the sum of the DC and AC (or changing) components of the drain current and gate-source voltage using

$$i_D = i_d(\text{AC}) + I_D(\text{DC}) = \frac{\beta}{2}\left(V_{GS}(\text{DC}) + v_{gs}(\text{AC}) - V_{THN}\right)^2 \tag{19.11}$$

If we hold the DC values constant and assume they are large compared to the AC components, then by simply taking the derivative of i_D with respect to v_{gs}, we can determine g_m. Doing this results in

$$g_m = \beta(V_{GS} - V_{THN}) = \sqrt{2\beta I_D} \tag{19.12}$$

Following this same procedure for the MOSFET operating in the subthreshold region results in

$$g_m = \frac{I_D \cdot q}{kT} \tag{19.13}$$

Notice how the change in transconductance is linear with drain current when the device is operating in the subthreshold region. The larger incremental increase in g_m is due to the exponential relationship between I_D and V_{GS} when the device is operating in the weak inversion region (as compared to the square law relationship when the device is operating in the strong inversion region).

MOSFET Open-Circuit Voltage Gain

At this point, we can ask, "What's the largest possible voltage gain I can get from a MOSFET under ideal biasing conditions?" Consider the schematic diagram shown in Fig. 19.4(a) without biasing circuitry shown and with the effects of finite MOSFET output resistance modeled by an external resistor. The open-circuit voltage gain can be written as

$$|A| = \frac{v_{out}}{v_{gs}} = g_m r_o = \frac{\sqrt{2\beta I_D}}{\lambda I_D} = \frac{\sqrt{2\beta}}{\lambda\sqrt{I_D}} \text{ (for normal operation)} \tag{19.14}$$

which increases as the DC drain biasing current is reduced (and the MOSFET intrinsic speed decreases) until the MOSFET enters the subthreshold region. Once in the subthreshold region, the voltage gain flattens out and becomes

$$A_v = g_m r_o = \frac{I_D \cdot q}{kT} \cdot \frac{1}{\lambda I_D} = \frac{q}{\lambda \cdot kT} \text{ (in the subthreshold region)} \tag{19.15}$$

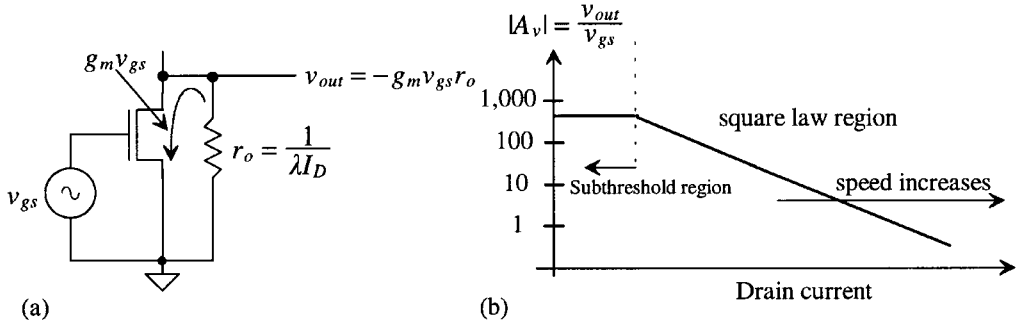


FIGURE 19.4 Open-circuit voltage gain (DC biasing not shown).

Layout of the MOSFET

Figure 19.5 shows the layout of both n-channel and p-channel devices. In this layout, we are assuming that a p-type substrate is used for the body of the NMOS (the body connection for the n-channel MOSFET is made through the p^+ diffusion on the left in the layout). An n-well is used for the body of the PMOS devices (the connection to the n-well is made through the n^+ diffusion on the right in the layout). An important thing to notice from this layout is that the intersection of polysilicon (poly for short) and n^+ (in the p-substrate) or p^+ (in the n-well) forms a MOSFET. The length and width of the MOSFET, as seen in Fig. 19.5, is determined by the size of this overlap. Also note that the four metal connections to the terminals of each MOSFET, in this layout, are floating; that is, not connected to anything but the MOSFETs themselves. It is important to understand, since the p-substrate is common to all MOSFETs on the chip, that this would require the body of the n-channel MOSFET (again the p^+ diffusion on the left in the layout) be connected to *VSS* (ground for most digital applications).

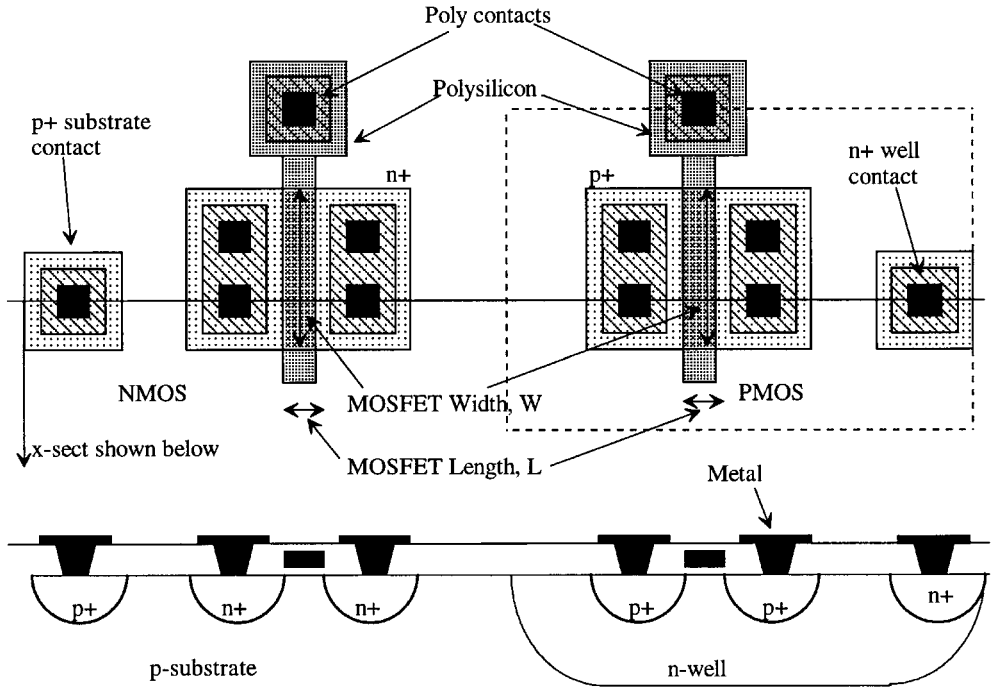


FIGURE 19.5 Layout and cross-sectional views of NMOS and PMOS devices.

19.2 Biasing Circuits

A fundamental component of any CMOS amplifier is a biasing circuit. This section presents important design topologies and considerations used in the design of CMOS biasing circuits. We begin this section with a discussion of current mirrors.

The Current Mirror

The basic CMOS current mirror is shown in Fig. 19.6. For the moment, we will not concern ourselves with the implementation of I_{REF}. By tying M1's gate to its drain, we set V_{GS} at a value given by

$$V_{GS} = V_{THN} + \sqrt{\frac{2I_{REF}}{\beta_1}} \tag{19.16}$$

For most design applications, the term under the square root is set to a few hundred millivolts or less. Because M2's gate-source voltage, and thus its drain current (I_{OUT}), is set by I_{REF}, we say that M2 mirrors the current in M1. If M2's β is different than M1's, we can relate the currents by

$$\frac{I_{OUT}}{I_{REF}} = \frac{2\beta_2(V_{GS} - V_{THN})^2}{2\beta_1(V_{GS} - V_{THN})^2} = \frac{\beta_2}{\beta_1} = \frac{W_2}{W_1} \text{ if } L_1 = L_2 \tag{19.17}$$

Notice that we have neglected the finite output resistance of the MOSFETs in this equation. If we include the effects of finite output resistance, and assume M1 and M2 are sized the same, we see from Fig. 19.6 that the only time I_{REF} and I_{OUT} are equal is when $V_{GS} = V_{OUT} = V_{DS2}$.

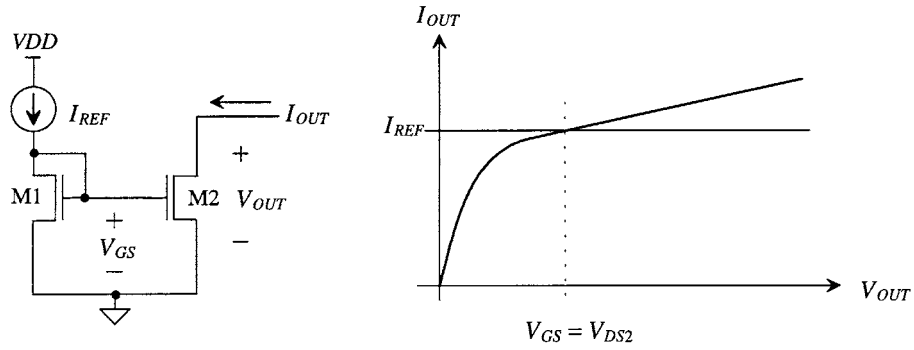


FIGURE 19.6 Basic CMOS current mirror.

Design Example

Design a 100-μA current sink and a 200-μA current source assuming that you are designing with a 0.5-μm CMOS process with KP_n = 150 μA/V², KP_p = 50 μA/V², V_{THN} = 0.8 V, and V_{THP} = 0.9 V. Assume λ was empirically determined (or determined from simulations) to be 0.05 V^{-1} with L = 2.0 μm. Assume that an I_{REF} of 50 μA is available for the design. Determine the output resistance of the current source/sink and the minimum voltage across the current source/sink.

The schematic of the design is shown in Fig. 19.7. If we design so that that term $\sqrt{(2I_{REF})/\beta_1}$ is 300 mV, the width of M1, W_1, is 15 μm (KP_n = 150 μA/V², L = 2 μm). The MOSFET, M1, has a gate-source voltage 300 mV in *excess* of the threshold voltage, or, in other words, the MOSFET V_{GS} is 1.1 V. For M2 and M3 to sink 100 μA (twice the current in M1), we simply increase their widths to 30 μm for the same V_{GS} bias supplied by M1. Note the minimum voltage required across M3, V_{DS3}, in order to keep M3 out of the triode region (V_{DS3} $V_{GS} - V_{THN}$), is simply the *excess gate voltage*, $\sqrt{(2I_{REF})/\beta_1}$, or, for this example, 300 mV. (Note: simply put, 300 mV is the minimum voltage on the drain of M3 required to keep it in the saturation region.) Increasing the widths of the MOSFETs lowers the minimum voltage required across the MOSFETs (lowers the excess gate voltage) so they remain in saturation at the price of increased layout area. Also, differences in MOSFET threshold voltage become more significant, affecting the matching between devices. Note that the output resistance of M3 is simply.

The purpose of M2 should be obvious at this point; it provides a 100-μA bias for the p-channel current mirror M4/M5. Again, if we set M4's excess gate voltage to 300 mV (so that the V_{SG} of the p-channel MOSFET is 1.2 V), the width of M4 can be calculated (assuming that L is 2 μm and KP_p = 50 μA/V²) to be 45 μm (or a factor of 3 times the n-channel width due to the differences in the transconductance

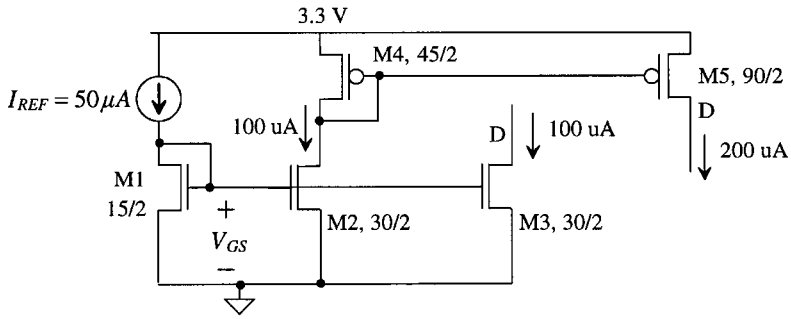


FIGURE 19.7 Design example.

parameters of the MOSFETs). Since the design required a current source of 200 μA, we increase the width of the p-channel, M5, to 90 μm (so that it mirrors twice the current that flows in M4). The output resistance of the current source, M5, is 100 kΩ, while the maximum voltage on the drain of M5 is 3 V (in order to keep M5 in saturation.)

Layout of Current Mirrors

In order to get the best matching between devices, we need to lay the MOSFETs out so that differences in the mirrored MOSFETs' widths and lengths are minimized. Figure 19.8 shows the layout of the M1 (15/2) and M2 (30/2) MOSFETs of Fig. 19.7. Notice how, instead of laying M2 out in a fashion similar to M1 (i.e., a single poly over active strip), we split M2 into two separate MOSFETs that have the same shape as M1.

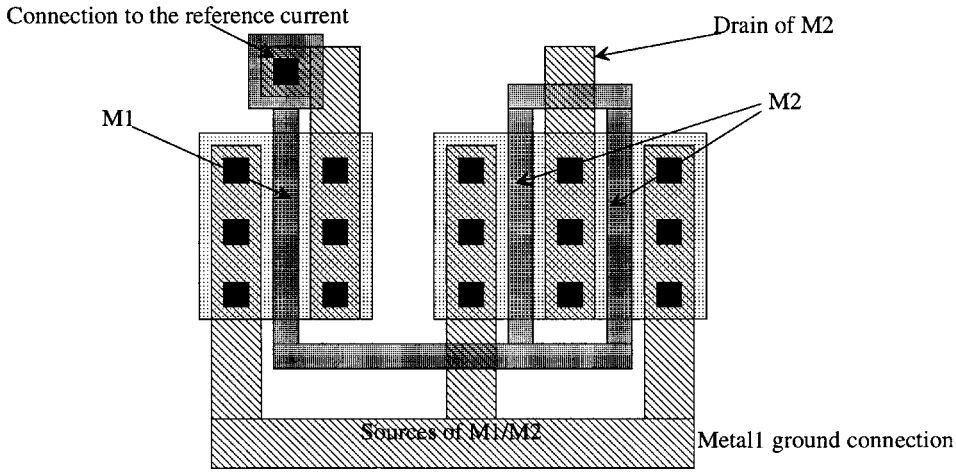


FIGURE 19.8 Layout of MOSFET mirror M1/M2 in Fig. 19.7.

The matching between current mirrors can also be improved using a *common-centroid* layout (Fig. 19.9). Parameters such as the threshold voltage and *KP* in practice can have a somewhat linear change with position on the wafer. This may be the result of varying temperature on the top of the wafer during processing, or fluctuations in implant dose with position. If we lay MOSFETs M2 and M1 out as shown

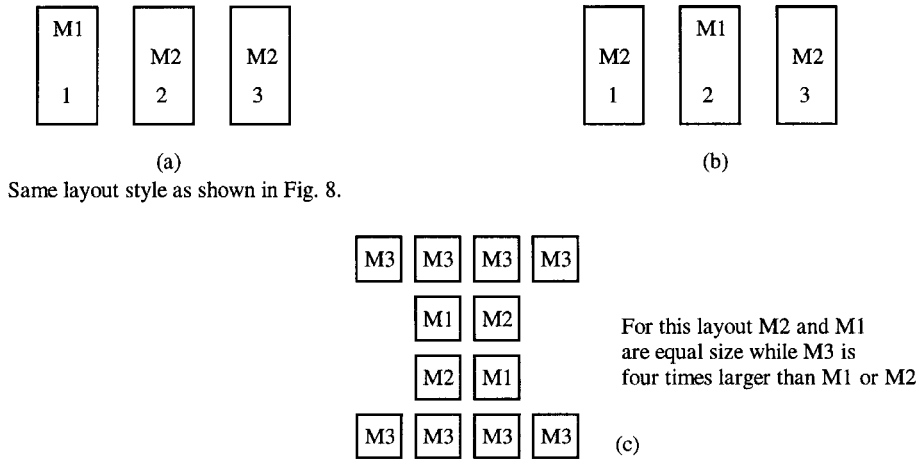


FIGURE 19.9 Common-centroid layout used to improve matching.

in Fig. 19.9(a) (which is the same way they were laid out in Fig. 19.8), M1 has a "weight" of 1 while M2 has a weight of 5. These numbers may correspond to the threshold voltages of three individual MOSFETs with numerical values 0.81, 0.82, and 0.83 V. By using the layout shown in Fig. 19.9(b), M1's or M2's average weight is 2. In other words, using the threshold voltages as an example, M1's threshold voltage is 0.82 V while the average of M2's threshold voltage is also 0.82 V. Similar discussions can be made if the transconductance parameters vary with position. Figure 19.9(c) shows how three devices can be matched using a common-centroid layout (M2 and M1 are the same size while M3 is 4 times their size.) A good exercise at this point is to modify the layout of Fig. 19.9(c) so that M2 is twice the size of M1 and one half the size of M3.

The Cascode Current Mirror

We saw that in order to improve the matching between the two currents in the basic current mirror of Fig. 19.6, we needed to force the drain-source voltage of M2 to be the same as the drain-source voltage of M1 (which is also V_{GS} in Fig. 19.6). This can be accomplished using the cascode connection of MOSFETs shown in Fig. 19.10. The term "cascode" comes from the days of vacuum tube design where a cascade of a common-cathode amplifier driving a common-grid amplifier was used to increase the speed and gain of an overall amplifier design.

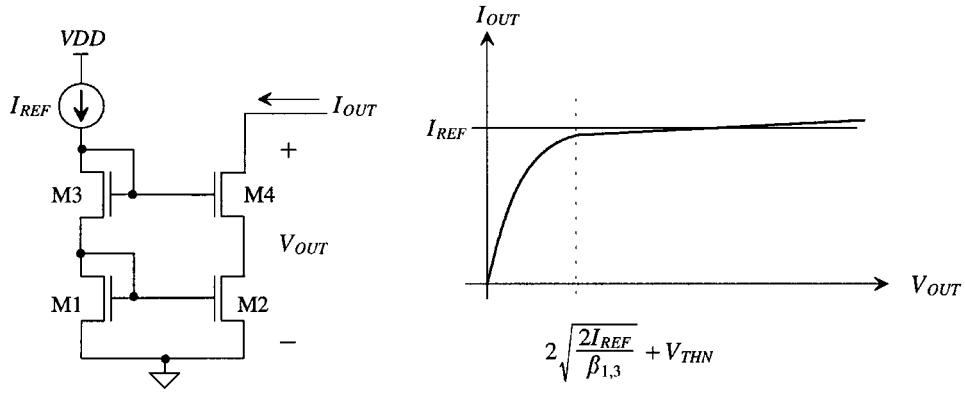


FIGURE 19.10 Basic cascode CMOS current mirror.

Using the cascode configuration results in higher output resistance and thus better matching. For the following discussion, we will assume that M1 and M3 are the same size, as are M2 and M4. Again, remember that M1 and M2 form a current mirror and operate in the same way as previously discussed. The addition of M3 and M4 helps force the drain-source voltages of M1/M2 to the same value. The minimum V_{OUT} allowable, in order to keep M4 out of the triode region, across the current mirror increases to

$$V_{OUT,\,min} = 2\sqrt{\frac{2I_{REF}}{\beta_{1,3}}} + V_{THN} \tag{19.18}$$

which is basically an increase of V_{GS} over the basic current mirror of Fig. 19.6.

The output resistance of the cascode configuration can be derived with the circuit model of Fig. 19.11. Here, we assume that the gates of M4 and M2 are at fixed DC potentials (which are set by I_{REF} flowing through M1/M3). Since the source of M2 is held at ground, we know that the AC component of v_{gs2} is 0. Therefore, we can replace M2 with a small-signal resistance $r_o = (1/\lambda I_{OUT})$. To determine the output resistance of the cascode current mirror, we apply an AC test voltage, v_{test}, and measure the AC test current

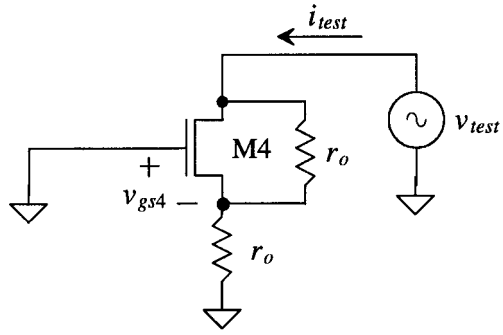


FIGURE 19.11 Determining the small-signal output resistance of the cascode current mirror.

that flows into the drain of M4. Ideally, only the DC component will flow through v_{test}. We can write the AC gate-source voltage of M4 as $v_{gs4} = -i_{test} \cdot r_o$. The drain current of M4 is then $g_{m4}v_{gs4} = -i_{test} \cdot g_{m4}r_o$ while the current through the small-signal output resistance of M4, r_o, is $(v_{test} - (-i_{test}r_o))/r_o$. Combining these equations yields the cascode output resistance of

$$R_{out,\,cascode} = \frac{v_{test}}{i_{test}} = r_o(1 + g_m r_o) + r_o \approx g_m r_o^2 \tag{19.19}$$

The cascode current source output resistance is $g_m r_o$ (the open-circuit voltage gain of a MOSFET) times larger than r_o (the simple current mirror output resistance.) The main drawback of the cascode configuration is the increase in the minimum required voltage across the current sink in order for all MOSFETs to remain in the saturation region of operation.

Low-Voltage Cascode Current Mirror

If we look at the cascode current mirror of Fig. 19.10, we see that the drain of M2 is held at the same potential as the drain of M1, that is, VGS or $\sqrt{(2I_{REF})/\beta} + V_{THN}$. We know that the voltage on the drain of M2 can be as low as $\sqrt{(2I_{REF})/\beta}$ before it starts to enter the triode region. Knowing this, consider the *wide-swing current mirror* shown in Fig. 19.12. Here, "wide-swing" means the minimum voltage across the current mirror is $2\sqrt{(2I_{REF})/\beta}$, the sum of the excess gate voltages of M2 and M4. To understand the operation of this circuit, assume that M1 through M4 have the same W/L ratio (their βs are all equal). We know that the V_{GS} of M1 and M2 is $\sqrt{(2I_{REF})/\beta} + V_{THN}$. It is desirable to keep M2's drain

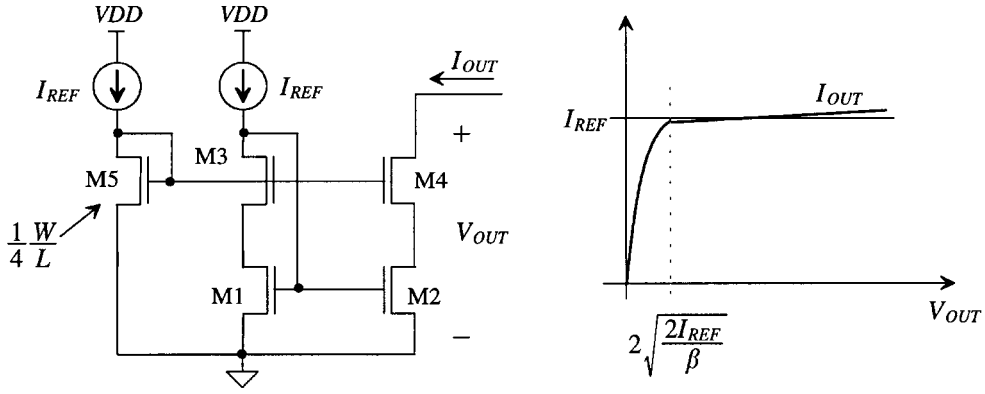


FIGURE 19.12 Wide-swing CMOS cascode current mirror.

at $\sqrt{(2I_{REF})/\beta}$ for wide-swing operation. This means, since M3/M4 are the same size as M1/M2, the gate voltage of M3/M4 must be $V_{GS} + \sqrt{(2I_{REF})/\beta}$ or $2\sqrt{(2I_{REF})/\beta} + V_{THN}$. By sizing M5's channel width so that it is one fourth of the size of the other transistor widths and forcing I_{REF} through the diode connected M5, we can generate this voltage. We should point out that the size (its W/L ratio) of M5 can be further decreased, say to 1/5, in order to keep M1/M2 from entering the triode region (and the output resistance from decreasing). The cost is an increase in the minimum allowable voltage across M2/M4, that is, V_{OUT}.

Simple Current Mirror Biasing Circuits

Figure 19.13 shows two simple circuits useful for generating the reference current, I_{REF}, used in the current mirrors discussed in the previous section. The circuit shown in Fig. 13(a) uses a simple resistor with a gate-drain connected MOSFET to generate a reference current. Note how, by adding MOSFETs mirroring the current in M1 or M2, we can generate any multiple of I_{REF} needed. The reference current, of Fig. 19.13(a), can be determined by solving

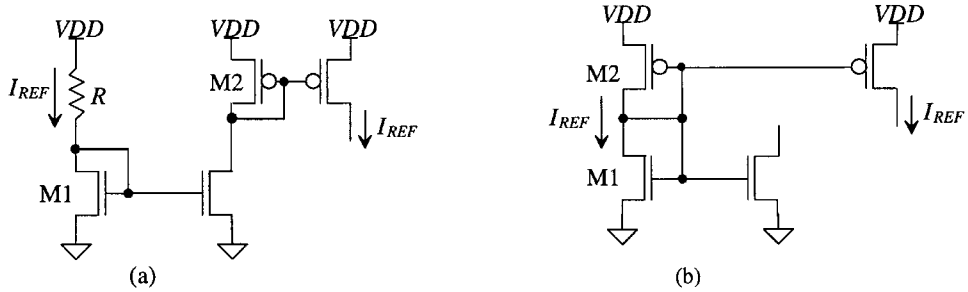


FIGURE 19.13 Simple biasing circuits.

$$I_{REF} = \frac{VDD - V_{GS}}{R} = \frac{\beta}{2}(V_{GS} - V_{THN})^2 \tag{19.20}$$

Figure 19.13(b) shows a MOSFET-only bias circuit. Since the same current flows in M1 and M2, we can mirror off of either MOSFET to generate our bias currents. The current flowing in M1/M2 is designed using

$$I_{REF} = \frac{\beta_1}{2}(V_{GS} - V_{THN})^2 = \frac{\beta_2}{2}(VDD - V_{GS} - V_{THN})^2 \tag{19.21}$$

Notice in both equations above that the reference current is a function of the power supply voltage. Fluctuations, as a result of power supply noise, in *VDD* directly affect the bias currents. In the next section, we will present a method for generating currents that reduces the currents' sensitivity to changes in *VDD*.

Temperature Dependence of Resistors and MOSFETS

Figure 19.14 shows how a resistor changes with temperature, assuming a linear dependence. The temperature coefficient is used to relate the value of a resistor at room temperature, or some known temperature T_0, to the value at a different temperature. This relationship can be written as

$$R(T) = R(T_0) \cdot [1 + TCR(T - T_0)] \tag{19.22}$$

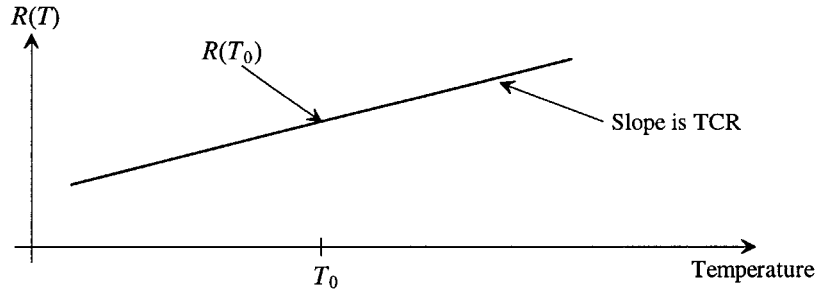


FIGURE 19.14 Variation of a resistor with temperature.

where *TCR* is the temperature coefficient of the resistor *ppm*/°C (parts per million, a multiplier of 10^{-6}, per degree C). Typical values for TCRs for n-well, n^+, p^+, and poly resistors are 2000, 500, 750, and 100 ppm/°C, respectively.

Figure 19.15 shows how the drain current of a MOSFET changes with temperature. At low gate-source voltages, the drain current increases with increasing temperature. This is a result of the threshold voltage decreasing with increasing temperature which dominates the I-V characteristics of the MOSFET. The temperature coefficient of the threshold voltage (NMOS or PMOS), TCV_{TH}, is generally around –3000 ppm/°C. We can relate the threshold voltage to temperature using

$$V_{TH}(T) = V_{TH}(T_0)[1 + TCV_{TH}(T - T_0)] \tag{19.23}$$

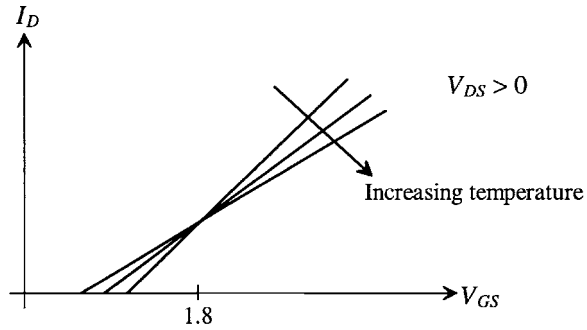


FIGURE 19.15 Temperature characteristics of a MOSFET.

At larger gate-source voltages, the drain current decreases with increasing temperature as a result of the electron or hole mobility decreasing with increasing temperature. In other words, at low gate-source voltages, the temperature changing the threshold voltage dominates the I-V characteristics of the MOSFET; while at larger gate-source voltages, the mobility changing with temperature dominates. Note that at around 1.8 V, for a typical CMOS process, the drain current does not change with temperature. The mobility can be related to temperature by

$$\mu(T) = \mu(T_0)\left(\frac{T}{T_0}\right)^{-1.5} \tag{19.24}$$

The Self-Biased Beta Multiplier Current Reference

Figure 19.16 shows the self-biased beta multiplier current reference. This circuit employs positive feedback, with a gain less than one, to reduce the sensitivity of the reference current to power supply changes.

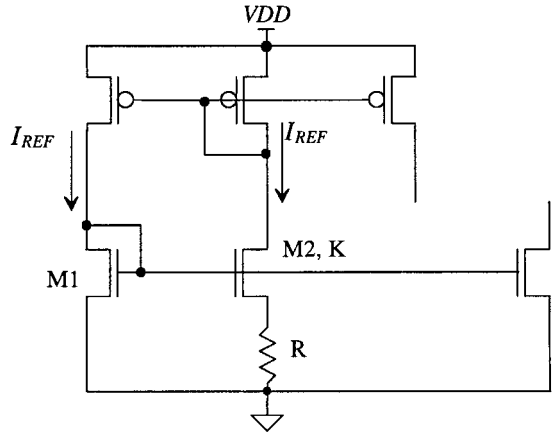


FIGURE 19.16 Beta multiplier current reference.

MOSFET M2 is made K times wider than MOSFET M1 (in other words, $\beta_2 = K\beta_1$; hence, the name beta multiplier). We know from this figure that $V_{GS1} = V_{GS2} + I_{REF}R$, where $V_{GS1} = \sqrt{(2I_{REF})/\beta_1} + V_{THN}$ and $V_{GS2} = \sqrt{(2I_{REF})/K\beta_1} + V_{THN}$; therefore, we can write the current in the circuit as

$$I_{REF} = \frac{2}{R^2\beta_1}\left[1 - \sqrt{\frac{1}{K}}\right]^2 \tag{19.25}$$

which shows no first-order dependence on the power supply voltage.

Start-up Circuit

One of the drawbacks of using the reference circuit of Fig. 19.16 is that it has two stable operating points [Fig. 19.17(a)]. The desirable operating point, point A, occurs when the current flowing in the circuit is I_{REF}. The undesirable operating point occurs when zero current flows in the circuit, point B. Because of the possibility of zero current flowing in the reference, a start-up circuit should always be used when using the beta multiplier. The purpose of the start-up circuit is to ensure that point B is avoided. When designed properly, the start-up circuit [Fig. 19.17(b)] does not affect the beta multiplier operation when I_{REF} is non-zero (M3 is off when operating at point A).

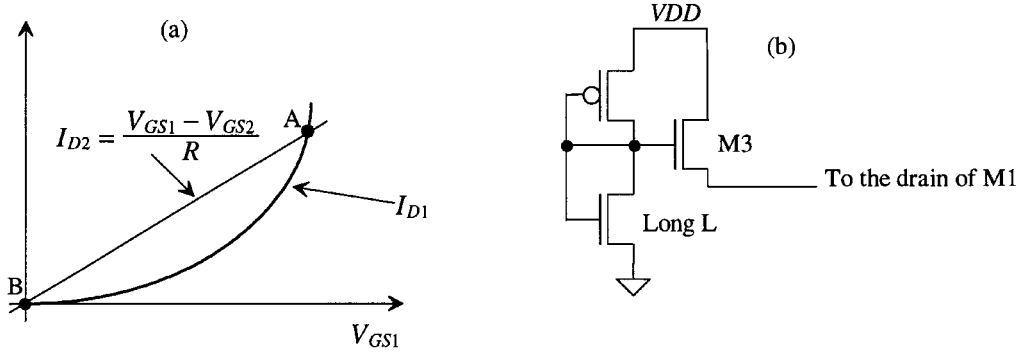


FIGURE 19.17 Start-up circuit for the beta multiplier circuit shown in Fig. 19.16.

A Comment About Stability

Since the beta multiplier employs positive feedback, it is possible that the circuit can become unstable and oscillate. However, with the inclusion of the resistor in series with the source of M2, the gain around the loop, from the gate of M2 to the drain/gate of M1, with the loop broken between the gates of M1 and M2, is less than one, keeping the reference stable. Adding a large capacitance across *R*, however, can increase the loop gain to the point of instability. This situation could easily occur if *R* is bonded out to externally set the current.

19.3 Amplifiers

Now that we have introduced biasing circuits and MOS device characteristics, we will immediately dive into the design of operational amplifiers. Since space is very limited, we will employ a top-down approach in which a series of increasingly complex circuits are dissected stage by stage and individual blocks analyzed.

Operational amplifiers typically are composed of either two or three stages consisting of a differential amplifier, a gain stage and an output stage as seen in Fig. 19.18. In some applications, the gain stage and the output stage are one and the same if the load is purely capacitive. However, if the output is to drive a resistive load or a large resistive load, then a high current gain buffer amplifier is used at the output. Each stage plays an important role in the performance of the amplifier.

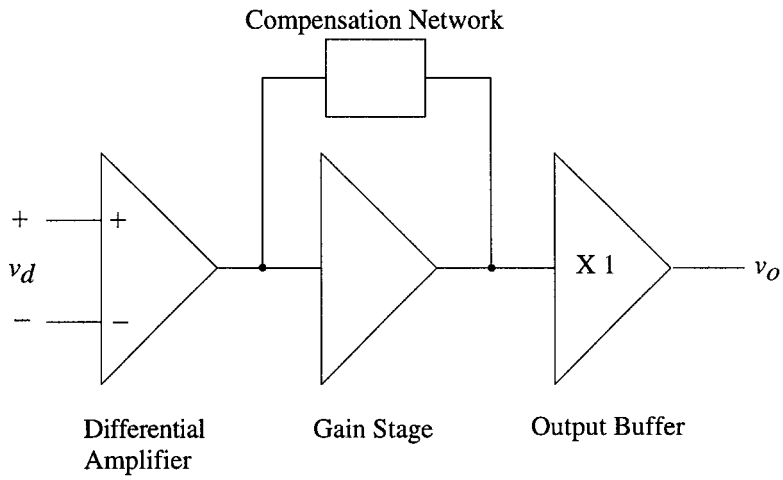


FIGURE 19.18 Block diagram for a generic op-amp.

The differential amplifier offers a variety of advantages and is always used as the input to the overall amplifier. Since it provides common-mode rejection, it eliminates noise common on both inputs, while at the same time amplifying any differences between the inputs. The limit for which this common mode rejection occurs is called *common-mode range* and signifies the upper and lower common mode signal values for which the devices in the diff-amp are saturated. The differential amplifier also provides gain. The gain stage is typically a common-source or cascode type amplifier. So that the amplifier is stable, a compensation network is used to intentionally lower the gain at higher frequencies. The output stage provides high current driving capability for either driving large capacitive or resistive loads. The output stage typically will have a low output impedance and high signal swing characteristics. In some cases, it may be advantageous to add bipolar devices to improve the performance of the circuitry. These will be presented as the block level circuits are analyzed.

The Simple Unbuffered Op-Amp

Examine the simple operational amplifier shown in Fig. 19.19. Here, the amplifier can be segregated into a biasing block, a differential input stage, an output stage, and a compensation capacitor.

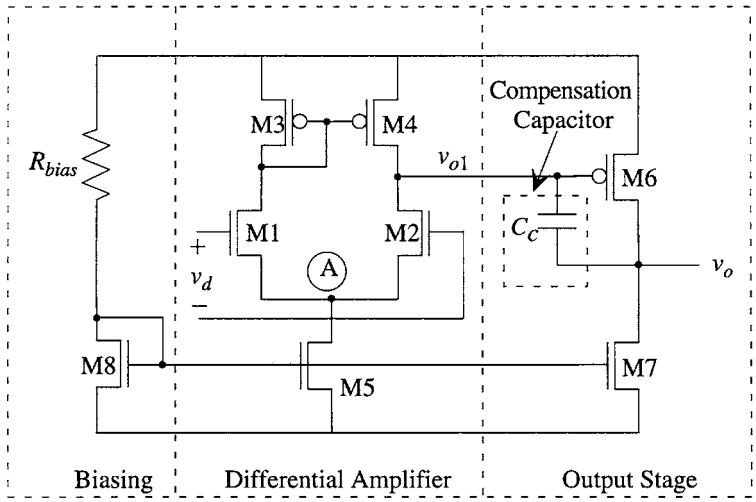


FIGURE 19.19 Basic two-stage op-amp.

The biasing circuit is a simple current mirror driver, consisting of the resistor R_{bias} and the transistor M8. The current through M8 is mirrored through both M5 and M7. Thus, the current through the entire circuit is set by the value of R_{bias} and the relative W/Ls of M8, M5, and M7. The actual values of this current will be discussed a little bit later on. When designing with R_{bias}, one must be careful not to ignore the effect of temperature on R_{bias}, and thus the values of the currents through the circuit. We will see later on how the bias current greatly affects the performance of the amplifier. For the commercial temperature range of 0°C to 125°C, the current through M8 should be simulated with R_{bias} at values of ±30 % of its nominal value. Other, more sophisticated voltage references (as discussed earlier) can be used in place of the resistor reference, and will be presented as we progress.

The Differential Amplifier

The differential amplifier is composed of M1, M2, M3, M4, and M5, with M1 matching M2 and M3 matching M4. The transistor M5 can be replaced by a current source in the ideal case to enhance one's understanding of the circuit's functionality. The node labeled as node A can be thought of as a virtual ground, since the current through M5 is constant and thus the voltage at node A is also constant.

Now assume that the gate of M2 is tied to ground as seen in Fig. 19.20. Any small signal on the gate of M1 will result in a small signal current i_{d1}, which will flow from drain to source of M1 and will also be mirrored from M3 to M4. Note that since M5 can be thought of as an ideal current source, it can be replaced with its ideal impedance of an open circuit. Therefore, i_{d1} will also flow from source to drain of M2. Remember that the small signal current is different from the DC current flowing through M2. The small signal current can be thought of as the current generated from the instantaneous (AC + DC) voltage at the output. If the instantaneous voltage at the output node swings up, then the small signal current through M2 will flow from drain to source. However, if the instantaneous voltage swings down, the change in voltage will be considered negative and thus the small signal current will assume an identical polarity, thus flowing from source to drain of M2. The small signal voltage produced at the output node will then be $2i_{d1}$ times the output impedance of the amplifier. In this case, R_{out} is simply $r_{o2}||r_{o4}$. The value of the small signal current, i_{d1}, is simply

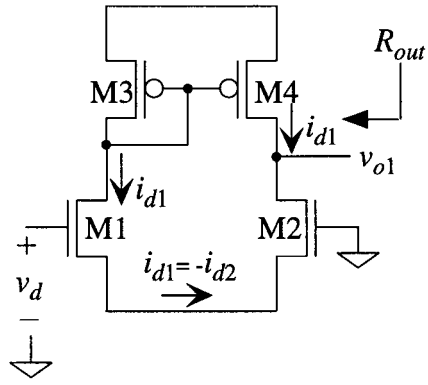


FIGURE 19.20 Pseudo-AC circuit showing small signal currents.

$$i_{d1} = g_{m1} v_{gs1} \tag{19.26}$$

and the differential voltage, $v_d = v_{gs1} + v_{sg2}$. Therefore, since $v_{gs1} = v_{gs2}$, then,

$$\frac{i_{d1}}{v_d} = \frac{g_{m1}}{2} \tag{19.27}$$

the small signal output voltage is simply

$$v_{o1} = 2 \cdot i_{d1} \cdot r_{o2} \| r_{o4} \tag{19.28}$$

and the small signal voltage gain can then be easily derived as

$$\frac{v_{o1}}{v_d} = g_{m1,2}(r_{o2} \| r_{o4}) \tag{19.29}$$

Now let us examine this equation more carefully. Substituting the expressions for g_m and r_o, the previous equation becomes

$$\frac{v_{o1}}{v_d} \approx \sqrt{2\beta_{1,2} I_{D1,2}} \cdot \frac{1}{2\lambda I_{D1,2}} = K' \cdot \sqrt{\frac{W_{1,2}}{L_{1,2} I_{D1,2}}} \cdot \frac{1}{\lambda} \tag{19.30}$$

where K' is a constant, which is uncontrollable by the designer. When designing analog circuits, it is just as important to understand the effects of the controllable variables on the specification as it is to know the absolute value of the gain using hand calculations. This is because the hand analysis and computer simulations will vary a great deal because of the complex modeling used in today's CAD tools. Examining Eq. (19.30), and knowing that the effect of λ on the gain diminishes as L increases such that $1/\lambda$ is directly proportional to channel length. Then, a proportionality can be established between $W/L_{1,2}$ and the drain current versus the small signal gain such that

$$\frac{v_{o1}}{v_d} \propto \sqrt{\frac{W_{1,2} \cdot L_{1,2}}{I_{D1,2}}} \tag{19.31}$$

Notice that the constant was not included since the value is not dependent on anything the designer can adjust. The importance of this equation tells us that the gain is proportional to the square root of the

product of W and L and inversely proportional to the square root of the drain current through M1 and M2, which is also 1/2 I_{D6}. So to increase the gain, one must increase W or L or decrease the value I_{D6}, which is dependent on the value of R_{bias}.

The Gain Stage

Now examine the output stage consisting of M6 and M7. Here, the driving transistor, M6, is a simple inverting amplifier with a current source load as seen in equivalent circuit shown in Fig. 19.21. The value of the small signal current is defined by the AC signal v_{o1}, which is equal to v_{sg6}. Therefore,

$$\frac{i_{d6}}{v_{o1}} = -g_{m6} \tag{19.32}$$

and the gain of the stage is simply

$$\frac{d6}{v_{o1}} \cdot \frac{v_o}{i_{d6}} = \frac{v_o}{v_{o1}} = -g^{m6} \cdot r_{o6} \| r_{o7} \tag{19.33}$$

Again, we can write the gain in terms of the designer's variables, so that the gain of the amplifier can be expressed as a proportion of

$$\frac{v_o}{v_{o1}} \propto \sqrt{\frac{W_6 \cdot L_{6,7}}{I_{D6,7}}} \tag{19.34}$$

Therefore, overall, the gain of the entire amplifier is

$$\frac{v_o}{v_d} = g_{m1,2}(r_{o2} \| r_{o4}) \cdot -g^{m6}(r_{o6} \| r_{o7}) \tag{19.35}$$

or as the proportionalities

$$\frac{v_o}{v_d} \propto \sqrt{\frac{W_{1,2} \cdot L_{1,2}}{I_{D1,2}}} \cdot \sqrt{\frac{W_6 \cdot L_{6,7}}{I_{D6,7}}} \tag{19.36}$$

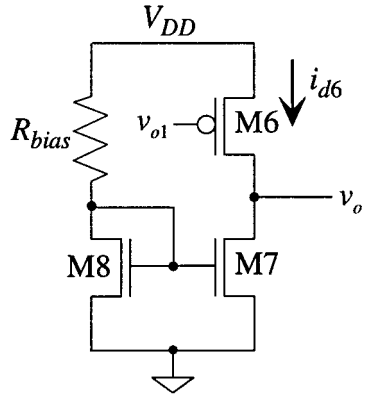


FIGURE 19.21 Output stage circuit.

So, the key variables for adjusting gain are the drain currents (the smaller the better) and the W and L ratios of M1, M2, and M6 (the larger the better). Of course, there are lower and upper limits to the drain currents and the W/Ls, respectively, that we will examine as we analyze other specifications that will ultimately determine the bounds of adjustability.

Frequency Response

Now examine the amplifier circuit shown in Fig. 19.22. In this particular circuit, the compensation capacitor, C_C, is removed and will be re-added shortly. The capacitors C_1 and C_2 represent the total lumped capacitance from each ground. Since the output nodes associated with each output is a high impedance, these nodes will be the dominant frequency-dependent nodes in the circuit. Since each node in a circuit contributes a high-frequency pole, the frequency response will be dominated by the high-impedance nodes.

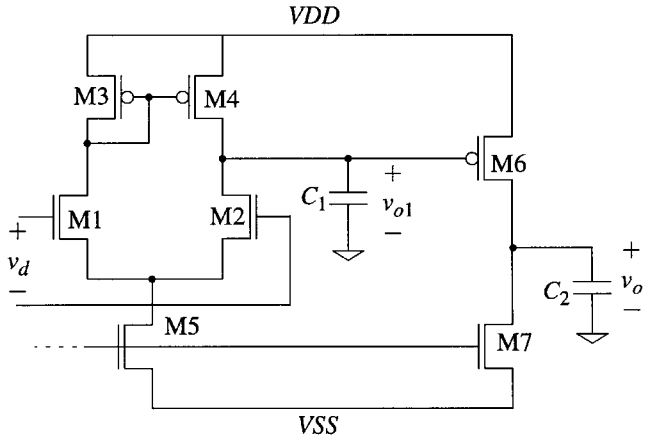


FIGURE 19.22 Two-stage op-amp with lumped parasitic capacitors.

An additional capacitor could have been included at the gate of M4 to ground. However, the equivalent impedance from the gate of M4 to ground is approximately equal to the impedance of a gate-drain connected device or $1/g_{m3}$ as seen in Fig. 19.23 (a)–(c). If a controlled source has the controlling voltage directly across its terminals [Fig. 19.23(b)], then the effective resistance is simply the controlling voltage

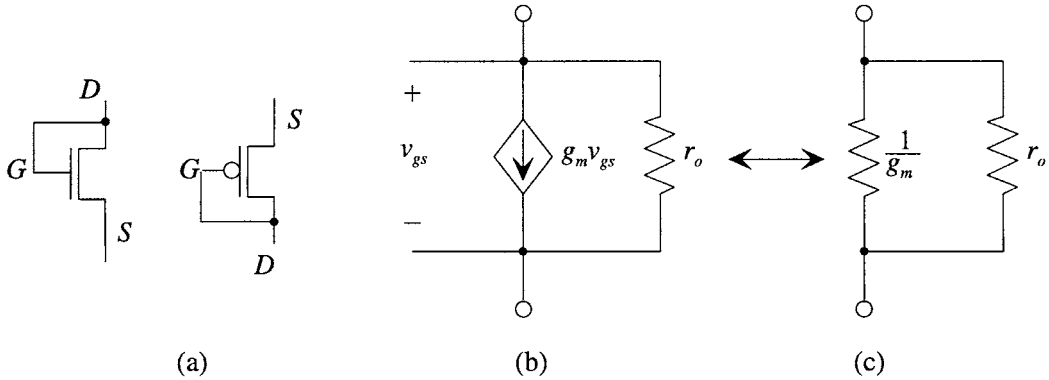


FIGURE 19.23 Equivalent resistance for a gate-drain connected MOSFET device.

(in this case, v_{gs}) divided by the controlled current ($g_m v_{gs}$), which is $1/g_m || r_o$ or approximately $1/g_m$ if r_o is much greater than $1/g_m$ [Fig. 19.23(c)]. Therefore, the impedance seen from the gate of M4 to ground is low, and the pole associated with the node will be at a much higher frequency than the high-impedance nodes. The same holds true with the node associated with the drain of M5. That node is considered an AC ground, so it has no effect on the frequency response of the small signal circuit. The only remaining node is the node which defines the current mirrors (the gate of M8). Since this node is not in the small signal path, and is a DC bias voltage, it can be also be considered to be an AC ground.

One can approximate the frequency response of the amplifier by examining both the effective impedance and parasitic capacitance at the output of each stage. The parasitic capacitances can be seen in the small signal model of a MOSFET in Fig. 19.24. The capacitors C_{gb}, C_{sb}, and C_{db} represent the bulk depletion capacitors of the transistors, while C_{gd} and C_{gs} represent the overlap capacitances from gate to drain and gate to source, respectively. Referring to Fig. 19.25, which shows the parasitic capacitors explicitly, C_1 can now be written as

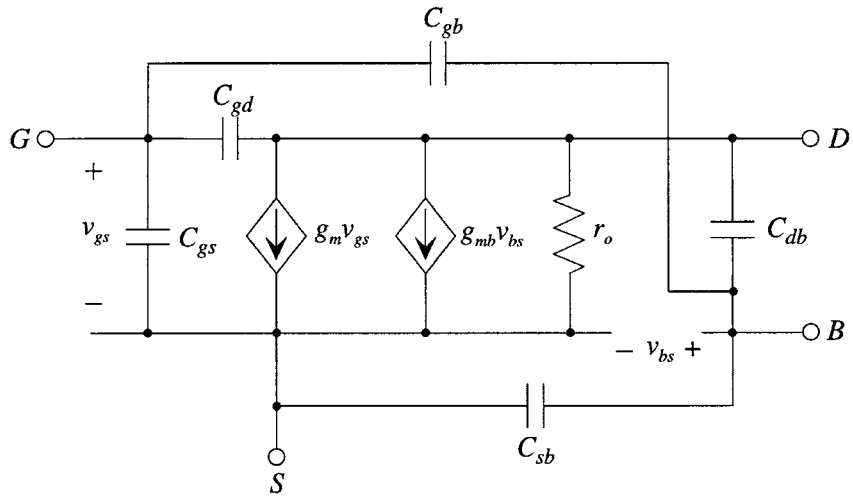


FIGURE 19.24 High-frequency, small signal model with parasitic capacitors.

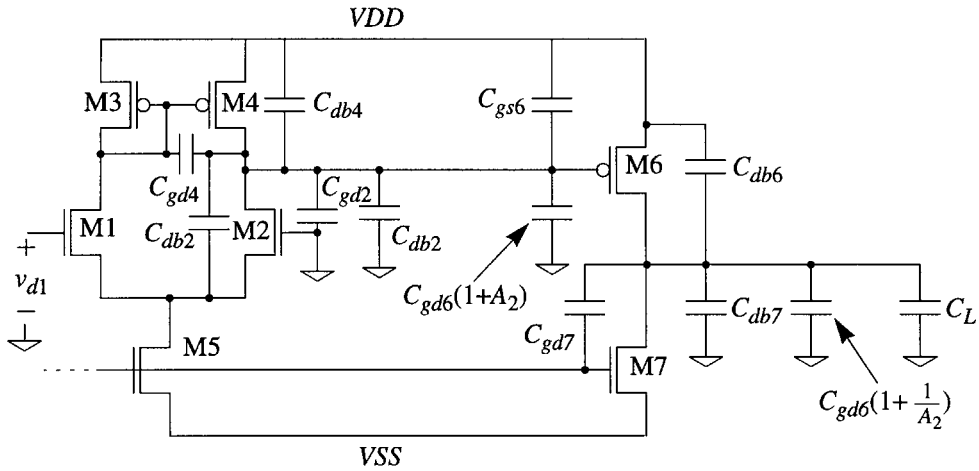


FIGURE 19.25 Two-stage op-amp with parasitics shown explicitly.

$$C_1 = C_{db4} + C_{gd4} + C_{db2} + C_{gd2} + C_{gs6} + C_{gd6}(1 + A_2) \tag{19.37}$$

Note that C_{gd4} is included as capacitor to ground since the low impedance caused by the gate-drain device of M3 can be considered equivalent to an AC ground. The capacitor C_{db2} is also connected to AC ground at the source-coupled node consisting of M1 and M2.

Miller's theorem was used to determine the effect of the bridging capacitor C_{db6}, connected from the gate to the drain of M6. Miller's theorem approximates the effects of the gate-drain capacitor by replacing the bridging capacitor with an equivalent input capacitor of value $C_{db6}\cdot(1 - A_2)$ and an equivalent output capacitor with a value of $C_{db6}\cdot(1 - 1/A_2)$. The term A_2 is the gain across the original bridging capacitor and is $-g_{m6}\cdot r_{o6}||r_{o7}$. The reader should consult Ref. 2 for a proof of Miller's theorem.

The capacitor C_2 can also be determined by examining Fig. 19.25,

$$C_2 = C_{db6} + C_{db7} + C_{gd7} + C_{gd6}\cdot\left(1 + \frac{1}{A_2}\right) + C_L \tag{19.38}$$

Now assume that C_1 is greater than C_2. This means that the pole associated with the diff-amp output will be lower in frequency than the pole associated with the output of the output stage. Thus, a good model for the op-amp can be seen in Fig. 19.26. The transfer function, ignoring C_C, will then become

$$\frac{v_o(s)}{v_{d1}(s)} = -g_{m1}(r_{o2}||r_{o4})\cdot g_{m6}(r_{o6}||r_{o7})\cdot\frac{1}{\left(j\frac{f}{f_{p1}} + 1\right)\left(\frac{f}{f_{p2}} + 1\right)} \tag{19.39}$$

where

$$f_{p1} = \frac{1}{2\pi R_{out1}C_1}, f_{p2} = \frac{1}{2\pi R_{out}C_2} \tag{19.40}$$

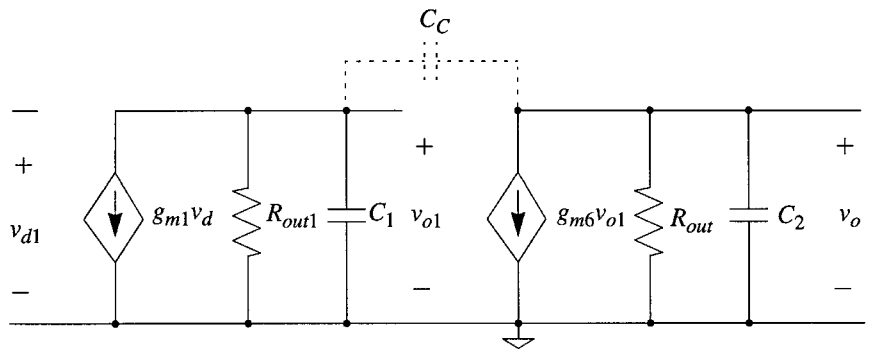


FIGURE 19.26 Model used to determine the frequency response of the two-stage op-amp.

The plot of the transfer function can be seen in Fig. 19.27. Note that in examining the frequency response, that the phase margin is virtually zero. Remember that phase margin is the difference between phase at the frequency at which the magnitude plot reaches 0 dB (also known as the gain-bandwidth product) and the phase at the frequency at which the phase has shifted –180°. It is recommended for stability reasons, that the phase margin of any amplifier be at least 45° (60° is recommended). A phase

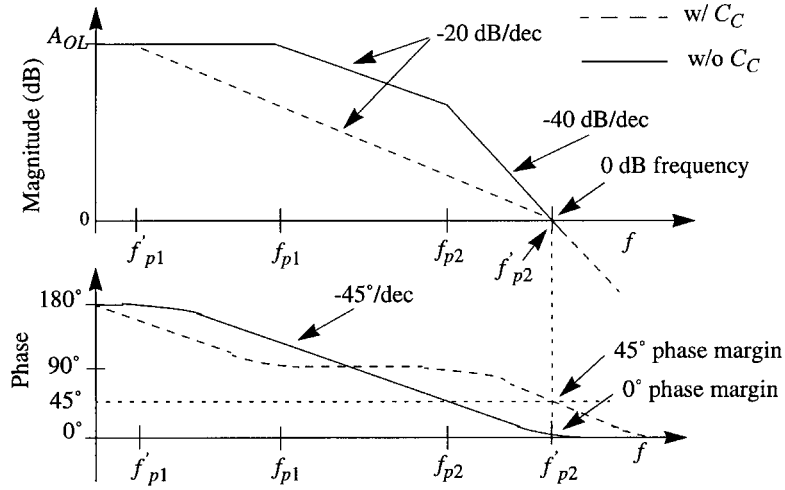


FIGURE 19.27 Magnitude and phase of the two-stage op-amp with and without compensation.

margin below 45° will result in long settling times and increased propagation delays. The system can also be thought of as a simple second-order linear controls system with the phase margin directly affecting the transient response of the system.

Compensation

Now we will include C_C in the circuit. If C_C is much greater than C_{gd6}, then the C_C will dominate the value of C_1 [especially since it is multiplied by $(1 - A_2)$] and will cause the pole, f_{p1}, to roll-off much earlier than without C_C to a new location, f_{p1}. One could solve, using circuit analysis, the circuit shown in Fig. 19.26 with C_C included to also prove that the second pole, f_{p2}, moves further out[3] to a higher frequency, f_{p2}. Ideally, the addition of C_C will cause an equivalent single pole roll-off of the overall frequency response. The second pole should not begin to affect the frequency response until after the magnitude response is below 0 dB. The new values of the poles are

$$f'_{p1} \approx \frac{1}{2\pi(g_{m6}R_{out})C_C R_{out1}}, f'_{p2} \approx \frac{g_{m6}C_C}{2\pi \cdot (C_2C_1 + C_2C_C + C_CC_1)} \approx \frac{g_{m6}}{2\pi \cdot C_2} \qquad (19.41)$$

If the previously discussed analysis was performed, one would also see that by using Miller's theorem, we are neglecting a right-hand plane (RHP) zero that could have negative consequences on our phase margin, since the phase of an RHP zero is similar to the phase of a left-hand plane (LHP) zero. An RHP zero behaves similarly to an LHP zero when examining the magnitude response; however, the phase response will cause the phase plot to shift –180° more quickly. The RHP zero is at a value of

$$f_{z1} = \frac{g_{m6}}{C_C} \qquad (19.42)$$

To avoid effects of the RHP zero, one must try to move the zero out well beyond the point at which the magnitude plot reaches 0 dB (suggested rule of thumb: factor of 10 greater). The comparison of the frequency response of the two-stage op-amp with and without C_C can be seen in Fig. 19.27.

One remedy to the zero problem is to add a resistor, R_z, in series with compensation capacitor as seen in Fig. 19.28. The expression of the zero after adding the resistor becomes

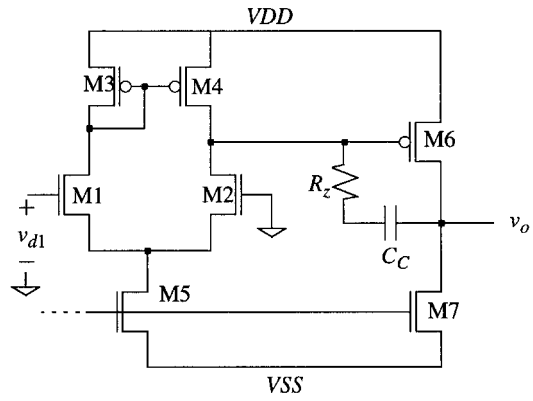


FIGURE 19.28 Compensation including a nulling resistor.

$$f_{z1} = \frac{1}{C_C\left(\frac{1}{g_{m6}} - R_Z\right)} \tag{19.43}$$

and the zero can be pushed into the LHP where it adds phase shift and increases phase margin if $R_z > 1/g_{m6}$. Fig. 19.29[4] shows the root locus plot for the zero as R_z is introduced. With $R_z = 0$, the zero location is on the RHP real axis. As R_z increases in value, the zero gets pushed to infinity at the point at which $R_z = 1/g_{m6}$. Once $R_z > 1/g_{m6}$, the zero appears in the LHP where its phase shift adds to the overall phase response, thus improving phase margin. This type of compensation is commonly referred to as *lead compensation,* and is commonly used as a simple method for improving the phase margin. One should be careful about using R_z, since the absolute values of the resistors are not well predicted. The value of the resistor should be simulated over its maximum and minimum values to ensure that no matter if the zero is pushed into the LHP or the RHP, that the value of the zero is always 10 times greater than the gain-bandwidth product.

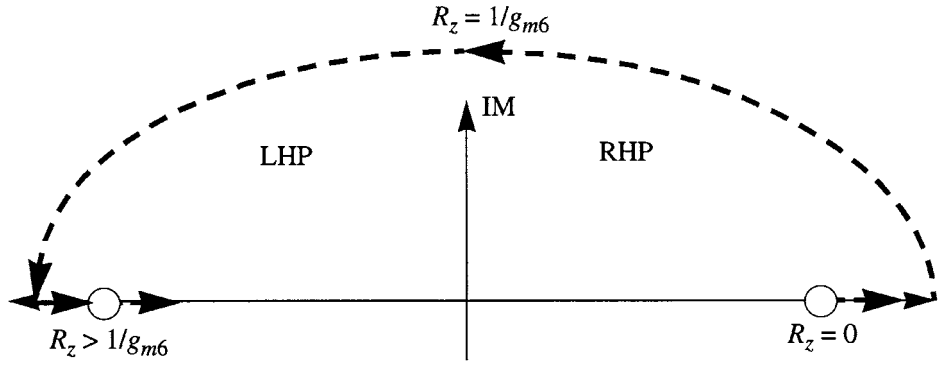


FIGURE 19.29 Root locus plot of how the zero shifts from the RHP to the LHP as R_z increases in value.

Other AC Specifications

Referring back to the equations for gain and the first pole location, we can now see adjustable parameters for affecting frequency response by plugging in the proportionalities for their corresponding factors. Since the value of the resistors are directly proportional to the length of L (the longer the L, the higher the resistance, since longer channel lengths diminish the effects of channel length modulation, λ).

The *gain-bandwidth product* for the compensated op-amp is the open-loop gain multiplied by the bandwidth of the amplifier (as set by f_{p1}). Therefore, the gain-bandwidth product is

$$GBW = (g_{m1} \cdot r_{o2} || r_{o4})(g_{m6} \cdot r_{o6} || r_{o7}) \left(\frac{1}{2\pi (g_{m6} R_{out}) C_C R_{out1}} \right) = \frac{g_{m1}}{2\pi C_C} \tag{19.44}$$

Or we can write the expression as

$$GBW \propto \frac{\sqrt{\frac{W}{L}_{1,2} I_{D1,2}}}{C_C} \tag{19.45}$$

which shows that the most efficient way to increase *GBW* is to decrease C_C. One could increase the $W/L_{1,2}$ ratio, but its increase is only by the square root of $W/L_{1,2}$. Increasing $I_{D1,2}$ will also yield an increase (also to the square root) in *GBW*, but one must remember that it simultaneously decreases the open-loop gain. The value of C_C must be large enough to affect the initial roll-off frequency as a larger C_C improves phase margin. One could easily determine the value of C_C by first knowing the gain-bandwidth specification, then iteratively choosing values for $W/L_{1,2}$ and $I_{D1,2}$ and then solving for C_C.

One other word of warning: the designer must not neglect the effects of loading on the circuit. Practically speaking, the load capacitor usually dominates the value of the capacitor, C_2 in Fig. 19.22. The value of the load capacitor directly affects the phase shift through the amplifier by changing the pole location associated with the output node. The second pole has a value of approximately $g_{m6}/2\pi C_L$ as was seen earlier. Since the second pole needs to be greater than the gain-bandwidth product, the following relationship can be deduced:

$$\frac{g_{m6}}{C_L} > \frac{g_{m1,2}}{C_C} \tag{19.46}$$

or

$$C_C > \frac{g_{m1,2}}{g_{m6}} \cdot C_L \tag{19.47}$$

Thus, one can see the effect of C_L on phase margin in that the minimum size of the compensation capacitor is directly dependent on the size of the load capacitor. For a phase margin of 60°, it can be shown[3] that the second pole must be 2.2 times greater than the *GBW*.

The *common-mode rejection ratio* (CMRR) measures how well the amplifier can reject signals common to both inputs and is written by

$$\text{CMRR} = 20\log\left|\frac{A_d}{A_{cm}}\right| = 20\log\left|\frac{v_o / v_d}{v_o / v_{cm}}\right| \tag{19.48}$$

where V_{cm} is a common mode input signal, which in this case is composed of an AC and a DC component (the higher the value of CMRR, the better the rejection). The circuit for calculating common mode gain can be seen in Fig. 19.30. The gain through the output stage will be the same for both the differential gain, A_d, and the common-mode gain, A_{cm}. The last stage will cancel itself out in the expression for CMRR; thus, the differential amplifier will determine how well the entire amplifier rejects common mode signals. This rejection is one of the most advantageous reasons for using a diff-amp as an input stage. If the inputs are subjected to the same noise source, the diff-amp has the ability to reject the noise signal and only amplify the difference in the inputs. For the diff-amp used in this example, the common-mode gain is

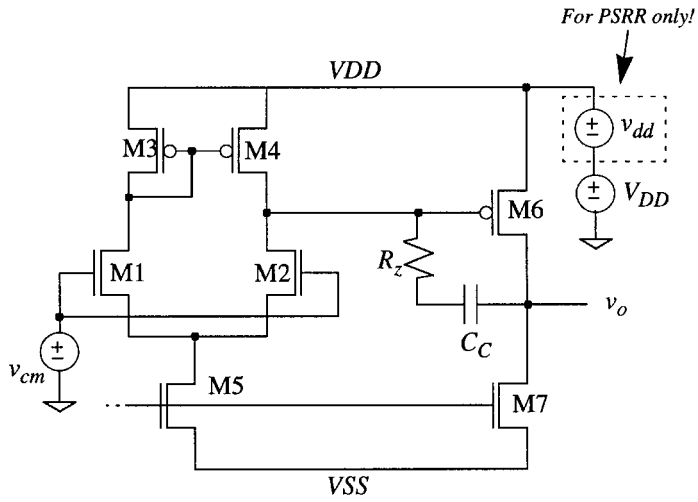


FIGURE 19.30 Circuit used to determine CMRR and PSRR.

$$\frac{v_{o1}}{v_{cm}} = -\frac{1}{2g_{m4}r_{o5}} \tag{19.49}$$

This equation bears some explanation. Since the inputs are tied together, the source-coupled node can no longer be considered an AC ground. Therefore, the resistance of the tail current device, M5, must be considered in the analysis. The AC currents flowing through both M1 and M2 are equal and $v_{gs3} = v_{gs4}$. The output impedance of the diff-amp will drop considerably when using a common-mode signal due to the feedback loop consisting of M1, M3, M4, and M2. As a result, the common-mode signal appearing on the drains of M3 and M4 will be identical.

One can determine this gain by using half-circuit analysis (Fig. 19.31). This is equivalent to a common source amplifier with a large source resistance of value $2r_{o5}$. When using half-circuit analysis, the current source resistance doubles because the current through the driving device M1 is one half of the original tail current. Therefore, the gain of this circuit can be approximately determined as the negative ratio between the resistance attached to the drain of M1 divided by the resistance attached to the source of M1, or

$$\frac{v_{o1}}{v_{cm}} = -\frac{1/g_{m3,4}}{2r_{o5}} \tag{19.50}$$

Adding the expression for the differential gain of the diff-amp, v_{o1}/v_d, we can write the expression for the CMRR as

$$\text{CMRR} = 20\log(2g_{m1,2}g_{m3,4}(r_{o2}||r_{o4})r_{o5}) \tag{19.51}$$

or as a proportion,

$$\text{CMRR} \propto 20\log\left(\sqrt{\frac{W_{1,2}\cdot L_{1,2}}{I_{D1,2}}}\cdot\sqrt{\frac{(W/L)_{3,4}}{2I_{D5}}}\cdot L_5\right) \tag{19.52}$$

So, it can be seen that the most efficient manner in which to increase the CMRR of this amplifier is to increase the channel length of M5 (the tail current device). This, too, has a large signal implication that we will discuss later in this section.

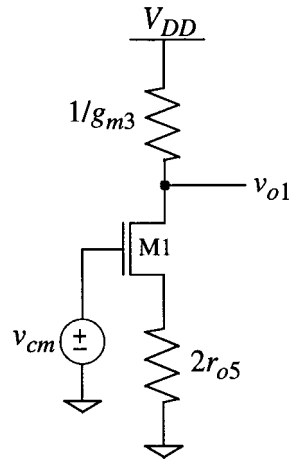


FIGURE 19.31 Half-circuit used to determine the common-mode gain.

The *power supply rejection ratio* (PSRR) measures how well the amplifier can reject changes in the power supply. This is also a critical specification because it would be desirable to reject noise on the power supply outright. PSRR from the positive supply is defined as:

$$\text{PSRR}^{v_{dd}} = \frac{v_o / v_d}{v_o / v_{dd}} \tag{19.53}$$

where the gain v_o/v_{dd} is the small signal gain from v_{dd} (refer back to Fig. 19.30) to the output of the amplifier with the input signal, v_d, equal to zero. PSRR can also be measured from V_{SS} by inserting a small signal source in series with ground. One should be careful, however, when simulating this specification to make sure that the inputs are properly biased so as to ensure that they are in saturation before simulating. It is best to use a fully differential (differential input and differential output) to most effectively reject power supply noise (to be discussed later).

Large Signal Considerations

Other considerations that must be discussed are the large signal tradeoffs. One cannot ignore the effects of adjusting the small signal specifications on the large signal characteristics. The large signal characteristics that are important include the common-mode range, slew rate, and output signal swing.

Slew rate is defined as the maximum rate of change of the output voltage due to a change in the input voltage. For this particular amplifier, the maximum output voltage is ultimately limited by how fast the tail current device (M5) can charge and discharge the compensation capacitor. The slew rate can then be approximated as

$$SR = \frac{dVo}{dt} \approx \frac{I_{D5}}{C_C} \tag{19.54}$$

Typically, the diff-amp is the major limitation when considering slew rate. However, the tradeoff issues again come into play. If I_{D5} is increased too much, the gain of the diff-amp may decrease below a satisfactory amount. If C_C is made too small, then the phase margin may decrease below an acceptable amount.

The *common-mode range* is defined as the range between the maximum and minimum common-mode voltages for which the amplifier behaves linearly. Referring to Fig. 19.32, suppose that the common-mode voltage is DC value and that the differential signal as is also shown. If the common-mode voltage is swept

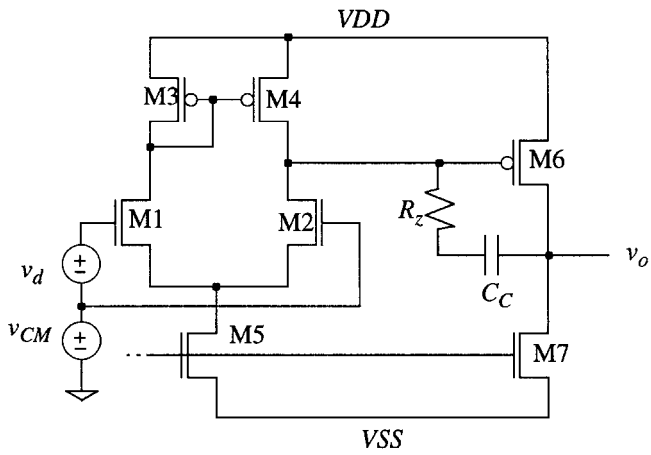


FIGURE 19.32 Determining the CMR for the two-stage op-amp.

from ground to V_{DD}, there will be a range for which the amplifier will behave normally and where the gain of the amplifier is relatively constant. Above or below that range, the gain drops considerably because the common-mode voltage forces one or more devices into the triode region.

The maximum common-mode voltage is limited by both M1 and M2 going into triode. This point can be defined by a borderline equation in which $V_{DS1,2} = V_{GS1,2} - V_{THN}$ or, in this case, $V_{D1,2} = V_{G1,2} - V_{THN}$. Substituting $V_{DD} - V_{SG3}$ for $V_{D1,2}$, and solving for $V_{G1,2}$, which now represents the maximum common-mode voltage, the expression becomes

$$V_{G1,\,2(max)} = V_{DD} - \left[\sqrt{\frac{I_{D5}}{\beta_3}} + V_{THP}\right] + V_{THN} \tag{19.55}$$

where the value of V_{SG3} is written in terms of its drain current using the saturation equation. If the threshold voltages are assumed to be approximately the same value, then the equation can be written as

$$V_{G1,\,2(max)} = V_{DD} - \sqrt{\frac{I_{D5}}{\beta_3}} = V_{DD} - \sqrt{\frac{L_3 \cdot I_{D5}}{W_3 \cdot K_3}} \tag{19.56}$$

The minimum voltage is limited by M5 being driven into nonsaturation by the common-mode voltage source. The borderline equation ($V_{D5} = V_{G5} - V_{THN}$) for this transistor can then be used with $V_{D5} = V_{G1,2} - V_{GS1,2}$ and writing both V_{G5} and $V_{GS1,2}$ in terms of its drain current yields

$$V_{G1,\,2(min)} = V_{SS} + \sqrt{\frac{2I_{D5}}{\beta_5}} + \sqrt{\frac{I_{D5}}{\beta_{1,2}}} = V_{SS} + \sqrt{\frac{2L_5 \cdot I_{D5}}{W_5 \cdot K_5}} + \sqrt{\frac{L_{1,2} \cdot I_{D5}}{W_{1,2} \cdot K_{1,2}}} \tag{19.57}$$

Now notice the influencing factors for improving the common-mode range ($V_{G1,2(max)}$ is increased and $V_{G1,2(min)}$ is decreased). Assume that V_{DD} and V_{SS} are defined by the circuit application and are not adjustable. To make $V_{G1,2(max)}$ as large as possible, I_{D5} and L_3 should be made as small as possible while W_3 is made as large as possible. And to make $V_{G1,2(min)}$ as small as possible, L_5, I_{D5}, and $L_{1,2}$ should be made as small as possible while increasing W_5 and $W_{1,2}$ as large as possible. Making the drain current as small as possible is in direct conflict with the slew rate. Decreasing L_5 will also degrade the common-mode rejection ratio, and increasing W_3 will affect the pole location of the output node associated with the diff-amp, thus altering the phase margin. All these tradeoffs must be considered as the designer chooses a circuit topology and begins the process of iterating to a final design.

The output swing of the amplifier is defined as the maximum and minimum values that can appear on the output of the amplifier. In analog applications, we are concerned with producing the largest swing possible while keeping the output driver, M6, in saturation. This can be determined by inspecting the output stage. If the output voltage exceeds the gate voltage of M6 by more than a threshold voltage, then M6 will go into triode. Thus, since the gate voltage of M6 is defined as $V_{DD} - V_{SG3,4}$, the maximum output voltage will be determined by the size of M3 and M4. The larger the channel width of M3 and M4, the smaller the value of $V_{SG3,4}$ and the higher the output can swing. However, again the tradeoff of making M3 and M4 too large is the reduction in bandwidth due to the increased parasitic capacitance associated with the output of the diff-amp.

The minimum value of the output swing is limited by the gate voltage of M7, which is defined by biasing circuitry. Again using the borderline equation, the drain of M7 may not go below the gate of M7 by more than a V_{THN}. Thus, to improve the swing, the value of V_{GS7} must be made small, which implies that the value of V_{GS8} and V_{GS5} also be made small, resulting in large values of M5, M8, and M7. This is not a very wise option because increasing M5 causes all the PMOS devices to increase by the same factor, resulting in large devices for the entire circuit. By carefully designing the bias device M8, one can design V_{GS8} to be around 0.3 V above VTHN. Thus, the output can swing to within 0.3 V of V_{SS}.

Tradeoff Example

When designing amplifiers, the tradeoff issues that occur are many. For example, there are many effects that occur just by increasing the drain current through the diff-amp: the open-loop gain goes down (by the square root of I_D) while the bandwidth increases (by I_D) due to the fact that the resistors r_{o2} and r_{o4} decrease by $1/I_D$. The overall effect is an increase (by the square root of I_D) in GBW, as predicted earlier. A table summarizing the various tradeoffs that occur from attempting to increase the DC gain can be seen in Table 19.1. It is assumed that if a designer only takes the one action listed that the following secondary effects will occur. In fact, the designer should understand the secondary effects enough to where a counteraction is taken to offset the secondary effects.

TABLE 19.1 Tradeoff Issues for Increasing the Gain of the Two-Stage Op-Amp

Desire	Action(s)	Secondary effects
Increase DC gain	Increase W/L1,2	Decreases phase margin
		Increases GBW
		Increases CMRR
		Decreases CMR
	Decrease ID5	Decreases SR
		Increases CMR
		Increases CMRR
		Increases phase margin
	Increase W/L6	Increases phase margin
		Increases output swing
	decrease ID6	Decreases output current drive
		Decreases phase margin

The key for the entire circuit design is the size of M5; the remaining transistors can be written as factors of W_5. The minimum amount of current flowing through M5 is determined by the slew rate. Since M3 and M4 carry half the current of M5, then the widths of M3 and M4 can be determined by assuming that $V_{SG3} = V_{SG4} = V_{GS5}$

$$\frac{2I_{D3,4}}{I_{D5}} \approx \frac{2K_P \cdot (W/L)_{3,4} \cdot (V_{SG3,4} - |V_{THP}|)^2}{K_n \cdot (W/L)_5 \cdot (V_{GS5} - V_{THN})^2} \approx \frac{2K_P \cdot (W/L)_{3,4}}{K_n \cdot (W/L)_5} \tag{19.58}$$

and since $L_3 = L_5$, and $K_n = 3K_p$, then that leads to the conclusion that $W_{3,4} = 1.5 \cdot W_5$. If the nulling resistor is used in the compensation network, the values for M6 and M7 are determined by the amount of load capacitance attached to the output. If a large capacitance is present, the widths of M6 and M7 will need to be large so as to provide enough sinking and sourcing current to and from the load capacitor. Suppose it was decided that the amount of current needed for M6 and M7 was twice that of M5. Then, W_7 would be twice as large as W_5, and W_6 would be six times larger than W_5 to account for the differing K values. Alternatively, if everything is saturated, then $I_{D3} = I_{D4}$, and the drain voltage at the output of the diff-amp is identical to the drain voltage of M3. This implies that under saturation conditions, the gate of M6 is at the same potential as the gate of M4; thus, again it must be emphasized that we are talking about quiescent conditions here, and the current through M6 will be defined by the ratio of M6 to M4. Therefore, since M6 is carrying four times as much current as M3, then $W_6 = 4W_{3,4}$, which is six times the value of W_5. Thus, every device except M1 and M2 is written in terms of M5.

The sizes of M1 and M2 are the most critical of the amplifier. If $W_{1,2}$ are made too large, then C_C will be big due to its relationship with C_L and the ratio of g_{m1} and g_{m6} [Eq. (19.48)]. However, if $W_{1,2}$ is made too small, the gain may not meet the requirements needed.

One word about high-impedance nodes. If two current sources are in a series as shown in Fig. 19.33(a), then the value of the voltage between them is difficult to predict. In the ideal case, both current sources have infinite impedances, so any slight mismatch between I_1 and I_2 will result in large swings in v_A. The same holds true for Fig. 19.33(b). Since the two devices form a high impedance at the output, and each device can be considered a current source, any mismatches in the currents defined by $\beta_2(V_{SG2} - V_{THP})^2$ and $\beta_1(V_{GS1} - V_{THN})^2$ will result in large voltage offsets at the output, with the device with the larger defined current being driven into triode. Thus, the smaller of the two defined currents will be the one flowing through both devices. Another way to visualize this condition is to place a large resistor representing the output impedance from v_o to ground. Any difference between the two transistor currents will flow into or out of the resistor, creating a large voltage offset. Feedback is typically used around the op-amp to stabilize the DC output value.

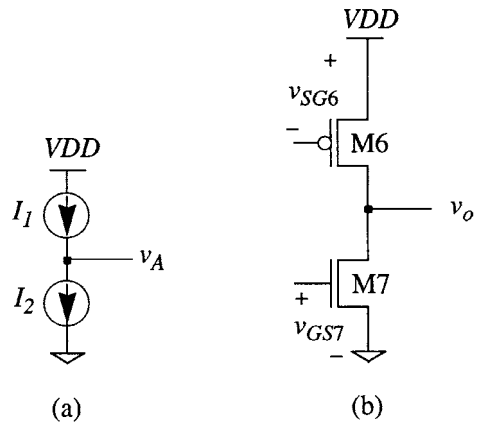


FIGURE 19.33 Illustration of a high-impedance node.

A Word about Circuit Simulation

Circuit simulators have become powerful design tools for analysis of complicated analog circuits. However, the designer must be very careful about the role of the simulator in the design. When simulating high-gain amplifier circuits, it is important to understand the trends and inner working of the circuit before simulations begin. One should always *interpret* rather than blindly trust the simulation results (the latter is a guaranteed recipe for disaster!). For example, the previously mentioned high-impedance nodes should always be given careful consideration when simulating the circuit.

Because these nodes are highly dependent on λ, predicting the actual DC value of the nodes either through simulation or hand analysis is a near impossibility. Before any AC analysis is performed, check the values of the DC points in the circuit to ensure that every device is in saturation. Failure to do so will result in very wrong answers.

Other Output Stages

With the preceding design, the output stage was a high-impedance driver, capable of handling only capacitive loads. If a resistive load is present, an additional stage should be added that has a low output impedance and high current drive capability. An example of the output stage can be seen in Fig. 19.34. Here, the output impedance is simply $1/g_{m9}||1/g_{m10}$. Since we do not wish to have a large output impedance, the values for L_9 and L_{10} should be made as small as possible. The transistors M11 and M12 are used to help bias the output devices such that M9 and M10 are just barely on under quiescent conditions. This kind of amplifier is known as a class AB output stage and has limitations in CMOS due to the body effect.

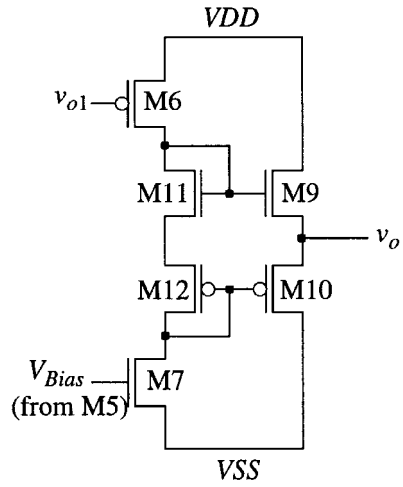


FIGURE 19.34 A low-impedance output stage.

In some cases, it is advantageous to use a bipolar output driver as seen in Fig. 19.35. Since most BiCMOS processes provide only one flavor of BJT (an npn), the transistor Q1 can be used for extra current drive. This results in a dual-sloped transfer curve characteristic as the output stage goes from sourcing to sinking. It should be noted that one could use this output stage with the complementary version of the two-stage amplifier previously discussed.

Another BiCMOS output stage can be seen in Fig. 19.36.[5] This is known as a "pseudo-push-pull" output stage. M6 and M7 can be output of the previously discussed two-stage op-amp. With the new pseudo-push-pull output attached, the amplifier is able to achieve high output swing with large current drive capability using very little silicon area. The transistor MQ1 is for level shifting purposes only. When the output signal needs to swing in the positive direction, Q2 behaves just like the BJT output driver shown in Fig. 19.35. When the output swings in the negative direction, Q1 drives the gate of M9 down, thus increasing I_{D9}. The increase in current is mirrored via M10 to M11, which is able to sink a large amount of current. The output voltage at its lowest voltage is approximately the same voltage as the emitter voltage of Q2. The transistor Q2 provides the needed low output impedance.

Another advantage of using BJT devices in the design is to provide large g_ms as compared to the MOS counterpart. BiCMOS circuits can be constructed, which offer high input impedance and gain bandwidth products. If both npn and pnp devices are available in the BiCMOS process, then the output stage seen in Fig. 19.37 can be used. This circuit functions as a buffer circuit with very low output impedance.

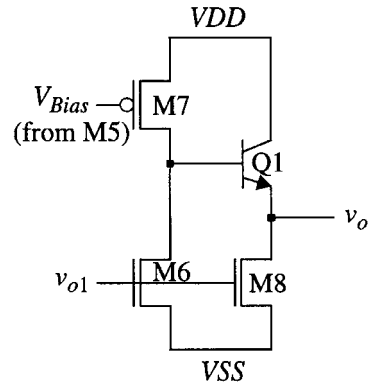


FIGURE 19.35 Using an NPN BJT as an output driver.

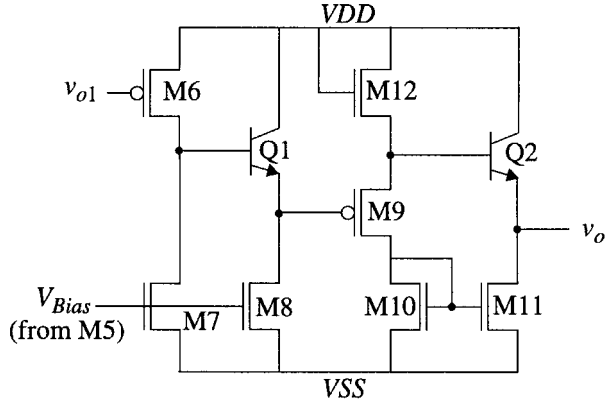


FIGURE 19.36 A "pseudo" push-pull npn only output driver.

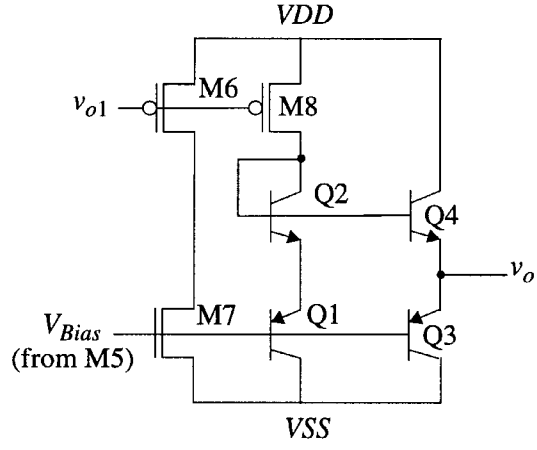


FIGURE 19.37 A low-impedance output stage using both npn and pnp devices.

High-Performance Operational-Amplifier Considerations

In the commercial world, it seems there are not many applications that require simple, easy to design op-amps. High bit rate communication systems and over-sampled data converters push the bandwidth and slew rate capabilities of CMOS op-amps, while battery-powered systems are required to squeeze just enough performance out of micro-amps of current, and a volt or two of power supply. Sensor interfaces and audio systems demand low noise and distortion. To further complicate things, CMOS analog circuits are often integrated with large digital circuits, making isolation from switching noise a major concern. This section will present solutions to some of the problems faced by op-amp designers who, because of budget constraints or digital compatibility, do not have the option to use bipolar junction transistors in their design. We will then give some hints on where the designer might use bipolar transistors if they are available.

Power Supply Rejection

Fully differential circuits like the OTA shown in Fig. 19.38 are used in mixed signal circuits because they provide good rejection of substrate noise and power supply noise. As long as the noise coupled from the substrate or power supply is equal for both outputs, the difference between the two signals is noise-free (differential component of the noise is zero). This is illustrated in Fig. 19.39. The top two traces in Fig. 19.39 are a differential signal corrupted with common-mode noise. The bottom trace is the difference between these two noisy signals. If the next circuit in the path has good common-mode rejection, the substrate and power supply noise will be ignored. In practical circuits, mismatches between the transistors of symmetrical halves of the differential circuit will lead to imperfect matching of noise on the outputs, and therefore reduced rejection of power supply noise. Common centroid layouts and large geometry transistors are necessary to minimize mismatches. Differential circuits are capable of twice the signal swing of single-ended circuits, making them especially welcome in low-voltage and low-noise applications.

Single-stage or multiple-stage op-amps can be made differential, but each stage requires a common-mode feedback circuit to give the differential output a common-mode reference. Consider the folded cascode OTA shown in Fig. 19.38. If the threshold voltage of M5A is slightly larger than the threshold voltage of M5B and M5C, the pull-down currents will be larger than the pull-up currents. This small current difference, in combination with the very high output impedance of the cascode current mirrors,

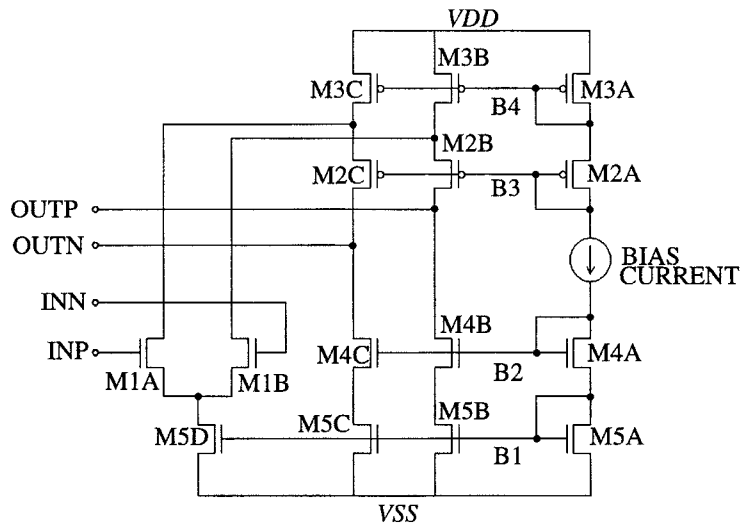


FIGURE 19.38 Folded cascode OTA.

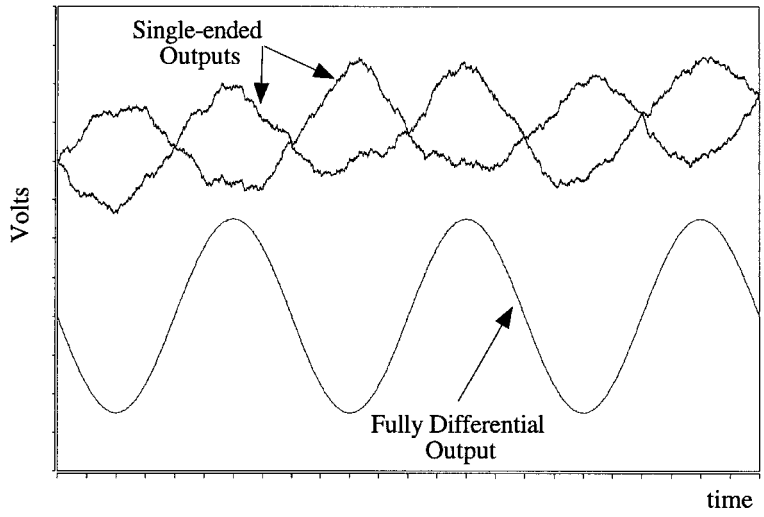


FIGURE 19.39 Simulation output illustrating the difference between single-ended and fully differential signals.

will cause the output voltages to be pegged at the negative power supply. This common-mode error cannot be corrected by applying feedback to the differential pair. A common-mode feedback circuit is needed to find the average of the output voltages, and control the pull-up or pull-down current in the outputs to maintain this average at the desired reference. A center-tapped resistor between the outputs could be used to sense the common-mode voltage if the outputs were buffered to drive such a load. Since a folded cascode OTA cannot drive resistors, a switched capacitor would be a better choice to sense the common-mode voltage as shown in Fig. 19.40. The PH1 and PH2 clock signals must be non-overlapping. When the PH1 switches are closed, C_{1A} and C_{1B} are discharged to zero, while C_{2A} and C_{2B} provide feedback to the common-mode amplifier. The PH1 switches are then opened, and a moment later, the PH2 switches closed. The charge transfer that takes place moves the center tap

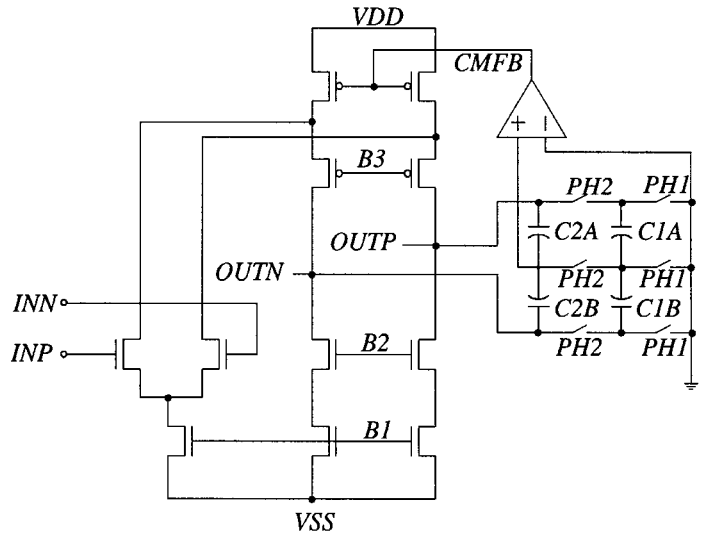


FIGURE 19.40 Folded cascode OTA using switched capacitor common mode feedback.

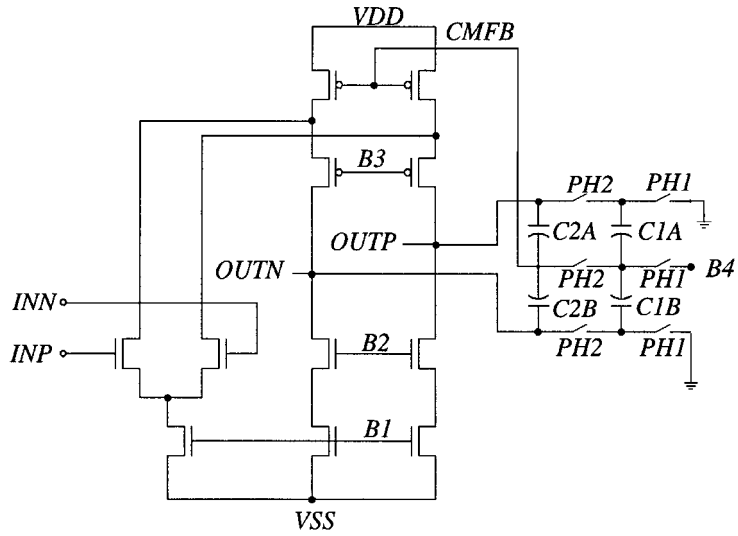


FIGURE 19.41 Using a bias voltage to precharge the switched capacitor common mode feedback capacitors.

between C_{2A} and C_{2B} toward the average of the two output voltages. After many clock cycles, the input to the common-mode feedback amplifier will be the average of the two voltages. C_{1A} and C_{1B} can be precharged to a bias voltage to provide a level shift. This allows direct feedback from the common-mode circuit to the pull-up bias as shown in Fig. 19.41.

A BiCMOS version of the folded cascode amplifier can be seen in Fig. 19.42.[6] Here, it is again assumed that only npn devices are available. Note that to best utilize the npn devices, the folded cascoded uses a diff-amp with P-channel input devices. The amplifier also uses the high swing current mirror presented in Section 19.2.

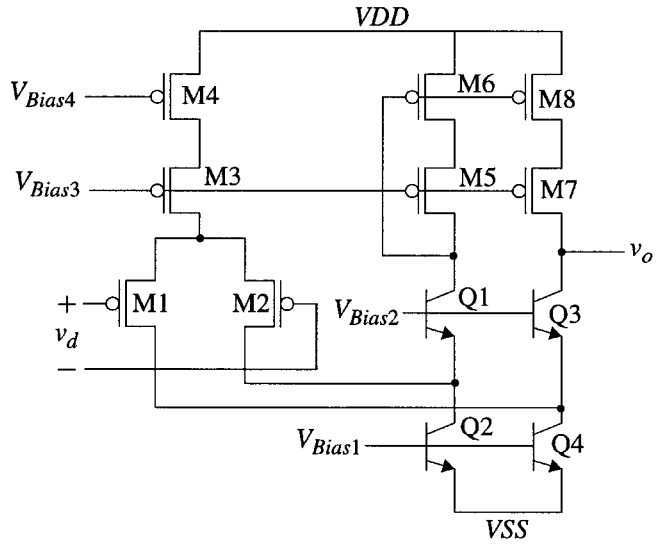


FIGURE 19.42 BiCMOS folded cascode amplifier.

Slew Rate

The slew rate of a single-stage class A OTA is the maximum current the output can source or sink, divided by the capacitive load. The slew rate is therefore proportional to the steady state power consumed. Class AB amplifiers give a better tradeoff between power and slew rate. The amplifier's maximum output current is not the same as the steady-state current. An example of a class AB single-stage op-amp is shown in Fig. 19.43. The differential pair is replaced by M1A, M1B, M2A, and M2B. A level shifter consisting of M3A, M3B, M4A, and M4B couples the input signal to M2A and M2B, and sets up the zero input bias current. If the width of M2A and M2B are three times the width of M1A and M1B, the small signal voltage on the nodes between M1 and M2 will be approximately zero. The current available from one of the input transistors is approximately

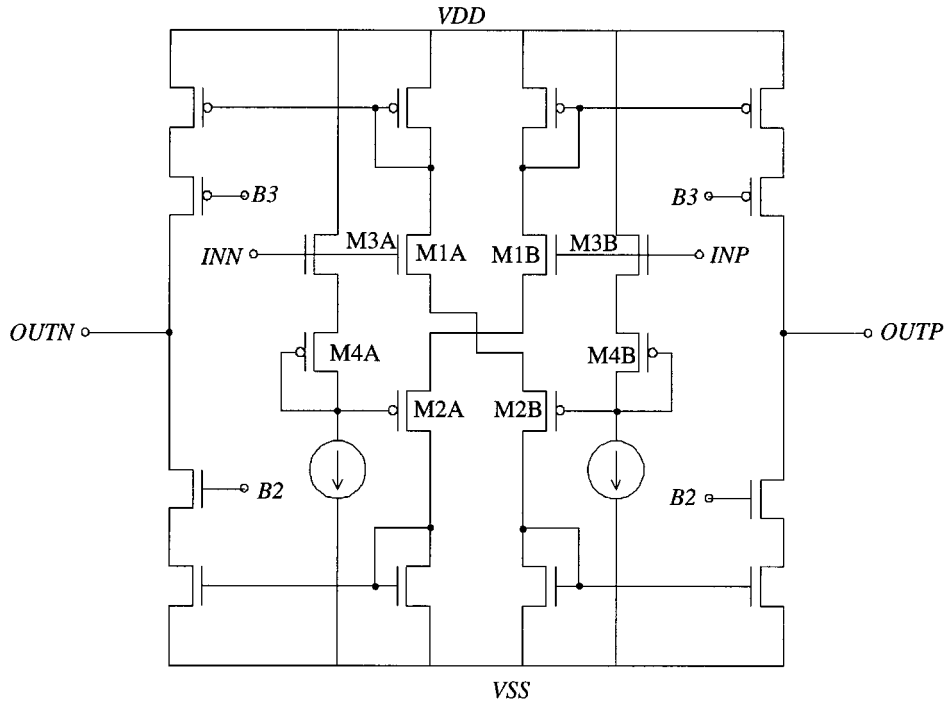


FIGURE 19.43 A class AB single stage op-amp with high slew rate capability.

$$I_{out} = KP\frac{W}{L}(V_{in} + V_{bias} + V_{THN})^2 \tag{19.59}$$

The differential current from the input stage is

$$I_{OUTP} - I_{OUTN} = 2 \cdot \beta(V_{in} + V_{BIAS} + V_{THN}) \tag{19.60}$$

It is interesting to note that the non-linearities cancel. The output current becomes non-linear again as soon as one of the input transistors turns off. The maximum current available from the input transistors is not limited by a current source as is a differential pair. It should be noted that it is impossible to keep all transistors saturated for low power supplies and large input common-mode swings. A similar BiCMOS circuit with high slew rate can be seen in Fig. 19.44.[7]

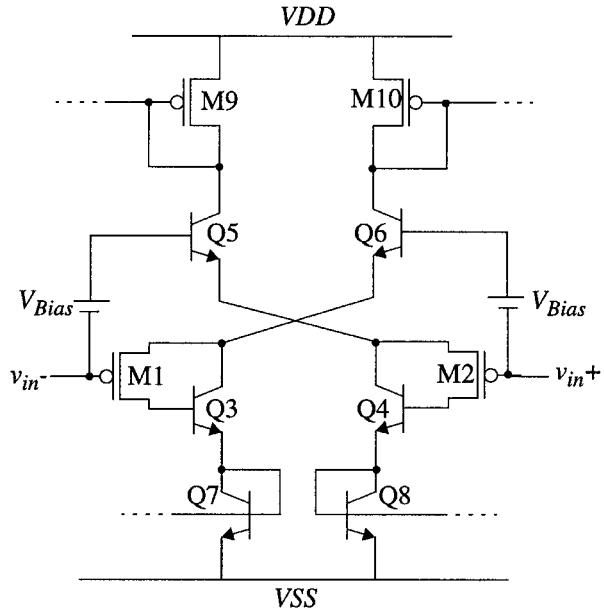


FIGURE 19.44 A BiCMOS class AB input stage.

Adaptive bias is another method to reduce the ratio of supply current to slew rate. Adaptive bias senses the current in each side of a conventional differential pair, and increases the bias current when the current on one side falls below a preset value. This guarantees that neither side of the differential pair will turn off. The side that is using most of the current must therefore be supplied with much more than the zero input amount. The differential pair current can be sensed by measuring the gate-to-source voltages as shown in Fig. 19.45, or by measuring the current in the load as shown in Fig. 19.46. Both of these adaptive

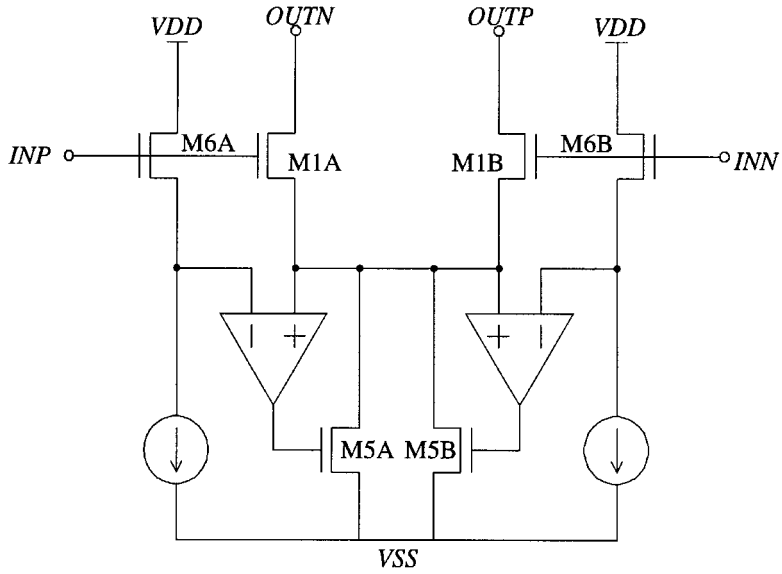


FIGURE 19.45 Adaptive biasing scheme for improved slew rate performance.

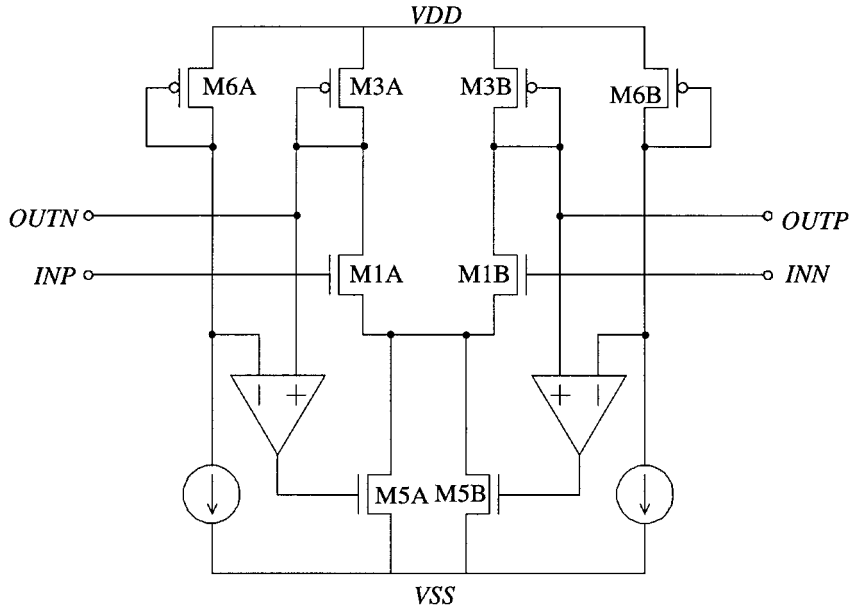


FIGURE 19.46 Another adaptive biasing circuit.

bias schemes depend on a feedback loop to control the current in the differential pair. These circuits improve settling time for low power, low bandwidth circuits, but the delay in the feedback path is a problem for high-speed circuits.

A second form of adaptive bias can be used when it is known that increased output current is needed just after clock edges, such as in switched capacitor filters. This type of adaptive bias can be realized using the switched capacitor bias boost shown in Fig. 19.47. In this circuit, $I_{D1} \cdot L_1/W_1 > I_{D2} \cdot L_2/W_2$, which makes $V_1 > V_2$. When the PH1 switches are closed, the steady-state voltage across the capacitor is $V_1 - V_2$, and

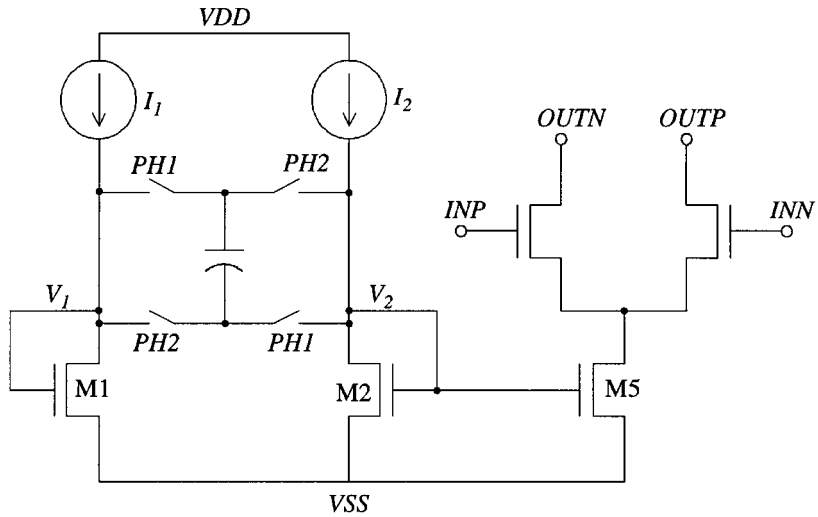


FIGURE 19.47 Adaptive biasing using a switched capacitor circuit.

the current in M3 is set by I_{D2}. When the PH2 switches close, V_2 is momentarily increased, while V_1 is decreased. The transient is repeated when the PH1 switches close. The net effect of the switched capacitor bias boost is that the current in M3 increases after both clock edges. Notice that the current is increased, whether it is needed or not. This circuit is fast because there is no feedback loop, but it is not the most efficient because it does not actually sense the current in the differential pair.

In all three approaches, the quiescent current is less than the current when the output is required to slew. Therefore, output swing and gain are not degraded when the bias current returns to its quiescent value. All three adaptive bias circuits will cause larger current spikes to be put on the supplies by the op-amps. The width of the power supply lines should be increased to compensate for the increased *IR* drop and crosstalk. Enhanced slew rate circuits are not linear time invariant systems because the transconductance and output impedance of the transistors are bias current dependent, and the bias is time varying. A transient analysis is the most dependable way to evaluate settling time in this case.

Output Swing

Decreasing power supply voltages put an uncomfortable squeeze on the design engineer. To maintain the desired signal-to-noise ratio with a smaller signal swing, circuit impedances must decrease, which often cancels any power savings that may be gained by a lower supply voltage. To get the best signal swing, differential circuits are used. If the output stages use cascode current mirrors, bias voltages must be generated, which keep both the mirror and cascode transistors in saturation with the minimum voltage on the output. Another example of a high swing bias circuit (refer also to Section 19.2) and a current mirror is shown in Fig. 19.48. First, let the *W*/*L* ratio of M2, M4, and M6 be equal, and the *W*/*L* ratio of M3 and M5 be equal. Now recall that to keep a MOSFET in saturation

$$V_{DS} \geq V_{GS} - V_{THN} \tag{19.61}$$

The minimum output voltage that will keep both M5 and M6 in saturation with proper biasing is

$$V_{out} \geq V_{GS6} - 2V_{THN} + V_{GS5} \tag{19.62}$$

ignoring the bulk effects. For a given current, the minimum drain voltage can be rewritten as

$$V_{DS} \geq \sqrt{\frac{I_D \cdot L}{K \cdot W}} \tag{19.63}$$

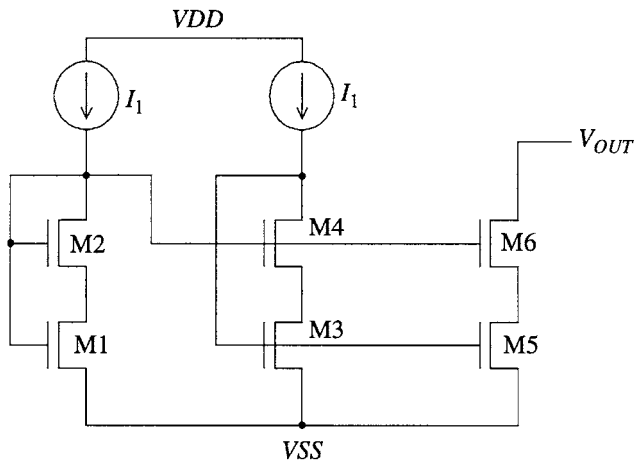


FIGURE 19.48 A high swing biasing circuit for low power supply applications.

The equation for the minimum V_{OUT} can be rewritten as

$$V_{OUT} \geq \sqrt{\frac{I_{OUT} \cdot L_6}{K_6 \cdot W_6}} + \sqrt{\frac{I_{OUT} \cdot L_5}{K_5 \cdot W_5}} \tag{19.64}$$

The trick to making this bias generator work is setting V_{DS} of M1 equal to the minimum V_{DS} required by M3 and M5. M1 is biased in the linear region, while we wish to keep M3 and M5 saturated. It is a good idea to set $L_1 = L_3 = L_5$ to match etching tolerances. A second trick is to make sure M3 and M4 stay saturated. M4 is inserted between the gate and drain connections of M3 to make $V_{DS3} = V_{DS5}$. If the W/L ratio of M3 is too small, M3 will be forced out of saturation by the source of M4. If the W/L_3 is too large, the gate voltage of M3 will not be large enough to keep M4 in saturation.

DC Gain

Start by calculating the DC gain of the folded cascode OTA in Fig. 19.38. If we assume the output impedance of the n-channel cascode current mirror is much greater than the output impedance of the individual transistors, then the DC gain is approximately

$$\frac{v_o}{v_{in}} = -g_{m1} \cdot r_{o1} \left| \right| r_{o3} \cdot g_{m2} \cdot r_{o2} \tag{19.65}$$

If we assume that the current from the M3 splits equally to M1 and M2, the gain can be written as

$$\frac{v_o}{v_{in}} \propto \frac{\sqrt{W_1 L_1 W_2 L_2}}{I_{D1}} \cdot \frac{L_3}{2L_1 + L_3} \tag{19.66}$$

We can see that the gate area of the differential pair and cascode transistors must both double each time current is doubled to maintain the same gain. We also note that it is desirable to make $L_3 > L_1$. If the current in the amplifier were raised to increase gain-bandwidth, or slew rate, it would be desirable to increase the widths of the transistors by the same factor to maintain output swing.

Regulated gate cascode outputs increase the gain of the OTA by effectively multiplying the g_m of M4 by the gain of the RGC amplifier. The stability of the RGC amplifier loop must be considered. An example of a gain boosted output stage is shown in Fig. 19.49.

Gain Bandwidth and Phase Margin

Again, start with the transfer function for the folded cascode OTA of Fig. 19.38. If we assume the output impedances of the n-channel cascode current mirrors are very large, and the gain of M2 is much greater than one, we have

$$\frac{v_o}{v_{in}} = \frac{g_{m1} r_{o1} g_{m2} r_{o2}}{(C_1 \cdot C_{out} \cdot r_{o1} \cdot r_{o2}) \left(s^2 + \left(\frac{g_{m2}}{C_1} + \frac{1}{C_1 \cdot r_{o1}} + \frac{1}{C_1 \cdot r_{o2}} + \frac{1}{C_{out} \cdot r_{o2}} \right) s + \frac{1}{C_1 \cdot C_{out} \cdot r_{o1} \cdot r_{o2}} \right)} \tag{19.67}$$

where ro_1 is now the parallel combination of r_{o1} and r_{o3}. If we further assume that the poles are spaced far apart, and that g_{m2} is much larger than $1/r_{o1}$ and $1/r_{o2}$, then the gain-bandwidth product is g_{m1}/C_{load}. The second pole, which will determine phase margin, is approximately

$$\omega = \frac{g_{m2}}{C_1} + \frac{1}{C_{out} \cdot r_{o2}}$$

The depletion capacitance of the drains of M1 and M3 will also add to this capacitance. As a first cut, let $C_1 = K_c \cdot W_2 \cdot L_2$. Now the equation for the second pole boils down to

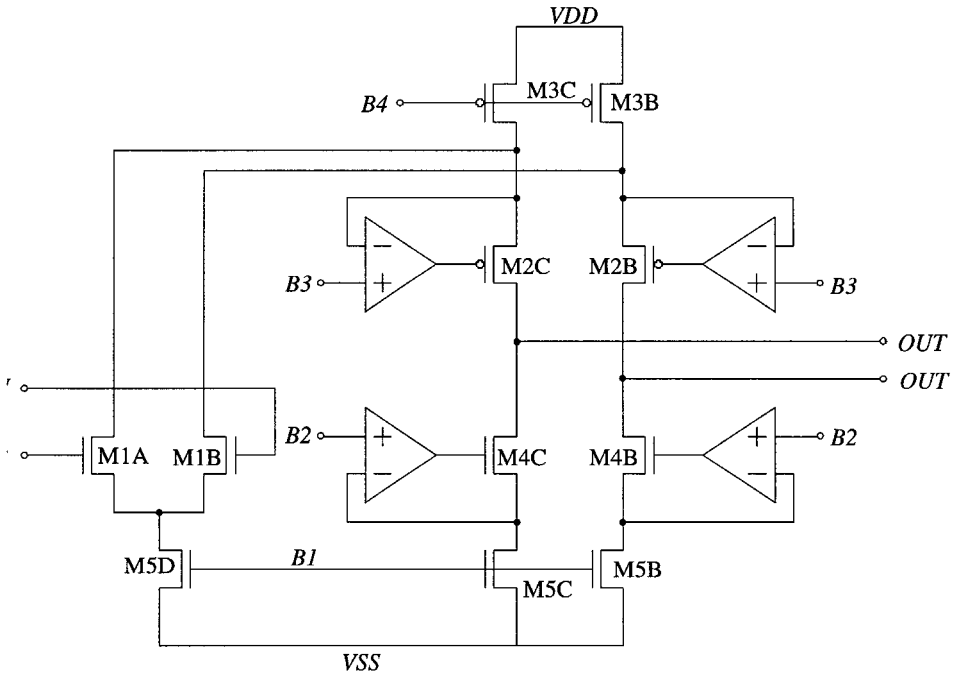


FIGURE 19.49 OTA with regulated gate cascode output.

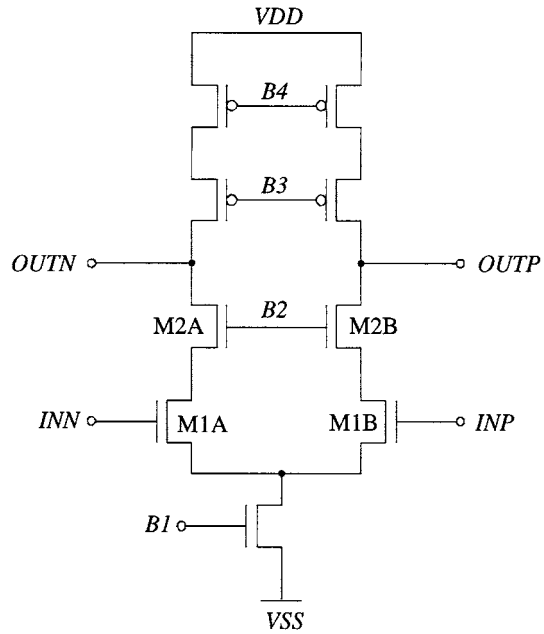


FIGURE 19.50 A telescopic OTA.

$$\omega \approx \frac{\sqrt{K \cdot I_{D2}}}{K_C \cdot L_2 \cdot \sqrt{W_2 \cdot L_2}} + \frac{I_{D2}}{L_2 \cdot C_{out}}$$

To get maximum phase margin, we clearly want to use as short a channel length as the gain specification will allow. The folded cascode OTA and two-stage OTA both have n-channel and p-channel transistors in the signal path. Since holes have lower mobility than electrons, it is necessary to make a silicon p-channel transistor about three times wider than an n-channel transistor of the same length to get the same transconductance. The added parasitic capacitance of the wider p-channel transistor is a hindrance for high-speed design. The telescopic OTA shown in Fig. 19.50 has only n-channel transistors in the signal path, and can therefore achieve very high bandwidths with acceptable phase margin. Its main drawback is that the output common-mode voltage must be more positive than the input common-mode voltage. This amplifier can achieve even wider bandwidth with acceptable phase margin if M2 is replaced by an npn bipolar transistor.

References

1. R. J. Baker, H. W. Li, and D. E. Boyce, *CMOS: Circuit Design, Layout, and Simulation*, IEEE Press, 1998.
2. A. S. Sedra and K. C. Smith, *Microelectronic Circuits, fourth edition*, Oxford University Press, London, 1998.
3. P. E. Allen and D. R. Holberg, *CMOS Analog Circuit Design*, Saunders College Publishing, Philadelphia, 1987.
4. P. R. Gray, *Basic MOS Operational Amplifier Design — An Overview, Analog MOS Integrated Circuits*, IEEE Press, 1980.
5. H. Qiuting, A CMOS power amplifier with a novel output structure, *IEEE J. of Solid-State Circuits*, vol. 27, no. 2, pp. 203-207, Feb. 1992.
6. M. Ismail, and T. Fiez, *Analog VLSI: Signal and Information Processing*, McGraw-Hill, Inc., New York, 1994.
7. S. Sen and B. Leung, A class-AB high-speed low-power operational amplifier in BiCMOS technology, *IEEE J. of Solid-State Circuits*, vol. 31, no. 9, pp. 1325-1330, Sept. 1996.

De Veirman, G.E. "Bipolar Amplifier Design"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

20

Bipolar Amplifier Design

Geert A. De Veirman
Silicon Systems, Inc.

20.1 Introduction

This chapter gives an overview of amplifier design techniques using bipolar transistors. An elementary understanding of the operation of the bipolar junction transistor is assumed. Section 20.2 reviews the basic principles of amplification and details the proper selection of an operating point. This section also introduces the three fundamental single-transistor amplifier stages: the common-emitter, the common-collector, and the common-base configurations. Section 20.3 deals with the problem of amplification of dc and difference signals. The emitter-coupled differential pair is discussed in great depth. Issues specific to output stages are presented in Section 20.4. Section 20.5 briefly touches on supply-independent biasing techniques. Next, Section 20.6 combines all the acquired building block knowledge in a condensed overview of operational amplifiers. A short conclusion is presented in Section 20.7, followed by a list of references.

20.2 Single-Transistor Amplifiers

Basic Principles

Bipolar Transistor Operation

Although prior exposure to bipolar transistor fundamentals is expected, a few elementary concepts are briefly reviewed here. The *bipolar junction transistor* (BJT) contains two back-to-back pn junctions,

known as the base–emitter and base–collector junctions. In this chapter, it is assumed that the BJT is operated in the so-called normal mode with a forward-bias applied at the base–emitter junction and a reverse voltage across the base–collector interface. Then, the collector current I_C, which is β times the base current I_B, is exponentially related to the base–emitter voltage V_{BE}. The mathematical expression of this dependency is known as the *Ebers-Moll equation.* For small signal variations, the relationship between the current *through* the transistor (I_C) and the *control input* voltage (V_{BE}) is expressed by the *transconductance* g_m.

When designing bipolar transistor amplifiers, it is also essential to understand the relationship between I_C and the voltage *across* the transistor (the collector-emitter voltage V_{CE}). For each input voltage V_{BE}, a different characteristic exists in the I_C-V_{CE} diagram, as shown in Fig. 20.1. As long as V_{CE} is high enough to keep the transistor in the linear region, the I_C vs. V_{CE} characteristics are fairly flat. When V_{BE} is held constant, increasing V_{CE} corresponds to applying more reverse voltage across the base–collector junction. As a result, the base width is modulated and this causes the I_C curves to increase approximately linearly with V_{CE}. When the I_C curves are extrapolated to negative V_{CE} values, they eventually all go through a single pivot point, which is referred to as the transistor's *Early voltage* V_A. For very small values of V_{CE}, the base–collector junction becomes forward-biased. As a result, the transistor saturates and the collector current drops off rapidly.

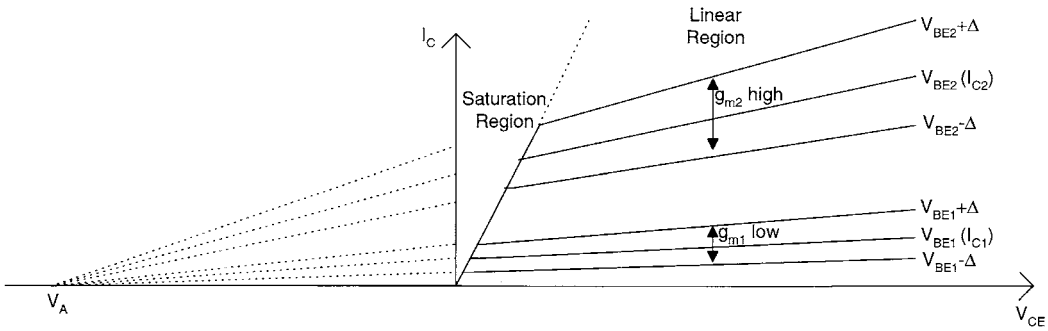


FIGURE 20.1 $I_C(I_{through})$ vs. $V_{CE}(V_{across})$ characteristics of the npn bipolar junction transistor.

In Fig. 20.1, V_{BE1} is less than V_{BE2}. In the linear operating region, the exponential nature of the BJT renders $g_{m1} < g_{m2}$, which means that for an identical incremental input voltage Δ, I_{C2} changes by a larger amount than I_{C1}. Also, since both curves merge at V_A, I_{C2} is a stronger function of V_{CE} than I_{C1}.

Mathematical expressions for the relationships discussed above will be introduced as needed in the following sections.

Basic Bipolar Amplifier Concepts

In order to amplify a voltage with a bipolar transistor, a load is placed in series with the device. In its simplest form, this load consists of a resistor R_L. Taking into account the polarity of electrons and the direction of current flow, one ends up with a transistor and supply arrangement as shown in Fig. 20.2, where the voltage to be amplified (V_i) drives the base. Hence, the only available output node (V_o) in Fig. 20.2 is the voltage between the transistor and the load resistor. This voltage can be determined in the I_C–V_{CE} diagram of Fig. 20.1 by subtracting the voltage drop across R_L from the supply voltage V_{CC}. The result is a straight line, which is known as the *load line.* The load line, shown in Fig. 20.3, graphically represents the possible operating points of the bipolar transistor. The line is straight simply because the resistor R_L is a linear element. Every point on the load line represents a state, which is determined by the voltage at the transistor's base. This state in turn determines how much current flows through the transistor and how much voltage there exists across this device.

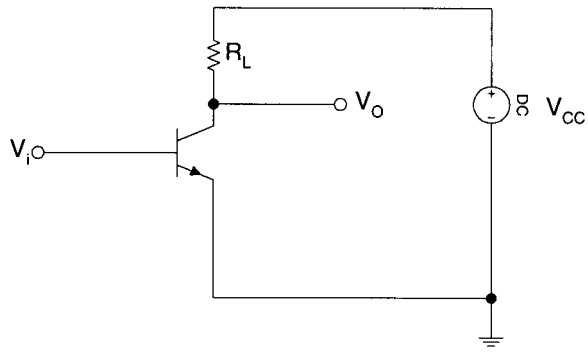


FIGURE 20.2 Conceptual schematic of single-transistor amplifier.

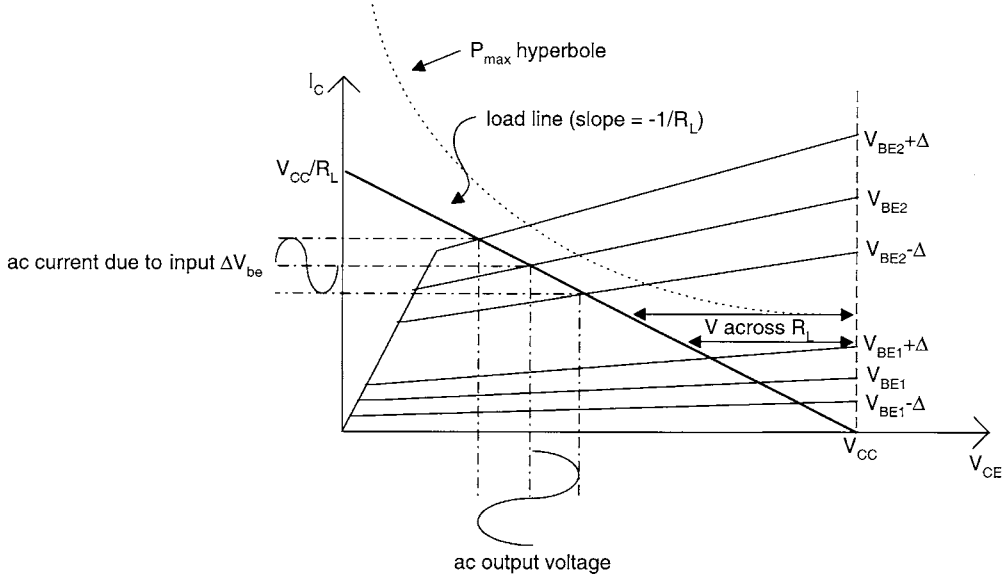


FIGURE 20.3 Load line characteristic.

To determine the circuit's voltage amplification, all one has to do is to plot the output voltage as a function of the input voltage. For different values of R_L, this process results in the *transfer characteristics* depicted in Fig. 20.4. The transfer characteristics allow us to visually determine two important items: first, the range in which the input voltage V_i should lie in order to obtain the maximum possible voltage variation at the output; and second, the optimum value of R_L. Since the transfer characteristics in Fig. 20.4 are far from linear, the best one can do is to identify a region where the curves can be linearized, provided the signal swings are not too large. Also, note that the transfer curves do not pass through the (0,0) coordinate. This means that the desired proportionality between input and output is not present. Consequently, one would want to move the V_o and V_i axes, so that their origin coincides with a desired point on the curves, as is the case for the V_o'/V_i' coordinate system illustrated in Fig. 20.4. The next subsection will discuss how such a suitable *operating point* (or *quiescent point*) can be established.

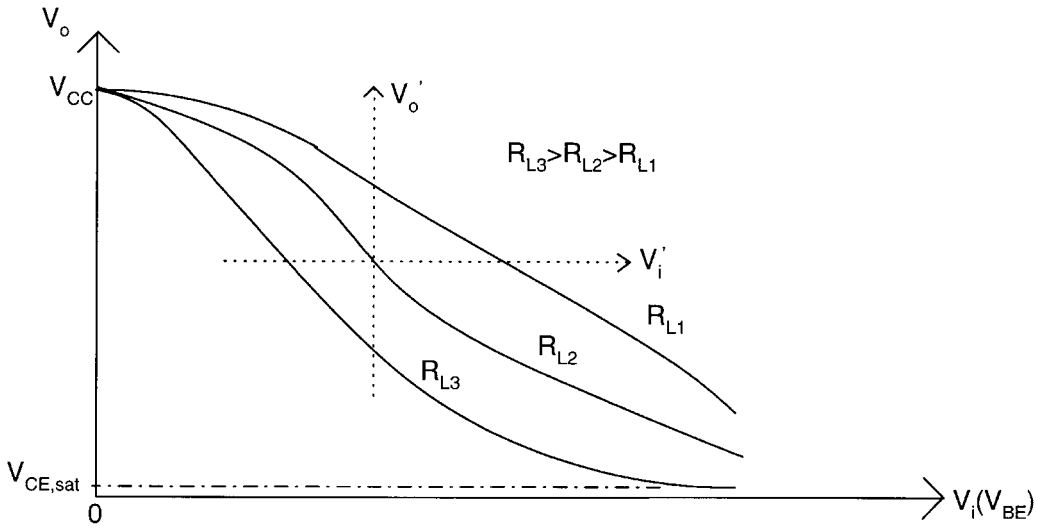


FIGURE 20.4 Transfer characteristics.

Figure 20.4 furthermore graphically clarifies how the bipolar transistor can be used in three distinct ways: first, as a switch; second, as a linear amplifier for small signals; and third, as a linear amplifier for large signals. When operating the BJT as a switch, the (logic) designer is primarily concerned about the speed of the transitions between the OFF and ON states. Such obviously non-linear modus operandi is a topic outside the scope of this chapter. Large signals pose particular difficulties, which will be covered a bit later in the sections on differential pairs and output stages. In the next several sections, we will deal mainly with linear small-signal operation.

Setting a Suitable Operating Point

The choice of a proper operating point depends on several factors. Three key decision criteria are:

1. Maximum allowable power dissipation of the active element. Each transistor has a maximum power limit. If this limit is exceeded, permanent damage (e.g., degradation of β) occurs or, in the worst case, the device may be destroyed. (Actually, maximum power ratings can be temporarily exceeded, as long as such transgressions are very fast. But such operation goes beyond this discussion.) In the I_C–V_{CE} diagram, constant power curves are represented by hyperboles. One such hyperbole, representing the maximum allowable power dissipation P_{max}, is included in Fig. 20.3. The designer must guarantee that excursions from the operating point along the amplifier's load line do not cross into the forbidden zone above the P_{max} curve.
2. Proper location on the transfer curve. As mentioned above during the discussion of Fig. 20.4, this choice directly affects the achievable linearity and signal swing.
3. Bias current. Figure 20.5 illustrates how the vertical axis in the V_o/V_i diagram can be moved. All one has to do is apply a proper dc bias voltage and superpose a small signal input. Making use of a separate bias voltage, as shown in Fig. 20.5(a), is one possibility. Alternatively, the dc voltage at the base can be set by means of a resistive divider from V_{CC}, and the ac signal can be capacitively coupled through C_{ci}, as depicted in Fig. 20.5(b). Similarly, a coupling capacitor C_{co} can be employed to separate the dc bias at V_o from the ac signal swing. As such, C_{co} moves the horizontal axis in the V_o/V_i diagram.

In Fig. 20.6(a), a second load resistor R_L' is added. Since the capacitor C_{co} acts as an open circuit for dc signals, the operating point remains solely determined by the intersection of the chosen transistor characteristic and the *static* load line, which results from the combination of V_{CC} and R_L. Assuming C_{co}

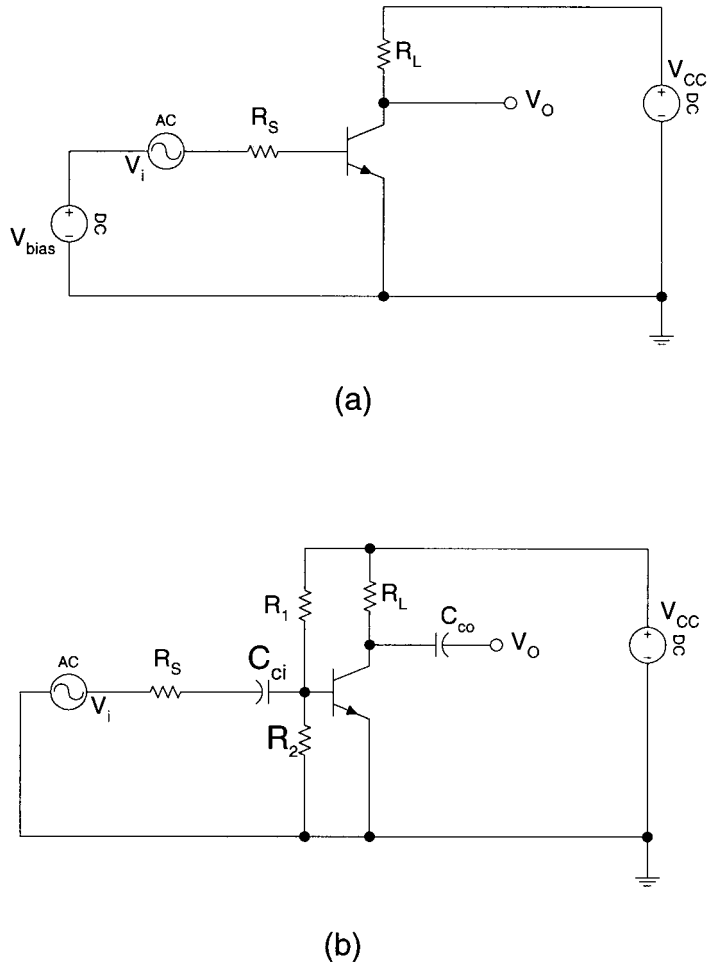


FIGURE 20.5 Common-emitter amplifier: (a) V_{bias} in series with ac signal source, (b) arrangement with coupling capacitors.

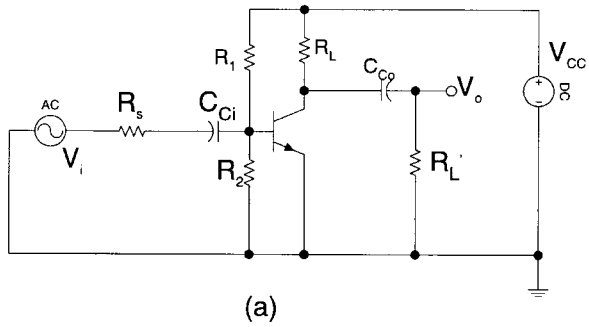
is large, this coupling capacitor behaves as a short-circuit for ac signals. Hence, R_L' is effectively in parallel with R_L. Therefore, signal excursions occur along a new *dynamic* load line, as illustrated in Fig. 20.6(b).

Common-Emitter Amplifier

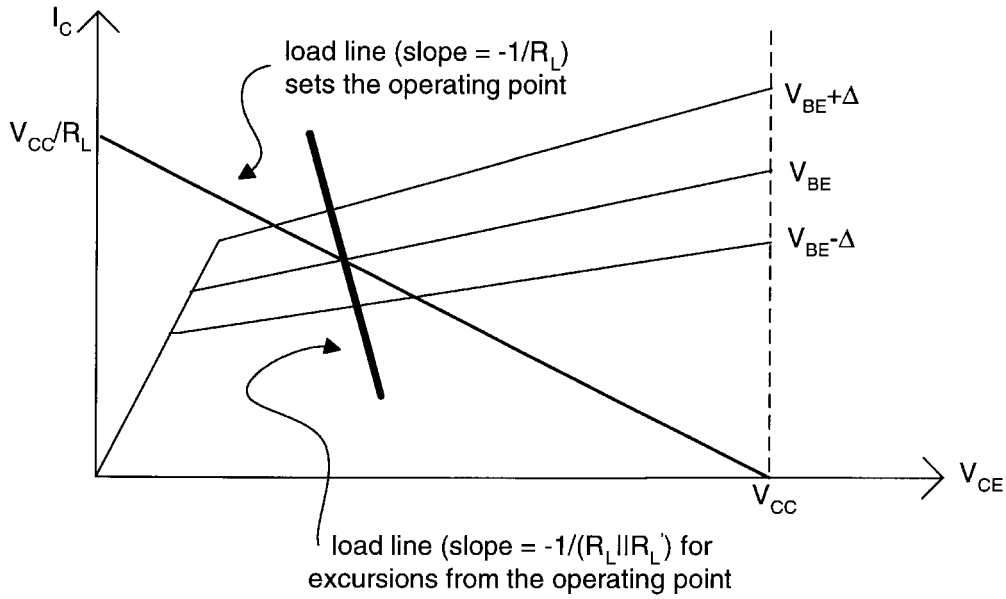
The circuits in Figs. 20.5 or 20.6(a) are known as *common-emitter amplifiers.* This name is derived from the fact that the emitter terminal is common with the ground node.

Small-Signal Gain

The small-signal equivalent circuit of the common-emitter amplifier is shown in Fig. 20.7. This circuit is derived based on the observation that for ac signals, fixed dc bias sources are identical to ac ground. We have also assumed (for now) that the coupling capacitors C_{ci} and C_{co} are so large that for any frequencies of interest, they effectively behave as short-circuits. Moreover, parasitic capacitances internal to the transistor are ignored. R_L' has been eliminated for simplicity (or one could assume that R_L represents the effective parallel resistance). R_S is the series output resistance of the ac source V_i and r_b stands for the physical resistance of the base. r_π models the linearized input voltage–current characteristic



(a)



(b)

FIGURE 20.6 (a) common-emitter amplifier with ac coupled load R'_L, (b) I_C vs. V_{CE} diagram with static and dynamic load lines.

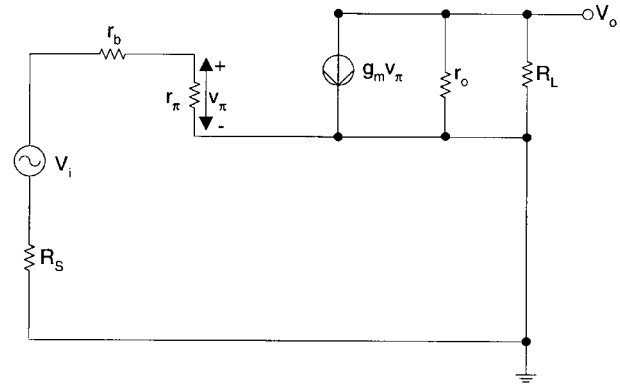


FIGURE 20.7 Small-signal equivalent circuit for the common-emitter amplifier.

of the base–emitter junction. In other words, r_π is the ratio between v_{be} and i_b for small-signal excursions from the operating point set by V_{BE}. Although the parallel combination of R_1 and R_2 shunts r_π, we assume here that both resistors have such high ohmic values that their effect on r_π can be regarded as immaterial. From the Ebers-Moll equation (with $V_{BE} >> V_T$),

$$I_C = \beta I_B = I_S \exp\left(\frac{V_{BE}}{V_T}\right)\left(1 + \frac{V_{CE}}{V_A}\right) \tag{20.1}$$

where I_S is the transistor's saturation current and $V_T = kT/q$ the thermal voltage, one readily derives the following expressions for the transconductance g_m, the input resistance r_π, and the output resistance r_o.

$$g_m = \frac{dI_C}{dV_{BE}} = \frac{I_C}{V_T} \tag{20.2}$$

$$r_\pi = \left(\frac{dI_B}{dV_{BE}}\right)^{-1} = \left(\frac{I_C}{\beta V_T}\right)^{-1} = \frac{\beta}{g_m} \tag{20.3}$$

$$r_o = \left(\frac{dI_C}{dV_{CE}}\right)^{-1} = \left(\frac{I_S}{V_A}\exp\left(\frac{V_{BE}}{V_T}\right)\right)^{-1} \tag{20.4}$$

In general, $V_A >> V_{CE}$. Therefore,

$$r_o \approx \frac{V_A}{I_C} \tag{20.5}$$

The following two equations can be written for the circuit in Fig. 20.7.

$$v_\pi = V_i \frac{r_\pi}{r_b + R_S + r_\pi} \tag{20.6}$$

$$V_o = -g_m v_\pi (r_o \parallel R_L) \tag{20.7}$$

where $\parallel$ denotes the parallel combination of two resistors. Combining Eqs. (20.6) and (20.7) leads to the expression for the small-signal gain

$$A = \frac{V_o}{V_i} = -g_m \frac{r_\pi}{r_b + R_S + r_\pi} \frac{r_o R_L}{r_o + R_L} \tag{20.8}$$

Assuming $r_b + R_S << r_\pi$ and $R_L << r_o$,

$$A \approx -g_m R_L \tag{20.9}$$

Equation (20.9) implies that selecting a higher-valued load resistor R_L results in higher gain. However, this conclusion is only valid as long as $R_L << r_o$. Conversely, for a given R_L, the gain can be increased by choosing a higher g_m. According to Eq. (20.2), this corresponds to a higher I_C. However, r_o goes down with increasing I_C (see Eq. (20.5) and Fig. 20.1), causing some reduction in gain. Furthermore, assuming β is constant over the range of operating points, Eq. (20.3) specifies that r_π also decreases with increasing g_m, augmenting the reduction in gain. To compound the problem, β actually starts rolling off at high current levels.

The best choice of operating point is most often determined by ensuring the largest possible output signal swing. In that case,

The best choice of operating point is most often determined by ensuring the largest possible output signal swing. In that case,

$$I_C R_L = \frac{V_{CC} - V_{CE,\,sat}}{2} \approx \frac{V_{CC}}{2} \tag{20.10}$$

Under this condition, the gain expression reduces to

$$A \approx -\frac{V_{CC}}{2\,V_T} \tag{20.11}$$

Since V_T is about 25 mV at room temperature, a 5-V supply yields a gain A approximately equal to 100.

The apparent paradox (a high R_L limits g_m and vice versa) could be circumvented if a scheme were conceived that combined a static low-R_L load line (allowing a high bias current and hence g_m) with a dynamic high-R_L load line. One potential solution is to steer some dc current away from R_L by means of a current source, as illustrated in Fig. 20.8(a). Figure 20.8(b) depicts the corresponding vertical movement of the load line. The current source can be made with a pnp transistor and three resistors, as shown in Fig. 20.8(c). Why this particular configuration makes a good current source will become clear later from the discussion of the reciprocal npn circuit.

The logical next step consists of the complete removal of R_L and replacing this *passive* component with an *active* pnp load. A conceptual drawing, including a current generator and mirror, is given in Fig. 20.9(a). Figure 20.9(b) shows that the operating point is now determined by the intersection of the npn and pnp I_C–V_{CE} curves. For the amplifier in Fig. 20.9(a),

$$A \approx -g_m(r_{o,n} \parallel r_{o,p}) \approx -\frac{I_C}{V_T}\left(\frac{V_{A,n}}{I_C} \parallel \frac{V_{A,p}}{I_C}\right) \approx -\frac{V_{A,n}\,V_{A,p}}{V_T(V_{A,n} + V_{A,p})} \tag{20.12}$$

Similar to Eq. (20.11), the gain expressed by Eq. (20.12) is independent of the bias current I_C. Furthermore, A no longer depends on the supply voltage V_{CC}. The circuit in Fig. 20.9(a), however, is only conceptual in nature. In practice, it is dc unstable. Indeed, due to the small slopes of the intersecting transistor characteristics (or, equivalently, the high output resistances of both transistors), minor mismatches between the two currents, small temperature differences, or simply basic variations in processing will cause a sizable shift of the operating point. Intuitively, some kind of feedback network is needed to guarantee quiescent stability. The discussion of such circuitry is deferred to Section 20.3 and Fig. 20.32.

Stabilizing the Common-Emitter's Operating Point

Up to this point, we have (implicitly) assumed that the operating point is set by means of a fixed dc voltage V_B at the base of the transistor. However, due to the exponential nature of the BJT, such arrangement leaves both I_C and the operating point very sensitive to even small changes in this base voltage. Indeed,

$$\frac{dI_C}{dV_{BE}} = \frac{d\left(I_S \exp\left(\frac{V_{BE}}{V_T}\right)\right)}{dV_{BE}} = \frac{I_S}{V_T}\exp\left(\frac{V_{BE}}{V_T}\right) = \frac{I_C}{V_T} \tag{20.13}$$

Therefore, $\Delta I_C/I_C = \Delta V_{BE}/V_T = \Delta V_B/V_T$. At room temperature, a 1-mV change in V_B translates into a 4% collector current change. Furthermore, the strong temperature dependence of V_T as well as I_S requires V_B to decrease by about 2 mV per degree rise in temperature if a constant current I_C is to be

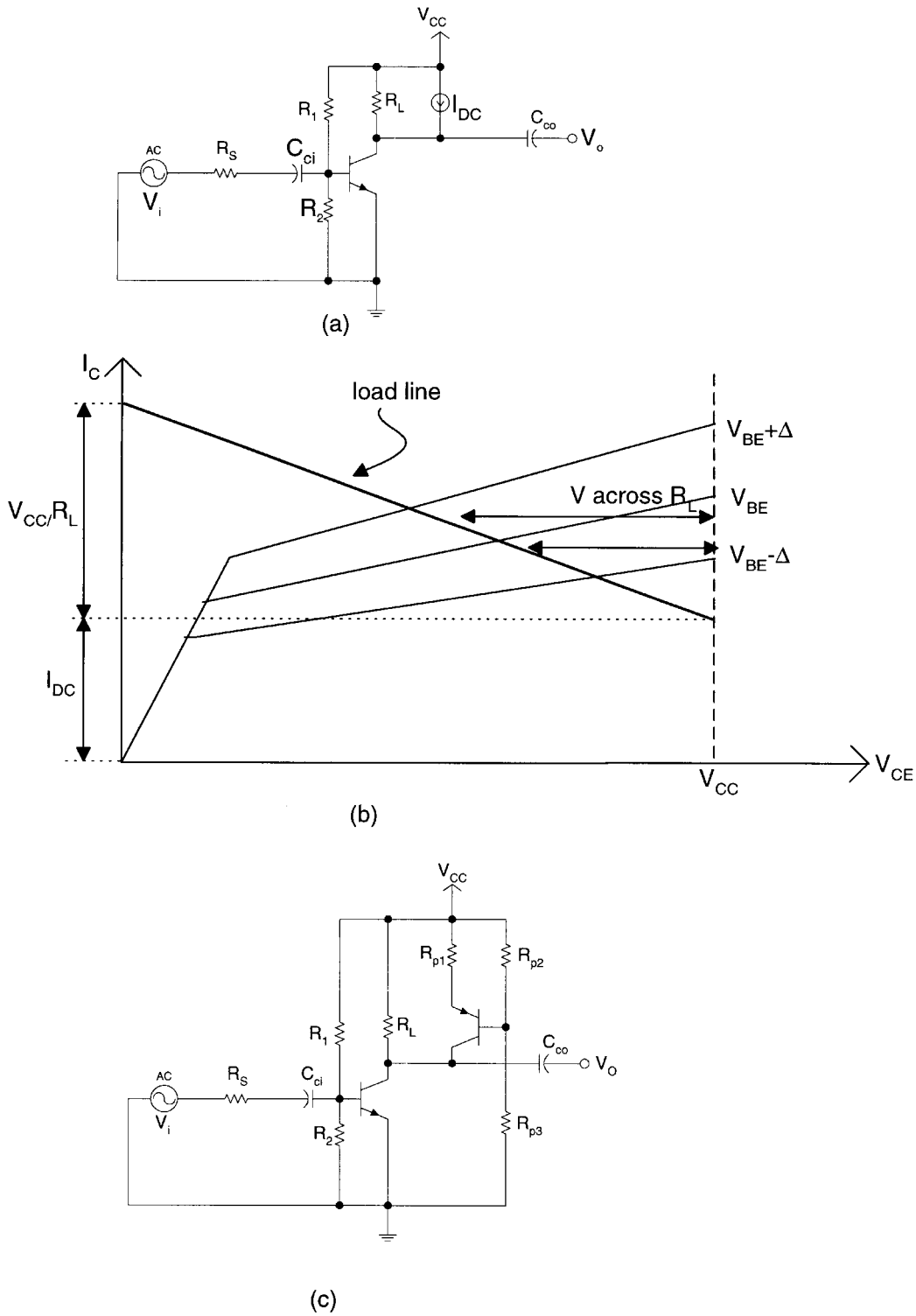


FIGURE 20.8 (a) Common-emitter amplifier with dc current bypass of R_L; (b) I_C vs. V_{CE} diagram with load line; (c) current source implemented by means of pnp and three resistors.

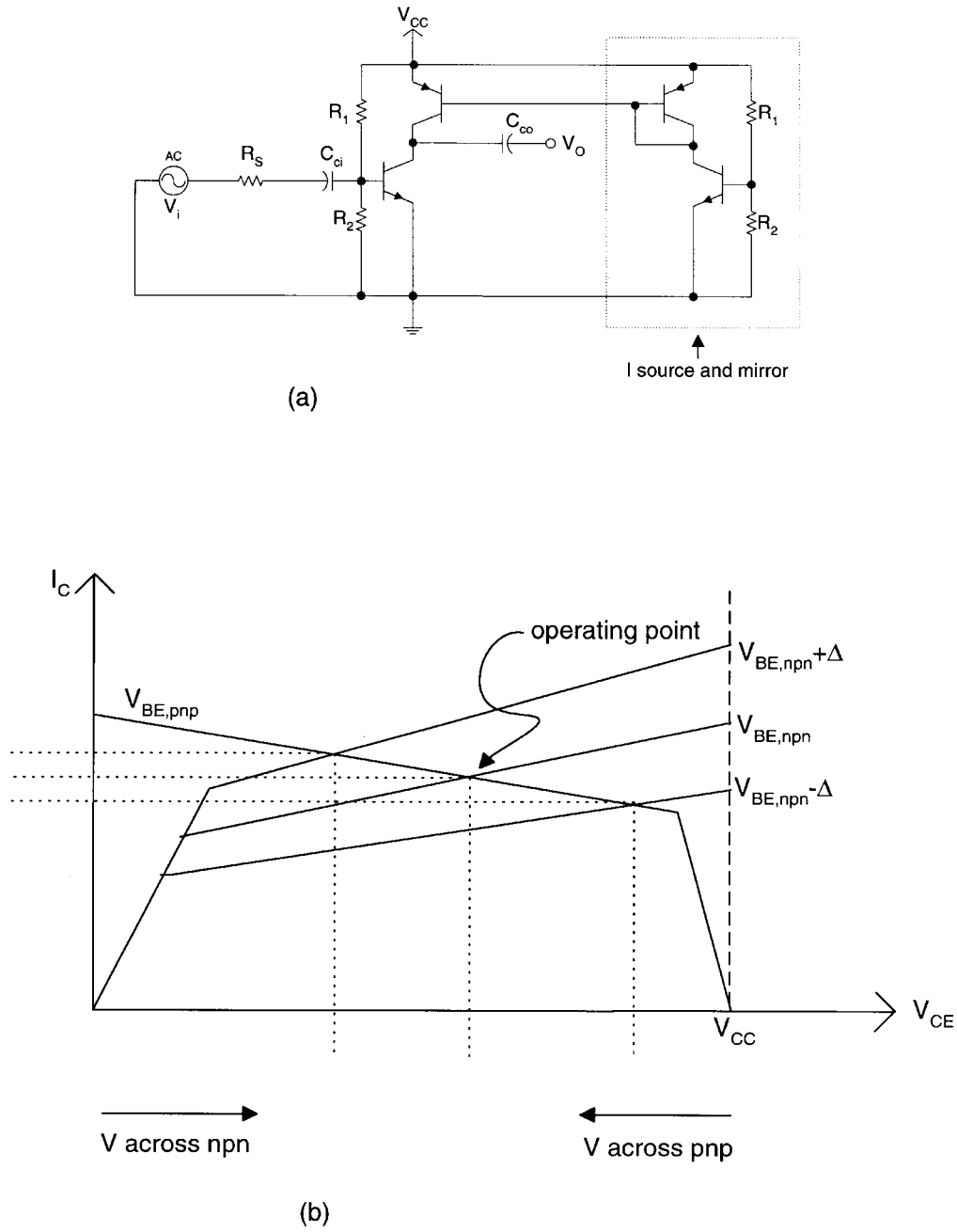


FIGURE 20.9 (a) Common-emitter amplifier with active load; (b) I_C vs. V_{CE} diagram.

maintained. Rather than fixing the voltage V_B at the base, one could try to force a fixed base current I_B and rely on the $I_C = \beta I_B$ relationship to set the collector current. However, β is strongly temperature dependent as well as subject to wide process variations. Moreover, β varies according to the transistor's current bias.

Consequently, the only way a stable operating point can be secured is by fixing the emitter current I_E. Since

$$I_C = \alpha I_E = \frac{\beta}{\beta + 1} I_E \approx I_E \tag{20.14}$$

fixing I_E yields a predictable collector current I_C. In discrete realizations, the value of R_L is well-determined (e.g., better than 1%). In integrated circuits (ICs), on the other hand, resistors are subject to wide absolute value tolerances. However, resistor ratios are usually tightly controlled. If one is able to set $I_C = V_{ref}/R_C$, where V_{ref} is a well-defined voltage, such as a bandgap reference (see Section 20.5), the product $I_C R_L$ only depends on V_{ref} and a resistor ratio. Thus, the operating point can be firmly established.

A simple way to set I_E consists of inserting a resistor R_E in series with the emitter, as shown in Fig. 20.10. The insertion of R_E introduces negative feedback, which makes the biasing circuit self-correcting. By inspection,

$$V_B = V_{BE} + R_E I_E \approx V_{BE} + R_E I_C \tag{20.15}$$

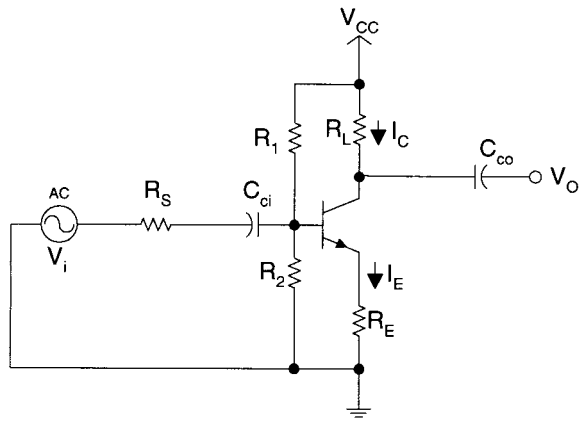


FIGURE 20.10 Common-emitter amplifier with degeneration resistor.

If the base current I_B is assumed to be much smaller than the current through the resistive divider network consisting of R_1 and R_2, then

$$V_B \approx \frac{R_2}{R_1 + R_2} V_{CC} \tag{20.16}$$

Combined with the Ebers-Moll identity

$$V_{BE} = V_T \ln\left(\frac{I_C}{I_S}\right) \tag{20.17}$$

Eqs. (20.15) and (20.16) can be solved for I_C. However, no closed-form solution is possible and numerical iterations must be performed. Intuitively, when I_C increases, the voltage drop across R_E also increases. Since V_B is constant, V_{BE} must go down, forcing I_C to decrease instead. Conversely, when I_C begins to decrease, the voltage across R_E goes down, V_{BE} is increased, which in turn leads to a reversal in the direction of the change in I_C. Eventually, an equilibrium point is reached. The dc voltage at the output can be expressed as

$$V_{o,dc} = V_{CC} - I_C R_C = V_{CC} - (V_B - V_{BE})\frac{R_L}{R_E} \tag{20.18}$$

Apparently, the arrangement of R_1, R_2, R_E and the npn transistor approximates a current source quite well. The approximation more closely approaches an ideal current source as the voltage drop across R_E is increased. In practice, however, the available supply voltage V_{CC} is fixed (or at least limited) and the emitter degeneration voltage subtracts from the achievable signal swing. Consequently, an acceptable compromise must be made.

The fact that the transistor's collector current is more or less constant, independent of the voltage at the output, implies that the configuration possesses a high output impedance. A mathematical expression for the output resistance of the degenerated common-emitter configuration can be derived from the small-signal equivalent circuit depicted in Fig. 20.11. If a test voltage V_{test} and a test current I_{test} are applied, the following equations can be written

$$I_{test} = g_m v_\pi + \frac{V_{test} - V_E}{r_o} \tag{20.19}$$

$$I_{test} = \frac{V_E}{R_E} + \frac{V_E}{r_b + R_S + r_\pi} \tag{20.20}$$

$$v_\pi = -\frac{r_\pi}{r_b + R_S + r_\pi} V_E \tag{20.21}$$

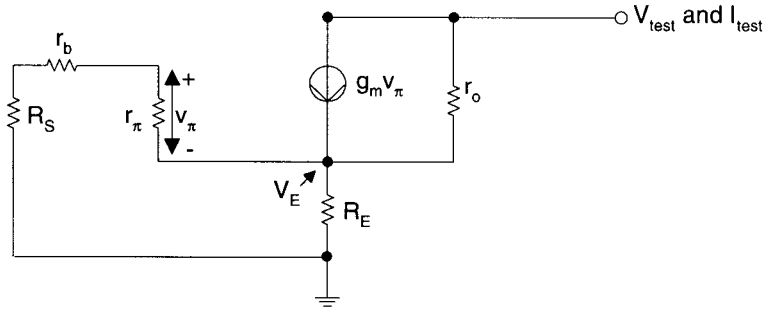


FIGURE 20.11 Small-signal equivalent circuit for calculation of the output resistance of the common-emitter configuration with degeneration.

The output resistance R_o is simply the ratio of V_{test} and I_{test}. Thus,

$$R_o = \frac{V_{test}}{I_{test}} = r_o\left(1 + \frac{g_m r_\pi R_E}{r_b + R_S + r_\pi + R_E} + \frac{R_E(r_b + R_S + r_\pi)}{r_o(r_b + R_S + r_\pi + R_E)}\right) \tag{20.22}$$

Assuming typical values for each of the resistances, Eq. (20.22) can be simplified to

$$R_o \approx r_o\left(1 + g_m R_E + \frac{R_E}{r_o}\right) \approx r_o(1 + g_m R_E) \tag{20.23}$$

In conclusion, the applied degeneration significantly boosts the transistor's output resistance r_o, more specifically by a multiplication factor $(1 + g_m R_E)$. The reader should take note that Fig. 20.11 and

Eq. (20.23) only represent the degenerated transistor. The output resistance of the complete degenerated common-emitter amplifier of Fig. 20.10 is the shunt combination of R_o and R_L.

The effect of R_E on the input impedance can similarly be derived from the small-signal equivalent circuit in Fig. 20.12.

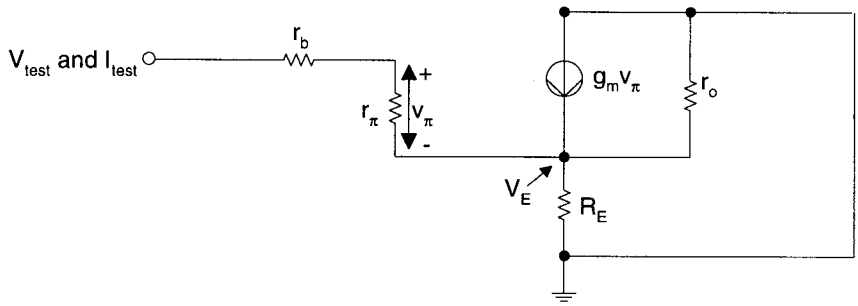


FIGURE 20.12 Small-signal equivalent circuit for calculation of the input resistance of the common-emitter configuration with degeneration.

$$I_{test} = \frac{V_E}{R_E} + \frac{V_E}{r_o} - g_m v_\pi \tag{20.24}$$

$$V_{test} = (r_b + r_\pi) I_{test} + V_E \tag{20.25}$$

$$v_\pi = r_\pi I_{test} \tag{20.26}$$

Combining Eqs. (20.24) through (20.26) yields

$$R_i = \frac{V_{test}}{I_{test}} = r_b + r_\pi + (1 + g_m r_\pi)\frac{r_o R_E}{r_o + R_E} \approx r_b + r_\pi + (\beta + 1)R_E \tag{20.27}$$

$$\approx r_\pi + \beta R_E \approx r_\pi(1 + g_m R_E)$$

The desirable multiplication factor found in Eq. (20.23) is likewise recognized in Eq. (20.27).

Lastly, the effect of R_E on the small-signal gain is calculated from the circuit in Fig. 20.13.

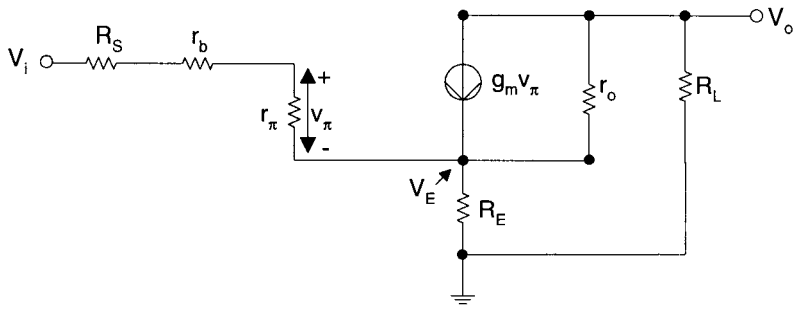


FIGURE 20.13 Small-signal equivalent circuit for calculation of the gain of the common-emitter amplifier with degeneration.

$$\frac{V_E}{R_E} + \frac{V_E - V_i}{r_b + R_S + r_\pi} + \frac{V_E - V_o}{r_o} - g_m v_\pi = 0 \tag{20.28}$$

$$\frac{V_o}{R_L} + \frac{V_o - V_E}{r_o} + g_m v_\pi = 0 \tag{20.29}$$

$$v_\pi = (V_i - V_E)\frac{r_\pi}{r_b + R_S + R_\pi} \tag{20.30}$$

After some manipulation, and assuming r_o is quite high, as well as r_π being much larger than either r_b, R_S, or R_L, one finds

$$A = \frac{V_o}{V_i} \approx -\frac{g_m R_L}{1 + g_m R_E} \approx -\frac{R_L}{R_E} \tag{20.31}$$

Again, the factor $(1 + g_m R_E)$ shows up. However, this time it appears in the denominator, which translates into a large unwanted reduction in gain. A possible solution lies in the use of a decoupling capacitor C_E placed in parallel with R_E. Then, Eq. (20.31) becomes

$$A \approx -\frac{g_m R_L (1 + sC_E R_E)}{1 + g_m R_E + sC_E R_E} \tag{20.32}$$

Eq. (20.32) has a zero at

$$s_z = -\frac{1}{C_E R_E} \tag{20.33}$$

and a pole at a higher frequency given by

$$s_p = -\frac{1 + g_m R_E}{C_E R_E} \approx -\frac{g_m}{C_E} \tag{20.34}$$

If C_E is chosen sufficiently large, the desirable dc degeneration resistance properties can be combined with a small-signal gain, which for the frequencies of interest approaches the $-g_m R_L$ value realized by the undegenerated common-emitter amplifier.

The stabilization method shown in Fig. 20.10 is very commonly used in discrete amplifier realizations. As mentioned before, integrated circuit implementations have the advantage of near-perfect matching between like components. Rather than stabilizing the current in each transistor by the method described above, usually a single accurate bias current generator is incorporated on a chip and the resulting reference current is mirrored throughout. The discussion of suitable voltage and current reference generators is postponed until Section 20.5. Figure 20.14(a) shows how a reference current I_{REF} is mirrored to the common-emitter amplifier by means of a basic npn current mirror. Following the previous derivations

$$R_{E,\,\text{eff}} = r_o(1 + g_m R_{E2}) = r_o\left(1 + \frac{I_{REF}}{V_T} R_{E2}\right) = \frac{V_A}{I_{REF}}\left(1 + \frac{I_{REF}}{V_T} R_{E2}\right) \approx \frac{V_A}{V_T} R_{E2} \tag{20.35}$$

$R_{E,\text{eff}}$ is much higher than could be obtained by a resistor only, under the restriction of an equal voltage drop. Thus, we have a higher performance current source. Also, for a given C_E, the pole and zero, which obey Eqs. (20.33) and (20.34), move to lower frequencies. Conversely, a lower valued C_E could be used.

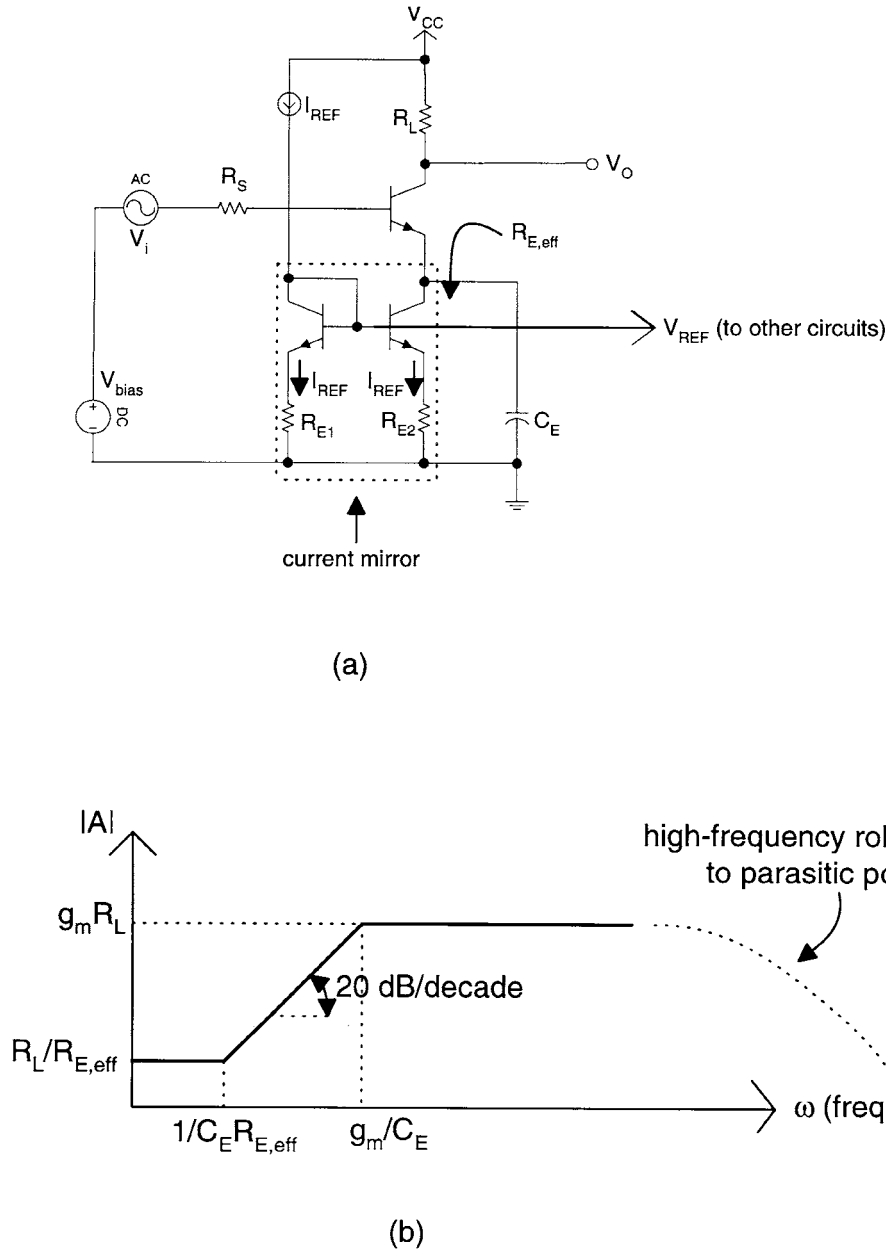


FIGURE 20.14 (a) Common-emitter amplifier with bias current generator and mirror, and (b) magnitude response vs. frequency.

This property is quite useful since integrated circuit capacitors consume a rather large amount of silicon real estate, and die size directly relates to cost.

Frequency Response of the Common-Emitter Amplifier

With the exception of C_E, we have thus far ignored the effect of the capacitors. At low frequencies, the ac coupling capacitances C_{ci} and C_{co} result in a gain reduction. In that sense, they behave similarly to C_E. At high frequencies, the internal transistor parasitics lead to a decrease in gain. Three parasitic

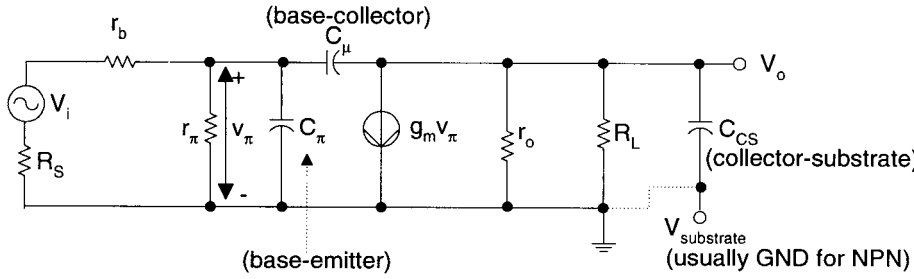


FIGURE 20.15 High-frequency small-signal equivalent circuit for the common-emitter amplifier.

capacitors must be accounted for: the emitter–base capacitance C_π, the collector–base capacitance C_μ, and the collector–substrate capacitance C_{cs}. Figure 20.15 shows the high-frequency small-signal equivalent circuit of the common-emitter amplifier. With R_i representing the parallel combination of $(R_S + r_b)$ with r_π, and R_C similarly designating the parallel combination of R_L with r_o, Eq. (20.9) must be rewritten as

$$A = -\frac{g_m R_i R_C}{R_S + r_b}\frac{N(s)}{D(s)} \tag{20.36}$$

where

$$\frac{N(s)}{D(s)} = \left(1 - \frac{sC_\mu}{g_m}\right)\Big/\left(1 + s(C_\pi R_i + C_\mu R_i + C_\mu R_C + C_\mu R_i R_C g_m + C_{cs} R_C) + s^2(C_\pi C_{cs} R_i R_C + C_\pi C_\mu R_i R_C + C_\mu C_{cs} R_i R_C)\right) \tag{20.37}$$

One observes that Eq. (20.37) contains a right half-plane zero located at $s_z = g_m/C_\mu$, resulting from the capacitive feedthrough from input to output. However, this right half-plane zero is usually at such a high frequency that it can be ignored in most applications. Furthermore, in most cases, one can assume that $D(s)$ contains a dominant pole p_1 and a second pole p_2 at a substantially higher frequency. If the dominant pole assumption is valid, $D(s)$ can be factored in the following manner

$$D(s) = \left(1 - \frac{s}{p_1}\right)\left(1 - \frac{s}{p_2}\right) \approx 1 - \frac{s}{p_1} + \frac{s^2}{p_1 p_2} \tag{20.38}$$

Equating Eqs. (20.37) and (20.38) yields

$$p_1 = -\frac{1}{R_i}\frac{1}{C_\pi + C_{cs}\dfrac{R_C}{R_i} + C_\mu\left(1 + g_m R_C + \dfrac{R_C}{R_i}\right)} \tag{20.39}$$

$$p_2 = -\frac{1}{R_C(C_\mu + C_{cs})}\frac{C_\pi + C_{cs}\dfrac{R_C}{R_i} + C_\mu\left(1 + g_m R_C + \dfrac{R_C}{R_i}\right)}{C_\pi + \dfrac{C_\mu C_{cs}}{C_\mu + C_{cs}}} \tag{20.40}$$

In Eqs. (20.39) and (20.40), the collector–base capacitance C_μ appears with a multiplication factor. One readily recognizes that this factor is dominated by the term $g_m R_C$, which is equal to the amplifier's gain A. A very important conclusion is that, looking into the common-emitter amplifier's input, the base–collector capacitance C_μ appears approximately A times larger than its actual physical value. This phenomenon is widely referred to as the *Miller effect*. As a result, C_μ often constitutes the predominant frequency limiter in high-gain common-emitter amplifiers.

If the dominant pole assumption is a valid model for the amplifier's high-frequency response, the –3 dB frequency can be more easily calculated by the method of *time constants*, rather than by deriving the complete transfer function. Indeed, it can be shown that

$$\omega_{3dB,HF} = \frac{1}{\sum_{n} C_i R_i} \tag{20.41}$$

where C_i are the capacitors in the high-frequency equivalent scheme (Fig. 20.15) and R_i are the resistances measured across each C_i when all other capacitors are replaced by open circuits. The reader can quickly verify that

$$R_\pi = r_\pi \| (r_b + R_S) = R_i \tag{20.42}$$

$$R_\mu = (r_\pi \| (r_b + R_S)) + (R_L \| r_o)(1 + g_m(r_\pi \| (r_b + R_S))) = R_i + R_C(1 + g_m R_i) \tag{20.43}$$

$$R_{Ccs} = R_L \| r_o = R_C \tag{20.44}$$

Then, according to Eq. (20.41),

$$\omega_{3dB,HF} = \frac{1}{R_\pi C_\pi + R_\mu C_\mu + R_{Ccs} C_{cs}} \tag{20.45}$$

which is identical to Eq. (20.39).

Likewise, one can show that if the low-frequency behavior can be described by a single-pole model,

$$\omega_{3dB,LF} = \sum_{j=1}^{m} \frac{1}{C_j R_j} \tag{21.46}$$

where C_j are the decoupling and coupling capacitors C_E, C_{ci}, C_{co}, and R_j are the resistances individually measured across their terminals when all other capacitors are simultaneously shorted.

This concludes the analysis of the common-emitter amplifier. The bipolar transistor, however, is a three-terminal device (actually there are four terminals if the substrate node is included), so there are obviously different ways to hook the BJT up into a circuit. Based on an understanding of the transistor's operation, it clearly does not make much sense to consider the base as an output, nor to use the collector as an input. This leaves us with two other meaningful possibilities, which will be studied in further detail.

Common-Collector Amplifier (Emitter Follower)

In the amplifier of Fig. 20.16, the input signal is provided to the base of the transistor, while the output is taken at the emitter. Similar to the common-emitter amplifier, this circuit configuration derives its name based on the terminal tied to a common ac ground node. In Fig. 20.16, the collector is connected directly to the supply voltage V_{CC}, hence the name *common-collector* amplifier.

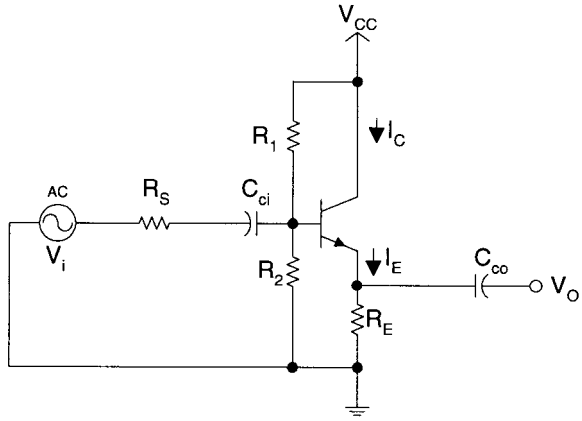


FIGURE 20.16 Common-collector amplifier (emitter follower).

Small-Signal Gain

The common-collector amplifier's mid-frequency small-signal equivalent circuit is shown in Fig. 20.17. By inspection,

$$\frac{V_o}{R_E} + \frac{V_o - V_i}{r_b + R_S + r_\pi} + \frac{V_o}{r_o} - g_m v_\pi = 0 \tag{20.47}$$

$$v_\pi = (V_i - V_o)\frac{r_\pi}{r_b + R_S + r_\pi} \tag{20.48}$$

$$A = \frac{V_o}{V_i} = \frac{1}{1 + \dfrac{r_b + R_S + r_\pi}{(\beta + 1)(R_E \parallel r_o)}} \tag{20.49}$$

$$A \approx \frac{1}{1 + \dfrac{r_\pi}{\beta R_E}} \approx \frac{g_m R_E}{1 + g_m R_E} \tag{20.50}$$

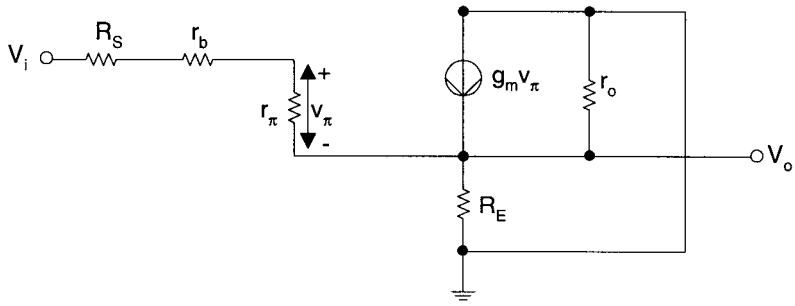


FIGURE 20.17 Small-signal equivalent circuit for calculation of the gain of the common-collector amplifier (emitter follower).

In most cases, $g_m R_E >> 1$ and, thus,

$$A \approx 1 \tag{20.51}$$

This means that the output voltage at the emitter nearly follows the input at the base. Eq. (20.51) is the reason why the common-collector amplifier is most often referred to as the *emitter follower* configuration.

The emitter follower's input resistance is identical to that of the degenerated common-emitter's. Thus, R_i is high and given by Eq. (20.27). Its output resistance R_o can be calculated from Fig. 20.17 by applying V_{test} and I_{test}, as demonstrated previously for the common-emitter circuit.

$$R_o = R_E \parallel r_o \parallel \frac{r_b + R_S + r_\pi}{\beta + 1} \approx \frac{1}{g_m} + \frac{R_S}{\beta} \tag{20.52}$$

Generally, Eq. (20.52) means that R_o is low. When the emitter follower is driven from a low-impedance (near-ideal) voltage source, R_o approaches its lower limit $1/g_m$. At room temperature, for example, a 1-mA bias current yields $R_o \approx 25\ \Omega$.

Since there is no gain, the emitter follower does not experience the aforementioned Miller effect and, correspondingly, has a wide bandwidth.

At this time, the reader may interject, "Why not use a (non-degenerated) common-emitter amplifier with a $1/g_m$ load, rather than the emitter follower?" Indeed, both circuits have identical input and output resistances as well as a gain of 1. The answer, however, does not lie in these small-signal parameters, but depends on the signal level. For example, for the common-emitter, a 1-V input is large (actually it would steer the transistor from the OFF state all the way into saturation). For the emitter follower, on the other hand, such a signal is considered small. Thus, the emitter follower is particularly suited as an output stage for large signals. As such, the emitter follower circuit will be revisited in Section 20.4.

In Fig. 20.18, R_E is replaced by a current source. As derived above, the effective degeneration resistor is now equal to $r_o(1 + g_m R_E)$, and thus much higher. According to Eq. (20.50), the circuit's gain is therefore even closer to 1. The drawback of using a current source is the introduction of the parasitic capacitor C_E, which adversely affects the frequency response.

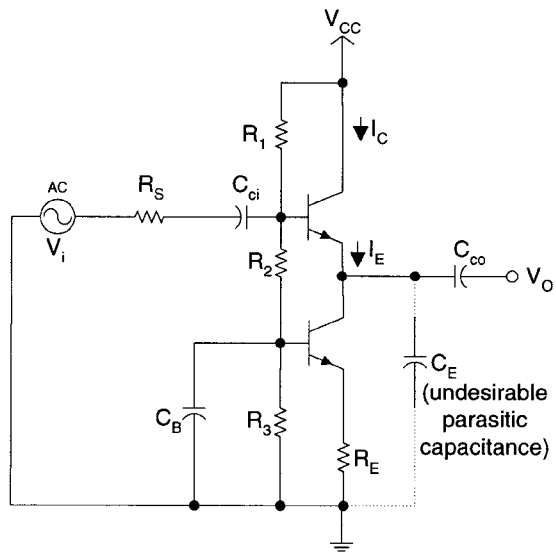


FIGURE 20.18 Common-collector amplifier (emitter follower) with current source bias.

Common-Base Amplifier

Figure 20.19 introduces an amplifier, where the input is applied at the emitter and the output taken at the collector. Since the base is connected to ac ground, this circuit is known as a *common-base* configuration.

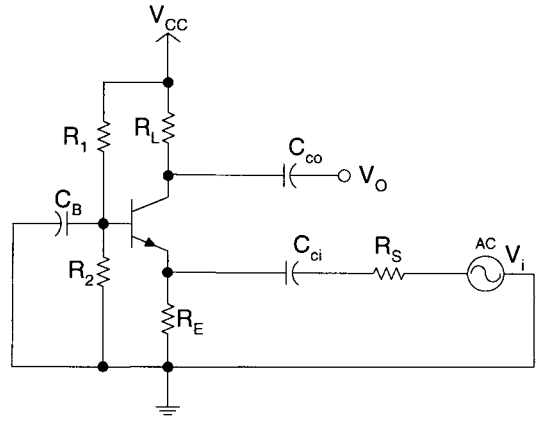


FIGURE 20.19 Common-base amplifier.

Subsequent to the analysis of the common-emitter and emitter follower configurations, it is straightforward to derive an expression for the input resistance R_i.

$$R_i \approx \frac{r_b + r_\pi}{\beta + 1} \approx \frac{1}{g_m} \tag{20.53}$$

According to Eq. (20.53), R_i is very low. Thus, it seems logical that the common-base circuit should be driven by a source with an equally low impedance. As we have seen above, such a source exists in the form of an emitter follower. Figure 20.20 shows the resulting combined circuit. Especially noteworthy in Fig. 20.20 is the symmetry in the arrangement of both transistors and the resistor R_E. This particular building block is called an *emitter-coupled pair*, as the reader might have expected based on the device connections. The emitter-coupled pair is by far the most frequently used configuration in integrated circuit bipolar amplifiers. It forms the cornerstone in the amplification of dc and difference signals and, as such, is the subject of an in-depth study in Section 20.3.

The common-base amplifier, however, can also be driven from a high-impedance source, as illustrated in Fig. 20.21. There, the outlined two-transistor arrangement is known as a *cascode* configuration. In Fig. 20.21, the common-emitter circuit at the bottom is loaded by the low-input impedance $(1/g_m)$ of the cascode common-base stage. Therefore, it provides no voltage amplification, but only current gain. The common-base stage, in turn, has a current gain approximately equal to 1 (to be exact, $\alpha = \beta/(\beta + 1)$), but contributes voltage gain by means of the load resistor R_L. The overall voltage transfer function is

$$A = \frac{V_o}{V_i} \approx -g_m R_L \tag{20.54}$$

The gain expressed in Eq. (20.54) is identical to the gain realized by the single-transistor common-emitter circuit. The reader may wonder, "What is the benefit of adding a second transistor in the common-base configuration?" The answer and advantage are related to the circuit's high-frequency response. Indeed, the common-base cascode stage effectively eliminates the Miller effect and, thereby, broadbands the circuit.

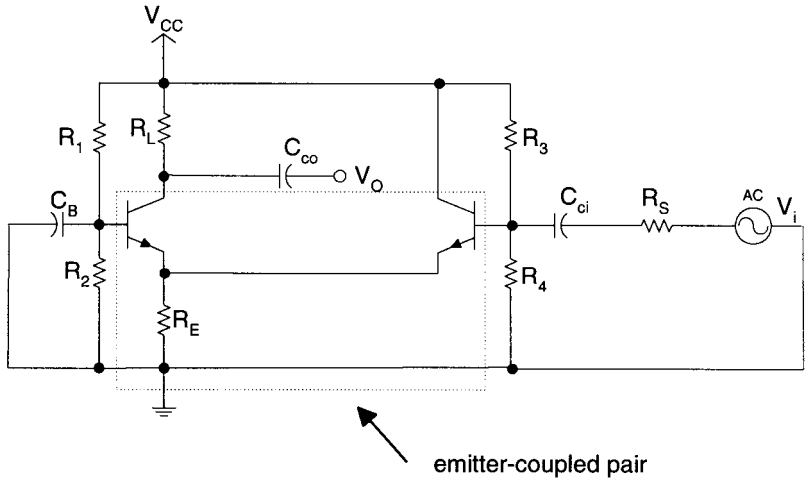


FIGURE 20.20 Common-base amplifier driven by an emitter follower.

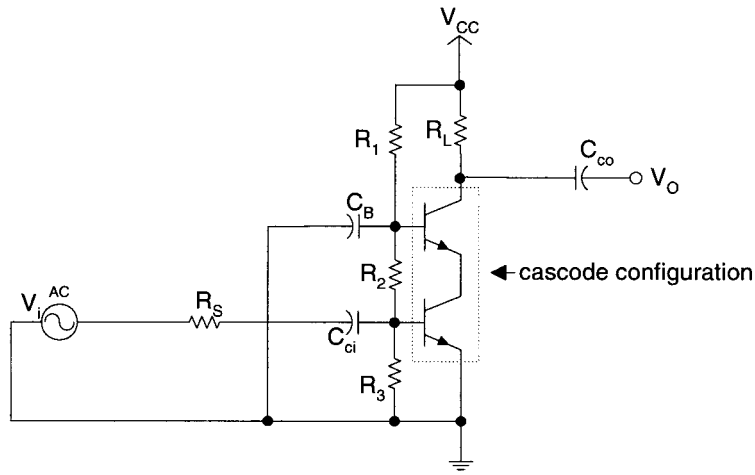


FIGURE 20.21 Common-base amplifier driven from common-emitter stage.

Darlington And Pseudo-Darlington Pairs

Whereas this chapter section was entitled single-transistor amplifiers, two two-transistor configurations, namely the emitter-coupled pair and the cascode arrangement, were briefly presented. Figure 20.22 introduces two other important two-transistor combinations: the *Darlington* and *pseudo-Darlington pairs*. The Darlington pair consists of two npn transistors connected as shown in Fig. 20.22(a). Combined, the two transistors effectively behave as a single npn with a (high) current gain $\beta = \beta_1\beta_2$. This improved current gain is traded off against a higher voltage requirement: the base–emitter voltage is twice that of a single transistor and the minimum base–collector voltage equals $(V_{CE,sat} + V_{BE})$. The pseudo-Darlington pair, shown in Fig. 20.22(b), is made out of the combination of a *npn* and a *pnp*. Both transistors jointly act as a single *pnp* with $\beta = \beta_1\beta_2$. The pseudo-Darlington's improved current gain is an important positive feature, as pnp's are generally far inferior in this respect compared to npn's. similar to the Darlington pair, the required collector–emitter voltage of the pseudo configuration is equal to $(V_{CE,sat} + V_{BE})$. Its

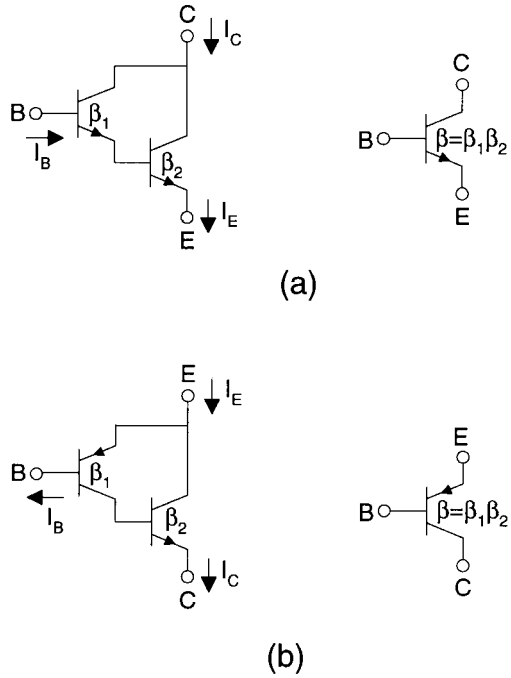


FIGURE 20.22 (a) Darlington pair, and (b) pseudo-Darlington pair.

base–emitter voltage requirement equals (V_{BE} + $V_{CE,sat}$). While higher than for a single transistor, this is not quite as high as needed by the cascade of two transistors.

20.3 Differential Amplifiers[1]

Introduction: Amplification of dc and Difference Signals

Differential amplifiers represent a class of amplifiers that amplify the difference between two signals, including dc. We will show that to obtain their desired characteristics, differential amplifiers critically depend on component matching. Therefore, discrete implementations necessitate careful component pairing, which is not only painstaking to the design engineer, but also significantly raises the amplifier's cost. In contrast, integrated circuit technology with its inherent small relative component tolerances is particularly suited for this application.

Section 20.2 emphasized that the active elements used for amplification are far from linear devices. To circumvent the problems associated with the bipolar transistor's non-linear input–output relationship, the amplifier circuits were linearized through the selection of a suitable operating point. Recalling Fig. 20.10, the input bias and dc level at the output were separated from the desired input–output signals by means of coupling capacitors C_{ci} and C_{co}. Further, an emitter degeneration resistor R_E reduced the drift of the operating point and a decoupling capacitor C_E was added to counteract the associated undesired gain reduction.

Obviously, the presence of coupling and decoupling capacitors makes the circuit in Fig. 20.10 unsuitable for dc amplification. But, even at low frequencies, this amplifier scheme is not usable due to the

[1] This section largely parallels the bipolar technology part of Ref. 12 by the same author.

large capacitor values required, which in turn give rise to large RC time constants and, consequently, slow recovery times from any transient disturbances.

The requirement to avoid capacitors in low-frequency or dc amplifiers unavoidably leads to a mixing of the concepts of bias and signal. A second characteristic of such amplifiers, as will become evident, is the application of symmetry to compensate for the drift of the active components.

An intuitive solution appears to lie in the use of two amplifiers connected in a difference configuration, as illustrated in Fig. 20.23. For this circuit, one can write the following equations

$$V_3 = A_1 V_1 \tag{20.55}$$

$$V_4 = A_2 V_2 \tag{20.56}$$

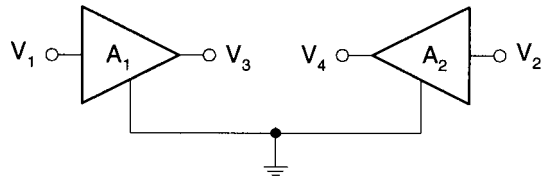


FIGURE 20.23 Initial concept of difference amplifier.

Assuming A_1 approximately equal to A_2 (i.e., $A_1 = A + \Delta/2$ and $A_2 = A - \Delta/2$) yields

$$V_3 - V_4 = A(V_1 - V_2) + \frac{\Delta}{2}(V_1 + V_2) \tag{20.57}$$

$$V_3 + V_4 = A(V_1 + V_2) + \frac{\Delta}{2}(V_1 - V_2) \tag{20.58}$$

Clearly, when both amplifiers are perfectly matched or $\Delta = 0$, the difference mode is completely separated from the sum mode.

Upon further reflection, however, the difference amplifier of Fig. 20.23 is not really the solution to the amplification problem under consideration. Indeed, in many instances, small signals, which sit on top of large pedestal voltages, need to be amplified. For example, assume $A = 100$, a difference signal of 10 mV, and a bias voltage equal to 10 V. The amplifier scheme of Fig. 20.23 would result in a 1-V difference signal in addition to a 1000-V output voltage common to both amplifiers. It is clearly unrealistic to assume that both amplifiers will remain linear and matched over such an extended voltage range. Instead, the real solution is obtained by coupling the amplifiers. The resulting arrangement is known as the *differential pair* amplifier.

In the mathematical description of the differential pair, four amplification factors are generally defined to express the relationship between the differential or difference (subscript D) and common or sum mode (subscript C) input and output signals. Applied to the circuit in Fig. 20.23, one can write

$$V_3 - V_4 = V_{oD} = A_{DD} V_{iD} + A_{DC} V_{iC} \tag{20.59}$$

$$V_3 + V_4 = 2V_{oC} = 2A_{CC} V_{iC} + 2A_{CD} V_{iD} \tag{20.60}$$

where $V_{iD} = V_1 - V_2$ and $V_{iC} = (V_1 + V_2)/2$.

The ratio A_{DD}/A_{CC} is commonly referred to as the amplifier's *common-mode rejection ratio* or CMRR. While an amplifier's CMRR is an important characteristic, maximizing its value is not a designer's goal

in itself. Rather, the real purpose is to suppress large sum signals so that the two amplifiers exhibit a small output swing and, thereby, indeed operate as a matched pair. Furthermore, the ultimate goal is to avoid that the application of a common-mode input signal results in a differential signal at the output. This objective can only be accomplished through a combination of careful device matching, precise selection of the amplifier's bias and operating point, as well as by a high common-mode rejection. While we have only considered differential output signals up to this point, in some instances a single-ended output is desired. Eqs. (20.59) and (20.60) can be rearranged as

$$V_3 = V_{oC} + \frac{1}{2}V_{oD} = \left(A_{CD} + \frac{1}{2}A_{DD}\right)V_{iD} + \left(A_{CC} + \frac{1}{2}A_{DC}\right)V_{iC} \tag{20.61}$$

$$V_4 = V_{oC} - \frac{1}{2}V_{oD} = \left(A_{CD} - \frac{1}{2}A_{DD}\right)V_{iD} + \left(A_{CC} - \frac{1}{2}A_{DC}\right)V_{iC} \tag{20.62}$$

One concludes that in the single-ended case, all three ratios A_{DD}/A_{CC}, A_{DD}/A_{CD}, and A_{DD}/A_{DC} must be high to yield the desired result.

Bipolar Differential Pairs (Emitter-Coupled Pairs)

Emitter-Coupled Pairs

Figure 20.24 depicts the basic circuit diagram of a bipolar differential pair. A differential signal V_{iD} is applied between the bases of two transistors, which, unless otherwise noted, are assumed to be identical. The dc bias voltage V_{bias} and a common-mode signal V_{iC} are also present. The transistors' common-emitter node (hence the name *emitter-coupled pair*) is connected to ground through a biasing network, which for simplicity is represented by a single resistor R_O. The amplifier output is taken differentially across the two collectors, which are tied to the power supply V_{CC} by means of a matched pair of load resistors R_L.

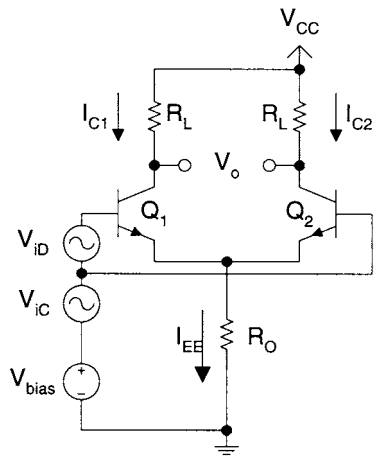


FIGURE 20.24 Emitter-coupled pair.

Low-Frequency Large-Signal Analysis

Applying the bipolar transistor's Ebers-Moll relationship with $V_{BE} >> V_T$ (where $V_T = kT/q$ is the thermal voltage) and assuming that both transistors are matched (i.e., the saturation currents $I_{S1} = I_{S2}$), the difference voltage V_{iD} can be expressed as follows

$$V_{iD} = V_{BE1} - V_{BE2} = V_T \ln\left(\frac{I_{C1}}{I_{C2}}\right) \tag{20.63}$$

After some manipulation and substituting $I_{C1} + I_{C2} = \alpha I_{EE}$ (the total current flowing through R_O), one gets

$$I_{C1} = \frac{\alpha I_{EE}}{1 + \exp\left(\frac{-V_{iD}}{V_T}\right)} \tag{20.64}$$

$$I_{C2} = \frac{\alpha I_{EE}}{1 + \exp\left(\frac{V_{iD}}{V_T}\right)} \tag{20.65}$$

where α is defined as $\beta/(\beta + 1)$.

Since $V_{oD} = -R_L (I_{C1} - I_{C2})$, the expression for the differential output voltage V_{oD} becomes

$$V_{oD} = -\alpha R_L I_{EE} \frac{\exp\left(\frac{V_{iD}}{2V_T}\right) - \exp\left(\frac{-V_{iD}}{2V_T}\right)}{\exp\left(\frac{V_{iD}}{2V_T}\right) + \exp\left(\frac{-V_{iD}}{2V_T}\right)} = -\alpha R_L I_{EE} \tanh\left(\frac{V_{iD}}{2V_T}\right) = -\alpha R_L I_{EE} \tanh x \tag{20.66}$$

The transfer function expressed in Eq. (20.66) is quite non-linear. A graphical representation is given in Fig. 20.25. When $V_{iD} > 2V_T$, the current through one of the two transistors is almost completely cut off and for further increases in V_{iD} the differential output signal eventually clips at $-\alpha R_L I_{EE}$. On the other hand, for small values of x, $\tanh x \approx x$. Under this small-signal assumption,

$$A_{DD} = \frac{V_{oD}}{V_{iD}} = -\frac{\alpha I_{EE}}{2V_T} R_L = -g_m R_L \tag{20.67}$$

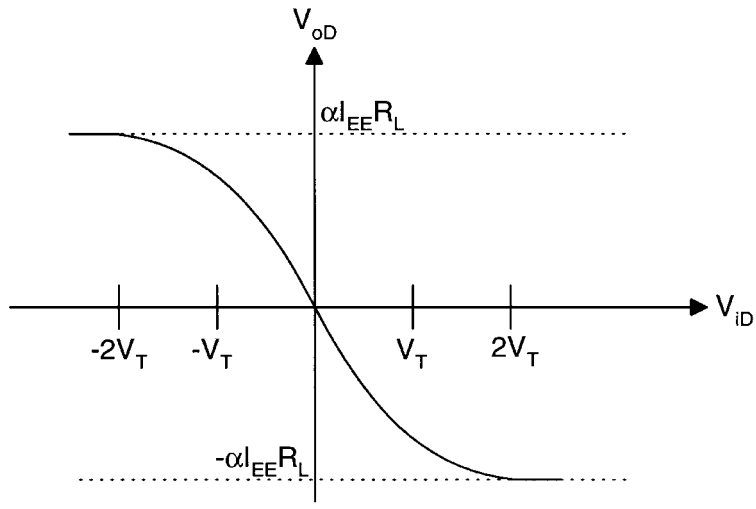


FIGURE 20.25 Emitter-coupled pair's dc transfer characteristic.

While the next subsection contains a more rigorous small-signal analysis, a noteworthy observation here is that, under conditions of equal power dissipation, the differential amplifier of Fig. 20.24 has only one half the transconductance value and hence only one half the gain of a single-transistor common-emitter amplifier. From Eq. (20.67), one furthermore concludes that when the tail current I_{EE} is derived from a voltage source, which is proportional to absolute temperature (PTAT), and a resistor of the same type as R_L, the transistor pair's differential gain is determined solely by a resistor ratio. As such, the gain is well-controlled and insensitive to absolute process variations. A similar observation was made in Section 20.2 during the discussion of the common-emitter amplifier's operating point stability. In Section 20.5, a suitable PTAT reference will be presented.

An intuitive analysis of the common-mode gain can be carried out under the assumption that R_O is large (e.g., assume R_O represents the output resistance of a current source). Then, a common-mode input signal V_{iC} results only in a small current change i_C through R_O and therefore V_{BE} remains approximately constant. With $i_C \approx V_{iC}/R_O$ and $V_{oC} = -R_L i_C/2$, the common-mode gain can be expressed as

$$A_{CC} = \frac{V_{oC}}{V_{iC}} \approx -\frac{R_L}{2R_o} \tag{20.68}$$

Combining Eqs. (20.67) and (20.68) yields

$$\text{CMRR} = \frac{A_{DD}}{A_{CC}} \approx 2g_m R_O \tag{20.69}$$

Half-Circuit Equivalents

Figure 20.26 illustrates the derivation of the emitter-coupled pair's differential mode half-circuit equivalent representation. For a small differential signal, the sum of the currents through both transistors remains constant and the current through R_O is unchanged. Therefore, the voltage at the emitters remains constant. The transistors operate as if no degeneration resistor were present, resulting in a high gain. In sum mode, on the other hand, the common resistor R_O provides negative feedback, which significantly lowers the common-mode gain. In fact, with identical signals at both inputs, the symmetrical circuit can be split into two halves, each with a degeneration resistor $2R_O$ as depicted in Fig. 20.27.

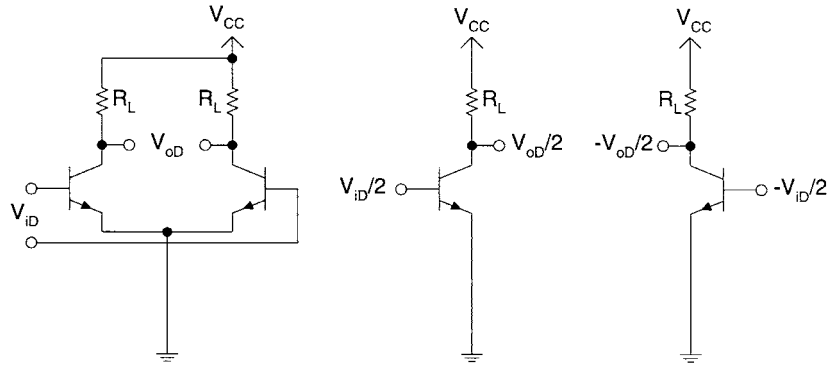


FIGURE 20.26 Difference mode.

Low-Frequency Small-Signal Analysis

Figure 20.28 represents the low-frequency small-signal differential mode equivalent circuit wherein R_{SD} models the corresponding source impedance. Under the presumption of matched devices,

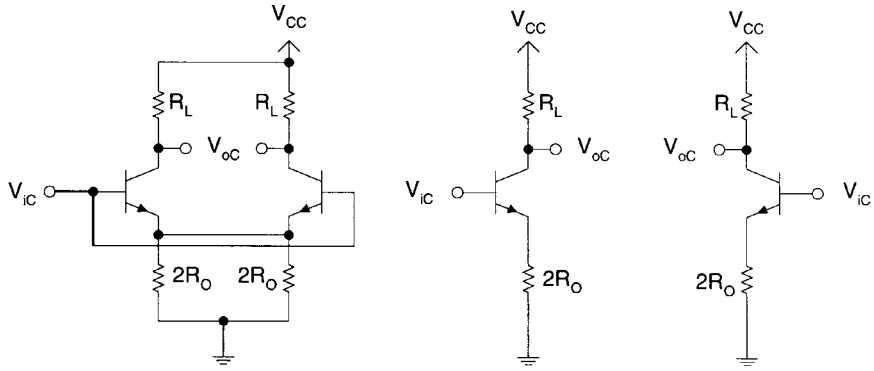


FIGURE 20.27 Sum mode.

$$A_{DD} = \frac{V_{oD}}{V_{iD}} = -g_m R_L \frac{r_\pi}{r_\pi + r_b + \frac{1}{2}R_{SD}} \tag{20.70}$$

With $r_b << r_\pi$ and assuming a low-impedance voltage source, Eq. (20.70) can be simplified to $-g_m R_L$ as was previously derived in Eqs. (20.67) or (20.9).

The low-frequency common-mode equivalent circuit is shown in Fig. 20.29. Under similar assumptions as in Eq. (20.70) and with R_{SC} representing the common-mode source impedance, one finds

$$A_{CC} = \frac{V_{oC}}{V_{iC}} = -g_m R_L \frac{r_\pi}{r_\pi + r_b + 2R_{SC} + 2(\beta + 1)R_O} \tag{20.71}$$

Upon substitution of $\beta = g_m r_\pi >> 1$, Eq. (20.71) reduces to $-R_L/2R_O$, the intuitive result obtained earlier in Eq. (20.68). For $R_E = 2R_O$, this result is also identical to Eq. (20.31).

The combination of Eqs. (20.70) and (20.71) leads to

$$\text{CMRR} = \frac{A_{DD}}{A_{CC}} = \frac{r_\pi + r_b + 2R_{SC} + 2(\beta + 1)R_O}{r_\pi + r_b + \frac{1}{2}R_{SD}} \approx 2g_m R_O \tag{20.72}$$

Let us consider the special case where R_O models the output resistance of a current source, implemented by a single bipolar transistor. Then, $R_O = V_A/I_{EE}$, where V_A is the transistor's Early voltage. With $g_m = \alpha I_{EE}/2V_T$,

$$\text{CMRR} = \frac{\alpha V_A}{V_T} \tag{20.73}$$

which is independent of the amplifier's bias conditions, but only depends on the process technology and temperature. At room temperature, with $\alpha \approx 1$ and $V_A \approx 25$ V, the amplifier's CMRR would be approximately 60 dB. The use of an improved current source, for example a bipolar transistor in series with an emitter degeneration resistor R_D, can significantly increase the CMRR. More specifically,

$$\text{CMRR} = \frac{\alpha V_A}{V_T}\left(1 + \frac{I_{EE}R_D}{V_T}\right) \tag{20.74}$$

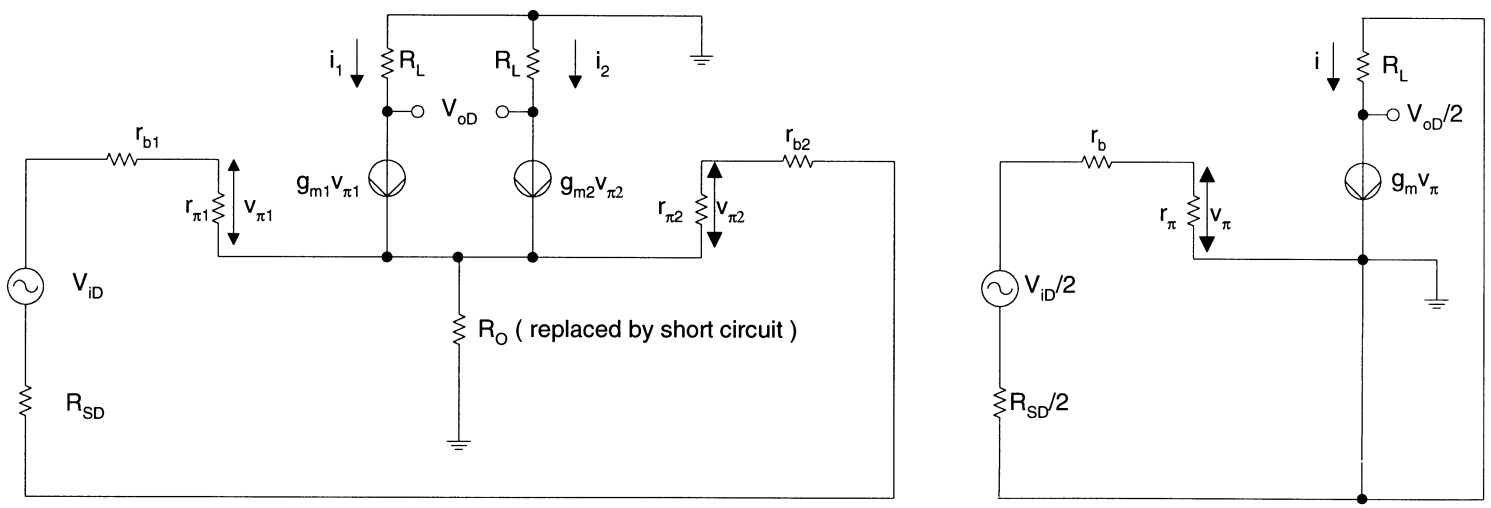


FIGURE 20.28 Small-signal equivalent circuit for difference mode.

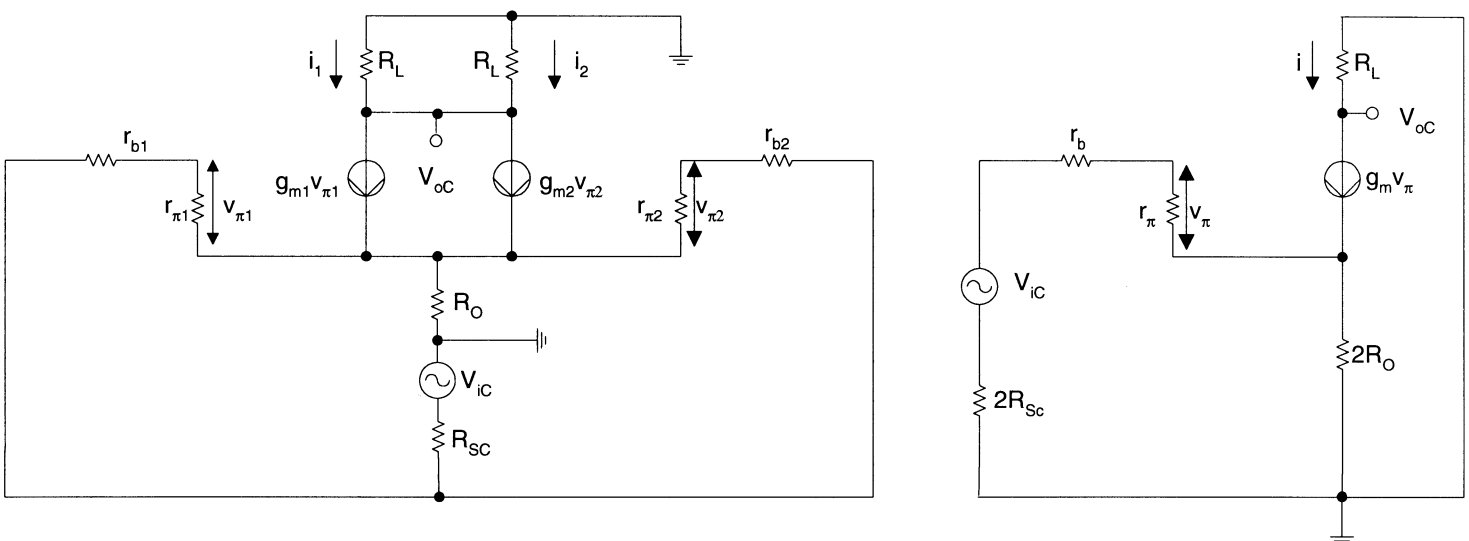


FIGURE 20.29 Small-signal equivalent circuit for sum mode.

For $I_{EE}R_D$ = 250 mV, the CMRR in Eq. (20.74) is eleven times higher than in Eq. (20.73).

In addition to expressions for the gain, the emitter-coupled pair's differential and common-mode input resistances can readily be derived from the small-signal circuits in Figs. 20.28 and 20.29.

$$R_{inD} = 2r_\pi \tag{20.75}$$

$$R_{inC} = \frac{1}{2}r_\pi + R_O(\beta + 1) \tag{20.76}$$

Taking into account the thermal noise of the transistors' base resistances and the load resistors R_L, as well as the shot noise caused by the collector currents, the emitter-coupled pair's total input referred squared noise voltage per Hertz is given by

$$\frac{V_{iN}^2}{\Delta f} = 8kT\left(r_b + \frac{1}{2g_m} + \frac{1}{g_m^2 R_L}\right) \tag{20.77}$$

Due to the presence of base currents, there is also a small input noise current, which however will be ignored here and in further discussions.

Small-Signal Frequency Response

When the emitter-base capacitance C_π, the collector-base capacitance C_μ, the collector-substrate capacitance C_{cs}, and the transistor's output resistance r_o are added to the transistor's hybrid-π small-signal model in Fig. 20.28, the differential gain transfer function becomes frequency dependent. Although the differential-mode small-signal equivalent circuit is identical to that of the non-degenerated common-emitter amplifier analyzed in Section 20.2, the high-frequency analysis is repeated here for the sake of completeness. With R_i representing the parallel combination of $(R_{SD}/2 + r_b)$ with r_π, and R_C similarly designating the parallel combination of R_L with r_o, Eq. (20.70) must be rewritten as

$$A_{DD} = -\frac{g_m R_i R_C}{\frac{1}{2}R_{SD} + r_b}\frac{N(s)}{D(s)} \tag{20.78}$$

where

$$\frac{N(s)}{D(s)} = \left(1 - \frac{sC_\mu}{g_m}\right)/(1 + s(C_\pi R_i + C_\mu R_i + C_\mu R_C + C_\mu R_i R_C g_m + C_{cs}R_C) + s^2(C_\pi C_{cs} R_i R_C + C_\pi C_\mu R_i R_C + C_\mu C_{cs} R_i R_C)) \tag{20.79}$$

As mentioned before, the right half-plane zero located at $s_z = g_m/C_\mu$ results from the capacitive feedthrough from input to output. This right half-plane zero is usually at such a high frequency that it can be ignored in most applications. Unlike in a single-ended amplifier, in a differential pair this zero can easily be canceled. One only has to add two capacitors $C_C = C_\mu$ between the bases of the transistors and the opposite collectors, as illustrated in Fig. 20.30(a). Rather than using physical capacitors, perfect tracking can be achieved by making use of the base–collector capacitances of transistors, whose emitters are either floating or shorted to the bases, as illustrated in Figs. 20.30(b) and (c). Note, however, that the compensating transistors contribute additional collector-substrate parasitics, which to some extent counteract the intended broadbanding effect.

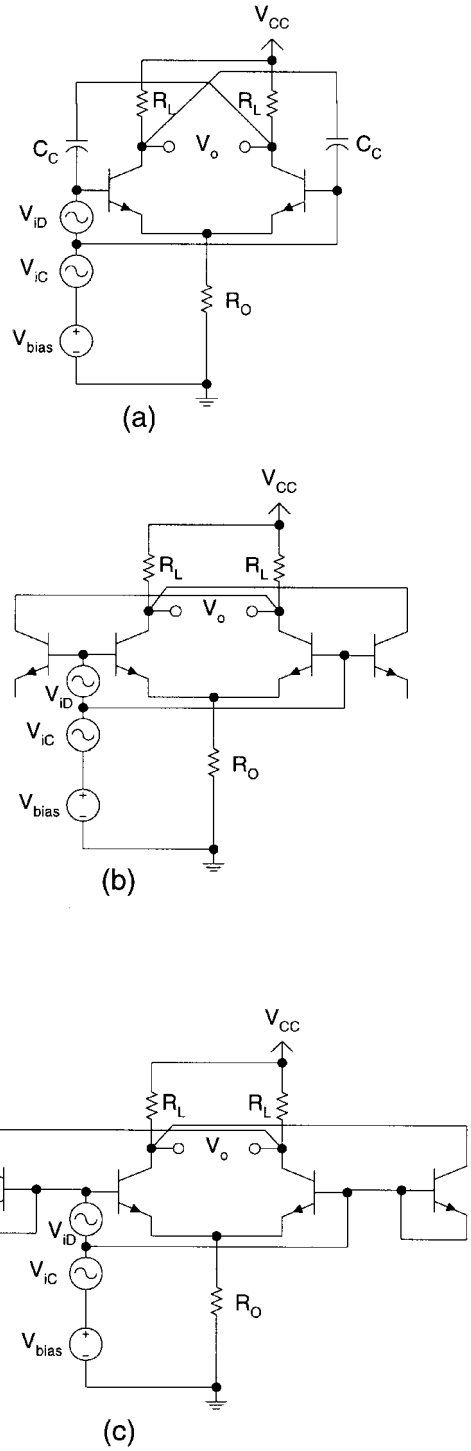


FIGURE 20.30 C_μ cancellation: (a) capacitor implementation, (b) transistor with floating emitter, and (c) transistor with shorted base–emitter junction.

If the dominant pole assumption is valid, $D(s)$ can be factored in the following manner

$$D(s) = \left(1 - \frac{s}{p_1}\right)\left(1 - \frac{s}{p_2}\right) \approx 1 - \frac{s}{p_1} + \frac{s^2}{p_1 p_2} \tag{20.80}$$

Equating Eqs. (20.79) and (20.80) yields

$$p_1 = -\frac{1}{R_i} \frac{1}{C_\pi + C_{cs}\frac{R_C}{R_i} + C_\mu\left(1 + g_m R_C + \frac{R_C}{R_i}\right)} \tag{20.81}$$

$$p_2 = -\frac{1}{R_C(C_\mu + C_{cs})} \frac{C_\pi + C_{cs}\frac{R_C}{R_i} + C_\mu\left(1 + g_m R_C + \frac{R_C}{R_i}\right)}{C_\pi + \frac{C_\mu C_{cs}}{C_\mu + C_{cs}}} \tag{20.82}$$

Rather than getting into a likewise detailed analysis, the discussion of the emitter-coupled pair's common-mode frequency response is limited here to the effect of the unavoidable capacitor C_O (representing, for instance, the collector–base and collector–substrate parasitic capacitances of the BJT), which shunts R_O. The parallel combination of R_O and C_O yields a zero in the common-mode transfer function. Correspondingly, a pole appears in the expression for the amplifier's CMRR. Specifically,

$$\text{CMRR} = 2g_m \frac{R_O}{1 + sC_O R_O} \tag{20.83}$$

The important conclusion from Eq. (20.83) is that at higher frequencies, the amplifier's CMRR rolls off by 20 dB per decade.

dc Offset

Input Offset Voltage

Until now, perfect matching between like components has been assumed. While ratio tolerances in integrated circuit technology can be very tightly controlled, minor random variations between "equal" components are unavoidable. These minor mismatches result in a differential output voltage, even if no differential input signal is applied. When the two bases in Fig. 20.24 are tied together, but the transistors and load resistors are slightly mismatched, the resulting differential output offset voltage can be expressed as

$$\begin{aligned} V_{oO} &= -(R_L + \Delta R_L)(I_C + \Delta I_C) + R_L I_C = \\ &-(R_L + \Delta R_L)(I_S + \Delta I_S)\exp\left(\frac{V_{BE}}{V_T}\right) + R_L I_S \exp\left(\frac{V_{BE}}{V_T}\right) \end{aligned} \tag{20.84}$$

or

$$V_{oO} \approx -\left(\frac{\Delta R_L}{R_L} + \frac{\Delta I_S}{I_S}\right) R_L I_S \exp\left(\frac{V_{BE}}{V_T}\right) = -\left(\frac{\Delta R_L}{R_L} + \frac{\Delta I_S}{I_S}\right) R_L I_C \tag{20.85}$$

Conversely, the output offset can be referred back to the input through a division by the amplifier's differential gain.

$$V_{iO} = \frac{V_{oO}}{-g_m R_L} = V_T\left(\frac{\Delta R_L}{R_L} + \frac{\Delta I_S}{I_S}\right) \tag{20.86}$$

The input referred offset voltage V_{iO} represents the voltage that must be applied between the input terminals in order to nullify the differential output voltage. In many instances, the absolute value of the offset voltage is not important because it can easily be measured and canceled, either by an auto-zero technique or by trimming. Rather, when offset compensation is applied, the offset stability under varying environmental conditions becomes the primary concern. The drift in offset voltage over temperature can be calculated by differentiating Eq. (20.86):

$$\frac{dV_{iO}}{dT} = \frac{V_{iO}}{T} \tag{20.87}$$

From Eq. (20.87), one concludes that the drift is proportional to the magnitude of the offset voltage and inversely related to the change in temperature.

Input Offset Current

Since in most applications the differential pair is driven by a low-impedance voltage source, its input offset voltage is an important parameter. Alternatively, the amplifier can be controlled by high-impedance current sources. Under this condition, the input offset current I_{iO}, which originates from a mismatch in the base currents, is the offset parameter of primary concern.

Parallel to the definition of V_{iO}, I_{iO} is the value of the current source that must be placed between the amplifier's open-circuited input terminals to reduce the differential output voltage to zero.

$$I_{iO} = \frac{I_C + \Delta I_C}{\beta + \Delta\beta} - \frac{I_C}{\beta} \approx \frac{I_C}{\beta}\left(\frac{\Delta I_C}{I_C} - \frac{\Delta\beta}{\beta}\right) \tag{20.88}$$

The requirement of zero voltage difference across the output terminals can be expressed as

$$(R_L + \Delta R_L)(I_C + \Delta I_C) = (R_L I_C) \tag{20.89}$$

Eq. (20.89) can be rearranged as

$$\frac{\Delta I_C}{I_C} \approx -\frac{\Delta R_L}{R_L} \tag{20.90}$$

Substituting Eq. (20.90) into Eq. (20.88) yields

$$I_{iO} = -\frac{I_{EE}}{2\beta}\left(\frac{\Delta R_L}{R_L} + \frac{\Delta\beta}{\beta}\right) \tag{20.91}$$

I_{iO}'s linear dependence on the bias current and its inverse relationship to the transistors' current gain β as expressed by Eq. (20.91) intuitively make sense.

Gain Enhancement Techniques

From Eq. (20.67), one concludes that there are two ways to increase the emitter-coupled pair's gain: namely, an increase in the bias current or the use of a larger valued load resistor. Similar to the earlier discussion of the common-emitter amplifier, practical limitations of the available supply voltage and the corresponding limit on the allowable I-R voltage drop across the load resistors (in order to avoid saturating either of the two transistors), however, limit the maximum gain that can be achieved by a single stage.

This section introduces two methods that generally allow the realization of higher gain while avoiding the dc bias limitations.

Negative Resistance Load

In the circuit of Fig. 20.31, a gain boosting positive feedback circuit is connected between the output terminals. The output dc bias voltage is simply determined by V_{CC}, together with the product of R_L and the current flowing through it, which is now equal to $(I_E + I_R)/2$. However, for ac signals, the added circuit — consisting of two transistors with cross-coupled base–collector connections and the resistors R_C between the emitters — represents a negative resistance of value $-2(R_C + 1/g_{mc})$, where $g_{mc} = \alpha I_R/2V_T$. The amplifier's differential gain can now be expressed as

$$A_{DD} \approx -g_m R_L \frac{1}{1 - \dfrac{g_{mc} R_L}{1 + g_{mc} R_C}} \tag{20.92}$$

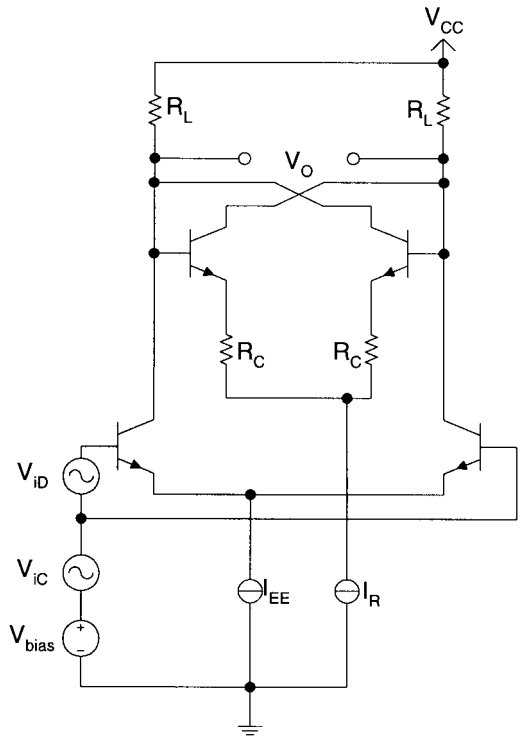


FIGURE 20.31 Emitter-coupled pair with negative resistor load.

Active Load

Another approach to increase the gain consists of replacing the load resistors by active elements, such as pnp transistors. Figure 20.32 shows a fully differential realization of an emitter-coupled pair with active loads. The differential gain is determined by the product of the transconductance of the input devices and the parallel combination of the output resistances of the npn and pnp transistors. Since $g_m = I_C/V_T$, $r_{on} = V_{An}/I_C$, and $r_{op} = V_{Ap}/I_C$, the gain becomes

$$A_{DD} = -g_m \frac{V_{An} V_{Ap}}{(V_{An} + V_{Ap}) I_C} = -\frac{1}{V_T} \frac{V_{An} V_{Ap}}{V_{An} + V_{Ap}} \tag{20.93}$$

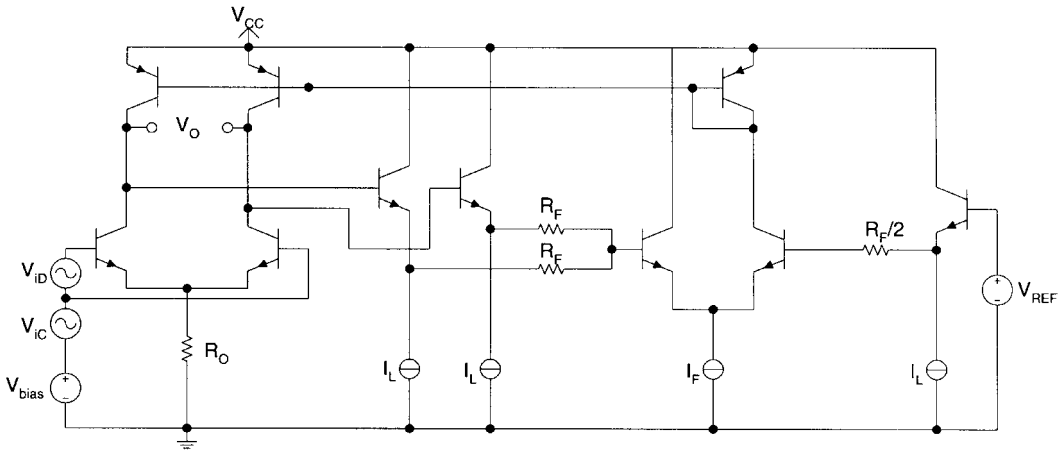


FIGURE 20.32 Emitter-coupled pair with active pnp load.

Consequently, the gain is relatively high and independent of the bias conditions. The disadvantage of the fully differential realization with active loads is that the output common-mode voltage is not well-defined. This problem also exists for the corresponding single-ended implementation (see Fig. 20.9(a)), as mentioned in Section 20.2. If one were to use a fixed biasing scheme for both types of transistors in Fig. 20.32, minor, but unavoidable mismatches between the currents in the npn and pnp transistors will result in a significant shift of the operating point. The solution lies in a common-mode feedback (CMFB) circuit that controls the bases of the active loads and forces a predetermined voltage at the output nodes. The CMFB circuit has high gain for common-mode signals, but does not respond to differential signals present at its inputs. A possible realization of such a CMFB circuit is seen in the right portion of Fig. 20.32. Via emitter followers and resistors R_P the output nodes are connected to one input of a differential pair, whose other input terminal is similarly tied to a reference voltage V_{REF}. The negative feedback provided to the pnp load transistors forces an equilibrium state where the dc voltages at the output terminals of the differential pair gain stage are equal to V_{REF}. An alternative active load implementation with a single-ended output is shown in Fig. 20.33. Contrary to the low CMRR of a single-ended realization

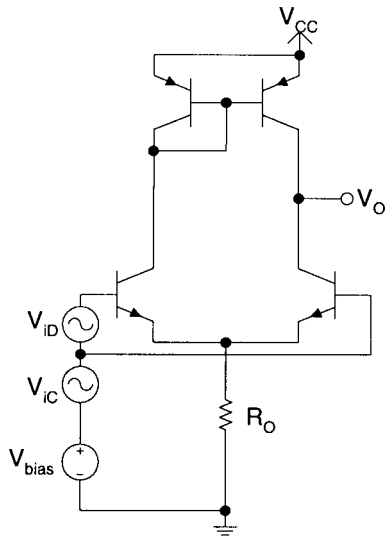


FIGURE 20.33 Emitter-coupled pair with active load and single-ended output.

with resistive loads, the circuit in Fig. 20.33 inherently possesses the same CMRR as a differential realization since the output voltage depends on a current differencing as a result of the pnp mirror configuration. The drawback of the single-ended circuit is a lower frequency response, particularly when low-bandwidth lateral pnp transistors are used.

Linearization Techniques

As derived previously, the linear range of operation of the emitter-coupled pair is limited to approximately $V_{iD} \approx 2V_T$. This section describes two techniques that can be used to extend the linear range of operation.

Emitter Degeneration

The most common technique to increase the linear range of the emitter-coupled pair relies on the inclusion of emitter degeneration resistors, as shown in Fig. 20.34. The analysis of the differential gain transfer function proceeds as before; however, no closed-form expression can be derived. Intuitively, the inclusion of R_E introduces negative feedback, which lowers the gain and extends the amplifier's linear operating region to a voltage range approximately equal to the product of $R_E I_E$. The small-signal differential gain can be expressed as

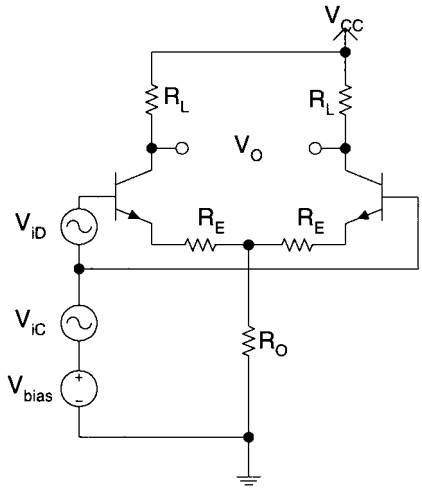


FIGURE 20.34 Differential pair with degeneration resistors.

$$A_{DD} \approx -G_M R_L \tag{20.94}$$

where G_M is the effective transconductance of the degenerated input stage. Therefore,

$$G_M = \frac{g_m}{1 + g_m R_E} \approx \frac{1}{R_E} \tag{20.95}$$

Consequently,

$$A_{DD} \approx -\frac{g_m R_L}{1 + g_m R_E} \tag{20.96}$$

In case $g_m R_E >> 1$,

$$A_{DD} \approx -\frac{R_L}{R_E} \tag{20.97}$$

In comparison to the undegenerated differential pair, the gain is reduced by an amount $(1 + g_m R_E) \approx g_m R_E$, which is proportional to the increase in linear input range. The common-mode gain transfer function for the circuit in Fig. 20.34 is

$$A_{CC} \approx -\frac{R_L}{2R_O + R_E} \tag{20.98}$$

For practical values of R_E, A_{CC} remains relatively unchanged compared to the undegenerated prototype. As a result, the amplifier's CMRR is reduced approximately by the amount $g_m R_E$. Also, the input referred squared noise voltage per Hertz can be derived as

$$\frac{V_{iN}^2}{\Delta f} = 8kT\left[r_b + \frac{1}{2g_m}(g_m^2 R_E^2) + \frac{1}{g_m^2 R_L}(g_m^2 R_E^2) + R_E\right] \tag{20.99}$$

This means that, to a first order, the noise also increases by the factor $g_m R_E$. Consequently, although the amplifier's linear input range is increased, its signal-to-noise ratio (SNR) remains unchanged. To complete the discussion of the emitter degenerated differential pair, the positive effect emitter degeneration has on the differential input resistance R_{inD}, and, to a lesser extent, on R_{inC} should be mentioned. For the circuit in Fig. 20.34,

$$R_{inD} = 2[r_\pi + (\beta + 1)R_E] \tag{20.100}$$

$$R_{inC} = \frac{1}{2}r_\pi + \frac{(\beta + 1)R_E}{2} + R_O(\beta + 1) \tag{20.101}$$

Parallel Combination of Asymmetrical Differential Pairs

A second linearization technique consists of adding the output currents of two parallel asymmetrical differential pairs with respective transistor ratios 1:r and r:1 as shown in Fig. 20.35. The reader will

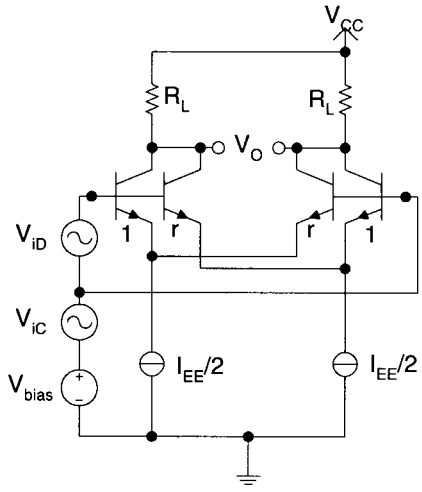


FIGURE 20.35 Parallel asymmetrical pairs.

observe that each differential pair in Fig. 20.35 is biased by a current source of magnitude $I_{EE}/2$ so that the power dissipation as well as the output common-mode voltage remain the same as for the prototype circuit in Fig. 20.24. Assuming, as before, an ideal exponential input voltage–output current relationship for the bipolar transistors, the following voltage transfer function can be derived:

$$V_{oD} = -\frac{\alpha}{2} I_{EE} R_L \left[\tanh\left(\frac{V_{iD}}{2V_T} - \frac{\ln r}{2} \right) + \tanh\left(\frac{V_{iD}}{2V_T} + \frac{\ln r}{2} \right) \right] \tag{20.102}$$

After Taylor series expansion and some manipulation, Eq. (20.102) can be rewritten as

$$V_{oD} = -\alpha I_{EE} R_L (1 - d) \left[\frac{V_{iD}}{2V_T} + \left(d - \frac{1}{3} \right) \left(\frac{V_{iD}}{2V_T} \right)^3 + \ldots \right] \tag{20.103}$$

where

$$d = \left(\frac{r-1}{r+1} \right)^2 \tag{20.104}$$

Equation (20.103) indicates that the dominant third-harmonic distortion component can be canceled by setting $d = 1/3$ or $r = 2 + \sqrt{3} = 3.732$. The presence of parasitic resistances within the transistors tends to require a somewhat higher ratio r for optimum linearization. In practice, the more easily realizable ratio $r = 4$ (or $d = 9/25$) is frequently used. When the linear input ranges at a 1% total harmonic distortion (THD) level of the single symmetrical emitter-coupled pair in Fig. 20.24 and the dual circuit with $r = 4$ in Fig. 20.35 are compared, a nearly threefold increase is noted. For $r = 4$ and neglecting higher-order terms, Eq. (20.103) becomes

$$A_{DD} \approx -0.64 g_m R_L \tag{20.105}$$

where $g_m = \alpha I_{EE}/2V_T$ as before. Equation (20.105) means that the tradeoff for the linearization is a reduction in the differential gain to 64% of the value obtained by a single symmetrical emitter-coupled pair with equal power dissipation. The squared input referred noise voltage per Hertz for the two parallel asymmetrical pairs can be expressed as

$$\frac{V_{iN}^2}{\Delta f} = \frac{8kT}{(0.64)^2} \left(\frac{r_b}{5} + \frac{1}{2g_m} + \frac{1}{g_m^2 R_L} \right) \tag{20.106}$$

The factor $r_b/5$ appears because of an effective reduction in the base resistance by a factor $(r + 1)$ due to the presence of five transistors vs. one in the derivation of Eq. (20.77). If the unit transistor size in Fig. 20.35 is scaled down accordingly, a subsequent comparison of Eqs. (20.77) and (20.106) reveals that the input referred noise for the linearized circuit of Fig. 20.35 is 1/0.64, or 1.56 times higher than for the circuit in Fig. 20.24. Combined with the nearly threefold increase in linear input range, this means that the SNR nearly doubles. The approximately 6-dB increase in SNR is a distinct advantage over the emitter degeneration linearization technique. Moreover, the linearization approach introduced in this section can be extended to a summation of the output currents of three, four, or more parallel asymmetrical pairs. However, there is a diminished return in the improvement. Also, for more than two pairs, the required device ratios become quite large and the sensitivity of the linear input range to small mismatches in the actual ratios versus their theoretical values increases as well.

Rail-to-Rail Common-Mode Inputs and Minimum Supply Voltage Requirement

With the consistent trend toward lower power supplies, the usable input common-mode range as a percentage of the supply voltage is an important characteristic of differential amplifiers. Full rail-to-rail input compliance is a highly desirable property. Particularly for low-power applications, the ability to operate from a minimal supply voltage is equally important. For the basic emitter-coupled pair in Fig. 20.24, the input is limited on the positive side when the npn transistors saturate. Therefore,

$$V_{\text{iC, pos}} = VCC - \frac{1}{2} R_{\text{L}} I_{\text{EE}} + V_{\text{bc, forward}} \tag{20.107}$$

If one limits $R_{\text{L}} I_{\text{EE}}/2 < V_{\text{bc,forward}}$, $V_{\text{iC,pos}}$ can be as high as V_{CC} or even slightly higher. On the negative side, the common-mode input voltage is limited to that level, where the tail current source starts saturating. Assuming a single bipolar transistor is used as the current source,

$$V_{\text{iC, neg}} > V_{\text{BE}} + V_{\text{CE, sat}} \approx 1 \text{ V} \tag{20.108}$$

The opposite relationships hold for the equivalent pnp transistor-based circuit. As a result, the rail-to-rail common-mode input requirement can be resolved by putting two complementary stages in parallel. In general, as the input common-mode traverses between V_{CC} and ground, three distinct operating conditions can occur: (1) at high voltage levels, only the npn stage is active; (2) at intermediate voltage levels, both the npn and pnp differential pairs are enabled; and finally, (3) for very low input voltages, only the pnp stage is operating. If care is not taken, three distinct gain ranges can occur: based on g_{mn} only; resulting from $g_{\text{mn}} + g_{\text{mp}}$; and, contributed by g_{mp} only. Non-constant g_{m} and gain that depends on the input common-mode is usually not desirable for several reasons, not the least of which is phase compensation if the differential pair is used as the first stage in an operational amplifier. Fortunately, the solution to this problem is straightforward if one recognizes that the transconductance of the bipolar transistor is proportional to its bias current. Therefore, the only requirement for a bipolar constant-g_{m} complementary circuit with full rail-to-rail input compliance is that under all operating conditions the sum of the bias currents of the npn and pnp subcircuits remains constant. A possible implementation is shown in Fig. 20.36. If $V_{\text{iC}} < V_{\text{REF}}$, the pnp input stage is enabled and the npn input transistors are off. When $V_{\text{iC}} > V_{\text{REF}}$, the bias current is switched to the npn pair and the pnp input devices turn off. For $R_{\text{L}} I_{\text{EE}}/2 < V_{\text{cb,forward,n}}$, the minimum required power supply voltage is $V_{\text{BE,n}} + V_{\text{BE,p}} + V_{\text{CE,sat,n}} + V_{\text{CE,sat,p}}$, which is lower than 2 V.

20.4 Output Stages

Output stages are specially designed to deliver large signals (as close to rail-to-rail as possible) and a significant amount of power to a specified load. The load to be driven is often very low-ohmic in nature; for example, 4 to 8 ohms in the case of audio loudspeakers. Therefore, output stages must possess a low output impedance and be able to supply high amounts of current — without distorting the signal. The output stage also needs to have a relatively wide bandwidth, so that it does not contribute major frequency limitations to the overall amplifier. Equally desirable is a high efficiency in the power transfer. Preferably, the output stage consumes no (or very little) power in the quiescent state, when no input signal is present. In this section, two major classes of amplifiers, distinguished by their quiescent power needs, are discussed. *Class A amplifiers* consume the same amount of power regardless of the presence of an ac signal. *Class B amplifiers*, on the other hand, only consume power while activated by an input signal and dissipate absolutely no power in stand-by mode. A hybrid between these two distinct cases is *Class AB* operation, which consumes only a small amount of quiescent power. Because output stages deliver high power levels, care must be taken to guarantee

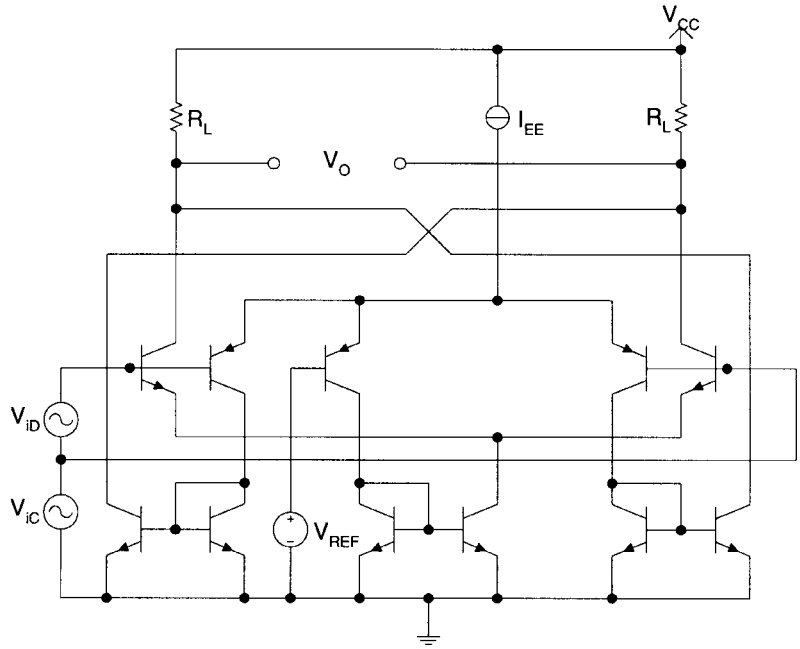


FIGURE 20.36 Low-voltage rail-to-rail input circuit.

that the transistors do not exceed their maximum power ratings, even under unintended operating conditions, such as when the output is shorted to ground. To avoid permanent damage or total destruction, output stages often include some sort of overload protection circuitry.

Class A Operation

The emitter follower, analyzed in Section 20.2, immediately comes to mind as a potential output stage configuration. The circuit is revisited here with an emphasis on its signal-handling capability and power efficiency. Figure 20.37(a) shows an emitter follower transistor Q_1 biased by a current source Q_2 and loaded by a resistor R_L. For generality, separate positive (V_{CC}) and negative ($-V_{EE}$) supplies, as well as ground are used in Fig. 20.37(a) and the analysis below.

The following identities can readily be derived

$$V_i = V_{BE,1} + V_o \tag{20.109}$$

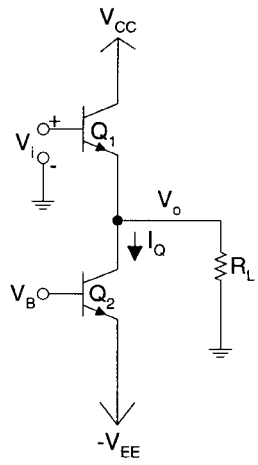
$$V_i = V_T \ln \frac{I_1}{I_S} + V_o = V_T \ln \frac{I_Q + \dfrac{V_o}{R_L}}{I_S} + V_o \tag{20.110}$$

where I_Q is the quiescent current supplied by the current source Q_2.

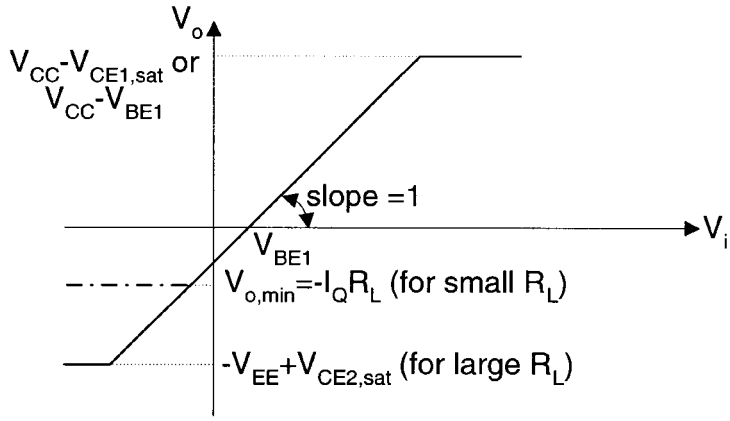
If one assumes that R_L is quite large, the output current $I_o = V_o/R_L$ is relatively small and $V_{BE,1}$ is approximately constant. Then,

$$V_o \approx V_i - V_T \ln \frac{I_Q}{I_S} \tag{20.111}$$

or there is approximately a fixed voltage drop between input and output.



(a)



(b)

FIGURE 20.37 (a) Emitter-follower output stage (Class A), and (b) voltage transfer diagram.

The positive output excursion is limited by the eventual saturation of Q_1. This means that V_o cannot exceed ($V_{CC} - V_{CEsat,1}$). $V_{Cesat,1}$ is less than $V_{BE,1}$. However, since V_i is most often provided by a previous gain stage, its voltage level can generally not be raised above the supply. In practice, this means that the maximum output voltage is limited to

$$V_{o,\,max} = V_{CC} - V_{BE,\,1} \approx V_{CC} - V_T \ln \frac{I_Q}{I_S} \tag{20.112}$$

Similarly, the negative excursion is limited by the saturation of Q_2. Therefore,

$$V_{o,\,min} = -V_{EE} + V_{CEsat,\,2} \tag{20.113}$$

If the assumption about the load resistor is not valid and R_L is small, the slope of the transfer characteristic is not exactly 1 and some curvature occurs for larger signal excursions. Also, negative output clipping can occur sooner. Indeed, the maximum current flow through R_L during the negative excursion is bounded by the bias I_Q. Consequently,

$$V_{o,\,min} = -I_Q R_L \tag{20.114}$$

Figure 20.37(b) graphically represents the emitter follower's transfer characteristic for both R_L assumptions.

If V_{CC} and V_{EE} are much larger than V_{BE} and V_{CEsat}, the maximum symmetrical swing one can obtain is

$$V_{o,\,peak} \approx \frac{V_{CC} + V_{EE}}{2} \tag{20.115}$$

provided V_i has the proper dc bias and the quiescent current is equal to or greater than the optimum value

$$I_{Q,\,opt} = I_{o,\,peak} = \frac{V_{o,\,peak}}{R_L} \tag{20.116}$$

Under the conditions of Eqs. (20.115) and (20.116), the emitter follower's power dissipation is

$$P_{supply} = (V_{CC} + V_{EE}) I_{Q,\,opt} \approx 2\, V_{o,\,peak} I_{o,\,peak} \tag{20.117}$$

For sinusoidal input conditions, the average power delivered to the load is expressed as

$$P_{load} = \frac{1}{2} V_{o,\,peak} I_{o,\,peak} \tag{20.118}$$

Therefore, the highest achievable power efficiency is limited to

$$\eta = \frac{P_{load}}{P_{supply}} \approx 25\% \tag{20.119}$$

In conclusion, while the emitter follower can be used as an output stage, it suffers from two major limitations. First, the output swing is asymmetrical and not rail-to-rail. Second, the operation is Class A and thus consumes dc power. The emitter follower's maximum power efficiency, which is only reached at full signal swing, is very poor.

Class B and Class AB Operation

Figure 20.38(a) shows a symmetrical configuration with the emitters of an npn and a pnp transistor tied together at the output node, while the input is provided to their joint bases. This dual emitter follower combination is frequently referred to as a *push-pull* arrangement. When no input is applied, clearly no current flow is possible and, thus, the operation is Class B. The push-pull configuration, however, does not solve all the problems of the single-ended emitter follower. Indeed, the output swing is still not rail-to-rail, but is limited to one V_{BE} drop from either supply rail (assuming V_i cannot exceed V_{CC} or V_{EE}). The circuit's transfer characteristic is shown in Fig. 20.38(b). Note that both transistors are off, not just for zero input as desired, but they also do not turn on when small inputs are applied. In effect, for $V_{BE,p} < V_i < V_{BE,n}$, the output has a dead zone. Such *hard non-linearity* leads to undesirable *cross-over distortion*. A method to overcome this problem will be discussed shortly. For larger inputs, only one of the transistors conducts.

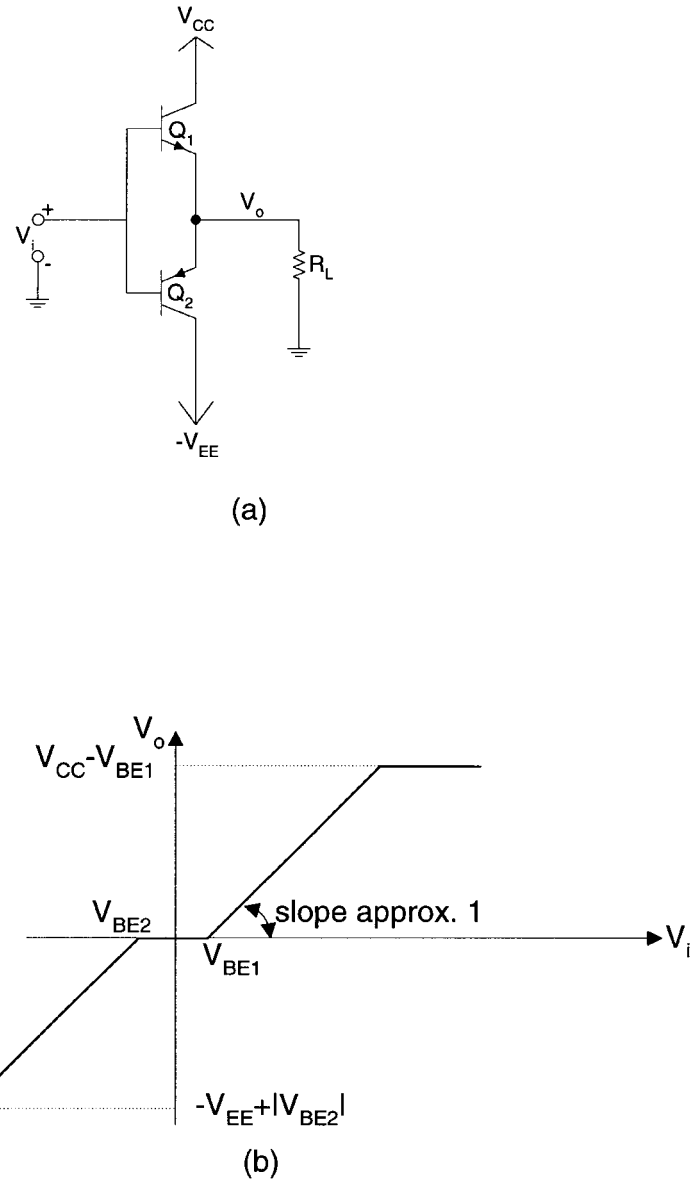


FIGURE 20.38 (a) Emitter-follower push-pull output stage (Class B), and (b) voltage transfer diagram.

The push-pull configuration's power efficiency under sinusoidal input conditions can be derived as follows. The dissipated power is equal to

$$P_{\text{supply}} = (V_{\text{CC}} + V_{\text{EE}})I_{\text{supply}} = (V_{\text{CC}} + V_{\text{EE}})\frac{V_{\text{o, peak}}}{\pi R_{\text{L}}} \tag{20.120}$$

while the power delivered to the load is expressed as

$$P_{\text{load}} = \frac{V_{\text{o, peak}}^2}{2R_{\text{L}}} \tag{20.121}$$

Thus,

$$\eta = \frac{P_{\text{load}}}{P_{\text{supply}}} = \frac{\pi}{2}\frac{V_{\text{o, peak}}}{V_{\text{CC}} + V_{\text{EE}}} \tag{20.122}$$

Equation (20.122) says that the efficiency is directly proportional to the peak output amplitude. In case the base–emitter voltage can be neglected relative to the supply voltages,

$$V_{\text{o, peak, max}} = \frac{V_{\text{CC}} + V_{\text{EE}} - V_{\text{BE, n}} - V_{\text{BE, p}}}{2} \approx \frac{V_{\text{CC}} + V_{\text{EE}}}{2} \tag{20.123}$$

The power efficiency's upper bound is therefore given by

$$\eta_{\text{max}} \approx \frac{\pi}{4} \approx 78.6\% \tag{20.124}$$

The dead zone and resulting cross-over distortion can be eliminated by inserting two conducting diodes between the bases of the npn and pnp output devices, as shown in Fig. 20.39(a). Strictly speaking, the push-pull circuit is then no longer Class A, but rather becomes Class AB as a small stand-by current constantly flows through both output devices. The resulting linear transfer characteristic is shown in Fig. 20.39(b). One should, however, note that the actual quiescent current is not well-defined, as it depends on matching between the base–emitter voltage drops of the diodes and the output transistors. For discrete implementations, this configuration would clearly be too sensitive. Even in integrated circuits, it is often desirable to stabilize the operating point through the inclusion of degeneration resistors, as illustrated in Fig. 20.40(a). In addition, these series resistors act as passive current limiters. Indeed, a potential problem occurs in the circuit of Fig. 20.39(a) when the output node is shorted to ground ($R_L = 0$). If the input voltage is large and positive, Q_2 and the diode string are off, forcing all the current I_B to flow into the base of Q_1, where it gets multiplied by the (high) current gain β_1. The resulting collector current may be so large as to cause Q_1 to self-destruct. Obviously, the voltage drop, which builds across either degeneration resistor, limits the maximum current and, as such, can prevent damage. Unfortunately, the inclusion of series resistors is a far from perfect solution as the values needed for quiescent current stabilization and overdrive protection are often unacceptably large. Thus, the degeneration resistors reduce the power efficiency, limit the output swing, and raise the circuit's output resistance. Fortunately, this issue can be circumvented if a diode is added in parallel with the degeneration resistor, as shown in Fig. 20.40(a). For low current values, the diode is off and the circuit is characterized by the high resistance of R_1 (R_2). At higher current levels, the diode turns on and provides a low dynamic resistance. For small input signals, the degeneration resistor is in series with the load and voltage division occurs. At higher input levels when the diode is on, the output again follows the input. The transfer characteristic, illustrated in Fig. 20.40(b), shows that rather than being completely eliminated as in the circuit of Fig. 20.39, the dead zone is replaced by a "slow zone." In other words, the hard non-linearity of the simple push-pull circuit in Fig. 20.38 is transformed into a more acceptable *soft non-linearity*. The transfer characteristics can be further linearized by applying negative feedback around the complete amplifier. The reader should observe, however, that the diode voltage drop further limits the maximum signal swing. While the diodes may reduce the passive current limiting provided by the resistors, superior active limiting is achieved when they are replaced by transistors, as depicted in Fig. 20.40(c). When the voltage drop across R_1 becomes high enough to forward-bias the base–emitter junction of Q_{D1}, the transistor starts to pull current away from the base of Q_1, delivering it harmlessly to the load, without additional multiplication. One should note that the limiting effect for large negative currents is not nearly as effective in this scheme, since this is largely determined by the current sinking capability of the driving transistor (not shown in Fig. 20.40), which pulls current out of the base of Q_2. On the other hand, pnp transistors are usually lateral or substrate devices with relatively low current gain, which furthermore rolls off very quickly at higher current

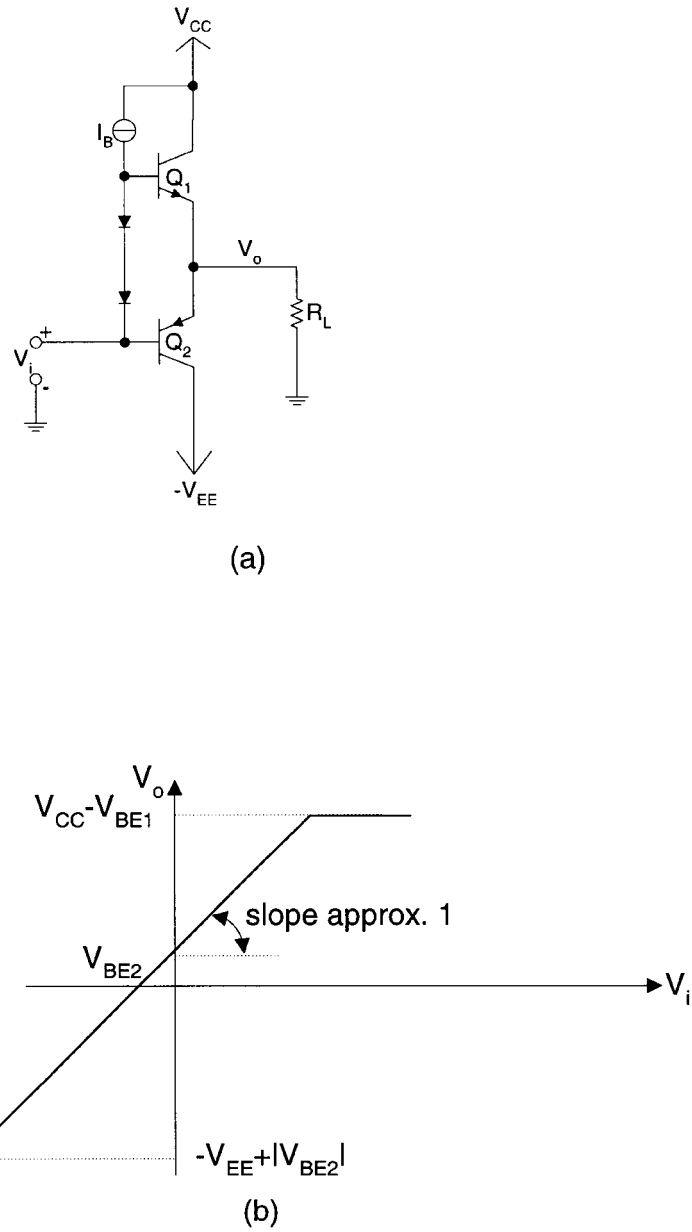


FIGURE 20.39 (a) Emitter-follower push-pull output stage (Class AB), and (b) voltage transfer diagram.

levels. Consequently, the potential problem of negative current overload is inherently less severe. Whereas these lateral or substrate pnp's are generally adequate for low to moderate power applications, if high power must be delivered, a complementary bipolar process with isolated vertical pnp's is required. When such a more complicated and expensive process is not available, alternatively quasi-complementary structures can be used. In Fig. 20.41, for example, the pnp transistor is replaced by a pseudo-Darlington pair (see Section 20.2).

In summary, the Class AB push-pull configuration meets the requirements of an output stage with relatively high efficiency in its power transfer and low stand-by dissipation. For very low voltage

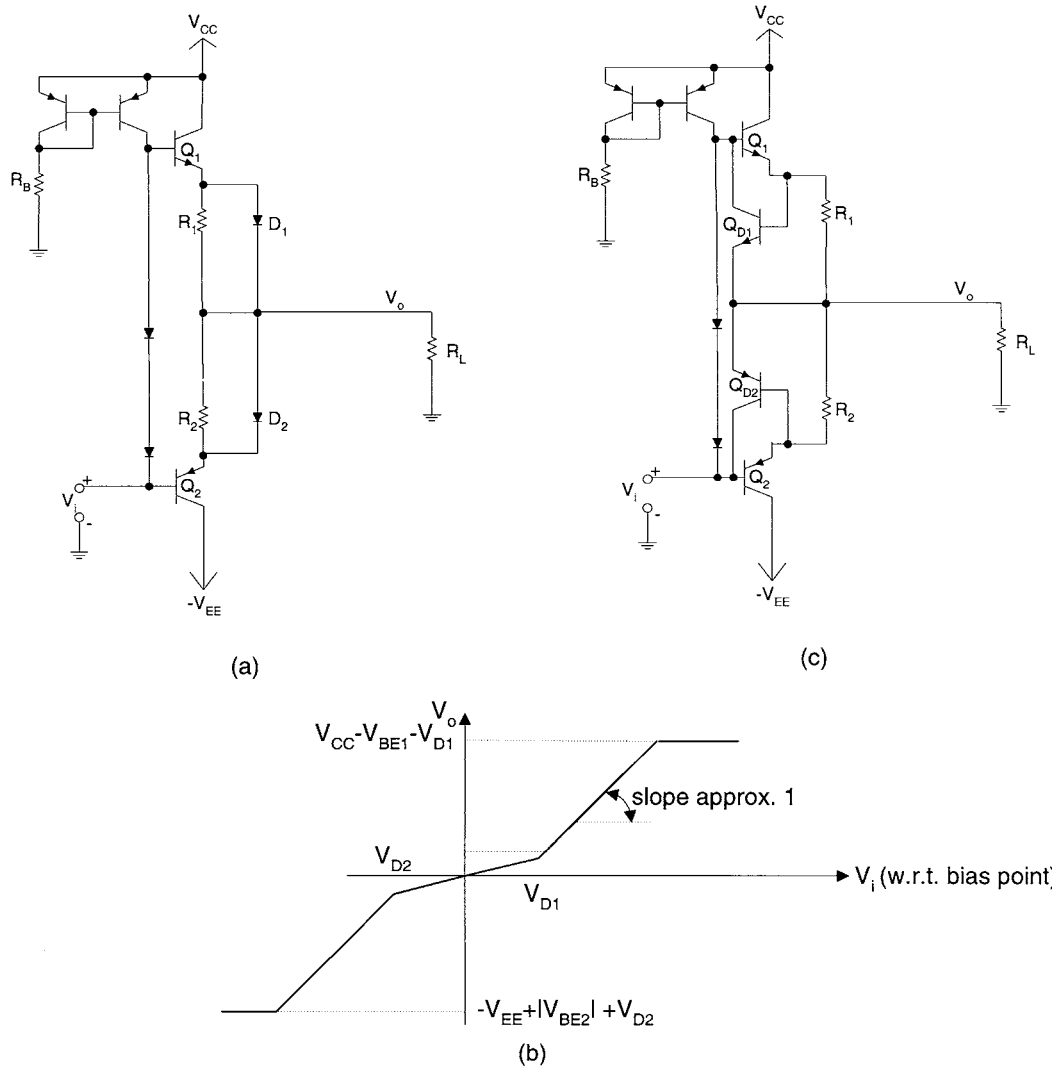


FIGURE 20.40 (a) Emitter-follower push-pull output stage (Class AB) with resistors and diodes; (b) voltage transfer diagram; and (c) emitter-follower push-pull output stage (Class AB) with resistors and transistors.

applications, however, the voltage drop across the base–emitter junction (and the series diodes) constitutes a serious limitation. A true rail-to-rail output swing (apart from an unavoidable, but low V_{CEsat}) can only be achieved by a complementary common-emitter output stage, as drawn in Fig. 20.42. Following the discussion in Section 20.2, the reader will likely interject that this arrangement suffers from a high output impedance and potential frequency limitations. An in-depth treatment of low-voltage common-emitter output configurations is beyond the scope of this chapter. The interested reader is referred to Ref. 5.

20.5 Bias Reference

It is definitely not the author's intention to present an in-depth discussion of voltage and current reference design. However, on several occasions, the terms "bandgap voltage" and "PTAT voltage" were mentioned.

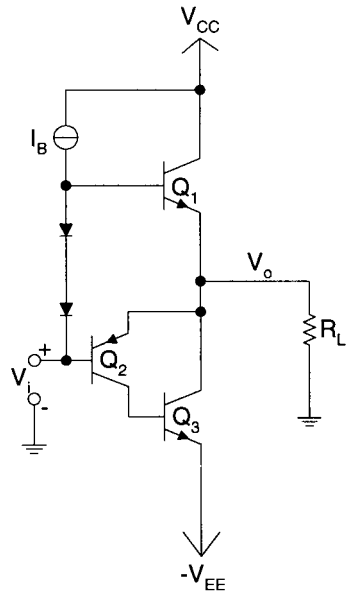


FIGURE 20.41 Quasi-complementary push-pull output stage.

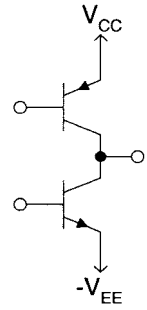


FIGURE 20.42 Rail-to-rail complementary common-emitter output stage.

It was also noted that a current, which is proportional to absolute temperature and inversely related to a resistor, is quite often desired in order to stabilize an amplifier's gain. A circuit that provides such supply-independent voltages and currents is shown in Fig. 20.43. By inspection,

$$V_{BG} = V_{BE1} + R_2 I = V_{BE1} + V_{PTAT} \tag{20.125}$$

The current mirror consisting of Q_3 and Q_4 forces the current I to split evenly between Q_1 and Q_2. As a result, the following identity is valid:

$$V_{BE1} = V_{BE2} + R_1 \frac{I}{2} \tag{20.126}$$

Substituting the Ebers-Moll identity into Eq. (20.126), where I_{S2} is N times I_{S1}, yields

$$I = \frac{2}{R_1}\left(V_T \ln\frac{I}{2I_{S1}} - V_T \ln\frac{I}{2I_{S2}}\right) = \frac{2}{R_1} V_T \ln N \tag{20.127}$$

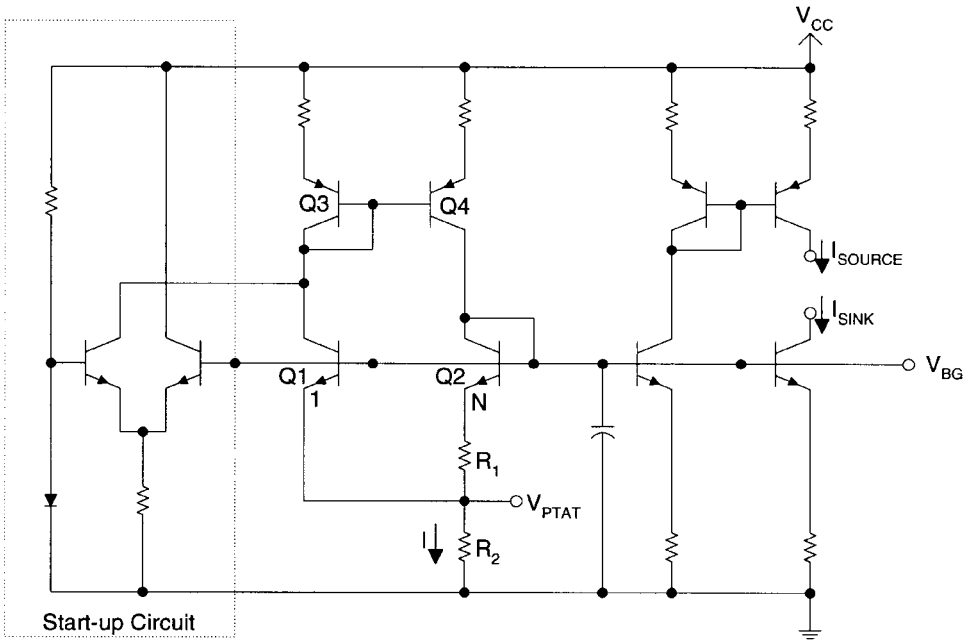


FIGURE 20.43 Bias reference.

By combining Eqs. (20.125) and (20.127), one gets

$$V_{BG} = V_{BE1} + 2\frac{R_2}{R_1}V_T \ln N \tag{20.128}$$

Also,

$$V_{PTAT} = 2\frac{R_2}{R_1}V_T \ln N = 2\frac{R_2}{R_1}\frac{kT}{q}\ln N \tag{20.129}$$

From Eq. (20.129), one concludes that V_{PTAT} is indeed proportional to absolute temperature. Apart from the absolute temperature T, V_{PTAT} only depends on physical constants (k and q), an area multiple N, and the ratio of two resistors. In addition to V_{PTAT}, the expression for V_{BG} (Eq. (20.128)) contains the term V_{BE1}, which decreases by 2 mV per degree increase in temperature. Through an appropriate selection of the resistors R_1 and R_2, the voltage V_{BG} can be made temperature independent. This occurs for $V_{BG} \approx$ 1.25 V, known as the BJT bandgap voltage. The currents I_{SOURCE} and I_{SINK} in Fig. 20.43 are simply mirrored copies of I, and thus exhibit the desired PTAT and resistor dependence.

Further observation of the circuit in Fig. 20.43 reveals that it possesses a second (although unstable) operating point. Indeed, the circuit is also in equilibrium when $V_{BG} = V_{PTAT} = 0$ V and there is no current flow. To prevent the bias reference from being stuck in this undesired state, an initial start-up circuit is added as shown. When power is first applied, the start-up circuit injects a small current into the mirror Q_3-Q_4, forcing the circuit to wake up and drift away from the zero state. As V_{BG} increases toward 1.25 V, the differential pair eventually switches and the start-up current is simply thrown away. The reader should realize that the circuit in Fig. 20.43 is conceptual in nature and specifically drawn to show that a supply voltage $V_{CC} < 2$ V suffices. However, it suffers from non-idealities due to base currents and poor supply rejection resulting from the simple current mirrors with relatively low output impedances. At the expense of added circuit complexity and the need for a higher supply voltage, significant improvements can be made. Such specialized bias reference discussion, however, goes beyond the scope of this chapter.

20.6 Operational Amplifiers

Introduction

Operational amplifiers (or op-amps) are key analog circuit building blocks that find widespread use in a variety of applications, such as precision amplification circuits (e.g., instrumentation amplifiers), continuous-time and switched-capacitor filters, etc. The traditional symbolic representation of the operational amplifier is shown in Fig. 20.44. The op-amp is a five-terminal device, with inverting and non-inverting input terminals (hence, accommodating a differential input signal), a single-ended output terminal, as well as positive and negative supply terminals. Most commercially available op-amps require a dual supply system of equal, but opposite value (e.g. ±15 V or ±5 V); however, asymmetrical or single-supply circuits are also available (e.g., +5 V and ground). Special-purpose operational amplifiers with differential or fully balanced outputs also exist. The internal circuitry of op-amps combines the different building blocks, which were previously discussed. Op-amps typically consist of two or three stages. The input stage, based on a differential pair, provides the initial amplification. A second or intermediate stage may be included to boost the amplifier's gain. Differential to single-ended conversion is also accomplished in the first stage, or, if applicable, in the second stage. The output stage, typically an emitter follower push-pull configuration, provides a low impedance, large swing, and high current drive. An elementary op-amp schematic can be found in Fig. 20.47(a).

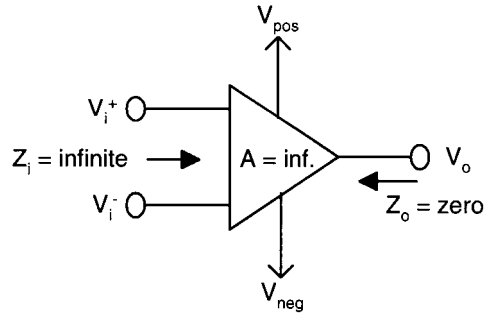


FIGURE 20.44 Op-amp symbol.

Ideal Op-Amps

The op-amp's input–output relationship can be expressed as

$$V_o = A(V_i^+ - V_i^-) \tag{20.130}$$

In the case of an ideal op-amp, A is assumed to have infinite magnitude as well as bandwidth. As such, there is no phase shift over frequency, as illustrated in Fig. 20.45(a). Since the output V_o must remain bounded, the assumption of infinite gain leads to the fundamental principle of *virtual ground* (virtual short). In other words, if the op-amp is ideal, there cannot exist a voltage difference between the input terminals, and V_i^+ must equal V_i^-. The assumption of ideality also calls for an infinite input impedance (i.e., the op-amp does not load the driving source) and a zero output impedance (i.e., the op-amp can accommodate arbitrarily small loads).

Op-Amp Non-idealities

Real operational amplifiers generally approximate the ideal op-amp model reasonably well; however, they naturally have finite gain, finite bandwidth, and finite input as well as output impedances. Specific non-idealities are itemized below.

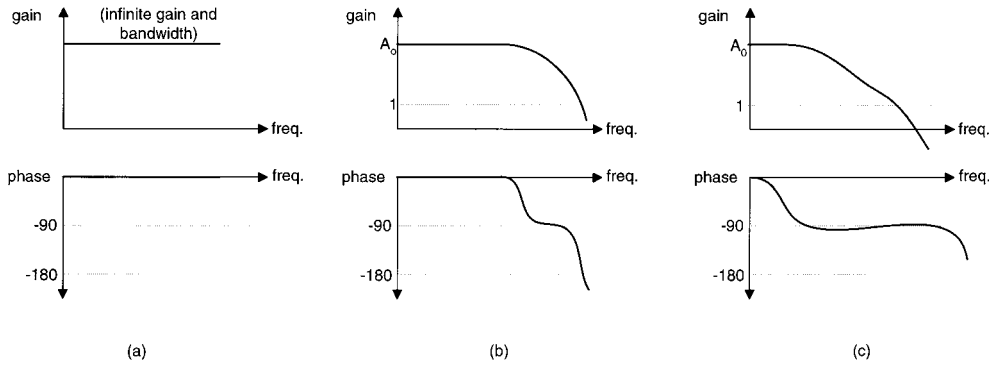


FIGURE 20.45 Magnitude and phase responses: (a) ideal op-amp; (b) real op-amp (potentially unstable in unity-gain feedback loop); (c) real op-amp with internal compensation (guaranteed stable, phase margin approx. 90 degrees).

Finite Gain

The op-amp's (low-frequency) gain can be made quite high, particularly when multiple gain stages are cascaded. The absolute gain value, however, is not very well-defined as it depends on widely varying process parameters, such as the transistor's current gain β. If a precise gain is required, the op-amp must be configured into a feedback network; for example, the non-inverting and inverting gain stages shown in Figs. 20.46(a) and (b), respectively. Assuming finite op-amp gain but infinite input impedance, the gain of the non-inverting amplifier in Fig. 20.46(a) can be expressed as

$$\frac{V_o}{V_i} = A\frac{R_i + R_f}{R_i + R_f + AR_i} \tag{20.131}$$

If A is high, Eq. (20.131) can be approximated by

$$\frac{V_o}{V_i} \approx \frac{A}{A+1}\frac{R_i + R_f}{R_i} \approx 1 + \frac{R_f}{R_i} \tag{20.132}$$

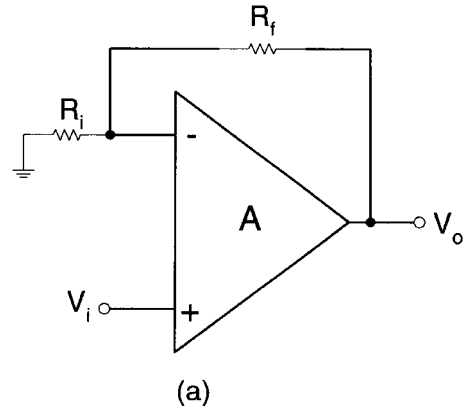
For practical resistor values, the *closed-loop* gain in Eq. (20.132) is not very high (at least compared to the op-amp's open-loop gain A), but it can be accurately set, even over process corners, by the ratio of two like components. Similarly, the inverting amplifier of Fig. 20.46(b) provides a closed-loop gain

$$\frac{V_o}{V_i} = -A\frac{R_f}{R_i + R_f + AR_i} \approx -\frac{A}{A+1}\frac{R_f}{R_i} \approx -\frac{R_f}{R_i} \tag{20.133}$$

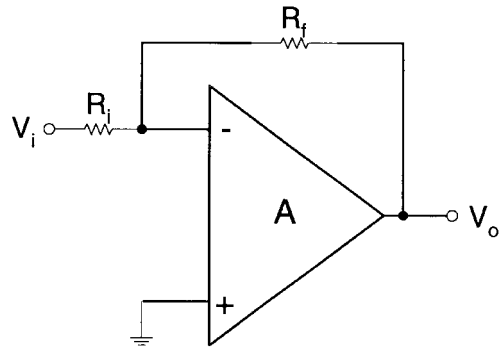
An important building block in active filter design is the inverting integrator circuit displayed in Fig. 20.46(c). By inspection,

$$\frac{V_o}{V_i} = \frac{-A}{(A+1)sR_iC_f + 1} \approx \frac{-1}{sR_iC_f} \tag{20.134}$$

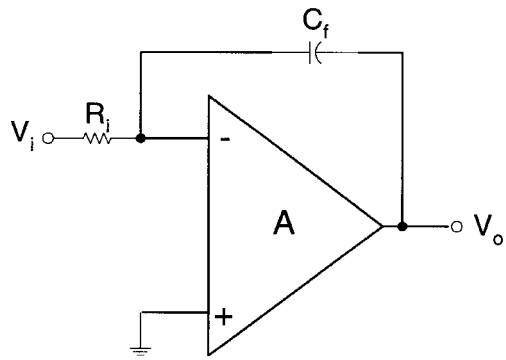
Unlike for the previous two amplifiers, the gain in Eq. (20.134) depends on the product of absolute resistor and capacitor values. When implemented as a monolithic circuit, the integrator's time constant is therefore subject to large tolerances, which need to be compensated for by trimming or an automatic tuning loop.



(a)



(b)



(c)

FIGURE 20.46 (a) Noninverting amplifier; (b) inverting amplifier; and (c) inverting integrator.

Finite Bandwidth

The op-amp's internal electronics are characterized by parasitic poles and/or zeros. As previously discussed, these unavoidable parasitics cause the gain to roll off at high frequencies and also introduce phase shifts. A typical op-amp's magnitude and phase responses are shown in Fig. 20.45(b). Assuming the op-amp has a dominant pole, the gain initially decreases by 20 dB/decade and the phase shift approaches –90°. At higher frequencies, the gain starts to decrease faster due to the combined effect of additional non-dominant parasitics and the phase increases as well. Since op-amps are used in gain stabilizing negative feedback circuits, as discussed in the previous sub-section, this high-frequency behavior can lead to instability. The worst possible situation occurs for unity feedback configurations. To avoid potential oscillation problems even under these conditions, commercial op-amps are nearly always internally compensated. Typically, a low-frequency dominant pole is introduced, which causes the gain to drop below unity at the –180° phase cross-over frequency, as shown in Fig. 20.45(c). The difference between the actual phase shift at the op-amp's unity-gain frequency and –180° is referred to as the *phase margin*. Whereas an op-amp is unconditionally stable when its phase margin is positive, values well above 45° are highly desirable to obtain quick settling to input transients. Conversely, the ratio between unity and the actual gain at the frequency corresponding to –180° phase shift is called the *gain margin*. If the dominant pole assumption is valid, the op-amp's gain A should be rewritten as

$$A(s) = A(j\omega) = \frac{A_o\omega_o}{s+\omega_o} \approx \frac{A_o\omega_o}{s} \tag{20.135}$$

where A_o is the dc gain and ω_o is the radial –3 dB frequency. $A_o\omega_o$ is referred to as the op-amp's *gain–bandwidth product*. When Eq. (20.135) is substituted into Eqs. (20.133) and (20.134), the latter gain expressions become frequency dependent. If Eq. (20.135) is likewise combined with the integrator gain in Eq. (20.134), the latter's denominator turns into a second-order polynomial.

Finite Input Impedance

An op-amp's input stage is usually a differential pair. Expressions for its differential and common-mode input resistances have been previously derived. If a high differential input resistance is desired, several design options exist. First, npn input pairs are better than pnp's, thanks to the higher β. Also, the input impedance increases as the input transconductance is lowered, either by reducing the bias current or by adding degeneration resistors. A Darlington pair can be used when an even higher input resistance is required. At high frequencies, the input impedance becomes capacitive (and thus decreases) due to the BJT's C_π and C_μ.

Input Bias Current

The bipolar transistors in the input differential pair require base current. For npn transistors, the current flows into the base, whereas current must be pulled out of the base in the case of pnp transistors. As a result of the higher β, for a given transconductance, the required input bias current is lower in absolute value in the case of an npn input stage compared to a pnp. Darlington or pseudo-Darlington input pairs can further reduce the input bias current requirement. Alternatively, input bias or base current cancellation techniques are sometimes applied.

Input Offset Voltage

Unavoidable device mismatches require the application of a small difference voltage between the input terminals in order to get zero volts at the output terminal. The reader is referred back to Section 20.3 (Input Offset Voltage).

Input Offset Current

Similar to the input offset voltage, when the inputs are currents rather than voltages, small component mismatches require the application of a small input difference current in order to get zero volts at the output node. See Section 20.3 (Input Offset Current) for more details.

Finite Output Impedance

A typical output stage consists of an emitter follower push-pull configuration. Hence, the output resistance depends on $1/g_m$, plus the resistance of the driving stage divided by β. Since β rolls off at high frequencies, R_o increases accordingly. As such, the output impedance appears inductive. This phenomenon can lead to stability problems when driving capacitive loads.

Finite Common-Mode Rejection Ratio

In Section 20.3, the common-mode rejection ratio was defined as the ratio between the differential and common-mode gains. However, an op-amp's CMRR can be more meaningfully explained in terms of the input offset voltage. In this way, the CMRR can be defined as the change in input offset voltage due to a unit change in common-mode input voltage. The CMRR of commercial op-amps typically measures 100 to 120 dB.

Finite Power Supply Rejection Ratio

Similar to the CMRR, the *power supply rejection ratio* (PSRR) is defined as the change in input offset voltage due to a unit change in the supply voltage. An op-amp's PSRR is nearly always different with respect to its positive and negative supplies. Hence, two individual performance numbers are specified. Typical values are in the range of the CMRR.

Slew Rate, Full-Power Bandwidth, and Unity-Gain Frequency

When a step input is applied, the op-amp's output cannot change instantaneously. Rather, a finite transition time is needed. The maximum rate of change in output voltage is referred to as the opamp's *slew rate* (SR). To determine an expression for the slew rate, consider the elementary op-amp in Fig. 20.47(a). This basic circuit consists of an input differential pair gain stage, which also converts the input to a single-ended signal. An intermediate stage provides additional gain. In addition, a dominant pole is introduced by means of the Miller capacitor C_c. The output stage is a Class AB emitter follower push-pull configuration. For the purpose of slew rate calculation, the elementary op-amp can be replaced by the equivalent circuit in Fig. 20.47(b). Then,

$$SR = \left.\frac{dV_o}{dt}\right|_{max} = \left.\frac{g_m V_i}{C_c}\right|_{max} \tag{20.136}$$

The maximum value of $g_m V_i$ is equal to α times the tail current source I_{EE}. Thus,

$$SR = \frac{\alpha I_{EE}}{C_c} \tag{20.137}$$

Based on the equivalent circuit in Fig. 20.47(b), the op-amp's radial unity-gain frequency is readily determined as

$$\omega_u = \frac{g_m}{C_c} \tag{20.138}$$

By combining Eqs. (20.137) and (20.138), the following relationship can be identified between the op-amp's slew rate and its unity-gain frequency

$$SR = \frac{\alpha I_{EE}}{g_m}\omega_u \tag{20.139}$$

From Eq. (20.137), one could conclude that to improve the slew rate, I_{EE} must be increased and/or C_c lowered. However, for the elementary opamp in Fig. 20.47(a), I_{EE} is also directly proportional to g_m.

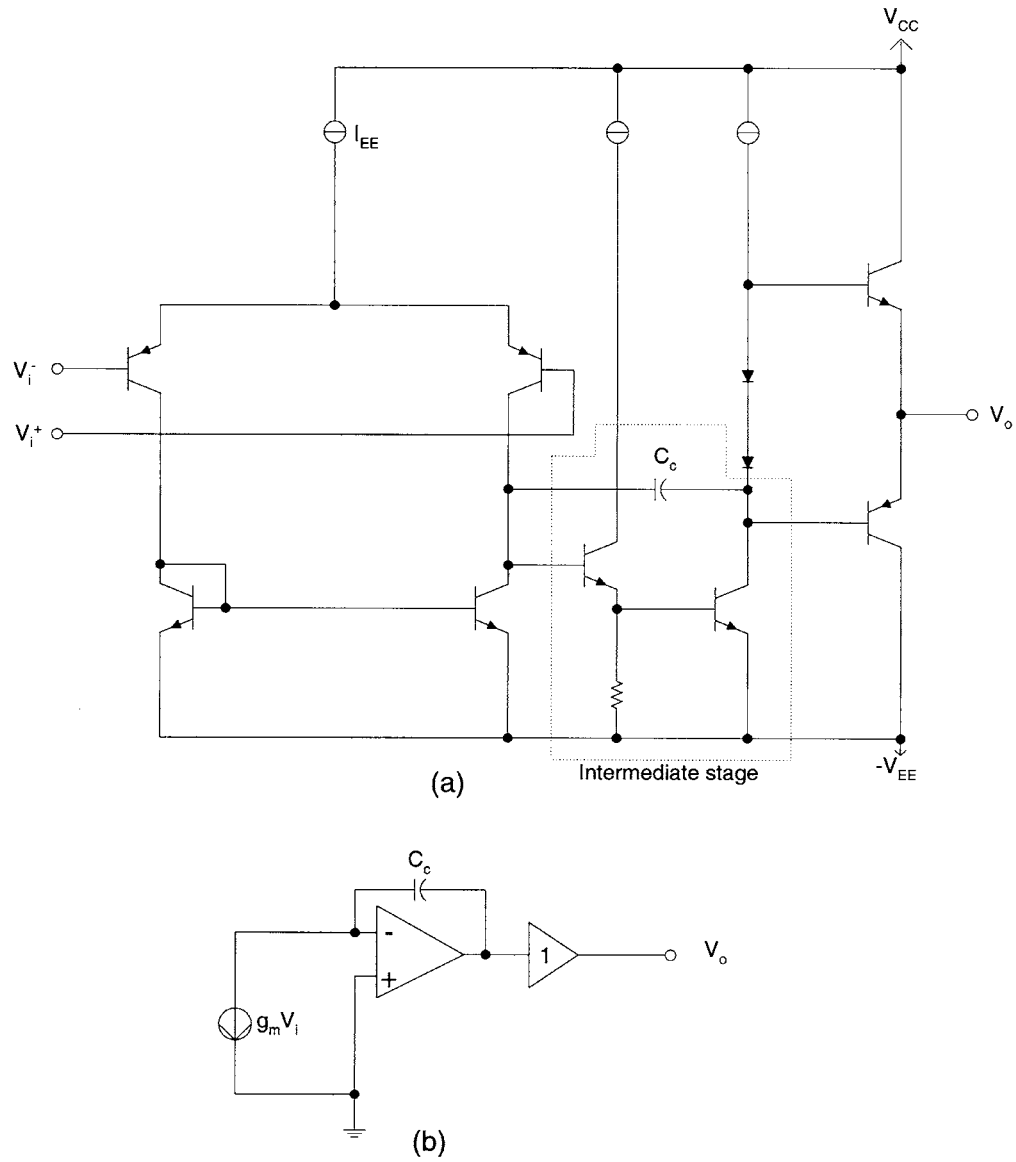


FIGURE 20.47 (a) Elementary op-amp, and (b) equivalent circuit for slew rate calculation.

Thus, both methods of increasing the slew rate at the same time increase the unity-gain bandwith ω_u. Obviously, a higher ω_u would also be desirable. However, ω_u has an upper limit, which is determined by the op-amp's non-dominant poles. Indeed, less than –180° phase shift must be maintained at ω_u in order to guarantee stability when external feedback is applied. This requirement for ω_u severely limits our flexibility to enhance the op-amp's slew rate. For a given ω_u, Eq. (20.139) suggests that the slew rate can be improved by increasing the I_{EE}/g_m ratio. Two approaches that fall into this category have been previously described in Section 20.3 on differential pair linearization. First, instead of a simple emitter-coupled pair, a differential pair with degeneration resistors can be used. Unfortunately, as pointed out before, the degeneration resistors negatively impact the circuit's noise and also degrade its offset performance. A better approach for improved slew rate is the use of parallel asymmetrical differential pairs,

such as two pairs with a 1:r emitter ratio. Although the emphasis in Section 20.3 was on linearization, and a ratio $r = 4$ was chosen specifically to eliminate the third harmonic distortion, the reader may recall that the tradeoff was a reduction in current efficiency (g_m/I_{EE}) to 64% of that of a simple differential pair. In case of an op-amp, a wide linear input range is of lesser or no concern (when feedback is applied, there is virtually no differential signal across the input terminals) and therefore different emitter ratios can be chosen. g_m/I_{EE} further decreases with larger r values. As such, the slew rate can be improved, while keeping ω_u constant. The reader is referred to Ref. 7 for further details.

Another parameter that can directly be correlated to the slew rate is the full-power bandwidth ω_{max}. The full-power bandwidth is defined as the maximum radian frequency for which the op-amp achieves a full output swing under the assumption of a sinusoidal signal. Let

$$V_o = \hat{V}\sin\omega t \tag{20.140}$$

Then,

$$\frac{dV_o}{dt} = \hat{V}\omega\cos\omega t \tag{20.141}$$

and

$$\left.\frac{dV_o}{dt}\right|_{max} = \hat{V}\omega_{max} \tag{20.142}$$

The left-hand side in Eq. (20.142) is, per definition, the slew rate. Hence, the slew rate and full-power bandwidth are simply related by the amplitude of the sinusoidal output signal.

20.7 Conclusion

In this chapter, the basic concepts behind signal amplification using bipolar transistors were introduced. Different circuit configurations, which fulfill distinct roles as building blocks in multi-stage designs or operational amplifiers, were presented. During their analysis, no parallel was drawn nor was a comparison made with respect to competing CMOS designs. This chapter would, however, not be complete if the main pros and cons of bipolar amplifiers are not very briefly mentioned. Bipolar amplifiers typically enjoy an advantage by offering higher gain, wider bandwidth, lower noise, and smaller offsets compared to their CMOS counterparts. The transconductance of an MOS transistor, while proportional to the device's size, is generally much lower than for bipolar and, rather than linearly, only increases with the square root of the bias current.

One disadvantage of bipolar amplifiers is the need for input bias currents, coupled with unavoidable input offset currents. Second, MOS transistors in saturation approximate a square-law I–V relationship and are therefore inherently more linear than bipolar devices, which are exponential in nature. Additionally, the linearity of MOS designs can be improved simply by proper device sizing. But, perhaps the most significant drawback of bipolar technology is a more complicated and expensive process (especially in the case of a complementary process with true isolated pnp transistors). Furthermore, bipolar's incompatibility with mainstream VLSI processes prevents higher levels of system integration.

References

Text Books

1. R. D. Middlebrook, *Differential Amplifiers*, Wiley, New York, 1963.
2. L. J. Giacoletto, *Differential Amplifiers*, Wiley, New York, 1970.

3. P. R. Gray and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, 2nd ed., Wiley, New York, 1984.
4. A. B. Grebene, *Bipolar and MOS Analog Integrated Circuit Design*, Wiley, New York, 1984.
5. J. Fonderie, *Design of Low-Voltage Bipolar Operational Amplifiers*, Delft University Press, Delft, 1991.

Articles

6. J. E. Solomon, The monolithic opamp: a tutorial study, *IEEE J. Solid-State Circuits*, vol. SC-9, pp. 314-332, Dec. 1974.
7. J. Schmoock, An input stage transconductance reduction technique for high-slew rate operational amplifiers, *IEEE J. Solid-State Circuits,* vol. SC-10, pp. 407-411, Dec. 1975.
8. J. O. Voorman, W. H. A. Bruls, and P. J. Barth, Bipolar integration of analog gyrator and laguerre type filters, *Proc. ECCTD*, 1983, Stuttgart, pp. 108-110.
9. J. H. Huijsing and D. Linebarger, Low-voltage operational amplifier with rail-to-rail input and output ranges, *IEEE J. Solid-State Circuits*, vol. SC-20, pp. 1144-1150, Dec. 1985.
10. R. J. Widlar and M. Yamatake, A fast settling opamp with low supply currents, *IEEE J. Solid-State Circuits,* vol. SC-24, pp. 796-802, June 1989.
11. G. A. De Veirman, S. Ueda, J. Cheng, S. Tam, K. Fukahori, M. Kurisu, and E. Shinozaki, A 3.0 V 40 Mbit/s hard disk drive read channel IC, *IEEE J. Solid-State Circuits*, vol. SC-30, pp. 788-199, July 1995.
12. G. A. De Veirman, Differential circuits, in *Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster, Editor, Wiley, New York, 1999.

Toumazou, C., Payne, A. "High-Frequency Amplifiers"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

21

High-Frequency Amplifiers

Chris Toumazou
Alison Payne
Imperial College, University of London

21.1 Introduction

As the operating frequency of communication channels for both video and wireless increases, there is an ever-increasing demand for high-frequency amplifiers. Furthermore, the quest for single-chip integration has led to a whole new generation of amplifiers predominantly geared toward CMOS VLSI. In this chapter, we will focus on the design of high-frequency amplifiers for potential applications in the front-end of video, optical, and RF systems. Figure 21.1 shows, for example, the architecture of a typical mobile phone transceiver front-end. With channel frequencies approaching the 2-GHz range, coupled with demands for reduced chip size and power consumption, there is an increasing quest for VLSI at microwave frequencies. The shrinking feature size of CMOS has facilitated the design of complex analog circuits and systems in the 1- to 2-GHz range, where more traditional low-frequency lumped circuit techniques are now becoming feasible. Since the amplifier is the core component in such systems, there has been an abundance of circuit design methodologies for high-speed, low-voltage, low-noise, and low distortion operation.

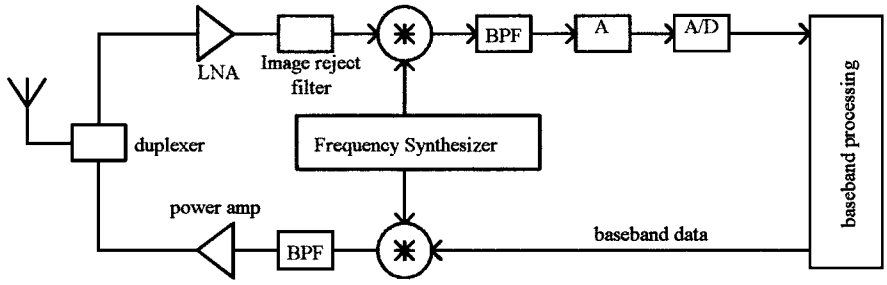


FIGURE 21.1 Generic wireless transceiver architecture.

This chapter will present various amplifier designs that aim to satisfy these demanding requirements. In particular, we will review, and in some cases present new ideas for power amps, LNAs, and transconductance cells, which form core building blocks for systems such as Fig. 21.1. Section 21.2 begins by reviewing the concept of current-feedback, and shows how this concept can be employed in the development of low-voltage, high-speed, constant-bandwidth CMOS amplifiers. The next two sections of the chapter focus on amplifiers for wireless receiver applications, investigating performance requirements and design strategies for optical receiver amplifiers (Section 21.3) and high-frequency low-noise amplifiers (Section 21.4). Section 21.5 considers the design of amplifiers for the transmitter side, and in particular the design and feasibility of Class E power amps are discussed. Finally, Section 21.6 reviews a very recent low-distortion amplifier design strategy termed "log-domain," which has shown enormous potential for high-frequency, low-distortion tunable filters.

21.2 The Current Feedback Op-Amp

Current Feedback Op-Amp Basics

The operational amplifier (op-amp) is one of the fundamental building blocks of analog circuit design.[1,2] High-performance signal processing functions such as amplifiers, filters, oscillators, etc. can be readily implemented with the availability of high-speed, low-distortion op-amps. In the last decade, the development of complementary bipolar technology has enabled the implementation of single-chip video op-amps.[3–7] The emergence of op-amps with non-traditional topologies, such as the current feedback op-amp, has improved the speed of these devices even further.[8–11] Current feedback op-amp structures are well known for their ability to overcome (to a first-order approximation) the gain-bandwidth tradeoff and slew rate limitation that characterizes traditional voltage feedback op-amps.[12]

Figure 21.2 shows a simple macromodel of a current feedback op-amp (CFOA), along with a simplified circuit diagram of the basic architecture. The topology of the current feedback op-amp differs from the conventional voltage feedback op-amp (VOA) in two respects. First, the input stage of a CFOA is a unity-gain voltage buffer connected between the inputs of the op-amp. Its function is to force V_n to follow V_p, very much like a conventional VOA does via negative feedback. In the case of the CFOA, because of the low output impedance of the buffer, current can flow in or out of the inverting input, although in normal operation (with negative feedback) this current is extremely small. Secondly, a CFOA provides a high open-loop transimpedance gain $Z(j\omega)$, rather than open-loop voltage gain as with a VOA. This is shown in Fig. 21.2, where a current-controlled current source senses the current I_{INV} delivered by the buffer to the external feedback network, and copies this current to a high impedance $Z(j\omega)$. The voltage conveyed to the output is given by Eq. 21.1:

$$V_{OUT} = Z(j\omega) \cdot I_{INV} \Rightarrow \frac{V_{OUT}}{I_{INV}}(j\omega) = Z(j\omega) \tag{21.1}$$

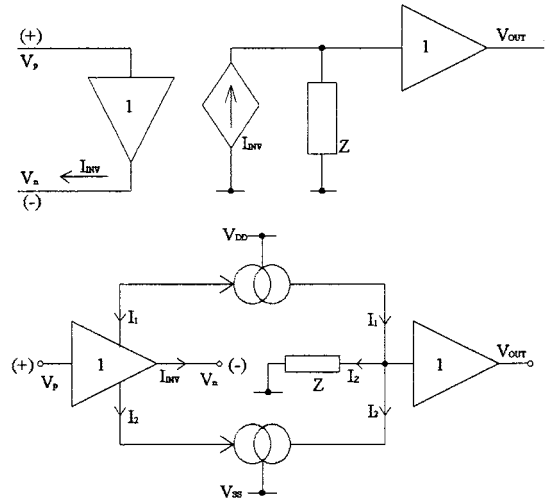


FIGURE 21.2 Current feedback op-amp macromodel.

When the negative feedback loop is closed, any voltage imbalance between the two inputs due to some external agent, will cause the input voltage buffer to deliver an error current I_{INV} to the external network. This error current $I_{INV} = I_1 - I_2 = I_Z$ is then conveyed by the current mirrors to the impedance Z, resulting in an ouput voltage as given by Eq. 21.1. The application of negative feedback ensures that V_{OUT} will move in the direction that reduces the error current I_{INV} and equalizes the input voltages.

We can approximate the open-loop dynamics of the current feedback op-amp as a single pole response. Assuming that the total impedance $Z(j\omega)$ at the gain node is the combination of the output resistance of the current mirrors R_o in parallel with a compensation capacitor C, we can write:

$$Z(j\omega) = \frac{R_o}{1 + j\omega R_o C} = \frac{R_o}{1 + j\dfrac{\omega}{\omega_o}} \tag{21.2}$$

where $\omega_o = 1/R_o \cdot C$ represents the frequency where the open-loop transimpedance gain is 3 dB down from its low frequency value R_o. In general, R_o is designed to be very high in value.

Referring to the non-inverting amplifier configuration shown in Fig. 21.3:

$$I_{INV} = \frac{V_{IN}}{R_G} - \frac{V_{OUT} - V_{IN}}{R_F} = \frac{V_{IN}}{R_G // R_F} - \frac{V_{OUT}}{R_F} \tag{21.3}$$

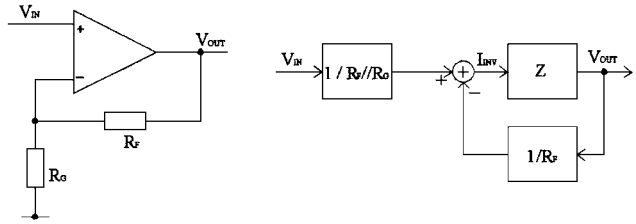


FIGURE 21.3 CFOA non-inverting amplifier configuration.

Substituting Eq. 21.1 into Eq. 21.3 yields the following expression for the closed-loop gain:

$$A_{CL}(j\omega) = \left(1 + \frac{R_F}{R_G}\right) \cdot \frac{Z(j\omega)}{R_F + Z(j\omega)} = \left(1 + \frac{R_F}{R_G}\right) \cdot \frac{1}{1 + \frac{R_F}{Z(j\omega)}} \tag{21.4}$$

Combining Eqs. 21.2 and 21.4, and assuming that the low frequency value of the open-loop transimpedance is much higher than the feedback resistor ($R_o >> R_F$) gives:

$$A_{CL}(j\omega) = \left(1 + \frac{R_F}{R_G}\right) \cdot \frac{1}{1 + j\frac{R_F \cdot \omega}{R_o \cdot \omega_o}} = \frac{A_{Vo}}{1 + j\frac{\omega}{\omega_\alpha}} \tag{21.5}$$

Referring to Eq. 21.5, the closed-loop gain $A_{Vo} = 1 + R_F/R_G$, while the closed-loop –3 dB frequency ω_α is given by:

$$\omega_\alpha = \frac{R_o}{R_F} \cdot \omega_o \tag{21.6}$$

Eq. 21.6 indicates that the closed-loop bandwidth does not depend on the closed-loop gain as in the case of a conventional VOA, but is determined by the feedback resistor R_F. Explaining this intuitively, the current available to charge the compensation capacitor at the gain node is determined by the value of the feedback resistor R_F and not R_o, provided that$R_o >> R_F$. So, once the bandwidth of the amplifier is set via R_F, the gain can be independently varied by changing R_G. The ability to control the gain independently of bandwidth constitutes a major advantage of current feedback op-amps over conventional voltage feedback op-amps.

The other major advantage of the CFOA compared to the VFOA is the inherent absence of slew rate limiting. For the circuit of Fig. 21.3, assume that the input buffer is very fast and thus a change in voltage at the non-inverting input is instantaneously converted to the inverting input. When a step ΔV_{IN} is applied to the non-inverting input, the buffer output current can be derived as:

$$I_{INV} = \frac{V_{IN} - V_{OUT}}{R_F} + \frac{V_{IN}}{R_G} \tag{21.7}$$

Eq. 21.7 indicates that the current available to charge/discharge the compensation capacitor is proportional to the input step regardless of its size, that is, there is no upper limit. The rate of change of the output voltage is thus:

$$\frac{dV_{OUT}}{dt} = \frac{I_{INV}}{C} \Rightarrow V_{OUT}(t) = \Delta V_{IN} \cdot \left(1 + \frac{R_F}{R_G}\right) \cdot (1 - e^{-t/R_f \cdot C}) \tag{21.8}$$

Eq. 21.8 indicates an exponential output transition with time constant $\tau = R_F \cdot C$. Similar to the small-signal frequency response, the large-signal transient response is governed by R_F alone, regardless of the magnitude of the closed-loop gain. The absence of slew rate limiting allows for faster settling times and eliminates slew rate-related non-linearities.

In most practical bipolar realizations, Darlington-pair transistors are used in the input stage to reduce input bias currents, which makes the op-amp somewhat noisier and increases the input offset voltage. This is not necessary in CMOS realizations due to the inherently high MOSFET input impedance. However, in a closed-loop CFOA, R_G should be much larger than the output impedance of the buffer. In bipolar realizations, it is fairly simple to obtain a buffer with low output resistance, but this becomes more of a

problem in CMOS due to the inherently lower gain of MOSFET devices. As a result, R_G typically needs to be higher in a CMOS CFOA than in a bipolar realization, and consequently, R_E needs to be increased above the value required for optimum high-frequency performance. Additionally, the fact that the input buffer is not in the feedback loop imposes linearity limitations on the structure, especially if the impedance at the gain node is not very high. Regardless of these problems, current feedback op-amps exhibit excellent high-frequency characteristics and are increasingly popular in video and communications applications.[13]

The following sections outline the development of a novel low-output impedance CMOS buffer, which is then employed in a CMOS CFOA to reduce the minimum allowable value of R_G.

CMOS Compound Device

A simple PMOS source follower is shown in Fig. 21.4. The output impedance seen looking into the source of M1 is approximately $Z_{out} = 1/g_m$, where g_m is the small signal transconductance of M1. To increase g_m, the drain current of M1 could be increased, which leads to an increased power dissipation. Alternatively, the dimensions of M1 can be increased, resulting in additional parasitic capacitance and hence an inferior frequency response. Figure 21.5 shows a configuration that achieves a higher transconductance than the simple follower of Fig. 21.3 for the same bias current.[11] The current of M2 is fed back to M1 through the a:1 current mirror. This configuration can be viewed as a compound transistor whose gate is the gate of M1 and whose source is the source of M2. The impedance looking into the compound source can be approximated as $Z_{out} = (g_{m1} - a \cdot g_{m2})/(g_{m1} \cdot g_{m2})$, where g_{m1} and g_{m2} represent the small signal transconductance of M1 and M2, respectively. The output impedance can be made small by setting the current mirror transfer ratio $a = g_{m1}/g_{m2}$.

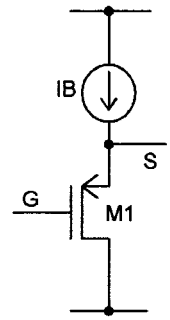


FIGURE 21.4 Simple PMOS source follower.

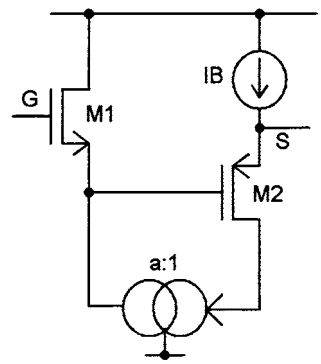


FIGURE 21.5 Compound MOS device.

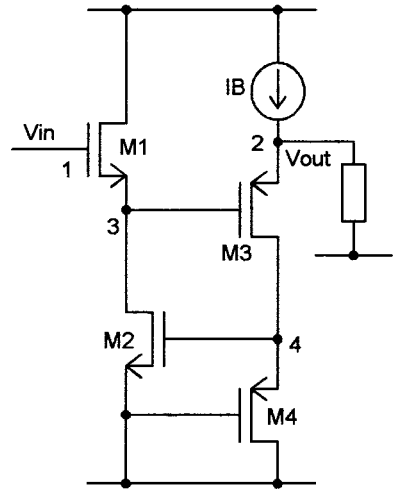


FIGURE 21.6 Actual p-compound device implementation.

The p-compound device is practically implemented as in Fig. 21.6. In order to obtain a linear voltage transfer function from node 1 to 2, the gate-source voltages of M1 and M3 must cancel. The current mirror (M4-M2) acts as an NMOS-PMOS gate-source voltage matching circuit[14] and compensates for the difference in the gate-source voltages of M1 and M3, which would normally appear as an output offset. DC analysis, assuming a square law model for the MOSFETs, shows that the output voltage exactly follows the input voltage. However, in practice, channel length modulation and body effects preclude exact cancellation.[15]

Buffer and CFOA Implementation

The current feedback op-amp shown in Fig. 21.7 has been implemented in a single-well 0.6-μm digital CMOS process[11]; the corresponding layout plot is shown in Fig. 21.8. The chip has an area of 280 μm by

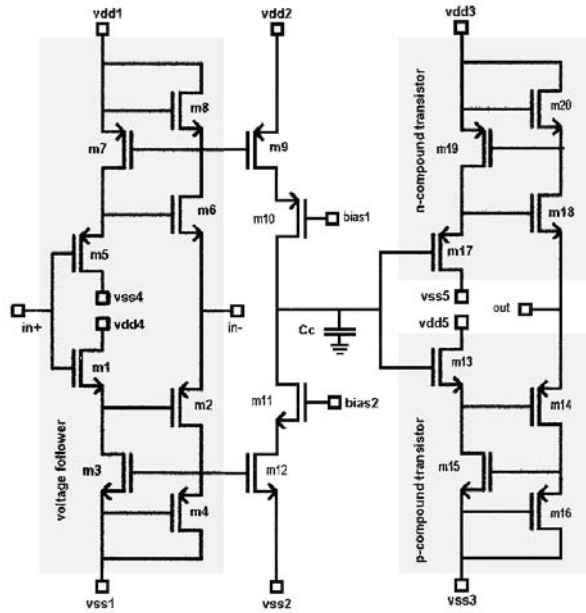


FIGURE 21.7 Current feedback op-amp schematic.

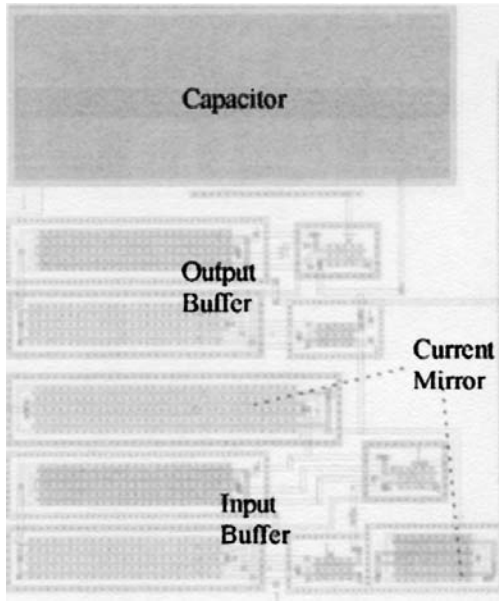


FIGURE 21.8 Current feedback op-amp layout plot.

330 μm and a power dissipation of 12 mW. The amplifier comprises two voltage followers (input and output) connected by cascoded current mirrors to enhance the gain node impedance. A compensation capacitor (Cc = 0.5 pF) at the gain node ensures adequate phase margin and thus closed-loop stability. The voltage followers have been implemented with two compound transistors, p-type and n-type, in a push-pull arrangement. Two such compound transistors in the output stage are shown shaded in Fig. 21.7. The input voltage follower of the current feedback op-amp was initialy tested open-loop, and measured results are summarized in Table 21.1. The load is set to 10 kΩ/10 pF, except where mentioned otherwise, 10 kΩ being a limit imposed by overall power dissipation of the chip. Intermodulation distortion was measured with two tones separated by 200 kHz. The measured output impedance of the buffer is given in Fig. 21.9. It remains below 80 Ω up to a frequency of about 60 MHz, when it enters an inductive region. A maximum impedance of 140 Ω is reached around 160 MHz. Beyond this frequency, the output impedance is dominated by parasitic capacitances. The inductive behavior is characteristic of the use of feedback to reduce output impedance, and can cause stability problems when driving capacitive loads. Small-signal analysis (summarized in Table 21.2) predicts a double zero in the output impedance.[15]

TABLE 21.1 Voltage Buffer Performance

Power Supply	5 V	Dissipation	5 mW
DC gain (no load)	–3.3dB	Bandwidth	140 MHz
Output impedance	75Ω	Min. load resistance	10 KΩ
HD2 (V_{in} = 200 mV_{rms})	1 MHz	–50 dB	
	10 MHz	–49 dB	
	20 MHz	–45 dB	
IM3 (V_{in}=200 mV_{rms})	20 MHz, Δf = 200 KHz	–53 dB	
Slew rate	(Load = 10 pF)	+ 130 V/μs	-72 V/μs
Input referred noise	10 $nV\sqrt{Hz}$		

Note: Load = 10 kΩ/10 pF, except for slew rate measurement.

Making factor G in Table 21.2 small will reduce the output impedance, but also moves the double zero to lower frequencies and intensifies the inductive behavior. The principal tradeoff in this configuration is between output impedance magnitude and inductive behavior. In practice. the output impedance

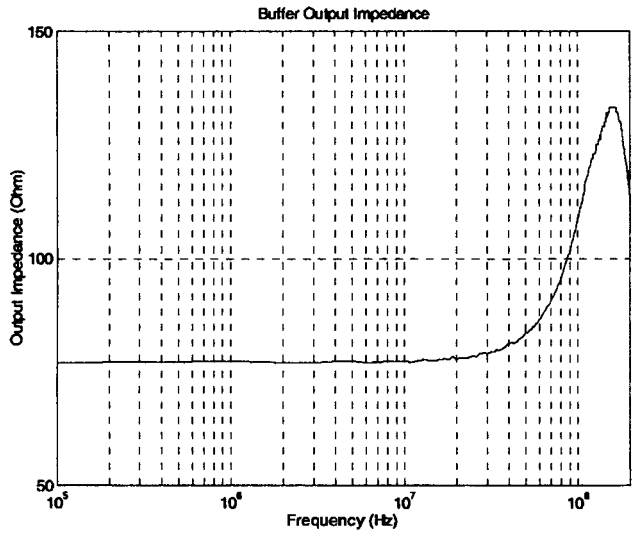


FIGURE 21.9 Measured buffer output impedance characteristics.

TABLE 21.2 Voltage Transfer Function and Output Impedance of Compound Device

$$Z_{out} = \frac{G}{(g_{m1} + g_{ds1} + g_{ds2}) \cdot (g_{m3} + g_{ds3}) \cdot (g_{m4} + g_{ds4})}$$

$$\frac{V_{out}}{V_{in}} = \frac{g_{m1} \cdot g_{m3} \cdot (g_{m4} + g_{ds4})}{(g_{m1} + g_{ds1} + g_{ds2}) \cdot (g_{m3} + g_{ds3}) \cdot (g_{m4} + g_{ds4}) + g_L \cdot G}$$

$$G = (g_{m1} + g_{ds1} + g_{ds2}) \cdot (g_{m4} + g_{ds4} + g_{ds3}) - g_{m2} \cdot g_{m3}$$

can be reduced by a factor of 3 while still maintaining good stability when driving capacitive loads. Figure 21.10 shows the measured frequency response of the buffer. Given the low power dissipation, excellent slew rates have been achieved (Table 21.2).

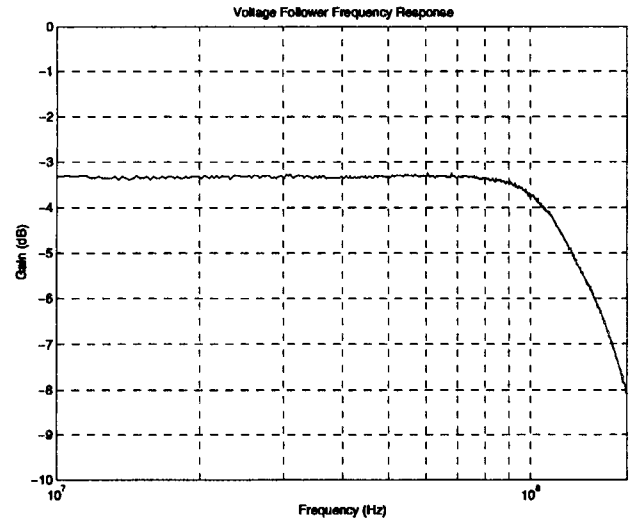


FIGURE 21.10 Measured buffer frequency response.

After the characterization of the input buffer stage, the entire CFOA was tested to confirm the suitability of the compound transistors for the implementation of more complex building blocks. Open-loop transimpedance measurements are shown in Fig. 21.11. The bandwidth of the amplifier was measured at gain settings of 1, 2, 5, and 10 in a non-inverting configuration, and the feedback resistor was trimmed to achieve maximum bandwidth at each gain setting separately. CFOA measurements are summarized in Table 21.3, loading conditions are again 10 kΩ/10 pF.

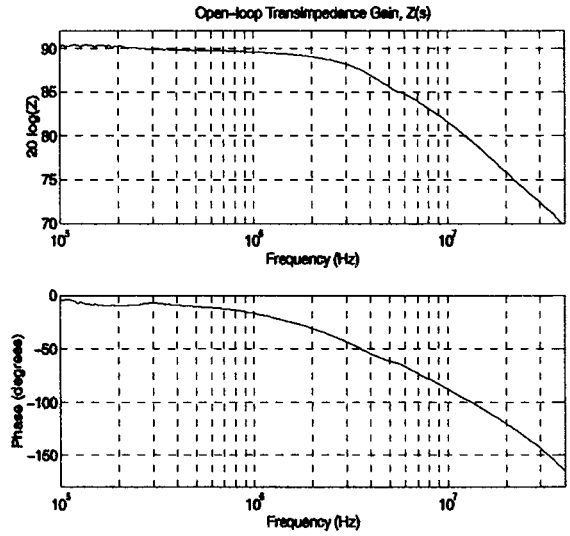


FIGURE 21.11 Measured CFOA open-loop transimpedance gain.

TABLE 21.3 Current Feedback Op-Amp Measurement Summary

Power Supply	5 V	Power Dissipation	12 mW
Gain	Bandwidth (MHz)		
1	117		
2	118		
5	113		
10	42		
Frequency	Input (mV rms)	Gain	HD2 (dB)
1 MHz	140	2	–51
	40	5	–50
	10	10	–49
10 MHz	80	2	–42
	40	5	–42
	13	10	–43

Fig. 21.12 shows the measured frequency response for various gain settings. The bandwidth remains constant at 110 MHz for gains of 1, 2, and 5, consistent with the expected behavior of a CFOA. The bandwidth falls to 42 MHz for a gain of 10 due to the finite output impedance of the input buffer stage which series as the CFOA inverting input. Figure 21.13 illustrates the step response of the CFOA driving a 10 kΩ/10 pF load at a voltage gain of 2. It can be seen that the inductive behavior of the buffers has little effect on the step response. Finally, distortion measurements were carried out for the entire CFOA for gain settings 2, 5, and 10 and are summarized in Table 21.3. HD2 levels can be further improved by employing a double-balanced topology. A distortion spectrum is shown in Fig. 21.14; the onset of HD3 is due to clipping at the test conditions.

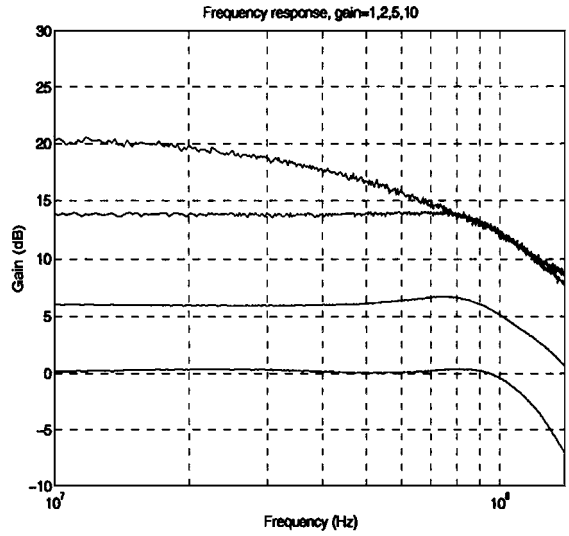


FIGURE 21.12 Measured CFOA closed-loop frequency response.

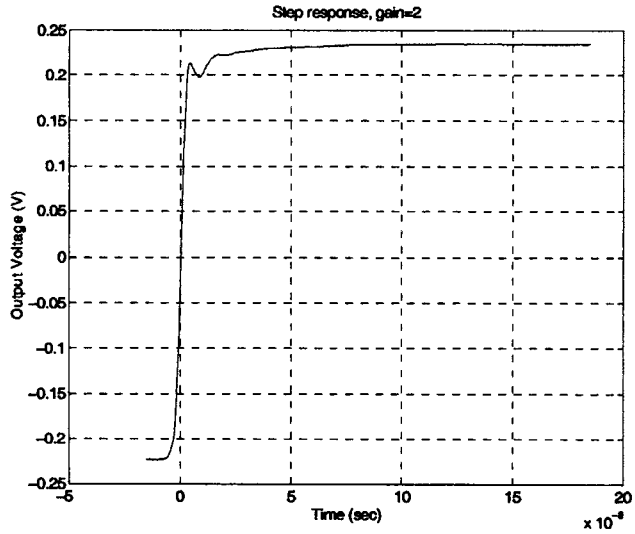


FIGURE 21.13 Measured CFOA step response.

21.3 RF Low-Noise Amplifiers

This section reviews the important performance criteria demanded of the front-end amplifier in a wireless communication receiver. The design of CMOS LNAs for front-end wireless communication receiver applications is then addressed. Section 21.4 considers the related topic of low-noise amplifiers for optical receiver front-ends.

Specifications

The front-end amplifier in a wireless receiver must satisfy demanding requirements in terms of noise, gain, impedance matching, and linearity.

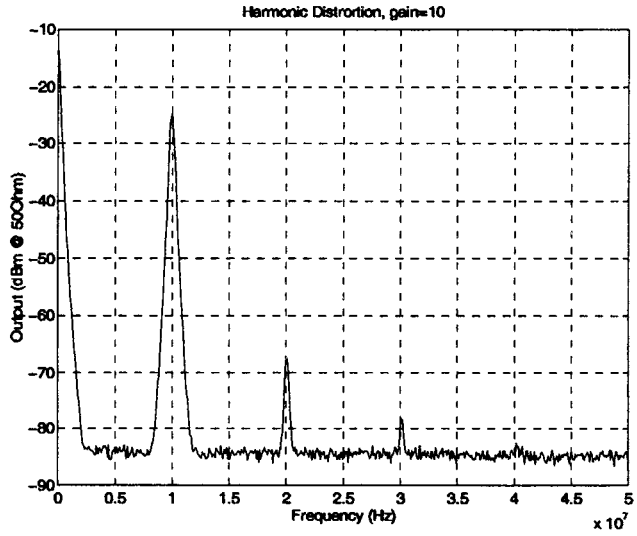


FIGURE 21.14 CFOA harmonic distortion measurements.

Noise

Since the incoming signal is usually weak, the front-end circuits of the receiver must possess very low noise characteristics so that the original signal can be recovered. Provided that the gain of the front-end amplifier is sufficient so as to suppress noise from the subsequent stages, the receiver noise performance is determined predominantly by the front-end amplifier. Hence, the front-end amplifier should be a low-noise amplifier (LNA).

Gain

The voltage gain of the LNA must be high enough to ensure that noise contributions from the following stages can be safely neglected. As an example, Fig. 21.15 shows the first three stages in a generic front-end receiver, where the gain and output-referred noise of each stage are represented by G_i and N_i ($i = 1, 2, 3$), respectively. The total noise at the third stage output is given by:

$$N_{out} = N_{in}G_1G_2G_3 + N_1G_2G_3 + N_2G_3 + N_3 \tag{21.9}$$

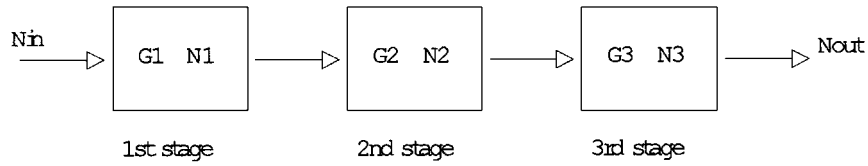


FIGURE 21.15 Three-stage building block with gain G_i and noise N_i per stage.

This output noise (N_{out}) can be referred to the input to derive an equivalent input noise (N_{eq}):

$$N_{eq} = \frac{N_{out}}{Gain} = \frac{N_{out}}{G_1G_2G_3} = N_{in} + \frac{N_1}{G_1} + \frac{N_2}{G_1G_2} + \frac{N_3}{G_1G_2G_3} \tag{21.10}$$

According to Eq. 22.10, the gain of the first stage should be high in order to reduce noise contributions from subsequent stages. However, if the gain is too high, a large input signal may saturate the

subsequent stages, yielding intermodulation products which corrupt the desired signal. Thus, optimization is inevitable.

Input Impedance Matching

The input impedance of the LNA must be matched to the antenna impedance over the frequency range of interest, in order to transfer the maximum available power to the receiver.

Linearity

Unwanted signals at frequencies fairly near the frequency band of interest may reach the LNA with signal strengths many times higher than that of the wanted signal. The LNA must be sufficiently linear to prevent these out-of-band signals from generating intermodulation products within the wanted frequency band, and thus degrading the reception of the desired signal. Since third-order mixing products are usually dominant, the linearity of the LNA is related to the "third-order intercept point" (IP3), which is defined as the input power level that results in equal power levels for the output fundamental frequency component and the third-order intermodulation components. The dynamic range of a wireless receiver is limited at the lower bound by noise and at the upper band by non-linearity.

CMOS Common-Source LNA: Simplified Analysis

Input Impedance Matching by Source Degeneration

For maximum power transfer, the input impedance of the LNA must be matched to the source resistance, which is normally 50 Ω. Impedance-matching circuits consist of reactive components and therefore are (ideally) lossless and noiseless. Figure 21.16 shows the small signal equivalent circuit of a CS LNA input stage with impedance-matching circuit, where the gate-drain capacitance C_{gd} is assumed to have negligible effect and is thus neglected.[16,17] The input impedance of this CS input stage is given by:

$$Z_{in} = j\omega(L_g + L_s) + \frac{1}{j\omega C_{gs}} + \frac{g_m}{C_{gs}}L_s \tag{21.11}$$

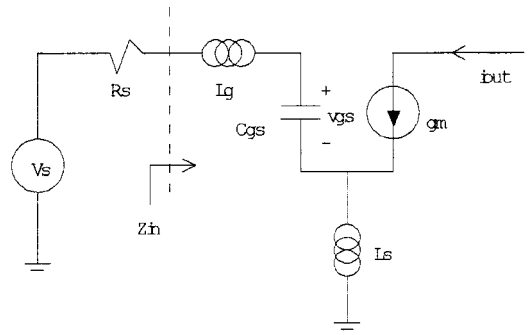


FIGURE 21.16 Simplified small-signal equivalent circuit of the CS stage.

Thus, for matching, the two conditions below must be satisfied:

$$\text{(i)} \quad \omega_o^2 = \frac{1}{(L_g + L_s)C_{gs}} \quad \text{and} \quad \text{(ii)} \quad \frac{g_m}{C_{gs}}L_s = Rs \tag{21.12}$$

Noise Figure of CS Input Stage

Two main noise sources exist in a CS input stage as shown in Fig. 21.17; thermal noise from the source resistor R_s (denoted $\overline{v_{Rs}^2}$) and channel thermal noise from the input transistor (denoted $\overline{i_d^2}$). The output noise current due to $\overline{v_{Rs}^2}$ can be determined from Fig. 21.17 as:

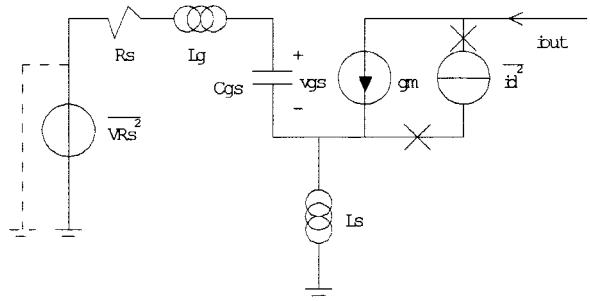


FIGURE 21.17 Simplified noise equivalent circuit of the CS stage. $\overline{V_{Rs}^2} = 4kT_{Rs}$; $\overline{id^2} = KT\Gamma g_{dc}$.

$$\overline{i_{nout1}^2} = \frac{g_m^2 \overline{v_{Rs}^2}}{\omega^2 (g_m L_s + R_s C_{gs})^2} = \frac{g_m^2}{4\omega^2 R_s^2 C_{gs}^2} \overline{v_{Rs}^2} \tag{21.13}$$

while the output noise current due to $\overline{i_d^2}$ can be evaluated as:

$$\overline{i_{nout2}} = \frac{\overline{i_d}}{\left(1 + \frac{g_m L_s}{R_s C_{gs}}\right)} = \frac{1}{2}\overline{i_d} \qquad \therefore \overline{i_{nout2}^2} = \frac{1}{4}\overline{i_d^2} \tag{21.14}$$

From Eqs. 21.13 and 21.14, the noise figure of the CS input stage is determined as:

$$NF = 1 + \frac{\overline{i_{nout2}^2}}{\overline{i_{nout1}^2}} = 1 + \Gamma\left(\frac{\omega_o^2 R_s C_{gs}^2}{g_m}\right) = 1 + \Gamma\left(\frac{L_s}{L_s + L_g}\right) \tag{21.15}$$

In practice, any inductor (especially a fully integrated inductor) has an associated resistance that will contribute thermal noise, degrading the noise figure in Eq. 21.15.

Voltage Amplifier with Inductive Load

Referring to Fig. 21.15, the small signal current output is given by:

$$i_{out} = \frac{g_m v_s}{[1 - \omega^2 C_{gs}(L_g + L_s)] + j\omega(g_m L_s + R_s C_{gs})} \tag{21.16}$$

For an inductive load (L_1) with a series internal resistance r_{L1}, the output voltage is thus:

$$v_{out} = -i_{out}(r_{L1} + j\omega L_1) = \frac{-(r_{L1} + j\omega L_1) g_m v_s}{[1 - \omega^2 C_{gs}(L_g + L_s)] + j\omega(g_m L_s + R_s C_{gs})} \tag{21.17}$$

Assuming that the input is impedance matched, the voltage gain at the output is given by:

$$\left|\frac{v_{out}}{v_s}\right| = \frac{\sqrt{(r_{L1})^2 + (\omega_o)^2 (L_1)^2}}{2\omega_o L_s} = \frac{r_{L1}}{2\omega_o L_s}\sqrt{1 + \left(\frac{\omega_o L_1}{r_{L1}}\right)^2} \cong \frac{1}{2}\omega_o \left(\frac{L_1}{L_s}\right)\left(\frac{L_1}{r_{L1}}\right) \tag{21.18}$$

CMOS Common-Source LNA: Effect of C_{gd}

In the analysis so far, the gate-drain capacitance (C_{gd}) has been assumed to be negligible. However, at very high frequencies, this component cannot be neglected. Figure 21.18 shows the modified input stage of a CS LNA including C_{gd} and an input ac-coupling capacitance C_{in}. Small signal analysis shows that the input impedance is now given by:

$$Z_{in} = \frac{g_m L_s}{C_{gs} + C_{gd}\left[j\omega L_s g_m + \dfrac{g_m(1-\omega^2 L_s C_{gd})}{\dfrac{1}{Z_L} + j\omega C_{gd}} \right]} \tag{21.19}$$

Equation 21.19 exhibits resonance frequencies that occur when:

$$1 - \omega^2 L_s C_{gs} = 0 \qquad \text{and} \qquad 1 - \omega^2 L_g C_{in} = 0 \tag{21.20}$$

Equation 21.19 indicates that the input impedance matching is degraded by the load Z_L when C_{gd} is included in the analysis.

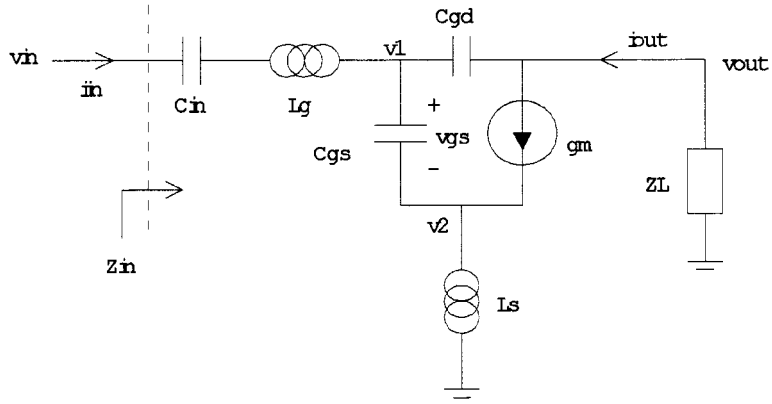


FIGURE 21.18 Noise equivalent circuit of the CS stage, including effects of C_{gd}.

Input Impedance with Capacitive Load

If the load Z_L is purely capacitive, that is,

$$Z_L = \frac{1}{j\omega C_L} \tag{21.21}$$

then the input impedance can be easily matched to the source resistor R_s. Substituting Eq. 21.21 for Z_L, the bracketed term in the denominator of Eq. 21.19 becomes:

$$d_1 = j\omega L_s g_m + \frac{g_m(1-\omega^2 L_s C_{gd})}{j\omega(C_{gd} + C_L)} = 0 \tag{21.22}$$

under the condition that

$$1 - \omega^2 L_s(2C_{gd} + C_L) = 0 \tag{21.23}$$

The three conditions in Eqs. 21.20 and 21.23 should be met to ensure input impedance matching. However, in practice, we are unlikely to be in the situation of using a load capacitor.

Input Impedance with Inductive Load

If $Z_L = j\omega L_L$, the CS LNA input impedance is given by:

$$Z_{in} = \frac{g_m L_s}{C_{gs} + j\omega C_{gd} g_m \left[L_s + L_L \left(\frac{1 - \omega^2 L_s C_{gd}}{1 - \omega^2 L_L C_{gd}} \right) \right]} \tag{21.24}$$

In order to match to a purely resistive input, the value of the reactive term in Eq. 21.24 must be negligible, which is difficult to achieve.

Cascode CS LNA

Input Matching

As outlined in the paragraph above, the gate-drain capacitance (C_{gd}) degrades the input impedance matching and therefore reduces the power transfer efficiency. In order to reduce the effect of C_{gd}, a cascoded structure can be used.[18–20] Figure 21.19 shows a cascode CS LNA. Since the voltage gain from the gate to the drain of M1 is unity, the gate-drain capacitance (C_{gd1}) no longer sees the full input-output voltage swing which greatly improves the input-output isolation. The input impedance can be approximated by Eq. 21.11, thus allowing a simple matching circuit to be employed.[18]

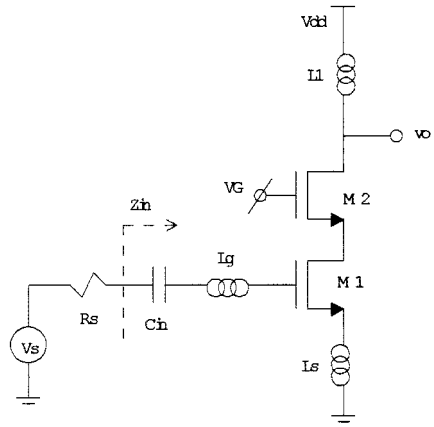


FIGURE 21.19 Cascode CS LNA.

Voltage Gain

Figure 21.20 shows the small-signal equivalent circuit of the cascode CS LNA. Assuming that input is fully matched to the source, the voltage gain of the amplifier is given by:

$$\frac{v_{out}}{v_s} = -\frac{1}{2}\left(\frac{j\omega L_1}{1 - \omega^2 L_1 C_{gd2}} \right)\left(\frac{g_{m2}}{g_{m2} + j\omega C_{gs2}} \right)\left[\frac{g_{m1}}{(1 - \omega^2 L_s C_{gs1}) + j\omega L_s g_{m1}} \right] \tag{21.25}$$

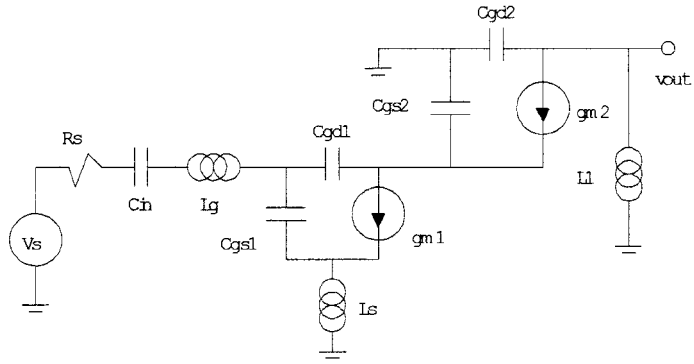


FIGURE 21.20 Equivalent circuit of cascode CS LNA.

At the resonant frequency, the voltage gain is given by:

$$\frac{v_{out}}{v_s}(\omega_o) = -\frac{1}{2}\left(\frac{L_1}{L_s}\right)\left(\frac{1}{1-\omega_o^2 L_1 C_{gd2}}\right)\times\frac{1}{1+j\omega_o\left(\frac{C_{gs1}}{g_{m2}}\right)} \approx -\frac{1}{2}\left(\frac{L_1}{L_s}\right)\times\frac{1}{1+j\left(\frac{\omega_o}{\omega_T}\right)} \quad (21.26)$$

From Eq. 21.26, the voltage gain is dependent on the ratio of the load and source inductance values. Therefore, high gain accuracy can be achieved since this ratio is largely process independent.

Noise Figure

Figure 21.21 shows an equivalent circuit of the cascode CS LNA for noise calculations. Three main noise sources can be identified: the thermal noise voltage from R_s, and the channel thermal noise currents from M1 and M2. Assuming that the input impedance is matched to the sources, the output noise current due to $\overline{v_{RS}^2}$ can be derived as:

$$\overline{i_{out1}} = \frac{1}{2j\omega_o L_s(1-\omega_o^2 L_1 C_{gd2})}\left(\frac{g_{m2}}{g_{m2}+j\omega_o C_{gs2}}\right)\overline{v_{RS}} \quad (21.27)$$

The output noise current contribution due to $\overline{i_{d1}^2}$ of M1 is given by:

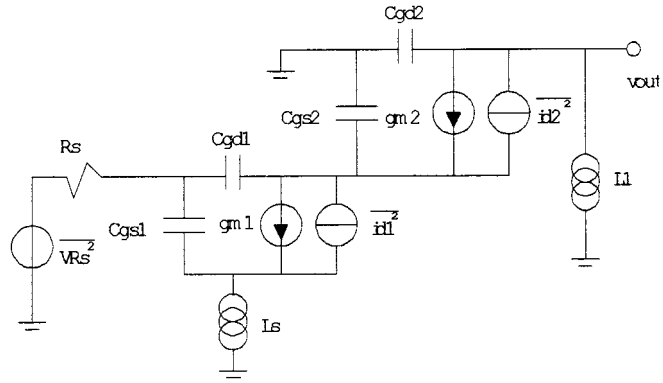


FIGURE 21.21 Noise equivalent circuit of cascode CS LNA.

$$\overline{i_{out2}} = \frac{1}{2(1-\omega_o^2 L_1 C_{gd2})}\left(\frac{g_{m2}}{g_{m2}+j\omega_o C_{ds2}}\right)\overline{i_{d1}} \tag{21.28}$$

The output noise current due to $\overline{i_{d2}^2}$ of M2 is given by:

$$\overline{i_{out3}} = \frac{j\omega_o C_{gs2}}{(1-\omega_o^2 L_1 C_{gd2})(g_{m2}+j\omega_o C_{gs2})}\overline{i_{d2}} \tag{21.29}$$

The noise figure of the cascode CS LNA can thus be derived as:

$$NF = 1 + \frac{\overline{i_{out2}^2}}{\overline{i_{out1}^2}} + \frac{\overline{i_{out3}^2}}{\overline{i_{out1}^2}} = 1 + \Gamma\left(1 + \frac{4\omega_o^2 C_{gs2}^2}{g_{m1}g_{m2}}\right) \tag{21.30}$$

In order to improve the noise figure, the transconductance values (g_m) of M1 and M2 should be increased. Since the gate-source capacitance (C_{gs2}) of M2 is directly proportional to the gate width, the gate width of M2 cannot be enlarged to increase the transconductance. Instead, this increase should be realized by increasing the gate bias voltage.

21.4 Optical Low-Noise Preamplifiers

Figure 21.22 shows a simple schematic diagram of an optical receiver, consisting of a photodetector, a preamplifier, a wide-band voltage amplifier, and a pre-detection filter. Since the front-end transimpedance preamplifier is critical in determining the overall receiver performance, it should possess a wide bandwidth so as not to distort the received signal, high gain to reject noise from subsequent stages, low noise to achieve high sensitivity, wide dynamic range, and low inter-symbol-interference (ISI).

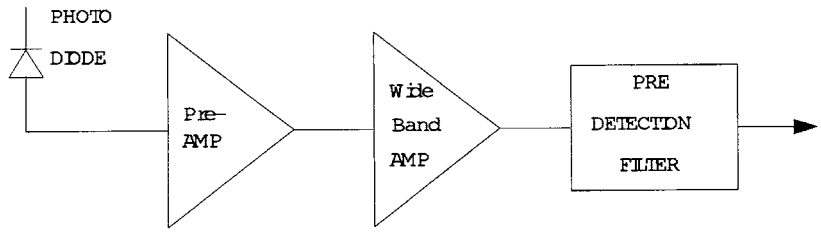


FIGURE 21.22 Front-end optical receiver.

Front-End Noise Sources

Receiver noise is dominated by two main noise sources: the detector (PIN photodiode) noise and the amplifier noise. Figure 21.23 illustrates the noise equivalent circuit of the optical receiver.

PIN Photodiode Noise

The noise generated by a PIN photodiode arises mainly from three shot noise contributions: quantum noise $S_q(f)$, thermally generated dark-current shot noise $S_D(f)$, and surface leakage-current shot noise $S_L(f)$. Other noise sources in a PIN photodiode, such as series resistor noise, are negligible in comparison. The quantum noise $S_q(f)$, also called signal-dependent shot noise, is produced by the light-generating nature of photonic detection and has a spectral density $S_q(f) = 2qI_{pd}\Delta f$, where I_{pd} is the mean signal current arising from the Poisson statistics. The dark-current shot noise $S_D(f)$ arises in the photodiode

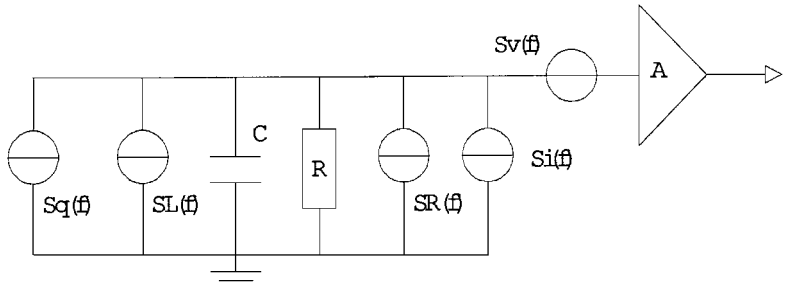


FIGURE 21.23 Noise equivalent circuit of the front-end optical receiver.

bulk material. Even when there is no incident optical power, a small reverse leakage current still flows, resulting in shot noise with a spectral density $S_D(f) = 2qI_{DB}\Delta f$, where I_{DB} is the mean thermally generated dark current. The leakage shot noise $S_L(f)$ occurs because of surface effects around the active region, and is described by $S_L(f) = 2qI_{SL}\Delta f$, where I_{SL} is the mean surface leakage current.

Amplifier Noise

For a simple noise analysis, the pre- and post-amplifiers in Fig. 21.22 are merged to a single amplifier with a transfer function of $A_v(\omega)$. The input impedance of the amplifier is modeled as a parallel combination of R_{in} and C_{in}.

If the photodiode noise is negligibly small, the amplifier noise will dominate the whole receiver noise performance, as can be inferred from Fig. 21.23. The equivalent noise current and voltage spectral densities of the amplifier are represented as $S_i(\text{A}^2/\text{Hz})$ and $S_v(\text{V}^2/\text{Hz})$, respectively.

Resistor Noise

The thermal noise generated by a resistor is directly proportional to the absolute temperature T and is represented by a series noise voltage generator or by a shunt noise current generator[21] of value:

$$\overline{v_R^2} = 4kTR\Delta f \qquad \text{or} \qquad \overline{i_R^2} = 4kT\frac{1}{R}\Delta f \tag{21.31}$$

where k is Boltzmann's constant and R is the resistance.

Receiver Performance Criteria

Equivalent Input Noise Current $\langle \overline{i_{eq}^2} \rangle$

The transfer function from the current input to the amplifier output voltage is given by:

$$Z_T(\omega) = \frac{V_{out}}{I_{pd}} = Z_{in}A_v(\omega) = \frac{R_{in}}{1j\omega R_{in}(C_{pd} + C_{in})}A_v(\omega) \tag{21.32}$$

where C_{pd} is the photodiode capacitance, and R_{in} and C_{in} are the input resistance and capacitance of the amplifier, respectively. Assuming that the photodiode noise contributions are negligible and that the amplifier noise sources are uncorrelated, the equivalent input noise current spectral density can be derived from Fig. 21.23 as:

$$S_{eq}(f) = S_i + \frac{S_v}{[Z_{in}]^2} = S_i + S_v\left[\frac{1}{R_{in}^2} + (2\pi f)^2(C_{pd} + C_{in})^2\right] \tag{21.33}$$

The total mean-square noise output voltage $\langle \overline{v_{no}^2} \rangle$ is calculated by combining Eqs. 21.32 and 21.33 as follows:

$$\langle \overline{v_{no}^2} \rangle = \int_0^\infty S_{eq}(f)|Z_T(f)|^2 df \tag{21.34}$$

This total noise voltage can be referred to the input of the amplifier by dividing it by the squared dc gain $|Z_T(0)|^2$ of the receiver, to give an equivalent input mean-square noise current:

$$\begin{aligned}\langle \overline{i_{eq}^2} \rangle &= \frac{\langle \overline{v_{no}^2} \rangle}{|Z_T(0)|^2} = \left(S_i + \frac{S_v}{R_{in}^2}\right)\int_0^\infty \frac{|Z_T(f)|^2}{|Z_T(0)|^2} df + S_v[2\pi(C_{pd} + C_{in})]^2 \int_0^\infty f^2 \frac{|Z_T(f)|^2}{|Z_T(0)|^2} df \\ &= \left(S_i + \frac{S_v}{R_{in}^2}\right) I_2 B + [2\pi(C_{pd} + C_{in})]^2 I_3 B^3 S_v\end{aligned} \tag{21.35}$$

where B is the operating bit-rate, and I_2(= 0.56) and I_3(= 0.083) are the Personick second and third integrals, respectively, as given in Ref. 22.

According to Morikoni et al.,[23] the Personick integral in Eq. 21.35 is correct only if a receiver produces a raised-cosine output response from a rectangular input signal at the cut-off bit rate above which the frequency response of the receiver is zero. However, the Personick integration method is generally preferred when comparing the noise (or sensitivity) performance of different amplifiers.

Optical Sensitivity

Optical sensitivity is defined as the minimum received optical power incident on a perfectly efficient photodiode connected to the amplifier, such that the presence of the amplifier noise corrupts on average only one bit per 10^9 bits of incoming data. Therefore, a detected power greater than the sensitivity level guarantees system operation at the desired performance. The optical sensitivity is predicted theoretically by calculating the equivalent input noise spectral density of the receiver, and is calculated[24] via Eq. 21.36:

$$S = 10\log_{10}\left(Q\frac{hc}{q\lambda}\sqrt{\langle \overline{i_{eq}^2} \rangle} \cdot \frac{1}{1\,mW}\right) \text{ (dBm)} \tag{21.36}$$

where h is Planck's constant, c is the speed of light, q is electronic charge, and λ (μm) is the wavelength of light in an optical fiber. $Q(= \sqrt{\text{SNR}})$, where SNR represents the required signal-to-noise ratio (SNR). The value of Q should be 6 for a bit error rate (BER) of 10^{-9}, and 7.04 for a BER of 10^{-12}. The relation between Q and BER is given by:

$$BER = \frac{\exp(-Q^2/2)}{\sqrt{2\pi}Q} \tag{21.37}$$

Since the number of photogenerated electrons in a single bit is very large (more than 10^4) for optoelectronic integrated receivers,[25] Gaussian statistics of the above BER equation can be used to describe the detection probability in PIN photodiodes.

SNR at the Photodiode Terminal[22]

Among the photodiode noise sources, quantum noise is generally dominant and can be estimated as:

$$\langle \overline{i_n^2} \rangle_q = 2qI_{pd}B_{eq} \tag{21.38}$$

where I_{pd} is the mean signal current and B_{eq} is the equivalent noise bandwidth. The signal-to-noise-ratio (SNR) referred to the photodiode terminal is thus given by:

$$\text{SNR} = \frac{{I_{pd}}^2}{\langle \overline{i_n^2} \rangle_{pd} + \frac{4kTB_{eq}}{R_B} + \langle \overline{i_{eq}^2} \rangle_{amp}} \tag{21.39}$$

where all noise contributions due to the amplifier are represented by the equivalent noise current where all noise contributions due to the amplifier are represented by the equivalent noise current $\langle \overline{i_{eq}^2} \rangle_{amp}$. It is often convenient to combine the noise contributions from the amplifier and the photodiode with the thermal noise from the bias resistor, by defining a noise figure NF:

$$\langle \overline{i_n^2} \rangle_{pd} + \frac{4kTB_{eq}}{R_B} + \langle \overline{i_{eq}^2} \rangle_{amp} = \frac{4kTB_{eq}\text{NF}}{R_B} \tag{21.40}$$

The SNR at the photodiode input is thus given by:

$$\text{SNR} \cong \frac{{I_{pd}}^2 R_B}{4kTB_{eq}\text{NF}} \tag{21.41}$$

Inter-Symbol Interference (ISI)

When a pulse passes through a band-limited channel, it gradually disperses. When the channel bandwidth is close to the signal bandwidth, the expanded rise and fall times of the pulse signal will cause successive pulses to overlap, deteriorating the system performance and giving higher error rates. This pulse overlapping is known as inter-symbol interference (ISI). Even with raised signal power levels, the error performance cannot be improved.[26]

In digital optical communication systems, sampling at the output must occur at the point of maximum signal in order to achieve the minimum error rate. The output pulse shape should therefore be chosen to maximize the pulse amplitude at the sampling instant and give a zero at other sampling points; that is, at multiples of $1/B$, where B is the data-rate. Although the best choice for this purpose is the sinc-function pulse, in practice a raised-cosine spectrum pulse is used instead. This is because the sinc-function pulse is very sensitive to changes in the input pulse shape and variations in component values, and because it is impossible to generate an ideal sinc-function.

Dynamic Range

The dynamic range of an optical receiver quantifies the range of detected power levels within which correct system operation is guaranteed. Dynamic range is conventionally defined as the difference between the minimum input power (which determines sensitivity) and the maximum input power (limited by overload level). Above the overload level, the bit-error-rate (BER) rises due to the distortion of the received signal.

Transimpedance (TZ) Amplifers

High-impedance (HZ) amplifiers are effectively open-loop architectures, and exhibit a high gain but a relatively low bandwidth. The frequency response is similar to that of an integrator, and thus HZ amplifiers require an output equalizer to extend their frequency capabilities. In contrast, the transimpedance (TZ) configuration exploits resistive negative feedback, providing an inherently wider bandwidth and eliminating the need for an output equalizer. In addition, the use of negative feedback provides a relatively low input resistance and thus the architecture is less sensitive to the photodiode parameters. In a TZ amplifier, the photodiode bias resistor R_B can be omitted, since bias current is now supplied through the feedback resistor.

In addition to wider bandwidth, TZ amplifiers offer a larger dynamic range because the transimpedance gain is determined by a linear feedback resistor, and not by a non-linear open-loop amplifier as is the case for HZ amplifiers. The dynamic range of TZ amplifiers is set by the maximum voltage swing available at the amplifier output, provided no integration of the received signal occurs at the front end. Since the TZ output stage is a voltage buffer, the voltage swing at the output can be increased with high current operation. The improvement in dynamic range in comparison to the HZ architecture is approximately equal to the ratio of open-loop to closed-loop gain.[27] Conclusively, the TZ configuration offers the better performance compromise compared to the HZ topology, and hence this architecture is preferred in optical receiver applications.

A schematic diagram of a TZ amplifier with PIN photodiode is shown in Fig. 21.24. With an open-loop, high-gain amplifier and a feedback resistor, the closed-loop transfer function of the TZ amplifier is given by:

$$Z_T(s) = \frac{-R_f}{\left(\frac{1+A}{A}\right) + sR_f\left[\frac{C_{in} + (1+A)C_f}{A}\right]} \cong \frac{R_f}{1 + sR_f\left(\frac{C_{in}}{A} + C_f\right)} \tag{21.42}$$

where A is the open-loop mid-band gain of the amplifier which is assumed to be greater than unity, R_f is the feedback resistance, C_{in} is the total input capacitance of the amplifier including the photodiode and the parasitic capacitance, and C_f represents the stray feedback capacitance. The –3 dB bandwidth of the TZ amplifier is approximately given by:

$$f_{-3d} = \frac{(1+A)}{2\pi R_f C_T} \tag{21.43}$$

where C_T is the total input capacitance including the photodiode capacitance. The TZ amplifier can thus have wider bandwidth by increasing the open-loop gain, although the open-loop gain cannot be increased indefinitely without stability problems.

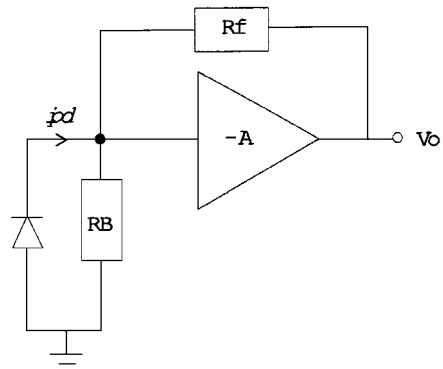


FIGURE 21.24 Schematic diagram of a transimpedance amplifier with photodiode.

However, a tradeoff between low noise and wide bandwidth exists, since the equivalent input noise current spectral density of TZ amplifier is given by:

$$S_{eq}(f) = \frac{4kT}{R_f} + \frac{4kT}{R_B} + S_i(f) + S_v(f)\left[\left(\frac{1}{R_f} + \frac{1}{R_B}\right)^2 + (2\pi f)^2(C_{pd} + C_{in})^2\right] \tag{21.44}$$

where C_{in} is the input capacitance of the input transistor. Increasing the value of R_f reduces the noise current in Eq. 21.44 but also shrinks the bandwidth in Eq. 21.43. This conflict can be mitigated by making A in Eq. 21.43 as large as the closed-loop stability allows.[28] However, the feedback resistance R_f cannot be increased indefinitely due to the dynamic range requirements of the amplifier, since too large a feedback resistance causes the amplifier to be overloaded at high signal levels. This overloading can be avoided by using automatic gain control (AGC) circuitry, which automatically reduces the transimpedance gain in discrete steps to keep the peak output signal constant.[27]

The upper limit of R_f is set by the peak amplitude of the input signal. Since the dc transimpedance gain is approximately equal to the feedback resistance R_f, the output voltage is given by $I_{pd} \times R_f$, where I_{pd} is the signal photocurrent. If this output voltage exceeds the maximum voltage swing at the output, the amplifier will be saturated and the output will be distorted, yielding bit errors. The minimum value of R_f is determined by the output signal level at which the performance of the receiver is degraded due to noise and offsets. For typical fiber-optic communication systems, the input signal power is unknown, and may vary from just above the noise floor to a large value enough to generate 0.5 mA at the detector diode.[29]

The TZ configuration has some disadvantages over HZ amplifiers. The power consumption is fairly high, partly due to the broadband operation provided by negative feedback. A propagation delay exists in the closed-loop of the feedback amplifier that may reduce the phase margin of the amplifier and cause peaking in the frequency response. Additionally, any stray feedback capacitance C_f will further deteriorate the ac performance.

Among three types of TZ configuration in CMOS technology (common-source, common-drain, and common-gate TZ amplifiers), the common-gate configuration has potentially the highest bandwidth due to its inherently lower input resistance. Using a common-gate input configuration, the resulting amplifier bandwidth can be made independent of the photodiode capacitance (which is usually the limiting factor in achieving GHz preamplifier designs). Recently, a novel common-gate TZ amplifier has been demonstrated, which shows superior performance compared to various other configurations.[30,31]

Layout for HF Operation

Wideband high-gain amplifiers have isolation problems irrespective of the choice of technology. Coupling from output to input, from the power supply rails, and from the substrate are all possible. Therefore, careful layout is necessary, and special attention must be given to stray capacitance, both on the integrated circuit and associated with the package.[32]

Input/Output Isolation

For stable operation, a high level of isolation between I/O is necessary. Three main factors degrade the I/O isolation[33,34]: (1) capacitive coupling between I/O signal paths through the air and through the substrate; (2) feedback through the dc power supply rails and ground-line inductance; and (3) the package cavity resonance since at the cavity resonant frequency, the coupling between I/O can become very large.

In order to reduce the unwanted coupling (or to provide good isolation, typically more than 60 dB) between I/O, the I/O pads should be laid out to be diagonally opposite each other on the chip with a thin 'left-to-right' geometry between I/O. The small input signal enters on the left-hand side of the chip, while the large output signal exits on the far right-hand side. This helps to isolate the sensitive input stages from the larger signal output stages.[35,36]

The use of fine line-widths and shielding are effective techniques to reduce coupling through the air. Substrate coupling can be reduced by shielding and by using a thin and low-dielectric substrate. Akazawa et al.[33] suggest a structure for effective isolation: a coaxial-like signal-line for high shielding, and a very thin dielectric dc feed-line structure for low characteristic impedance.

Reduction of Feedback Through the Power Supply Rails

Careful attention should be given to layout of power supply rails for stable operation and gain flatness. Power lines are generally inductive; thus, on-chip capacitive decoupling is necessary to reduce the

high-frequency power line impedance. However, a resonance between these inductive and capacitive components may occur at frequencies as low as several hundred MHz, causing a serious dip in the gain-frequency response and an upward peaking in the isolation-frequency characteristics. One way to reduce this resonance is to add a series damping resistor to the power supply line, making the Q factor of the LC resonance small. Additionally, the power supply line should be widened to reduce the characteristic impedance/inductance. In practice, if the characteristic impedance is as small as several ohms, the dip and peaking do not occur, even without resistive termination.[33]

Resonance also occurs between the IC pad capacitance (C_{pd}) and the bond-wire inductance (L_{bond}). This resonance frequency is typically above 2 GHz in miniature RF packages. Also in layout, the power supply rails of each IC chip stage should be split from the other stages in order to reduce the parasitic feedback (or coupling effect through wire-bonding inductance), which causes oscillation.[34] This helps to minimize crosstalk through power supply rail. The IC is powered through several pads and each pad is individually bonded to the power supply line.

I/O Pads

The bond pads on the critical signal path (e.g., input pad and output pads) should be made as small as possible to minimize the pad-to-substrate capacitance.[35] A floating n-well placed underneath the pad will further reduce the pad capacitance since the well capacitance will appear in series with the pad capacitance. This floating well also prevents the pad metal from spiking into the substrate.

High-Frequency (HF) Ground

The best possible HF grounds to the sources of the driver devices (and hence the minimization of interstage crosstalk) can be obtained by separate bonding of each source pad of the driver MOSFETs to the ground plane that is very close to the chip.[36] A typical bond-wire has a self-inductance of a few nH, which can cause serious peaking within the bandwidth of amplifiers or even instability. By using multiple bond-wires in parallel, the ground-line inductance can be reduced to less than 1 nH.

Flip-Chip Connection

In noisy environments, the noise-insensitive benefits of optical fibers may be lost at the receiver connection between the photodiode and the preamplifier. Therefore, proper shielding, or the integration of both components onto the same substrate, is necessary to prevent this problem. However, proper shielding is costly, while integration restricts the design to GaAs technologies.

As an alternative, the flip-chip interconnection technique using solder bumps has been used.[37,38] Small solder bumps minimize the parasitics due to the short interconnection lengths and avoid damages by mechanical stress. Also, it needs relatively low-temperature bonding and hence further reduces damage to the devices. Easy alignment and precise positioning of the bonding can be obtained by a self-alignment effect. Loose chip alignment is sufficient because the surface tension of the molten solder during re-flow produces precise self-alignment of the pads.[34] Solder bumps are fabricated onto the photodiode junction area to reduce parasitic inductance between the photodiode and the preamplifier.

21.5 Fundamentals of RF Power Amplifier Design

PA Requirements

An important functional block in wireless communication transceivers is the power amplifier (PA). The transceiver PA takes as input the modulated signal to be transmitted, and amplifies this to the power level required to drive the antenna. Because the levels of power required to transmit the signal reliably are often fairly high, the PA is one of the major sources of power consumption in the transceiver. In many systems, power consumption may not be a major concern, as long as the signal can be transmitted with adequate power. For battery-powered systems, however, the limited amount of available energy means that the power consumed by all devices must be minimized so as to extend the transmit time.

Therefore, power efficiency is one of the most important factors when evaluating the performance of a wireless system.

The basic requirement for a power amplifier is the ability to work at low supply voltages as well as high operating frequencies, and the design becomes especially difficult due to the tradeoffs between supply voltage, output power, distortion, and power efficiency that can be made. Moreover, since the PA deals with large signals, small-signal analysis methods cannot be applied directly. As a result, both the analysis and the design of PAs are challenging tasks.

This section will first present a study of various configurations employed in the design of state-of-the-art non-linear RF power amplifiers. Practical considerations toward achieving full integration of PAs in CMOS technology will also be highlighted.

Power Amplifier Classification

Power amplifiers currently employed for wireless communication applications can be classified into two categories: linear power amplifiers and non-linear power amplifiers. For linear power amplifiers, the output signal is controlled by the amplitude, frequency, and phase of the input signal. Conversely, for non-linear power amplifiers, the output signal is only controlled by the frequency of input signal.

Conventionally, linear power amplifiers can be classified as Class A, Class B, or Class AB. These PAs produce a magnified replica of the input signal voltage or current waveform, and are typically used where accurate reproduction of both the envelope and the phase of the signal is required. However, either poor power efficiency or large distortion prevents them from being extensively employed in wireless communications.

Many applications do not require linear RF amplification. Gaussian Minimum Shift Keying (GMSK),[39] the modulation scheme used in the European standard for mobile communications (GSM), is an example of constant envelope modulation. In this case, the system can make use of the greater efficiency and simplicity offered by non-linear PAs. The increased efficiency of non-linear PAs, such as Class C, Class D, and Class E, results from techniques that reduce the average collector voltage–current product (i.e., power dissipation) in the switching device. Theoretically, these switching-mode PAs have 100% power efficiency since, ideally, there is no power loss in the switching device.

Linear Power Amplifiers

Class A

The basic structure of the Class A power amplifier is shown in Fig. 21.25.[40] For Class A amplification, the conduction angle of the device is 360°, that is, the transistor is in its active region for the entire input

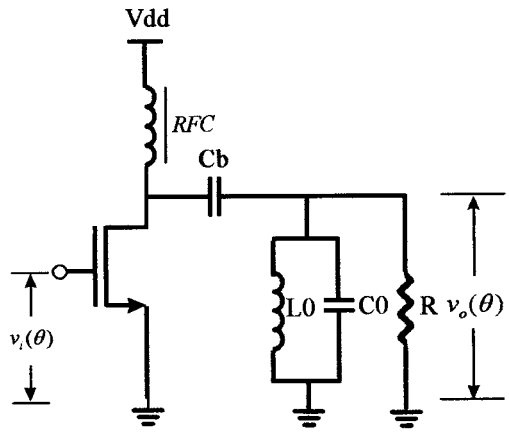


FIGURE 21.25 Single-ended Class A power amplifier.

cycle. The serious shortcoming with Class A PAs is their inherently poor power efficiency, since the transistor is always dissipating power. The efficiency of a single-ended Class A PA is ideally limited to 50%. However, in practice, few designs can reach this ideal efficiency due to additional power loss in the passive components. In an inductorless configuration, the efficiency is only about 25%.[41]

Class B

A PA is defined as Class B when the conduction angle for each transistor of a push-pull pair is 180° during any one cycle. Figure 21.26 shows an inductorless Class B power amplifier. Since each transistor only conducts for half of the cycle, the output suffers crossover distortion due to the finite threshold voltage of each transistor. When no signal is applied, there is no current flowing; as a result, any current through either device flows directly to the load, thereby maximizing the efficiency. The ideal efficiency can reach 78%,[41] allowing this architecture to be of use in applications where linearity is not the main concern.

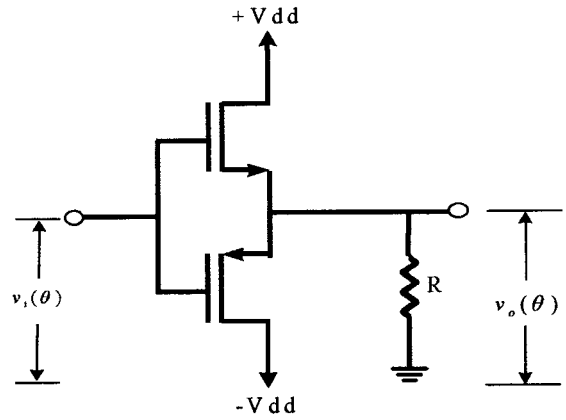


FIGURE 21.26 Inductorless Class B power amplifier.

Class AB

The basic idea of Class AB amplification is to preserve the Class B push-pull configuration while improving the linearity by biasing each device slightly above threshold. The implementation of Class AB PAs is similar to Class B configurations. By allowing the two devices to conduct current for a short period, the output voltage waveform during the crossover period can be smoothed, which thus reduces the crossover distortion of the output signal.

Nonlinear Power Amplifiers

Class C

A Class C power amplifier is the most popular non-linear power amplifier used in the RF band. The conduction angle is less than 180° since the switching transistor is biased on the verge of conduction. A portion of the input signal will make the transistor operate in the amplifying region, and thus the drain current of the transistor is a pulsed signal. Figures 21.27(a) and (b) show the basic configuration of a Class C power amplifier and its corresponding waveforms; clearly, the input and output voltages are not linearly related.

The efficiency of an ideal Class C amplifier is 100% since at any point in time, either the voltage or the current waveforms are zero. In practice, this ideal situation cannot be achieved, and the power efficiency should be maximized by reducing the power loss in the transistor. That is, minimize the current through the transistor when the voltage across the output is high, and minimize the voltage across the output when the current flows through the device.

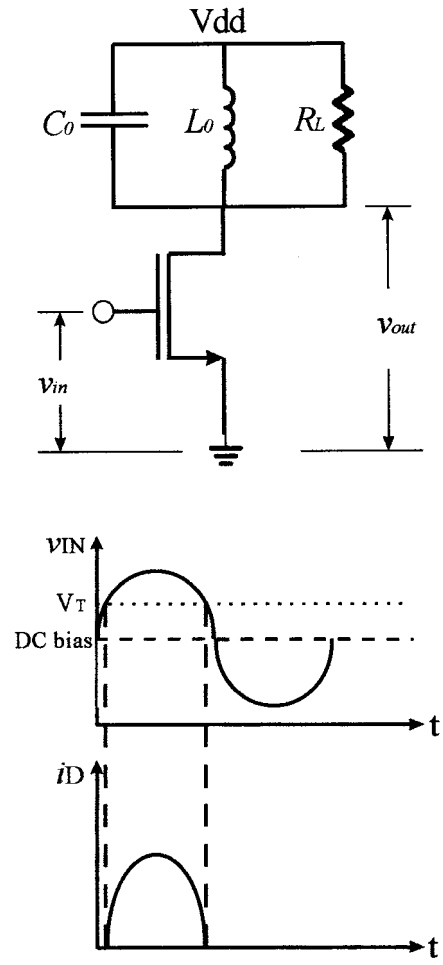


FIGURE 21.27 (a) Class C power amplifier, and (b) Class C waveforms.

Class D

A Class D amplifier employs a pair of transistors and a tuned output circuit, where the transistors are driven to act as a two-pole switch and the output circuit is tuned to the switching frequency. The theoretical power efficiency is 100%. Figure 21.28 shows the voltage-switching configuration of a Class D amplifier. The input signals of transistors Q_1 and Q_2 are out of phase, and consequently when Q_1 is on, Q_2 is off, and vice versa. Since the load network is a tuned circuit, we can assume that it provides little impedance to the operating frequency of the voltage v_d and high impedance to other harmonics. Since v_d is a square wave, its Fourier expansion is given by

$$v_d(\omega t) = V_{dc}\left[\frac{1}{2} + \frac{2}{\pi}\sin(\omega t) + \frac{2}{3\pi}\sin(3\omega t)\ldots\right] \tag{21.45}$$

The impedance of the RLC series load at resonance is equal to R_L, and thus the current is given by:

$$i_L(\omega t) = \frac{2V_{dc}}{\pi R_L}\sin(\omega t) \tag{21.46}$$

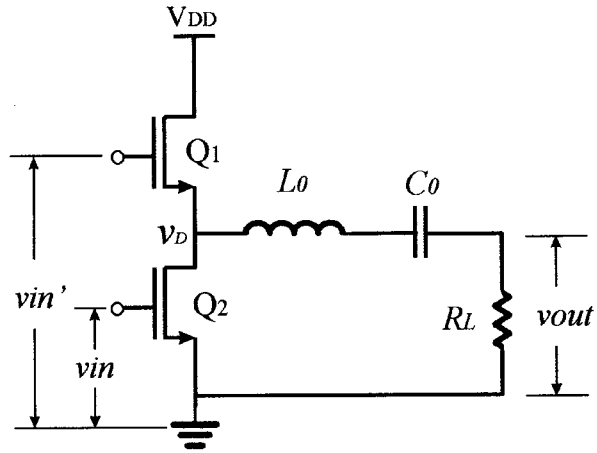


FIGURE 21.28 Class D power amplifier.

Each of the devices carries the current during one half of the switching cycle. Therefore, the output power is given by:

$$P_o = \frac{2}{\pi^2}\frac{V_{dc}^{\ 2}}{R_L} \tag{21.47}$$

Design efforts should focus on reducing the switching loss of both transistors as well as generating the input driving signals.

Class E

The idea behind the Class E PA is to employ non-overlapping output voltage and output current waveforms. Several criteria for optimizing the performance can be found in Ref. 42. Following these guidelines, Class E PAs have high power efficiency, simplicity, and relatively high tolerance to circuit variations.[43] Since there is no power loss in the transistor as well as in the other passive components, the ideal power efficiency is 100%. Figure 21.29 shows a class E PA, and the corresponding waveforms are given in Fig. 21.30.

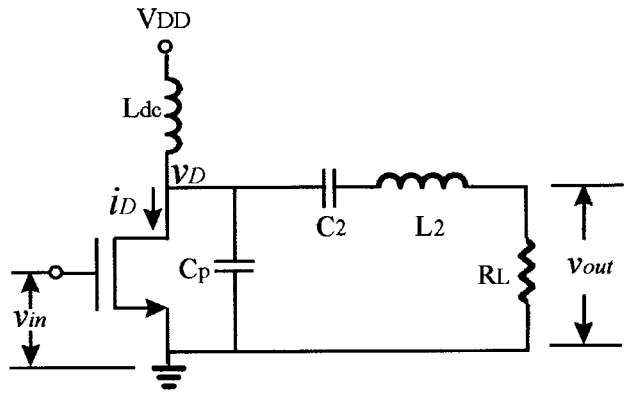


FIGURE 21.29 Class E power amplifier.

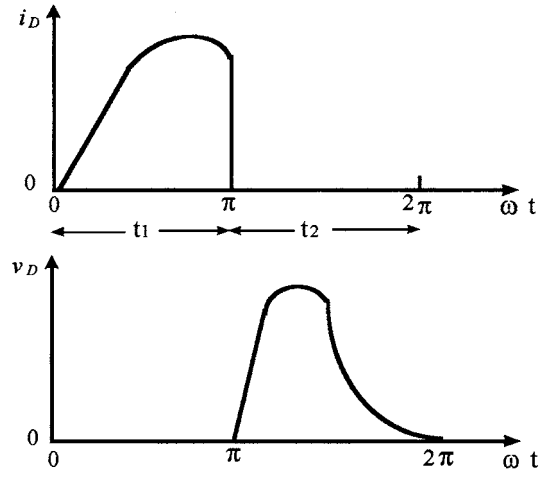


FIGURE 21.30 Waveforms of Class E operation.

The Class E waveforms indicate that the transistor should be completely off before the voltage across it changes, and that the device should be completely on before it starts to allow current to flow through it. Refs. 44 and 45 demonstrate practical Class E operation at RF frequencies using a GaAs process.

Practical Considerations for RF Power Amplifiers

More recently, single-chip solutions for RF transceivers have become a goal for modern wireless communications due to potential savings in power, size, and cost. CMOS must clearly be the technology of choice for a single-chip transceiver due to the large amount of digital baseband processing required. However, the power amplifier design presents a bottleneck toward full integration, since CMOS power amplifiers are still not available. The requirements of low supply voltage, gigahertz-band operation, and high output power make the implementation of CMOS PAs very demanding. The proposal of "microcell" communications may lead to a relaxed demand for output power levels that can be met by designs such as that described in Ref. 46, where a CMOS Class C PA has demonstrated up to 50% power efficiency with 20 mW output power.

Non-linear power amplifiers seem to be popular for modern wireless communications due to their inherent high power efficiency. Since significant power losses occur in the passive inductors as well as the switching devices, the availability of on-chip, low-loss passive inductors is important. The implementation of CMOS on-chip spiral inductors has therefore become an active research topic.[47]

Due to the poor spectral efficiency of a constant envelope modulation scheme, the high power efficiency benefit of non-linear power amplifiers is eliminated. A recently proposed linear transmitter using a non-linear power amplifier may prove to be an alternative solution.[48] The development of high mobility devices such as SiGe HBTs has led to the design of PAs demonstrating output power levels up to 23 dBm at 1.9 GHz with power-added efficiency of 37%.[49] Practical power amplifier designs require that much attention be paid to issues of package and harmonic terminations. Power losses in the matching networks must be absolutely minimized, and tradeoffs between power-added efficiency and linearity are usually achieved through impedance matching. Although GaAs processes provide low-loss impedance matching structures on the semi-insulating substrate, good shielding techniques for CMOS may prove to be another alternative.

Conclusions

Although linear power amplifiers provide conventional "easy-design" characteristics and linearity for modulation schemes such as π/4-DQPSK, modern wireless transceivers are more likely to employ

non-linear power amplifiers due to their much higher power efficiency. As the development of high-quality on-chip passive components makes progress, the trend toward full integration of the PA is becoming increasingly plausible.

The rapid development of CMOS technology seems to be the most promising choice for PA integration, and vast improvements in frequency performance have been gained through device scaling. These improvements are expected to continue as silicon CMOS technologies scale further, driven by the demand for high-performance microprocessors. The further development of high mobility devices such as SiGe HBTs may finally see GaAs MOSFETs being replaced by wireless communication applications, since SiGe technology is compatible with CMOS.

21.6 Applications of High-Q Resonators in IF-Sampling Receiver Architectures

Transconductance-C (gm-C) filters are currently the most popular design approach for realizing continuous-time filters in the intermediate frequency range in telecommunications systems. This section will consider the special application area of high-Q resonators for receiver architectures employing IF sampling.

IF Sampling

A design approach for contemporary receiver architectures that is currently gaining popularity is IF digitization, whereby low-frequency operations such as second mixing and filtering can be performed more efficiently in the digital domain. A typical architecture is shown in Fig. 21.31. The IF signal is digitized, multiplied with the quadrature phases of a digital sinusoid, and lowpass filtered to yield the quadrature baseband signals. Since processing takes place in the digital domain, I/Q mismatch problems are eliminated. The principal issue in this approach, however, is the performance required from the A/D converter (ADC). Noise referred to the input of the ADC must be very low so that selectivity remains high. At the same time, the linearity of the ADC must be high to minimize corruption of the wanted signal through intermodulation effects. Both the above requirements should be achieved at an input bandwidth commensurate with the value of the IF frequency, and at an acceptable power budget.

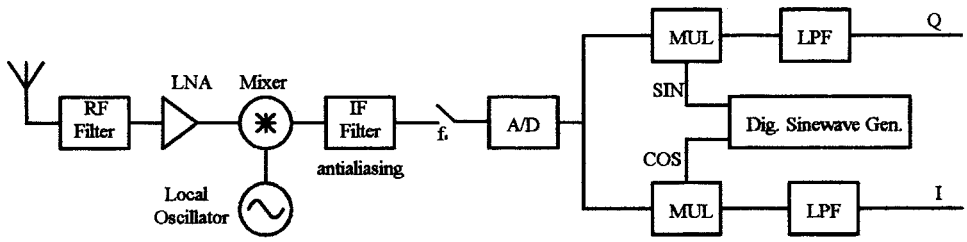


FIGURE 21.31 IF-sampling receiver.

Oversampling has become popular in recent years because it avoids many of the difficulties encountered with conventional methods for A/D and D/A conversion. Conventional converters are often difficult to implement in fine-line, very large-scale integration (VLSI) technology, because they require precise analog components and are very sensitive to noise and interference. In contrast, oversampling converters trade off resolution in time for resolution in amplitude, in such a way that the imprecise nature of the analog circuits can be tolerated. At the same time, they make extensive use of digital signal processing power, taking advantage of the fact that fine-line VLSI is better suited for providing fast digital circuits than for

providing precise analog circuits. Therefore, IF-digitization techniques utilizing oversampling Sigma-Delta modulators are very well suited to modern sub-micron CMOS technologies, and their potential has made them the subject of active research.

Most Delta-Sigma modulators are implemented with discrete-time circuits, switched-capacitor (SC) implementations being by far the most common. This is mainly due to the ease with which monolithic SC filters can be designed, as well as the high linearity which they offer. The demand for high-speed ΣΔ oversampling ADCs, especially for converting bandpass signals, makes it necessary to look for a technique that is faster than switched-capacitor. This demand has stimulated researchers to develop a method for designing continuous-time ΔΣ ADCs. Although continuous-time modulators are not easy to integrate, they possess a key advantage over their discrete-time counterparts. The sampling operation takes place inside the modulator loop, making it is possible to "noise-shape" the errors introduced by sampling, and provide a certain amount of anti-aliasing filtering at no cost. On the other hand, they are sensitive to memory effects in the DACs and are very sensitive to jitter. They must also process continuous-time signals with high linearity. In communications applications, meeting the latter requirement is complicated by the fact that the signals are located at very high frequencies.

As shown in Fig. 21.32, integrated bandpass implementations of continuous-time modulators require integrated continuous-time resonators to provide the noise shaping function. The gm-C approach of realizing continuous-time resonators offers advantages of complete system integration and total design freedom. However, the design of CMOS high-Q high-linearity resonators at the tens of MHz is very challenging. Since the linearity of the modulator is limited by the linearity of the resonators utilized, the continuous-time resonator is considered to be the most demanding analog sub-block of a bandpass continuous-time Sigma-Delta modulator. Typical specifications for a gm-C resonator used to provide the noise-shaping function in a ΣΔ modulator in a mobile receiver (see Fig. 21.32) are summarized in Table 21.4.

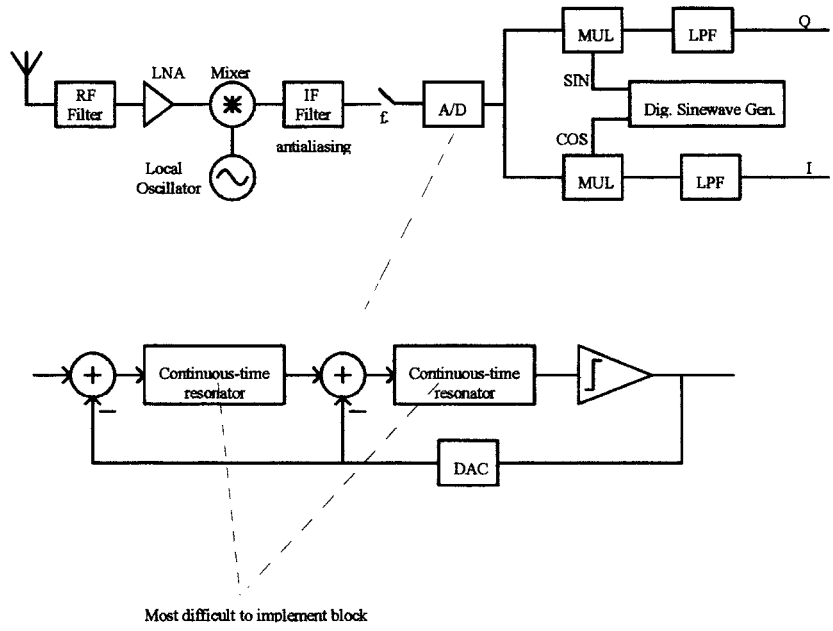


FIGURE 21.32 Continuous-time ΣΔ A/D in IF-sampling receiver.

Linear Region Transconductor Implementation

The implementation of fully integrated, high-selectivity filters operating at tens to hundreds of MHz provides benefits for wireless transceiver design, including chip area economy and cost reduction. The

TABLE 21.4 Fully Integrated Continuous-Time Resonator Specifications

Resonator Specifications	
Center frequency	50 MHz
Quality factor	50
Spurious free dynamic range	>30 dB
Power dissipation	Minimal

main disadvantages of on-chip active filter implementations when compared to off-chip passives include increased power dissipation, deterioration in the available dynamic range with increasing Q, and Q and resonant frequency integrity (because of process variations, temperature drifts, and aging, automatic tuning is often unavoidable, especially in high-Q applications). The transconductor-capacitor (gm-C) technique is a popular technique for implementing high-speed continuous time filters and is widely used in many industrial applications.[52] Because gm-C filters are based on integrators built from an open-loop transconductance amplifier driving a capacitor, they are typically very fast but have limited linear dynamic range. Linearization techniques that reduce distortion levels can be used, but often lead to a compromise between speed, dynamic range, and power consumption.

As an example of the tradeoffs in design, consider the transconductor shown in Fig. 21.33. This design consists of a main transconductor cell (M1, M2, M3, M4, M10, M11, and M14) with a negative resistance load (M5, M6, M7, M8, M9 , M12, and M13). Transistors M1 and M2 are biased in the triode region of operation using cascode devices M3 and M4 and determine the transconductance gain of the cell. In the triode region of operation, the drain current versus terminal voltage relation can be approximated (for simple hand calculations) as $I_D = K[2(V_{GS} - V_T)V_{DS} - V_{DS}^2]$, where K and V_T are the transconductance parameter and the threshold voltage respectively. Assuming that V_{DS} is constant for both M1 and M2, both the differential mode and the common mode transconductance gains can be derived as $G_{DM} = G_{CM} = 2KV_{DS}$, which can thus be tuned by varying V_{DS}.

The high value of common-mode transconductance is undesirable since it may result in regenerative feedback loops in high-order filters. To improve the CMRR transistor and avoid the formation of such loops, M10 is used to bias the transconductor, thus transforming it from a pseudo-differential to a fully differential transconductor.[53] Transistors M11 and M14 constitute a floating voltage source, thus maintaining a constant drain-source voltage for M1 and M2.

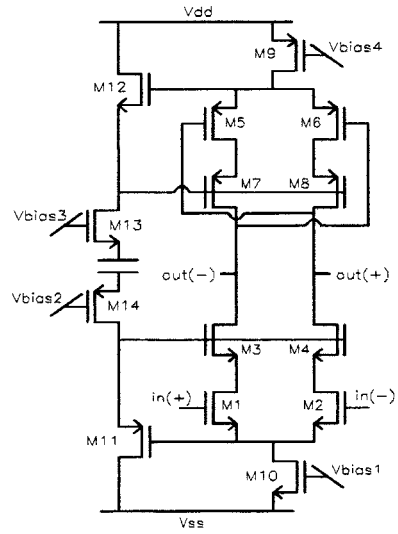


FIGURE 21.33 Triode region transconductor.

The non-linearities in the voltage-to-current transfer of this stage are mainly due to three effects. The first is the finite impedance levels at the sources of the cascode devices, which cause a signal-dependent variation of the corresponding drain-source voltages of M1 and M2. A fast floating voltage source and large cascode transistors therefore need to be used to minimize this non-linearity. The second cause of non-linearity is the variation of carrier mobility μ of the input devices M1 and M2 with $V_{GS} - V_T$, which becomes more apparent when short-channel devices are used ($K = \mu \cdot C_{ox} \cdot W/2 \cdot L$). A simple first-order model for transverse-field mobility degradation is given by $\mu = \mu_0/(1 + \theta \cdot (V_{GS} - V_T))$, where μ_0 and θ are the zero field mobility and the mobility reduction parameter, respectively. Using this model, the third-order distortion can be determined by a Maclaurin series expansion as $\theta^2/4(1 + \theta(V_{CM} - V_T))$.[54] This expression cannot be regarded as exact, although it is useful to obtain insight. Furthermore, it is valid only at low frequencies, where reactive effects can be ignored and the coefficients of the Maclaurin series expansion are frequency independent. At high frequencies or when very low values of distortion are predicted by the Maclaurin series method, a generalized power series method (Volterra series) must be employed.[55,56] Finally, a further cause of non-linearity is mismatch between M1 and M2, which can be minimized by good layout. A detailed linearity analysis of this transconductance stage is presented in Ref. 60.

To provide a load for the main transconductor cell, a similar cell implemented by p-devices is used. The gates of the linear devices M5 and M6 are now cross-coupled with the drains of the cascode devices M7 and M8. In this way, weak positive feedback is introduced. The differential-mode output resistance can now become negative and is tuned by the V_{DS} of M5 and M6 (M12 and M13 form a floating voltage source), while the common-mode output resistance attains a small value.

When connected to the output of the main transconductor cell as shown in Fig. 21.33, the cross-coupled p-cell forms a high-ohmic load for differential signals and a low-ohmic load for common-mode signals, resulting in a controlled common-mode voltage at the output.[54,57] CMRR can be increased even further using M10, as described previously. Transistor M9 is biased in the triode region of operation and is used to compensate the offset common-mode voltage at the output.

The key performance parameter of an integrator is the phase shift at its unity-gain frequency. Deviations from the ideal –90° phase include phase lead due to finite dc gain and phase lag due to high-frequency parasitic poles. In the transconductor design of Fig. 21.33, dc gain is traded for phase accuracy, thus compensating the phase lag introduced by the parasitic poles. The reduction in dc gain for increased phase accuracy is not a major problem for bandpass filter applications, since phase accuracy at the center frequency is extremely important while dc gain has to be adequate to ensure that attenuation specifications are met at frequencies below the passband.

From simulation results using parameters from a 0.8-μm CMOS process, with the transconductor unity gain frequency set at 50 MHz, third-order intermodulation components were observed at –78 dB with respect to the fundamental signals (two input signals at 49.9 MHz and 50.1 MHz were applied, each at 50 mVpp).

A gm-C Bandpass Biquad

Filter Implementation

The implementation of on-chip high-Q resonant circuits presents a difficult challenge. Integrated passive inductors have generally poor quality factors, which limits the Q of any resonant network in which they are employed. For applications in the hundreds of MHz to a few GHz, one approach is to implement the resonant circuit using low-Q passive on-chip inductors with additional Q-enhancing circuitry. However, for lower frequencies (tens of MHz), on-chip inductors occupy a huge area and this approach is not attractive.

As disscussed above, an alternative method is to use active circuitry to eliminate the need for inductors. gm-C-based implementations are attractive due to their high-speed potential and good tunability. A bandpass biquadratic section based upon the transconductor of Fig. 21.33 is shown in Fig. 21.34. The transfer function of Fig. 21.34 is given by:

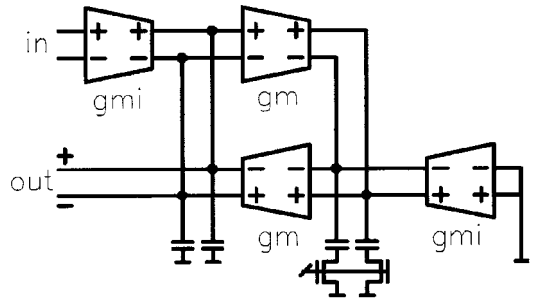


FIGURE 21.34 Biquad bandpass.

$$\frac{V_o}{V_i} = \frac{g_{mi} \cdot R_o}{(R_o \cdot C)^2} \cdot \frac{(1 + s \cdot R_o \cdot C)}{\left\{ s^2 + s\frac{2 \cdot R_o + g_m{}^2 \cdot R_o{}^2 \cdot R}{R_o{}^2 \cdot C} + \frac{1 + g_m{}^2 \cdot R_o{}^2}{R_o{}^2 \cdot C^2} \right\}} \tag{21.48}$$

R_o represents the total resistance at the nodes due to the finite output resistance of the transconductors. R represents the effective resistance of the linear region transistors in the transconductor (see Fig. 21.33), and is used here to introduce damping and control the Q. From Eq. 21.48, it can be shown that $\omega_o \approx g_m/C$, $Q \approx g_m \cdot R_o/(2 + R_o \cdot R \cdot g_m{}^2)$, $Q_{max} = Q|_{r=0} = (g_m \cdot R_o)/2$ and $A_o = g_{mi} \cdot Q$. Thus, g_m is used to set the central frequency, R is used to control the Q, and g_{mi} controls the bandpass gain A_o. A dummy g_{mi} is used to provide symmetry and thus better stability due to process variations, temperature, and aging.

One of the main problems when implementing high-Q high-frequency resonators is maintaining the stability of the center frequency ω_o and the quality factor Q. This problem calls for very careful layout and the implementation of an automatic tuning system. Another fundamental limitation regarding available dynamic range occurs: namely, that the dynamic range (DR) of high-Q gm-C filters has been found to be inversely proportional to the filter Q.[57] The maximum dynamic range is given by:

$$\text{DR} = \frac{V_{max}{}^2}{V_{noise}{}^2} = \frac{V_{max}{}^2 \cdot C}{4 \cdot k \cdot T \cdot \xi \cdot Q} \tag{21.49}$$

where V_{max} is the maximum rms voltage across the filter capacitors, C is the total capacitance, k is Boltzman's constant, T is the absolute temperature, and ξ is the noise factor of the active circuitry ($\xi = 1$ corresponds to output noise equal to the thermal noise of a resistor of value $R = 1/g_m$, where g_m is the transconductor value used in the filter).

In practice, the dynamic range achieved will be less than this maximum value due to the amplification of both noise and intermodulation components around the resonant frequency. This is a fundamental limitation, and the only solution is to design the transconductors for low noise and high linearity. The linearity performance in narrowband systems is characterized by the spurious-free dynamic range (SFDR). SFDR is defined as the signal-to-noise ratio when the power of the third-order intermodulation products equals the noise power. As shown in Ref. 60, the SFDR of the resonator in Fig. 21.34 is given by:

$$\text{SFDR} = \frac{1}{2(k \cdot T)^{2/3}} \left(\frac{3 \cdot V_{o,\,peak}{}^2 \cdot C}{4 \cdot \xi \cdot IM_{3,\,int}} \right)^{2/3} \frac{1}{Q^2} \tag{21.50}$$

where $IM_{3,\,int}$ is the third-order intermodulation point of the integrator used to implement the resonator. The spurious free dynamic range of the resonator thus deteriorates by 6 dB if the quality factor is doubled,

assuming that the output swing remains the same. In contrast, implementing a resonant circuit using low-Q passive on-chip inductors with additional Q-enhancing circuitry leads to a dynamic range amplified by a factor Q_o, where Q_o is the quality factor of the on-chip inductor itself.[59] However, as stated above, for frequencies in the tens of MHz, on-chip inductors occupy a huge area and thus the Q_o improvement in dynamic range is not high enough to justify the area increase.

Simulation Results

To confirm operation, the filter shown in Fig. 21.34 has been simulated in HSPICE using process parameters from a commercial 0.8-μm CMOS process. Figure 21.35 shows the simulated frequency and phase response of the filter for a center frequency of 50 MHz and a quality factor of 50. Figure 21.36 shows the simulated output of the filter when the input consists of two tones at 49.9 MHz and 50.1 MHz, respectively, each at 40 mVpp. At this level of input signal, the third-order intermodulation components were found to be at the same level as the noise. Thus, the predicted SFDR is about 34 dB with Q = 50. Table 21.5 summarizes the simulation results.

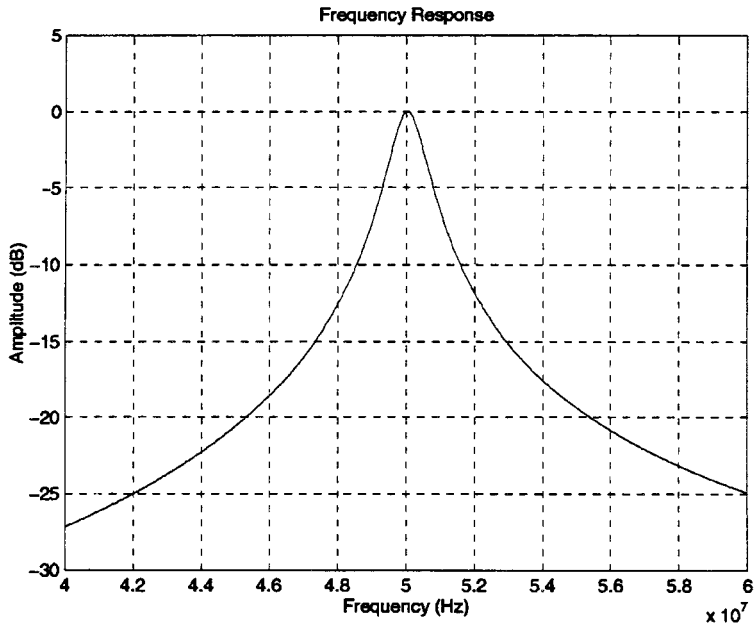


FIGURE 21.35 Simulated bandpass frequency response.

21.7 Log-Domain Processing

Instantaneous Companding

The concept of *instantaneous companding* is an emerging area of interest within the field of analog integrated circuit design. Currently, the main area of application for this technique is the implementation of continuous-time, fully integrated filters with wide dynamic range, high-frequency potential, and wide tunability.

With the drive toward lower supply voltages and higher operating frequencies, traditional analog integrated circuit design methodologies are proving inadequate. Conventional techniques to linearize inherently non-linear devices require an overhead in terms of increased power consumption or reduced operating speed. Recently, the use of companding, originally developed for audio transmission, has been

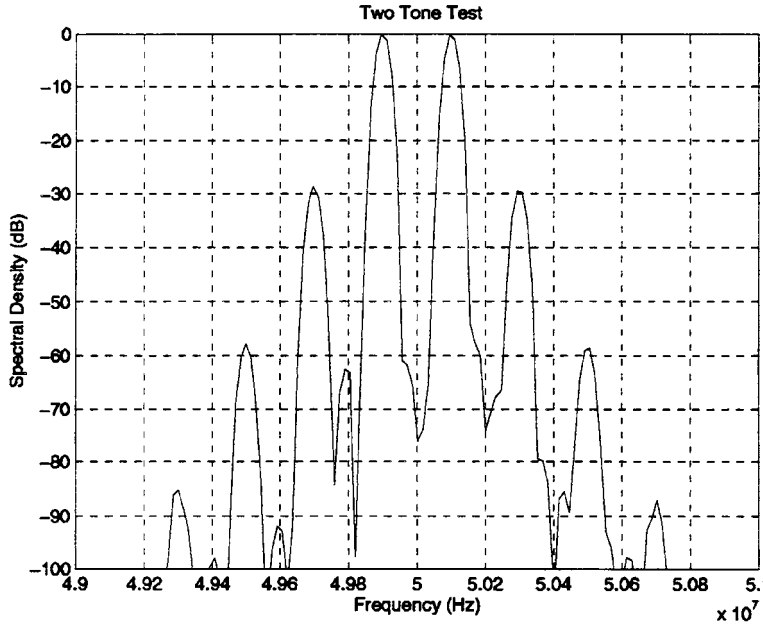


FIGURE 21.36 Simulated two-tone intermodulation test.

TABLE 21.5 Simulation Results

Power dissipation (Supply voltage = 5 V)	12.5 mW
Common-mode output offset	<1 mV
Center frequency	50 MHz
Quality factor	50
Output noise voltage (integrated over the band from 40 MHz to 60 MHz with Q = 50)	500 μV_{rms}
Output signal voltage (so that intermodulation components are at the same level as the noise, Q = 50)	25.2 mV_{rms}
Spurious free dynamic range (Q = 50)	34 dB

proposed as an elegant solution to the problem of maintaining dynamic range and high-frequency operation under low supply voltage.[50] In this approach, non-linear amplifiers are used to compress the dynamic range of the input signal to ensure, for example, that signal levels always remain above a certain noise threshold but below levels that may cause overload. The overall system must thus adapt its operation to ensure that input-output linearity is maintained as the instantaneous signal level alters. Although the system is input-output linear, signals internal to the system are inherently non-linear. Companding has traditionally been realized in two ways — syllabic and instantaneous — depending on how the system adapts in response to the changing input signal level. In *syllabic* companding, the level of compression (and expansion) is a non-linear function of a slowly varying property of the signal (e.g., envelope or power). In contrast, *instantaneous* companding adapts the compression and expansion ratios instantaneously with the changing input signal amplitude.

Perhaps the best-known recent example of this approach is the log-domain technique, where the exponential I–V characteristics of bipolar junction transistors (BJTs) are directly exploited to implement input-output linear filters. Since the large signal device equations are utilized, there is no need for small signal operation or local linearization techniques, and the resulting circuits have the potential for wide dynamic range and high-frequency operation under low power supply voltages. Log-domain circuits are generally implemented using BJTs and capacitors only and thus are inherently suitable for integration.

The filter parameters are easily tunable, which makes them robust. In addition, the design procedure is systematic, which suggests that these desirable features can be reproduced for different system implementations. The following section provides an introduction to the synthesis of log-domain filters and highlights various performance issues. Interested readers are advised to consult Refs. 62 through 77 for a more detailed treatment of the subject.

Log-Domain Filter Synthesis

The synthesis of log-domain filters using state-space transformations was originally proposed by Frey.[63] As an example of this methodology, consider a biquad filter with the following transfer function:

$$Y(s) = \frac{s\omega_o U_1(s) + \omega_o^{\,2} U_2(s)}{s^2 + (\omega_o/Q)s + \omega_o^{\,2}} \tag{21.51}$$

Thus, using u_1 as input gives a bandpass response at the output y, while using u_2 as input gives a lowpass response. This system can also be described by the following state space equations:

$$\begin{aligned} \dot{x}_1 &= -(\omega_o/Q)x_1 - \omega_o x_2 + \omega_o u_1 \\ \dot{x}_2 &= -\omega_o x_1 - \omega_o u_2 \\ y &= x_1 \end{aligned} \tag{21.52}$$

where a 'dot' denotes differentiation in time. To transform these linear state equations into non-linear nodal equations that can be directly implemented using bipolar transistors, the following exponential transformations are defined:

$$\begin{aligned} x_1 &= I_S \exp\left(\frac{V_1}{V_t}\right) \qquad & u_1 &= \left(\frac{I_S^{\,2}}{I_o}\right)\exp\left(\frac{V_{in1}}{V_t}\right) \\ x_2 &= I_o \exp\left(\frac{V_2}{V_t}\right) \qquad & u_2 &= I_S \exp\left(\frac{V_{in2}}{V_t}\right) \end{aligned} \tag{21.53}$$

These transformations map the linear state variables to currents flowing through bipolar transistors (BJTs) biased in the active region. I_S represents the BJT reverse saturation current, while V_t is the thermal voltage. Substituting these transformations into the state (Eq. 21.52) gives:

$$\begin{aligned} C\dot{V}_1 &= -\frac{I_o}{Q} - \frac{I_o^{\,2}}{I_S}\exp\left(\frac{V_2 - V_1}{V_t}\right) + I_S \exp\left(\frac{V_{o1} - V_1}{V_t}\right) \\ C\dot{V}_2 &= I_S \exp\left(\frac{V_1 - V_2}{V_t}\right) - I_S \exp\left(\frac{V_{o2} - V_2}{V_t}\right) \\ y &= I_S \exp\left(\frac{V_1}{V_t}\right) \end{aligned} \tag{21.54}$$

In Eq. 21.54, a tuning current $I_o = C\omega_o V_t$ is defined, where C is a scaling factor which represents a capacitance. The linear state space equations have thus been transformed into a set of non-linear nodal equations, and the task for the designer is now to realize a circuit architecture that will implement these non-linear equations. Considering the first two equations in Eq. 21.54, the terms on the LHS can be implemented as currents flowing through grounded capacitors of value C connected at nodes V_1 and V_2, respectively. The expressions on the RHS can be implemented by constant current sources in conjunction

with appropriately biased bipolar transistors to realize the exponential terms. The third equation in Eq. 21.54 indicates that the output y can be obtained as the collector current of a bipolar transistor biased with a base–emitter voltage V_1.

Figure 21.37 shows a possible circuit implementation, which is derived using Gilbert's translinear circuit principle.[64] The detailed circuit implementation is described in more detail in Ref. 63. The center frequency of this filter is given by $\omega_o = (I_o/CV_t)$ and can thus be tuned by varying the value of the bias current I_o.

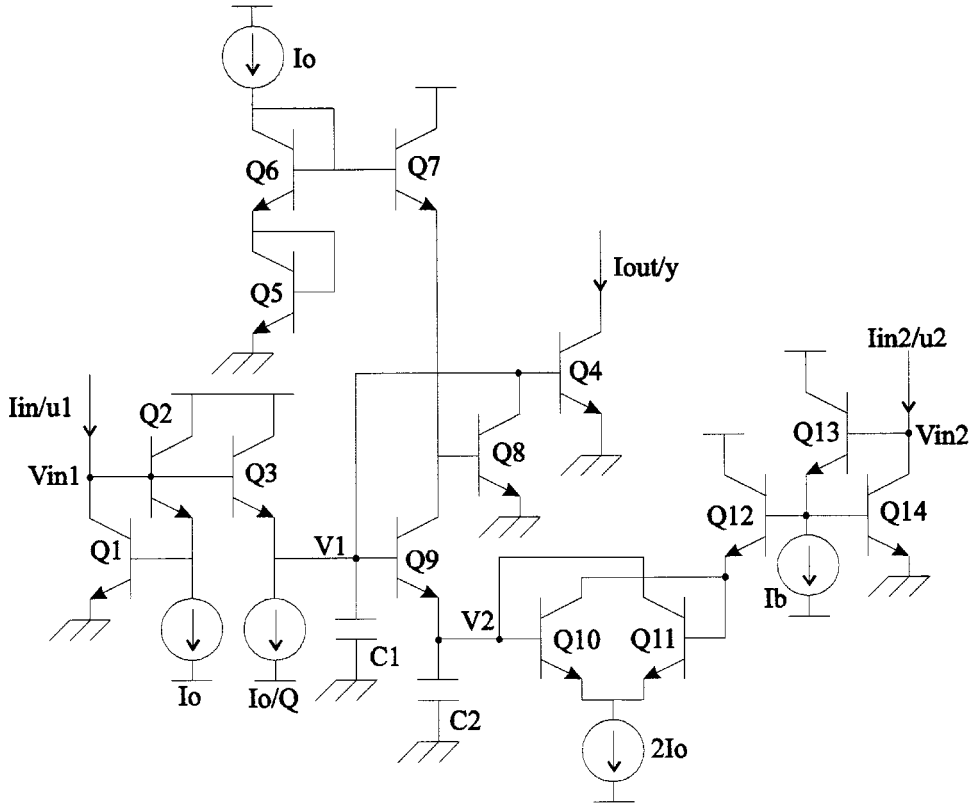


FIGURE 21.37 A log-domain biquadratic filter. A lowpass response is obtained by applying a signal at I_{u2} and keeping I_{u1} constant, while a bandpass response is obtained by applying a signal at I_{u1} and keeping I_{u2} constant.

Performance Aspects

Tuning Range

Both the quiescent bias current I_o and capacitance value C can be varied to alter the filter response. However, the allowable capacitor values are generally limited to within a certain range. On the lower side, C must not become smaller than the parasitic device capacitance. The base–emitter diffusion capacitance of the transistors is particularly important, since this is generally the largest device capacitance (up to the pF range) and is also non-linear. In addition, as the value of C decreases, it becomes more difficult to match capacitance values. The silicon area available limits the maximum value of C; for example, in a typical technology, a 50-pF poly-poly capacitor consumes 100 μm × 1000 μm of silicon area.

The range of allowable currents in a modern BJT is, in principle, fairly large (several decades); however, at very low and very high current levels, the current gain (β) is degraded. For high-frequency operation, particular attention must be paid to the actual f_t of the transistors, which is given by[6]:

$$f_t = \frac{g_m}{2\pi(C_\pi + C_\mu)} = \frac{g_m}{2\pi(C_d + C_{je} + C_{jc})} \tag{21.55}$$

C_{je} and C_{jc} represent the emitter and collector junction capacitance, respectively, and are only a weak function of the collector bias current. C_d represents the base–emitter diffusion capacitance and is proportional to the instantaneous collector current; $C_d = (\tau_f I_c/V_t)$, where τ_f is the effective base transit time. g_m represents the device transconductance, which is again dependent on the collector current; $g_m = (I_c/V_t)$. At high current levels, the diffusion capacitance is much larger than the junction capacitance, and thus:

$$f_t = \frac{g_m}{2\pi C_d} = \frac{1}{2\pi\tau_f} \tag{21.56}$$

At lower current levels, C_d and g_m decrease, whereas C_{je} and C_{jc} remain constant and f_t is reduced:

$$f_t = \frac{g_m}{2\pi(C_{je} + C_{jc})} \tag{21.57}$$

At very high current levels, f_t again reduces due to the effects of high-level injection.

Finite Current Gain

To verify the performance of the biquad filter, the circuit of Fig. 21.37 was simulated using HSPICE with transistor parameters from a typical bipolar process. Transient (large signal) simulations were carried out to confirm the circuit operation, and the lowpass characteristic (input at I_{u2}) is found to be very close to the ideal response. However, with an input signal applied at I_{u1} to obtain a bandpass response, the filter performance clearly deviates from the ideal characteristic as shown in Fig. 21.38. At low frequencies, the stop-band attenuation is only 25 dB. Re-simulating the circuit with ideal transistor models gives the required ideal bandpass characteristic as shown in Fig. 21.38; and by re-introducing the transistor parameters one at a time, the cause of the problem is found to be the finite current gain (β) of the bipolar transistors.

To increase the effective β, each transistor can be replaced by a Darlington pair, as shown in Fig. 21.39. This combination acts as a bipolar transistor with $\beta = \beta_a\beta_b + \beta_a + \beta_b \approx \beta^2$. Simulating the bandpass response of the circuit with Darlington pairs results in improved stopband attenuation as shown in Fig. 21.40, where the dc attenuation $H(0)$ is now approximately –50 dB. A disadvantage, however, is that a higher supply voltage is now required, since the effective base–emitter voltage V_{be} of the Darlington pair is double that of a single device. In addition, the current in device Q_a of Fig. 21.39 is now β times smaller than the design current I_o, resulting in a much lower f_t for this device.

An alternative method to improve the bandpass characteristic is described below. The (non-ideal) transfer function from the bandpass input I_{u1} is described by:

$$H_1(s) = \frac{I_{out}(s)}{I_{u1}(s)} = \frac{\omega_o(s + \omega_z)}{s^2 + (\omega_o/Q)s + \omega_o^2} \tag{21.58}$$

ω_z is a parasitic zero, which describes the low-frequency "flattening-out" of the bandpass characteristic. The transfer function from the second input I_{u2} is a lowpass response:

$$H_2(s) = \frac{I_{out}(s)}{I_{u2}(s)} = \frac{\omega_o^2}{s^2 + (\omega_o/Q)s + \omega_o^2} \tag{21.59}$$

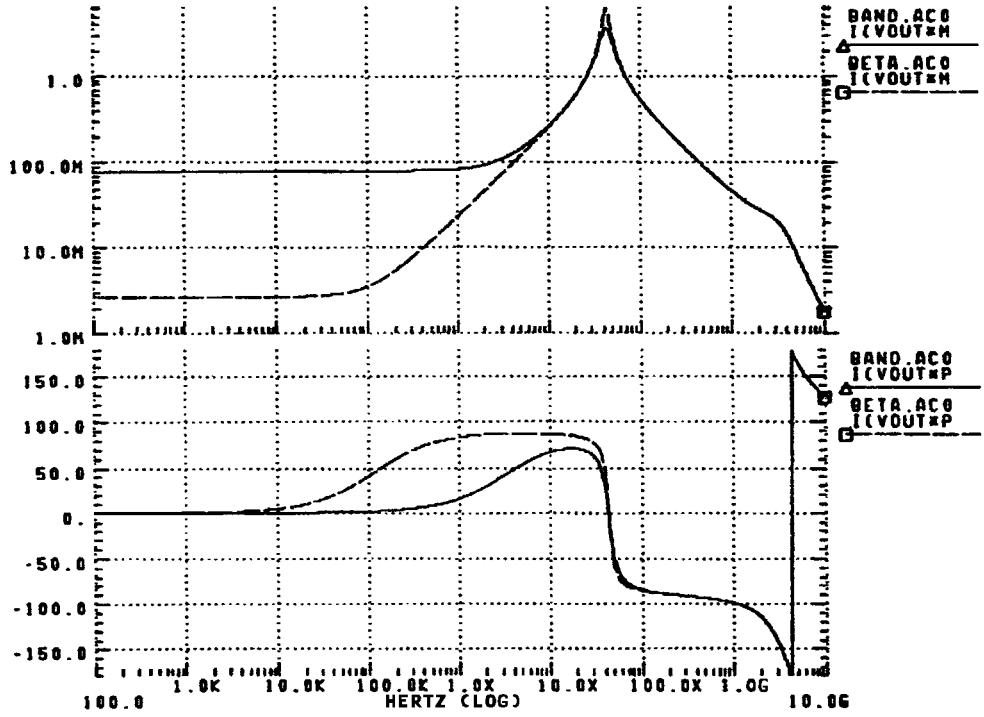


FIGURE 21.38 Bandpass response of the biquad of Fig. 21.37. Notice the poor low-frequency stop-band attenuation (approx. 25 dB) if real transistor models are used (solid line), versus the much better stop-band attenuation with β set to 1000 (dashed line).

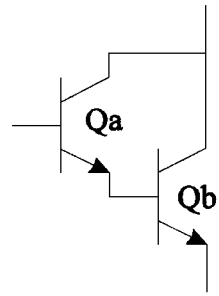


FIGURE 21.39 Darlington pair.

By applying a scaled version of the input signal I_{u1} to the lowpass input I_{u2}, the unwanted zero ω_z can thus be compensated. Setting $I_{u2} = -(\omega_z/\omega_o)\, I_{u1}$

$$I_{out}(s) = \frac{\omega_o(s+\omega_z)I_{u1}(s) + \omega_o^2\left(-\frac{\omega_z}{\omega_o}I_{u1}(s)\right)}{s^2 + (\omega_o/Q)s + \omega_o^2} = \frac{\omega_o s}{s^2 + (\omega_o/Q)s + \omega_o^2}I_{u1}(s) \tag{21.60}$$

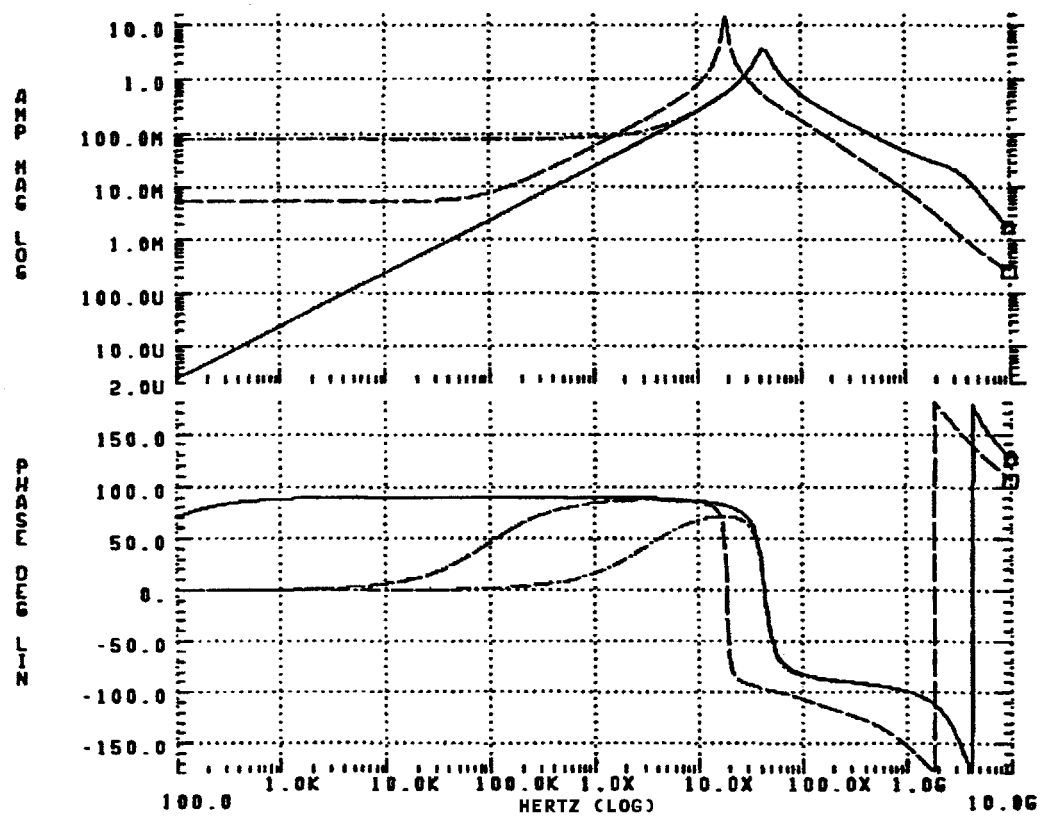


FIGURE 21.40 Possible solutions for the effect of finite β. The "dashed-dotted" line is the original response with a stopband attenuatin of only 25 dB. The dashed line (–50 dB) is the result of replacing all BJTs by Darlington pairs. The solid line is the result of feeding a fraction of the input signal to the second (i.e., lowpass) input.

This idea is confirmed by simulation as shown in Fig. 21.40. However, in practice, since this technique is very sensitive to the exact factor by which the input signal is scaled, active on-chip tuning would be required.

Frequency Performance

The log-domain filter operates in current mode, and all nodes within the circuit have an impedance of the order of $1/g_m$. Thus, any parasitic poles or zeros within the circuit are of the order of g_m/C_p, where C_p represents the parasitic capacitance at the given node. Typically, the parasitic capacitors are dominated by base–emitter diffusion capacitance Cπ, and the parasitic poles are situated close to the f_t of the technology. This underscores the potential of the log-domain technique for high-frequency operation. In practice, as with all high-frequency circuits, careful design and layout are required to achieve the maximum frequency potential.[76,77]

The Basic Log-Domain Integrator

The previous section has outlined some of the limitations of practical log-domain circuits that result from intrinsic device parasitics. This section analyzes these non-idealities in more detail by considering the simplest log-domain circuit, a first-order lowpass filter (lossy integrator).

A log-domain first-order lowpass filter is shown in Fig. 21.41. Referring to this circuit, and assuming an ideal exponential characteristic for each BJT, the following set of equations can be written (assuming matched components and neglecting the effect of base currents):

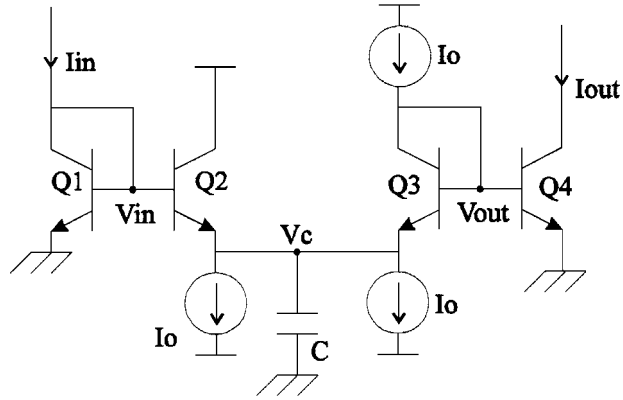


FIGURE 21.41 A first-order lowpass filter (lossy integrator).

$$\begin{aligned} I_{in} &= I_S \exp\left(\frac{V_{in}}{V_t}\right) & I_{out} &= I_S \exp\left(\frac{V_{out}}{V_t}\right) \\ V_{out} &= V_c + V_t \ln\left(\frac{I_o}{I_S}\right) & I_o + C\frac{dV_c}{dt} &= I_S \exp\left(\frac{V_{in} - V_c}{Vt}\right) \end{aligned} \tag{21.61}$$

Combining these equations results in the following linear first-order differential equation:

$$\left(\frac{CV_t}{I_o}\right)\frac{dI_{out}}{dt} + I_{out} = I_{in} \tag{21.62}$$

Taking the Laplace transform produces the following linear transfer function:

$$\frac{I_{out}(s)}{I_{in}(s)} = \frac{I_o}{I_o + sCV_t} \tag{21.63}$$

Equation 21.63 describes a first-order lowpass circuit with –3 dB frequency $f_c = I_o/2\pi CV_t$. This transfer function has been derived assuming an ideal exponential BJT characteristic; however, in practice, device non-idealities will introduce deviations from this ideal exponential characteristic, resulting in transfer function errors and distortion. This is similar to the case of "conventional" linear system design, where any deviation from the ideal linear building block response will contribute to output distortion. A brief discussion of the performance limitations of log-domain filters due to device non-idealities is given below; further discussion of distortion and performance limitations can be found in Refs. 66 and 67.

Effects of Finite Current Gain

Assuming all transistors have equal β, Eq. 21.62 is modified to the following non-linear differential equation (neglecting terms with β^2 or higher in the denominator):

$$\left(\frac{CV_t}{I_o}\right)\frac{dI_{out}}{dt}\left(1 + \frac{I_o}{I_{out}(1+\beta)}\right) + I_{out}\left(1 + \frac{I_{out}}{\beta I_o}\right) + \frac{2I_o}{1+\beta} = I_{in}\left(1 + \frac{I_{in}}{\beta}\right) \tag{21.64}$$

An analytical solution to Eq. 21.64 is difficult to obtain; thus, a qualitative discussion is given. At low frequencies, neglecting the differential terms, finite β causes a dc gain error and quadratic (even-order)

distortion. At high frequencies, the differential term is modified, depending on the values of I_{out} and β; therefore, scalar error and modulation of the 3 dB frequency are expected. In practice, the effects of finite β are further complicated due to its dependence on frequency and device collector current.

Component Mismatch

Emitter area mismatches cause variations in the saturation current I_S between transistors. Taking the emitter area into account, Eq. 21.62 can be rewritten as:

$$\left(\frac{CV_t}{I_o}\right)\frac{dI_{out}}{dt} + I_{out} = \lambda I_{in} \tag{21.65}$$

where $\lambda = (I_{S3}I_{S4}/I_{S1}I_{S2}) = (A_3A_4/A_1A_2)$. It is clear from Eq. 21.65 that area mismatches introduce only a change in the proportionality constant or dc gain of the integrator, and do not have any effect on the linearity or time constant of the integrator. The gain error can be compensated by easily adjusting one of the dc bias currents; thus, I_s mismatches do not seem to be a significant problem.

Ohmic Resistance

Ohmic resistances include the base and emitter diffusion resistance, and the resistance of interconnects and contact interfaces. For simplicity, all theses resistances can be referred to the base as an equivalent ohmic base resistance r_b. The device collector current can thus be defined as:

$$I_c = I_S\exp\left(\frac{V_{be} - I_b r_b}{V_t}\right) = I_S\exp\left(\frac{V_{be}}{V_t}\right)\exp(-\alpha I_c) \tag{21.66}$$

where V_{be} is the applied (extrinsic) base–emitter voltage and $\alpha = r_b/\beta V_t$. When the first-order filter of Fig. 21.41 is analyzed using the expression given in Eq. 21.66, a modified differential equation is obtained (assuming all base resistances are equal):

$$\left(\frac{CV_t}{I_o}\right)\frac{d[I_{out}e^{\alpha I_{out}}]}{dt} + I_{out}e^{\alpha I_{out}} = (e^{\partial\alpha I_o})I_{in}e^{\alpha I_{in}}e^{\left(-\alpha\frac{CV_t}{I_o}\frac{dI_{out}}{dt}\left(1+\alpha I_{out}\right)\right)} \tag{21.67}$$

The term $\partial\alpha$ represents the difference between two α values and will be close to zero if all base resistances are assumed equal, and thus can be neglected. At frequencies well below ω_o, the time derivative in the exponent can also be neglected to give:

$$\left(\frac{CV_t}{I_o}\right)\frac{d[I_{out}\exp(\alpha I_{out})]}{dt} + I_{out}\exp(\alpha I_{out}) = I_{in}\exp(\alpha I_{in}) \tag{21.68}$$

Expanding the differential term in Eq. 21.68:

$$I_{out}\exp(\alpha I_{out}) + \tau\frac{dI_{out}}{dt}\exp(\alpha I_{out}) + \tau\alpha I_{out}\frac{dI_{out}}{dt}\exp(\alpha I_{out}) = I_{in}\exp(\alpha I_{in}) \tag{21.69}$$

Equation 21.69 is clearly no longer linear; and thus, distortion products will be present at the output. To quantify this distortion, we apply as an input signal $I_{in} = A\exp(j\omega t)$, and expect as output:

$$I_{out} = B\exp j(\omega w + \theta) + c\exp j(2\omega t + \phi) + D\exp j(3\omega t + \psi) \tag{21.70}$$

Expanding the exponential terms in Eq. 21.69 by the first few terms of their series expansion:

$$\exp(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} \tag{21.71}$$

and identifying the terms in $\exp(j\omega t)$, $\exp(2j\omega t)$, etc. results in:

$$B\exp j\theta = \frac{A}{1+s\tau} \qquad C\exp j\phi = \alpha A^2 p(\tau) \qquad D\exp j\psi = \alpha^2 A^3 q(s\tau) \tag{21.72}$$

The expression for B is (as expected) the first-order transfer function of the system. The expressions for C and D give the second and third harmonic terms:

$$\mathrm{HD}_2 = \frac{|C\exp j\phi|}{|A|} \qquad \mathrm{HD}_3 = \frac{|D\exp j\psi|}{|A|} \tag{21.73}$$

$p(s\tau)$ and $q(s\tau)$ are rational functions in $s\tau$; thus, the output distortion is frequency dependent. Distortion is low at low frequencies, and peaks around the cut-off frequency ($s\tau = 1$), where $p = 0.25$ and $q = 2$. The maximum distortion levels can be approximated as:

$$\begin{aligned} \mathrm{HD}_{2(\max)} &= 0.25\alpha A = 0.25\frac{r_b}{\beta V_t} I_{in} \\ \mathrm{HD}_{3(\max)} &= 2(\alpha A)^2 = 2\left(\frac{r_b}{\beta V_t}\right)^2 I_{in}{}^2 \end{aligned} \tag{21.74}$$

These expressions demonstrate that when the voltage drop across the base resistance becomes comparable to V_t, there is significant distortion. This analysis also predicts that the distortion is larger at frequencies closer to the cut-off frequency, which is confirmed by circuit simulation.

In practice, the base resistance can be minimized by good layout (multiple base contacts), or by connecting several transistors in parallel. The latter, however, decreases the current through each transistor, which may lead to a decrease in f_t. It should be noted that, from a technological point of view, a high f_t and a low r_b are opposing goals. To achieve a high f_t, the base should be as shallow as possible, reducing base transit time. At the same time, however, a shallow base increases the silicon (intrinsic) base resistance.

Early Effect

The Early effect (base-width modulation) causes the collector current to vary with the collector–emitter and base–collector voltages. Considering the variation of collector–emitter voltage, the collector current can be written as $I_c = (1 + V_{ce}/V_A)\exp(V_{be}/V_t)$, where V_A is the forward-biased Early voltage. An analysis of the circuit of Fig. 21.41 shows that the Early effect introduces a scalar error to the dc gain as in the case of emitter area (I_S) mismatch. In practice, the base-width modulation of the devices also introduces distortion because V_{ce} and V_{bc} are signal dependent. However, since voltage swings in current-mode circuits are minimized, this is not believed to be a major source of distortion.

Frequency Limitations

Each bipolar transistor has various intrinsic capacitors, the most important being C_μ (base–collector junction capacitance), C_{cs} (collector–substrate junction capacitance), and C_π (the sum of the base–emitter junction capacitance and the base–emitter diffusion capacitance). The junction capacitors depend only slightly on the operating point, while the base–emitter diffusion capacitance is proportional to the bias current, as given by $C_d = \tau_f I_c/V_t$. To determine the position of the parasitic poles and zeros, Fig. 21.41 should be analyzed using large-signal device models. Unfortunately, the complexity of the resulting expressions renders this approach impractical even for the simplest log-domain circuits. To

gain an intuitive understanding of the high-frequency limitations of the circuit, a small-signal analysis can be performed. Although log-domain circuits are capable of large-signal operation, this small-signal approach is justified to some extent since small signals can be considered "special case" of large signals. If the circuit fails to operate correctly for small signals, then it will almost certainly fail in large-signal operation (unfortunately, the opposite does not hold; if a circuit operates correctly for small signals, it does not necessarily work well for large signals). Analyzing the circuit of Fig. 21.41, replacing each transistor by a small-signal (hybrid-π) equivalent model (comprising g_m, r_b, r_π, C_π, C_μ),[65] results in the following expression:

$$\frac{I_{out}}{I_{in}} = \left(\frac{g_{m4}}{g_{m1}}\right)\frac{1+\tau_z s}{(1+\tau_{p1}s)(1+\tau_{p2}s)} \tag{21.75}$$

τ_{p1} is the time constant of the first (dominant) pole, given by:

$$\tau_{p1} = \frac{C}{g_{m2}} + (C_{\mu4}+C_{\pi4})\left(r_{b4}+\frac{1}{g_{m2}}+\frac{1}{g_{m3}}\right)+\frac{C_{\pi1}}{g_{m1}}+\frac{C_{\pi2}}{g_{m2}} + \frac{C_{\pi3}}{g_{m3}} + C_{\mu1}r_{b1} + C_{\mu2}r_{b2} + C_{\mu3}r_{b3} \tag{21.76}$$

Ideally, this pole location should depend only on the design capacitance C, and not on the device parasitics. Therefore, $C_\mu r_b$, $C_\pi/g_m << C/g_{m2}$. Since $1/C_\mu r_b$ is typically much greater than f_t, this first constraint does not form a limit. The value of C_π/g_m depends on the operating point. For large currents, C_π is dominated by diffusion capacitance C_d so that $C_\pi/g_m = 1/f_t$. For smaller currents, g_m decreases while C_π becomes dominated by junction capacitance C_{je}, so that $C_\pi/g_m >> 1/f_t$. Thus, it would seem that the usable cut-off frequency of the basic log-domain first-order filter is limited by the actual f_t of the transistors. The second pole time constant τ_{p2} (assuming that $\tau_{p1} = C/g_{m2}$) is:

$$\tau_{p2} = (C_{\mu4}+C_{\pi4})\left(r_{b4}+\frac{1}{g_{m3}}\right)+(C_{\mu2}+C_{\pi2})\left(r_{b2}+\frac{1}{g_{m1}}\right)+\frac{C_{\pi1}+C_{cs1}}{g_{m1}} + \frac{C_{\pi13}+C_{cs3}}{g_{m3}} + C_{\mu1}r_{b1} + C_{\mu4}r_{b4} \tag{21.77}$$

This corresponds approximately to the f_t of the transistors, although Eq. 21.77 shows that the collector–substrate capacitance also contributes toward limiting the maximum operating frequency. The zero time constant τ_z is given by:

$$\tau_z = (C_{\pi1}+C_{\mu1})r_{b1} + \frac{C_{\pi2}}{g_{m2}} + \frac{C_{\pi4}}{g_{m4}} + C_{\mu4}r_{b4} \tag{21.78}$$

This is of the same order of magnitude as the second pole. This means that the first zero and the second pole will be close together, and will compensate to a certain degree. However, in reality, there are more poles and zeros than Eq. 21.75 would suggest, and it is likely that others will also occur around the actual f_t of the transistors.

Noise

Noise in companding and log-domain circuits is discussed in some detail in Refs. 69 to 71, and a complete treatment is beyond the scope of this discussion. For linear (non-companding) circuits, noise is generally assumed to be independent of signal level, and the signal-to-noise ratio (SNR) will increase with increasing input signal level. This is not true for log-domain systems. At small input signal levels, the noise value can be assumed approximately constant, and an increase in signal level will give an increase in

SNR. At high signal levels, the instantaneous value of noise will increase, and thus the SNR levels out at a constant value. This can be considered as an intermodulation of signal and noise power. For the Class A circuits discussed above, the peak signal level is limited by the dc bias current. In this case, the large-signal noise is found to be of the same order of magnitude as the quiescent noise level, and thus a linear approximation is generally acceptable (this is not the case for Class AB circuits).

Synthesis of Higher-Order Log-Domain Filters

The state-space synthesis technique outlined above proves difficult if implementation of high-order filters is required, since it becomes difficult to define and manipulate a large set of state equations. One solution is to use the signal flow graph (SFG) synthesis method proposed by Perry and Roberts[72] to simulate LC ladder filters using log-domain building blocks. The interested reader is also referred to Refs. 73 through 75, which present modular and transistor-level synthesis techniques that can be easily extended to higher-order filters.

References

1. A. Sedra and K. Smith, *Microelectronic Circuits*, Oxford, 1998.
2. P. R. Gray and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, Wiley, 1993.
3. W. H. Gross, New high speed amplifier designs, design techniques and layout problems, in *Analog Circuit Design*, Ed. J. S. Huijsing, R. J. van der Plassche, W. Sansen, Kluwer Academic, 1993.
4. D.F. Bowers, The impact of new architectures on the ubiquitous operational amplifier, in *Analog Circuit Design*, Ed. J. S. Huijsing, R. J. van der Plassche, W. Sansen, Kluwer Academic, 1993.
5. J. Fonderie and J. H. Huijsing, Design of low-voltage bipolar opamps, in *Analog Circuit Design*, Ed. J. S. Huijsing, R. J. van der Plassche, and W. Sansen, Kluwer Academic, 1993.
6. M. Steyaert and W. Sansen, Opamp design towards maximum gain-bandwidth, in *Analog Circuit Design*, Ed. J. S. Huijsing, R. J. van der Plassche, W. Sansen, Kluwer Academic, 1993.
7. K. Bult and G. Geelen, The CMOS gain-boosting technique, in *Analog Circuit Design*, Ed. J. S. Huijsing, R. J. van der Plassche, W. Sansen, Kluwer Academic, 1993.
8. J. Bales, A low power, high-speed, current feedback opamp with a novel class AB high current output stage, *IEEE Journal Solid-State Circuits*, vol. 32, no. 9, Sep. 1997, p. 1470.
9. C. Toumazou, Analogue signal processing: the 'current way' of thinking, *Int. Journal of High-Speed Electronics*, vol. 32, no. 3-4, p. 297, 1992.
10. K. Manetakis and C. Toumazou, A new CMOS CFOA suitable for VLSI technology, *Electron. Letters*, vol. 32, no. 12, June 1996.
11. K. Manetakis, C. Toumazou, and C. Papavassiliou, A 120MHz, 12mW CMOS current feedback opamp, *Proc. of IEEE Custom Int. Circuits Conf.*, p. 365, 1998.
12. D. A. Johns and K. Martin, *Analog Integrated Circuit Design*, Wiley, 1997.
13. C. Toumazou, J. Lidgey, and A. Payne, Emerging techniques for high-frequency BJT amplifier design: a current-mode perspective, *Parchment Press for Int. Conf. on Electron. Circuits Syst.*, Cairo, 1994.
14. M.C.H Cheng and C. Toumazou, 3V MOS current conveyor cell for VLSI technology, *Electron. Lett.*, vol. 29, p. 317, 1993.
15. K. Manetakis, Intermediate frequency CMOS analogue cells for wireless communications, Ph.D. thesis, Imperial College, London, 1998.
16. R. A. Johnson et al., A 2.4GHz silicon-on-sapphire CMOS low-noise amplifier, *IEEE Microwave and Guided Wave Lett.*, vol. 7, no. 10, pp. 350-352, Oct. 1997.
17. A. N. Karanicolas, A 2.7V 900MHz CMOS LNA and mixer, *IEEE Digest of I.S.S.C.C.*, pp. 50-51, 1996.
18. D. K. Shaffer and T. H. Lee, A 1.5-V, 1.5-GHz CMOS low noise amplifier, *IEEE J.S.S.C.*, vol. 32, no. 5, pp. 745-759, May 1997.

19. J. C. Rudell et al., A 1.9GHz wide-band if double conversion CMOS integrated receiver for cordless telephone applications, *Digest of IEEE I.S.S.C.C.*, pp. 304-305, 1997.
20. E. Abou-Allam et al., CMOS front end RF amplifier with on-chip tuning, *Proc. of IEEE ISCAS96*, pp. 148-151, 1996.
21. P. R. Gray and R. G. Meyer, *Analysis and Design of Analogue Integrated Circuits and Systems*, Chap. 11, 3rd ed., John Wiley & Sons, New York, 1993.
22. M. J. N. Sibley, *Optical Communications*, Chap. 4-6, Macmillan, 1995.
23. J. J. Morikuni et al., Improvements to the standard theory for photoreceiver noise, *J. Lightwave Tech.*, vol. 12, no. 4, pp. 1174-1184, Jul. 1994.
24. A. A. Abidi, Gigahertz transresistance amplifiers in fine line NMOS, *IEEE J.S.S.C.*, vol. SC-19, no. 6, pp. 986-994, Dec. 1984.
25. M. B. Das, J. Chen, and E. John, Designing optoelectronic integrated circuit (OEIC) receivers for high sensitivity and maximally flat frequency response, *J. of Lightwave Tech.*, vol. 13, no. 9, pp. 1876-1884, Sep. 1995.
26. B. Sklar, *Digital Communication: Fundamentals and Applications*, Prentice-Hall 1988.
27. S. D. Personick, Receiver design for optical fiber systems, *IEEE Proc.*, vol. 65, no. 12, pp. 1670-1678, Dec. 1977.
28. J. M. Senior, *Optical Fiber Communications: Principles and Practice*, Chap. 8-10, PHI, 1985.
29. N. Scheinberg et al, Monolithic GaAs transimpedance amplifiers for fiber-optic receivers, *IEEE J.S.C.C.*, vol. 26, no. 12, pp. 1834-1839, Dec. 1991.
30. C. Toumazou and S. M. Park, Wide-band low noise CMOS transimpedance amplifier for gigahertz operation, *Electron. Lett.*, vol. 32, no. 13, pp. 1194-1196, Jun. 1996.
31. S. M. Park and C. Toumazou, Giga-hertz low noise CMOS transimpedance amplifier, *Proc. IEEE ISCAS*, vol. 1, pp. 209-212, June 1997.
32. D. M. Pietruszynski et al, A 50-Mbit/s CMOS monolithic optical receiver, *IEEE J.S.S.C.*, vol. 23, no. 6, pp. 1426-1432, Dec. 1988.
33. Y. Akazawa et al., A design and packaging technique for a high-gain, gigahertz-band single-chip amplifier, *IEEE J.S.S.C.*, vol. SC-21, no. 3, pp. 417-423, Jun. 1986.
34. N. Ishihara et. al., A Design technique for a high-gain, 10-GHz class-bandwidth GaAs MESFET amplifier IC module, *IEEE J.S.S.C.*, vol. 27, no. 4, pp. 554-561, Apr. 1992.
35. M. Lee and M. A. Brooke, Design, fabrication, and test of a 125Mb/s transimpedance amplifier using MOSIS 1.2 μm standard digital CMOS process, *Proc. 37th Midwest Sym., Cir. and Sys.*, vol. 1, pp. 155-157, Aug. 1994.
36. R. P. Jindal, Gigahertz-band high-gain low-noise AGC amplifiers in fine-line NMOS, *IEEE J.S.S.C.*, vol. SC-22, no. 4, pp. 512-520, Aug. 1987.
37. N. Takachio et al., A 10Gb/s optical heterodyne detection experiment using a 23GHz bandwidth balanced receiver, *IEEE Trans. M.T.T.*, vol. 38, no. 12, pp. 1900-1904, Dec. 1990.
38. K. Katsura et al., A novel flip-chip interconnection technique using solder bumps for high-speed photoreceivers, *J. Lightwave Tech.*, vol. 8, no. 9, pp. 1323-1326, Sep. 1990.
39. K. Murota and K. Hirade, GMSK modulation for digital mobile radio telephony, *IEEE Trans. Commun.*, vol. 29, pp. 1044-1050, 1981.
40. H. Krauss, C. W. Bostian, and F. H. Raab, *Solid State Radio Engineering*, New York, Wiley, 1980.
41. A. S. Sedra and K. C. Smith, *Microelectronic Circuits*, 4th Edition 1998.
42. N. O. Sokal and A. D. Sokal, Class E, A new class of high efficiency tuned single-ended switching power amplifiers, *IEEE Journal of Solid-State Circuits*, vol. SC-10, pp. 168-176, June 1975.
43. F. H. Raab, Effects of circuit variations on the class E tuned power amplifier, *IEEE Journal of Solid-State Circuits*, vol. SC-13, pp. 239-247, 1978.
44. T. Sowlati, C. A. T. Salama, J. Sitch, G. Robjohn, and D. Smith, Low voltage, high efficiency class E GaAs power amplifiers for mobile communications, in *IEEE GaAs IC Symp. Tech. Dig.*, pp. 171-174, 1994.

45. _____ Low voltage, high efficiency GaAs class E power amplifiers for wireless transmitters, *IEEE Journal of Solid-State Circuits*, vol. SC-13, no. 10, pp. 1074-1080, 1995.
46. A. Rofougaran et al., A single-chip 900 MHz spread-spectrum wireless transceiver in 1-μm CMOS. part I: architecture and transmitter design, *IEEE Journal of Solid-State Circuits*, vol. SC-33, no. 4, pp. 515-534.
47. J. Chang, A. A. Abidi, and M. Gaitan, Large suspended inductors on silicon and their use in a 2-μm CMOS RF amplifier, *IEEE Electron Device Letters*, vol. 14, no. 5, May 1993.
48. T. Sowlati et al., Linearized high efficiency class E power amplifier for wireless communications, *IEEE Custom Integrated Circuits Conf. Proc.*, pp. 201-204, 1996.
49. G. N. Henderson, M. F. OKeefe, T. E. Boless, P. Noonan, et al., SiGe bipolar junction transistors for microwave power applications, *IEEE MTT-S Int. Microwave Symp. Dig.*, pp. 1299-1302, 1997
50. O. Shoaei and W. M. Snelgrove, A wide-range tunable 25MHz-110MHz BiCMOS continuous-time filter, *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),* Atlanta, 1996.
51. P.-H. Lu, C.-Y. Wu, and M.-K. Tsai, Design techniques for VHF/UHF high-Q tunable bandpass filters using simple CMOS inverter-based transresistance amplifiers, *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, May 1996.
52. Y. Tsividis, Integrated continuous-time filter design - an overview, *IEEE Journal of Solid-State Circuits*, vol. 29, no. 3, Mar. 1994.
53. F. Rezzi, A. Baschirotto, and R. Castello, A 3V 12-55MHz BiCMOS Pseudo-Differential Continuous-Time Filter, *IEEE Trans. on Circuits and Systems-I*, vol. 42, no. 11, Nov. 1995.
54. B. Nauta, *Analog CMOS Filters for Very High Frequencies*, Kluwer Academic Publishers.
55. C. Toumazou, F. Lidgey, and D. Haigh, *Analogue IC Design: The Current-Mode Approach*, Peter Peregrinus Ltd. for IEEE Press, 1990.
56. S. Szczepanski and R. Schauman, Nonlinearity-induced distortion of the transfer function shape in high-order filters, *Kluwer Journal of Analog Int. Circuits and Signal Processing*, vol. 3, p. 143-151, 1993.
57. S. Szczepanski, VHF fully-differential linearized CMOS transconductance element and its applications, *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS),* London 1994.
58. A. A. Abidi, Noise in active resonators and the available dynamic range, *IEEE Trans. on Circuits and Systems-I*, vol. 39, no. 4, Apr. 1992.
59. S. Pipilos and Y. Tsividis, RLC active filters with electronically tunable center frequency and quality factor, *Electron. Letters*, vol. 30, no. 6, Mar. 1994.
60. K. Manetakis, Intermediate frequency CMOS analogue cells for wireless communications, Ph.D. thesis, Imperial College, London, 1998.
61. K. Manetakis and C. Toumazou, A 50MHz high-Q bandpass CMOS filter, *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Hong-Kong, 1997.
62. Y. Tsividis, Externally linear time invariant systems and their application to companding signal processors, *IEEE Trans. CAS-II*, vol. 44(2), pp. 65-85, Feb.1997.
63. D. Frey, Log-domain filtering: an approach to current-mode filtering, *IEE Proc.-G*, vol. 140, pp. 406-416, 1993.
64. B. Gilbert, Translinear circuits: a proposed classification, 1975, *Electron. Lett.*, vol. 11, no. 1, pp. 14-16, 1975.
65. P. Grey, and R. Meyer, *Analysis and Design of Analog Integrated Circuits*, John Wiley & Sons Inc., New York, 3rd Edition, 1993.
66. E. M. Drakakis, A. Payne, and C. Toumazou, Log-domain state-space: a systematic transistor-level approach for log-domain filtering, accepted for publication in *IEEE Trans. CAS-II,* 1998.
67. V. Leung, M. El-Gamal, and G. Roberts, Effects of transistor non-idealities on log-domain filters, *Proc. IEEE Int. Symp. Circuits Syst.*, Hong-Kong, pp. 109-112, 1997.
68. D. Perry and G. Roberts, Log domain filters based on LC-ladder synthesis, *Proc. 1997 IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, pp. 311-314, Seattle, 1995.

69. J. Mulder, M. Kouwenhoven, and A. van Roermund, Signal × noise intermodulation in translinear filters, *Electron. Lett*, vol. 33(14), pp. 1205-1207.
70. M. Punzenberger and C. Enz, Noise in instantaneous companding filters, *Proc. 1997 IEEE Int. Symp. Circuits Syst.*, Hong Kong, pp. 337-340, June 1997.
71. M. Punzenberger and C. Enz, A 1.2V low-power BiCMOS class-AB log-domain filter, *IEEE J. Solid-State Circuits*, vol. SC-32(12), pp. 1968-1978, Dec. 1997.
72. D. Perry and G. Roberts, Log-domain filters based on LC ladder synthesis, *Proc. 1995 IEEE Int. Symp. Circuits Syst.*, Seattle, pp. 311-314, 1995.
73. E. Drakakis, A. Payne, and C. Toumazou, Bernoulli operator: a low-level approach to log-domain processing, *Electron. Lett.*, vol. 33(12), pp. 1008-1009, 1997.
74. F. Yang, C. Enz, and G. Ruymbeke, Design of low-power and low-voltage log-domain filters, *Proc. 1996 IEEE Int. Symp. Circuits Syst.*, Atlanta, pp. 125-128, 1996.
75. J. Mahattanakul and C. Toumazou, Modular log-domain filters, *Electron. Lett.*, vol. 33(12), pp. 1130-1131, 1997.
76. D. Frey, A 3.3 V electronically tuneable active filter useable to beyond 1 GHz, *Proc. 1994 IEEE Int. Symp. Circuits Syst.*, London, pp. 493-496, 1994.
77. M. El-Gamal, V. Leung, and G. Roberts, Balanced log-domain filters for VHF applications, *Proc. 1997 IEEE Int. Symp. Circuits Syst.*, Monterey, pp. 493-496, 1997.

Wassenaar, R.F., et al."Operational Transconductance Amplifiers"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

22

Operational Transconductance Amplifiers

R.F. Wassenaar
University of Twente

Mohammed Ismail
The Ohio State University

Chi-Hung Lin
The Ohio State University

22.1 Introduction

In many analog or mixed analog/digital VLSI applications, an operational amplifier may not be appropriate to use for an active element. For example, when designing integrated high-frequency active filter circuitry, a much simpler building block, called an operational transconductance amplifier (OTA), is often used.[1] This type of amplifier is characterized as a voltage-driven current source and in its simplest form is a combination of a differential input pair with a current mirror as shown in Fig. 22.1. It is a simple circuit with a relatively small chip area. Further, it has a high bandwidth and also a good common-mode rejection ratio up to very high frequencies. The small signal transconductance, $g_m = \partial I_{out}/\partial V_{in}$, can be controlled by the tail current. This chapter discusses CMOS OTA design for modern VLSI applications. We begin the chapter with a brief study of noise in OTAs, followed by OTA design techniques.

22.2 Noise Behavior of the OTA

The noise behavior of the OTA is discussed here. Attention will be paid to thermal and flicker noise and to the fact that, for minimal noise, some voltage gain, from the input of the differential pair to the input of the current mirror, is required. Then, only the noise of the input pair becomes dominant and the other noise sources can be neglected to first order. The noise behavior of a single MOS transistor is modeled by a single noise voltage source. This noise voltage source is placed in series with the input (gate) of a "noiseless" transistor. Fig. 22.2(a) shows the simple OTA, including the noise sources, while Fig. 22.2(b) shows the same circuit with all the noise referred to the input of the stage.

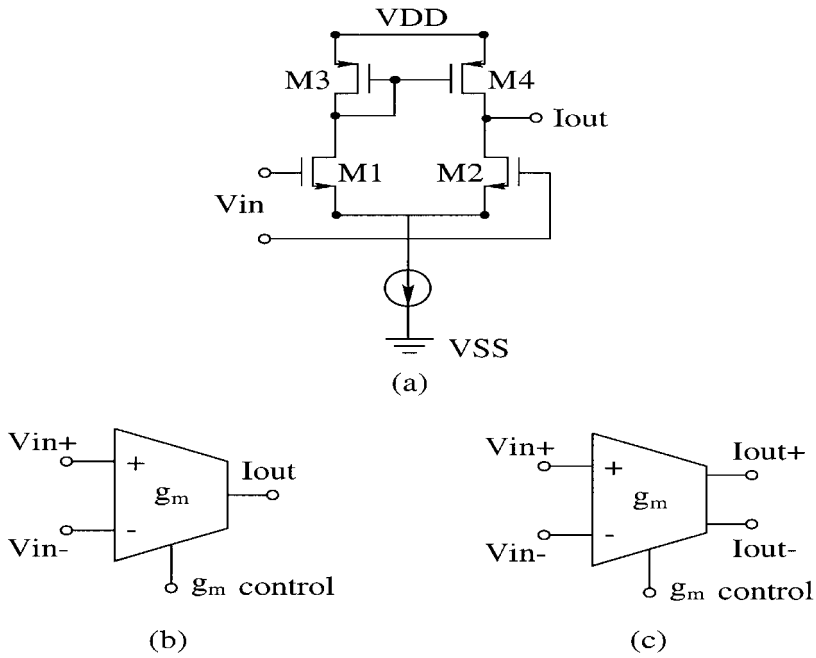


FIGURE 22.1 (a) A NMOS differential pair with a PMOS current mirror forming an OTA; (b) the symbol for a single-ended OTA; and (c) the symbol for a fully differential OTA.

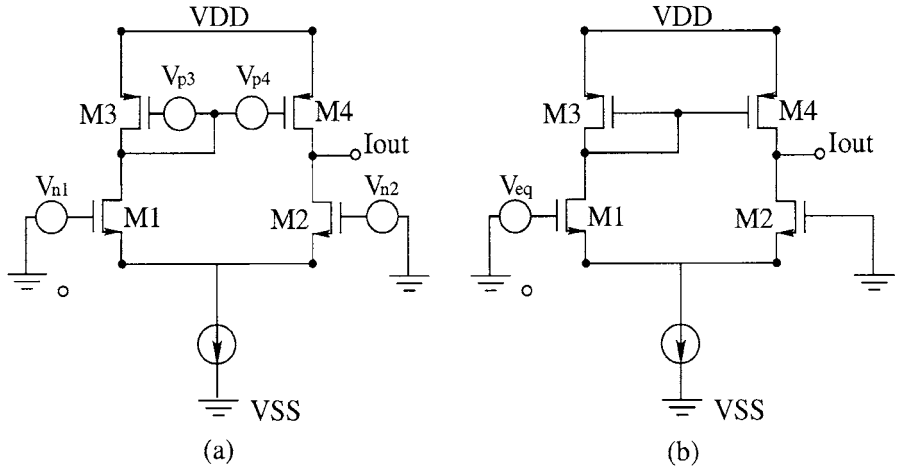


FIGURE 22.2 (a) The OTA with its noise voltage sources, and (b) the same circuit with the noise voltage sources referred to one of the input nodes.

All the noise sources indicated in Fig. 22.2(a) are converted to equivalent input noise voltags, which are then added to form a single noise source at the input (Fig. 22.2(b)). As a result, we obtain (assuming $g_{m1} = g_{m2}$ and $g_{m3} = g_{m4}$) the following mean-square input referred noise voltage

$$\overline{V_{eq}^2} = \overline{V_{n1}^2} + \overline{V_{n2}^2} + \left(\frac{g_{m3}}{g_{m1}}\right)^2 (\overline{V_{p3}^2} + \overline{V_{p4}^2}) \tag{22.1}$$

The thermal noise contribution of one transistor, over a band Δf, is written as:

$$\overline{V_{th}^2} = \frac{2}{3}4kT\frac{1}{g_m}\Delta f \tag{22.2}$$

where k is the Boltzman constant and T is the absolute temperature.

The equivalent noise voltage $\overline{V_{th_{eq}}^2}$ becomes:

$$\overline{V_{th_{eq}}^2} = \frac{2}{3}4kT\left(\frac{1}{g_{m1}} + \frac{1}{g_{m2}} + \left(\frac{g_{m3}}{g_{m1}}\right)^2\right)\left(\frac{1}{g_{m3}} + \frac{1}{g_{m4}}\right)\Delta f \tag{22.3}$$

and because $g_{m1} = g_{m2}$ and $g_{m3} = g_{m4}$, $\overline{V_{th_{eq}}^2}$ becomes:

$$\overline{V_{th_{eq}}^2} = \frac{16}{3}kT\left(\frac{1}{g_{m1}} + \left(\frac{g_{m3}}{g_{m1}}\right)^2\frac{1}{g_{m3}}\right)\Delta f \tag{22.4}$$

or

$$V_{th_{eq}}^2 = \frac{16}{3}\frac{kT}{g_{m1}}\left(1 + \frac{g_{m3}}{g_{m1}}\right)\Delta f \tag{22.5}$$

Expressing g_m in physical parameters results in:

$$\overline{V_{th_{eq}}^2} = \frac{16kT}{3\sqrt{\mu_n C_{ox}(W/L)_1 I_0}}\left(1 + \sqrt{\frac{\mu_p(W/L)_3}{\mu_n(W/L)_1}}\right)\Delta f \tag{22.6}$$

In this equation, I_0 represents the tail current of the differential pair. Note that the term between brackets represents the relative noise contribution of the current mirror. This term can be neglected if M_3 and M_4 are chosen relatively long and narrow in comparison to M_1 and M_2.

It should be mentioned that the thermal noise of an N-MOS transistor and a P-MOS transistor with equal transconductance is the same. In most standard IC processes, a three to ten times lower $1/f$ noise is observed for P-MOS transistors in comparison to N-MOS transistors of the same size. However, in modern processes, the $1/f$ noise contribution of N- and P-MOS transistors tends to be equal.

For the $1/f$ noise, it is usually assumed for standard IC processes that:

$$\overline{V_{1/f}^2} = \frac{K'}{WLC_{ox}f}\Delta f \tag{22.7}$$

where K' is the flicker noise coefficient in the range of 10^{-24} J for N-MOS transistors and in the range of 3×10^{-25} to 10^{-25} J for P-MOS transistors. The equivalent $1/f$ input noise source of the OTA in Fig. 22.2(b) yields:

$$\overline{V_{eq(1/f)}^2} = \frac{2K'_n\Delta f}{W_1L_1C_{ox}f}\left(1 + \frac{K'_p\mu_pL_1^2}{K'_n\mu_nL_3^2}\right) \tag{22.8}$$

Here, the noise contributions of the current mirror (M_3, M_4) will be negligible if L_3 is chosen much larger than L_1.

The offset voltage of a differential pair is lowest when the transistors are in the weak-inversion mode; but on the contrary, the mismatch in the current transfer of a current mirror is lowest when the transistors are deep in strong inversion. Hence, the conditions that have to be fulfilled for both minimal equivalent input noise and minimal offset are easy to combine.

22.3 An OTA with an Improved Output Swing

A CMOS OTA with an output swing much higher than that in Fig. 22.1(a) is shown in Fig. 22.3. This configuration needs two extra current mirrors and consumes more current, but the output voltage "window" is, in the case when common-mode input voltage is zero, about doubled. The rules discussed earlier for sizing the input transistors and current-mirror transistors to reduce noise and offset still apply. However, there is still a tradeoff. On the one hand, a high voltage gain from the input nodes to the current mirror is good for reducing noise and mismatch effects; on the other hand, too much gain also reduces the upper limit of the common-mode input voltage range and the phase margin needed to ensure stability (this will be discussed later).[2] A voltage gain on the order of 3 to 10 is advised. The frequency behavior of the OTA in Fig. 22.3 is rather complex since there are two different signal paths in parallel, as shown in Fig. 22.4. In this scheme, r_p represents the parallel value of the output resistance of the stage ($r_{o6} \| r_{o8}$) and the load resistance (R_L); therefore,

$$r_p = r_{o6} \| r_{o8} \| R_L \tag{22.9}$$

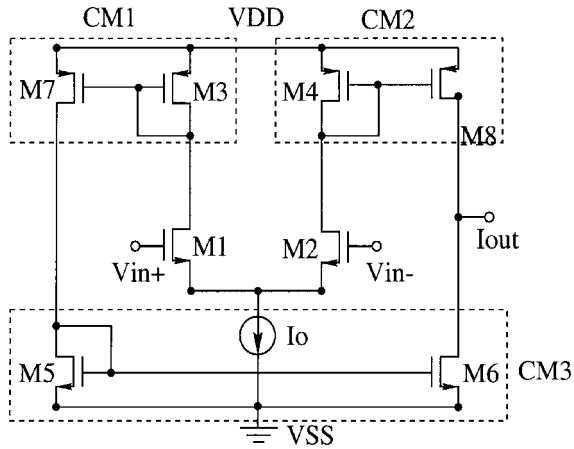


FIGURE 22.3 An OTA with an improved output window.

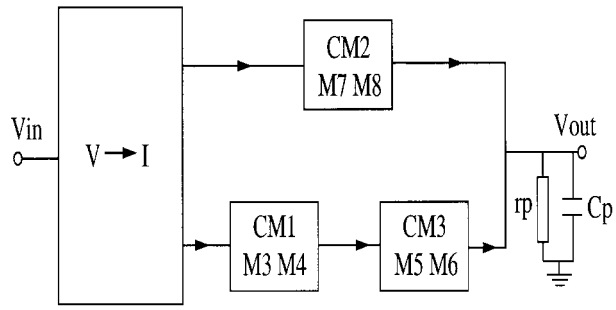


FIGURE 22.4 The signal paths of the OTA in Fig. 22.3.

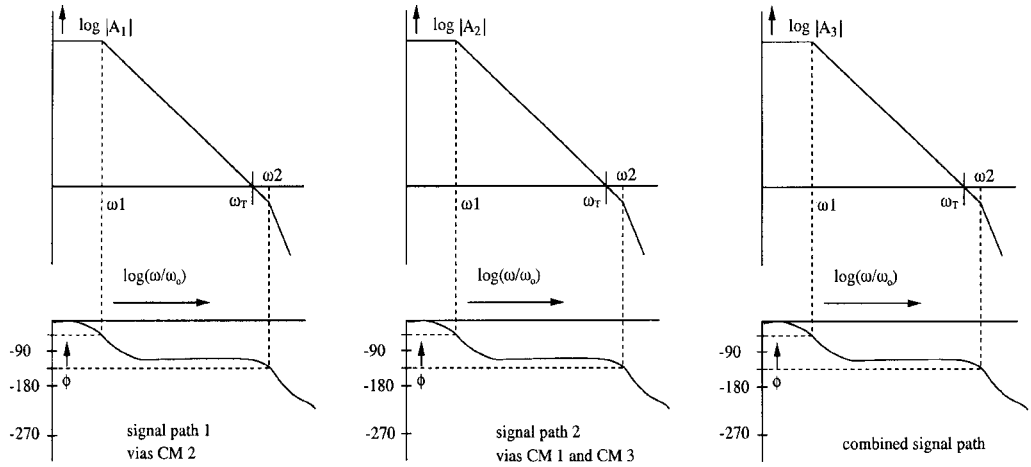


FIGURE 22.5 (a) The Bode plot belonging to signal path 1 in the OTA in Fig. 22.3 and 22.4, (b) signal path 2, and (c) to the combined signal path.

The capacitor C_p represents the sum of the parasitic output capacitance and the load capacitance $C_p = C_o + C_L$. Using the half-circuit principle for the differential pair, a fast signal path can be seen from M_2 via current mirror M_7, M_8 to the output. This signal path contributes an extra high-frequency pole. The other signal path leads from transistor M_1 via both current mirrors M_3, M_4 and M_5, M_6 to the output. In this path, two extra poles are added. The transfer of both signal paths and their combination are shown in the plots in Fig. 22.5, assuming equal pole positions of all three current mirrors. Note that the first (dominant) pole (ω_1) is determined by r_p and C_p.

$$\omega_1 = \frac{1}{r_p C_p} \tag{22.10}$$

The second pole (ω_2) is determined by the transconductance of M_3 and the sum of the gate-source capacitance of M_3 and M_4. If M_3 and M_4 are equal, the second pole is located at:

$$\omega_2 = \frac{g_{m3}}{2C_{gs3}} \tag{22.11}$$

The unity-gain corner frequency ω_T of the loaded OTA is at:

$$\omega_T = \frac{g_{m1}}{C_p} \tag{22.12}$$

Therefore, the ratio ω_2/ω_T is:

$$\frac{\omega_2}{\omega_T} = \frac{g_{m3}}{g_{m1}} \frac{C_p}{2C_{gs3}} \tag{22.13}$$

When the OTA is used for high-frequency filter design, an integrator behavior is required, that is, a constant 90° phase at least at frequencies around ω_T. Therefore, a high value of the ratio ω_2/ω_T is needed in order to have as little influence as possible from the second pole. It is obvious from Eq. 22.12 that the

low-frequency voltabe gain from the input nodes of the circuit to the input of the current mirrors (= g_{m1}/g_{m3}) must not be chosen too high. As mentioned, this is in contrast to the requirements for minimum noise and offset.

Sometimes, OTAs are used as unity gain voltage buffers; for example, in switched capacitor filters. In this case, the emphasis is put more on obtaining high open-loop voltage gain, improved output window, and good capability to drive capacitive loads efficiently (or small resistors); its integrator behavior is of less importance.

To increase the unloaded voltage gain, cascode transistors can be added in the output stage. This greatly increases the output impedance of the OTA and hardly decreases the phase margin. The penalty that has to be paid is an additional pole in the signal path and some reduction of the maximum possible output swing. This reduction can be very small if the cascode transistors are biased on the weak-inversion mode. The open-loop voltage gain can be in the order of 40 to 60 dB. A possible realization of such a configuration is shown in Fig. 22.6.[3]

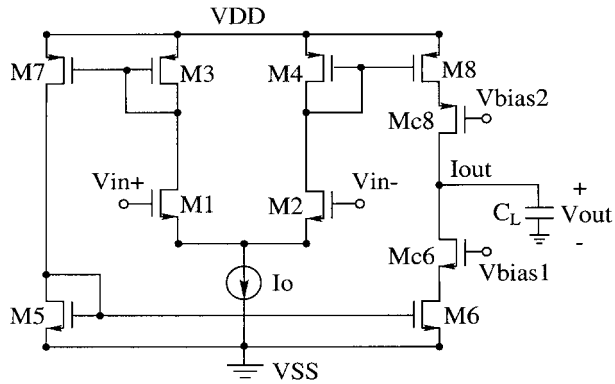


FIGURE 22.6 An OTA with improved output impedance.

22.4 OTAs with High Drive Capability

For driving capacitive loads (or small resistors), a large available output current is necessary. In the OTAs shown so far, the amount of output current available is equal to twice the quiescent current (i.e., the tail current I_0). In some situations, this current can be too small. There are several ways to increase the available current in an efficient way. To achieve this, four design principles will be discussed here:

1. Increasing the quiescent current by using current mirrors with a current transfer ratio greater than 1
2. Using a two-transistor level structure to drive the output transistors
3. Adaptive biasing techniques
4. Class AB techniques

OTAs with 1:B Current Mirrors

One way to increase available output current is to increase the transfer ratio of the current mirrors CM1 and CM2 by a factor B, as indicated in Fig. 22.7.[4] The amount of available output current and also the overall transconductance increase by the same factor. Unfortunately, the –3 dB frequency of the CM1-CM2 current mirrors will be reduced by a factor $(B + 1)/2$ due to the larger gate-source capacitance of the mirror output transistors. Moreover, ω_T will increase, ω_2 will decrease, and the ratio ω_2/ω_T will be

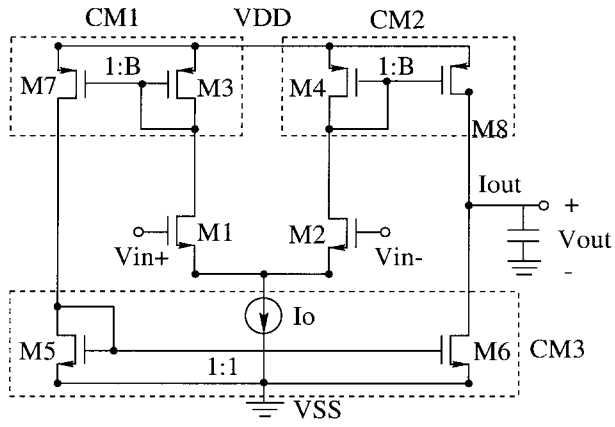


FIGURE 22.7 An OTA with improved load current using 1:B current mirrors.

strongly deteriorated. The amount of available output current though is B times the tail current. It is also possible to increase the current transfer ratio of current mirror CM3 instead of CM1. A better current efficiency then results, but at the expense of more asymmetry in the two signal paths. Although the amount of the maximum available output current is B times the tail current in both situations, the ratio between the maximum available current and quiescent current of the output stage remains equal to two, just as in the OTAs discussed previously.

OTA with Improved Output Stage

Another way of increasing the maximal available output current is illustrated in Fig. 22.8.[6] It improves upon the factor-two relationship between quiescent and maximal available current. Assuming equal K factors for all transistors shown in the circuit leads to the conclusion that the effective gate-source voltage of transistor M_{11} ($= V_{GS11} - V_{T11}$) equals that of transistor M_1 ($=V_{GS1} - V_{T1}$), since they carry the same current, assuming that transistors M_1, M_4, and M_6 are in saturation. Because the current drawn through transistor M_9 is equal to the current in transistor M_2, their effective source-gate voltages must also be equal assuming equal K factor for M_2 and M_9. Since the sum of the effective gate-source voltages transistors M_{11} and M_{12}, and also of M_9 and M_{10}, is fixed and equal to V_B, a situation exists which is equivalent to the two transistor level structure described in Ref. 5.

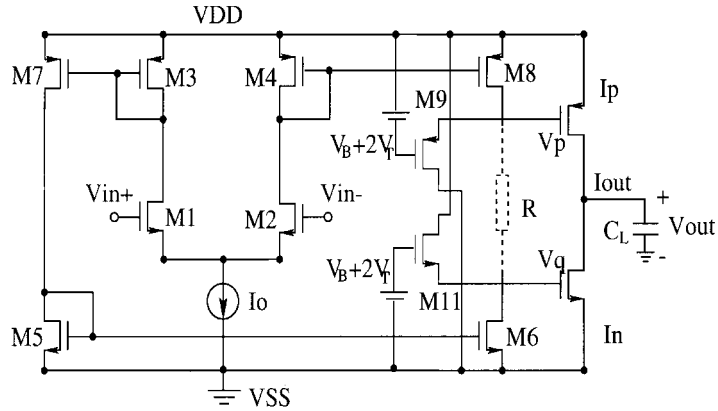


FIGURE 22.8 An OTA with an improved ratio between the maximum available current and the quiescent current of the output stage.

The ratio between the maximum available output current and the quiescent current of the output stage can be chosen by the designer. It is equal to: $(V_B/(V_B - V_{GS0}))^2$, where V_{GS0} is the quiescent gate-source voltage of transistor M_{11}.

If the OTA is used in an over-drive situation $|V_{in}| > \sqrt{(2I_0)/K}$, then either M_6 or M_5 will be cut off, while the other transistor carries its maximum current. As a result, one of the output transistors (M_{10} or M_{12}) carries its maximum current, while the other transistor is in a low-current stand-by situation. The maximum current that one of the output transistors carries is therefore proportional to V_B^2. With the high ohmic resistor R (indicated in Fig. 22.8 with dotted lines), this maximum current corresponds to either $(V_P - V_{SS} - V_{TN})^2$ or $(V_{DD} - V_Q - V_{TP})^2$, because in that situation no current flows through the resistor. Hence, with the extra resistor, it becomes possible to increase the maximum current in over-drive situations and therefore reduce the slewing time. Because resistor R is chosen to be high, it does not disturb the behavior of the circuit discussed previously. In practice, resistor R is replaced by transistor M_R working in the triode region, as shown in Fig. 22.9(a). Figure 22.9(b) shows the circuit which was used in Ref. 5 for biasing the gates of transistors M_0, M_9, and M_{11}. It is much like the so-called "replica biasing." The current in the circuit is strongly determined by the voltage across R (and its value) and is therefore very sensitive to variations in the supply voltage.

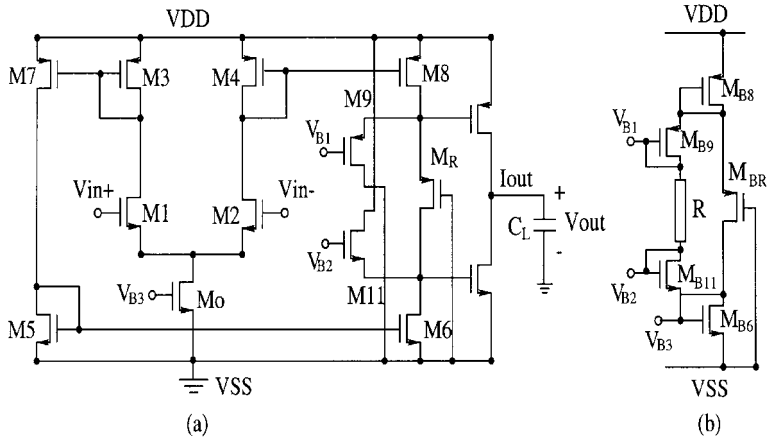


FIGURE 22.9 (a) The complete OTA, (b) and its bias stage.

Adaptively Biased OTAs

Another combination of high available output current with low standby current can be realized by making the tail current of the differential input pair signal dependent. Figure 22.10 shows the basic idea of such an OTA with adaptive biasing.[7] The tail current I_0 of the differential pair is the sum of a fixed value I_R and an additional current equal to the absolute value of the difference between the drain currents multiplied by the current feedback factor B ($I_0 = I_R + B|I_1 - I_2|$). Therefore, with zero differential input voltage, only a low bias current I_R flows through the input pair. A differential input voltage, V_{ind}, will cause a difference in the drain currents which will increase the tail current. This, in turn, again gives rise to a greater difference in the drain current, and so on. This is the kind of positive feedback that can bring the differential input pair from the weak-inversion mode into the strong-inversion mode, depending on the input voltage and the chosen current feedback factor B.

Normally, when $V_{ind} = V_{in+} - V_{in-}$ is small, the input transistors are in weak inversion. The differential output current $(I_1 - I_2)$ of a differential pair operating in weak inversion equals the tail current times $\tanh((qV_{ind})/(2AkT))$. This leads to the following equation:

$$(I_1 - I_2) = I_R + B|I_1 - I_2|\tanh\left(\frac{qV_{ind}}{2AkT}\right) \qquad (22.14)$$

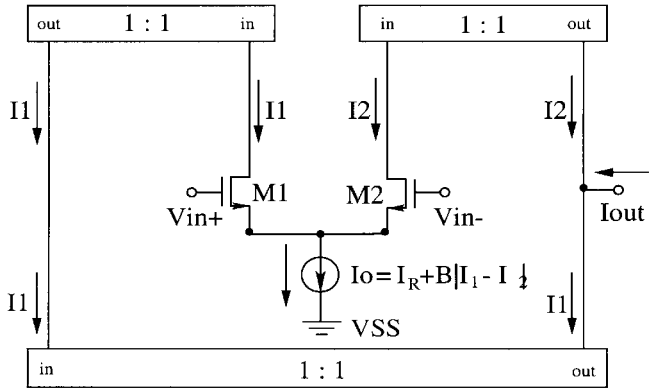


FIGURE 22.10 An OTA with an input dependent tail current.

or

$$(I_1 - I_2) = \frac{\tanh\left(\frac{qV_{ind}}{2AkT}\right)}{1 - B\left|\tanh\left(\frac{qV_{ind}}{2AkT}\right)\right|} I_R \tag{22.15}$$

and because $I_{out} = (I_1 - I_2)$:

$$I_{out} = \frac{\tanh\left(\frac{qV_{ind}}{2AkT}\right)}{1 - B\left|\tanh\left(\frac{qV_{ind}}{2AkT}\right)\right|} I_R \tag{22.16}$$

However, in the case of large currents, this expression will no longer be valid since $M_1 - M_2$ will leave the weak-inversion domain and enter the strong-inversion region. If that is the case, the output current becomes:

$$I_{out} = \begin{cases} \frac{k}{2} V_{ind} \sqrt{\frac{4I_R}{K} - (1 - B^2) V_{ind}^2} + B\frac{K}{2} V_{ind}^2 & \text{for } V_{ind} > 0 \\ \frac{k}{2} V_{ind} \sqrt{\frac{4I_R}{K} - (1 - B^2) V_{ind}^2} - B\frac{K}{2} V_{ind}^2 & \text{for } V_{ind} < 0 \end{cases} \tag{22.17}$$

In order to keep some control over the output current, a negative overall feedback must be applied, which is usually the case. For example, when an OTA is used as a unity-gain buffer with a load of C_L (see Fig. 22.11) and assuming a positive input step is applied, then the output current increases dramatically due to the positive feedback action described previously and, as a result, the output voltage will increase. This will lead to a decrease of the differential input voltage V_{ind} ($V_{ind} = V_s - V_{out}$). The result will be a very fast settling of the output voltage, and that is what we wanted to have. In order to realize current $|I_1 - I_2|$, two current-subtracter circuits can be combined (see Fig. 22.12). If the current I_2 is larger than current I_1, the output of current-subtracter circuit 1 (I_{out1}) will carry a current; otherwise, the output current will be zero. The opposite situation is found for the output current of subtracter circuit 2 because of the interchange of their input

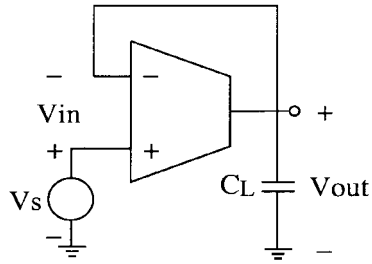


FIGURE 22.11 An OTA used as a unity gain buffer.

currents ($I_{out2} = B(I_1 - I_2)$). Consequently, either I_{out1} or I_{out2} will draw a current $B|I_1 - I_2|$ and the other current will be zero. It is for this reason that the upper current mirrors (in Fig. 22.13) have two extra outputs to support the currents for the circuit in Fig. 22.12. A practical realization of the adaptive biasing OTA is shown in Fig. 22.13. In order to avoid unwanted, relatively high stand-by currents due to transistors mismatches, the transfer ratio of the current mirrors (M_{12}, M_{13}) and (M_{19}, M_{18}) can be chosen somewhat larger than 1. This ensures an inactive region of the input voltage range whereby the feedback loop is deactivated.

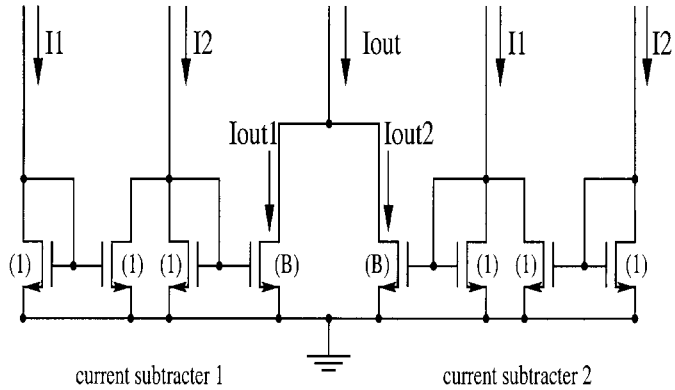


FIGURE 22.12 A combination of two current subtracters for realizing the adaptive biasing current for the circuit in Fig. 22.10.

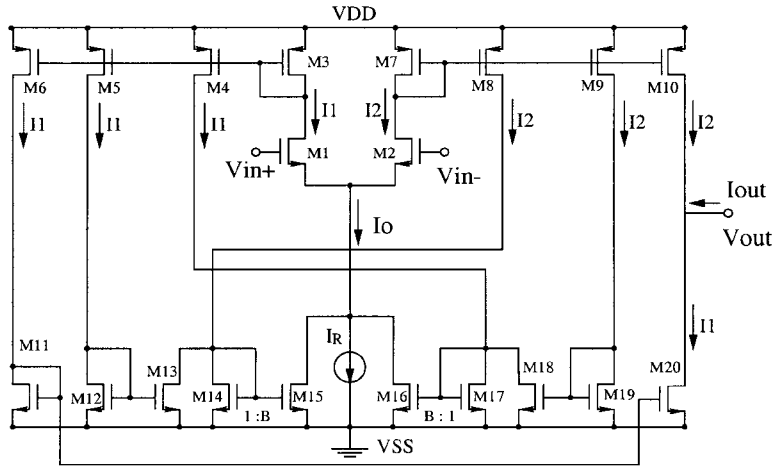


FIGURE 22.13 A practical realization of OTA with an adaptive biasing of its tail current.

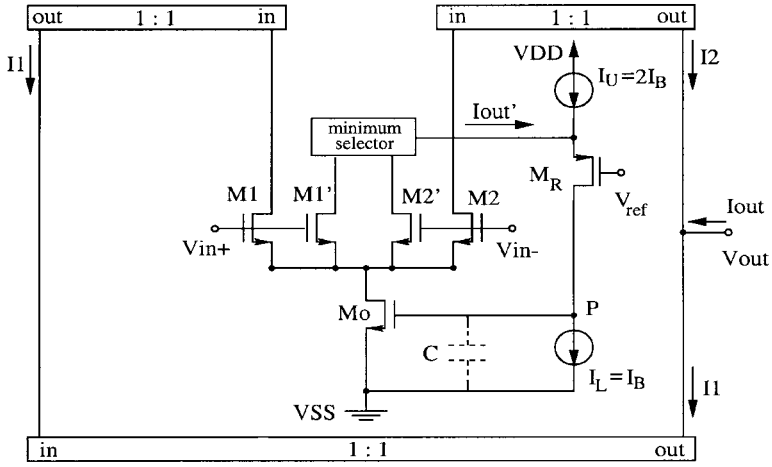


FIGURE 22.14 An OTA using a minimum selector for adapting the tail current.

Another example of an adaptive tail current circuit is shown in Fig. 22.14.[8] It has a normal OTA structure except that the input pair is realized in twofold, and the tail current transistor is used in a feedback loop. This feedback loop includes the inner differential pair and tail current transistor M_0 as well as a minimum current selector, the current source I_U, transistor M_R, and a current sink I_L. The minimum current selector[9] delivers an output current equal to the lowest value of its input currents ($I'_{out} = Min(I'_1, I'_2)$). The feedback loop ensures that the output current of the minimum current selector is equal to the difference in currents between the upper and lower current sources. Assume that the upper current carries a current $2I_B$ and the lower current source carries I_B, then the feedback loop will bias the tail current in such a way that either I'_1 or I'_2 becomes equal to I_B; for positive values of V_{ind}, that will be I'_2. It should be realized that at $V_{ind} = 0$, all four input transistors are biased at the same gate-source voltage (V_{GS0}), corresponding to a drain current I_B. In the case of positive input voltages, the gate-source voltage of M_2/M'_2 will not change.

Therefore, all the input voltage will be added to the bias voltage of M_1/M'_1, that is,

$$V_{GS1} = V_{GS0} + V_{ind} \tag{22.18}$$

Figure 22.15 shows the I_D vs. V_{GS} characteristic for both transistors M_1/M'_1 and M_2/M'_2. Accordingly, the relationship between $(I_1 - I_2)$ vs. V_{ind} (for $V_{ind} > 0$) follows the right side of the $I_D - V_{GS}$ curve of M_1,

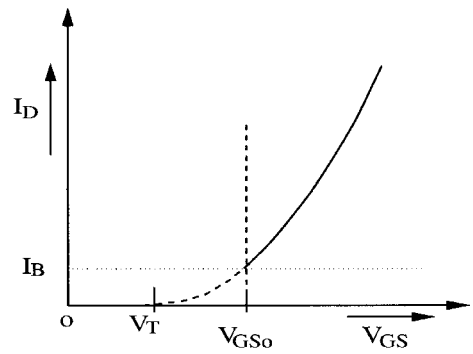


FIGURE 22.15 The I_D vs. V_{GS} characteristic for transistors M_1/M'_1 and $M_2/M(_2$, showing their standby point V_{GS0}, I_B.

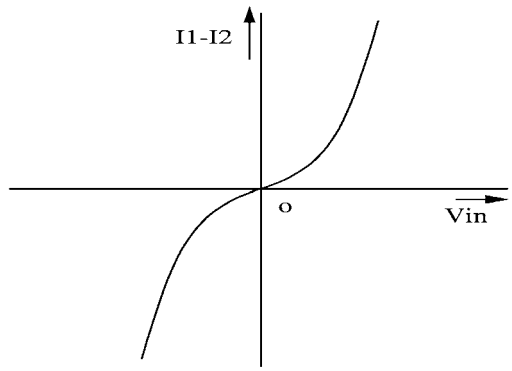


FIGURE 22.16 $I_1 - I_2$ vs. V_{ind}.

starting from the stand-by point (V_{GS0}, I_B) as indicated by the solid curve in Fig. 22.15. A similar view can be taken for negative values of the input voltage V_{ind}, resulting in an equal ($I_1 - I_2$) vs. V_{in} curve rotated 180°. The result is shown in Fig. 22.16.

Note that this input stage has a relationship between ($I_1 - I_2$) and V_{ind} that is different from that of a simple differential input stage. By increasing V_{ind}, the slope increases and, to a first-order approximation, there will not be a limit for the maximum value of ($I_1 - I_2$).

Note that there is an additional MOS transistor M_R in the circuit in Fig. 22.14 to fix the output voltage of the minimum current selector circuit. The lower current source I_L is necessary to be able to discharge the gate-source capacitor C of M_0 (indicated in Fig. 22.14 with dotted lines). The OTA in Fig. 22.14 is simpler than that in Fig. 22.13. However, its bandwidth is lower due to the high impedance of node P in the feedback loop.

Class AB OTAs

Another possibility to design an OTA with a good current efficiency is to use an input stage exhibiting a class AB characteristics.[11] The input stage in Fig. 22.17 contains two CMOS pairs[12] connected as Class AB input transistors. They are driven by four source-followers. By applying a differential input voltage,

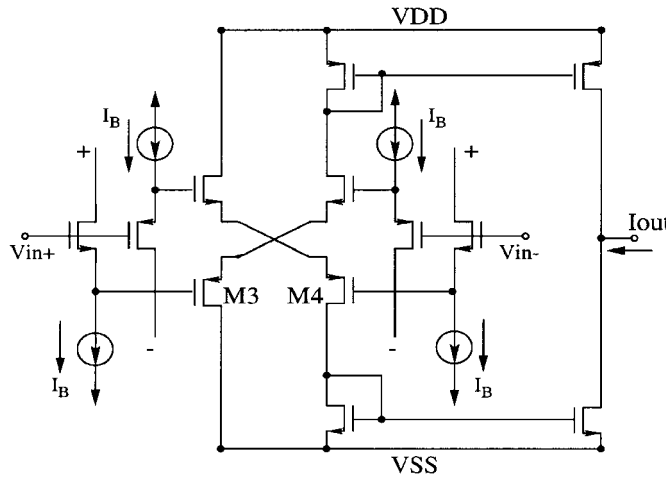


FIGURE 22.17 An OTA having a class AB input stage.

the current through one of the input pairs will increase while the current through the other will decrease. The maximum current that can flow through the CMOS pair is, to first order, unlimited. In practice, it is limited by the supply voltage, the K_{eq} factor, the mobility reduction factor, and the series resistance. The currents are delivered to the output with the help of two current mirrors. In the OTA shown in Fig. 22.17, only one of the two outputs of each CMOS pair is used. The other output currents flow directly to the supply rails. Instead of wasting the other output currents, they can be used to supply an extra output. So with the addition of two current mirrors, an OTA with complementary outputs as shown in Fig. 22.18 can be achieved.[10] An improvement of the output impedance and low-frequency voltage gain can be obtained by cascoding the output transistors of the current mirrors (Fig. 22.19). Usually, this reduces the output window. The function of transistors M_{41}-M_{44} is to control the dc output voltages. They form a part of a common-mode feedback system, which will be discussed next.

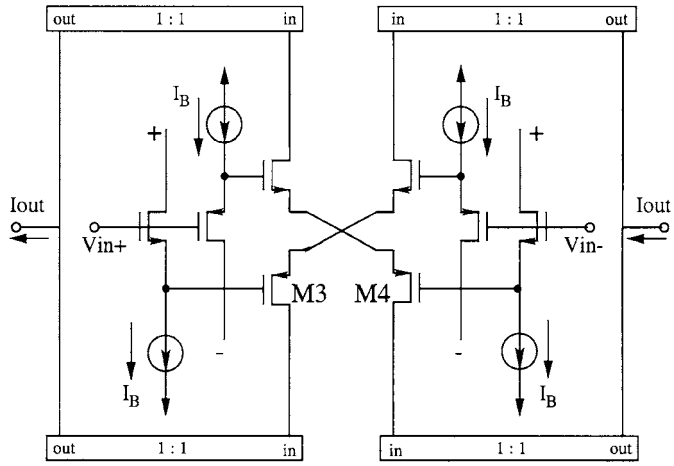


FIGURE 22.18 An OTA having a class AB input stage and two complementary outputs.

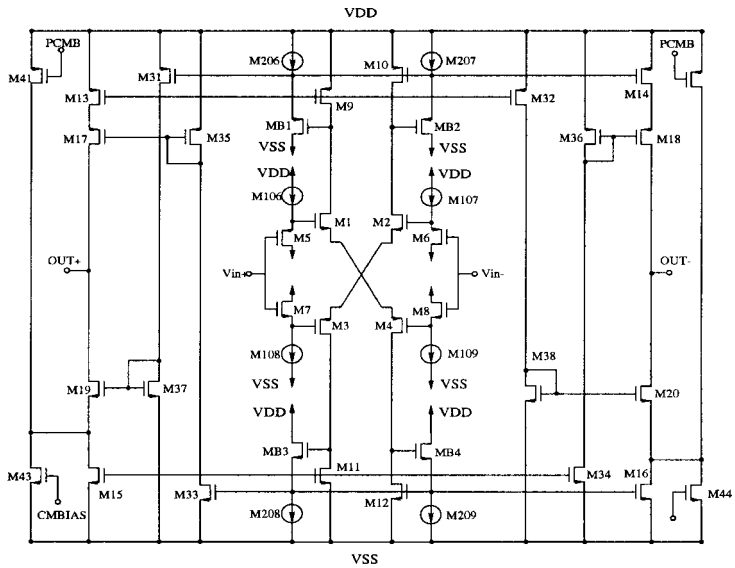


FIGURE 22.19 An improved fully differential OTA. (From Ref. 10. With permission.)

The relationship between the differential input voltage V_{in} and one of the output currents I_{out} is shown in Fig. 22.20. There is a linear relationship between V_{ind} and I_{out} for small to moderate values of V_{ind}. In the case of larger values of V_{ind}, one of the CMOS pairs becomes cut off, resulting in a quasi-quadratic relationship. At a further increase of V_{ind}, the output current will be somewhat saturated due to mobility reduction and to the fact that one of the transistors of the CMOS pair leaves saturation mode. The latter effect is, of course, also strongly dependent on the common input voltage.

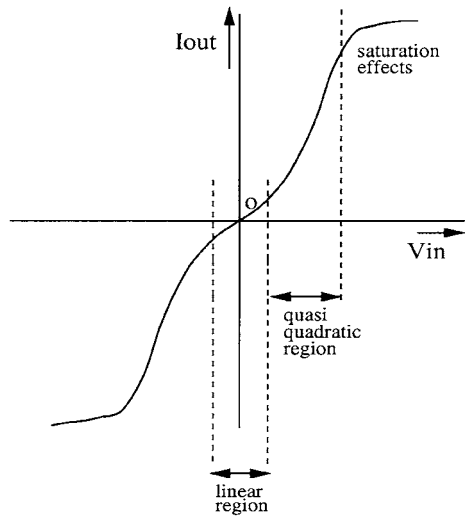


FIGURE 22.20 The $V_{in} \rightarrow I_{out}$ characteristic of the OTA in Fig. 5.19.

22.5 Common-Mode Feedback

A fully differential OTA circuit, as in Fig. 22.19, has many advantages compared with its single-ended counterpart. It is a basic building block in filter design. A fully differential approach, in general, leads to a more efficient current use, doubling of the maximum output-voltage swing, and an improvement of the power-supply rejection ratio (PSRR). It also leads to a significant reduction of the total harmonic distortion, since all even harmonics are canceled out due to the symmetrical structure. Even when there is a small imperfection in the symmetry, the reduction in distortion will be significant.

However, this type of symmetrical circuit needs an extra feedback loop. The feedback around a single-ended OTA usually only provides a differential-mode feedback and is ineffective for common-mode signals.

So, in the case of the fully differential OTA, a common-mode feedback (CMFB) circuit is needed to control the common output voltage. Without a CMFB, the common-mode output voltage of the OTA is not defined and it may drift out of its high-gain region. The general structure of a simple OTA circuit with a differential output and a CMFB circuit is shown in Fig. 22.21. The need for a CMFB circuit is a drawback since it counters many of the advantages of the fully differential approach. The CMFB circuit requires chip area and power, introduces noise, and limits the output-voltage swing.

Figure 22.22(b) shows a simple implementation of a CMFB circuit. A differential pair (M_1, M_2) is used to sense the common-mode output voltage. So, the voltage at the common source of this differential pair (V_s) is used. Its voltage provides, with a level shift of one V_{GS}, the common-mode output voltage of the OTA. The voltage at this node is the first order insensitive to the differential input voltage. The relationship between the differential input voltage V_{in} of the differential pair, superimposed on a common-mode input voltage V_{CM}, and its common-source voltage V_s is shown in Fig. 22.22(a). The common-mode output

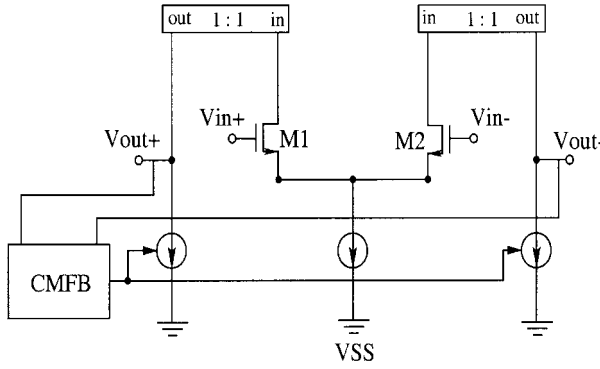


FIGURE 22.21 The general structure of a simple OTA circuit having a differential output and the required CMFB.

voltage of the OTA is determined by the V_{GS} of M_1/M_2 and M_9/M_{10} and can be controlled by the voltage source V_0. There might be an offset in the dc value of the two output voltages due to a mismatch in transistors M_9 and M_{10}.

If the amplitude of the differential output voltage increases, the common-mode voltage will not remain constant, but will be slightly modulated by the differential output voltage, with a modulation frequency that is twice the differential input signal frequency. This modulation is caused by the "non-flat" characteristic of the V_s vs. V_{in} characteristic of the differential pair (M_1, M_2) (see Fig. 22.22(a)).

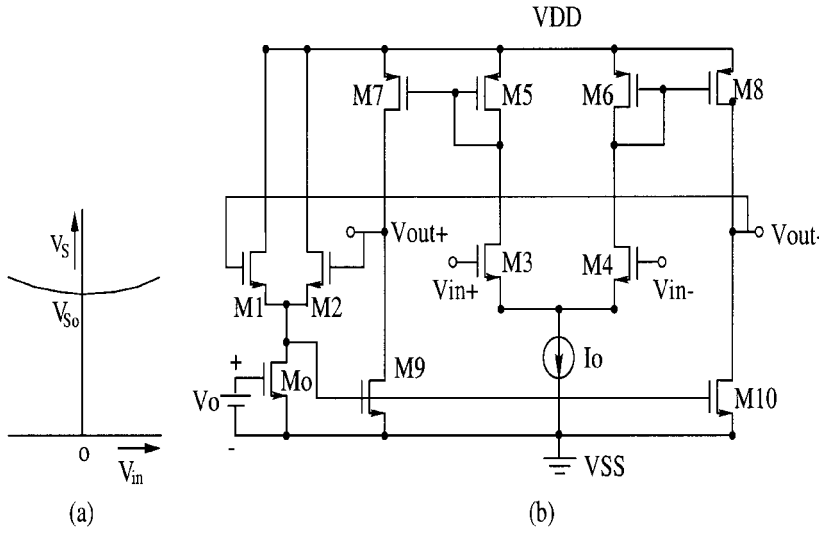


FIGURE 22.22 (a) The relationship between the differential input voltage, superimposed on a common-mode voltage V_{CM} of a differential pair (M_1, M_2) and its common-source voltage V_s; (b), a fully differential OTA with the differential pair (M_1, M_2) for providing a common-mode feedback.

Another commonly used CMFB circuit is shown in the fully-differential folded cascode OTA in Fig. 22.23.[13] In this circuit, a similar high-output resistance and high unloaded voltage gain can be achieved as in the normal cascode circuits. An advantage of the folded cascode technique, however, is a higher

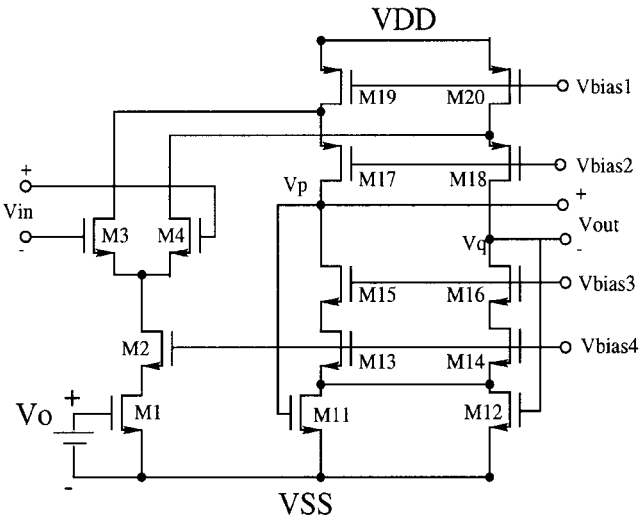


FIGURE 22.23 A fully differential folded cascode OTA with another commonly used CMFB circuit.

accuracy in the signal-current transfer because current mirrors are avoided. In Fig. 22.23, all transistors are in saturation, with the exception of M_1, M_{11}, and M_{12}, which are in the triode region. The CMFB is provided with the help of M_{11} and M_{12}. These two transistors sense the output voltages V_P and V_Q. Since they operate in the triode region, their sum-current is insensitive to the differential output voltage ($V_P - V_Q$) and depends only on the common output voltage ($(V_P+V_Q)/2$). Because the current that flows through M_{17} and M_{18} forces the value of the above-mentioned sum-current, they also determine, together with V_{bias4}, the common-mode output voltage. By choose V_{bias1} in such a way that I_{M19} is twice I_{M17}, and making the width of transistor M_1 twice that of M_{11} (= M_{12}), the nominal common-mode output voltage will be equal to the gate voltage of M_1.

22.6 Filter Applications with Low-Voltage OTAs

Usually, g_m-C filters are considered suitable candidates for high-speed and low-power applications. Compared with the SC op-amp and RC op-amp techniques, the applicability of the g_m-C filter is limited by the low dynamic range and medium, even poor, linearity. The strategy of both simplifying the architecture and designing an ultra-low-voltage OTA[16] are used to meet low-power and dynamic range requirements. The filter topology is derived from a passive ladder form of 5th-order elliptic filtering. Using element replacement and sharing multiple inputs for gyrators, a fully differential 5th-order elliptic filter is shown in Fig. 22.24.[14] This multi-input sharing in the filter design reduces the numbers of OTAs from 11 to 6. Especially for wide bandwidth design, the method saves almost half of the die area and power dissipation. This design uses balanced signals to reduce even harmonics and to relax parasitic matching requirements. The capacitors realizing the filter poles are connected between the outputs of the transconductors and signal ground. This helps the stability of the common-mode feedback circuit because of the loading to both common-mode and fully-differential signal paths> The inherent 6 dB loss at low frequency is compensated for by the first transconductor with $2g_m$ gain. Fig. 22.25 shows the frequency response and the passband of the filter. The filter has –3dB frequency tuning ranges of 1.2 MHz to 2.9 MHz and –60 dB stop band rejection. Obviously, the tuning range is limited by the low-power supply rail. This presents a problem for automatic tuning design at very low supply voltages. A possible application for this filter is for channel selection in wideband handy phones.[15]

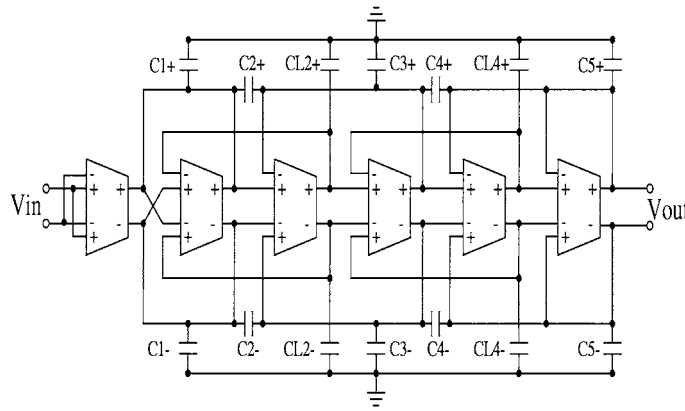


FIGURE 22.24 A g_m-C elliptic filter with low-voltage OTAs.

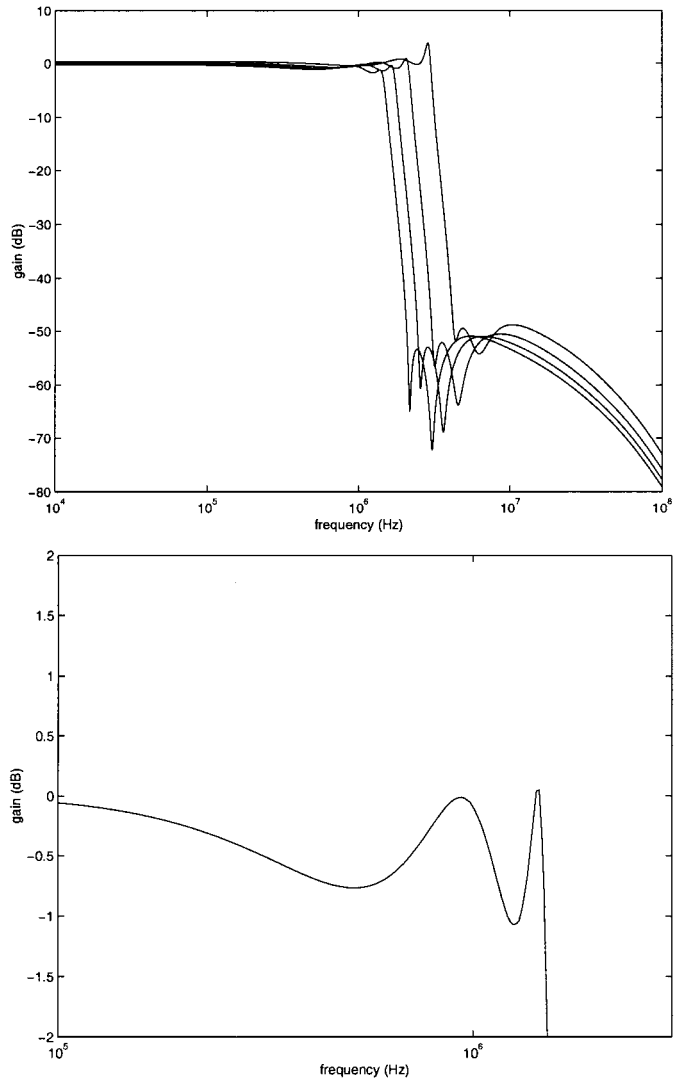


FIGURE 22.25 Frequency response and passband of the filter.

References

1. M. Ismail and T. Fiez, *Analog VLSI Signal and Information Processing*, McGraw-Hill, 1994.
2. E. A. Vittoz, The design of high-performance analog circuits on digital CMOS chips, *IEEE J. Solid-State Circuits*, vol. SC-20, pp. 657-665, June 1985.
3. F. Krummenacher, High voltage gain CMOS OTA for micro-power SC filters, *Electronics Letters*, vol. 17, pp. 160-162, 1981.
4. M. S. J. Steyaert, W. Bijker, P. Vorenkamp, and J. Sevenhans, ECL-CMOS and CMOS-ECL Interface in 1.27mm CMOS for 150 MHz Digital ECL Data Transmission Systems, *IEEE J. Solid-State Circuits*, vol. SC-26, pp. 15-24, Jan. 1991.
5. R. F. Wassenaar, Analysis of analog C-MOS circuits, Ph.D. thesis, University of Twente, The Netherlands, 1996.
6. S. L. Wong and C. A. T. Salama, An efficient CMOS buffer for driving large capacitive loads, *IEEE J. Solid-State Circuits*, vol. SC-21, pp. 464-469, June 1986.
7. M. G. Degrauwe, J. Rijmenants, E. A. Vittoz, and H. J. DeMan, Adative biasing CMOS amplifiers, *IEEE J. Solid-State Circuits*, vol. SC-17, pp. 522-528, June 1982.
8. E. Seevinck, R. F. Wassenaar, and W. de Jager, Universal adaptive biasing principle for micro-power amplifiers, *Digest of Technical Papers ESSCIRC'84*, pp. 59-62, Sept. 1984.
9. R. F. Wassenaar, Current-Mode Minimax Circuit, *IEEE Circuits and Devices*, vol. 8, pp. 47, Nov. 1992.
10. S. H. Lewis and P. R. Gray, A pipelined 5MHz 9b ADC, *Proceedings ISSCC'87*, pp. 210-211, 1987.
11. S. Dupuie and M. Ismail, High frequency CMOS transconductors, Ch. 5 in *Analog IC Design: The Current-Mode Approach*, Toumazou, Lidgey, and Haight, Eds., Peter Peregrinus, Ltd., London, 1990.
12. E. Seevinck and R. F. Wassenaar, A versatile CMOS linear transconductor/square-law function circuit, *IEEE Journal of Solid-State Circuts*, vol. SC-22, no. 3, pp. 366-377, June 1987.
13. T. C. Choi, R. T. Kaneshiro, R. Brodersen, and P. R. Gray, High-frequency CMOS switched capacitor filters for communication applications, *Proceedings ISSCC'83*, pp. 246-247, 314, 1983.
14. R. Schaumann, Continuous-Time Integrated Filters, Ch. 80, *The Circuits and Filters Handbook*, W.-K. Chen, Editor-in-Chief, CRC Press and IEEE Press, New York, 1995.
15. C.-C. Hung, K. A. I. Halonen, M. Ismail, V. Porra, and A. Hyogo, A low-voltage, low-power CMOS fifth-order elliptic GM-C filter for baseband mobile, wireless communication, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 4, pp. 584-592, August 1997.
16. C.-H. Lin and M. Ismail, Design and analysis of an ultra low-voltage CMOS class-AB V-I converter for dynamic range enhancement, *International Symposium on Circuits and Systems*, Orlando, Florida, June 1999.

Muroga, S. "Expressions of Logic Functions"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

23

Expressions of Logic Functions

Saburo Muroga
University of Illinois at Urbana-Champaign

23.1 Introduction to Basic Logic Operations

In a contemporary digital computer, logic operations for computational tasks are usually done with signals that take values of 0 or 1. These logic operations are performed by many logic networks which constitute the computer. Each logic network has input variables $x_1, x_2, \ldots, x_n$ and output functions $f_1, f_2, \ldots, f_m$. Each of the input variables and output functions take only binary value, 0 or 1. Now let us consider one of these output functions, f. Any **logic function** f can be expressed by a **combination table** (also called a **truth table**) exemplified in Table 23.1.

TABLE 23.1 Combination Table

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Basic Logic Expressions

Any logic function can be expressed with three basic logic operations: OR, AND, and NOT. It can also be expressed with other logic operations, as explained in a later section.

The **OR operation** of n variables $x_1, x_2, \ldots, x_n$ yields the value 1 whenever at least one of the variables is 1, and 0 otherwise, where each of $x_1, x_2, \ldots, x_n$ assumes the value 0 or 1. This is denoted by $x_1 \vee x_2 \vee$

$\ldots \vee x_n$. The OR operation defined above is sometimes called **logical sum**, or **disjunction**. Also, some authors use "+", but throughout Section V, we use $\vee$, and + is used to mean an arithmetic addition.

The **AND operation** of n variables yields the value 1 if and only if all variables $x_1, x_2, \ldots, x_n$ are simultaneously 1. This is denoted by $x_1 \cdot x_2 \cdot x_3 \ldots x_n$. These dots are usually omitted: $x_1x_2x_3 \ldots x_n$. The AND operation is sometimes called **conjunction**, or **logical product**.

The **NOT operation** of a variable x yields the value 1 if $x = 0$, and 0 if $x = 1$. This is denoted by $\bar{x}$ or x'. The NOT operation is sometimes called **complement** or **inversion**.

Using these operations, AND, OR, and NOT, a logic function, such as the one shown in Table 23.1 can be expressed in the following formula:

$$f = \bar{x}y\bar{z} \vee \bar{x}yz \vee xyz \tag{23.1}$$

Logic Expressions

Expressions with logic operations with AND, OR, and NOT, such as Eq. 23.1, are called **switching expressions** or **logic expressions**. Variables x, y, and z are sometimes called **switching variables** or **logic variables**, and they assume only binary values 0 and 1. In logic expressions such as Eq. 23.1, each variable, x_i, appears with or without the NOT operation, that is, as $\bar{x}_i$ or x_i. Henceforth, $\bar{x}_i$ and x_i are called the **literals** of a variable x_i.

Logic Expressions with Cubes

Logic expressions such as $f = x\bar{y} \vee yz \vee \bar{x}y\bar{z}$ can be expressed alternatively as a set, {(10-), (-11), (010)}, using components in a vector expression such that the first, second, and third components of the vector represent x, y, and z, respectively, where the value "1" represents x_i, "0" represents $\bar{x}_i$, and "-" represents the lack of the variable. For example, (10-) represents $x\bar{y}$. These vectors are called **cubes**. Logic expressions with cubes are used often because of their convenience for processing by a computer.

23.2 Truth Tables

The value of a function f for different combinations of values of variables can be shown in a table, as exemplified in Tables 23.1 and 23.2. The table for n variables has 2^n rows. Thus the table size increases rapidly as n increases.

TABLE 23.2 Truth Table with Don't-Care Conditions

Decimal Number of Row	Variables			Function
	x	y	z	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	d
3	0	1	1	0
4	1	0	0	d
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Under certain circumstances, some of the combinations of input variable values never occur, or even if they occur, we do not care what values f assumes. These combinations are called **don't-care conditions,** or simply **don't-cares,** and are denoted by "d" or "*", as shown in Table 23.2.

Decimal Specifications

A concise means of expressing the truth table is to list only rows with $f = 1$ and d, identifying these rows with their decimal numbers in the following **decimal specifications**. For example, the truth table of Table 23.2 can be expressed, using Σ, as

$$f(x, y, z) = \Sigma(1, 5, 6) + d(2, 4)$$

If only rows with $f = 0$ and d are considered, the truth table in Table 23.2 can be expressed, using Π, as

$$f(x, y, z) = \Pi(0, 3, 7) + d(2, 4)$$

23.3 Karnaugh Maps

Logic functions can be visually expressed using a Karnaugh map, which is simply a different way of representing a truth table, as exemplified for four variables in Fig. 23.1(a). For the case of four variables, for example, a Karnaugh map consists of 16 cells; that is, 16 small squares as shown in Fig. 23.1(a). Here, two-bit numbers along the horizontal line above the squares show the values of x_1 and x_2, and two-bit binary numbers along the vertical line on the left of the squares show the values of x_3 and x_4. The top left cell in Fig. 23.1(a) has 1 inside for $x_1 = x_2 = x_3 = x_4 = 0$. Also, the cell in the second row and the second column from the left has d inside. This means $f = d$ (i.e., don't-care) for $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, and $x_4 = 1$. The binary numbers that express variables are arranged in such a way that binary numbers for any two cells that are horizontally or vertically adjacent differ in only one bit position. Also, the two numbers in each row in the first and last columns differ in only one bit position and are interpreted to be adjacent. Also, the two numbers in each column in the top and bottom rows are similarly interpreted to be adjacent. Thus, the four cells in the top row are interpreted to be adjacent to the four cells in the bottom row in each column. The four cells in the first column are interpreted to be adjacent to the four cells in the last column in each row. With this arrangement of cells and this interpretation, a Karnaugh map is more than a concise representation of a truth table; it can express many important algebraic concepts, as we will see later. A Karnaugh map is a two-dimensional representation of the 16 cells on the surface of a torus, as shown in Fig. 23.1(b), where the two ends of the map are connected vertically and horizontally.

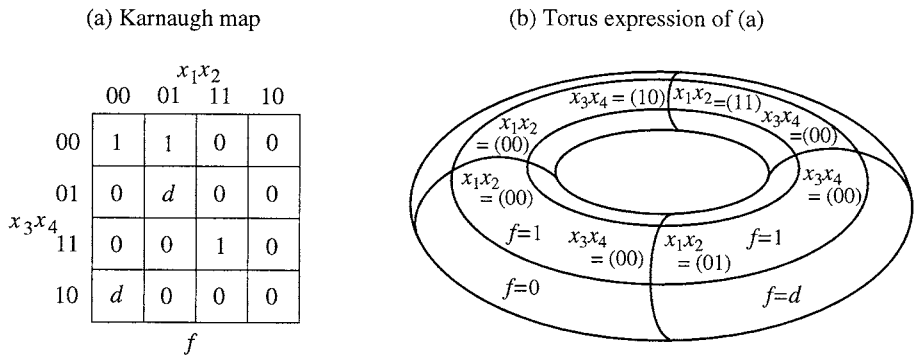


FIGURE 23.1 Karnaugh map for four variables.

Figure 23.2 shows the correspondence between the cells in the map in Fig. 23.2(a) and the rows in the truth table in Fig. 23.2(b). Notice that the rows in the truth table are not shown in consecutive order in the Karnaugh map. The Karnaugh map labeled with variable letters, instead of with binary numbers,

(a) Karnaugh map labeled with binary numbers

x_3x_4 \ x_1x_2	00	01	11	10
00	f_0	f_4	f_{12}	f_8
01	f_1	f_5	f_{13}	f_9
11	f_3	f_7	f_{15}	f_{11}
10	f_2	f_6	f_{14}	f_{10}

f

(b) Truth table

x_1	x_2	x_3	x_4	f
0	0	0	0	f_0
0	0	0	1	f_1
0	0	1	0	f_2
0	0	1	1	f_3
0	1	0	0	f_4
0	1	0	1	f_5
0	1	1	0	f_6
0	1	1	1	f_7
1	0	0	0	f_8
1	0	0	1	f_9
1	0	1	0	f_{10}
1	0	1	1	f_{11}
1	1	0	0	f_{12}
1	1	0	1	f_{13}
1	1	1	0	f_{14}
1	1	1	1	f_{15}

(c) Karnaugh map labeled with variable letters

x_3 \ x_1				
	f_0	f_4	f_{12}	f_8
	f_1	f_5	f_{13}	f_9
	f_3	f_7	f_{15}	f_{11}
	f_2	f_6	f_{14}	f_{10}

x_4 (right brace), x_2 (bottom brace)

FIGURE 23.2 Correspondence between the cells in a Karanaugh map for four variables and the rows in a truth table.

shown in Fig. 23.2(c), is also often used. Although a 1 or 0 shows the function's value corresponding to a particular cell, 0 is often not shown in each cell. Cells that contain 1's are called **1-cells** (similarly, **0-cells**).

Patterns of Karnaugh maps for two and three variables are shown in Figs. 23.3(a) and (b), respectively. As we extend this treatment to the cases of 5 or more variables, the maps, which will be explained in a later subsection, become increasingly complicated.

(a) Two variables

x_2 \ x_1	0	1
0	f_0	f_2
1	f_1	f_3

f

(b) Three variables

x_3 \ x_1x_2	00	01	11	10
0	f_0	f_2	f_6	f_4
1	f_1	f_3	f_7	f_5

f

FIGURE 23.3 Karanaugh map for two and three variables.

A rectangular loop that consists of 2^i 1-cells without including any 0-cells expresses a product of literals for any i, where $i \geq 1$. For example, the square loop consisting four 1-cells in Fig. 23.4(a) represents the product $x_2\bar{x}_3$, as we can see it from the fact that $x_2\bar{x}_3$ takes value 1 only for these 1-cells, i.e., $x_1 = 0$, $x_2 = 1$, $x_3 = x_4 = 0$; $x_1 = x_2 = 1$, $x_3 = x_4 = 0$; $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$; and $x_1 = x_2 = 1$, $x_3 = 0$, $x_4 = 1$. A rectangular loop consisting of a single 1-cell, such as the one in Fig. 23.4(b), for example, represents the product of literals that the cube (0001) expresses, i.e., $\bar{x}_1\bar{x}_2\bar{x}_3x_4$. Thus, the map in Fig. 23.4(a) expresses the function $x_2\bar{x}_3 \vee x_1x_3\bar{x}_4$ and the map in Fig. 23.4(b) expresses the function $\bar{x}_1\bar{x}_2\bar{x}_3x_4 \vee x_1x_2\bar{x}_3 \vee x_2\bar{x}_4$.

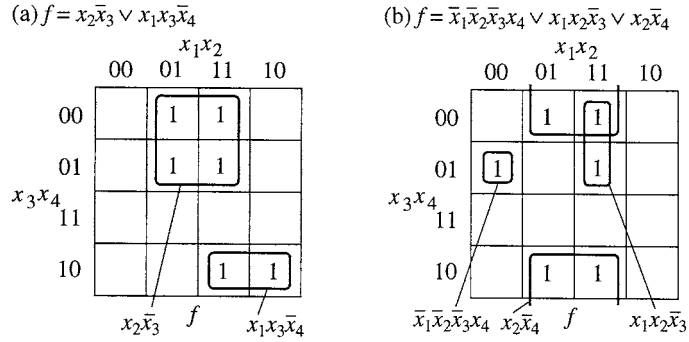


FIGURE 23.4 Products of literals expressed on Karnaugh maps.

Two Completely Specified Functions to Express an Incompletely Specified Function

Suppose we have an incompletely specified function f, as shown in Fig. 23.5(a), i.e., a function that has some don't-cares. This incompletely specified function f can be expressed alternatively with two completely specified functions, f^{ON} and f^{OFF}, shown in Fig. 23.5(b) and (c), respectively. f^{ON} is called **ON-set** of f and is the function whose value is 1 for $f = 1$and 0 for $f = 0$ and d. f^{OFF} is called **OFF-set** of f and is 1 for $f = 0$ and 0 for $f = 1$ and d. **Don't-care set** of f can be derived as

$$f^{DC} = \overline{f^{ON} \vee f^{OFF}}$$

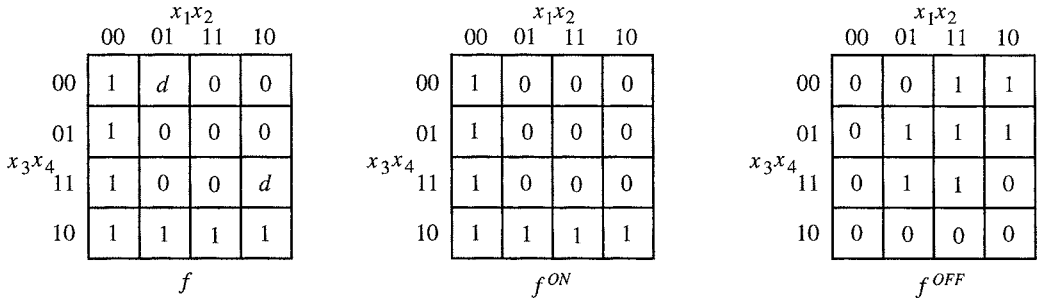


FIGURE 23.5 Expression of an incompletely specified function f with two completely specified functions.

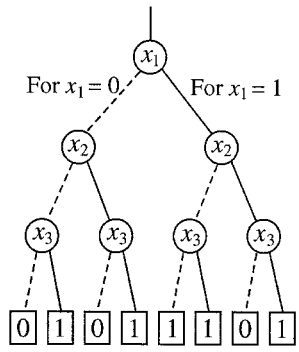
23.4 Binary Decision Diagrams

Truth tables or logic expressions can be expressed with **binary decision diagrams**, which are usually abbreviated as **BDDs**. Compared with logic expressions or truth tables, BDDs have unique features, such as unique concise representation, processing speed, and the memory space, as discussed in a later section.

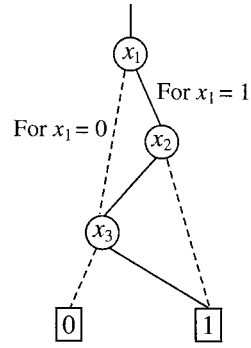
Let us consider a logic expression, $x_1\bar{x}_2 \vee x_3$, for example. This can be expressed as the truth table shown in Fig. 23.6(a). Then, this can be expressed as the BDD shown in Fig. 23.6(b). It is easy to see why Fig. 23.6(b) represents the truth table in Fig. 23.6(a). Let us consider the row, $(x_1x_2x_3) = (011)$, for example, in Fig. 23.6(a). In Fig. 23.6(b), starting from the top node which represents x_1, we go down to

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

(a) Truth table



(b) BDD equivalent to (a)



(c) Reduced BDD for (b)

FIGURE 23.6 Binary decision diagram.

the left node which represents x_2, following the dotted line corresponding to $x_1 = 0$. From this node, we go down to the second node from the left which represents x_3, following the solid line corresponding to $x_2 = 1$. From this node, we go down to the fourth rectangle from the left which represents, $f = 1$, following the solid line corresponding to $x_3 = 1$. Thus, we have the value of f that is shown for the row, $(x_1, x_2, x_3) = (011)$ in the truth table in Fig. 23.6(a). Similarly, for any row in the truth table, we reach the rectangle that shows the value of f identical to that in the truth table in Fig. 23.6(a), by following a solid or dotted line corresponding to 1 or 0 for each of x_1, x_2, x_3, respectively. BDD in Fig. 23.6(b) can be simplified to the BDD shown in Fig. 23.6(c), which is called the reduced BDD, by the reduction to be described in a later section.

When a function has don't-cares, *d*'s, we can treat it in the same manner by considering a rectangle for *d*'s, as shown in Fig. 23.7.

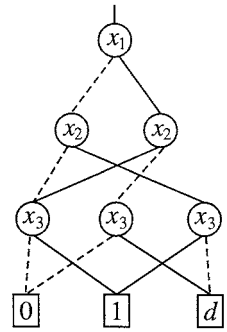


FIGURE 23.7 Reduced BDD with don't-cares.

Muroga, S."Basic Theory of Logic Functions"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

24

Basic Theory of Logic Functions

Saburo Muroga
University of Illinois at Urbana-Champaign

24.1 Basic Theorems

Theory on logic functions where the values of variables and their functions are 0 or 1 only is called **switching theory**. Here, let us discuss the basics of switching theory.

Let us denote the set of input variables by the vector expression $(x_1, x_2, \ldots, x_n)$. There are 2^n different input vectors when each of these n variables assume the value 1 or 0. An input vector $(x_1, x_2, \ldots, x_n)$ such that $f(x_1, x_2, \ldots, x_n) = 1$ or 0 is called a **true (input) vector**, or a **false (input) vector** of f, respectively. Vectors with n components are often called ***n*-dimensional vectors** if we want to emphasize that there are n components. When the value of a logic function f is specified for each of the 2^n vectors (i.e., for every combination of the values of $x_1, x_2, \ldots, x_n$), f is said to be **completely specified**. Otherwise, f is said to be **incompletely specified**; that is, the value of f is specified for fewer than 2^n vectors. Input vectors for which the value of f is not specified are called **don't-care conditions** usually denoted by "d" or "*", as described in Chapter 23. These input vectors are never applied to a network whose output realizes f, or the values of f for these input vectors are not important. Thus, the corresponding values of f need not be considered.

If there exists a pair of input vectors $(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$ and $(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$ that differ only in a particular variable x_i, such that the values of f for these two vectors differ, the logic function $f(x_1, x_2, \ldots, x_n)$ is said to be **dependent on** x_i. Otherwise, it is said to be **independent of** x_i. In this case, f can be expressed without the x_i in the logic expression of f. If f is independent of x_i, x_i is called a **dummy variable**. If $f(x_1, x_2, \ldots, x_n)$ depends on all its variables, it is said to be **non-degenerate**; otherwise, **degenerate**. For example, $x_1 \vee x_2x_3 \vee x_2\bar{x}_3$ can be expressed as $x_1 \vee x_2$ without dummy variable x_3.

Logic Expressions and Expansions

Given variable x_i, x_i and $\bar{x}_i$ are called **literals** of x_i, as already explained.

Definition 24.1: (1) A conjunction (i.e., a logical product) of literals where a literal for each variable appears at most once is called a **term** (or a **product**). A term may consist of a single literal. A disjunction (i.e., logical sum) of terms is called a **disjunctive form** (or a sum of products). (2) Similarly, a disjunction (i.e., a logical sum) of literals where a literal for each variable appears at most

once is called an **alterm**. An alterm may consist of a single literal. A conjunction of alterms is called a **conjunctive form** (or a product of sums).

For example, $x_1\bar{x}_2x_3$ is a term, and $x_1 \vee x_2 \vee \bar{x}_3$ is an alterm. Also, $x_1x_2 \vee x_1 \vee x_2$ and $x_1 \vee \bar{x}_1x_2$ are disjunctive forms that are equivalent to the logic function $x_1 \vee x_2$, but $x_1 \vee x_2(\bar{x}_1 \vee \bar{x}_2)$ is not a disjunctive form, although it expresses the same function. A disjunctive form does not contain products of literals that are identically 0 (e.g., $x_1\bar{x}_2x_2$) from the first sentence of (1). Similarly, a conjunctive form does not contain disjunctions of literals that are identically 1 (e.g., $x_1 \vee \bar{x}_2 \vee x_2 \vee x_3$).

The following expressions of a logic function are important special cases of a disjunctive form and a conjunctive form.

Definition 24.2: Assume that n variables, $x_1, x_2, \ldots, x_n$, are under consideration. (1) A **minterm** is defined as the conjunction of exactly n literals, where exactly one literal for each variable (x_i and $\bar{x}_i$ are two literals of a variable x_i) appears. When a logic function f of n variables is expressed as a disjunction of minterms without repetition, it is called the **minterm expansion** of f. (2) A **maxterm** is defined as a disjunction of exactly n literals, where exactly one literal for each variable appears. When f is expressed as a conjunction of maxterms without repetition, it is called the **maxterm expansion** of f.

For example, when three variables, x_1, x_2, and x_3, are considered, there exist $2^3 = 8$ minterms: $\bar{x}_1\bar{x}_2\bar{x}_3$, $\bar{x}_1\bar{x}_2x_3$, $\bar{x}_1x_2\bar{x}_3$, $\bar{x}_1x_2x_3$, $x_1\bar{x}_2\bar{x}_3$, $x_1\bar{x}_2x_3$, $x_1x_2\bar{x}_3$, and $x_1x_2x_3$. For the given function $x_1 \vee x_2x_3$, the minterm expansion is $\bar{x}_1x_2x_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3 \vee x_1x_2x_3$ and the maxterm expansion is $(x_1 \vee x_2 \vee x_3)$ $(x_1 \vee x_2 \vee \bar{x}_3)$ $(x_1 \vee \bar{x}_2 \vee x_3)$, as explained in the following.

Also notice that the row for each true vector in a truth table and also its corresponding 1-cell in the Karnaugh map correspond to a minterm. If $f = 1$ for $x_1 = x_2 = 1$ and $x_3 = 0$, then this row in the truth table and its corresponding 1-cell in the Karnaugh map corresponds to a minterm $x_1x_2\bar{x}_3$. Also, as will be described in a later section, the row for each false vector in a truth table and also its corresponding 0-cell in the Karnaugh map corresponds to a maxterm. For example, if $f = 0$ for $x_1 = x_2 = 1$ and $x_3 = 0$, then this row in the truth table and its corresponding 0-cell in the Karnaugh map corresponds to a maxterm $(x_1 \vee x_2 \vee x_3)$.

Theorem 24.1: Any logic function can be uniquely expanded with minterms and also with maxterms.

For example, $f(x_1, x_2) = x_1 \vee x_2\bar{x}_3$ can be uniquely expanded with the minterms as

$$x_1 \vee x_2\bar{x}_3 = \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1\bar{x}_2x_3 \vee x_1x_2\bar{x}_3 \vee x_1x_2x_3$$

and also can be uniquely expressed with maxterms as

$$x_1 \vee x_2\bar{x}_3 = (x_1 \vee x_2 \vee x_3)(x_1 \vee x_2 \vee \bar{x}_3)(x_1 \vee \bar{x}_2 \vee \bar{x}_3) .$$

These expansions have different expressions but both express the same function $x_1 \vee x_2\bar{x}_3$.

The following expansions, called **Shannon's expansions,** are often useful.

Any function $f(x_1, x_2, \ldots, x_n)$ can be expanded into the following expression with respect x_1:

$$f(x_1, x_2, \ldots, x_n) = x_1 f(1, x_2, x_3, \ldots, x_n) \vee \bar{x}_1 f(0, x_2, x_3, \ldots, x_n)$$

where $f(1, x_2, x_3, \ldots, x_n)$ and $f(0, x_2, x_3, \ldots, x_n)$, which are $f(x_1, x_2, \ldots, x_n)$ with x_1 set to 1 and 0, respectively, are called **cofactors**. By further expanding each of $f(1, x_2, x_3, \ldots, x_n)$ and $f(0, x_2, x_3, \ldots, x_n)$ with respect to x_2, we have

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) = {} & x_1x_2 f(1, x_2, x_3, \ldots, x_n) \vee x_1\bar{x}_2 f(1, 0, x_3, \ldots, x_n) \\ & \vee \bar{x}_1x_2 f(0, 1, x_3, \ldots, x_n) \vee \bar{x}_1\bar{x}_2 f(0, 0, x_3, \ldots, x_n) \end{aligned}$$

Then we can further expand with respect to x_3. And so on.

Also, similarly

$$f(x_1, x_2, \ldots, x_n) = (x_1 \vee f(0, x_2, x_3, \ldots, x_n))(\bar{x}_1 \vee f(1, x_2, x_3, \ldots, x_n))$$
$$f(x_1, x_2, \ldots, x_n) = (x_1 \vee x_2 \vee f(0, 0, x_3, \ldots, x_n))(x_1 \vee \bar{x}_2 \vee f(0, 1, x_3, \ldots, x_n))$$
$$(\bar{x}_1 \vee x_2 \vee f(1, 0, x_3, \ldots, x_n))(\bar{x}_1 \vee \bar{x}_2 \vee f(1, 1, x_3, \ldots, x_n))$$

And so on.

These expansions can be extended to the case with m variables factored out, where $m \leq n$, although the only expansions for $m = 1$ (i.e., x_1) and 2 (i.e., x_1 and x_2) are shown above. Of course, when $m = n$, the expansions become the minterm and maxterm expansions.

Theorem 24.2: De Morgan's Theorem —

$$\overline{(x_1 \vee x_2 \vee \ldots \vee x_n)} = \bar{x}_1\bar{x}_2\ldots\bar{x}_n \qquad \text{and} \qquad \overline{(x_1x_2\ldots x_n)} = \bar{x}_1 \vee \bar{x}_2 \vee \ldots \vee \bar{x}_n$$

A logic function is realized by a **logic network** that consists of **logic gates**, where logic gates are realized with hardware, such as transistor circuits.

De Morgan's Theorem 24.2 has many applications. For example, it asserts that a NOR gate, i.e., a logic gate whose output expresses the complement of the OR operation on its inputs, with noncomplemented variable input $x_1, x_2, \ldots, x_n$ is interchangeable with an AND gate, i.e., a logic gate whose output expresses the AND operation on its complemented variable inputs $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$, since the outputs of both gates express the same function. This is illustrated in Figure 24.1 for $n = 2$.

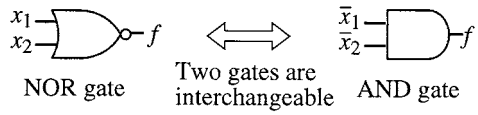


FIGURE 24.1 Application of De Morgan's theorem.

Definition 24.3: The **dual** of a logic function $f(x_1, x_2, \ldots, x_n)$ is defined as $\bar{f}(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$ where $\bar{f}$ denotes the complement of the entire function $f(x_1, x_2, \ldots, x_n)$ [in order to denote the complement of a function $f(x_1, x_2, \ldots, x_n)$, the notation $\overline{f(x_1, x_2, \ldots, x_n)}$ might be used instead of $\bar{f}$]. Let it be denoted by $f^d(x_1, x_2, \ldots, x_n)$. In particular, if $f(x_1, x_2, \ldots, x_n) = f^d(x_1, x_2, \ldots, x_n)$, then $f(x_1, x_2, \ldots, x_n)$ is called a **self-dual function**.

For example, when $f(x_1, x_2) = x_1 \vee x_2$ is given, we have $f^d(x_1, x_2) = \bar{f}(\bar{x}_1, \bar{x}_2) = \overline{\bar{x}_1 \vee \bar{x}_2}$. This is equal to x_1x_2 by the first identity of Theorem 24.2. In other words, $f^d(x_1, x_2) = x_1x_2$. The function $x_1x_2 \vee x_2x_3 \vee x_1x_3$ is self-dual, as can be seen by applying the two identities of Theorem 24.2.

Notice that, if f^d is the dual of f, the dual of f^d is f.

The concept of a dual function has many important applications. For example, it is useful in the conversion of networks with different types of gates, as in Fig. 24.2, where the replacement of the AND and OR gates in Fig. 24.2(a) by OR and AND gates, respectively, yields the output function f^d in Fig. 24.2(b), which is dual to the output f of Fig. 24.2(a).

As will be explained in a later chapter, a logic gate in CMOS is another important application example of the concept of "dual," where **CMOS** stands for complementary MOS and is a **logic family**, i.e., a type of transistor circuit for realizing a logic gate. Duality is utilized for reducing the power consumption of a logic gate in CMOS.

The following theorem shows a more convenient method of computing the dual of a function than direct use of Definition 24.3.

(a)A network with AND and OR gates

(b)A network with AND and OR gates interchanged

FIGURE 24.2 Duality relation between two networks.

Theorem 24.3: Generalized De Morgan's Theorem — Let $f(x_1, x_2, \ldots, x_n)$ be a function expressed by "$\vee$" and "$\cdot$" (and possibly also by parentheses and the constants 0 and 1). Let $g(x_1, x_2, \ldots, x_n)$ be a function that is obtained by replacing every "$\vee$" and "$\cdot$" by "$\cdot$" and "$\vee$", respectively, throughout the logic expression of $f(x_1, x_2, \ldots, x_n)$ (and also, if 0 or 1 is contained in the original expression, by replacing 0 and 1 by 1 and 0, respectively). Then,

$$f^d(x_1, x_2, \ldots, x_n) = g(x_1, x_2, \ldots, x_n)$$

For example, when $f(x_1, x_2) = x_1 \vee x_2 \cdot \bar{x}_3$ is given, $f^d = x_1 \cdot (x_2 \vee \bar{x}_3)$ is obtained by this theorem. Here, notice that in f, the calculation of $\cdot$ precedes that of $\vee$; and in f^d, the $\vee$ must correspondingly be calculated first [thus, the parentheses are placed as $(x_2 \vee \bar{x}_3)$]. When $f = x_1 \vee 0 \cdot x_2 \cdot \bar{x}_3$ is given, $f^d = x_1 \cdot (1 \vee x_2 \vee \bar{x}_3)$ results by this theorem. When $f = x_1 \vee 1 \cdot x_2 \cdot \bar{x}_3$ is given, $f^d = x_1 \cdot (0 \vee x_2 \vee \bar{x}_3)$ results.

For example, the dual of $x_1 \vee x_2 x_3$ is $\overline{\bar{x}_1 \vee \bar{x}_2 \bar{x}_3}$ according to Definition 24.3, which is a somewhat complicated expression. But by using the generalized De Morgan's theorem, we can immediately obtain the expression without bars, $x_1 \cdot (x_2 \vee x_3) = x_1 x_2 \vee x_1 x_3$.

24.2 Implication Relations and Prime Implicants

In this section, we discuss the algebraic manipulation of logic expressions; that is, how to convert a given logic expression into others. This is very useful for simplification of logic expression. Although simplification of a logic expression based on a Karnaugh map, which will be discussed in Chapter 25, is convenient in many cases, algebraic manipulation is more convenient in many other situations.[3]

Definition 24.4: Let two logic functions be $f(x_1, x_2, \ldots, x_n)$ and $g(x_1, x_2, \ldots, x_n)$. If every vector $(x_1, x_2, \ldots, x_n)$ satisfying $f(x_1, x_2, \ldots, x_n) = 1$ satisfies also $g(x_1, x_2, \ldots, x_n) = 1$ but the converse does not necessarily hold, we write

$$f(x_1, x_2, \ldots, x_n) \subseteq g(x_1, x_2, \ldots, x_n) \tag{24.1}$$

and we say that f **implies** g. In addition, if there exists a certain vector $(x_1, x_2, \ldots, x_n)$ satisfying simultaneously $f(x_1, x_2, \ldots, x_n) = 0$ and $g(x_1, x_2, \ldots, x_n) = 1$, we write

$$f(x_1, x_2, \ldots, x_n) \subset g(x_1, x_2, \ldots, x_n) \tag{24.2}$$

and we say that f **strictly implies** g (some authors use different symbols instead of $\subset$). Therefore, Eq. 24.1 means $f(x_1, x_2, \ldots, x_n) \subset g(x_1, x_2, \ldots, x_n)$ or $f(x_1, x_2, \ldots, x_n) = g(x_1, x_2, \ldots, x_n)$. These relations are called **implication relations**. The left- and right-hand sides of Eq. (24.1) or (24.2) are called **antecedent** and **consequent**, respectively. If an implication relation holds between f and g. That is, if $f \subseteq g$ or $f \supseteq g$ holds, f and g are said to be **comparable** (more precisely, "$\subseteq$-comparable" or "implication-comparable"). Otherwise, they are **incomparable**.

When two functions, f and g, are given, we can find by the following methods at least whether or not there exists an implication relation between f and g; for example, using a truth table for f and g, directly based on Definition 24.4. If and only if there is no row in which $f = 1$ and $g = 0$, the implication relation $f \subseteq g$ holds. Furthermore, if there is at least one row in which $f = 0$ and $g = 1$, the relation is tightened to $f \subset g$. Table 24.1 shows the truth table for $f = x_1\bar{x}_3 \vee \bar{x}_1x_2x_3$ and $g = x_1 \vee x_2$. There is no row with $f = 1$ and $g = 0$, so $f \subseteq g$ holds. Furthermore, there is a row with $f = 0$ and $g = 1$, so the relation is actually $f \subset g$.

TABLE 24.1 Example for $f \subseteq g$

x_1	x_2	x_3	f	g
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	1	1
1	0	1	0	1
1	1	0	1	1
1	1	1	0	1

Although "g implies f" means "if $g = 1$, then $f = 1$," it is to be noticed that "g does not imply f" does not necessarily mean "f implies g" but does mean either "f implies g" or "g and f are incomparable." In other words, it does mean "if $g = 1$, then f becomes a function other than the constant function which is identically equal to 1." (As a special case, f could be identically equal to 0.) Notice that "g does not imply f" does not necessarily mean "if $g = 0$, then $f = 0$."

Definition 24.5: An **implicant** of a logic function f is a term that implies f.

For example, x_1, x_2, x_1x_2, , and $x_1x_3\bar{x}_2$ are examples of implicants of the function $x_1 \vee x_2$. But $\bar{x}_1\bar{x}_2$ is not. Notice that x_1x_3 is an implicant of $x_1 \vee x_2$ even though $x_1 \vee x_2$ is independent of x_3. (Notice that every product of an implicant of f with any dummy variables is also an implicant of f. Thus, f has an infinite number of implicants.) But x_1x_2 is not an implicant of $f = x_1x_3 \vee \bar{x}_2$ because x_1x_2 does not imply f. (When $x_1x_2 = 1$, we have $f = x_3$, which can be 0. Therefore, even if $x_1x_2 = 1$, f may become 0.) Some implicants are not obvious from a given expression of a function. For example, $x_1x_2 \vee \bar{x}_1x_2$ has implicants x_2x_3 and $x_2x_3x_4$. Also, $x_1\bar{x}_2 \vee x_2x_3 \vee \bar{x}_1x_3$ has an implicant x_3 because, if $x_3 = 1$, $x_1\bar{x}_2 \vee x_2x_3 \vee \bar{x}_1x_3$ becomes $x_1\bar{x}_2 \vee x_2 \vee \bar{x}_1 = x_1 \vee x_2 \vee \bar{x}_1 = (x_1 \vee \bar{x}_1) \vee x_2 = 1 \vee x_2$, which is equal to 1.

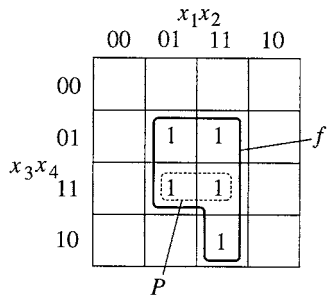
Definition 24.6: A term P is said to **subsume** another term Q if all the literals of Q are contained among those of P. If a term P which subsumes another term Q contains literals that Q does not have, P is said to **strictly subsume** Q.

For example, term $x_1\bar{x}_2x_3\bar{x}_4$ subsumes $x_1x_3\bar{x}_4$ and also itself. More precisely speaking, $x_1\bar{x}_2x_3\bar{x}_4$ strictly subsumes $x_1x_3\bar{x}_4$. Notice that Definition 24.6 can be equivalently stated as follows: "A term P is said to subsume another term Q if P implies Q; that is, $P \subseteq Q$. Term P strictly subsumes another term Q if $P \subset Q$."

Notice that when we have terms P and Q, we can say, "P implies Q" or, equivalently, "P subsumes Q." But the word "subsume" is ordinarily not used in other cases, except for comparing two alterms (as we will see in Section 25.4). For example, when we have functions f and g that are not in single terms, we usually do not say "f subsumes g."

On a Karnaugh map, if the loop representing a term P (always a single rectangular loop consisting of 2^i 1-cells because P is a product of literals) is part of the 1-cells representing function f, or is contained in the loop representing a term Q, P implies f or subsumes Q, respectively. Figure 24.3 illustrates this. Conversely, it is easy to see that, if a term P, which does not contain any dummy variables of f, implies f, the loop for P must consist of some 1-cells of f, and if a term P, which does not contain any dummy variables of another term Q, implies Q, the loop for P must be inside the loop for Q.

(a) Term P implies a function f.



(b) Term P subsumes a term Q.

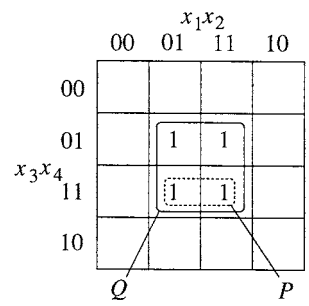


FIGURE 24.3 Comparison of "imply" and "subsume."

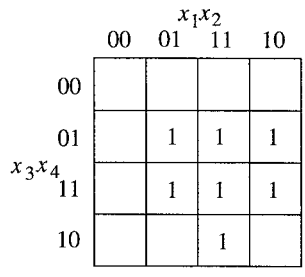
The following concept of "prime implicant" is useful for deriving a simplest disjunctive form for a given function f (recall that logic expressions for f are not unique) and consequently for deriving a simplest logic network realizing f.

Definition 24.7: A **prime implicant** of a given function f is defined as an implicant of f such that no other term subsumed by it is an implicant of f.

For example, when $f = x_1x_2 \vee \bar{x}_1x_3 \vee x_1x_2x_3 \vee \bar{x}_1x_2x_3$ is given, x_1x_2, $\bar{x}_1x_3$, and x_2x_3 are prime implicants. But $x_1x_2x_3$ and $\bar{x}_1x_2x_3$ are not prime implicants, although they are implicants (i.e., if any of them is 1, then $f = 1$). Prime implicants of a function f can be obtained from other implicants of f by stripping off unnecessary literals until further stripping makes the remainder no longer imply f. Thus, x_2x_3 is a prime implicant of $x_1x_2 \vee \bar{x}_1x_3$, and $x_2x_3x_4$ is an implicant of this function but not a prime implicant. As seen from this example, some implicants, such as x_2x_3, and accordingly some prime implicants are not obvious from a given expression of a function. Notice that, unlike implicants, **a prime implicant cannot contain a literal of any dummy variable of a function**.

On a Karnaugh map, all prime implicants of a given function f of at least up to four variables can be easily found. As is readily seen from Definition 24.7, each rectangular loop that consists of 2^i 1-cells, with i chosen as large as possible, is a prime implicant of f. If we find all such loops, we will have found all prime implicants of f. Suppose that a function f is given as shown in Fig. 24.4(a). Then, the prime implicants are shown in Fig. 24.4(b). In this figure, we cannot make the size of the rectangular loops any bigger. (If we increase the size of any one of these loops, the new rectangular loop will contain a number of 1-cells that is not 2^i for any i, or will include one or more 0-cells.)

(a) A function f



(b) Prime implicants

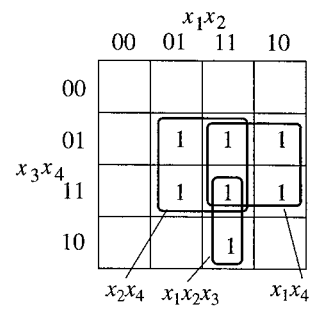


FIGURE 24.4 Expression of prime implicants on Karnaugh maps.

Consensus

Next, let us systematically find all prime implicants, including those not obvious, for a given logic function. To facilitate our discussion, let us define a consensus.

Definition 24.8: Assume that two terms, P and Q, are given. If there is exactly one variable, say x, appearing noncomplemented in one term and complemented in the other — in other words, if $P = xP'$ and $Q = \bar{x}\,Q'$ (no other variables appear complemented in either P' or Q', and noncomplemented in the other) — then the product of all literals except the literals of x, that is, $P'Q'$ with duplicates of literals deleted, is called the **consensus** of P and Q.

For example, if we have two terms, $x_1\bar{x}_2x_3$ and $\bar{x}_1\bar{x}_2x_4\bar{x}_5$, the consensus is $\bar{x}_2x_3x_4\bar{x}_5$. But $x_1\bar{x}_2x_3$ and $\bar{x}_1x_2x_4\bar{x}_5$ do not have a consensus because two variables, x_1 and x_2, appear noncomplemented and complemented in these two terms.

A consensus can easily be shown on a Karnaugh map. For example, Fig. 24.5 shows a function $f = x_1x_2 \vee \bar{x}_1x_4$. In addition to the two loops shown in Fig. 24.5(a), which corresponds to the two prime implicants, x_1x_2 and $\bar{x}_1x_4$, of f, this f can have another rectangular loop, which consists of 2^i 1-cells with i chosen as large as possible, as shown in Fig. 24.5(b). This third loop, which represents x_2x_4, the consensus of x_1x_2 and $\bar{x}_1x_4$, intersects the two loops in Fig. 24.5(a) and is contained within the 1-cells that represent x_1x_2 and $\bar{x}_1x_4$. This is an important characteristic of a loop representing a consensus. Notice that these three terms, x_2x_4, x_1x_2, and $\bar{x}_1x_4$, are prime implicants of f. **When rectangular loops of 2^i 1-cells are adjacent (not necessarily exactly in the same row or column), the consensus is a rectangular loop of 2^i 1-cells, with i chosen as large as possible, that intersects and is contained within these loops. Therefore, if we obtain all largest possible rectangular loops of 2^i 1-cells, we can obtain all prime implicants, including consensuses, which intersect and are contained within other pairs of loops.** Sometimes, a consensus term can be obtained from a pair consisting of another consensus and a term, or a pair of other consensuses that do not appear in a given expression. For example, $x_1\bar{x}_2$ and x_2x_3 of $x_1\bar{x}_2 \vee x_2x_3 \vee \bar{x}_1x_3$ yield consensus x_1x_3, which in turn yields consensus x_3 with $\bar{x}_1x_3$. Each such consensus is also obtained among the above largest possible rectangular loops.

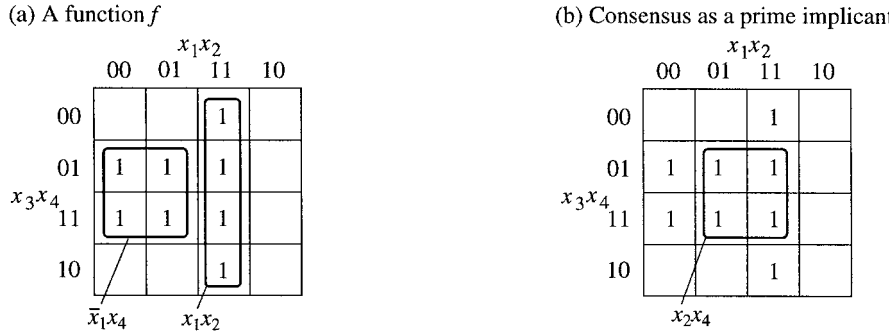


FIGURE 24.5 Expression of a consensus on Karnaugh maps.

As we can easily prove, **every consensus that is obtained from terms of a given function f implies f.** In other words, every consensus generated is an implicant of f, although not necessarily a prime implicant.

Derivation of All Prime Implicants from a Disjunctive Form

The derivation of all prime implicants of a given function f is easy, using a Karnaugh map. If, however, the function has five or more variables, the derivation becomes increasingly complicated on a Karnaugh

map. Therefore, let us discuss an algebraic method, which is convenient for implementation in a computer program, although for functions of many variables even algebraic methods are too time consuming and we need to resort to heuristic methods.

The following algebraic method to find all prime implicants of a given function, which Tison[4,5] devised, is more efficient than the previously known **iterated-consensus method**, which was proposed for the first time by Blake in 1937.[2]

Definition 24.9: Suppose that p products, $A_1, \ldots, A_p$, are given. A variable such that only one of its literals appears throughout $A_1, \ldots, A_p$ is called a **unate variable**. A variable such that both of its literals appear in $A_1, \ldots, A_p$ is called a **binate variable**.

For example, when $x_1\bar{x}_2$, $\bar{x}_1x_3$, $x_3\bar{x}_4$, and $\bar{x}_2x_4$ are given, x_1 and x_4 are binate variables, since x_1 and x_4 as well as their complements, $\bar{x}_1$ and $\bar{x}_4$ appear, and x_2 and x_3 are unate variables.

Procedure 24.1: The Tison Method — Derivation of All Prime Implicants of a Given Function

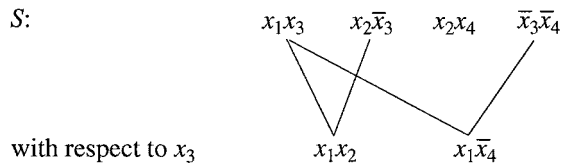
Assume that a function f is given in a disjunctive from $f = P \vee Q \vee \ldots \vee T$, where $P, Q, \ldots, T$ are terms, and that we want to find all prime implicants of f. Let S denote the set $\{P, Q, \ldots, T\}$.

1. Among $P, Q, \ldots, T$ in set S, first delete every term subsuming any other term. Among all binate variables, choose one of them.

For example, when $f = x_1x_2x_4 \vee x_1x_3 \vee x_2\bar{x}_3 \vee x_2x_4 \vee \bar{x}_3\bar{x}_4$ is given, delete $x_1x_2x_4$, which subsumes x_2x_4. The binate variables are x_3 and x_4. Let us choose x_3 first.

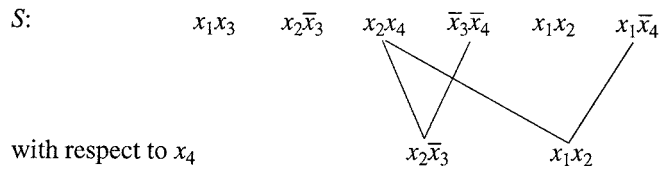
2. For each pair of terms, that is, one with the complemented literal of the chosen variable and the other with the noncomplemented literal of that variable, generate the consensus. Then add the generated consensus to S. From S, delete every term that subsumes another.

For our example, x_3 is chosen as the first binate variable. Thus we get consensus x_1x_2 from pair x_1x_3 and $x_2\bar{x}_3$, and consensus $x_1\bar{x}_4$ from pair x_1x_3 and $\bar{x}_3\bar{x}_4$. None subsumes another. Thus, S, the set of prime implicants, becomes x_1x_3, $x_2\bar{x}_3$, x_2x_4, $\bar{x}_3\bar{x}_4$, x_1x_2, and $x_1\bar{x}_4$.



3. Choose another binate variable in the current S. Then go to Step 2. If all binate variables are tried, go to Step 4.

For the above example, for the second iteration of Step 2, we choose x_4 as the second binate variable and generate two new consensuses as follows:



But when they are added to S, each one subsumes some term contained in S. Therefore, they are eliminated.

4. The procedure terminates because all binate variables are processed, and all the products in S are desired prime implicants.

The last expression is called the **complete sum** or **the all-prime-implicant disjunction**. The complete sum is the first important step in deriving the most concise expressions for a given function. □

Generation of prime implicants for an incompletely specified function, which is more general than the case of completely specified function described in Procedure 24.1, is significantly speeded up with the use of BDD (described in Chapters 24.1 and 24.4) by Coudert and Madre.[1]

References

1. Coudert, O. and J. C. Madre, "Implicit and incremental computation of primes and essential primes of Boolean functions," *Design Automation Conf.*, pp. 36-39, 1992.
2. Brown, F. M., "The origin of the iterated consensus," *IEEE Tr. Comput.*, p. 802, Aug. 1968.
3. Muroga, S., *Logic Design and Switching Theory*, John Wiley & Sons, 1979 (now available from Krieger Publishing Co.).
4. Tison, P., Ph.D. dissertation, Faculty of Science, University of Grenoble, France, 1965.
5. Tison, P., "Generalization of consensus theory and application to the minimization of Boolean functions," *IEEE Tr. Electron. Comput.*, pp. 446-456, Aug. 1967.

Muroga, S. "Simplification of Logic Expressions"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

25

Simplification of Logic Expressions

Saburo Muroga
University of Illinois at Urbana-Champaign

25.1 Minimal Sums

In this chapter, using only AND and OR gates, we will synthesize a two-level logic network. This is the fastest network, if we assume that every gate has the same delay, since the number of levels cannot be reduced further unless a given function can be realized with a single AND or OR gate. If there is more than one such network, we will derive the simplest network. Such a network has a close relation to important concepts in switching algebra, that is, irredundant disjunctive forms and minimal disjunctive forms (or minimal sums), as we discuss in the following.

Now let us explore basic properties of logic functions.

For many functions, some terms in their complete sums are redundant. In other words, even if we eliminate some terms from a complete sum, the remaining expression may still represent the original function for which the complete sum was obtained. Thus, we have the following concept.

Definition 25.1: An **irredundant disjunctive form** for f (sometimes called an irredundant sum-of-products expression or an irredundant sum) is a disjunction of prime implicants such that removal of any of the prime implicants makes the remaining expression not express the original f.

An irredundant disjunctive form for a function is not necessarily unique.

Definition 25.2: Prime implicants that appear in every irredundant disjunctive form for f are called **essential prime implicants** of f. Prime implicants that do not appear in any irredundant disjunctive form for f are called **absolutely eliminable prime implicants** of f. Prime implicants that appear in some irredundant disjunctive forms for f but not in all are called **conditionally eliminable prime implicants** of f.

Different types of prime implicants are shown in the Karnaugh map in Fig. 25.1 for a function $f = \bar{x}_1 x_3 \bar{x}_4 \vee \bar{x}_1 x_2 x_3 \vee \bar{x}_1 x_2 x_4 \vee x_2 x_3 x_4 \vee x_1 x_3 x_4 \vee x_1 \bar{x}_2 x_4$. Here, $\bar{x}_1 x_2 \bar{x}_4$, $\bar{x}_1 x_2 x_4$, and $x_1 \bar{x}_2 x_4$ are essential prime implicants, $x_2 x_3 x_4$ and $x_1 x_3 x_4$ are conditionally eliminable prime implicants and $\bar{x}_1 x_2 x_3$ is absolutely eliminable prime implicant.

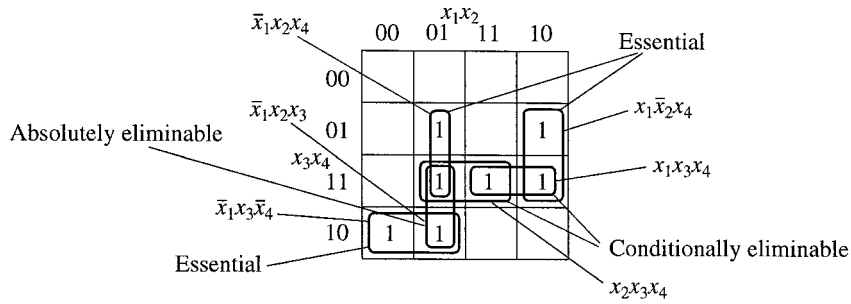


FIGURE 25.1 Different types of prime implicants.

The concepts defined in the following play an important role in switching theory.

Definition 25.3: Among all irredundant disjunctive forms of f, those with a minimum number of prime implicants are called **minimal sums** (some authors call them as "sloppy minimal sum"). Among minimal sums, those with a minimum number of literals are called **absolute minimal sums** for f.

Irredundant disjunctive forms for a given function f can be obtained by deleting prime implicants one by one from the complete sum in all possible ways, after obtaining the complete sum by the Tison method discussed in Chapter 24. Then the minimal sums can be found among the irredundant disjunctive forms. Usually, however, this approach is excessively time-consuming because a function has many prime implicants when the number of variables is very large. When the number of variables is too large, derivation of a minimal sum is practically impossible.

Later, we will discuss efficient methods to derive minimal sums within reasonable computation time when the number of variables is few.

25.2 Derivation of Minimal Sums by Karnaugh Map

Because of its pictorial nature, a Karnaugh map is a very powerful tool for deriving manually all prime implicants, irredundant disjunctive forms, minimal sums, and also absolute minimal sums. Algebraic concepts such as prime implicants and consensuses can be better understood on a map.

One can derive all prime implicants, irredundant disjunctive forms, and minimal sums by the following procedures on Karnaugh maps, when the number of variables is small enough for the map to be manageable.

Procedure 25.1: Derivation of Minimal Sums on a Karnaugh Map

This procedure consists of three steps:

1. On a Karnaugh map, encircle all the 1-cells with rectangles (also squares as a special case), each of which consists of 2^i 1-cells, choosing i as large as possible, where i is a non-negative integer. Let us call these loops **prime implicant loops**, since they correspond to prime implicants in the case of the Karnaugh map for four or fewer variables. (In the case of a five- or six-variable map, the correspondence is more complex, as will be explained later.) Examples are shown in Fig. 25.2(a).
2. Cover all the 1-cells with prime-implicant loops so that removal of any loops leaves some 1-cells uncovered. These sets of loops represent **irredundant disjunctive forms**. Figs. 25.2(b) through 25.2(e) represent four irredundant disjunctive forms, obtained by choosing the loops in Fig. 25.2(a) in four different ways. For example, if the prime-implicant loop $x_1\bar{x}_3x_4$ is omitted in Fig. 25.2(b), the 1-cells for $(x_1, x_2, x_3, x_4) = (1\ 1\ 0\ 1)$ and $(1\ 0\ 0\ 1)$ are not covered by any loops.
3. From the sets of prime implicant loops formed in Step 2 for irredundant disjunctive forms, choose the sets with a minimum number of loops. Among these sets, the sets that contain as many of the largest loops as possible (a larger loop represents a product of fewer literals) represent minimal sums. Figure 25.2(c) expresses the unique minimal sum for this function, since it contains one less loop than Fig. 25.2(b), (d), or (e).

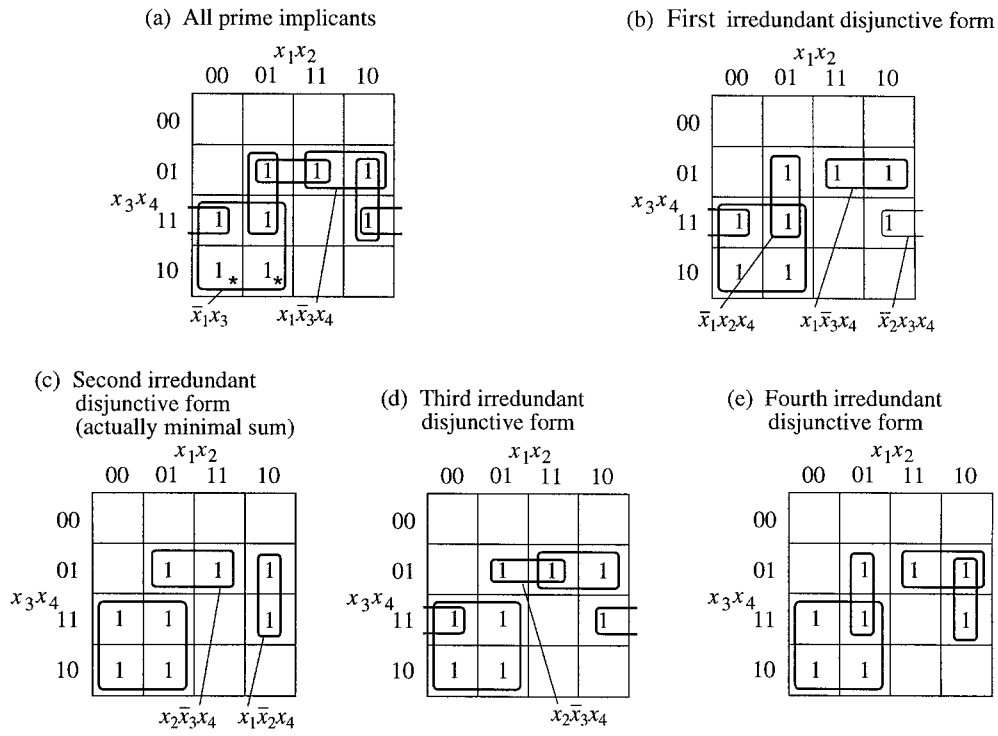


FIGURE 25.2 Irredundunt disjunctive forms and a minimal sum.

It is easy to see, from the definitions of prime implicants, irredundant disjunctive forms, and minimal sums, why Procedure 25.1 works.

When we derive irredundant disjunctive forms or minimal sums by Procedure 25.1, the following property is useful. When we find all prime-implicant loops by Step 1, some 1-cells may be contained in only one loop. Such 1-cells are called **distinguished 1-cells** and are labeled with asterisks. (The 1-cells shown with asterisks in Fig. 25.2(a) are distinguished 1-cells.) A prime implicant loop that contains distinguished 1-cells is called an **essential prime implicant loop**. The corresponding prime implicant is an essential prime implicant, as already defined. In every irredundant disjunctive form and every minimal sum to be found in Step 2 and 3, respectively, essential prime implicants must be included, since each 1-cell on the map must be contained in at least one prime implicant loop and distinguished 1-cells can be contained only in essential prime implicant loops. Hence, if essential prime implicant loops are first identified and chosen, Procedure 25.1 is quickly processed.

Even if the don't-care condition d is contained in some cells, prime implicants can be formed in the same manner, by simply regarding d as being 1 or 0 whenever necessary to draw a greater prime implicant loop. For example, in Fig. 25.3, we can draw a greater rectangular loop by regarding two d's as being 1. One d is left outside and is regarded as being 0. We need not consider loops consisting of d's only.

Maps for Five and Six Variables

The Karnaugh map is most useful for functions of four or fewer variables, but it is often useful also for functions of five or six variables. A map for five variables consists of two four-variable maps, as shown in Fig. 25.4, one for each value of the first variable. A map for six variables consists of four four-variable maps, as shown in Fig. 25.5, one for each combination of values of the first two variables. Note that **the four maps in Fig. 25.5 are arranged so that binary numbers represented by x_1 and x_2 differ in only one bit horizontally and vertically** (the map for $x_1 = x_2 = 1$ goes to the bottom right-hand side).

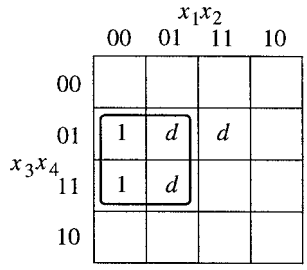


FIGURE 25.3 A map with don't-care conditions.

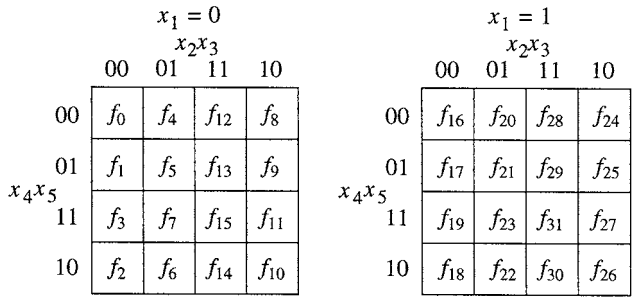


FIGURE 25.4 Karnaugh map for five variables.

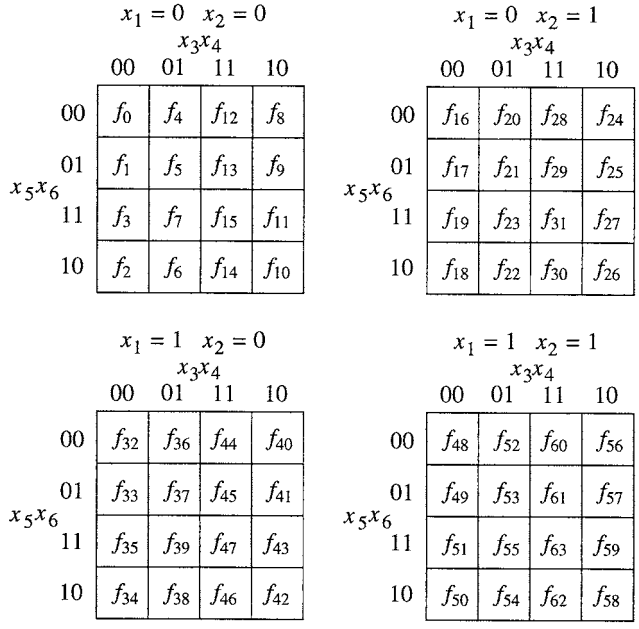


FIGURE 25.5 Karnaugh map for six variables.

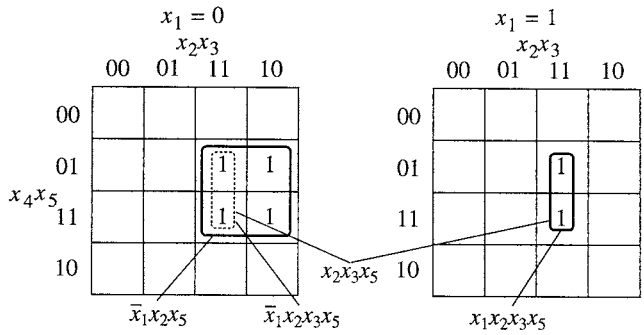
In a five-variable map, 1-cells are combined in the same way as in the four-variable case, with the additional feature that rectangular loops of 2^i 1-cells that are on different four-variable maps can be combined to form a greater loop replacing the two original loops only if they occupy the same relative position on their respective four-variable maps. Notice that these loops may be inside other loops in each four-variable map. For example, if f_{15} and f_{31} are 1, they can be combined; but even if $f_{15} = f_{29} = 1$, f_{15} and f_{29} cannot. In a six-variable map, only 1-cells in two maps that are horizontally or vertically adjacent can be combined if they are in the same relative positions. In Fig. 25.5, for example, if f_5 and f_{37} are 1, they can be combined; but even if f_5 and f_{53} are 1, f_5 and f_{53} cannot. Also, four 1-cells that occupy the same relative positions in all four-variable maps can be combined as representing a single product. For example, f_5, f_{21}, f_{37}, and f_{53} can be combined if they are 1.

In the case of a five-variable map, we can find prime implicant loops as follows.

Procedure 25.2: Derivation of Minimal Sums on a Five-Variable Map

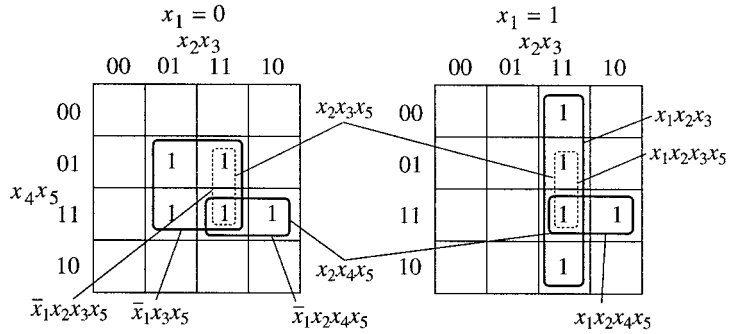
1. Unlike Step 1 of Procedure 25.1 for a function of four variables, this step requires the following two substeps to form prime implicant loops:
   a. On each four-variable map, encircle all the 1-cells with rectangles, each of which consists of 2^i 1-cells, choosing the number of 1-cells contained in each rectangle as large as possible. Unlike the case of Procedure 25.1, **these loops are not necessarily prime implicant loops** because they may not represent prime implicant, depending on the outcome of substep b.

In Figs. 25.6 and 25.7, for example, loops formed in this manner are shown with solid lines.



The prime implicants of this function are $\bar{x}_1x_2x_5$ and $x_2x_3x_5$.

FIGURE 25.6 Prime implicant loops on a map for five variables.



The prime implicants of this function are $\bar{x}_1x_3x_5$, $x_2x_3x_5$, $x_2x_4x_5$, and $x_1x_2x_3$.

FIGURE 25.7 Prime implicant loops on a map for five variables.

b. On each four-variable map, encircle all the 1-cells with rectangles, each of which consists of 2^i 1-cells in exactly the same relative position on the two maps, choosing i as great as possible. Then connect each pair of the corresponding loops with an arc. On each four-variable map, some of these loops may be inside some loops formed in substep a.

In Fig. 25.6, one pair of loops that is in the same relative position is newly formed. One member of the pair, shown in a dotted line, is contained inside a loop formed in substep a. The pair is connected with an arc. The other loop coincides with a loop formed in substep a. In Fig. 25.7, two pairs of loops are formed: one pair is newly formed, as shown in dotted lines, and the second pair is the connection of loops formed in substep a.

The loops formed in substep b and also those formed in substep a but not contained in any loop formed in substep b are **prime implicant loops,** since they correspond to prime implicants.

In Fig. 25.6, the loop formed in substep a, which represents $\bar{x}_1 x_2 x_3 x_5$, is contained in the prime implicant loop formed in substep b, which represents $x_2 x_3 x_5$. Thus, the former loop is not a prime implicant loop, and consequently $\bar{x}_1 x_2 x_3 x_5$ is not a prime implicant.

2. Processes for deriving irredundant disjunctive forms and minimal sums are the same as Steps 2 and 3 of Procedure 25.1. □

In the case of six-variable map, the derivation of prime implicant loops requires more comparisons of four-variable maps, as follows.

Procedure 25.3: Derivation of Minimal Sums on a Six-Variable Map

1. Derivation of all prime-implicant loops requires the following three substeps:
   a. On each four-variable map, encircle all the 1-cells with rectangles, each of which consists of 2^i 1-cells, choosing i as great as possible.
   b. Find the rectangles (each of which consists of 2^i 1-cells) occupying the same relative position on every two adjacent four-variable maps, choosing i as great as possible. (Two maps in diagonal positions are not adjacent.) Thus, we need four comparisons of two maps (i.e., upper two maps, lower two maps, left two maps, and right two maps).
   c. Then find the rectangles (each of which consists of 2^i 1-cells) occupying the same relative position on all four-variable maps, choosing i as great as possible. **Prime implicant loops** are loops formed at substeps c, loops formed at b but not contained inside those at c, and loops formed at a but not contained inside those formed at b or c.
2. Processes for deriving irredundant disjunctive forms and minimal sums are the same as Steps 2 and 3 of Procedure 25.1. □

An example is shown in Fig. 25.8.

Irredundant disjunctive forms and minimal sums are derived in the same manner as in the case of four variables.

Procedures 25.1, 25.2, and 25.3 can be extended to the cases of seven or more variables with increasing complexity. It is usually hard to find a minimal sum, however, because each prime implicant loop consists of 1-cells scattered in many maps.

25.3 Derivation of Minimal Sums for a Single Function by Other Means

Derivation of a minimal sum for a single function by Karnaugh maps is convenient for manual processing because designers can know the nature of logic networks better than automated minimization, to be described in Chapter 27, but its usefulness is limited to functions of four or five variables.

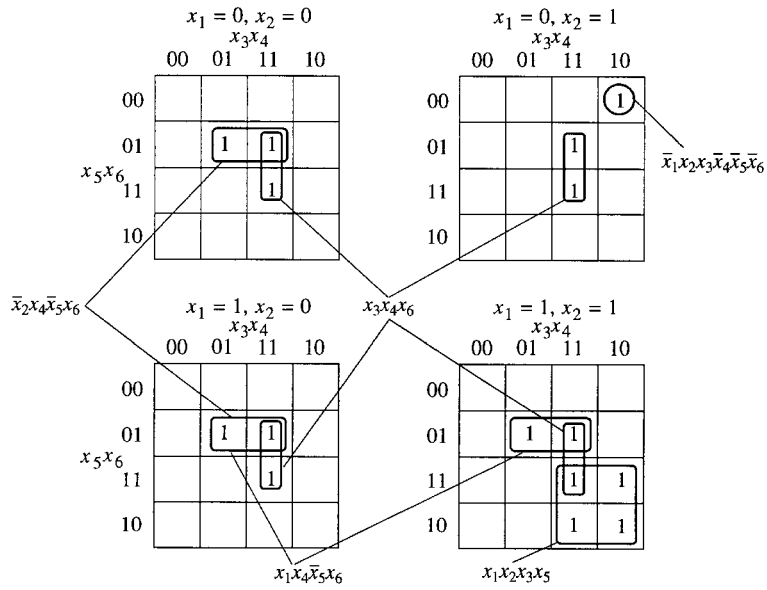


FIGURE 25.8 Prime-implicant loops on a map for six variables.

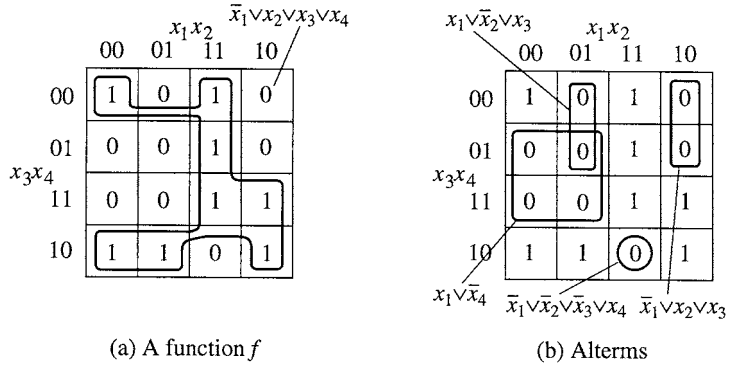
There are several methods for derivation of a minimal sum for a single function. A minimal sum can be found by forming a so-called a covering table where the minterms of a given function *f* are listed on the horizontal coordinate and all the prime implicants of *f* are listed on the vertical coordinate. A minimal sum can be described as a solution, that is, a minimal set of prime implicants that covers all the minterms.[2] The feasibility of this approach based on the covering table is limited by the number of minterms and prime implicants rather than the number of variables, which is the limiting factor for the feasibility of the derivation of minimal sums based on Karnaugh maps. This approach based on the table can be converted to an algebraic method, with prime implicants derived by the Tison method, as described by Procedure 4.6.1 in Ref. 3. This is much faster than the approach based on the table. Another approach is generation of irredundant disjunctive forms and then derive a minimal sum among them.[4,5] The feasibility of this approach is limited by the number of consensuses rather than the number of variables, minterms, or prime implicants.

As the number of variables, minterms, or prime implicants increases, the derivation of absolute minimal sums or even prime sums becomes too time-consuming, although the enhancement of the feasibility has been explored.[1] When too time-consuming, we need to resort to heuristic minimization, as will be described in Chapter 27 and Chapter 42, Section 42.3.

25.4 Prime Implicates, Irredundant Conjunctive Forms, and Minimal Products

The "implicates," "irredundant conjunctive forms," and "minimal products," which can be defined all based on the concept of conjunctive form, will be useful for deriving a minimal network that has OR gates in the first level and one AND gate in the second level.

First, let us represent the maxterm expansion of a given function *f* on a Karnaugh map. Unlike the map representation of the minterm expansion of *f*, where each minterm contained in the expansion is represented by a 1-cell on a Karnaugh map, each maxterm is represented by a 0-cell. Suppose that a function *f* is given as shown by the 1-cells in Fig. 25.9(a). This function can be expressed in the maxterm expansion:



(a) A function f (b) Alterms

FIGURE 25.9 Representation of a function f on a map with alterms.

$$f = (\bar{x}_1 \vee x_2 \vee x_3 \vee x_4)(\bar{x}_1 \vee x_2 \vee x_3 \vee \bar{x}_4)(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4)(x_1 \vee \bar{x}_2 \vee x_3 \vee x_4)$$
$$(x_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x}_4)(x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)(x_1 \vee x_2 \vee x_3 \vee \bar{x}_4)(x_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4)$$

The first maxterm, $(\bar{x}_1 \vee x_2 \vee x_3 \vee x_4)$, for example, is represented by the 0-cell that has components $(x_1, x_2, x_3, x_4) = (1\ 0\ 0\ 0)$ in the Karnaugh map in Fig. 25.9(a) (i.e., $(\bar{x}_1 \vee x_2 \vee x_3 \vee x_4)$ becomes 0 for $x_1 = 1$, $x_2 = x_3 = x_4 = 0$). Notice that each literal in the maxterm is complemented or noncomplemented, corresponding to 1 or 0 in the corresponding component, respectively, instead of corresponding to 0 or 1 in the case of minterm. All other maxterms are similarly represented by 0-cells. It may look somewhat strange that these 0-cells represent the f expressed by 1-cell on the Karnaugh map in Fig. 25.9(a). But the first maxterm, $(\bar{x}_1 \vee x_2 \vee x_3 \vee x_4)$, for example, assumes the value 0 only for $(x_1, x_2, x_3, x_4) = (1\ 0\ 0\ 0)$ and assumes the value 1 for all other combinations of the components. The situation is similar with other maxterms. Therefore, the conjunction of these maxterms becomes 0 only when any of the maxterms becomes 0. Thus, these 0-cells represent f through its maxterms. This is what we discussed earlier about the maxterm expansion or the Π-decimal specification of f, based on the false vectors of f.

Like the representation of a product on a Karnaugh map, any alterm can be represented by a rectangular loop of 2^i 0-cells by repeatedly combining a pair of 0-cell loops that are horizontally or vertically adjacent in the same rows or columns, where i is a non-negative integer. For example, the two adjacent 0-calls in the same column representing maxterms $(\bar{x}_1 \vee x_2 \vee x_3 \vee x_4)$ and $(\bar{x}_1 \vee x_2 \vee x_3 \vee \bar{x}_4)$ can be combined to form alterm $\bar{x}_1 \vee x_2 \vee x_3$, by deleting literals x_4 and $\bar{x}_4$, as shown in a solid-line loop in Fig. 25.9(b).

The function f in Fig. 25.9 can be expressed in the conjunctive form $f = (\bar{x}_1 \vee x_2 \vee x_3)(x_1 \vee \bar{x}_2 \vee x_3)(x_1 \vee \bar{x}_4)(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4)$, using a minimum number of such loops. The alterms in this expansion are represented by the loops shown in Fig. 25.9(b).

Definition 25.4: An implicate of f is an alterm implied by a function f.

Notice that an implicate's relationship with f is opposite to that of an implicant that implies f.

For example, $(x_1 \vee x_2 \vee x_3)$ and $(x_1 \vee x_2 \vee \bar{x}_3)$ are implicates of function $f = x_1 \vee x_2$, since whenever $f = 1$, both $(x_1 \vee x_2 \vee x_3)$ and $(x_1 \vee x_2 \vee \bar{x}_3)$ become 1. The implication relationship between an alterm P and a function f can sometimes be more conveniently found, however, by using the property "f implies P if $f = 0$ whenever $P = 0$," which is a restatement of the property "f implies P if $P = 1$ whenever $f = 1$" defined earlier (we can easily see this by writing a truth table for f and P). Thus, $(x_1 \vee x_3)$, $(x_1 \vee \bar{x}_2)$, and $(x_1 \vee x_2 \vee x_3)$ are implicates of $f = (x_1 \vee \bar{x}_2)(\bar{x}_2 \vee x_3)$, because when $(x_1 \vee x_3)$, $(x_1 \vee \bar{x}_2)$, or $(x_1 \vee x_2 \vee x_3)$ is 0, f is 0. [For example, when $(x_1 \vee x_3)$ is 0, $x_1 = x_3 = 0$ must hold. Thus, $f = (\bar{x}_2)(x_2) = 0$. Consequently, $(x_1 \vee x_3)$ is an alterm implied by f, although this is not obvious from the given expressions of f and $(x_1 \vee x_3)$.] Also, $(x_1 \vee \bar{x}_2 \vee x_4)$ and $(x_1 \vee x_2 \vee x_3 \vee x_4)$, which contain the literals of a dummy variable x_4 of this f, are implicates of f.

Definition 25.5: An alterm P is said to **subsume** another alterm Q if all the literals in Q are among the literals of P.

The alterm $(x_1 \vee \bar{x}_2 \vee x_3 \vee x_4)$ subsumes $(\bar{x}_2 \vee x_3)$. Summarizing the two definitions of "subsume" for "alterms" in Definition 25.5 and "terms" in Definition 24.6, we have that **"*P* subsumes *Q*" simply means "*P* contains all the literals in *Q*," regardless of whether *P* and *Q* are terms or alterms**. But the relationships between "subsume" and "imply" in the two cases are opposite. If an alterm P subsumes another alterm Q, then $Q \subseteq P$ holds; whereas if a term P subsumes another term Q, then $P \subseteq Q$ holds.

Definition 25.6: A **prime implicate** of a function f is defined as an implicate of f such that no other alterm subsumed by it is an implicate of f.

In other words, if deletion of any literal from an implicate of f makes the remainder not an implicate of f, the implicate is a prime implicate. For example, $(x_2 \vee x_3)$ and $(x_1 \vee x_3)$ are prime implicates of $f = (x_1 \vee \bar{x}_2)(x_2 \vee x_3)(\bar{x}_1 \vee x_2 \vee x_3)$, but $x_1 \vee x_3 \vee x_4$ is not a prime implicate of f, although it is still an implicate of this f. As seen from these examples, some implicates are not obvious from a given conjunctive form of f. Such an example is $x_1 \vee x_3$ in the above example. As a matter of fact, $x_1 \vee x_3$ can be obtained as the consensus by the following Definition 25.7 of two alterms, $(x_1 \vee \bar{x}_2)$ and $(x_2 \vee x_3)$, of f. Also notice that, unlike implicates, prime implicates cannot contain a literal of any dummy variable of f.

On a Karnaugh map, **a loop for an alterm *P* that subsumes another alterm *Q* is contained in the loop for *Q***. For example, in Fig. 25.10, the loop for alterm $(x_1 \vee \bar{x}_3 \vee x_4)$, which subsumes alterm $(x_1 \vee \bar{x}_3)$, is contained in the loop for $x_1 \vee \bar{x}_3$. Thus, **a rectangular loop that consists of 2^i 0-cells, with *i* as large as possible, represents a prime implicate of *f*.**

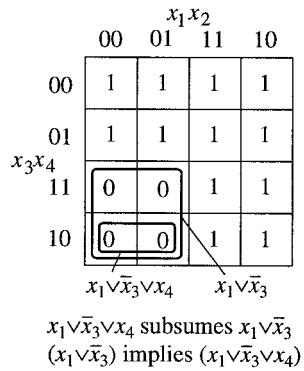


FIGURE 25.10 An alterm that subsumes another alterm.

The consensus of two **alterms**, V and W, is defined in a manner similar to the consensus of two terms.

Definition 25.7: If there is exactly one variable, say x, appearing noncomplemented in one alterm and complemented in the other (i.e., if two alterms, V and W, can be written as $V = x \vee V'$ and $W = \bar{x} \vee W'$, where V' and W' are alterms free of literals of x), the disjunction of all literals except those of x (i.e., $V' \vee W'$ with duplicate literals deleted) is called the **consensus** of the two alterms V and W.

For example, when $V = x \vee y \vee \bar{z} \vee u$ and $W = \bar{x} \vee y \vee u \vee \bar{v}$ are given, their consensus is $y \vee \bar{z} \vee u \vee \bar{v}$.

On a Karnaugh map, a consensus is represented by the largest rectangular loop of 2^i 0-cells that intersects, and is contained within, two adjacent loops of 0-cells that represent two alterms. For example, in Fig. 25.11 the dotted-line loop represents the consensus of two alterms, $(\bar{x}_2 \vee \bar{x}_4)$ and $(\bar{x}_1 \vee x_2 \vee \bar{x}_3)$, which are represented by the two adjacent loops.

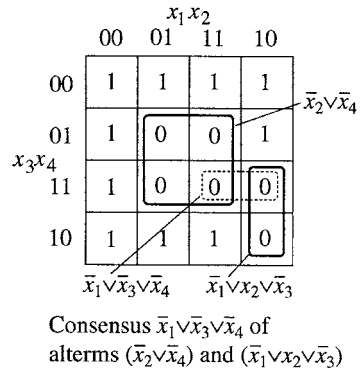


FIGURE 25.11 Consensus of two adjacent alterms.

All prime implicates of a function f can be algebraically obtained from a conjunctive form for f by modifying the Tison method discussed in Procedure 24.1; that is, by using dual operations in the method.

We can define the following concepts, which are dual to irredundant disjunctive forms, minimal sums, absolute minimal sums, essential prime implicants, complete sums, and others.

Definition 25.8: An **irredundant conjunctive form** for a function f is a conjunction of prime implicates such that removal of any of them makes the remainder not express f. The **minimal products** are irredundant conjunctive forms for f with a minimum number of prime implicates. The **absolute minimal products** are minimal products with a minimum number of literals. Prime implicates that appear in every irredundant conjunctive form for f are called **essential prime implicates** of f. **Conditionally eliminable prime implicates** are prime implicates that appear in some irredundant conjunctive forms for f, but not in others. **Absolutely eliminable prime implicates** are prime implicates that do not appear in any irredundant conjunctive form for f. The **complete product** for a function f is the product of all prime implicates of f.

25.5 Derivation of Minimal Products by Karnaugh Map

Minimal products can be derived by the following procedure, based on a Karnaugh map.

Procedure 25.4: Derivation of Minimal Products on a Karnaugh Map

Consider the case of a map for four or fewer variables (cases for five or more variables are similar, using more than one four-variable map).

1. Encircle **0-cells**, instead of 1-cells, with rectangular loops, each of which consists of 2^i 0-cells, choosing i as large as possible. These loops are called **prime implicate loops** because they represent prime implicates (not prime implicants). Examples of prime implicate loops are shown in Fig. 25.12 (including the dotted-line loop). The **prime implicate** corresponding to a loop is formed by making a **disjunction** of literals, instead of a conjunction, **using a noncomplemented variable corresponding to 0 of a coordinate of the map and a complemented variable corresponding to 1.** (Recall that, in forming a prime implicant, variables corresponding to 0's were complemented and those corresponding to 1's were noncomplemented.) Thus, corresponding to the loops of Fig. 25.12, we get the prime implicates, $(x_1 \vee \bar{x}_4)$, $(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$ and $(\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)$.
2. Each **irredundant conjunctive form** is derived by the conjunction of prime implicates corresponding to a set of loops, so that removal of any loop leaves some 0-cells uncovered by loops. An example is the set of two solid-line loops in Fig. 25.12, from which the irredundant conjunctive form $(x_1 \vee \bar{x}_4)(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$ is derived.

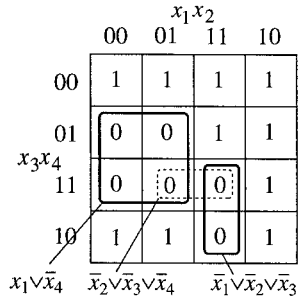


FIGURE 25.12 Prime-implicate loops and the corresponding prime implicates.

3. Among sets of a minimum number of prime implicate loops, the sets that contain as many of the largest loops as possible yield **minimal products**.

The don't-care conditions are dealt with in the same manner as in case of minimal sums. In other words, whenever possible, we can form a larger prime implicate loop by interpreting some *d*'s as 0-cells. Any prime-implicate loops consisting of only *d*-cells need not be formed. □

Procedure 25.4 can be extended to five or more variables in the same manner as Procedures 25.2 and 25.3, although the procedure will be increasingly complex.

References

1. Coudert, O., "Two-Level Logic Minimization: An Overview," *Integration*, vol. 17, pp. 97-140, Oct. 1994.
2. McCluskey, E. J., "Minimization of Boolean functions," *Bell System Tech. J.*, vol. 35, no. 5, pp. 1417-1444, Nov. 1956.
3. Muroga, S., *Logic Design and Switching Theory*, John Wiley & Sons, 1979 (now available from Krieger Publishing Co.).
4. Tison, P., Ph.D. dissertation, Faculty of Science, University of Grenoble, France, 1965.
5. Tison, P., "Generalization of consensus theory and application to the minimization of Boolean functions," *IEEE Tr. Electron. Comput.*, pp. 446-456, Aug. 1967.

Minato, S., Muroga, S."Binary Decision Diagrams"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

26

Binary Decision Diagrams

Shin-ichi Minato
NTT Network Innovation Laboratories

Saburo Muroga
University of Illinois at Urbana-Champaign

26.1 Basic Concepts

Binary decision diagrams (BDDs), which were introduced in Chapter 23, Section 23.1, are a powerful means for computer processing of logic functions because in many cases, with BDDs, smaller memory space is required for storing logic functions and values of functions can be calculated faster than with truth tables or logic expressions. As logic design has been done in recent years with computers, BDDs are extensively used because of these features. BDDs are used in computer programs for automation of logic design, verification (i.e., identifying whether two logic networks represent the identical logic functions), diagnosis of logic networks, simplification of transistor circuits (such as ECL and MOSFET circuits, as explained in Chapters 35, 36, and 37), and other areas, including those not related to logic design.

In this chapter, we discuss the basic data structures and algorithms for manipulating BDDs. Then we describe the variable ordering problem, which is important for the effective use of BDDs. The concept of BDD was devised by Lee in 1959.[15] Binary decision programs that Lee discussed are essentially binary decision diagrams. Then, in 1978, its usefulness for expressing logic functions was shown by Akers.[3,4] But since Bryant[6] developed a reduction method in 1986, it has been extensively used for design automation for logic design and related areas.

From a truth table, we can easily derive the corresponding binary decision diagram. For example, the truth table shown in Fig. 26.1(a) can be converted into the BDD in Fig. 26.1(b). But there are generally many BDDs for a given truth table, that is, the logic function expressed by this truth table. For example, all BDDs shown in Fig. 26.1(c) through (e) represent the logic function that the truth table in Fig. 26.1(a) expresses. Here, note that in each of Figs. 26.1 (b), (c), and (d), the variables appear in the same order and none of them appears more than once in every path from the top node. But in (e), they appear in different orders in different paths. BDDs in Figs. 26.1(b), (c), and (d) are called **ordered BDD**s (or **OBDD**s). But the BDD in Fig. 26.1(e) is called an **unordered BDD**. These BDDs can be reduced into a simple BDD by the following procedure.

In a BDD, the top node is called the **root** that represents the given function $f(x_1, x_2, \ldots, x_n)$. Rectangles that have 1 or 0 inside are called **terminal nodes**. They are also called **1-terminal** and **0-terminal**.

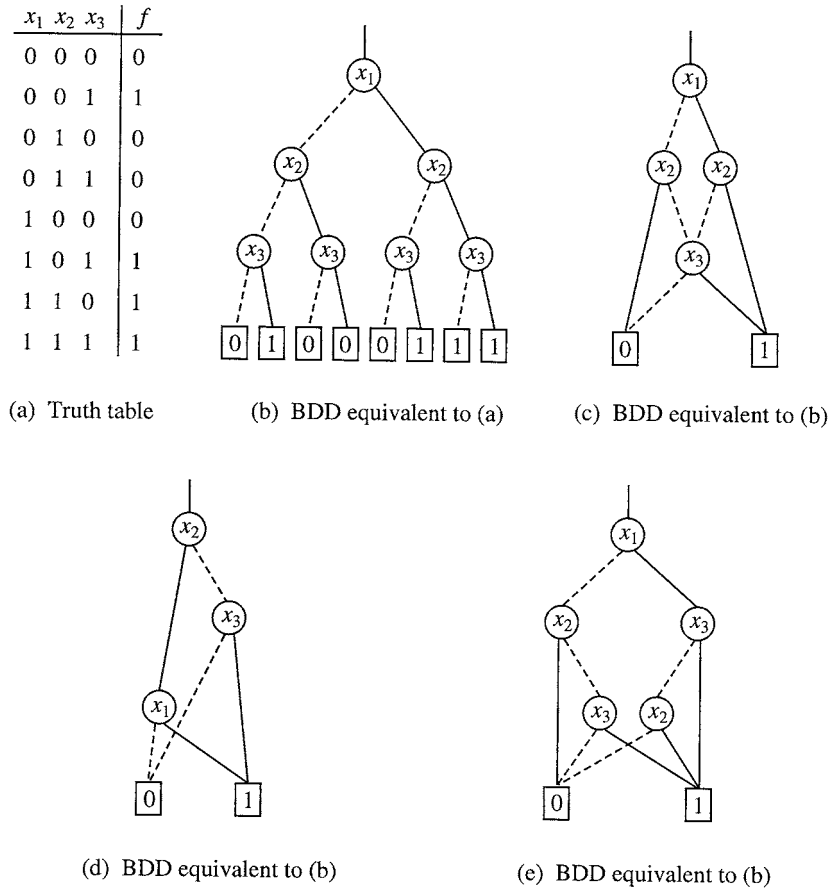


FIGURE 26.1 Truth table and BDDs for $x_1x_2 \vee \bar{x}_2x_3$.

Other nodes are called **non-terminal nodes** denoted by circles with variables inside. They are also called simply **nodes**, differentiating themselves from the 0- and 1-terminals. From a node with x_i inside, two lines go down. The solid line is called **1-edge**, representing $x_i = 1$; and the dotted line is called **0-edge**, representing $x_i = 0$.

In an OBDD, the value of the function f can be evaluated by following a path of edges from the root node to one of the terminal nodes. If the nodes in every path from the root node to a terminal node are assigned with variables, x_1, x_2, x_3, …, and x_n, in this order, then f can be expressed as follows, according to the Shannon expansion (described in Chapter 24). By branching with respect to x_1 from the root node, $f(x_1, x_2, \ldots, x_n)$ can be expanded as follows, where $f(0, x_2, \ldots, x_n)$ and $f(1, x_2, \ldots, x_n)$ are functions that the nodes at the low ends of 0-edge and 1-edge from the root represent, respectively:

$$f(x_1, x_2, \ldots, x_n) = \bar{x}_1 f(0, x_2, \ldots, x_n) \vee x_1 f(1, x_2, \ldots, x_n)$$

Then, by branching with respect to x_2 from each of these two nodes that $f(0, x_2, \ldots, x_n)$ and $f(1, x_2, \ldots, x_n)$ represent, each $f(0, x_2, \ldots, x_n)$ and $f(1, x_2, \ldots, x_n)$ can be expanded as follows:

$$f(0, x_2, \ldots x_n) = \bar{x}_2 f(0, 0, x_3, \ldots, x_n) \vee x_2 f(0, 1, x_3, \ldots, x_n)$$

and

$$f(1, x_2, \ldots, x_n) = \bar{x}_2 f(1, 0, x_3, \ldots, x_n) \vee x_2 f(1, 1, x_3, \ldots, x_n)$$

And so on. As we go down from the root node toward the 0- or 1-terminal, more variables of f are set to 0 or 1. Each term excluding x_i or $\bar{x}_i$ in each of these expansions [i.e., $f(0, x_2, \ldots, x_n)$ and $f(1, x_2, \ldots, x_n)$ in the first expansion, $f(0, 0, \ldots, x_n)$ and $f(0, 1, \ldots, x_n)$ in the second expansion, ect.], are called **cofactors**. Each node at the low ends of 0-edge and 1-edge from a node in an OBDD represents cofactors of the **Shannon expansion** of the logic function at the node, from which these 0-edge and 1-edge come down.

Procedure 26.1: Reduction of a BDD

1. For the given BDD, apply the following steps in any order.
   a. If two nodes, v_a and v_b, that represent the same variable x_i, branch to the same nodes in a lower level for each of $x_i = 0$ and $x_i = 1$, then combine them into one node that still represents variable x_i.

In Fig. 26.2(a), two nodes, v_a and v_b, that represent variable x_i, go to the same node v_u for $x_i = 0$ and the same node v_v for $x_i = 1$. Then, according to Step a, two nodes, v_a and v_b, are merged into one node v_{ab}, as shown in Fig. 26.2(b). The node v_{ab} represents variable x_i.

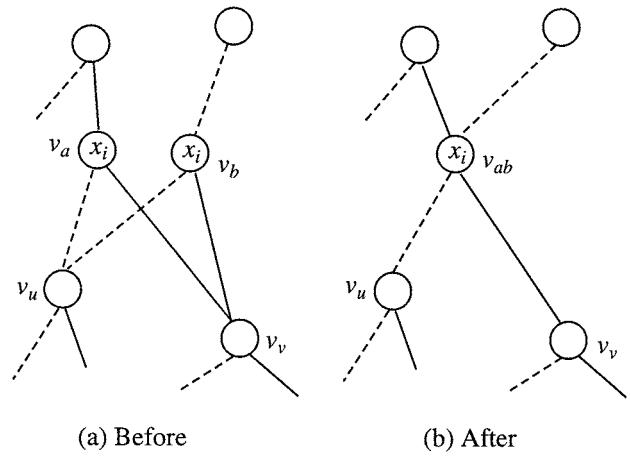


FIGURE 26.2 Step 1(a) of Procedure 26.1.

   b. If a node that represents a variable x_i branches to the same node in a lower level for both $x_i = 0$ and $x_i = 1$, then that node is deleted, and the 0- and 1-edges that come down to the former are extended to the latter.

In Fig. 26.3(a), node v_a that represents variable x_i branches to the same node v_b for both $x_i = 0$ and $x_i = 1$. Then, according to Step b, node v_a is deleted, as shown in Fig. 26.3(b), where the node v_a is a don't-care for x_i, and the edges that come down to v_a are extended to v_u.

   c. Terminals nodes with the same value, 0 or 1, are merged into the terminal node with the same original value.

All the 0-terminals in Fig. 26.1(b) are combined into one 0-terminal in each of Figs. 26.1(c), (d), and (e). All the 1-terminals are similarly combined.

2. When we cannot apply any step after repeatedly applying these steps (a), (b), and (c), the **reduced ordered BDD** (i.e., **ROBDD**) or simply called the **reduced BDD** (i.e., **RBDD**), is obtained for the given function.

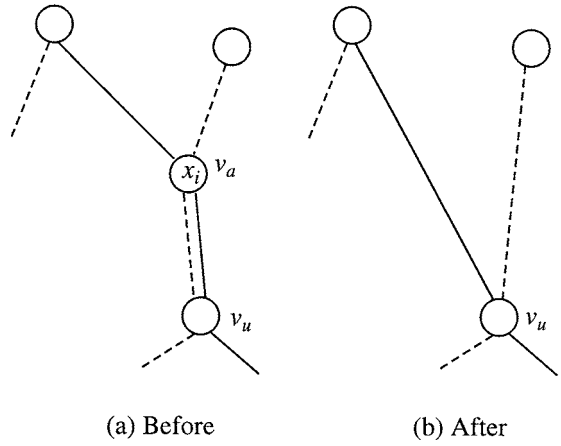


FIGURE 26.3 Step 1(b) of Procedure 26.1.

The following process is faster than the repeated application of Step 1(a) of Procedure 26.1. Suppose a BDD contains two nodes, v_a and v_b, both of which represent x_i, such that a sub-BDD, B_1, that stretches downward from v_a is completely identical to another sub-BDD, B_2, that stretches downward from v_b. In this case, each sub-BDD is said to be **isomorphic** to the other. Then the two sub-BDDs can be merged. For example, the two sub-BDDs, B_1 and B_2, shown in the two dotted rectangles in Fig. 26.4(a), both representing x_4, can be merged into one, B_3, as shown in Fig. 26.4(b).

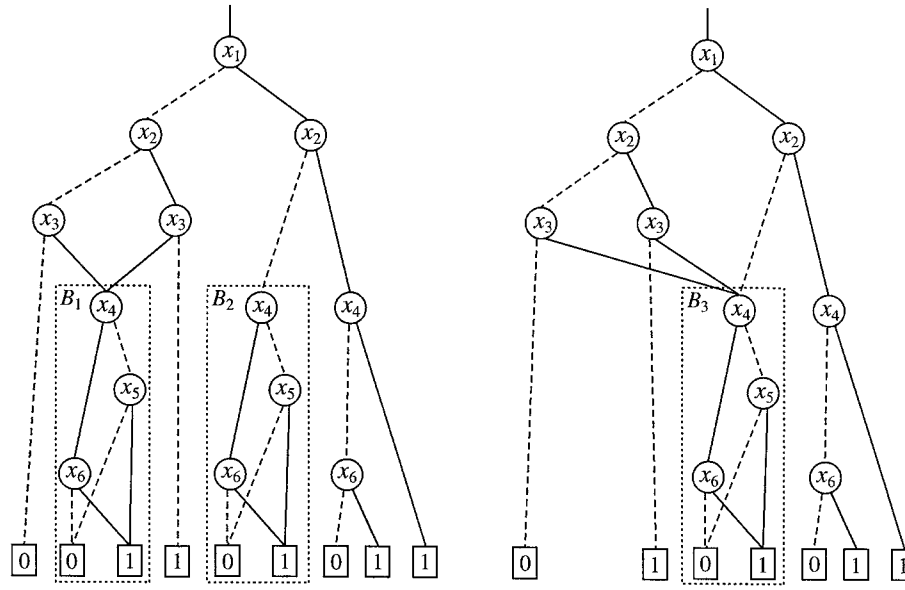


FIGURE 26.4 Merger of two isomorphic sub-BDDs.

Theorem 26.1: Any completely or incompletely specified function has a unique reduced ordered BDD and any other ordered BDD for the function in the same order of variables (i.e., not reduced) has more nodes.

According to this theorem, the ROBDD is unique for a given logic function when the order of the variables is fixed. (A BDD that has don't-cares can be expressed with addition of the *d*-terminal or two BDDs for the ON-set and OFF-set completely specified functions, as explained with Fig. 23.5. For details, see Ref. 24.) Thus, ROBDDs give canonical forms for logic functions. This property is very important to practical applications, as we can easily check the equivalence of two logic functions by only checking the isomorphism of their ROBDDs. **Henceforth in this chapter, ROBDDs will be referred to as BDDs for the sake of simplicity.**

It is known that a BDD for an *n*-variable function requires an exponentially increasing memory space in the worst case, as *n* increases.[16] However, the size of memory space for the BDD varies with the types of functions, unlike truth tables which always require memory space proportional to 2^n. But many logic functions that we encounter in design practice can be shown with BDDs without large memory space.[14] This is an attractive feature of BDDs. Fig. 26.5 shows the BDDs representing typical functions, AND, OR, and the parity function with three and *n* input variables. The parity function for *n* variables, $x_1 \oplus x_2 \oplus \ \ldots \oplus x_n$, can be shown with the BDD of $2(n-1)+1$ nodes and 2 terminals; whereas, if a truth table or a logic expression is used, the size increases exponentially as *n* increases.

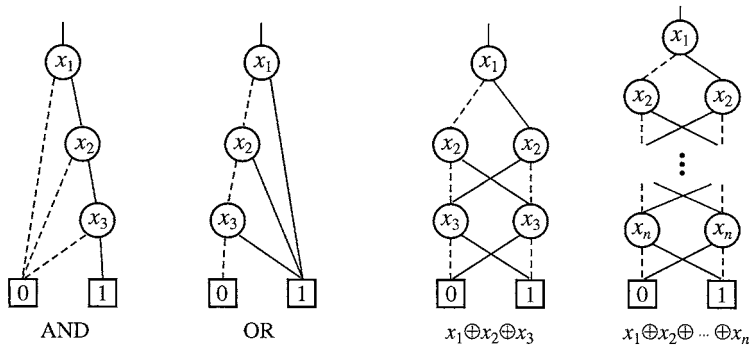


FIGURE 26.5 BDDs for typical logic functions.

A set of BDDs representing many functions can be combined into a graph that consists of BDDs sharing sub-graphs among them, as shown in Fig. 26.6. This idea saves time and space for duplicating isomorphic BDDs. By sharing all the isomorphic sub-graphs completely, no two nodes that express the same function coexist in the graph. We call it **shared BDDs** (**SBDDs**),[25] or **multi-rooted BDDs**. In a shared BDD, no two root nodes express the same function.

Shared BDDs are now widely used and those algorithms are more concise than ordinary BDDs. In the remainder of this chapter, we deal with shared BDD only.

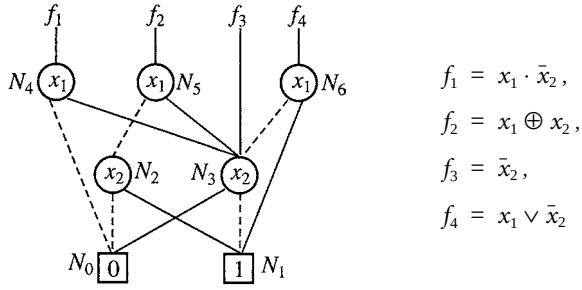


FIGURE 26.6 A shared BDD.

26.2 Construction of BDD Based on a Logic Expression

Procedure 26.1 shows a way of constructing compact BDDs from the truth table for a function f of n variables. This procedure, however, is not efficient because the size of its initial BDD is always of the order of 2^n, even for a very simple function. To avoid this problem, Bryant[6] presented a method to construct BDDs by applying a sequence of logic operations in a logic expression. Figure 26.7 shows a simple example of constructing a BDD for $f = (x_2 \cdot x_3) \vee x_1$. First, trivial BDDs for $x_2 \cdot x_3$, and x_3 are created in Fig. 26.7(a). Next, applying the *AND* operation between x_2 and x_3, the BDD for $x_2 \cdot x_3$ is then generated. Then, the BDD for the entire expression is obtained as the result of the *OR* operation between $(x_2 \cdot x_3)$ and x_1. After deleting the nodes that are not on the paths from the root node for f toward the 0- or 1-terminal, the final BDD is shown in Fig. 26.7(b).

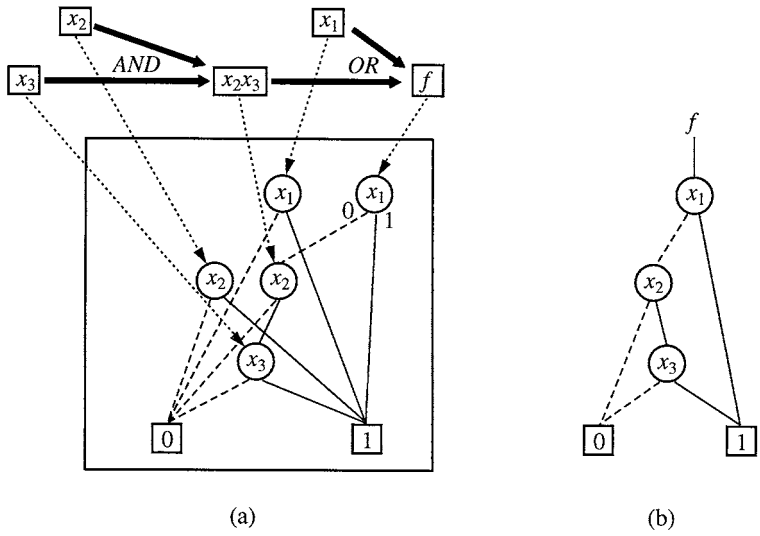


FIGURE 26.7 Generation of BDDs for $f = (x_2 \cdot x_3) \vee x_1$.

In the following, we show a formal algorithm for constructing BDDs for an arbitrary logic expression. This algorithm is generally far more efficient than that based on a truth table.

Binary logic operation

Suppose we perform a binary operation between two functions, g and h, and this operation is denoted by $g \circ h$, where "$\circ$" is one of *OR*, *AND*, *Exclusive-OR*, and others. Then by the Shannon expansion of a function explained previously, $g \circ h$ can be expanded as follows:

$$g \circ h = \bar{x}_i \cdot \left(g_{(x_i = 0)} \circ h_{(x_i = 0)} \right) \vee x_i \cdot \left(g_{(x_i = 1)} \circ h_{(x_i = 1)} \right)$$

with respect to variable x_i, which is the variable of the node that is in the highest level among all the nodes in g and h. This expansion creates the 0- and 1-edges which go to the next two nodes from the node for the variable x_i, by operations $\left(g_{(x_i = 0)} \circ h_{(x_i = 0)} \right)$ and $\left(g_{(x_i = 1)} \circ h_{(x_i = 1)} \right)$. The two subgraphs whose roots are these two nodes represent the cofactors derived by the Shannon expansion, $\left(g_{(x_i = 0)} \circ h_{(x_i = 0)} \right)$ and $\left(g_{(x_i = 1)} \circ h_{(x_i = 1)} \right)$. Repeating this expansion for all the variables, as we go down ontually trivial operations, such as $g \cdot 1 = g$, $g \oplus g = 0$, and $h \vee 0 = h$, finishing the construction of a BDD for $g \circ h$.

When BDDs for functions g and h are given, we can derive a new BDD for function $f = g \circ h$ by the following procedure.

Procedure 26.2: Construction of a BDD for Function $f = g \circ h$, Given BDDs for g and h.

Given BDDs for functions g and h (e.g., Fig. 26.8(a)), let us construct a new BDD (e.g., Fig. 26.8(b)) with respect to the specified logic operation $g \circ h$ and then reduce it by the previous Procedure 26.1 (e.g., Fig. 26.8(d)).

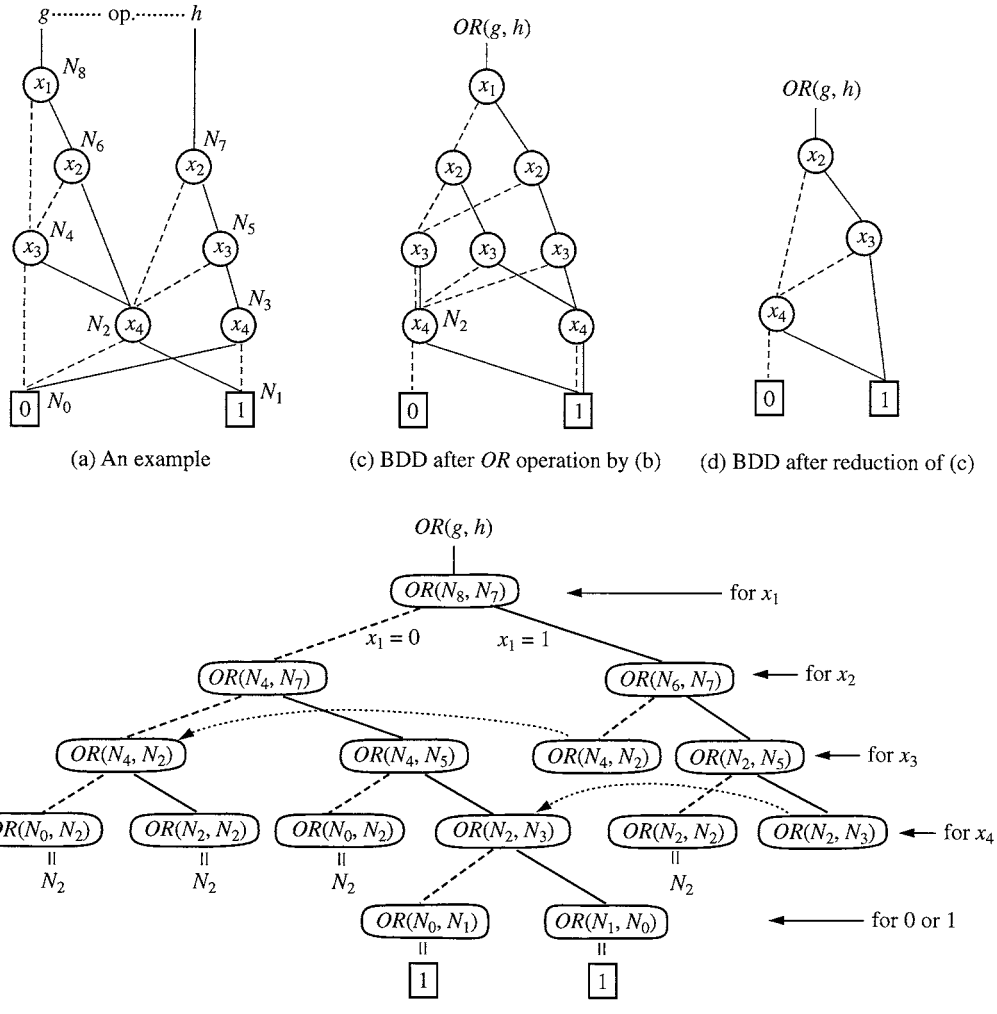


FIGURE 26.8 Procedure of binary operation.

1. Starting with the root nodes for g and h and going down toward to the 0- or 1-terminal, apply repeatedly steps a and b in the following, considering steps c and d.
   a. When the node under consideration, say N_a, in one of the two BDDs for g and h is in a higher level than the other node, N_b:
   If N_a and N_b are for variables x_a and x_b, respectively, and

if the 0-edge from N_a for $x_a = 0$ goes to node N_{a0} and the 1-edge from N_a for $x_a = 1$ goes to node N_{a1},

then we create the following two new nodes, to which the 0- and 1-edges go down from the corresponding node N'_a for the variable x_a in the new BDD (i.e., N'_a corresponds to this combination of N_a and N_b), as a part of the new BDD:

One new node for the operation ○ between N_{a0} and N_b for $x_a = 0$, and
the other new node for the operation ○ between N_{a1} and N_b for $x_a = 1$.

The variable for the first node, i.e., the node for N_{a0} ○ N_b will be the one in the higher level of the two nodes, N_{a0} and N_b, in the BDDs for g and h; and the variable for the second node (i.e., the node for N_{a1} ○ N_b) will be the one in the higher level of the two nodes, N_{a1} and N_b, in the BDDs for g and h.

In this case, creation of edges is not considered with respect to N_b in the original BDDs.

For example, suppose binary operation g ○ h is to be performed on the functions g and h in the BDD shown in Fig. 26.8(a) and furthermore, "○" is *OR* in this case. We need to start the *OR* operation with the root nodes for g and h (i.e., N_8 and N_7 respectively). N_8 is in a higher level than N_7. So, in Fig. 26.8(b) which explains how to construct the new BDD according to this procedure, we create two new nodes; one for $OR(N_4, N_7)$ for the 0-edge for $x_1 = 0$ from the node for $OR(N_8, N_7)$ and the other for $OR(N_6, N_7)$ for the 1-edge for $x_1 = 1$ from the node for $OR(N_8, N_7)$, corresponding to this step (a). Thus, we need next to form the *OR* between N_4 and N_7 and also the *OR* between N_6 and N_7 at these nodes in the second level in Fig. 26.8(b). These two nodes are both for variable x_2.

For $OR(N_4, N_7)$, N_7 is now in a higher level than N_4. So, corresponding to this step (a), we create the new nodes for $OR(N_4, N_2)$ and also for $OR(N_4, N_5)$ for 0- and 1-edge, respectively, in the third level in Fig. 26.8(b).

b. When the node under consideration, say N_a, in one of the BDDs for g and h is in the same level as the other node, N_b. (If N_a is for a variables x_a, these two nodes are for the same variable x_a because both g and h have this variable.):
If the 0-edge from N_a for $x_a = 0$ goes to node N_{a0} and the 1-edge from N_a for $x_a = 1$ goes to node N_{a1}, and
if the 0-edge from N_b for $x_b = 0$ goes to node N_{b0} and the 1-edge from N_b for $x_b = 1$ goes to node N_{b1},

then we create the following two new nodes, to which the 0- and 1-edges come down from the corresponding node N'_a for the variable x_a in the new BDD (i.e., N'_a corresponds to this combination of N_a and N_b), as a part of the new BDD:

one new node for the operation ○ between N_{a0} and N_{b0} for $x_a = 0$, and
the other new node for the operation ○ between N_{a1} and N_{b1} for $x_a = 1$.

The variable for the first node, i.e., the node for N_{a0} ○ N_{b0} will be the one in the higher level of the two nodes, N_{a0} and N_{b0}, in the BDDs for g and h; and the variable for the second node, i.e., the node for N_{a1} ○ N_{b1} will be the one in the higher level of the two nodes, N_{a1} and N_{b1}, in the BDDs for g and h.

For the example in Fig. 26.8(b), we need to create two new nodes for the operations, $OR(N_4, N_2)$ and $OR(N_2, N_5)$ to which the 0- and 1-edges go for $x_2 = 0$ and $x_2 = 1$, respectively, from the node $OR(N_6, N_7)$ because N_6 and N_7 are in the same level for variable x_2 in Fig. 26.8(a). These two nodes are both for variable x_3.

c. In the new BDD for the operation g ○ h, we need not have more than one subgraph that represent the same logic function. So, during the construction of the new BDD, whenever a new node (i.e., a root node of a subgraph) is for the same operation as an existing node, we need not continue creation of succeeding nodes from that new node.

For the example, in Fig. 26.8(b), the node for operation $OR(N_4, N_2)$ appears twice, as shown with a dotted arc, and we do not need to continue the creation of new nodes from the second and succeeding nodes for $OR(N_4, N_2)$. $OR(N_2, N_3)$ also appears twice, as shown with a dotted arc.

d. In the new BDD for the operation $g \circ h$, if one of N_a and N_b is the 0- or 1-terminal, or N_a and N_b are the same, i.e., $N_a = N_b$, then we can apply the logic operation that $g \circ h$ defines. If $g \circ h$ is for *AND* operation, for example, the node for $AND(N_1, N_a)$ represents N_a because N_1 is the 1-terminal in Fig. 26.8(a), and furthermore, if N_a represents function g, this node represents g.

Also, it is important to notice that only this step (d) is relevant to the logic operation defined by $g \circ h$, whereas all other steps from (a) through (c) are irrelevant of the logic operation $g \circ h$.

For the example in Fig. 26.8(b), the node for operation $OR(N_0, N_2)$ represents N_2, which is for variable x_4. Also, the node for operation $OR(N_0, N_1)$ represents constant value 1 because N_0 and N_1 are the 0- and 1-terminals, respectively, and consequently, $0 \vee 1$.

By using binary operation $f \oplus 1$, $\bar{f}$ can be performed and its processing time is linearly proportional to the size of a BDD. However, it is improved to a constant time by using the **complement edge**, which is discussed in the following. This technique is now commonly used.

2. Convert each node in the new BDD that is constructed in Step 1 to a node with the corresponding variable inside. Then derive the reduced BDD by Procedure 26.1.

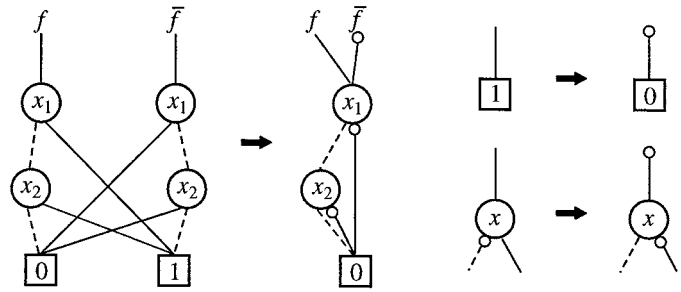
For the example, each node for operation $OR(N_a, N_b)$ in Fig. 26.8(b) is converted to a node with the corresponding variable inside in Fig. 26.8(c), which is reduced to the BDD in Fig. 26.8(d).

Complement Edges

Complement edge is a technique to reduce computation time and memory requirement of BDDs by using edges that indicates to complement the function of the subgraph pointed to by the edge, as shown in Fig. 26.9(a). This idea was first shown by Akers[3] and later discussed by Madre and Billon.[18] The use of complement edges brings the following outstanding merits.

- The BDD size is reduced by up to a half
- Negation can be performed in constant time
- Logic operations are sped by applying the rules, such as $f \cdot \bar{f} = 0$, $f \vee \bar{f} = 1$, and $f \oplus \bar{f} = 1$

Use of complement edges may break the uniqueness of BDDs. Therefore, we have to adopt the two rules, as illustrated in Fig. 26.9(b):



(a) Effect of complement edges (b) Rules for usage

FIGURE 26.9 Complement edges.

1. Using the 0-terminal only.
2. Not using a complement edge as the 0-edge of any node (i.e., use it on 1-edge only). If necessary, the complement edges can be carried over to higher nodes.

Derivation of a Logic Expression from a BDD

A logic expression for f can be easily derived from a BDD for f. A path from the root node for f to the 1-terminal in a BDD is called **1-path**, where the values of the variables on the edges on this path make f equal 1. For each 1-path, a product of literals is formed by choosing x_i for $x_i = 1$ or its complement, $\bar{x}_i$ for $x_i = 0$. The disjunction of such products for all 1-paths yields a logic expression for f. For example, in Fig. 26.8(a), the sequence of nodes, N_8-N_6-N_4-N_2-N_1, is a 1-path and the values of variables on this path, $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, and $x_4 = 1$, make the value of f (g for this example) equal 1. Finding all 1-paths, a logic expression for f is derived as $f = x_1\bar{x}_2x_3x_4 \vee x_1x_2x_4 \vee \bar{x}_1x_3x_4$. It is important to notice that logic expressions that are derived from all 1-paths are usually not minimum sums for the given functions.

26.3 Data Structure

In a typical realization of a BDD manipulator, all the nodes are stored in a single table in the main memory of the computer. Figure 26.10 is a simple example of realization for the BDD shown in Fig. 26.6. Each node has three basic attributes: input variable and the next nodes accessed by the 0- and 1-edges. Also, 0- and 1-terminals are first allocated in the table as special nodes. The other non-terminal nodes are created one by one during the execution of logic operations.

Current node	Variable	0-edge	1-edge	
N_0	-	-	-	← 0-terminal
N_1	-	-	-	← 1-terminal
N_2	x_2	N_0	N_1	
N_3	x_2	N_1	N_0	← f_3 ($= \bar{x}_2$)
N_4	x_1	N_0	N_3	← f_1 ($= x_1 \cdot \bar{x}_2$)
N_5	x_1	N_2	N_3	← f_2 ($= x_1 \oplus x_2$)
N_6	x_1	N_3	N_1	← f_4 ($= x_1 \vee \bar{x}_2$)

FIGURE 26.10 Table-based realization of a shared BDD.

Before creating a new node, we check the reduction rules of Procedure 26.1. If the 0- and 1-edges go to the same next node (Step 1(b) of Procedure 26.1) or if an equivalent node already exists (Step 1(a)), then we do not create a new node but simply copy that node as the next node. To find an equivalent node, we check a table which displays all the existing nodes. The hash table technique is very effective to accelerate this checking. (It can be done in a constant time for any large-scale BDDs, unless the table overflows in main memories.)

When generating BDDs for logic expressions, such as Procedure 26.2, many intermediate BDDs are temporarily generated. It is important for memory efficiency to delete such unnecessary BDDs. In order to determine the necessity of the nodes, a **reference counter** is attached to each node, which shows the number of incoming edges to the node.

In a typical implementation, the BDD manipulator consumes 20 to 30 bytes of memory for each node. Today, there are personal computers and workstations with more than 100 Mbytes of memory, and those facilitate us to generate BDDs containing as many as millions of nodes. However, the BDDs still grow beyond the memory capacity in some practical applications.

26.4 Ordering of Variables for Compact BDDs

BDDs are a canonical representation of logic functions under a fixed order of input variables. A change of the order of variable, however, may yield different BDDs of significantly different sizes for the same function. The effect of variable ordering depends on logic functions, changing sometimes dramatically the size of BDDs. Variable ordering is an important problem in using BDDs.

It is generally very time consuming to find the best order.[30] Currently known algorithms are limited to run on the small size of BDDs with up to about 17 variables.[13] It is difficult to find the best order for larger problems in reasonably short processing time. However, if we can find a fairly good order, it is useful for practical applications. There are many research works on heuristic methods of variable ordering.

Empirically, the following properties are known on the variable ordering.

1. Closely-related variables:
   Variables that are in close relationship in a logic expression should be close in variable order (e.g., x_1 in $x_1 \cdot x_2 \vee x_3 \cdot x_4$ is in closer relationship with x_2 than x_3). The logic network of AND-OR gates with $2n$ inputs in 2-level shown in Fig. 26.11(a) has $2n$ nodes in the best order with the expression $(x_1 \cdot x_2) \vee (x_3 \cdot x_4) \vee \ldots \vee (x_{2n-1} \cdot x_{2n})$, as shown for $n = 2$ in Fig. 26.11(b), while it needs $(2^{n+1} - 2)$ nodes in the worst order, as shown in Fig. 26.11(c). If the same order of variables as the one in Fig. 26.11(b) is kept on the BDD, Fig. 26.11(c) represents the function $(x_1 \cdot x_{n+1}) \vee (x_2 \cdot x_{n+2}) \vee \ldots \vee (x_n \cdot x_{2n})$.

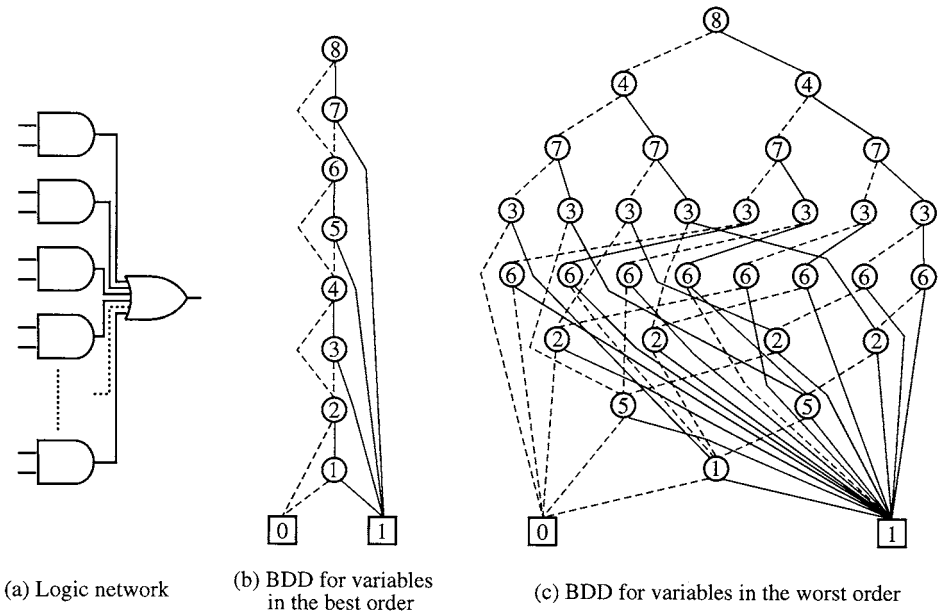
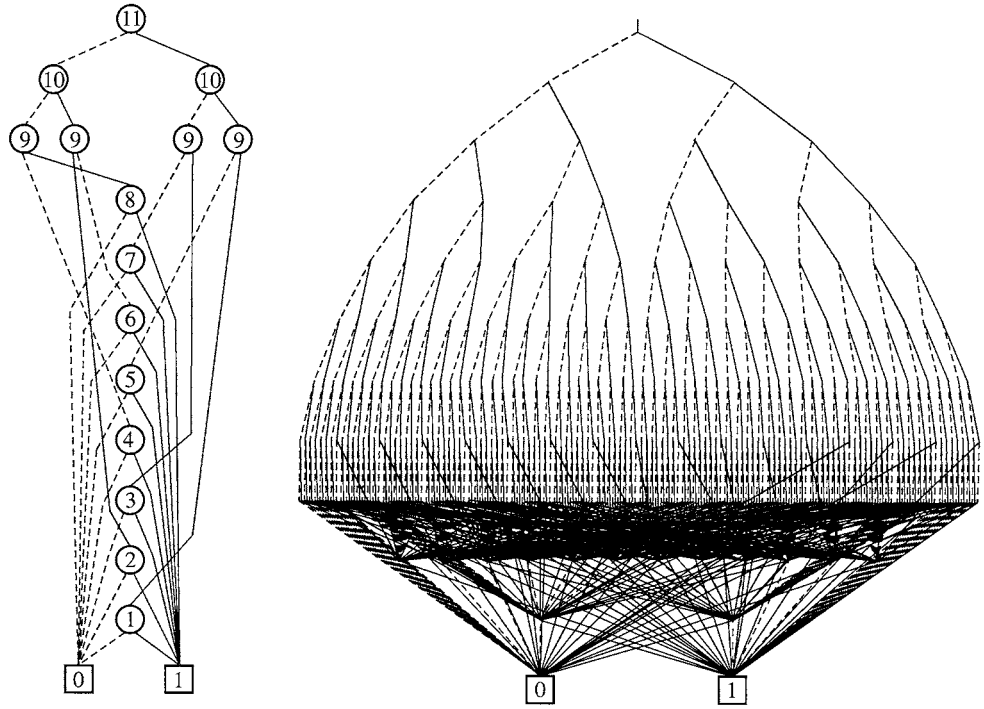


FIGURE 26.11 BDDs for 2-level logic network with AND/OR gates.

2. Influential variables:
   The variables that greatly influence the nature of a function should be at higher position. For example, the **8-to-1 selector** shown in Fig. 26.12(a) can be represented by a linear size of BDD when the three control inputs are ordered high, but when the order is reversed, it becomes of exponentially increasing size as the number of variables (i.e., the total number of data inputs and control inputs) increases, as shown in Fig. 26.12(b).



(a) Best order (b) Worst order

FIGURE 26.12 BDDs for 8-to-1 selector.

Based on empirical rules like this, Fujita et al.[9] and Malik et al.[20] presented methods; in these methods, an output of the given logic networks is reached, traversing in a depth-first manner, then an input variable that can be reached by going back toward the inputs of the network is placed at highest position in variable ordering. Minato[25] devised another heuristic method in which each output function of the given logic networks is weighted and then input variables are ordered by the weight of each input variable determined by how each input can be reached from an output of the network. Butler et al.[8] proposed another heuristic based on a measure which uses not only the connection configuration of the network, but also the output functions of the network. These methods probably find a good order before generating BDDs. They find good orders in many cases, but there is no method that is always effective to a given network.

Another approach reduces the size of BDDs by reordering input variables. A greedy local exchange (swapping adjacent variables) method was developed by Fujita et al.[10] Minato[22] presented another reordering method which measures the width of BDDs as a cost function. In many cases, these methods find a fairly good order using no additional information. A drawback of the approach of these methods is that they cannot start if an initial BDD is too large.

One remarkable work is **dynamic variable ordering,** presented by Rudell.[27] In this technique, the BDD package itself determines and maintains the variable order. Every time the BDDs grow to a certain size, the reordering process is invoked automatically. This method is very effective in terms of the reduction of BDD size, although it sometimes takes a long computation time.

Table 26.1 shows experimental results on the effect of variable ordering. The logic network "sel8" is an 8-bit data selector, and "enc8" is an 8-bit encoder. "add8" and "mult6" are an 8-bit adder and a 6-bit multiplier. The rest is chosen from the benchmark networks in MCNC'90.[1] Table 26.1 compares four different orders: the original order, a random order, and two heuristic orders. The results show that the heuristic ordering methods are very effective except for a few cases which are insensitive to the order.

TABLE 26.1 Effect of Variable Ordering

	Network Feature			BDD Size (with complement edges)			
Name	No. of Inputs	No. of Outputs	No. of Inputs to All Gates	Original	Random	Heur-1	Heur-2
sel8	12	2	29	16	88	23	19
enc8	9	4	31	28	29	28	27
add8	17	9	65	83	885	41	41
mult6	12	12	187	2183	2927	2471	2281
vg2	25	8	97	117	842	97	86
c432	36	7	203	3986	(>500k)	27302	1361
c499	41	32	275	115654	(>500k)	52369	40288
c880	60	26	464	(>500k)	(>500k)	23364	9114

Note: Heuristic-1: Heuristic order based on connection configuration.[5]
Heuristic-2: BDD reduction by exchanging variables.[14]

Unfortunately, there are some hard examples where variable ordering is powerless. For example, Bryant[6] proved that an *n*-bit multiplier function requires an exponentially increasing number of BDD nodes for any variable order, as *n* increases. However, for many other practical functions, the variable ordering methods are useful for generating compact BDDs in a reasonably short time.

26.5 Remarks

Several groups developed BDD packages, and some of them are open to the public. For example, the CUDD package[12] is well-known to BDD researchers in the U.S. Many other BDD packages may be found on the Internet. BDD packages, in general, are based on the quick search of hash tables and linked-list data structures. They greatly benefit from the property of the **random access machine model**,[2] where any data in main memory can be accessed in constant time.

Presently, considerable research is in progress. Detection of total or partial symmetry of a logic function with respect to variables has been a very time-consuming problem, but now it can be done in a short time by BDDs.[26,29] Also, decomposition of a logic function, which used to be very difficult, can be quickly solved with BDD.[5,21] A number of new types of BDDs have been proposed in recent years. For example, the Zero-Suppressed BDD (ZBDD)[23] is useful for solving covering problems, which are used in deriving a minimal sum and other combinatorial problems. The Binary Moment Diagram (BMD)[7] is another type of BDD that is used for representing logic functions for arithmetic operations. For those who are interested in more detailed techniques related to BDDs, several good surveys[11,24,28] are available.

References

1. *ACM/SIGDA Benchmark Newsletter*, DAC '93 Edition, June 1993.
2. Aho, A. V., J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.
3. Akers, S. B., "Functional testing with binary decision diagrams," *Proc. 8th Ann. IEEE Conf. Fault-Tolerant Comput.*, pp. 75-82, 1978.
4. Akers, S. B., "Binary decision diagrams," *IEEE Trans. on Computers*, vol. C-27, no. 6, pp. 509-516, June 1978.
5. Bertacco, V. and M. Damiani, "The disjunctive decomposition of logic functions," *ICCAD '97*, pp. 78-82, Nov. 1997.
6. Bryant, R. E., "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. on Computers*, vol. C-35, no. 8, pp. 677-691, Aug. 1986.

7. Bryant, R. E. and Y.-A. Chen, "Verification of arithmetic functions with binary moment diagrams," *Proc. 32nd ACM/IEEE DAC,* pp. 535-541, June 1995.
8. Butler, K. M., D. E. Ross, R. Kapur, and M. R. Mercer, "Heuristics to compute variable orderings for efficient manipulation of ordered binary decision diagrams," *Proc. of 28th ACM/IEEE DAC,* pp. 417-420, June 1991.
9. Fujita, M., H. Fujisawa, and N. Kawato, "Evaluation and improvement of Boolean comparison method based on binary decision diagrams," *Proc. IEEE/ACM ICCAD '88,* pp. 2-5, Nov. 1988.
10. Fujita, M., Y. Matsunaga, and T. Kakuda, "On variable ordering of binary decision diagrams for the application of multi-level logic synthesis," *Proc. IEEE EDAC '91,* pp. 50-54, Feb. 1991.
11. Hachtel, G. and F. Somenzi, *Logic Synthesis and Verification Algorithms,* Kluwer Academic Publishers, 1996.
12. http://vlsi.colorado.edu/software.html.
13. Ishiura, N., H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables," *Proc. IEEE/ACM ICCAD '91,* pp. 472-475, Nov. 1991.
14. Ishiura, N. and S. Yajima, "A class of logic functions expressible by a polynomial-size binary decision diagrams," in *Proc. Synthesis and Simulation Meeting and International Interchange (SASIMII '90. Japan),* pp. 48-54, Oct. 1990.
15. Lee, C.Y., "Representation of switching circuits by binary-decision programs," *Bell Sys. Tech. Jour.,* vol. 38, pp. 985-999, July 1959.
16. Liaw, H.-T. and C.-S. Lin, "On the OBDD-representation of general Boolean functions," *IEEE Trans. on Computers,* vol. C-41, no. 6, pp. 661-664, June 1992.
17. Lin, B. and F. Somenzi, "Minimization of symbolic relations," *Proc. IEEE/ACM ICCAD '90,* pp. 88-91, Nov. 1990.
18. Madre, J. C. and J. P. Billon, "Proving circuit correctness using formal comparison between expected and extracted behaviour," *Proc. 25th ACM/IEEE DAC,* pp. 205-210, June 1988.
19. Madre, J. C. and O. Coudert, "A logically complete reasoning maintenance system based on a logical constraint solver," *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI'91),* pp. 294-299, Aug. 1991.
20. Malik, S., A. R. Wang, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Logic verification using binary decision diagrams in a logic synthesis environment," *Proc. IEEE/ACM ICCAD '88,* pp. 6-9, Nov. 1988.
21. Matsunaga, Y., "An exact and efficient algorithm for disjunctive decomposition," *SASIMI '98,* pp. 44-50, Oct. 1998.
22. Minato, S., "Minimum-width method of variable ordering for binary decision diagrams," *IEICE Trans. Fundamentals,* vol. E75-A, no. 3, pp. 392-399, Mar. 1992.
23. Minato, S., "Zero-suppressed BDDs for set manipulation in combinatorial problems," *Proc. 30th ACM/IEEE DAC,* pp. 272-277, June 1993.
24. Minato, S., *Binary Decision Diagrams and Applications for VLSI CAD,* Kluwer Academic Publishers, 1995.
25. Minato, S., N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation," *Proc. 27th IEEE/ACM DAC,* pp. 52-57, June 1990.
26. Möller, D., J. Mohnke, and M. Weber, "Detection of symmetry of Boolean functions represented by ROBDDs," *Proc. IEEE/ACM ICCAD '93,* pp. 680-684, Nov. 1993.
27. Rudell, R., "Dynamic variable ordering for ordered binary decision diagrams," *Proc. IEEE/ACM ICCAD '93,* pp. 42-47, Nov. 1993.
28. Sasao, T., Ed., *Representation of Discrete Functions,* Kluwer Academic Publishers, 1996.
29. Sawada, H., S. Yamashita, and A. Nagoya, "Restricted simple disjunctive decompositions based on grouping symmetric variables," *Proc. IEEE Great Lakes Symp. on VLSI,* pp. 39-44, Mar. 1997.
30. Tani, S., K. Hamaguchi, and S. Yajima, "The complexity of the optimal variable ordering problems of shared binary decision diagrams," *The 4th Int'l Symp. Algorithms and Computation, Lecture Notes in Computer Science,* vol. 762, Springer, 1993, pp. 389-398.

Muroga, S. "Logic Synthesis with AND and OR Gates in Two Levels"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

27

Logic Synthesis With AND and OR Gates in Two Levels

Saburo Muroga
University of Illinois at Urbana-Champaign

27.1 Introduction

When logic networks are realized in transistor circuits on an integrated circuit chip, each gate in logic networks usually realizes more complex functions than AND or OR gates. But handy design methods are not available for designing logic networks with such complex gates under very diversified complex constraints such as delay time and layout rules. Thus, designers often design logic networks with AND, OR, and NOT gates as a starting point for design with more complex gates under complex constraints. AND, OR, and NOT gates are much easier for human minds to deal with. Logic networks with AND, OR, and NOT gates, after designing such networks, are often converted into transistor circuits. This conversion process illustrated in Fig. 27.1 (the transistor circuit in this figure will be explained in later chapters) is called **technology mapping**. As can be seen, technology mapping is complex because logic gates and the corresponding transistor logic gates usually do not correspond one to one before and after technology mapping and layout has to be considered for speed and area.

Also, logic gates are realized by different types of transistor circuits, depending on design objectives. They are called **logic families**. There are several logic families, such as ECL, nMOS circuits, static CMOS, and dynamic CMOS, as discussed in Chapters 30, 35, and 36. Technology mapping is different for different logic families.

First, let us describe the design of logic networks with AND and OR gates in two levels, because handy methods are not available for designing logic networks with AND and OR gates in more than two levels. Although logic networks with AND and OR gates in two levels may not be directly useful for designing transistor circuits to be laid out on an integrated circuit chip, there are some cases where they are directly usable, such as programmable logic arrays to be described in a later chapter.

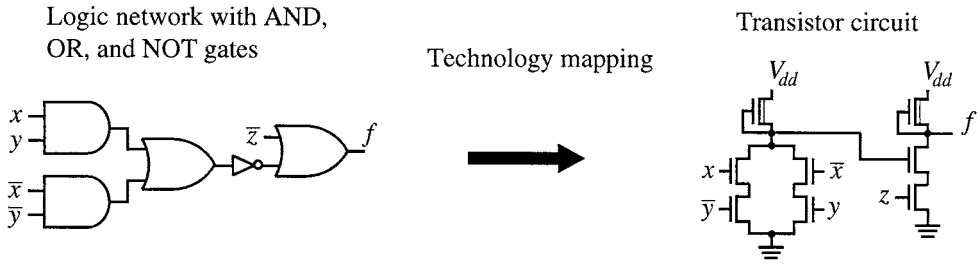


FIGURE 27.1 Conversion of a logic network with simple gates by technology mapping.

27.2 Design of Single-Output Minimal Networks with AND and OR Gates in Two Levels

Suppose that we want to obtain a two-level network with a minimum number of gates and, then, as a secondary objective, a minimum number of connections under Assumptions 27.1 in the following, regardless of whether we have AND or OR gates, respectively, in the first and second levels, or in the second and first levels. In this case, we have to design a network based on the minimal sum and another based on the minimal product, and then choose the better network. Suppose that we want to design a two-level AND/OR network for the function shown in Fig. 27.2(a). This function has only one minimal sum, as shown with loops in Fig. 27.2(a). Also, it has only one minimal product, as shown in Fig 27.3(a). The network in Fig. 27.3(b), based on this minimal product, requires one less gate, despite more loops, than the network based on the minimal sum in Fig. 27.2(b), and consequently the network in Fig. 27.3(b) is preferable.

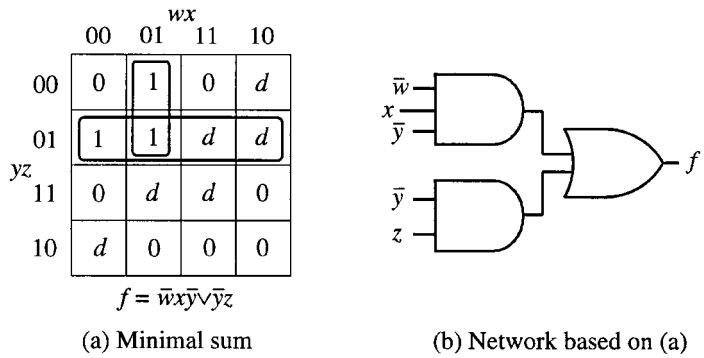


FIGURE 27.2 Minimal sum and the corresponding network.

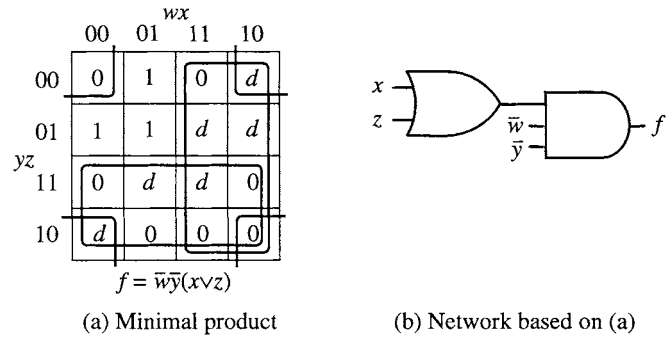


FIGURE 27.3 Minimal product and the corresponding network.

The above design procedure of a minimal network based on a minimal sum is meaningful under the following assumptions.

Assumptions 27.1: (1) The number of levels is at most two; (2) Only AND gates and OR gates are used in one level and second the other level; (3) Complemented variables $\bar{x}_i$'s as well as noncomplemented x_i's for each i are available as the network inputs; (4) No maximum fan-in restriction is imposed on any gate in a network to be designed; (5) Among networks realizable in two levels, we will choose networks that have a minimum number of gates. Then, from those with the minimum number of gates, we will choose a network that has a minimum number of connections.

If any of these is violated, the number of logic gates as the primary objective and the number of connections as the secondary objective are not minimized. If we do not have the restriction "at most two levels," we can generally have a network of fewer gates.

Karnaugh maps have been widely used because of convenience when the number of variables is small. But when the number of variables is many, maps are increasingly complex and processing with them become tedious, as discussed in Chapter 25. Furthermore, the corresponding logic networks with AND and OR gates in two levels are not useful because of excessive fan-ins and fan-outs.

27.3 Design of Multiple-Output Networks with AND and OR Gates in Two Levels

So far, we have discussed the synthesis of a two-level network with a single output. In many cases in practice, however, we need a two-level network with multiple outputs; so here let us discuss the synthesis of such a network, which is more complex than a network with a single output.

An obvious approach is to design a network for each output function separately. But this approach usually will not yield a more compact network than will synthesis of the functions collectively, because, for example, in Fig. 27.4, the AND gate h, which can be shared by two output gates for f_i and f_j, must be repeated in separate networks for f_i and f_j by this approach.

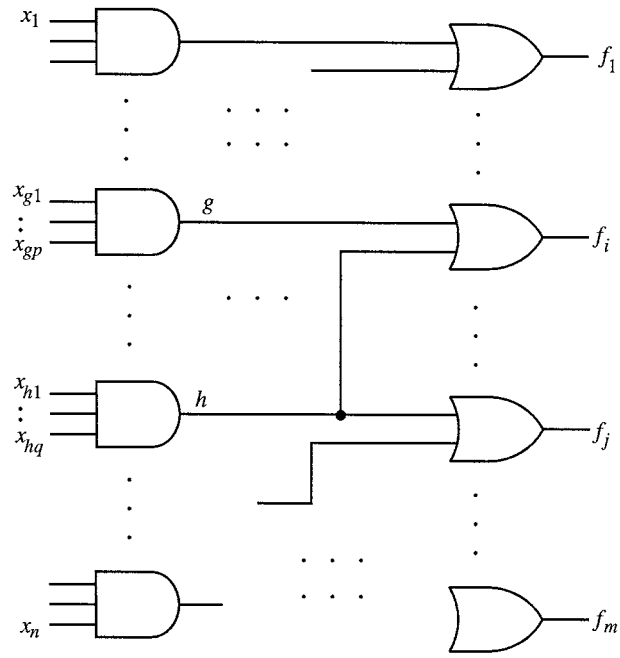


FIGURE 27.4 A two-level network with multiple outputs.

Before discussing a design procedure, let us study the properties of a minimal two-level network that has only AND gates in the first level and only OR gates for given output functions $f_1, f_2, \ldots, f_m$ in the second level, as shown in Fig. 27.4, where "a minimal network" means that the network has a minimum number of gates as the primary objective and a minimum number of connections as the secondary objective. The number of OR gates required for this network is at most m. Actually, when some functions are expressed as single products of literals, the number of OR gates can be less than m because these functions can be realized directly at the outputs of the AND gates without the use of OR gates. Also, when a function f_i has a prime implicant consisting of a single literal, that literal can be directly connected to the OR gate for f_i without the use of an AND gate. (These special cases can be treated easily by modifying the synthesis under the following assumption.) However, for simplicity, let us **assume that every variable input is connected only to AND gates in the first level and every function f_i, $1 \le i \le m$, is realized at the outputs of OR gates in the second level.** This is actually required with some electronic realizations of a network (e.g., when every variable input needs to have the same delay to the network outputs, or when PLAs which will be described in Chapter 42 are used to realize two-level networks).

Multiple-Output Prime Implicants

Suppose that we have a two-level, multiple-output network that has AND gates in the first level, and m OR gates in the second (i.e., output) level for functions $f_1, f_2, \ldots, f_m$ (as an example, f_1, f_2, and f_3 are given as shown in the Karnaugh maps in Fig. 27.5). The network has a minimum number of gates as the primary objective, and a minimum number of connections as the secondary objective. Let us explore the basic properties of such a minimal network.

Property 27.1: First consider an AND gate, g, that is connected to only one output gate, say for f_i, in Fig. 27.4. If gate g has inputs $x_{g1}, x_{g2}, \ldots, x_{gp}$, its output represents the product $x_{g1}x_{g2}\ldots x_{gp}$. Then, if the product assumes the value 1, f_i becomes 1. Thus, the product $x_{g1}x_{g2}\ldots x_{gp}$ is an implicant of f_i. Since the network is minimal, gate g has no unnecessary inputs, and the removal of any input from gate g will make the OR gate for f_i express a different function (i.e., in Fig. 27.5, the loop that the product represents becomes larger, containing some 0-cells, by deleting any variables from the product and then the new product is not an implicant of f_i). Thus, $x_{g1}, x_{g2}, \ldots, x_{gp}$ is a prime implicant of f_i.

Property 27.2: Next consider an AND gate, h, that is connected to two output gates for f_i and f_j in Fig. 27.4. This time, the situation is more complicated. If the product $x_{h1}x_{h2}\ldots x_{hq}$ realized at the output of gate h assumes the value 1, both f_i and f_j become 1 and, consequently, the product f_if_j of the two functions also becomes 1 (thus, if a Karnaugh map is drawn for f_if_j as shown in Fig. 27.5, the map has a loop consisting of only 1-cells for $x_{h1}x_{h2}\ldots x_{hq}$). Thus $x_{h1}x_{h2}\ldots x_{hq}$ is an implicant of product f_if_j. The network is minimal in realizing each of functions $f_1, f_2, \ldots, f_m$; so, if any input is removed from AND gate h, at least one of f_i and f_j will be a different function and $x_{h1}x_{h2}\ldots x_{hq}$ will no longer be an implicant of f_if_j. (That is, if any input is removed from AND gate h, a loop in the map for at least one of f_i and f_j will become larger. For example, if x_1 is deleted from M: $x_1\bar{x}_2x_3$ for f_1f_2, the loop for $\bar{x}_2x_3$, product of the remaining literals, appears in the maps for f_1 and also for f_2 in Fig. 27.5. The loop in the map for f_2 contains 0-cell. Thus, if a loop is formed in the map for f_1f_2 as a product of these loops in the maps for f_1 and f_2, the new loop contains 0-cell in the map for f_1f_2. This means that if any variable is removed from $x_{h1}x_{h2}\ldots x_{hq}$, the remainder is not an implicant of f_1f_2.) Thus, $x_{h1}x_{h2}\ldots x_{hq}$ is a prime implicant of the product f_if_j. (But notice that the product $x_{h1}x_{h2}\ldots x_{hq}$ is not necessarily a prime implicant of each single function f_i or f_j, as can be seen in Fig. 27.5. For example, M: $x_1\bar{x}_2x_3$ is a prime implicant of f_1f_2 in Fig. 27.5. Although this product, $x_1\bar{x}_2x_3$, is a prime implicant of f_2, it is not a prime implicant of f_1 in Fig. 27.5.)

Generalizing this, we obtain the following conclusion. Suppose that the output of an AND gate is connected to r OR gates for functions $f_{i1}, \ldots, f_{ir}$. Since the network is minimal, this gate has no unnecessary inputs. Then the product of input variables realized at the output of this AND gate is a prime implicant of the product $f_{i1}\ldots f_{ir}$ (but is not necessarily a prime implicant of any product of $r-1$ or fewer, of these $f_{i1}, \ldots, f_{ir}$).

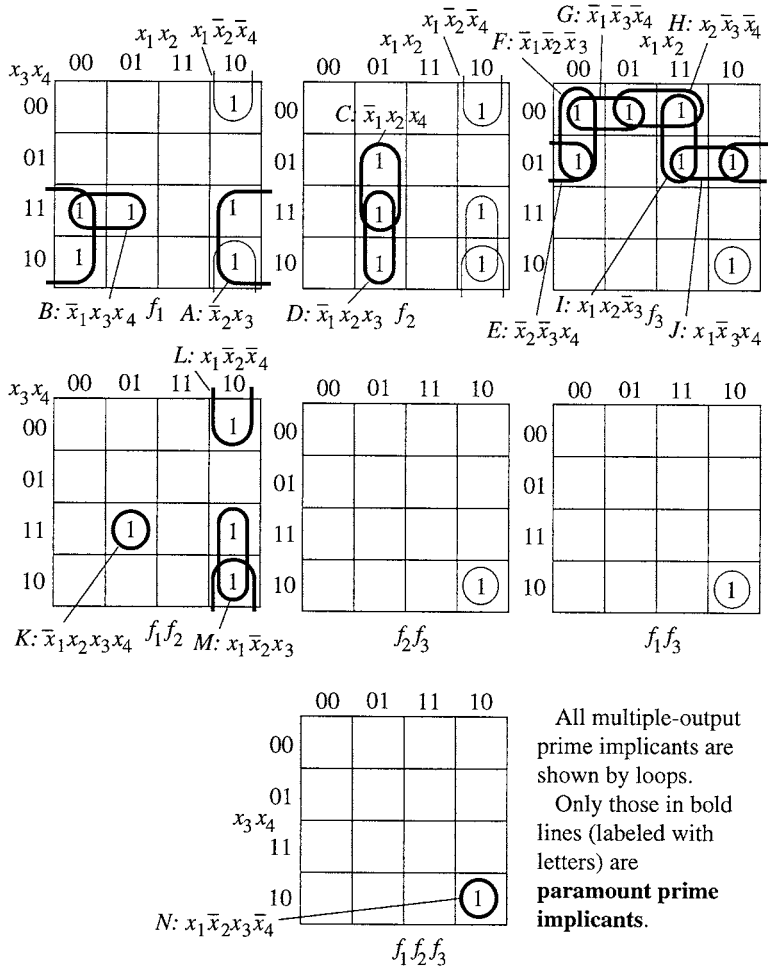


FIGURE 27.5 Multiple-output prime implicants on Karnaugh maps.

As in the synthesis of a single-output network, we need to find all prime implicants and then develop disjunctive forms by choosing an appropriate set of prime implicants. (Notice that each of these disjunctive forms is not necessarily a minimal sum for one of functions, f_i.) But, unlike the synthesis of a single-output network, in this case we must consider all prime implicants not only for each of the given functions $f_1, \ldots, f_m$, but also for all possible products of them, f_1f_2, f_1f_3, …, $f_{m-1}f_m$; $f_1f_2f_3$, $f_1f_2f_4$, …; …; …; $f_1f_2 \ldots f_{m-1}f_m$.

Definition 27.1: Suppose that m functions $f_1, \ldots, f_m$ are given. All prime implicants for each of these m functions, and also all prime implicants for every possible product of these functions, that is,

$$
\begin{aligned}
&f_1, f_2, \ldots, f_m \\
&f_1f_2, f_1f_3, \ldots, f_{m-1}f_m, \\
&f_1f_2f_3, f_1f_2f_4, \ldots, f_{m-2}f_{m-1}f_m \\
&\ldots, \ldots, \ldots, \\
&\ldots, \ldots, \ldots, \\
&f_1f_2 \ldots f_{m-1}, f_1f_2 \ldots f_{m-2}f_m, \ldots, f_2f_3 \ldots f_m \\
&f_1f_2f_3 \ldots f_{m-1}f_m
\end{aligned}
$$

are called the **multiple-output prime implicants** of $f_1, \ldots, f_m$.

When the number of variables is small, we can find all multiple-output prime implicants on Karnaugh maps, as illustrated in Fig. 27.5 for the case of three functions of four variables. In addition to the maps for given functions f_1, f_2, and f_3, we draw the maps for all possible products of these functions; that is, f_1f_2, f_2f_3, f_1f_3, and $f_1f_2f_3$. Then, prime-implicant loops are formed on each of these maps. These loops represent all the multiple-output prime implicants of given functions f_1, f_2, and f_3.

Paramount Prime Implicants

Suppose we find all multiple-output prime implicants for the given functions. Then, if a prime implicant P appears more than once as prime implicants for different products of functions, P for the product of the largest number of functions among these products of functions is called a **paramount prime implicant for *P*** in order to differentiate this P from other multiple-output prime implicants. As a special case, if P appears only once, it is the paramount prime implicant for P.

For example, among all multiple-output prime implicants for f_1, f_2, and f_3 (i.e., among all prime implicants for f_1f_2, f_2f_3, f_1f_3, and $f_1f_2f_3$ shown as loops in Fig. 27.5), the prime implicant $x_1\bar{x}_2\bar{x}_4$ appears three times in Fig. 27.5. In other words, $x_1\bar{x}_2\bar{x}_4$ appears as a prime implicant for the function f_1 (see the map for f_1 in Fig. 27.5), as a prime implicant for f_2, and also as a prime implicant for the product f_1f_2 (in the map for f_1f_2, this appears as L: $x_1\bar{x}_2\bar{x}_4$). Then, the prime implicant $x_1\bar{x}_2\bar{x}_4$ for f_1f_2 is the paramount prime implicant for $x_1\bar{x}_2\bar{x}_4$ because it is a prime implicant for the product of two functions, f_1 and f_2, but the prime implicant $x_1\bar{x}_2\bar{x}_4$ for f_1 or f_2 is a prime implicant for a single function f_1 or f_2 alone. In the two-level network, $x_1\bar{x}_2\bar{x}_4$ for the product f_1f_2 realizes the AND gate with output connections to the OR gates for f_1 and f_2, whereas $x_1\bar{x}_2\bar{x}_4$ for f_1 or f_2 realizes the AND gate with output connection to only the OR gate for f_1 or f_2, respectively. Thus, if we use $x_1\bar{x}_2\bar{x}_4$ for the product f_1f_2 instead of $x_1\bar{x}_2\bar{x}_4$ for f_1 or f_2 (in other words, if we use AND gates with more output connections) then the network will be realized with no more gates. In this sense, $x_1\bar{x}_2\bar{x}_4$ for the product f_1f_2 is more desirable than $x_1\bar{x}_2\bar{x}_4$ for the function f_1 or f_2. Prime implicant $x_1\bar{x}_2\bar{x}_4$ for the product f_1f_2 is called a paramount prime implicant, as formally defined in the following, and is shown with label L in a bold line in the map for f_1f_2 in Fig. 27.5; whereas $x_1\bar{x}_2\bar{x}_4$ for f_1 or f_2 alone is not labeled and is also not in a bold line in the map for f_1 or f_2. Thus, **when we provide an AND gate whose output realizes the prime implicant $x_1\bar{x}_2\bar{x}_4$, we can connect its output connection in three different ways, i.e., to f_1, f_2, or both f_1 and f_2, realizing the same output functions, f_1, f_2, and f_3. In this case, the paramount prime implicant means that we can connect the largest number of OR gates from this AND gate; in other words, this AND gate has the largest coverage, although some connections may turn out to be redundant later.**

Definition 27.2: Suppose that when all the multiple-output prime implicants of $f_1, \ldots, f_m$ are considered, a product of some literals, P, is a prime implicant for the product of k functions $f_{p1}, f_{p2}, \ldots, f_{pk}$ (possibly also prime implicants for products of $k-1$or fewer of these functions), but is not a prime implicant for any product of more functions that includes all these functions $f_{p1}, f_{p2}, \ldots, f_{pk}$. (For the above example, $x_1\bar{x}_2\bar{x}_4$ is a prime implicant for the product of two functions, f_1 and f_2, and also a prime implicant for a single function, f_1 or f_2. But $x_1\bar{x}_2\bar{x}_4$ is not a prime implicant for the product of more functions, including f_1 and f_2, that is, for the product $f_1f_2f_3$.) Then, P for the product of $f_{p1}, f_{p2}, \ldots, f_{pk}$ is called the **paramount prime implicant** for this prime implicant ($x_1\bar{x}_2\bar{x}_4$ for f_1f_2 is a paramount prime implicant, but $x_1\bar{x}_2\bar{x}_4$ for f_1 or f_2 is not). As a special case, if P is a prime implicant for only one function but not a prime implicant for any product of more than one function, it is a paramount prime implicant (B: $\bar{x}_1x_3x_4$ for f_1 is such an example in Fig. 27.5).

In Fig. 27.5, only labeled loops shown in bold lines represent paramount prime implicants.

If a two-level network is first designed only with the AND gates that correspond to the paramount prime implicants, we can minimize the number of gates. This does not necessarily minimize the number of connections as the secondary objective. But in some important electronic realizations of two-level networks, such as PLAs (which will be explained later) this is not important. Thus, let us consider only the minimization of the number of gates in the following for the sake of simplicity.

Design of a Two-Level Network with a Minimum Number of AND and OR Gates

In the following procedure, we will derive a two-level network with a minimum number of gates (but without minimizing the number of connections as the secondary objective) by finding a minimal number of paramount prime implicants to represent the given functions.

Procedure 27.1: Design of a Multiple-Output Two-Level Network That Has a Minimum Number of Gates Without Minimizing the Number of Connections

We want to design a two-level network that has AND gates in the first level and OR gates in the second level. We shall assume that variable inputs can be connected to AND gates only (not to OR gates), and that the given output functions $f_1, f_2, \ldots, f_m$ are to be realized only at the outputs of OR gates.

Suppose we have already found all the paramount prime implicants for the given functions and their products.

1. Find a set of a smallest number of paramount prime implicant loops that covers all 1-cells in Karnaugh maps for the given functions $f_1, f_2, \ldots, f_m$. In this case, maps for their products, such as f_1f_2, need not be considered. If there is more than one such set, choose a set that has as large loops as possible (i.e., choose a set of loops such that the total number of inputs to the AND gates that correspond to these loops is the smallest).

For example, suppose that the three functions of four variables, f_1, f_2, f_3, shown in the Karnaugh maps in Fig. 27.5 are given. Then, using only the bold-lined loops labeled with letters (i.e., paramount prime implicants), try to cover all 1-cells in the maps for f_1, f_2, and f_3 only (i.e., using the only top three maps in Fig. 27.5). Then, we find that more than one set of loops have the same number of loops with the same sizes. *AKLCDMNFHJ*, one of these sets, covers all functions f_1, f_2, and f_3, as illustrated in Fig. 27.6.

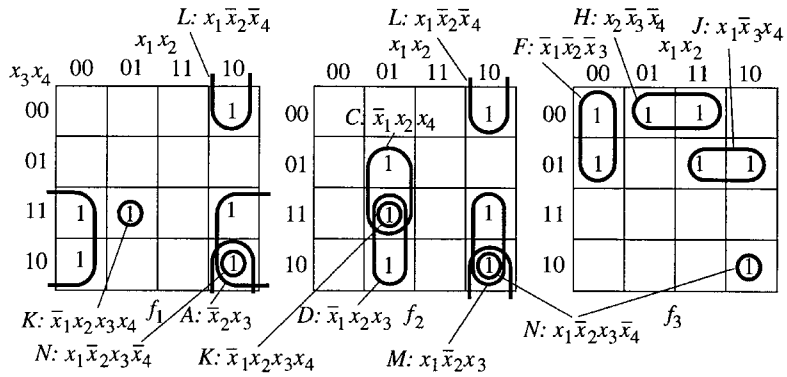


FIGURE 27.6 Covering f_1, f_2, and f_3 with the minimum number of paramount prime implicants.

2. Construct a network corresponding to the chosen set of paramount prime implicant loops.

Then, the network of 13 gates shown in Fig. 27.7 has been uniquely obtained. Letter *N* (i.e., $x_1\bar{x}_2x_3\bar{x}_4$), for example, is a paramount prime implicant for the product $f_1f_2f_3$, so the output of an AND gate with inputs, x_1, $\bar{x}_2$, x_3, and $\bar{x}_4$, is connected to the OR gates for f_1, f_2, and f_3.

3. Then delete unnecessary connections, or replace some logic gates by new ones, by the following steps from the logic networks derived in Step 2, by considering whether some Karnaugh maps have a paramount prime implicant loop that is inside another one, or adjacent to another one.

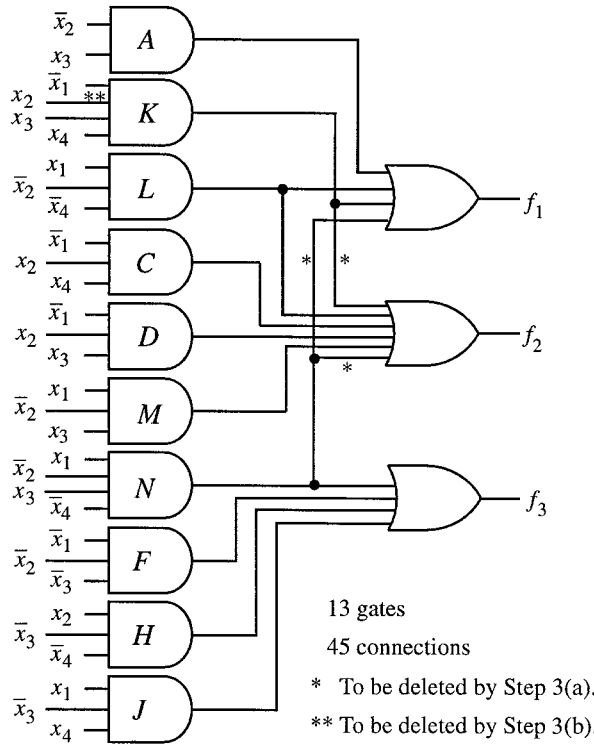


FIGURE 27.7 The network corresponding to *AKLCDMNFHJ*.

a. If, in a Karnaugh map for a function f_i, there is a paramount prime implicant loop that is inside another one, then the former is not necessary, because the 1-cells contained in this loop are covered by the latter loop. Thus, we can delete the connection from the AND gate corresponding to the former loop to the OR gate for f_i, without changing the output functions of the logic network.

For example, loop K in the map for f_2 is inside the loop C in Fig. 27.6. Thus, we can delete K from the map for f_2, because the 1-cell in K is covered by the loop C. This means that we can delete the connection (shown with * in Fig. 27.7) from the AND gate labeled K in Fig. 27.7. Similarly we can delete the loops N from the maps for f_1 and f_2, and accordingly, the connections (shown with * in Fig. 27.7) from the AND gates labeled N to the OR gates for f_1 and f_2.

b. If, in a Karnaugh map for a function f_i, there is a paramount prime implicant loop that is adjacent to another one, we may be able to expand the former by incorporating some 1-cells of the latter without having any 0-cells and at the same time replace the loop by the expanded loop (i.e., the number of logic gates unchanged). If we can do so, replace the former loop by the expanded loop. The expanded loop represents a product of fewer literals. Thus, we can delete the connection to the AND gate corresponding to the expanded loop without changing the output functions of the logic network.

For example, loop K in the map for f_1 has an adjacent loop A. Then we can replace the loop K by a larger loop (i.e., loop B in Fig. 27.5) by incorporating the 1-cell on its left, which represents the product $\bar{x}_1 x_3 x_4$. Also, K appears only in the map for f_2, beside K in the map for f_1. K in the map for f_2 is concluded to be eliminable, so the AND gate for K in Fig. 27.7 can be replaced by the new gate for B, keeping the number of logic gates unchanged. This means that we can delete the connection of input

x_2 (shown with ** in Fig. 27.7) to the AND gate labeled K in Fig. 27.7 (i.e., this is essentially replacement of K by B). Thus we can delete in totally, 4 connections from the logic network shown in Fig. 27.7, ending up with a simpler network with 13 gates and 41 connections.

When the number of paramount prime implicants is very small, we can find, directly on the maps, a minimum number of paramount prime implicants that cover all 1-cells in the maps for $f_1, f_2, \ldots, f_m$ (not their products) and derive a network with a minimum number of gates, using Procedure 27.1. But when the number of paramount prime implicants is many, the algebraic procedure of Section 4.6 in Ref. 5 is more efficient.

Procedure 27.1 with the deletion of unnecessary connections, however, may not necessarily yield a minimum number of connections as the secondary objective, although the number of gates is minimized as the primary objective. If we want to have a network with a minimum number of connections as the secondary objective, although the network has the same minimum number of gates as the primary objective, then we need to modify Procedure 27.1 and then delete unnecessary connections, as described as Procedure 5.2.1 in Ref. 5. But this procedure is more tedious and time-consuming.

Networks That Cannot be Designed by the Preceding Procedure

Notice that the design procedures in this section yield only a network that has all AND gates in the first level and all OR gates in the second level. (If 0-cells on Karnaugh maps are worked on instead of 1-cells in Procedure 27.1, we have a network with all OR gates in the first level and all AND gates in the second level.) If AND and OR gates are mixed in each level, or the network need not be in two levels, Procedure 27.1 does not guarantee the minimality of the number of logic gates.[6]

These two problems can be solved by the integer programming logical design method (to be mentioned in Chapter 31, Section 31.5), which is complex and can be applied to only networks of a small number of gates.

Applications of Procedure 27.1

Procedure 27.1 has important applications, such as PLAs, it but cannot be applied for designing large PLAs. For multiple-output functions with many variables, absolute minimization is increasingly time-consuming. Using BDD (described in Chapter 26), Coudert, Madre, and Lin extended the feasibility of absolute minimization.[2,3] But as it is becoming too time-consuming, we have to give up absolute minimization, resorting to heuristic minimization, such as a powerful method which is called MINI[4] and was later improved as ESPRESSO.[1]

References

1. Brayton, R., G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for Vlsi Synthesis*, Kluwer Academic Publishers, 1984.
2. Lin, B., O. Coudert, and J. C. Madre, "Symbolic prime generation for multiple-valued functions," *DAC* 1992, pp. 40-44, 1992.
3. Coudert, O., "On Solving Covering Problems," *DAC*, pp. 197-202, 1996.
4. Hong, S. J., R. G. Cain and D. L. Ostapko, "MINI: a heuristic approach for logic minimization," *IBM Jour. Res. Dev.*, pp. 443-458, Sept. 1974.
5. Muroga, S., *Logic Design and Switching Theory*, John Wiley & Sons, 1979 (Now available from Krieger Publishing Co.).
6. Weiner, P. and T. F. Dwyer, "Discussions of some flaws in the classical theory of two level minimizations of multiple-output switching networks," *IEEE Tr. Comput.*, pp. 184-186, Feb. 1968.

Muroga, S."Sequential Networks with AND and OR Gates"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

28

Sequential Networks

Saburo Muroga
University of Illinois at Urbana-Champaign

28.1 Introduction

A logic network is called a **sequential network** when the values of its outputs depend not only on the current values of inputs but also on some of the past values, whereas a logic network is called a **combinational network** when the values of its outputs depend only on the current values of inputs but not on any past values. Analysis and synthesis of sequential networks are far more complex than combinational networks. When reliable operation of the networks is very important, the operations of logic gates are often synchronized by clocks. Such clocked networks, whether they are combinational or sequential networks, are called **synchronous networks**, and networks without clock are called **asynchronous networks**.

28.2 Flip-Flops and Latches

Because in a sequential network the outputs assume values depending not only on the current values but also on some past values of the inputs, a sequential network must remember information about the past values of its inputs. Simple networks called **flip-flops** that are realized with logic gates are usually used as memories for this purpose. Semiconductor memories can also serve as memories for sequential networks, but flip-flops are used if higher speed is necessary to match the speed of logic gates. Let us explain the simplest flip-flops, which are called **latches**.

S-R Latches

The network in Fig. 28.1(a) which is called an ***S-R* latch**, consists of two NOR gates. Assume that the values at terminals S and R are 0, and the value at terminal Q is 0 (i.e., $S = R = 0$ and $Q = 0$). Since gate 1 has inputs of 0 and 0, the value at terminal $\overline{Q}$ is 1 (i.e., $\overline{Q} = 1$). Since gate 2 in the network has two inputs, 0 and 1, its output is $Q = 0$. Thus, signals 0 and 1 are maintained at terminals Q and $\overline{Q}$, respectively, as long as S and R remain 0. Now let us change the value at S to 1. Then, $\overline{Q}$ is changed to 0, and Q becomes 1 after a short time delay. Even if $Q = 1$ and $\overline{Q} = 0$ were their original values, the change of the value at S to 1 still yields $Q = 1$ and $\overline{Q} = 0$. In other words, Q is set to 1 by supplying 1 to S, no matter whether we originally had $Q = 0$, $\overline{Q} = 1$, or $Q = 1$, $\overline{Q} = 0$. Similarly, when 1 is supplied to R with S remaining at 0, Q and $\overline{Q}$ are set to 0 and 1, respectively, after a short time delay, no matter what values they had before. Thus, we get the first three combinations of the values of S and R shown in the table in Fig. 28.1(b). In other words, as long as $S = R = 0$, the values of Q and $\overline{Q}$ are not changed. If $S = 1$, Q is set to 1. If $R = 1$, Q is set to 0. Thus, S and R are called **set** and **reset terminals**, respectively. In order to let the latch work properly, the value 1 at S or R must be maintained until new values of Q and $\overline{Q}$ are established. The *S-R* latch is usually denoted as in Fig. 28.1(c). An *S-R* latch can also be realized with NAND gates, as shown in Fig. 28.1 (d). Latches and flip-flops have a direct reset-input terminal and a direct set-input terminal, although these input terminals are omitted in Fig. 28.1 and in the figures for other flip-flops for the sake of simplicity. These input terminals are convenient for initial setting to $Q = 1$, or resetting to $Q = 0$.

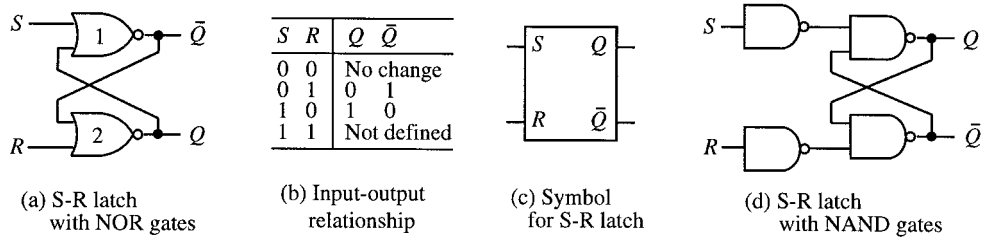


FIGURE 28.1 S-R latches.

When $S = R = 1$ occurs, the outputs Q and $\overline{Q}$ are both 0. If S and R simultaneously return to 0, these two outputs cannot maintain 0. Actually, a simultaneous change of S and R to 0 or 1 is physically impossible, often causing the network to malfunction, unless we make the network more sophisticated, such as synchronization of logic gates by a clock, as will be explained later. If S returns to 0 from 1 before R does, we have $Q = 0$ and $\overline{Q} = 1$. If R returns to 0 from 1 before S does, we have $Q = 1$ and $\overline{Q} = 0$. Thus, it is not possible to predict what values we will have at the outputs after having $S = R = 1$. The output of this network is not defined for $S = R = 1$, as this combination is not used. **For simplicity, let us assume that only one of the inputs to any network changes at a time, unless otherwise noted. This is a reasonable and important assumption.**

Flip-Flops

Usually, *S-R* latches are used in designing asynchronous sequential networks, although sequential networks can be designed without them. For example, the memory function can be realized with longer loops of gates than the loop in the latch, and also more sophisticated flip-flops than *S-R* latches can be used. For example, a loop consisting of a pair of inverters and a special gate called a transmission gate is used in CMOS networks, as we will see in Chapter 36, Section 36.2. But for synchronous networks, **raceless flip-flops** (described later in this chapter) are particularly important.

28.3 Sequential Networks in Fundamental Mode

In a sequential network, the value of the network output depends on both the current input values and some of past input values stored in the network. We can find what value the output has for each individual combination of input values, depending on what signal values are stored in the network. This essentially means interpreting the signals stored inside the network as new input variables called **internal variables** and then interpreting the entire network as a combinational network of the new input variables, plus the original inputs, which are **external input variables**.

Let us consider the sequential network in Fig. 28.2. Assume that the inputs are never changed unless the network is in a stable condition, that is, unless none of the internal variables is changing. Also assume that, whenever the inputs change, only one input changes at a time. Let y_1, $\bar{y}_1$, y_2, and $\bar{y}_2$ denote the outputs of the two *S-R* latches.

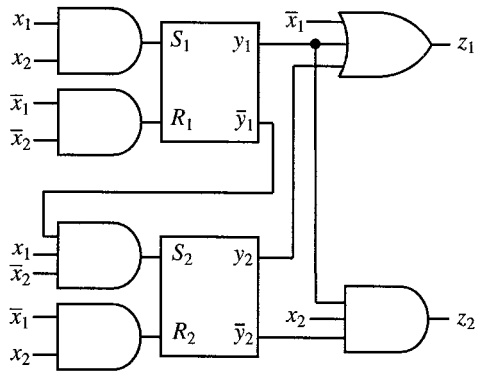


FIGURE 28.2 A sequential network with S-R latches.

Let us assume that $y_1 = y_2 = 0$ (accordingly, $\bar{y}_1 = \bar{y}_2 = 1$). $x_1 = 0$, and $x_2 = 1$. Then, as can be easily found, we have $z_1 = 1$ and $z_2 = 0$ for this combination of values. Because of $x_1 = 0$ and $x_2 = 1$, the inputs of the latches have values $R_2 = 1$ (accordingly, $y_2 = 0$) and $S_1 = S_2 = R_1 = 0$. Then y_1 and y_2 remain 0. As long as x_1, y_1, and y_2 remain 0 and x_2 remains 1, none of the signal values in this network changes and, consequently, this combination of values of x_1, x_2, y_1, and y_2 is called a **stable state**.

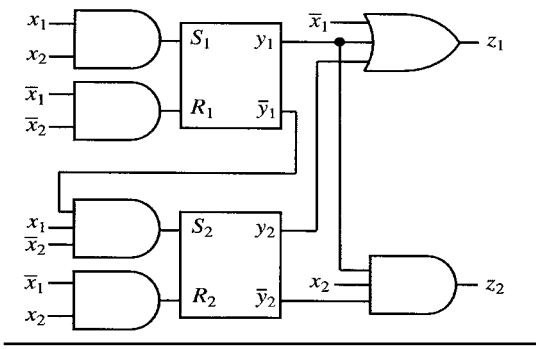
Next let us assume that x_1 is changed to 1, keeping $x_2 = 1$ and $y_1 = y_2 = 0$. For this new combination of input values, we get $z_1 = 0$ and $z_2 = 0$ after a time delay of τ, where τ is the switching time (delay time) of each gate, assuming for the sake of simplicity that each gate has the same τ (this is not necessarily true in practice). The two latches have new inputs $S_1 = 1$ and $S_2 = R_1 = R_2 = 0$ after a time delay of τ. Then, they have new output values $y_1 = 1$ (previously 0), $\bar{y}_1 = 0$, $y_2 = 0$, and $\bar{y}_2 = 1$ after a delay due to the response time of the latches. Outputs z_1 and z_2 both change from 0 to 1. After this change, the network does not change any further. Summarizing the above, we can say that, when the network has the combination $x_1 = x_2 = 1$, $z_1 = z_2 = y_1 = y_2 = 0$, the network does not remain in this combination, but changes into the new combination $x_1 = x_2 = z_1 = z_2 = y_1 = 1$, $y_2 = 0$. Also, outputs z_1 and z_2 change into $z_1 = z_2 = 1$, after assuming the values $z_1 = z_2 = 0$ temporarily. After the network changes into the combination $x_1 = x_2 = y_1 = 1$, $y_2 = 0$, and $z_1 = z_2 = 1$, it remains there. The combination $x_1 = x_2 = 1$ and $z_1 = z_2 = y_1 = y_2 = 0$ is called an **unstable state**. The transition from an unstable to a stable state, such as the above transition to the stable state, is the key to the analysis of a sequential network.

Transition Tables

This and other transitions for other combinations of values of x_1, x_2, y_1, y_2, z_1, and z_2 can be shown on the map in Table 28.1, which is called a **transition-output table,** showing the next values of y_1 and

y_2 as Y_1 and Y_2, respectively. The entry in each cell in Table 28.1 is next states Y_1, Y_2 and outputs z_1 and z_2. The above transition from $x_1 = 0$, $x_2 = 1$, $y_1 = y_2 = 0$ to $x_1 = x_2 = y_1 = 1$, $y_2 = 0$ is shown by the line with an arrow in Table 28.1. For this transition, the network is initially in the cell labeled with $*$ in Table 28.1; and when x_1 changes to 1, the network moves to the next cell labeled with the dot with this cell's entry, $Y_1 = 1$, $Y_2 = 0$, and $z_1 = z_2 = 0$, during the transient period. Here, it is important to notice that in this cell, next values of internal variables y_1 and y_2, i.e., Y_1 and Y_2 are shown but the network actually has current values of y_1 and y_2, that is, $y_1 = y_2 = 0$ during this transition period corresponding to this cell. Y_1 and Y_2 in this cell, which is in an unstable state, indicates what values y_1 and y_2 should take after the transition. Then, the network moves to the bottom cell where the values of y_1 and y_2 are identical to Y_1 and Y_2, respectively, as indicated by the line with an arrow, because Y_1 and Y_2 will be current y_1 and y_2 after the transition. The present values of the internal variables are shown with y_1 and y_2 in small letters and their next values are with Y_1 and Y_2 in capital letters. More specifically, variables y_1 and y_2 are called **present-state (internal) variables**, and Y_1 and Y_2 are called **next-state (internal) variables**. As can easily be seen, when the values of Y_1 and Y_2 shown inside a cell are identical to those of y_1 and y_2 shown on the left of the table, respectively, a state containing these values of Y_1, Y_2, y_1, and y_2 is stable, because there is no transition of y_1 and y_2 into a new, different state. Next-state variables in stable states are encircled in Table 28.1. Each column in this table corresponds to a combination of values of network inputs x_1 and x_2, that is, an **input state**. Each row corresponds to a combination of values of internal variables y_1 and y_2, that is, a present **internal state**. Thus, each cell corresponds to a **total state**, or simply a **state**; that is, a combination of values of x_1, x_2, y_1, and y_2.

TABLE 28.1 Transition-Output Table in Fundamental Mode



When only next states Y_1 and Y_2 are shown, instead of showing all next states Y_1, Y_2, and outputs z_1 and z_2 in each cell of a transition-output table, the table is called a **transition table**, and when only outputs z_1 and z_2 are shown, the table is called an **output table**.

Fundamental Mode

A sequential network is said to be operating in **fundamental mode** when the transition occurs horizontally in the transition table corresponding to each network input change and, then, unless the new state in the same row is stable, the transition continues vertically (not diagonally), settling in a new stable state, such as the transition shown with the line with an arrow in Table 28.1. The transitions of the network take place under the assumption that only one of the network inputs changes at a time, only when the network is not in transition; that is, only when the network is settled in a stable state.

By using a transition-output table, the output sequence [i.e., the sequence of output values corresponding to any input sequence, (i.e., the sequence of input values)] can be easily obtained. In other words, the output sequence can be obtained by choosing columns corresponding to the input sequence and then moving to stable states whenever states.

28.4 Malfunctions of Asynchronous Sequential Networks

An asynchronous sequential network does not work reliably unless appropriate binary numbers are assigned to internal variables for each input change.

Racing Problem of Sequential Networks

A difference in time delays of signal propagation along different paths may cause a malfunction of an asynchronous sequential network that is called a **race**. This is caused by a difference in the delay times of gates.

Let us consider the transition-output table of a certain asynchronous sequential network shown in Table 28.2.

TABLE 28.2 Transition-Output Table in Fundamental Mode

y_1y_2 \ x_1x_2	00	01	11	10
00	(00), 0	(00), 0	11, 0**	01, 0
01	(01), 0	00, 0	11, 1	(01), 1
11	(11), 0	10, 0	10, 1	(11), 1
10	—	00, 0	(10), 0	01, 0*

Y_1Y_2, z

Suppose that the network is in the stable state $(x_1, x_2, y_1, y_2) = (1110)$. If the inputs change from $(x_1, x_2) = (11)$ to (10), the network changes into the unstable state $(x_1, x_2, y_1, y_2) = (1010)$, marked with $*$ in Table 28.2. Because of $y_1 = 1$, $y_2 = 0$, $Y_1 = 0$ and $Y_2 = 1$, two logic gates whose outputs represent y_1 and y_2 in the network must change their output values simultaneously. However, it is extremely difficult for the two gates to finish their changes at exactly the same time, because no two paths that reach these gates have identical time delays. Actually, one of these two logic gates finishes its change before the other does. In other words, we have one of the following two cases:

1. y_2 changes first, making the transition of (y_1, y_2) from (10) to (11) and leading the network into stable state $(x_1, x_2, y_1, y_2) = (1011)$.
2. y_1 changes first, making the transition of (y_1, y_2) from (10) to (00) and reaching the unstable state $(x_1, x_2, y_1, y_2) = (1000)$, and then y_2 changes, making the further transition of (y_1, y_2) from (00) to (01), settling in the stable state $(x_1, x_2, y_1, y_2) = (1001)$.

Thus, the network will go to either state $(y_1, y_2) = (11)$, in case 1, or state $(y_1, y_2) = (00)$, in case 2, instead of going directly to $(y_1, y_2) = (01)$. If $(y_1, y_2) = (11)$ is reached, state (11) is stable, and the network stops here. If (00) is reached, this is an unstable state, and another transition to the next stable state, (01), occurs. State (01) is the **destination stable state** (i.e., desired stable state), but (11) is not. Thus, depending on which path of gates works faster, the network may malfunction. This situation is called a **race**. The network may or may not malfunction, depending on which case actually occurs.

Next, suppose that the network is in the stable state $(x_1, x_2, y_1, y_2) = (0100)$ and that inputs $(x_1, x_2) = (01)$ change to (11). The cell labeled with ** in Table 28.2 has $(Y_1\ Y_2) = (11)$. Thus, y_1 and y_2 must have simultaneous changes. Depending on which one of the two logic gates whose outputs represent y_1 and y_2 changes its output faster, there are two possible transitions. But in this case, both end up in the same stable state, $(y_1, y_2) = (10)$. This is another race, but the network does not have a malfunction depending on the order of change of internal variables. Hence, this is called a **noncritical race**, whereas the previous race for cases 1 and 2 is termed a **critical race** because the performance is unpredictable (it is hard to predict which path of gates will have the signal change faster), possibly causing the network to malfunction. We may have more complex critical racing. In other words, if we have a network such that the output of the gate whose output represents y_1 feeds back to the input of a gate on a path that reaches the gate whose output represents y_1, the y_1 may continue to change its value from 0 to 1 (or from 1 to 0), then change back to 0 by feedback of new value 1, and so on. This **oscillatory race** may continue for a long time.

Remedies for the Racing Problem

Whenever a network has critical races, we must eliminate them for reliable operation. One approach is to make the paths of gates have definitely different time delays. This approach, however, may not be most desirable for the following reasons. First, the speed may be sacrificed by adding gates for delay. Second, there are cases where this approach is impossible if a network contains more than one critical race. By eliminating a critical race in one column in the transition table by using a path of different time delay, a critical race in another column may occur or may be aggravated.

A better approach is to choose some entries in the transition table so that no critical races occur. Then, on the basis of this new table, we synthesize a network with the desired performance, according to the synthesis method to be described later. A change of entries in some unstable states without changing destination stable states, such that only one internal variable changes its value at a time, is one method for eliminating critical races. The critical race discussed above can be eliminated from Table 28.2 by replacing the entry marked with * by (00), as shown in Table 28.3, where only Y_1 and Y_2 are shown without the network output z. If every entry that causes the network to malfunction can be changed in this manner, a reliable network with the performance desired in the original is produced, because every destination stable state to which the network should go is not changed, and the output value for the destination stable state is also not changed. (We need not worry about noncritical races, since they cause no malfunctions.) However, sometimes, there are entries for some unstable states that cannot be changed in this manner. For example, consider Table 28.4 (some states are not shown, for the sake of simplicity). The entry (01) for $(x_1, x_2, y_1, y_2) = (0010)$ is such an entry

TABLE 28.3 Transition Table in Fundamental Mode, Derived by Modifying Table 28.2

y_1y_2 \ x_1x_2	00	01	11	10
00	(00)	(00)	11	01
01	(01)	00	11	(01)
11	(11)	10	10	(11)
10	—	00	(10)	00

Y_1Y_2

TABLE 28.4 Transition Table in Fundamental Mode

y_1y_2 \ x_1x_2	00	01	11	10
00	(00)			
01	(01)			
11	(11)			
10	01	(10)		

Y_1Y_2

and causes a critical race. [State $(x_1, x_2, y_1, y_2) = (0010)$ in Table 28.2 may have the same property, but actually, the network never enters this state because of the assumption that inputs x_1 and x_2 do not change simultaneously.] Since this entry requires simultaneous transitions of two internal variables, y_1 and y_2, we need to change it to (00) or (11). Both lead the network to stable states different from the destination stable state (01).

When a change of entries in unstable states does not work, we need to redesign the network completely by adding more states (e.g., 8 states with 3 internal variables, y_1, y_2, and y_3, instead of 4 states with 2 internal variables, y_1 and y_2, for Table 28.4) without changing transitions among stable states, as we will see later in this chapter. This redesign may include the reassignment of binary numbers to states, or the addition of intermediate unstable states through which the network goes from one stable state to another. The addition of more states, reassignment of binary numbers to states, and addition of intermediate unstable states for this redesign, however, is usually cumbersome. So, designers usually prefer the use of synchronous sequential networks because design procedures are simpler and the racing problem, including oscillatory races due to the existence of feedback loops in a sequential network, is completely eliminated.

28.5 Different Tables for the Description of Transitions of Sequential Networks

When we look at a given sequential network from the outside, we usually cannot observe the values (i.e., binary numbers) of the internal variables; that is, the inside of the network (e.g., if the network is implemented in an IC package, no internal variables may appear at its pins). Also, binary numbers are not very convenient to use. Therefore, binary numbers for the internal states of the network may be replaced in a transition-output table by arbitrary letters or decimal numbers. The table that results is called a **state-output table**. For example, Table 28.5 is the state-output table obtained from the transition-output table in Table 28.2, where y_1y_2 is replaced by s and Y_1Y_2 is replaced by S. A present state of the internal state is denoted by s and its next state by S.

In some state tables, networks go from one stable state to another through more than one unstable state, instead of exactly one unstable state. For example, in the state-output table in Table 28.5, when inputs (x_1, x_2) change from (00) to (01), the network goes from stable state $(x_1, x_2, s) = (00C)$ to another stable state, $(01A)$, by first going to unstable state $(01C)$, then to intermediate unstable state $(01D)$, and finally to stable state $(01A)$. Here, $(x_1, x_2, s) = (01D)$ is called an **intermediate unstable state**. Such a multiple transition from one stable state to another in a state-output table cannot be observed well from outside the network, and is not important as far as the external performance of the network is concerned. Even if each intermediate unstable state occurring during multiple transitions is replaced by the corresponding ultimate stable state, it does not make any difference if the

TABLE 28.5 State-Output Table Derived from Table 28.2

s \ x_1x_2	00	01	11	10
A	(A), 0	(A), 0	C, 0	B, 0
B	(B), 0	A, 0	C, 1	(B), 1
C	(C), 0	D, 0	D, 1	(C), 1
D	—	A, 0	(D), 0	B, 0

S, z

network performance is observed from outside. Such a table is called a **flow-output table**. The flow-output table corresponding to the state-output table in Table 28.5 is shown in Table 28.6, where D in the intermediate unstable state $(x_1, x_2, s) = (01C)$, for example, in Table 28.5 is replaced by A in Table 28.6.

TABLE 28.6 Flow-Output Table Derived from Table 28.5

s \ x_1x_2	00	01	11	10
A	(A), 0	(A), 0	D, 0	B, 0
B	(B), 0	A, 0	D, 1	(B), 1
C	(C), 0	A, 0	(D), 1	(C), 1
D	—	A, 0	(D), 0	B, 0

S, z

28.6 Steps for the Synthesis of Sequential Networks

Let us first introduce the general model for sequential networks and then describe a sequence of steps for designing a sequential network.

General Model of Sequential Networks

A sequential network may be generally expressed in the schematic form shown in Fig. 28.3. A large block represents a loopless network of logic gates only (without any flip-flops). All loops with and without flip-flops (i.e., both loops that do not contain flip-flops and loops that contain flip-flops) are drawn outside the large block.

This loopless network has external input variables $x_1, \ldots, x_n$, and internal variables $y_1, \ldots, y_p$, as its inputs. It also has external output variables $z_1, \ldots, z_m$, and **excitation variables** $e_1, \ldots, e_q$, as its outputs. Some of the excitation variables $e_1, \ldots, e_q$ are inputs to the flip-flops, serving to excite the flip-flops. The remainder of the excitation variables are starting points of the loops without flip-flops, which end up at some of the internal variables. For loops without flip-flops, $e_i = Y_i$ holds for each i. For example, the network in Fig. 28.2 can be redrawn in the format of Fig. 28.3, placing the latches outside the loopless network.

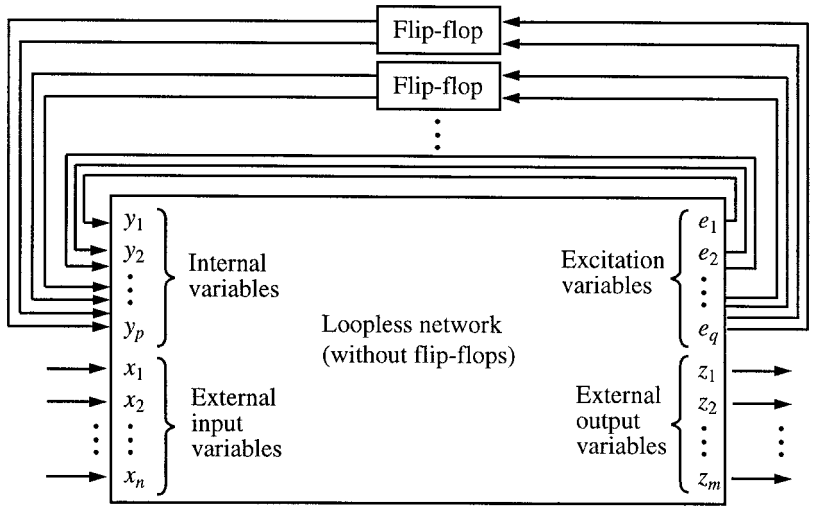


FIGURE 28.3 A general model of a sequential network.

Synthesis as a Reversal of Network Analysis

The outputs $e_1, \ldots, e_q, z_1, \ldots, z_m$ of the combinational network inside the sequential network in Fig. 28.3 express logic functions that have $y_1, \ldots, y_p, x_1, \ldots, x_n$ as their variables. Thus, if these logic functions are given, the loopless network can be designed by the methods discussed in earlier chapters. This means that we have designed the sequential network, since the rest of the general model in Fig. 28.3 is simply loops with or without flip-flops to be placed outside this combinational network. But we cannot derive these logic functions $e_1, \ldots, e_q, z_1, \ldots, z_m$ directly from the given design problem, so let us find, in the following, what gap to fill in.

When loops have no flip-flops, $e_i = Y_i$ holds for each i, where Y is the next value of y, and the binary-value relationship between the inputs $y_1, \ldots, y_p, x_1, \ldots, x_n$ and the outputs $Y_1, \ldots, Y_q, z_1, \ldots, z_m$ of the combinational network inside the sequential network in Fig. 28.3 is expressed by a transition-output table. But when loops have flip-flops, we need the relationship between $e_1, \ldots, e_q, z_1, \ldots, z_m$ and $y_1, \ldots, y_p, x_1, \ldots, x_n$, where $e_1, \ldots, e_q$ are the inputs $S_1, R_1, S_2, R_2 \ldots$ of the latches. A table that shows this relationship is called an **excitation-output table**. For this purpose, we need to derive the output-input relation of the S-R latch, as shown in Table 28.8, by reversing the input-output relationship in Table 28.7, which shows the output y of a latch and its next value Y for every feasible combination of the values of S and R.

TABLE 28.7 Input-Output Relationship of a S-R Latch

S	R	y	Y
0	0	0	0
		1	1
0	1	0	0
		1	0
1	0	0	1
		1	1

In Table 28.8, d's mean don't-care conditions. For example, $y = Y = 0$ in Table 28.8 results from $S = R = 0$ and also from $S = 0$, $R = 1$ in Table 28.7. Therefore, the first row in Table 28.8, $y = Y = 0$,

TABLE 28.8 Output-Input Relationship of a *S-R* Latch

y	Y	S	R
0	0	0	d
0	1	1	0
1	0	0	1
1	1	d	0

$S = 0$, $R = d$, is obtained, because $y = Y = 0$ results from $S = 0$ only, but R can be 0 or 1; that is, don't-care, d. By using Table 28.8, the transition-output table in Table 28.1, for example, is converted into the excitation-output table in Table 28.9. For example, corresponding to $y_1 = Y_1 = 0$ in the first row and the first column in Table 28.1, $S_1 = 0$, $R_1 = d$ is obtained as the first $0d$ in the cell in the first row and the first column of Table 28.9, because the first row in Table 28.8 corresponds to this case. Of course, when a network, for example, Fig. 28.2 is given, we can derive the excitation-output table directly from the network in Fig. 28.2 rather than the transition-output table in Table 28.1 (which was derived for Fig. 28.2). But when we are going to synthesize a sequential network from a transition-output table, we do not have the network yet and we need to construct an excitation-output table from a transition-output table, using Table 28.8.

TABLE 28.9 Excitation-Output Derived from Table 28.1

	x_1, x_2			
y_1y_2	00	01	11	10
00	$0d$, $0d$, 10	$0d$, $0d$, 10	10, $0d$, 00	$0d$, 10, 00
01	$0d$, $d0$, 10	$0d$, 01, 10	10, $d0$, 10	$0d$, $d0$, 10
11	01, $d0$, 10	$d0$, 01, 10	$d0$, $d0$, 10	$d0$, $d0$, 10
10	$0d$, $0d$, 10	$d0$, $0d$, 11	$d0$, $0d$, 11	$d0$, $0d$, 10
	S_1R_1, S_2R_2, z_1z_2			

Design Steps for Synthesis of Sequential Networks

A sequential network can be designed in the sequence of steps shown in Fig. 28.4. The required performance of a network to be designed for the given problem is first converted into a flow-output table. Then this table is converted into a state-output table, and next into a transition-output table, by choosing an appropriate assignment of binary numbers to all states. Then, the transition-output table is converted into an excitation-output table if the loops contain flip-flops. If the loops contain no flip-flops, the excitation-output table need not be prepared, since it is identical to the transition-output table. Finally, a network is designed, using the logic design procedures discussed in the preceding chapters.

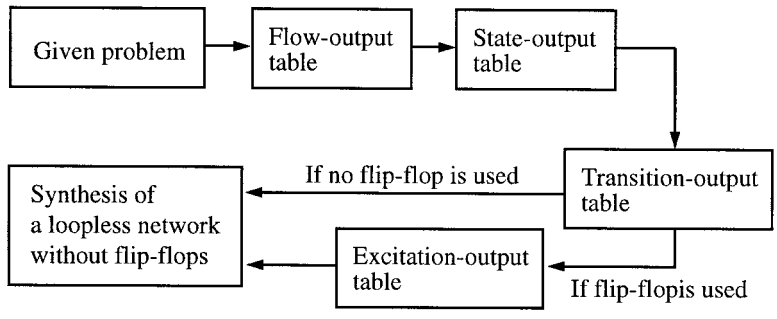


FIGURE 28.4 Steps for designing a sequential network.

28.7 Synthesis of Sequential Networks

Since the use of clocks has many advantages, synchronous sequential networks with clocks are in many cases preferred to asynchronous sequential networks. In addition to the ease of design and elimination of hazards including racing problems, the speed of logic networks can sometimes (e.g., Domino CMOS) be improved by synchronizing the operations of logic gates by clocks, and waveforms of signals can be reshaped into clean ones.

For synchronous sequential networks, more sophisticated flip-flops than latches are usually used, along with clocks. With these flip-flops, which are called raceless flip-flops, network malfunctioning due to hazards can be completely eliminated. Design of sequential networks with raceless flip-flops and clocks is much simpler than that of networks in fundamental mode, since we can use simpler flow-output and state-output tables, which are said to be in skew mode, and multiple changes of internal variables need not be avoided in assigning binary numbers to states.

Raceless Flip-Flops

Raceless flip-flops are flip-flops that have complex structures but eliminate network malfunctions due to races. Most widely used raceless flip-flops are master-slave flip-flops and edge-triggered flip-flops. A **master-slave flip-flop** (or simply, an MS flip-flop) consists of a pair of flip-flops called a master flip-flop and a slave flip-flop. Let us explain the features of master-slave flip-flops based on the ***J-K* master-slave flip-flop** shown in Fig. 28.5. For the sake of simplicity, all the gates in these flip-flops are assumed to have equal delay times, although in actual electronic implementations, this may not be true. Its symbol is shown in Fig. 28.6, where the letters *MS* are shown inside the rectangle. Each action of the master-slave flip-flop is controlled by the leading and trailing edges of a clock pulse, as explained in the following.

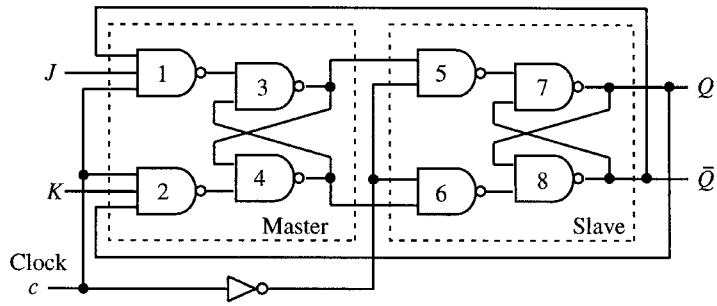


FIGURE 28.5 *J-K* master-slave flip-flop.

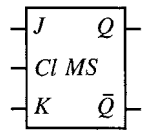


FIGURE 28.6 Symbol for *J-K* master-slave flip-flop.

The *J-K* master-slave flip-flop in Fig. 28.5 works with the clock pulse, as illustrated in Fig. 28.7, where the rise and fall of the pulse are exaggerated for illustration. When the clock has the value 0 (i.e., $c = 0$) NAND gates 1 and 2 in Fig. 28.5 have output values 1, and then the flip-flop consisting of gates 3 and 4 does not change its state. As long as the clock stays at 0, gates 5 and 6 force the flip-flop consisting of

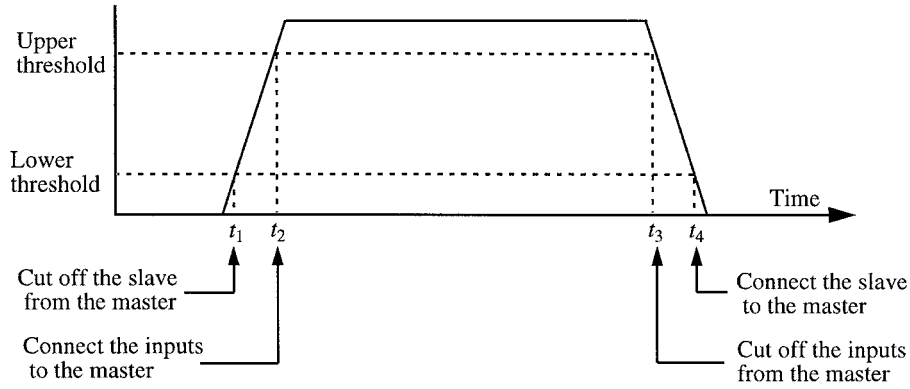


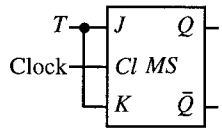
FIGURE 28.7 Clock pulse waveform.

gates 7 and 8 to assume the same output values as those of the flip-flop consisting of gates 3 and 4 (i.e., the former is slaved to the latter, which is the master). Each of the master and slave is a slight modification of the latch in Fig. 28.1(d).

When the clock pulse starts to rise to the lower threshold at time t_1 in Fig. 28.7, gates 5 and 6 are disabled: in other words, the slave is cut off from the master. (The lower threshold value of the clock pulse still presents the logic value 0 to gates 1 and 2. The clock waveform is inverted to gates 5 and 6 through the inverter. The inputs to gates 5 and 6 from the inverter present the logic value 0 also to gates 5 and 6, because the inverted waveform is still close to logic value 1 but is not large enough to let gates 5 and 6 work. The inverter is actually implemented by a diode which is forward-biased, so that the network works in this manner.) When the clock pulse reaches the upper threshold at t_2, (i.e., $c = 1$), gates 1 and 2 are enabled, and the information at J or K is read into the master flip-flop through gate 1 or 2. Since the slave is cut off from the master by disabled gates 5 and 6, the slave does not change its state, maintaining the previous output values of Q and $\overline{Q}$. When the clock pulse falls to the upper threshold at t_3 after its peak, gates 1 and 2 are disabled, cutting off J and K from the master. In other words, the outputs of 1 and 2 become 1, and the master maintains the current output values. When the clock pulse falls further to the lower threshold at t_4, gates 5 and 6 are enabled and the information stored at the master is transferred to the slave, gates 7 and 8.

The important feature of the master-slave flip-flop is that the reading of information into the flip-flop and the establishment of new output values are done at different times; in other words, the outputs of the flip-flop can be completely prevented from feeding back to the inputs, possibly going through some gates outside the master-slave flip-flop, while the network that contains the flip-flop is still in transition. The master-slave flip-flop does not respond to its input until the leading edge of the next clock pulse. Thus, we can avoid oscillatory races. Now we have a *J-K* flip-flop that works reliably, regardless of how long a clock pulse or signal 1 at terminal J or K lasts, since input gates 1 and 2 in Fig. 28.5 are gated by output gates 7 and 8, which do not assume new values before gates 1 and 2 are disconnected from J and K. As we will see later, sequential networks that work reliably can be constructed compactly with master-slave flip-flops, without worrying about hazards.

Other types of master-slave flip-flops are also used. The ***T* (type) master-slave flip-flop** (this is also called toggle flip-flop, trigger flip-flop, or T flip-flop) has only a single input, labeled T, as shown in Fig. 28.8(a), which is the *J-K* master-slave flip-flop with J and K tied together as T. Whenever we have $T = 1$ during the clock pulse, the outputs of the flip-flop change, as shown in Fig. 28.8(b). The T-type flip-flop, denoted as Fig. 28.8(c), is often used in counters. The ***D* (type) master-slave flip-flop** (D implies "delay") has only a single input D, and is realized, as shown in Fig. 28.9(a). As shown in Fig. 28.9(b), no matter what value Q has, Q is set to the value of D that is present during the clock pulse. The D-type flip-flop is used for delay of a signal or data storage and is denoted by the symbol shown in Fig. 28.9(c).

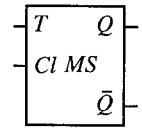


(a) *T*-type master-slave flip-flop based on *J-K* master-slave flip-flop

T	Q $\bar{Q}$
0	No change
1	$\bar{Q}_0$ Q_0

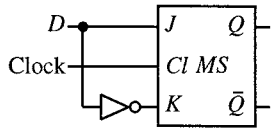
Q_0 is the previous value of Q

(b) Input-output relationship



(c) Symbol for (a)

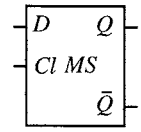
FIGURE 28.8 *T*-type master-slave flip-flop.



(a) *D*-type master-slave flip-flop based on *J-K* master-slave flip-flop

D	Q $\bar{Q}$
0	0 1
1	1 0

(b) Input-output relationship



(c) Symbol for (a)

FIGURE 28.9 *D*-type master-slave flip-flop.

Edge triggered flip-flops are another type of raceless flip-flop. Either the leading or the trailing edge of a clock pulse (not both) causes a flip-flop to respond to an input, and then the input is immediately disengaged from the flip-flop. Edge-triggered flip-flops are mostly used in the same manner as master-slave flip-flops.

Example of Design of a Synchronous Sequential Network

Let us now explain synthesis of a synchronous sequential network with an example.

Specification for the Design Example

Suppose we want to synthesize a clocked network with two inputs, x_1 and x_2, and single output z under the following specifications:

1. Inputs do not change during the presence of clock pulses, as illustrated in Fig. 28.10. Inputs x_1 and x_2 cannot assume value 1 simultaneously during the presence of clock pulses. During clock pulses, an input signal of value 1 appears at exactly one of two inputs, x_1 and x_2, of the network, or does not appear at all.
2. The value of z changes as follows.
   a. The value of z becomes 1 when the value 1 appears during the clock pulse at the same input at which the last value 1 appeared. Once z becomes 1, z remains 1 regardless of the presence or absence of clock pulses, until signal 1 starts to appear at the other input during clock pulses. (This includes the following case. Suppose that we have $z = 0$, when signal 1 appears at one of the inputs during a clock pulse. Then, signal 0 follows at both x_1 and x_2 during the succeeding clock pulses. If signal 1 comes back to the same input, z becomes 1.) As illustrated in Fig. 28.10, z becomes 1 at time t_1 at the leading edge of the second pulse because signal 1 is repeated at input x_1. Then z continues to remain 1 even though signal 1 appears at neither input at the third pulse starting t_2.
   b. The value of z becomes 0 when the value 1 appears at the other input during the clock pulse. Once z becomes 0, z remains 0 regardless of the presence or absence of clock pulses, until signal 1 starts to appear at the same input during clock pulses. In Fig. 28.10, z continues to be 1 until the leading edge of the pulse starting at time t_3. Then z continues to remain 0 until the time t_4.

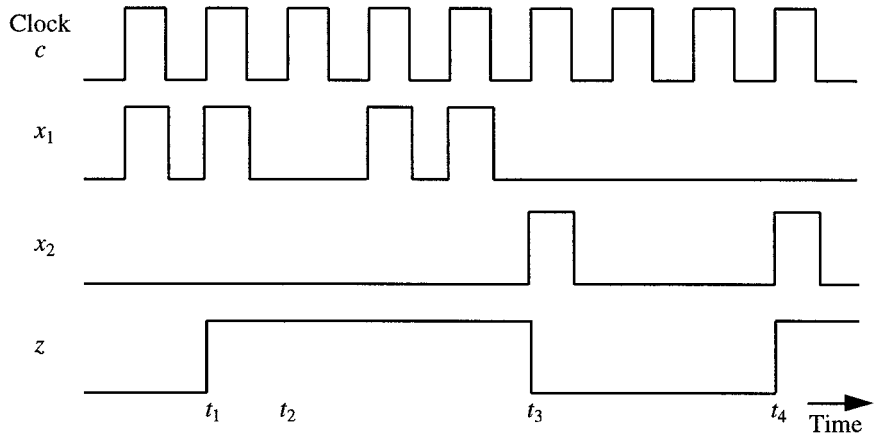


FIGURE 28.10 Waveform for the design example.

Let us prepare a flow-output table for this design problem in the following steps.

1. An output value of z must depend on an internal state only, because z has to maintain its value until next change. Let us assume that there are exactly two states, A and B, such that $z = 0$ when the network is in state A and $z = 1$ when the network is in state B. (We will try more than two states if two are found insufficient.)
2. Assume that during the absence of a clock pulse, the network is in state A. Let c denote the clock. Since the network must stay in this state as long as $c = 0$, the next state, S, in the column for $c = 0$ must be A, as shown in Table 28.10(a). Similarly, the next state for the second row in the column $c = 0$ must be B. It is to be noted that during $c = 0$, there are four combinations of values of x_1 and x_2 (i.e., $x_1 = x_2 = 0$; $x_1 = 1$ and $x_2 = 0$; $x_1 = 0$ and $x_2 = 1$; and $x_1 = x_2 = 1$). But the states of the network to be synthesized is irrelevant of the values of x_1 and x_2. Thus, in Table 28.10, we have only one column corresponding to $c = 0$, instead of four columns.

TABLE 28.10 Two States Are Not Enough

		$c = 1$				
		x_1, x_2				
z	s	$c = 0$	00	01	11	10
0	A	A, 0			—	A
1	B	B, 1			—	
		S, z				

(a) Entering A is not correct.

		$c = 1$				
		x_1, x_2				
z	s	$c = 0$	00	01	11	10
0	A	A, 0			—	B
1	B	B, 1			—	
		S, z				

(b) Entering B is not correct.

3. When the network is in state A during $c = 0$, suppose that we have $x_1 = 1$ at the next clock pulse. Let us choose A as the next state, S, as shown in the last column in Table 28.10(a). But this choice means that, if we apply value 1 repeatedly at x_1, the network goes back and forth between the two states in the first and last columns in the first row in Table 28.10(a). Then $z = 1$ must result from the specification of the network performance; but since the network stays in the first row, we must have $z = 0$. This is a contradiction. Thus, the next state for $(x_1, x_2, s) = (10A)$ cannot be A.
4. Next assume that the next state for $(x_1, x_2, s) = (10A)$ is B, as shown in Table 28.10(b). Suppose that the value 1 has been alternating between x_1 and x_2. If we had the last 1 at x_2, the network must be currently in A because of $z = 0$ for alternating 1's. When the next 1 appears at x_1, $z = 0$ must still hold because the value 1 is still alternating. But the network will produce $z = 1$ for state $(10A)$, because B is assumed to correspond to $z = 1$. This is a contradiction, and the choice of B is also wrong.

5. In conclusion, two states are not sufficient, so we try again with more states. For this example, considering only two states corresponding to $z = 0$ and 1 is not appropriate.
6. In the above, we had contradictions by assuming only two states, A and B, corresponding to $z = 0$ and 1; we did not know which input's 1 led to each state. Thus, in addition to the values of z, let us consider which input had the last 1. In other words, we have four states corresponding to the combinations of z and the last 1, as shown in Table 28.11. At this stage, we do not know whether or not three states are sufficient. But let us assume four states for the moment. As a matter of fact, both three states and four states require two internal variables. Hence, in terms of the number of internal variables, it does not matter whether we have three or four states, although the two cases may lead to different networks.

TABLE 28.11 Flow-Output Table in Skew-Mode

Last 1 at	z	s	$c = 0$	$c = 1$ x_1, x_2 00	01	11	10
x_1	0	A	$A, 0$	$A, 0$	$C, 0$	—	$B, 1$
x_1	1	B	$B, 1$	$B, 1$	$C, 0$	—	$B, 1$
x_2	0	C	$C, 0$	$C, 0$	$D, 1$	—	$A, 0$
x_2	1	D	$D, 1$	$D, 1$	$D, 1$	—	$A, 0$
					S, z		

7. **Derivation of a flow-output table in skew mode**: Let us form a flow-output table, assuming the use of J-K master-slave flip-flops. In the column of $c = 0$ in Table 28.11, the network must stay in each state during the absence of a clock pulse. Thus, all the next states S in all the cells in the column of $c = 0$ must be identical to s in each row. Suppose that the network is in state $(c, x_1, x_2, s) = (000A)$ with output $z = 0$ after having the last 1 at x_1. When the value 1 appears at x_1 and c becomes 1, the next state S must be B for the following reason. When the current clock pulse disappears, this 1 at x_1 will be "the last 1" at x_1 (so S must be A or B) and z will have to be 1 because of the repeated occurrence of 1's at x_1. (This contradicts $z = 0$, which we will have if A is entered as S.) Hence, the possibility of S being A is ruled out, and S must be B. Since value 1 is repeated at x_1, we have $z = 1$. The next states and the values of output z in all other cells in Table 28.11 can be found in a similar manner.

   Let us analyze how a transition among stable states occurs in this table. According to the problem specification, inputs x_1 and x_2 can change only during the absence of clock pulses.

   Suppose that during the absence of a clock pulse, the network is in state $(c, x_1, x_2, s) = (000A)$ in Table 28.11. Suppose that inputs (x_1, x_2) change from (00) to (10) sometime during the absence of a clock pulse and $(x_1, x_2) = (10)$ lasts at least until the trailing edge of the clock pulse. If this transition is interpreted on Table 28.11, the network moves from $(c, x_1, x_2, s) = (000A)$ to $(110A)$, as shown by the dotted-line with an arrow, at the leading edge of the clock pulse. Then the network must stay in this state, $(110A)$, until the trailing edge of the clock pulse, because the outputs of the J-K master-slave flip-flops keep the current values during the clock pulse, changing to its new values only at the trailing edge of the clock pulse and thus the internal state s does not change yet to its new state S. (This is different from the fundamental mode, in which the network does not stay in this state unless the state is a stable one, and vertically moves to a stable state in a different row in the same column.) Then the network moves from $(c, x_1, x_2, s) = (110A)$ to $(010B)$, as shown by the solid-line with an arrow in Table 28.11, when s assumes the new state S at the trailing edge of the clock pulse. Thus, the transition occurs horizontally and then **diagonally** in Table 28.11. This type of transition is called **skew mode**, in order to differentiate it from the fundamental mode.

Notice, however, that in the new state (110*A*) at the leading edge of the clock pulse in the above transition from (c, x_1, x_2, s) = (000*A*), the network z assumes the new output value. Consequently, in this new stable state during $c = 1$ in Table 28.11, the new current value of the network output, z, is shown, while the value of the internal state in this new stable state shows the next state S, though the network is actually in s. In the fundamental mode, when the network moves horizontally to a new unstable state in the same row in the state-output table, the current value of the network output is shown and the internal state represents the next state, S (in this sense the situation is not different), but the network output lasts only during a short, transient period, unless the new state is stable. In contrast, in skew mode, the output value for the new state is not transient (because the network stays in this state during the clock pulse) and is essential in the description of the network performance.

Let us synthesize a network for this flow-output table in skew mode later.

8. **Derivation of a flow-output table in fundamental mode**: Next let us try to interpret this table in fundamental mode. Table 28.11 shows that, when the network placed in state (c, x_1, x_2, s) = (000*A*) receives $x_2 = 1$ before the appearance of the next pulse, the next state will be *C* at the leading edge of the next pulse. But the network goes to unstable state (c, x_1, x_2, s) = (101*C*) that has entry *D*, since if we assume fundamental mode, it must move vertically, instead of the diagonal transition in skew mode. Hence, the network must go further to stable state (101*D*), without settling in the desired stable state, *C*, if the network still keeps $x_2 = 1$ and $c = 1$. Therefore, the network cannot be in fundamental mode. (Recall that the next state entries in a flow-output table, unlike a state-output table, do not show intermediate unstable states but do show the destination stable states.) The above difficulty can be avoided by adding two new rows, *E* and *F*, as shown in Table 28.12. When the network placed in state (c, x_1, x_2, s) = (000*A*) receives $x_2 = 1$, the network goes to the new stable state *F* in column (x_1, x_2) = (01) and in row *F*. For this state *F*, $z = 0$, without causing contradiction. When the clock pulse disappears, the network goes to stable state (000*C*) after passing through unstable state (000*F*). The problem with the other states is similarly eliminated. All stable states are encircled as stable states.

TABLE 28.12 Flow-Output Table in Fundamental Mode

				$c = 1$			
				x_1, x_2			
Last 1 at	z	s	$c = 0$	00	01	11	10
x_1	0	*A*	(*A*), 0	(*A*), 0	*F*, 0	—	*B*, *d*
x_1	0	*E*	*A*, 0	—	—	—	(*E*), 0
x_1	1	*B*	(*B*), 1	(*B*), 1	*F*, *d*	—	(*B*), 1
x_2	0	*C*	(*C*), 0	(*C*), 0	*D*, *d*	—	*E*, 0
x_2	0	*F*	*C*, 0	—	(*F*), 0	—	—
x_2	1	*D*	(*D*), 1	(*D*), 1	(*D*), 1	—	*E*, *d*
					S, z		

The values of z for stable states are easily entered. The values of z for unstable states can be entered with certain freedom. For example, suppose that the network placed in state (c, x_1, x_2, s) = (000*A*) receives $x_1 = 1$. Then the next state S is *B*. In this case, we may enter 0 or 1 as z for (110*A*) for the following reason. We have $z = 0$ for the initial stable state *A* during $c = 0$ and $z = 1$ for the destination stable state *B* during $c = 1$ and correspondingly $z = 0$ or 1. This does not make much difference as far as the external behavior of the network is concerned, because the network stays

in this unstable state (110*A*) for the short transient period and it simply means that $z = 1$ appears a little bit earlier or later. The network that results, however, may be different. Accordingly, $z = d$ (d denotes "don't-care") would be the best assignment, since this gives flexibility in designing the network later.

Design of Synchronous Sequential Networks in Skew Mode

Now let us design a synchronous sequential network based on a flow-output table in skew mode, using *J-K* master-slave flip-flops.

As pointed out previously, when master-slave flip-flops are used, the network does not have racing hazards even if internal variables make multiple changes, because the flip-flops do not respond to any input changes during clock pulses. Thus, we need not worry about hazards due to multiple changes of internal variables and consequently appropriate assignment of binary numbers to states in forming a transition-output table from a state-output table or a flow-output table in the design steps in Fig. 28.4. But the number of gates, connections, or levels in a network to be designed can differ, depending on how binary numbers are assigned to states (it is of secondary importance compared with the hazard problem, which makes networks useless if they malfunction). Making state assignments without considering multiple changes of internal variables is much easier than having to take these changes into account.

Let us derive the transition-output table shown in Table 28.13 from Table 28.11, using a state assignment as shown.

TABLE 28.13 Transition-Output Table in Skew Mode

			$c = 1$			
			x_1, x_2			
s	y_1y_2	$c = 0$	00	01	11	10
A	00	00, 0	00, 0	11, 0	*dd, d*	01, 0
B	01	01, 1	01,1	11, 0	*dd, d*	01, 1
C	11	11, 0	11, 0	10, 1	*dd, d*	00, 0
D	10	10, 1	10, 1	10, 1	*dd, d*	00, 0
				Y_1Y_2, z		

Reversing the inputs and outputs relationship of *J-K* master-slave flip-flops shown in Table 28.14(a), we have the output-input relationship shown in Table 28.14(b) (other master-slave flip-flop types can be treated in a similar manner). Table 28.14(b) shows what values inputs J and K must take for each change of internal variable y to its next value Y. (In order to have $y = Y = 0$, $J = K = 0$ or $J = 0$ and $K = 1$ must hold, as we can see in Table 28.14(a) and thus we have $S = 0$ and $R = d$ in Table 28.14(b).) Using Table 28.14(b), we form the excitation table in Table 28.15. Decomposing Table 28.15 into five Karnaugh maps for J_1, K_1, J_2, K_2 and z, we can find a minimal sum for each of J_1, K_1, J_2, K_2, and z. On the basis of these logic expressions, we design the loopless network inside the general model of sequential networks in Fig. 28.3. Then, placing two *J-K* master-slave flip-flops outside this loopless network, we have designed the sequential network shown in Fig. 28.11.

Master-slave flip-flops do not respond to changes in their inputs when and after their outputs change until the leading edges of next clock pulses. Thus, no network malfunction due to races occurs, and no post-analysis of whether or not the designed networks malfunction due to this is necessary. This is the advantage of clocked networks with raceless flip-flops.

Design of Asynchronous Sequential Networks in Fundamental Mode

Now let us design an asynchronous sequential network based on a flow-output table in fundamental mode.

According to the design steps of Fig. 28.4, we have to derive a transition-output table from the flow-output table shown in Table 28.12, by deriving a state-output table by assigning appropriate binary

TABLE 28.14 Input-Output Relationship and Output-Input Relationships of *J-K* Master-Slave Flip-Flop

(a) Input-output relationship

Inputs		Outputs	
J	*K*	*y*	*Y*
0	0	0	0
		1	1
0	1	0	0
		1	0
1	0	0	1
		1	1
1	1	0	1
		1	0

(b) Output-input relationship

Outputs		Inputs	
y	*Y*	*S*	*R*
0	0	0	*d*
0	1	1	*d*
1	0	*d*	1
1	1	*d*	0

TABLE 28.15 Excitation-Output Derived from Table 28.13

		$c=1$			
		x_1, x_2			
y_1y_2	$c=0$	00	01	11	10
00	0*d*, 0*d*, 0	0*d*, 0*d*, 0	1*d*, 1*d*, 0	*dd*, *d*, 0	0*d*, 1*d*, 0
01	0*d*, *d*0, 0	0*d*, *d*0, 0	1*d* *d*0, 0	*dd*, *d*, 0	0*d*, *d*0, 0
11	*d*0, *d*0, 0	*d*0, *d*0, 0	*d*0, *d*1, 0	*dd*, *d*, 0	*d*1, *d*1, 0
10	*d*0, 0*d*, 0	*d*0, 0*d*, 0	*d*0, 0*d*, 0	*dd*, *d*, 0	*d*1, 0*d*, 0
		J_1K_1, J_2K_2, z			

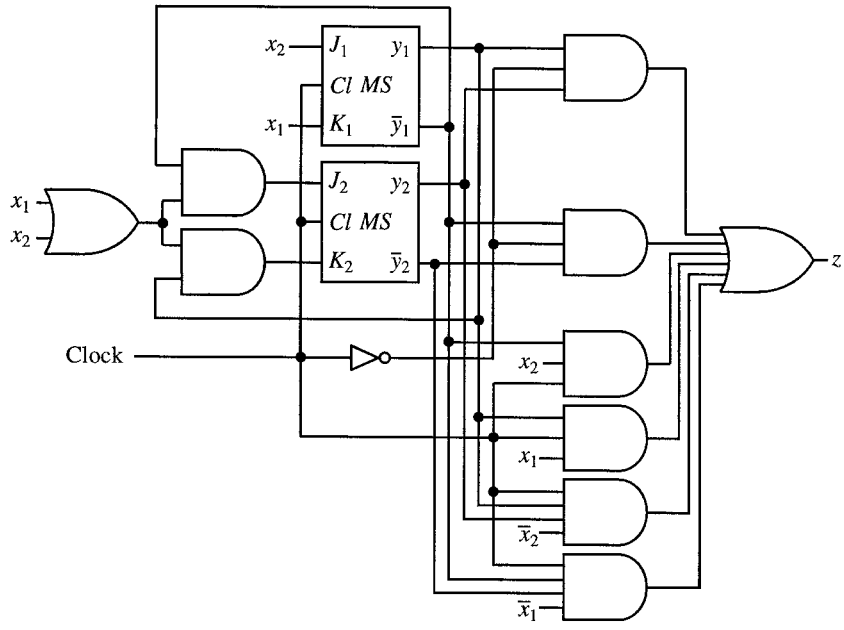


FIGURE 28.11 Synthesized network based on Table 28.15.

numbers to states such that multiple changes do not occur for every transition from one binary number to another, possibly changing some intermediate unstable states to others. Then, if the designers want to use *S-R* latches, we can derive an excitation table by finding the output-input relationship of the *S-R* latch, as illustrated with Tables 28.7, 28.8, and 28.9. Then, we can design the loopless network inside the general model of Fig. 28.3 by deriving minimal sums from the Karnaugh maps decomposed from the excitation-output table. If the designers do not want to use *S-R* latches, we can design the loopless network inside the general model of Fig. 28.3 by deriving minimal sums from the Karnaugh maps decomposed from the transition-output table without deriving an excitation-output table.

In the case of an asynchronous sequential network, we need post-analysis of whether the designed network works reliably.

Advantages of Skew Mode

The advantages of skew-mode operation with raceless flip-flops, such as master-slave or edge-triggered flip-flops, can be summarized as follows:

1. We can use no more complex and often simpler flow-output tables (or state-output tables) in skew mode than are required in fundamental mode, making design easier (because we need not consider both unstable and stable states for each input change, and need not consider adding extra states, or changing intermediate unstable states, which are to avoid multiple changes of internal variables).
2. State assignments are greatly simplified because we need not worry about hazard due to multiple changes of internal variables. (If we want to minimize the number of gates, connections, or levels, we need to try different state assignments. This is less important than the reliable operations of the networks to be synthesized.)
3. Networks synthesized in skew mode usually require fewer internal variables than those in fundamental mode.
4. After the network synthesis, we do not need to check whether the networks contain racing hazards or not. This is probably the greatest of all the advantages of skew mode, since checking hazards and finding remedies is usually very cumbersome and time-consuming.

References

1. Kohavi, Z., *Switching and Automata Theory*, 2nd ed., McGraw-Hill, 1978.
2. McCluskey, E. J., *Logic Design Principles: With Emphasis on Testable Semicustom Circuits*, Prentice-Hall, 1986.
3. Miller, R., *Switching Theory*, vol. 2, John Wiley & Sons, 1965.
4. Muroga, S., *Logic Design and Switching Theory*, John Wiley & Sons (now available from Krieger Publishing Co.), 1979.
5. Roth, C. H. Jr., *Fundamentals of Logic Design*, 4th ed., West Publishing Co., 1992.
6. Unger, S. H., *The Essence of Logic Circuits*, 2nd ed., IEEE Press, 1997.

Nakamura, Y., Muroga, S.
"Logic Synthesis with AND and OR Gates in Multi-levels"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

29

Logic Synthesis with AND and OR Gates in Multi-levels

Yuichi Nakamura
NEC Corporation

Saburo Muroga
University of Illinois at Urbana-Champaign

29.1 Logic Networks with AND and OR Gates in Multi-levels

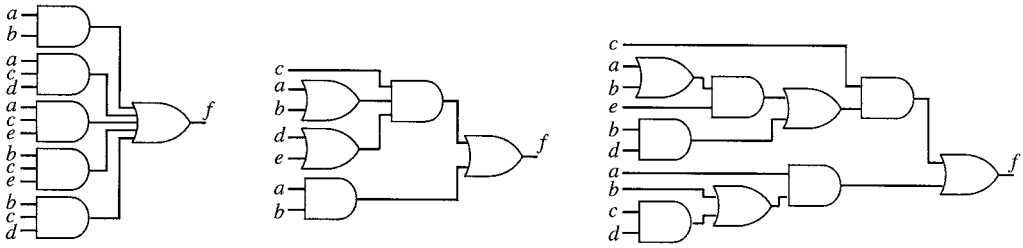
In logic networks, the number of levels is defined as the number of gates in the longest path from external inputs to external outputs. When we design logic networks with AND and OR gates, those in multi-levels can be designed with no more gates than those in two levels. Logic networks in multi-levels have more levels than those in two levels, but this does not necessarily mean that those in multi-levels have greater delay time than those in two levels because a logic gate that has many fan-out connections generally has greater delay time than gates that have fewer fan-out connections (Remark 29.1). Also, a logic gate that has many fan-in connections from other logic gates tends to have larger area in the chip and longer delay time than other gates that have fewer fan-in connections. Thus, if we want to design a logic network with a small delay time and small area, we need to design a logic network in many levels, keeping maximum fan-out and fan-in of each gate under a reasonably small limit.

Remark 29.1: When the line width in an IC chip is large, the delay time of logic gates is larger than those over connections and, once a logic network is designed, it can be laid out on the chip usually without further modifications. But when the line width becomes very short, under 0.25 μm, long connections add more delay due to parasitic capacitance and resistance than the delay of gates. But length of connections cannot be known until making layout on an IC chip after finishing logic design. So at the time of logic design, prior to layout, designers know only the number of fan-out connections from each gate, and this is only partial information on delay estimation. Thus, when the line width becomes very short, it is difficult to estimate precisely the delay of a logic network at the time of logic design. We need to modify a logic network interactively, as we lay it out on the chip.

Such a multi-level logic network can be derived by rewriting a logic expression with parentheses. For example, a logic expression

$$f = ab \vee acd \vee ace \vee bce \vee bcd \tag{29.1}$$

can be realized with five AND gates and one OR gate in two levels, as illustrated in Fig. 29.1(a). However,



(a) Two-level network (b) Network for divisor $a \vee b$ (c) Network for divisor $b \vee cd$

FIGURE 29.1 Networks for $f = ab \vee acd \vee ace \vee bce \vee bcd$.

$$f = c(a \vee b)(d \vee e) \vee ab \tag{29.2}$$

can be obtained by rewriting the expression with parentheses, as explained in the following. This logic expression can be realized with three OR gates and two AND gates in three levels, as illustrated in Fig. 29.1(b). The network in Fig. 29.1(b) would have a smaller area and smaller delay than the one in Fig. 29.1(a) because of fewer logic gates and a smaller maximum fan-in (a logic gate with five fan-ins, for example, has more than twice the area and delay of a gate with two fan-ins).

A logic network in two levels with the fewest gates can be derived by minimal sums or minimal products, which can be derived by reasonably simple algorithms (described in Chapter 27). But if we try to derive multi-level logic networks, only few reasonable algorithms are known. One of them is the weak division described in the following, although the minimality is not guaranteed and its execution is not straightforward. Another algorithm is the special case (i.e., AND and OR gates) of the map-factoring method (described in Chapter 31).

29.2 General Division

Rewriting of a logic expression using parentheses can be done by the following division. The division is based on the use of sub-expressions that can be found in the given logic expression. The given logic expression in a sum-of-products can be rewritten with parentheses if it has common sub-expressions. For example, the logic expression in Eq. 29.1 can be converted to the following expression, using a sub-expression $(a \vee b)$:

$$f = cd(a \vee b) \vee ce(a \vee b) \vee ab$$

This can be further rewritten into the following, by sharing the common sub-expression $(a \vee b)$:

$$f = c(a \vee b)(d \vee e) \vee ab \tag{29.3}$$

This rewriting can be regarded symbolically as division. Rewriting of the expression in Eq. 29.1 into the one in Eq. 29.3 may be regarded as division with the divisor $x = a \vee b$, the quotient $q = cd \vee ce$ and the remainder $r = ab$. Then, the expression f can be represented as follows:

$$f = xq \vee ab, \text{ with divisor } x = a \vee b, \text{ quotient } q = cd \vee ce = c(d \vee e), \text{ and remainder } r = ab.$$

The division is symbolically denoted as f/x. Generally, the quotient should not be 0, but the remainder may be 0.

The division, however, may yield many different results because there are many possibilities in choosing a divisor and also the given logic function can be written in many different logic expressions, as explained in the following.

Division can be repeated on one given logic expression. Suppose $f = \bar{a}b \vee \bar{a}c \vee \bar{b}a \vee \bar{b}c \vee \bar{c}a \vee \bar{c}b$ is given. Repeating division three times, choosing successively $b \vee c$, $a \vee c$, and $a \vee b$ as divisors, the following result is derived:

$$f = x_1\bar{a} \vee x_2\bar{b} \vee x_3\bar{c} \text{ with divisors } x_1 = \text{b} \vee c,\ x_2 = a \vee c, \text{ and } x_3 = a \vee b.$$

29.3 Selection of Divisors

Among the divisions, those with a certain type of divisor to be described in the following are called **weak divisions**. The objective of the weak division is the derivation of a logic network with a logic expression having a minimal total number of literals, repeatedly applying the weak division to the given logic expression until the division cannot apply to the logic expression any further. In this case, the total number of literals is intended to be an approximation of the total number of inputs to all the logic gates, although not exactly, as explained later. Thus, we should start with a logic expression in a minimal sum of products by using two-level logic minimization[1,2] before weak division in order to obtain a good result.

In the weak division, the divisor selection is the most important problem to produce a compact network because there are many divisor candidates to be found in the given logic expression and the result of the weak division depends on divisor selection. For example, the expression $f = ab \vee acd \vee ace \vee bce \vee bcd$ in Eq. 29.1 with 14 literals has many divisor candidates, a, b, c, $a \vee b$, $b \vee cd$, and others. When $a \vee b$ is first selected as the divisor, the resultant network illustrated in Fig. 29.1(b) for $f = c(a \vee b)(d \vee e) \vee ab$ with 7 literals is obtained. On the other hand, if $b \vee cd$ is first selected as the divisor, the resultant network for $f = c(e(a \vee b) \vee bd) \vee a(b \vee cd)$ with 10 literals illustrated in Fig. 29.1(c) is obtained, which is larger than the network illustrated in Fig. 29.1(b).

Divisors can be derived by finding sub-expressions called kernels. All the kernels for the given logic expression can be found as follows.

First, all the subsets of products in the given logic expression are enumerated. Next, for each subset of products, the product of the largest number of literals that is common with all the products in this subset is found. This product of the largest number of literals is called a **co-kernel**. Then the sum of products, from each of which this co-kernel (i.e., the product of the largest number of literals) is eliminated is obtained as a **kernel**. For example, the sum of products $abc \vee abd$ has the co-kernel ab as the product of the largest number of literals that is common to all the products, abc and abd. The kernel of $abc \vee abd$ is $c \vee d$. However, $ab \vee ac \vee d$ has no kernels, because it has no common literals for all products. The kernel $b \vee c$ with co-kernel a is found when the subset of products, $ab \vee ac$, is considered.

Certainly, by trying all divisor candidates and selecting the best one, we can derive a network as small as possible by the weak division. However, such an exhaustive search is too time-consuming, requiring a huge memory space. The branch-and-bound method is generally more efficient than the exhaustive search.[3]

Thus, the heuristic method that the type of divisor candidates is restricted to sum of products with specific feature is proposed.[2] This method is called **kernel decomposition,** reducing the number of divisor candidates.

A kernel of an expression for f is the sum with at least two products such that all the products in the sum contain no common literals (e.g., $ab \vee c$ is a kernel, but $ab \vee ac$ and abc are not kernels, because all products, ab and ac, in $ab \vee ac$ contain a, and abc is a single product), especially, a kernel whose subsets contain no other kernels is called a **level-0 kernel** (e.g., $a \vee b$ is a level-0 kernel, but $ab \vee ac \vee d$ is not a level-0 kernel because sub-expression $ab \vee ac$ contains kernel $b \vee c$). The level of kernels is defined recursively as a **level-K kernel** contains at the next lower level-$(K-1)$ kernel (e.g., $ab \vee ac \vee d$ is a level-1 kernel because it contains level-0 kernel $b \vee c$).

Usually, a level-K kernel with $K \geq 1$ is not used as a divisor to save processing time, because the results obtained by all level kernels as divisors are the almost same as those obtained by using only the level-0 kernels. Thus, all the kernels that are not level-0 kernels are excluded from divisor candidates.

For example, the logic expression illustrated in Fig. 29.1, $f = ab \vee acd \vee ace \vee bce \vee bcd$, has 16 kernels as shown in Table 29.1. By eliminating all the level-1 kernels, $ad \vee ae \vee bd \vee be$, $ea \vee eb \vee bd$ and others from Table 29.1, we have the divisor candidates in Table 29.2.

TABLE 29.1 Kernals for $f = ab \vee acd \vee ace \vee bce \vee bcd$

Kernel	Co-kernel	Level
$ad \vee ae \vee bd \vee be$	c	1
$ea \vee eb \vee bd$	c	1
$eb \vee ed \vee cd$	c	1
$ab \vee ae \vee bd$	c	1
$ad \vee ae \vee be$	c	1
$a \vee ce \vee cd$	b	1
$b \vee cd$	a	0
$b \vee ce$	a	0
$a \vee be$	b	0
$a \vee cd$	b	0
$d \vee e$	ac	0
$ad \vee be$	c	0
$a \vee b$	cd	0
$ae \vee bd$	c	0
$a \vee b$	ce	0
$d \vee e$	bc	0

TABLE 29.2 0-level Kernals and Co-kernels Derived from Table 29.1

Kernel	Co-kernel	Weight of Kernels
$b \vee cd$	a	1
$b \vee ce$	a	1
$a \vee be$	b	1
$a \vee cd$	b	1
$ad \vee be$	c	1
$ae \vee bd$	c	1
$a \vee b$	ce, cd	6
$d \vee e$	ac, bc	6

The next step is the selection of one divisor from all candidates. A candidate that decreases the largest number of literals in the expression by the weak division is selected as a divisor. If there is a tie, choose one of them. The difference in the number of literals before and after the weak division by the kernel is called the **weight of the kernel**. In the above example, the result of the weak division by the kernel $b \vee cd$ is $f = a(b \vee cd) \vee ace \vee bce \vee bcd$. The weight of the kernel $b \vee cd$ is 1, because the number of literals is reduced from 14 to 13 by this division. The weight of the kernels can be easily calculated by the number of literals in kernels and co-kernels without execution of weak division, because the quotient of the division by a kernel is a product of a co-kernel and other sub-expressions. The weight of the kernels for this example is shown in Table 29.2.

Then the kernel $a \vee b$ is selected as a divisor with the largest weight. The expression $f = ab \vee acd \vee ace \vee bce \vee bcd$ is divided by $a \vee b$. In the next division, $d \vee e$ is selected and divides the expression after division by $a \vee b$. Finally, the network $f = c(a \vee b)(d \vee e) \vee ab$ illustrated in Fig. 29.1(b) is obtained.

These operations, enumeration of all the kernels, calculation of the weights of all kernels, selection of the largest one, and division are applied repeatedly until no kernel can be found. In this case, we can choose a different sequence of divisors, deriving a different result. But often we do not have a different result, so it may not be worthwhile to try many different sequences of divisors.

Instead of the kernel decomposition method, a faster method is proposed.[4] In this method, the divisor candidates are restricted to only 0-level kernels with two products, along with introduction of complement-sharing that when both $\bar{a}b \vee a\bar{b}$ and $\bar{a}\bar{b} \vee ab$ are among divisor candidates, one is realized with a logic gate while realizing the other by complementing it by an inverter ($\overline{\bar{a}b \vee a\bar{b}} = (a \vee \bar{b})(\bar{a} \vee b) = \bar{a}\bar{b} \vee ab$ for this example). Although the restriction is stronger than the kernel decomposition method, the method produces smaller networks and can run faster than the kernel decomposition in many cases.

29.4 Limitation of Weak Division

Although simpler networks can be easily derived by the weak division, the weak division cannot derive certain types of logic networks because of its restrictions in its rewriting of logic expressions with parentheses. Rewriting of logic expressions without such restrictions is called **strong division**. For example, complements of sub-expressions, such as $f = \overline{(a\bar{b} \vee a\bar{c})} \vee b\bar{a} \vee c$, is not used in the weak division. Also, the two literals, x and $\bar{x}$, for each variable x are regarded as different variables without using identities such as $ab \vee a = a$, $a\bar{a} = 0$, and $a \vee \bar{a} = 1$ in the weak division.

Suppose the sum-of-products $f = a\bar{b} \vee a\bar{c} \vee b\bar{a} \vee b\bar{c} \vee c\bar{a} \vee c\bar{b}$ is given. This can be rewritten in the following two different logic expressions:

$$f_1 = a\bar{b} \vee a\bar{c} \vee b\bar{a} \vee b\bar{c} \vee c\bar{a} \vee c\bar{b}$$

and

$$f_2 = a\bar{a} \vee a\bar{b} \vee a\bar{c} \vee b\bar{a} \vee b\bar{b} \vee b\bar{c} \vee c\bar{a} \vee c\bar{b} \vee c\bar{c},$$

using the identities $a\bar{a} = 0$, $b\bar{b} = 0$, and $c\bar{c} = 0$

They can be further rewritten as follows:

$$f_1 = a(\bar{b} \vee \bar{c}) \vee b(\bar{a} \vee \bar{c}) \vee c(\bar{a} \vee \bar{c})$$

and

$$f_2 = (\bar{a} \vee \bar{b} \vee \bar{c})(a \vee b \vee c)$$

Both of these expressions can be written in the following expressions, using the divisors, quotients, and remainders, which are 0 in this particular example:

$$f_1 = x_{11}q_{11} \vee x_{12}q_{12} \vee x_{13}q_{13} \vee r_1$$

with divisors $x_{11} = \bar{b} \vee \bar{c}$, $x_{12} = \bar{a} \vee \bar{c}$, $x_{13} = \bar{a} \vee \bar{b}$, quotients $q_{11} = a$, $q_{12} = b$, $q_{13} = c$, and remainder $r_1 = 0$

$$f_2 = x_{21}q_{21} \vee r_2$$

with divisor $x_{21} = a \vee b \vee c$, quotient $q_{21} = \bar{a} \vee \bar{b} \vee \bar{c}$, and remainder $r_2 = 0$.

Corresponding to these expressions, we have two different logic networks, as shown in Fig. 29.2. The function f_2 is derived by division by a divisor above but actually cannot be obtained by the weak division. Thus, the logic network for f_2 is labeled as the result by strong division in Fig. 29.2. Strong division is rewriting of logic expressions using any form, including the complement of a sub-expression and accordingly has far greater possibilities than the division explained so far.

The results of division are evaluated by the number of literals contained in each of the obtained expressions. This number is an approximation of the total number of fan-ins of gates in networks in the

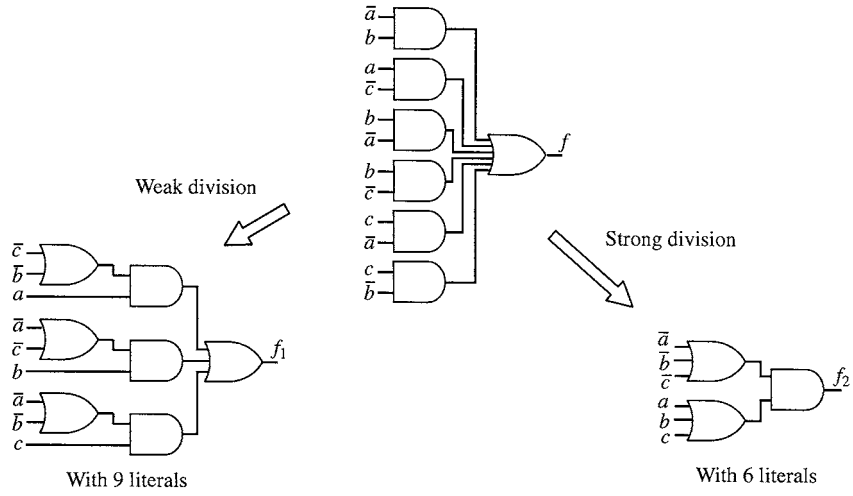


FIGURE 29.2 Strong and weak divisions.

following sense: inputs to a logic gate from other gates are not counted as literals. In Fig. 29.2, the number of literals of f_1 is 9, and the number of literals of f_2 is 6. But in the logic network for f_1, an input to each of three AND gates in Fig. 29.2, for example, is not counted as a literal. Counting them, the total number of fan-ins of all logic gates in the logic network for f_1, which is 15 in Fig. 29.2, is larger than the total number of fan-ins of all gates in the logic network for f_2, 8.

References

1. R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis,* Kluwer Academic Publishers, Boston, 1984.
2. R. K. Brayton, A. Sangiovanni-Vincentelli, and A. Wang, "MIS: A Multiple-Level Logic Optimization System," *IEEE Transaction on CAD*, CAD-6(6), pp. 1062-1081, July 1987.
3. G. De Micheli, A. Sangiovanni-Vincentelli, and P. Antognetti, *Design System for VLSI Circuits: Logic Synthesis and Silicon Compilation*, Martinus Nijhoff Publishers, pp. 197-248, 1987.
4. J. Rajski, and J. Vasudevamurthy, "Testability Preserving Transformations in Multi-level Logic Synthesis," *IEEE ITC,* pp. 265-273, 1990.

Muroga, S. "Logic Properties of Transistor Circuits"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

30

Logic Properties of Transistor Circuits

Saburo Muroga
University of Illinois at Urbana-Champaign

30.1 Basic Properties of Connecting Relays

Relays are probably the oldest means to realize logic operations. Relays, which are electromechanical devices, and their solid-state equivalents (i.e., transistors) are extensively used in many industrial products, such as computers. Relays are conceptually simple and appropriate for introducing physical realization of logic operations. More importantly, the connection configuration of a relay contact network is the same as that of transistors inside a logic gate realized with transistors, in particular MOSFETs (which stands for metal-oxide semiconductor field effect transistors).

A **relay** consists of an armature, a magnet, and a metal contact. An armature is a metal spring made of magnetic material with a metal contact on it. There are two different types of relays: a make-contact relay and a break-contact relay.

A **make-contact relay** is a relay such that, when there is no current through the magnet winding, the contact is open. When a direct current is supplied through the magnet winding, the armature is attracted to the magnet and, after a short time delay, the contact is closed. This type of relay contact is called a "make-contact" and is usually denoted by a lower-case x. The current through the magnet is denoted by a capital letter X, as shown in Fig. 30.1.

A **break-contact relay** is a relay such that when there is no current through the magnet winding, the contact closes. When a direct current is supplied, the armature is attracted to the magnet and, after a short time delay, the contact opens. This type of relay contact is called a "break-contact" and is usually denoted by $\bar{x}$. The current through the magnet is again denoted by X, as shown in Fig. 30.2.

In either case of a make-contact relay or a break-contact relay, no current in a magnet is represented by $X = 0$, and the flow of a current is represented by $X = 1$. Then $x = X$, no matter whether $X = 0$ or 1. But the contact of a make-contact relay is open or closed according as $X = 0$ or 1, because the contact is expressed by x, whereas the contact of a break-contact relay is closed or open according as $X = 0$ or 1, because the contact is expressed by $\bar{x}$.

Let us connect two make-contacts x and y in series, as shown in Fig. 30.3. Since X and x assume identical values at any time, the magnet, along with its symbol X, will henceforth be omitted in figures unless it is needed for some reason. Then we have the combinations of states shown in Table 30.1(a),

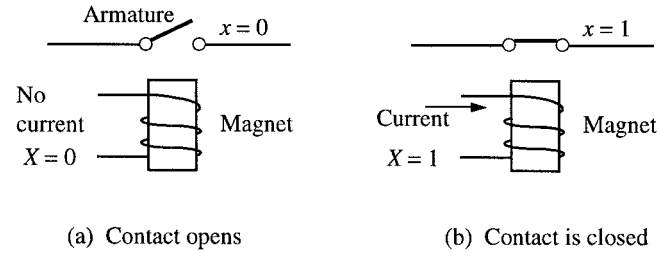


FIGURE 30.1 A make-contact relay.

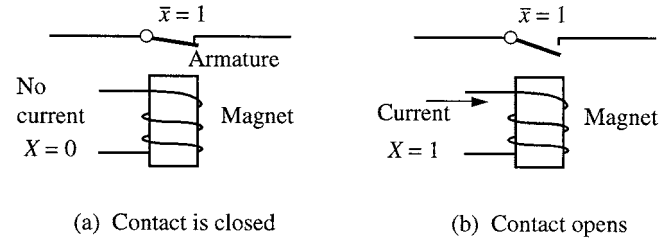


FIGURE 30.2 A break-contact relay.

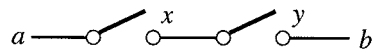


FIGURE 30.3 Series connection of relay contacts.

TABLE 30.1 Combinations of States for the Series Connection in Fig. 30.3

(a)			(b)		
x	y	Entire Path Between a and b	x	y	f
Open	Open	Open	0	0	0
Open	Closed	Open	0	1	0
Closed	Open	Open	1	0	0
Closed	Closed	Closed	1	1	1

which has only two states, "open" and "closed." Let f denote the state of the entire path between terminals a and b, where f is called the **transmission** of the network. Since "open" and "closed" of a make-contact are represented by $x = 0$ and $x = 1$, respectively, Table 30.1(a) may be rewritten as shown in Table 30.1(b). This table shows the AND of x and y, defined in Chapter 23 and denoted by $f = xy$. Thus the network of a series connection of make-contacts realizes the AND operation of x and y.

Let us connect two make-contacts x and y in parallel as shown in Fig. 30.4. Then we have the combinations of states shown in Table 30.2(a). Replacing "open" and "closed" by 0 and 1, respectively, we may rewrite Table 30.2(a) as shown in Table 30.2(b). This table shows the OR of x and y, defined in Chapter 24 and denoted by $f = x \vee y$.

30.2 Analysis of Relay-Contact Networks

Let us analyze a relay contact network. "Analysis of a network" means the description of the logic performance of the network in terms of a logic expression.[1]

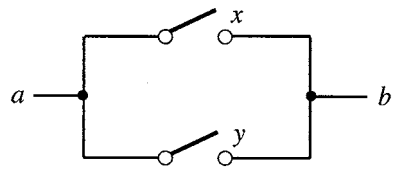


FIGURE 30.4 Parallel connection of relay contacts.

TABLE 30.2 Cominations of States for the Parallel Connection in Fig. 30.4

(a)			(b)		
x	y	Entire Path Between a and b	x	y	f
Open	Open	Open	0	0	0
Open	Closed	Closed	0	1	1
Closed	Open	Closed	1	0	1
Closed	Closed	Closed	1	1	1

Transmission of Relay-Contact Networks

We now discuss general procedures to calculate the transmission of a network in which relay contacts are connected in a more complex manner. The first general procedure is based on the concept of tie sets, defined as follows.

Definition 30.1: Consider a path that connects two external terminals, a and b, and no part of which forms a loop. Then the literals that represent the contacts on this path are called a **tie set** of this network.

An example of a tie set is the contacts x_1, x_6, $\bar{x}_2$, and x_5 on the path numbered 1 in Fig. 30.5.

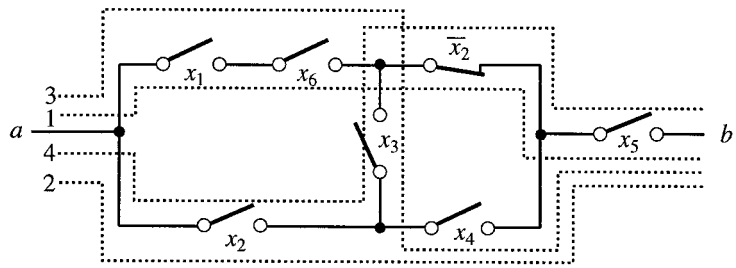


FIGURE 30.5 Tie sets of non-series-parallel network.

Procedure 30.1: Derivation of the Transmission of a Relay-Contact Network by Tie Sets

Find all the tie sets of the network. Form the product of all literals in each tie set. Then the disjunction of all these products yields the transmission of the given network. □

These tie sets represent all the shortest paths that connect terminals a and b. As an example, the network of Fig. 30.5 has the following tie sets:

For path 1: $x_1, x_6, \bar{x}_2, x_5$
For path 2: x_2, x_4, x_5
For Path 3: x_1, x_6, x_3, x_4, x_5
For path 4: $x_2, x_3, \bar{x}_2, x_5$

Then, we get the transmission of the network:

$$f = x_1x_6\bar{x}_2x_5 \vee x_2x_4x_5 \vee x_1x_6x_3x_4x_5 \vee x_2\bar{x}_2x_3x_5 \tag{30.1}$$

where the last term, $x_2\bar{x}_2x_3x_5$, may be eliminated, since it is identically equal to 0 for any value of x_2.

Procedure 30.1 yields the transmission of the given network because all the tie sets correspond to all the possibilities for making f equal 1. For example, the first term, $x_1x_6\bar{x}_2x_5$, in Eq. 30.1 becomes 1 for the combination of variables $x_1 = x_6 = x_5 = 1$ and $x_2 = 0$. Correspondingly, the two terminals a and b of the network in Fig. 30.5 are connected for this combination.

The second general procedure is given after the following definition.

Definition 30.2: Consider a set of contacts that satisfy the following conditions:

1. If all of the contacts in this set are opened simultaneously (ignoring functional relationship among contacts; in other words, even if two contacts, x and $\bar{x}$, are included in this set, it is assumed that contacts x and $\bar{x}$ can be opened simultaneously), the entire network is split into exactly two isolated subnetworks, one containing terminal a and the other containing b.

2. If any of the contacts are closed, the two subnetworks can be connected.

Then, the literals that represent these contacts are called a **cut set** of this network. □

As an example, let us find all the cut sets of the network in Fig. 30.5, which is reproduced in Fig. 30.6. First, let us open contacts x_1 and x_2 simultaneously, as shown in Fig. 30.6(a). Then terminals a and b are completely disconnected (thus condition 1 of Definition 30.2 is satisfied). If either of contacts x_1 and x_2 is closed, the two terminals a and b can be connected by closing the remaining contacts (thus, condition 2 of Definition 30.2 is satisfied). We have all the cut sets shown in the following list and also in Fig. 30.6(b):

For cut 1: x_1, x_2
For cut 2: x_6, x_2
For cut 3: x_1, x_3, x_4
For cut 4: x_6, x_3, x_4
For cut 5: $\bar{x}_2$, x_3, x_2
For cut 6: $\bar{x}_2$, x_4
For cut 7: x_5

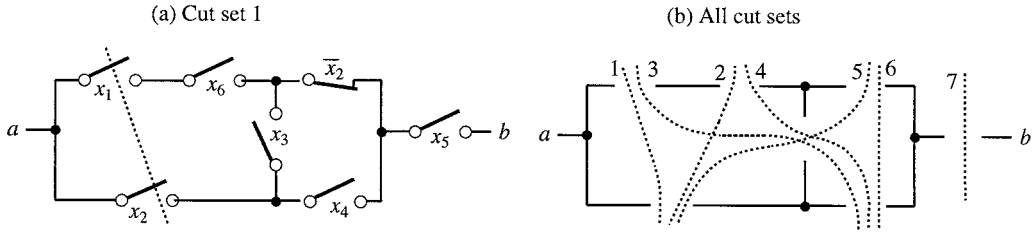


FIGURE 30.6 Cut sets of the network in Fig. 30.5.

Procedure 30.2: Derivation of the Transmission of Relay-Contact Network by Cut Sets

Find all the cut sets of a network. Form the disjunction of all literals in each cut set. Then the product of all these disjunctions yields the transmission of the given network. □

On the basis of the cut sets in the network of Fig. 30.5 derived above, we get

$$f = (x_1 \vee x_2)(x_6 \vee x_2)(x_1 \vee x_3 \vee x_4)(x_6 \vee x_3 \vee x_4)(\bar{x}_2 \vee x_3 \vee x_2)(\bar{x}_2 \vee x_4)(x_5) \tag{30.2}$$

This expression looks different from Eq. 30.1, but they are equivalent, since we can get identical truth tables for both expressions.

Procedure 30.2 yields the transmission of a relay contact network because all the cut sets correspond to all possible ways to disconnect two terminals, *a* and *b*, of a network; that is, all possibilities of making *f* equal 0. Any way to disconnect *a* and *b* which is not a cut set constitutes some cut set plus additional unnecessary open contacts, as can easily be seen.

The disjunction inside each pair of parentheses in Eq. 30.2 corresponds to a different cut set. A disjunction that contains the two different literals of any variable (e.g., $(\bar{x}_2 \vee x_3 \vee x_2)$ in Eq. 30.2 contains two literals, $\bar{x}_2$ and x_2, of the variable x_2) is identically equal to 1 and is insignificant in multiplying out *f*. Therefore, every cut set that contains the two literals of some variable need not be considered in Procedure 30.2.

30.3 Transistor Circuits

Bipolar transistor and MOSFET are currently the two most important types of transistors for integrated circuit chips, although MOSFET is becoming increasingly popular.[2] A transistor is made of pure silicon that contains a trace of impurities (i.e., n-type silicon or p-type silicon). When a larger amount of impurity than standard is added, we have n^+- and p^+-type silicon. When less, we have n^-- and p^--type silicon. A bipolar transistor has a structure of **n-type region** (or simply n-region) consisting of n-type silicon and **p-type region** (or simply p-region) consisting of p-type silicon, as illustrated in Fig. 30.7, different from that of MOSFET illustrated in Fig. 30.10.

Bipolar Transistors

An implementation example of an n-p-n bipolar transistor, which has three electrodes (i.e., an emitter, a base, and a collector) is shown in Fig. 30.7(a) along with its symbol in Fig. 30.7(b). A p-n-p transistor has the same structure except p-type regions and n-type regions exchanged (n^+- and p^+-type regions also exchanged).

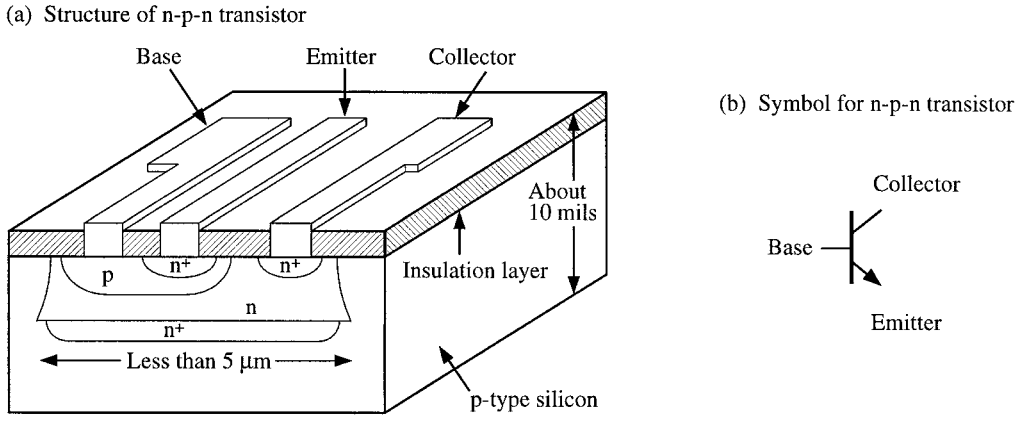


FIGURE 30.7 n-p-n transistor.

Suppose that an n-p-n transistor is connected to a power supply with a resistor and to the ground, as shown in Fig. 30.8(a). When the input voltage v_i increases, the collector current i_c gradually increases, as shown in Fig. 30.8(b). (Actually, i_c is 0 until v_i reaches about 0.6 V. Then i_c gradually increases and then starts to saturate.) As i_c increases, the output voltage v_o decreases from 5 V to 0.3 V or less because of the voltage difference across the resistor *R*, as shown in Fig. 30.8(c). Therefore, when the input v_i is a high

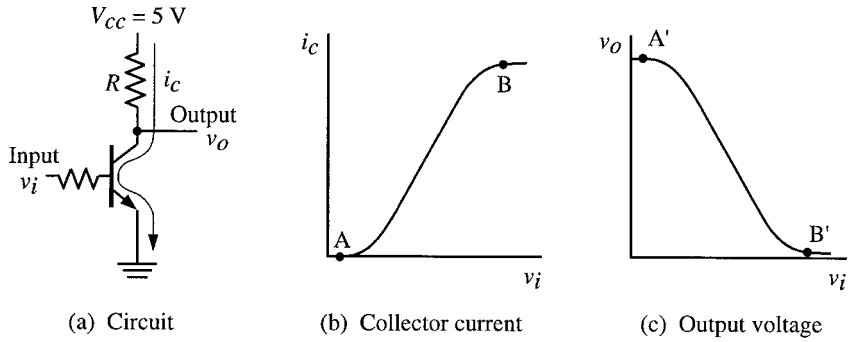


FIGURE 30.8 Inverter circuit.

voltage (about 5 V), the output v_o is a low voltage (about 0.3 V), and when v_i is a low voltage (about 0.3 V), v_o is a high voltage (about 5 V). This is illustrated in Table 30.3(a). Thus, if binary logic values 0 and 1 are represented by low and high voltages, respectively, we have the truth table in Table 30.3(b). This means that the circuit in Fig. 30.8(a) works as an **inverter**. In other words, if v_i represents a logic variable x, then v_o represents the logic function $\bar{x}$.

TABLE 30.3 Input-Output Relations of the Inverter in Fig. 30.8(a)

(a) Voltage Required

Input v_i	Output v_o
Low voltage	High voltage
High voltage	Low voltage

(b) Truth Table

v_i	v_o
0	1
1	0

Since we are concerned with binary logic values in designing logic networks, we will henceforth consider only the on-off states of currents or the corresponding voltages in electronic circuits (e.g., A and B in Fig. 30.8(b), or A′ and B′ in Fig. 30.8(c)), without considering their voltage magnitudes.

As we will see later, the transistor circuit in Fig. 30.8(a) is often used as part of more complex transistor circuits that constitute logic gates. Here, notice that if resistor R' is added between the emitter and the ground, and the output terminal v_o is connected to the emitter, instead of to the collector, as shown in Fig. 30.9, then the new circuit does not work as an inverter. In this case, when v_i is a high voltage, v_o is also a high voltage, because the current that flows through the transistor produces the voltage difference across resistor R'. When v_i is a low voltage, v_o is also a low voltage, because no current flows and

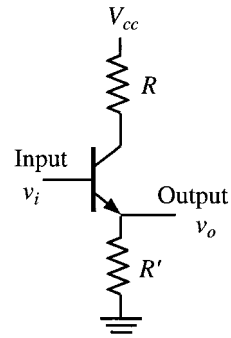


FIGURE 30.9 Emitter follower.

consequently no voltage difference develops across R'. So if v_i represents a variable x, v_o represents the logic function x, and no logic operation is performed. The transistor circuit in Fig. 30.9, which is often called an **emitter follower**, works as a current amplifier. This circuit is used often as part of other circuits to supply a large output current by connecting the collector of the transistor directly to the V_{cc} without R.

A logic gate based on bipolar transistors generally consists of many transistors which are connected in a more complex manner than Fig. 30.8 or 30.9, and realizes a more complex logic function than $\bar{x}$. (See Chapter 35 on ECL.)

MOSFET (Metal-Oxide Semiconductor Field Effect Transistor)

In integrated circuit chips, two types of MOSFETs are usually used; that is, **n-channel enhancement-mode MOSFET** (or abbreviated as n-channel enhancement MOS, or enhancement nMOS) and **p-channel enhancement-mode MOSFET** (or abbreviated as p-channel enhancement MOS, or enhancement pMOS). The structure of the former is illustrated in Fig. 30.10(a). They are expressed by the symbols shown in Fig. 30.10 (b) and (c), respectively. Each of them has three terminals: **gate**, **source,** and **drain**. In Fig. 30.10(a), the gate realized with metal is shown for the sake of simplicity, but a more complex structure, called **silicon-gate MOSFET**, is now far more widely used. **The "gate" in Fig. 30.10 should not be confused with "logic gates."** The thin area underneath the gate between the source and drain regions in Fig. 30.10(a) is called a **channel**, where a current flows whenever conductive.

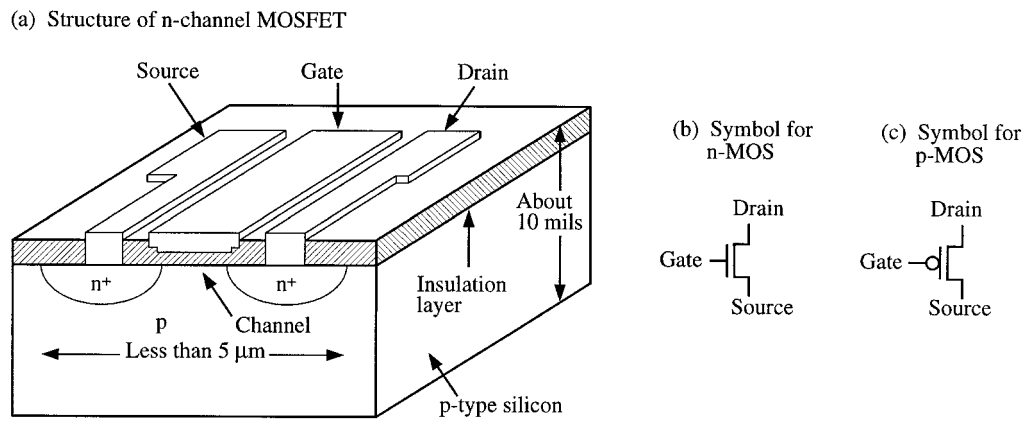


FIGURE 30.10 MOSFET.

Suppose that the source of an n-channel enhancement-mode MOSFET is grounded and the drain is connected to the power supply of 3.3 V through resistor R, as illustrated in Fig. 30.11(a). When the input voltage v_i increases from 0 V, the current i, which flows from the power supply to the ground through R and the MOSFET, increases as shown in Fig. 30.11(b), but for v_i smaller than the threshold voltage V_T, essentially no current flows. Then because of the voltage drop across R, the output voltage v_o, decreases, as shown in Fig. 30.11(c). Since we use binary logic, we need to use only two different voltage values, say 0.2 and 3.3 V. If v_i is 0.2 V, no current flows from the power supply to the ground through the MOSFET and v_o is 3.3 V. If v_i is 3.3 V, the MOSFET becomes conductive and a current flows. V_o is 0.2 V because of the voltage drop across R. Thus, if v_i is a low voltage (0.2 V), v_o is a high voltage (3.3 V); and if v_i is a high voltage, v_o is a low voltage, as shown in Table 30.4(a). If low and high voltages represent logic values 0 and 1, respectively, in other words, if we use **positive logic**, Table 30.4(a) is converted to the truth table in Table 30.4(b). (When low and high voltages represent 1 and 0, respectively, this is said to be in **negative logic**.) Thus, if v_i represents logic variable x, output v_o represents function $\bar{x}$. This means that the electronic circuit in Fig. 30.11(a) works as an **inverter**.

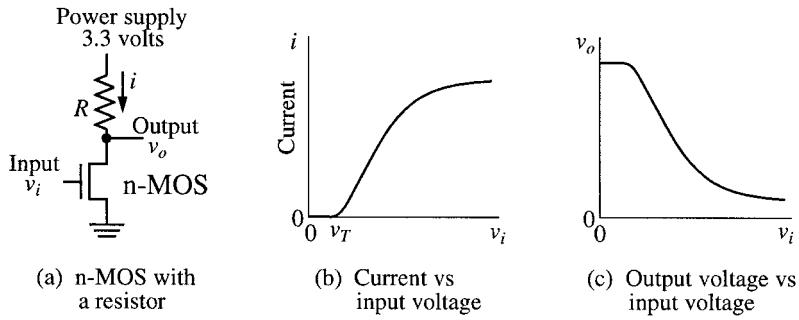


FIGURE 30.11 Inverter.

TABLE 30.4 Truth Table for the Circuit in Fig. 30.11

(a) Voltage Required

Input v_i	Output v_o
Low voltage	High voltage
High voltage	Low voltage

(b) Truth Table

Input x	Output f
0	1
1	0

Resistor R shown in Fig. 30.11 occupies a large area, so it is usually replaced by a MOSFET, called an n-channel depletion-mode MOSFET, as illustrated in Fig. 30.12, where the depletion-mode MOSFET is denoted by the MOSFET symbol with double lines. Notice that **the gate of this depletion-mode MOSFET is connected to the output terminal instead of the power supply**, the logic gate with depletion-mode MOSFET replacing the resistor work in the same manner as before. Logic gates with depletion-mode MOSFETs work faster and are more immune to noise.

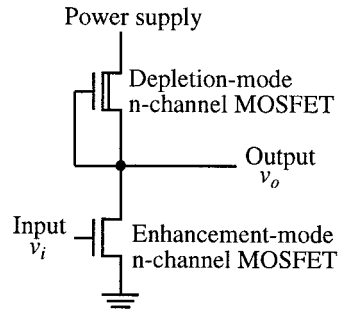


FIGURE 30.12 A logic gate with depletion-mode MOSFET.

The n-channel **depletion-mode MOSFET** is different from the n-channel enhancement-mode MOSFET in having a thin n-type silicon layer embedded underneath the gate, as illustrated in Fig. 30.13. When a positive voltage is applied at the drain against the source, a current flows through this thin n-type silicon layer even if the voltage at the gate is 0 V (against the source). As the gate voltage becomes more positive, a greater current flows. Or, as the gate voltage becomes more negative, a smaller current flows. If the gate voltage decreases beyond threshold voltage V_T, no current flows. This relationship, called a **transfer curve**, between the gate voltage V_{GS} (against the source) and the current i is shown in Fig. 30.14(a), as compared with that for the n-channel enhancement-mode MOSFET shown in Fig. 30.14(b).

By connecting many n-channel enhancement MOSFETs, we can realize any **negative function**, i.e., the complement of a sum-of-products where only non-complemented literals are used ($\overline{x \vee yz}$ is an example of negative function). For example, if we connect three MOSFETs in series, including the one

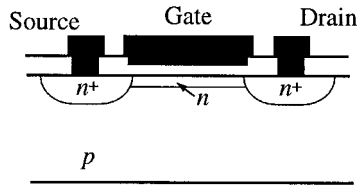


FIGURE 30.13 *n*-Channel depletion-mode MOSFET.

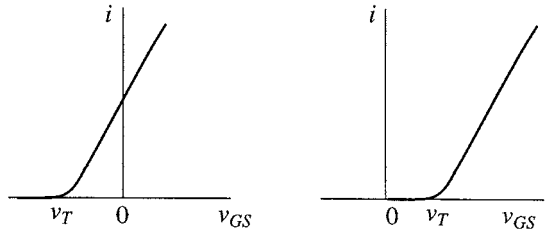


FIGURE 30.14 Shift of threshold voltage V_T (V_{GS} is a voltabe between gate and source).

for resistor replacement, as shown in Fig. 30.15, the output f realizes the NAND function of variables x and y. Only when both inputs x and y have high voltages, two MOSFETs for x and y become conductive and a current flows through them. Then the output voltage is low. Otherwise, at least one of them is non-conductive and no current flows. Then the output voltage is high. This relationship is shown in Table 30.5(a). In positive logic, this is converted to the truth table 30.5(b), concluding that the circuit represents $\overline{xy}$ which is called the NAND function. Figure 30.16 shows a logic circuit for $\overline{x \vee y}$ (called the NOR function) by connecting MOSFETs in parallel. A more complex example is shown in Fig. 30.17.

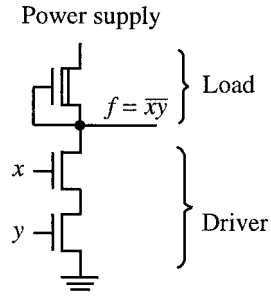


FIGURE 30.15 Logic gate for the NAND function.

TABLE 30.5 Truth Table for the Circuit in Fig. 30.15

(a) Voltage Relation

x	y	f
Low	Low	High
Low	High	High
High	Low	High
High	High	Low

(b) Truth Table

x	y	f
0	0	1
0	1	1
1	0	1
1	1	0

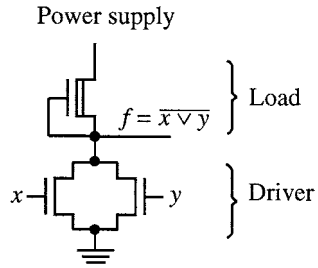


FIGURE 30.16 Logic gate for the NOR function.

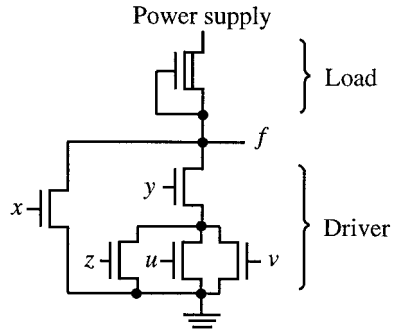


FIGURE 30.17 A logic gate with many MOSFETs.

The MOSFET that is connected between the power supply and the output terminal is called a **load** or **load MOSFET** in each of Figs. 30.15 through 30.17. Other MOSFETs that are directly involved in logic operations are called a **driver** or **driver MOSFETs** in each of these circuits.

Procedure 8.3: Calculation of the Logic Function of a MOS Logic Gate

The logic function f for the output of each of these MOS logic gates can be obtained as follows.

1. Calculate the transmission of the driver, regarding each n-channel MOSFET as a make-contact of relay, as illustrated in Fig. 30.18. When $x = 1$, a current flows through the magnet in the make-contact relay in Fig. 30.18 and the contact x is closed and becomes conductive, whereas n-MOS becomes conductive when $x = 1$, i.e., input x of nMOS is a high voltage.
2. Complement it. □

For example, the transmission of the driver in Fig. 30.16 is $x \vee y$. Then by complementing it, we have the output function $f = \overline{x \vee y}$. The output function of a more complex logic gate, such as Fig. 30.18, can be calculated in the same manner. Thus, **a MOS circuit expresses a negative function with respect to input variables connected to driver MOSFETs**.

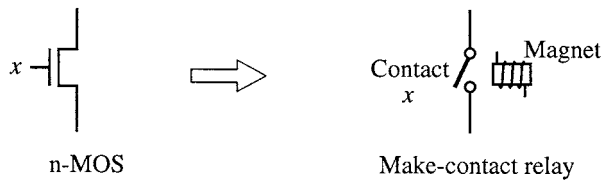


FIGURE 30.18 Analogy between n-MOS and a make-contact relay.

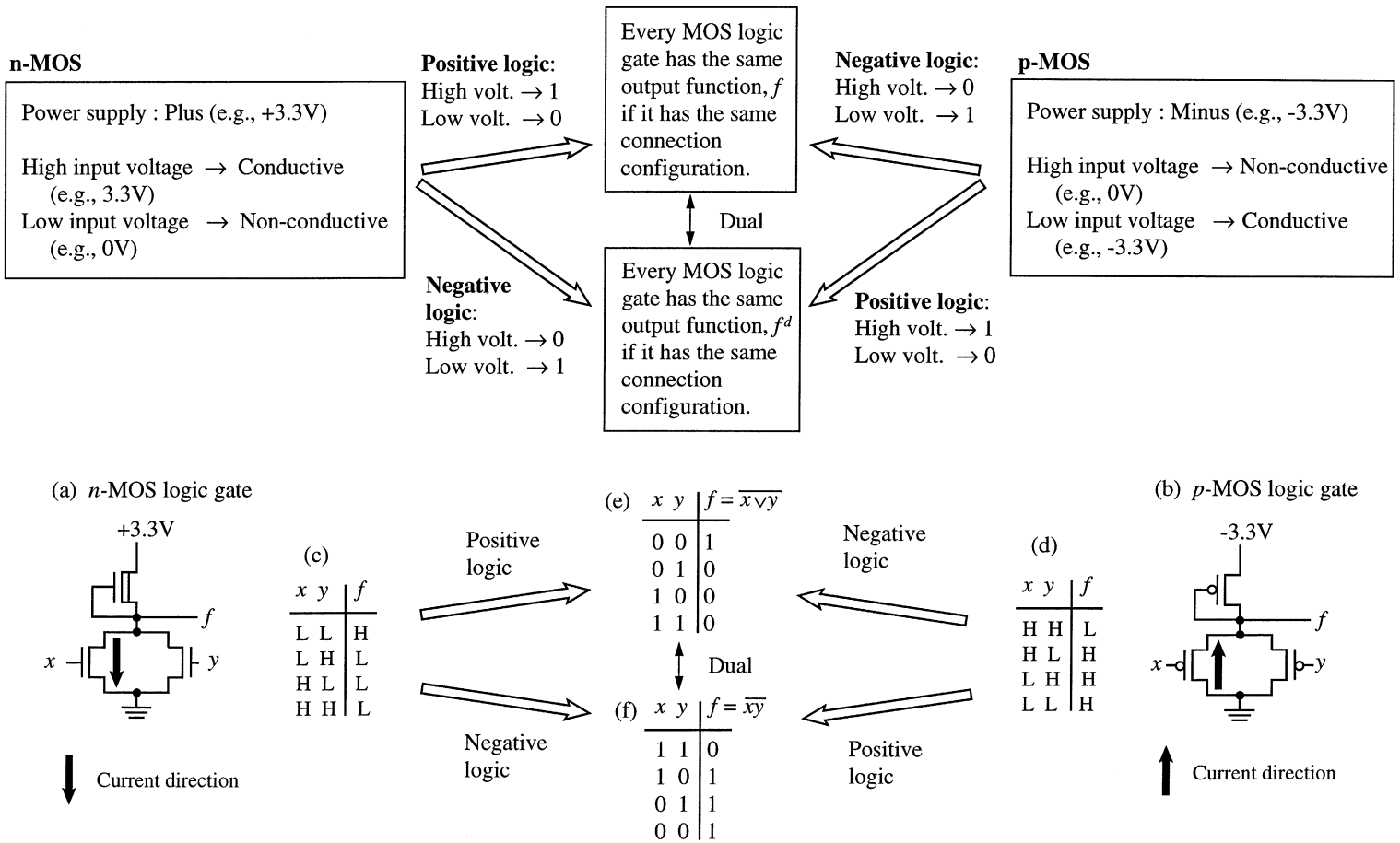


FIGURE 30.19 Behavior of logic gates with n-MOS and p-MOS.

Difference in the Behavior of n-MOS and p-MOS Logic Gates

As illustrated in Fig. 30.19, an n-MOS logic gate behaves differently from a p-MOS logic gate. In the case of an n-MOS logic gate which consists of all nMOSFETs, illustrated in Fig. 30.19(a), the power supply of the n-MOS logic gate must be positive voltage, say +3.3 V, whereas the power supply of the p-MOS logic gate which consists of all pMOSFETs, illustrated in Fig. 30.19(b), must be a negative voltage. Otherwise, these logic gates do not work. Each n-MOS in the driver in Fig. 30.19(a) becomes conductive when a high voltage (e.g., +3.3 V) is applied to its MOSFET gate, and becomes non-conductive when a low voltage (e.g., 0 V) is applied to its MOSFET gate. A direct current flows to the ground from the power supply, as shown by the bold arrow when the driver becomes conductive. In contrast, each p-MOS in the driver in Fig. 30.19(b) becomes non-conductive when a high voltage (e.g., 0 V) is applied to its MOSFET gate, and becomes conductive when a low voltage (e.g., – 3.3 V) is applied to its MOSFET gate. A direct current flows to the power supply from the ground (i.e., in the opposite direction to the case of n-MOS logic gate in (a)), as shown by the bold arrow when the driver becomes conductive. The relationship among voltages at the inputs and outputs for these logic gates are shown in Figs. 30.19(c) and (d). In positive logic, i.e., interpretation of a positive and negative voltages as 1 and 0, respectively, the table in Fig. 30.19(c) yields the truth table for NOR, i.e., $\overline{x \vee y}$, shown in (e), and in negative logic, i.e., interpretation of a positive and negative voltages as 0 and 1 respectively, the table in (d) also yields the truth table for NOR in (e). Similarly, in negative logic, the table in Fig. 30.19(c) yields the truth table for NAND, i.e., $\overline{xy}$, shown in (f), and in positive logic, the table in (d) also yields the truth table for NAND in (f). The two functions in (e) and (f) are dual. The relationships in these examples are extended into the general statements, as shown in the upper part of Fig. 30.19. Whether we use n-MOSFETs or p-MOSFETs in a logic gate, the output of the gate represents the same function by using positive or negative logic, respectively, or negative or positive logic, respectively. Thus, if we have a logic gate, realizing a function f, that consists of all n-MOS or all p-MOS, then the logic gate that is derived by exchanging n-MOS and p-MOS realizes its dual f^d, no matter whether positive logic or negative logic is used.

The output function of a logic gate that consists of all p-MOS can be calculated by Procedure 30.3, regarding each p-MOS as a make-contact relay. But each p-MOS is conductive (instead of being non-conductive) when its gate voltage is low, and is non-conductive (instead of being conductive) when its gate voltage is high. Also the output of the p-MOS logic gate is high when its driver is conductive (in the case of the n-MOS logic gate, the output of the gate is low when its driver is conductive) and is low when its driver is non-conductive (in the case of the n-MOS logic gate, the output of the gate is high when its driver is non-conductive). In other words, the polarity of voltages at the inputs and output of a p-MOS logic gate is opposite to that of an n-MOS logic gate. Thus, by using negative logic, a p-MOS logic gate has the same output function as the n-MOS logic gate in the same connection configuration in positive logic, as illustrated in Fig. 30.19.

References

1. Muroga, S., *Logic Design and Switching Theory*, John Wiley & Sons (now available from Krieger Publishing), 1979.
2. Muroga, S., *VLSI System Design*, John Wily & Sons, 1982.

Muroga, S. "Logic Synthesis with NAND (or NOR) Gates in Multi-levels"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

31

Logic Synthesis With NAND (or NOR) Gates in Multi-levels

Saburo Muroga
University of Illinois at Urbana-Champaign

31.1 Logic Synthesis with NAND (or NOR) Gates

In the previous sections, we have discussed the design of a two-level network with AND and OR gates in **double-rail input logic** (i.e., both x_i and $\bar{x}_i$ for each x_i are available as network inputs) that has a minimum number of gates as the primary objective and a minimum number of connections as the secondary objective. If a network need not be in two levels, we may be able to further reduce the number of gates or connections, but there is no known simple systematic design procedure for this purpose, whether tabular, algebraic, or graphical, that guarantees the minimality of the network. (The integer programming logic design method[4,6] can do this but is complex, requiring long processing time.) But when multi-level minimal networks with NAND gates only (or NOR gates only) are to be designed, there is a method called the **map-factoring method** to design a logic network based on a Karnaugh map.

In designing a logic network in **single-rail input logic** (i.e., only one of x_i and $\bar{x}_i$ for each x_i is available as a network input) with the map-factoring method, it is less easy to see the minimality of the number of gates, although when two-level minimal networks with NAND gates in double-rail input logic are to be designed, it is as easy to see the minimality as two-level minimal networks with AND and OR gates in double-rail input logic on Karnaugh maps. By using designer's intuition based on the pictorial nature of a Karnaugh map, at least reasonably good networks in multi-levels can be derived after trial-and-error efforts. As a matter of fact, minimal networks can sometimes be obtained, although the method does not enable us to prove their minimality. However, if we are satisfied with reasonably good networks, the map-factoring method is useful for manual design. It is actually an extension of the Karnaugh map method for minimal two-level networks with AND and OR gates discussed so far, with far greater

flexibility: by the map-factoring method, we can design not only in two-levels but also in multi-levels in single-rail or double-rail input logic, and also two-level minimal networks with NAND gates in double-rail input logic that can be designed by the map-factoring method are essentially two-level minimal networks with AND and OR gates, as will be discussed later. The map-factoring method, which was first described in Chapter 6 of Ref. 3 for single-rail input logic as discussed later in this chapter, is extended here with minor modification.[5]

Logic networks with NOR gates only or NAND gates only are useful in some cases, such as gate arrays (to be described in Chapter 43) because the simple connection configuration of MOSFETs in each of these gates makes the area small with high speed.

In designing a logic network in multi-levels, we usually minimize the number of logic gates as the primary objective and then the number of connections as the secondary objective. This is because the design with the minimization of the number of logic gates as the primary objective and then the number of connections as the secondary objective is easier than the minimization of the number of connections as the primary objective and then the number of logic gates as the secondary objective, although the minimization of the number of connections, or the lengths of connections (which is far more difficult to minimize), is important because connections occupy significantly large areas on an integrated circuit chip. But judging the results by an experiment by the integer programming logic design method in limited scale, we have the same or nearly same minimal logic networks by either approach.[7]

31.2 Design of NAND (or NOR) Networks in Double-Rail Input Logic by the Map-Factoring Method

NAND gates and NOR gates, which are realized with MOSFETs, are often used in realizing integrated circuit chips, although logic gates that express negative functions which are more complex than NAND or NOR are generally used. (A "negative function" is the complement of a disjunctive form of non-complemented variables. An example is $\overline{xy \vee z}$.) NAND gates are probably more often used than other types of logic gates realizing negative functions because a NAND gate in CMOS has a simple connection configuration of MOSFETs and is fast.

Let us consider representing a NAND gate on a Karnaugh map. The output of the NAND gate with inputs x, $\bar{y}$, and z shown in Fig. 31.1(a) can be expressed as the loop for $x\bar{y}z$ consisting of only 0-cells on the map in Fig. 31.1(b). (Recall that this loop represents product $x\bar{y}z$ on an ordinary Karnaugh map in the previous chapters.) In this case, it is important to note that only 0-cells are contained inside the loop and all the 1-cells are outside. This is because the output of this NAND gate is the complement of the AND operation of inputs, x, $\bar{y}$, and z (i.e., $\overline{x\bar{y}z}$). Thus, if all the 0-cells in the map can be encircled by a single rectangular loop representing a product of some literals (i.e., the number of 0-cells constituting this loop is 2^i where i is a non-negative integer), the map represents the output of a NAND gate whose

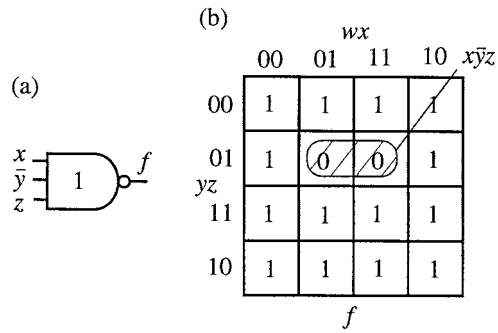


FIGURE 31.1 Representation of a NAND gate on a map.

inputs are the literals in the product represented by the loop. The value of f is 0 for $x = \bar{y} = z = 1$ (i.e., $x = z = 1$ and $y = 0$) in both Figs. 31.1(a) and (b). The value of f is 1 if at least one of x, $\bar{y}$, and z is 0.

Next, connect a new NAND gate (numbered gate 2) to the output of the above NAND gate (i.e., gate 1) and also connect inputs w and z to the new gate, as shown in Fig. 31.2(a). Unlike the case of gate 1 explained in Fig. 31.1, the output f of this new NAND gate is not expressed by the entire loop for wz in Fig. 31.2(b), but is expressed by the portion of the loop (i.e., only 0-cells inside the loop for wz) because the input of gate 2 from gate 1 becomes 0 for the combination, $w = x = z = 1$ and $y = 0$, and consequently f becomes 1 for this combination (if there were no connection from gate 1, the rectangular loop representing wz contains only 0-cells, like gate 1 explained in Fig. 31.1). This may be interpreted as the rectangular loop representing wz for gate 2 being **inhibited** by the loop for gate 1, as shown in Fig. 31.2(b). The remainder of the loop (i.e., all 0-cells, which is actually all the 0-cells throughout the map) is encircled by a loop and is shaded. This shaded loop represents the output of gate 2 and is said to be **associated** with the output of gate 2. In other words, the loop (labeled wz) which represents wz denotes NAND gate 2, whereas the shaded loop (labeled 2) inside this loop in Fig. 31.2(b) denotes the output function of gate 2. Notice that the entire loop for gate 1 is shaded to represent the output function of gate 1, because gate 1 has no inputs from other gates (i.e., gate 1 is inhibited by no other shaded loops) and consequently the loop representing gate 1 coincides with the shaded loop representing the output of gate 1 (i.e., the shaded loop associated with the output of gate 1).

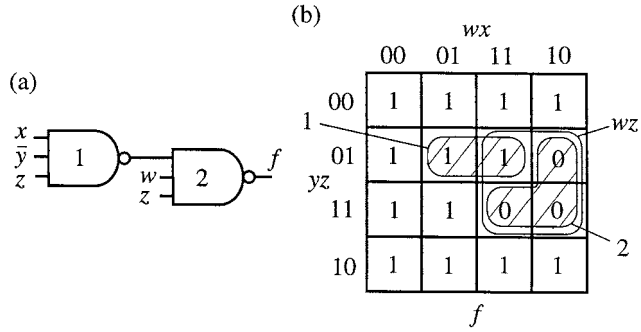


FIGURE 31.2 Representation of a network of two NAND gates.

Now let us state a formal procedure to design a NAND network on a Karnaugh map.

Procedure 31.1: The Map-Factoring Method: Design of a Network in Double-Rail Input Logic with as few NAND Gates as Possible

1. Make the first rectangular loop of 2^i cells. This loop may contain 1-cells, 0-cells, d-cells, or a mixture. Draw a NAND gate corresponding to this loop. As inputs to this gate, connect all the literals in the product that this loop represents. Shade the entirety of this loop.

For example, let us synthesize a network for $f = \bar{w}\bar{y} \vee x\bar{y} \vee \bar{z}$ shown in the Karnaugh map in Fig. 31.3(a). Let us make the first rectangular loop as shown in Fig. 31.3(a). (Of course, the first loop can be chosen elsewhere.) This loop represents product $w\bar{x}$. Draw gate 1, corresponding to this loop and connect inputs w and $\bar{x}$ to this gate. Shade the entirety of this loop because this gate has no input from another gate and consequently the entirety of this loop is associated with the output function of gate 1.

2. Make a rectangular loop consisting of 2^i cells, encircling 1-cells, 0-cells, d-cells, or a mixture. Draw a NAND gate corresponding to this loop. To this gate, connect literals in the product that this loop represents.

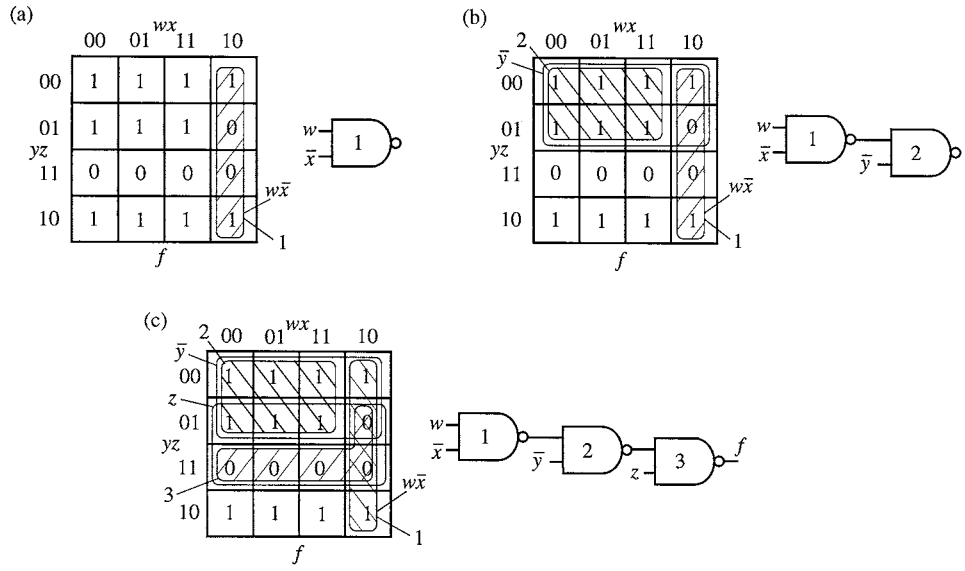


FIGURE 31.3 Example for Procedure 31.1.

Up to this point, this step is identical to Step 1. Now, to this new gate, we further connect the outputs of some or all of the gates already drawn, if we choose to do so. There are the following possibilities:

a. If we choose not to connect any previous gate to the new gate, the new loop is entirely shaded.
b. If we choose to connect some or all of the previously drawn gates to the new gate, encircle and shade the area inside the new loop, excluding the shaded loops of the previously drawn gates connected to the new gate. The shaded loop thus formed is associated with the output of this new gate.

Let us continue our example of Fig. 31.3. Let us make the loop labeled $\bar{y}$ shown in Fig. 31.3(b) as a next rectangular loop consisting of 2^i cells. Draw the corresponding gate 2. Connect input $\bar{y}$ to gate 2 because this loop represents $\bar{y}$. If we choose to connect the output of gate 1 also to gate 2 (i.e., by choosing the case b above), the shaded loop labeled 2 in Fig. 31.3(b) represents the output of gate 2.

3. Repeat Step 2 until the following condition is satisfied:

Termination condition: When a new loop and the corresponding new gate are introduced, all the 0-cells on the entire map and possibly some *d*-cells constitute the shaded loop associated with the output of the new gate.

Continuing our example, let us make the loop labeled z as a next rectangular loop consisting of 2^i, as shown in Fig. 31.3(c). Draw the corresponding gate 3 with input z connected. Choosing the case b in Step 2, connect the output of gate 2 as input of gate 3. (In this case b, we have three choices; that is., connection of the output of gate 1 only, connection of the output of gate 2 only, and connection of both outputs of gates 1 and 2. Let us take the second choice now.) Then, the output of gate 3 is expressed by the shaded loop labeled 3 in Fig. 31.3(c). Now, the termination condition is satisfied: all the 0-cells on the entire map constitute the shaded loop associated with the output of new gate 3. Thus, a network for the given function f has been obtained in Fig. 31.3(c).

For the sake of simplicity, the example does not contain *d*-cells. Even when a map contains *d*-cells, that is, cells for don't-care conditions, the map-factoring method, Procedure 31.1, can be easily used by appropriately interpreting each *d*-cell as a 0-cell or a 1-cell only when we examine the termination condition in Step 3. □

Notice that we can choose different loops (including the first one in Step 1) in each step, leading to different final networks.

Networks with AND and OR Gates in Two Levels, as a Special Case

If we circle only 1-cells possibly along with some *d*-cells but without any 0-cells in each step, we can derive a logic network with NAND gates in two levels, as illustrated in Fig. 31.4(a). This can be easily converted to the network with AND and OR gates in two levels shown in Fig. 31.4(b).

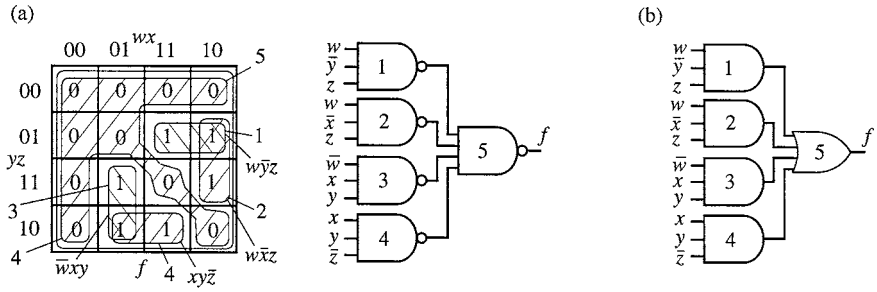


FIGURE 31.4 Network in two levels.

Consideration of Restrictions Such as Maximum Fan-in

If the restriction of maximum fan-in or fan-out is imposed, loops and connections must be chosen so as not to violate it. With the map-factoring method, it is easy to take such a restriction into consideration. Also, the maximum number of levels in a network can be easily controlled.

The Map-Factoring Method for NOR Network

A minimal network of NOR gates for a given function *f* can be designed by the following approach.

Use the map-factoring method to derive a minimal network of NAND gates for f^d, the dual of the given function *f*. Then replace NAND gates in the network with NOR gates. The result will be a minimal network of NOR gates for *f*.

31.3 Design of NAND (or NOR) Networks in Single-Rail Input Logic

In Section 31.2, we discussed the design of a multi-level NAND networks in double-rail input logic, using the map-factoring method. Now let us discuss the design of a multi-level NAND network in single-rail input logic (i.e., no complemented variables are available as inputs to the network) using the map-factoring method.

First let us define **permissible loops** on a Karnaugh map. A permissible loop is a rectangle consisting of cells that contains the cell whose coordinates are variable values of all 1's (i.e., the cell marked with the asterisk in each map in Fig. 31.5), where *i* is one of 0, 1, 2, …. In other words, **a permissible loop must contain the particular cell denoted by the asterisk in each map in Fig. 31.5, where this permissible loop consists of 1-cells, 0-cells, *d*-cells (i.e., don't-care cells), or a mixture**. All permissible loops for three variables are shown in Fig. 31.5.

In the following, let us describe the map-factoring method. The procedure is the same as Procedure 31.1 except the use of permissible loop, instead of rectangular loop of 2^i-cells at any place on the map, for representing a gate.

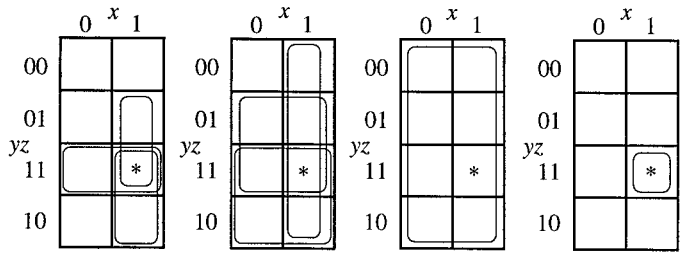


FIGURE 31.5 Permissible loops for three varibles.

Procedure 31.2: The Map-Factoring Method: Design of a Network in Single-Rail Input with as Few NAND Gates as Possible.

We want to design a network with as few NAND gates as possible, under **the assumption that non-complemented variables but no complemented variables are available as network inputs**.

1. Make the first permissible loop, encircling 1-cells, 0-cells, *d*-cells, or a mixture of them. Draw a NAND gate corresponding to this loop. As inputs to this gate, connect all the literals in the product that this loop represents. Shade the entirety of this loop.

For example, when a function $f = x \vee y \vee \bar{z}$ is given, let us choose the first permissible loop, as shown in Fig. 31.6(a), although there is no reason why we should choose this particular loop. (There is no guiding principle for finding which loop should be the first permissible loop. Another loop could be better, but we cannot guess at this moment which loop is the best choice. Another possibility in choosing the first permissible loop will be shown later in Fig. 31.7(a).) The loop we have chosen is labeled 1 and is entirely shaded. Then, NAND gate 1 is drawn. Since the loop represents x, we connect x to this gate.

2. Make a permissible loop, encircling 1-cells, 0-cells, *d*-cells, or mixture of them. Draw a NAND gate corresponding to this loop. To this new gate, connect literals in the product that this loop represents.

In the above example, a new permissible loop is chosen as shown in Fig. 31.6(b), although there is no strong reason why we should choose this particular permissible loop. This loop represents product yz, so y and z are connected to the new gate, which is labeled 2.

To this new NAND gate, we further connect the outputs of some gates already drawn, if we choose to do so. There are the following possibilities.

a. If we choose not to connect any previous gate to the new gate, the new permissible loop is entirely shaded.

If we prefer not to connect gate 1 to gate 2 in the above example, we get the network in Fig. 31.6(b). Ignoring the shaded loop for gate 1, the new permissible loop is entirely shaded and is labeled 2, as shown in Fig. 31.6(b).

b. If we choose to connect some previously drawn gates to the new gate, encircle and shade the area inside the new permissible loop, excluding the shaded loops of the previously drawn gates which are connected to the new gate. In other words, the new permissible loop, except its portion inhibited by the shaded areas associated with the outputs of the connected previously drawn gates, is shaded.

In the above example, if we choose to connect gate 1 to gate 2, we obtain the network in Fig. 31.6(b′). The portion of the new permissible loop that is not covered by the shaded loop associated with gate 1 is shaded and labeled 2.

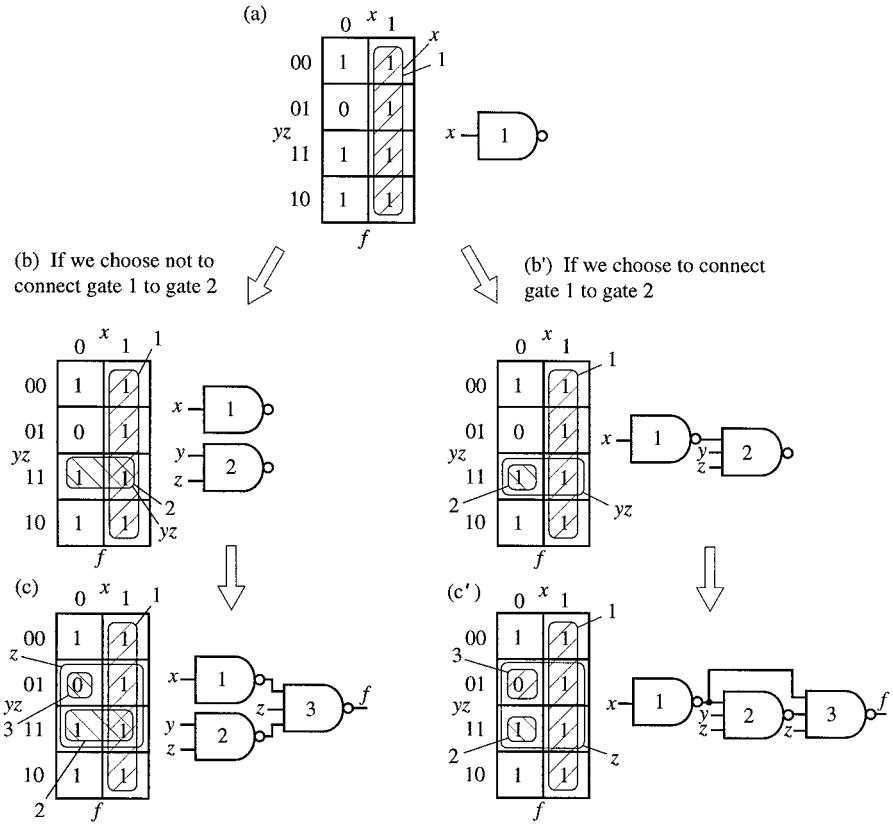


FIGURE 31.6 Map-factoring method applied for $f = x \vee y \vee \bar{z}$.

3. Repeat Step 2 until the following condition is satisfied:

Termination condition: When a new permissible loop and the corresponding new gate are introduced, all the 0-cells on the entire map and possibly some *d*-cells constitute the shaded loop associated with the output of the new gate (i.e., the shaded loop associated with the output of the new gate contains all the 0-cells on the entire map, but no 1-cell).

Let us continue Fig. 31.6(b). If we choose the new permissible loop as shown in Fig. 31.6(c), and if we choose to connect gates 1 and 2 to the new gate 3, the above termination condition has been satisfied; in other words, all the 0-cells on the entire map constitute the shaded loop associated with the output of the new gate 3.

We have obtained a network of NAND gates for the given function. Depending on what permissible loops we choose and how we make connections among gates, we can obtain different networks. After several trials, we choose the best network.

As an alternative for the network for f obtained in Fig. 31.6(c), let us continue Fig. 31.6(b′). If we choose the new permissible loop as shown in Fig. 31.6(c′), and gates 1 and 2 are connected to gate 3, we satisfy the termination condition in Step 3, and we have obtained the network in Fig. 31.6(c′), which is different from the one in Fig. 31.6(c).

If we choose the first permissible loop 1, as shown in Fig. 31.7(a), differently from Fig. 31.6(a), we can proceed with the map-factoring method as shown in Fig. 31.7(b), and we obtain the third network

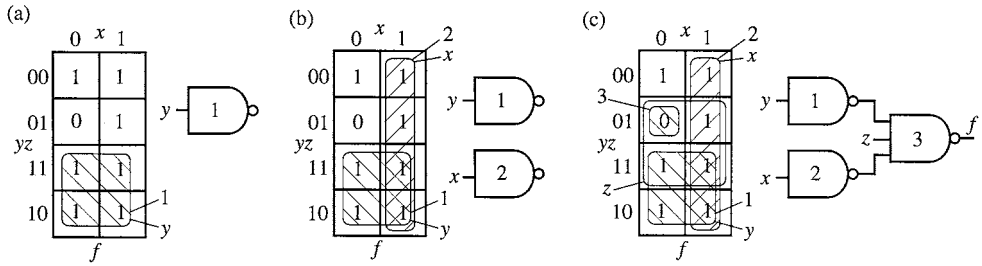


FIGURE 31.7 Network obtained by choosing permissible loops different from Fig. 31.6.

as shown in Fig. 31.7(c). Of course, we can continue differently in Figs. 31.7(b) and (c). Also, the first permissible loop can be chosen differently from Fig. 31.6(a) or 31.7(a), but it is too time-consuming to try all possibilities, so we have to be content with a few trials. We need a few trials to gain a good feeling of how to obtain good selections, and thereafter, we may obtain a reasonably good network. For the above example, $f = x \vee y \vee \bar{z}$, the network obtained in Fig. 31.7, happens to be the minimal network (the minimality of the number of gates and then, as the secondary objective, the minimality of the number of connections can be proved by the integer programming logical design method.).[4,6] □

As might be observed already, any permissible loop in the case of four variables represents a product of non-complemented literals (e.g., the loop chosen in Fig. 31.6(b) represents yz) by letting the permissible loop contain the cell with coordinates, $w = x = y = z = 1$, whereas any rectangular loop of 2^i cells, which does or does not contain this cell, represents a product of complemented literals, non-complemented literals, or a mixture as observed previously.

Extension of the Concept of Permissible Loop

By defining permissible loops differently, we can extend the map-factoring method to designing NAND gate networks with some variables complemented and others non-complemented as network inputs. This is useful when it is not convenient to have other combinations of complemented variables and non-complemented variables as network inputs (e.g., long connections are needed). For example, let us define a permissible loop as a rectangular loop consisting of 2^i cells which contains the cell with coordinates, $x = y = 0$ and $z = 1$. Then, we can design NAND networks with only $\bar{x}$, $\bar{y}$, and z as network inputs. Some functions can be realized with simpler networks by using some network inputs complemented. For example, the function in Fig. 31.7 can be realized with only one NAND gate by using $\bar{x}$, $\bar{y}$, and z as network inputs, instead of three gates in Fig. 31.7(c). Complemented inputs can be realized at the outputs of another network to be connected to the inputs of the network under consideration, or can be realized with inverters. When these network inputs run some distance on a PC board or a chip, the area covered by them is smaller than the case of double-rail input logic because the latter case requires another set of long-running lines for having their complements.

31.4 Features of the Map-Factoring Method

Very often in design practice, we need to design small networks. For example, we need to modify large networks by adding small networks, designing large networks by assembling small networks, or designing frequently used networks which are to be stored as cells in a cell library (these will be discussed in Chapter 45). Manual design is still useful in these cases because designers can understand very well the functional relationships among inputs, outputs, and gates and also complex constraints.

The map-factoring method has unique features that other design methods based on Karnaugh maps with AND and OR gates described in the previous sections do not have. The map-factoring method can synthesize NAND (or NOR) gate networks in single-rail input or double-rail input logic, in not only in two-levels but also in multi-levels. Also, constraints such as maximum fan-in, maximum fan-out, and the number of levels can be easily taken into account. In contrast, methods for designing logic networks with AND and OR gates on Karnaugh maps described in the previous sections can yield networks under several strong restrictions stated previously, such as only two levels and no maximum fan-in restriction. But networks in two levels in double-rail input logic with NAND (or NOR) gates derived by the map-factoring method are essentially networks in two-level of AND and OR gates in double rail-input logic. The usefulness of the map-factoring method is due to the use of a single gate type, NAND gate (or NOR gate type). Although the map-factoring method is intuitive, derivation of minimal networks with NAND gates in multi-levels is often not easy because there is no good guide line for each step (i.e., where we choose a loop) and in this sense, the method is heuristic.

The map-factoring method can be extended to a design problem where some inputs to a network are independent variables, x_1, x_2, ..., x_n and other inputs are logic functions of these variable inputs.

31.5 Other Design Methods of Multi-level Networks with a Minimum Number of Gates

If we are not content with heuristic design methods, such as the map-factoring method, that do not guarantee the minimality of networks and we want to design a network with a minimum number of NAND (or NOR) gates (or a mixture) under arbitrary restrictions, we cannot do so within the framework of switching algebra, unlike the case of minimal two-level AND/OR gate networks (which can be designed based on minimal sums or minimal products), and the **integer programming logic design method** is currently the only method available.[4,6] This method is not appropriate for hand processing, but can design minimal networks for up to about 10 gates within reasonable processing time by computer. The method can design multiple-output networks also, regardless of whether functions are completely or incompletely specified. Also, networks with a mixture of different gate types (such as NAND, NOR, AND, or OR gates), with gates having double outputs, or with wired-OR can be designed, although the processing time and the complexity of programs increase correspondingly. Also, the number of connections can be minimized as the primary, instead of as the secondary, objective.

Although the primary purpose of the integer programming logic design method is the design of minimal networks with a small number of variables, minimal networks for some important functions of an arbitrary number of variables (i.e., no matter how large n is for a function of n variables), such as adders[1,7] and parity functions,[2] have been derived by analyzing the intrinsic properties of minimal networks for these networks.

References

1. Lai, H. C. and S. Muroga, "Minimum binary adders with NOR (NAND) gates," *IEEE Tr. Computers*, vol. C-28, pp. 648-659, Sept. 1979.
2. Lai, H. C. and S. Muroga, "Logic networks with a minimum number of NOR (NAND) gates for parity functions of n variables," *IEEE Tr. Computers*, vol. C-36, pp. 157-166, Feb. 1987.
3. Maley, G. A., and J. Earle, *The Logical Design of Transistor Digital Computers*, Prentice-Hall, Englewood Cliffs, NJ, 1963.
4. Muroga, S. "Logic design of optimal digital networks by integer programming," *Advances in Information Systems Science*, vol. 3, ed. by J. T. Tou, Plenum Press, pp. 283-348, 1970.
5. Muroga, S., *Logic Design and Switching Theory*, John Wiley & Sons (now available from Krieger Publishing Co.), 1979.

6. Muroga, S., "Computer-aided logic synthesis for VLSI chips," *Advances in Computers*, vol. 32, ed. by M. C. Yovits, Academic Press, pp. 1-103, 1991.
7. Muroga, S. and H. C. Lai, "Minimization of logic networks under a generalized cost function," *IEEE Tr. Computers*, vol. C-25, pp. 893-907, Sept. 1976.
8. Sakurai, A. and S. Muroga, "Parallel binary adders with a minimum number of connections," *IEEE Tr. Computers*, vol. C-32, pp. 969-976, Oct. 1983. (In Fig. 7, labels a_0 and $\bar{c}_0$ should be exchanged.)

Muroga, S. "Logic Synthesis with a Minimum Number of Negative Gates"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

32

Logic Synthesis with a Minimum Number of Negative Gates

Saburo Muroga
University of Illinois at Urbana-Champaign

32.1 Logic Design of MOS Networks

A MOS logic gate can express a negative function and it is not directly associated with a simple logic expression such as a minimal sum. So, it is not a simple task to design a network with MOS logic gates so that the logic capability of each MOS logic gate to express a negative function is utilized to the fullest extent. Here let us describe one of few such design procedures that design transistor circuits directly from the given logic functions.[7,8]

A logic gate whose output represents a negative function is called a **negative gate**. A MOS logic gate is a negative gate. We now design a network with a minimum number of negative gates. The **feed-forward network** shown in Fig. 32.1 (the output of each gate feeds forward to the gates in the succeeding levels) can express any loopless network. Let us use Fig. 32.1 as the general model of a loopless network.

The following procedure designs a logic network with a minimum number of negative gates, assuming that only non-complemented input variables (i.e., $x_1, x_2, \ldots, x_n$) are available as network inputs (i.e., single-rail input logic), based on this model.[5,6,9]

Procedure 32.1: Design of Logic Networks with a Minimum Number of Negative Gates in Single-Rail Input Logic

We want to design a MOS network with a minimum number of MOS logic gates (i.e., negative gates) for the given function $f(x_1, x_2, \ldots, x_n)$. (The number of interconnections among logic gates is not necessarily minimized.) It is assumed that only non-complemented variables are available as network inputs. The network is supposed to consist of MOS logic gates g_i's whose outputs are denoted by u_i's, as shown in Fig. 32.1.

Phase 1

1. Arrange all input vectors $V = (x_1, x_2, \ldots, x_n)$ in a lattice, as shown in Fig. 32.2, where the nodes denote the corresponding input vectors shown in parentheses. White nodes, black nodes, and nodes with a cross in a circle, ⊗, denote true vectors, false vectors, and don't-care vectors, respectively. The number of 1's contained in each vector V is defined as the **weight** of the vector. All vectors with the same weight are on the same level, placing vectors with greater weights in higher levels, and every pair of vectors that differ only in one bit position is connected by a short line.

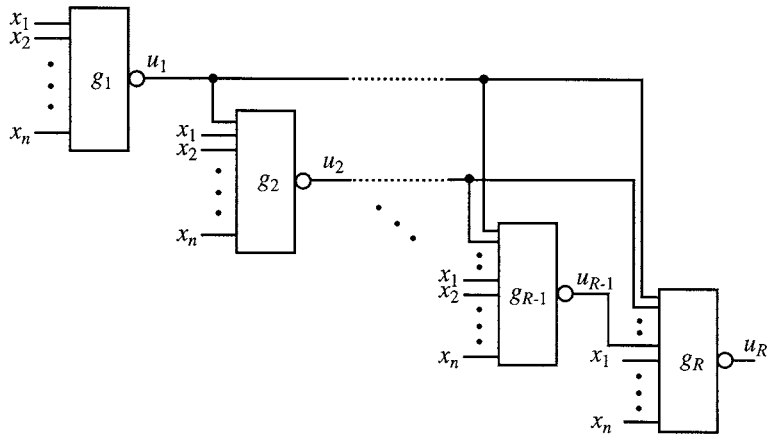


FIGURE 32.1 A feed-forward network of negative gates.

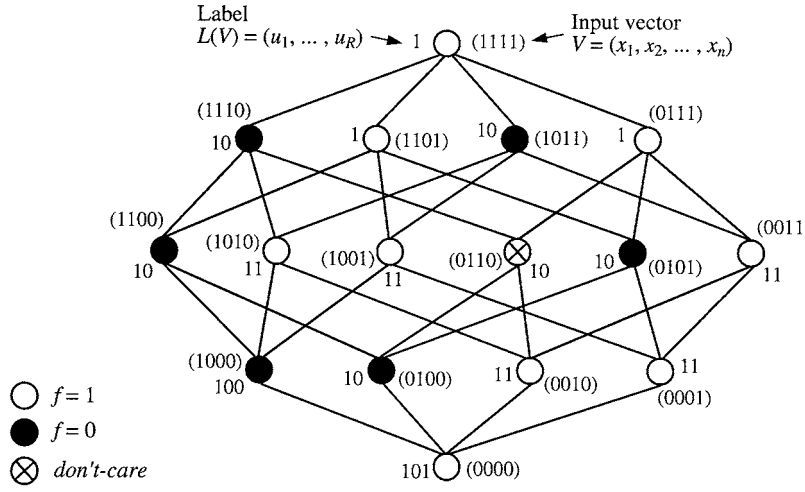


FIGURE 32.2 Lattice example for Procedure 32.1.

Figure 32.2 is a lattice example for an incompletely specified function of four variables.

2. We assign the label $L(V)$ to each vector $V = (x_1, x_2, \ldots, x_n)$ in the lattice in Steps 2 and 3. Henceforth, $L(V)$ is shown without parentheses in the lattice.

First assign the value of f to the vector $(11\ldots1)$ of weight n at the top node as $L(11\ldots1)$. If f for the top node is "don't-care," assign 0.

In the example in Fig. 32.2, we have $L(11\ldots1) = 1$ because the value of f for the top node is 1, as shown by the white node.

3. When we finish the assignment of $L(V)$ to each vector of weight w, where $0 < w \leq n$, assign $L(V')$ to each vector V' of weight $w-1$, the **smallest binary number** satisfying the following conditions:

If $f(V')$ is not "don't-care,"

a. The least significant bit of $L(V')$ is $f(V')$ (i.12e., the least significant bit of $L(V')$ is 0 or 1, according to whether f is 0 or 1), and

b. The other bits of $L(V')$ must be determined such that $L(V') \geq L(V)$ holds for every vector V of weight w that differs from V' in only one bit position. In other words, the binary number represented by $L(V')$ is not smaller than the binary number represented by $L(V)$.

If $f(V')$ is "don't-care," ignore (a), but consider (b) only. [Consequently, the least significant bit of $L(V')$ is determined such that (b) is met.]

For the example we get a label $L(1110) = 10$ for the node for vector $V' = (1110)$ because the last bit must be $0 = f(1110)$ by (a), and the number must be equal to or greater than the label 1 for the top node for vector (1111) by (b). Also, for vector $V' = (1000)$, we get a label $L(1000) = 100$ because the last bit must be $0 = f(1000)$ by (a), and the label $L(1000)$ as a binary number must be equal to or greater than each of the labels, 10, 11, and 11, already assigned to the three nodes (1100), (1010), and (1001), respectively.

4. Repeat Step 3 until a label $L(00\ldots0)$ is assigned to the bottom vector (00. . .0). Then the bit length of $L(00\ldots0)$ is the minimum number of MOS logic gates required to realize f. Denote it by R.

Then make all previously obtained $L(V)$ into binary numbers of the same length as $L(00\ldots0)$ by adding 0's in front of them such that every label $L(V)$ has exactly R bits. For the example, we have $R = 3$, so the label $L(11\ldots1) = 1$ obtained for the top node is changed to 001 by adding two 0's.

Phase 2

Now let us derive MOS logic gates from the $L(V)$'s found in Phase 1.

1. Denote each $L(V)$ obtained in Phase 1 as $(u_1,\ldots, u_i, u_{i+1},\ldots, u_R)$. As will be seen later, $u_1,\ldots, u_i, u_{i+1},\ldots, u_R$ are log functions realized at the outputs of logic gates, $g_1,\ldots, g_i, g_{i+1},\ldots, g_R$, respectively, as shown in Fig. 32.1.
2. For each $L(V) = (u_1,\ldots, u_i, u_{i+1},\ldots, u_R)$ that has $u_{i+1} = 0$, make a new vector $(V, u_1,\ldots, u_i)$ (i.e., $(x_1,\ldots, x_n, u_1,\ldots, u_i)$) which does not include $u_{i+1},\ldots, u_R$.

This means the following. For each i, find all labels $L(V)$'s whose $(i + 1)$-th bit is 0 and then for each of these labels, we need to create a new vector $(x_1,\ldots, x_n, u_1,\ldots, u_i)$ by containing only the first i bits of the label $L(V)$. For example, for $u_1 = 0$ (i.e., by setting i of "$u_{i+1} = 0$" to 0), the top node of the example lattice in Fig. 32.2 has the label $(u_1, u_2, u_3) = (001)$ which has $u_1 = 0$ (the label 1 which was labeled in Step 3 of Phase 1 was changed to 001 in Step 4). For this label, we need to create a new vector $(x_1, x_2, x_3, x_4, u_1,\ldots, u_i)$, but the last bit u_i becomes u_0 because we set $i = 0$ for $u_{i+1} = 0$. There is no u_0, so the new vector is $(x_1, x_2, x_3, x_4) = (1111)$, excluding $u_1,\ldots, u_i$. (In this sense, the case of $u_1 = 0$ is special, unlike the other cases of $u_2 = 0$ and $u_3 = 0$ in the following.) For other nodes with labels having $u_1 = 0$, we create new vectors in the same manner.

Next, for $u_2 = 0$ (i.e., by setting i of "$u_{i+1} = 0$" to 1), the top node has the label $(u_1, u_2, u_3) = (001)$ which has $u_2 = 0$. For this label, we need to create a new vector $((x_1, x_2, x_3, x_4, u_1)$, including u_1 this time. So, the new vector $((x_1, x_2, x_3, x_4, u_1) = (11110)$ results as the label $L(1111) = 001$ for the top node. Also, a new vector (11010) results for $L(1101) = 001$, and so on.

3. Find all the minimal vectors from the set of all the vectors found in Step 2, where the **minimal vectors** are defined as follows. When $a_k \geq b_k$ holds for every k for a pair of distinct vectors $A = (a_1,\ldots, a_m)$ and $B = (b_1,\ldots, b_m)$, then the relation is denoted by

$$A \succ B$$

and B is said to be smaller than A. In other words, A and B are compared bit-wise. If no vector in the set is smaller than B, B is called a minimal vector of the set. For example, $(10111) \succ (10101)$ because a bit (i.e., 1 or 0) in every bit position of (10111) is greater than or equal to the corresponding bit in

(10101). But (10110) and (10101) are **incomparable** because a bit in each of the first four bit positions of (10110) is greater than or equal to the corresponding bit of (10101), but the fifth bit of (10110) (i.e., 0) is smaller than the corresponding bit (i.e., 1) of (10101).

For the example, for $u_1 = 0$, the minimal vectors are (0100), (0010), (0001). Then, for $u_2 = 0$, we get (11110), (11010), (01110), (10001), and (00001) by Step 2. Then the minimal vectors are (11010), (01110), and (0000l). Here, notice that (11110), for example, cannot be a minimal vector because (11110) $\succ$ (11010). (11010) cannot be compared with other two, (01110) and (0001), with respect to $\succ$.

4. For every minimal vector, make the product of the variables that have 1's in the components of the vector, where the components of the vector (V, u_1,…, u_i) denote variables x_1, x_2, …, x_n, u_1, …, u_i, in this order. For example, we form $x_1x_2x_4$ for vector (11010). Then make a disjunction of all these products and denote it by $\bar{u}_{i+1}$.

For the example, we get

$$\bar{u}_2 = x_1x_2x_4 \vee x_2x_3x_4 \vee u_1$$

from the minimal vectors (11010), (01110), and (00001) for $u_2 = 0$.

5. Repeat Steps 2 through 4 for each of $i = 1, 2,\ldots, R-1$.

For the example, we get

$$\bar{u}_1 = x_2 \vee x_3 \vee x_4 \text{ and } \bar{u}_3 = x_1x_3x_4u_2 \vee x_1u_1 \vee x_2u_2$$

6. Arrange R MOS logic gates in a line along with their output functions, u_1, …, u_R. Then construct each MOS logic gate according to the disjunctive forms obtained in Steps 4 and 5, and make connections from other logic gates and input variables (e.g., MOS logic gate g_2, whose output function is u_2, has connections from x_1, x_2, x_3, x_4, and u_1 to the corresponding MOSFETs in g_2, according to disjunctive form $\bar{u}_2 = x_1x_2x_4 \vee x_2x_3x_4 \vee u_1$). The network output, u_3 (i.e., $u_3 = \overline{x_1x_3x_4u_2 \vee x_1u_1 \vee x_2u_2}$, which can be rewritten as $\bar{x}_1\bar{x}_2 \vee x_1\bar{x}_3x_4 \vee x_2x_3x_4 \vee \bar{x}_2x_3\bar{x}_4$) realizes the given function f.

For the example, we get the network shown in Fig. 32.3 (the network shown here is with logic gates in n-MOS only but it is easy to convert this into CMOS, as we will see in Chapter 36).

Phase 3

The bit length R in label $L(00\ldots0)$ for the bottom node shows the number of MOS logic gates in the network given at the end of Phase 2. Thus, if we do not necessarily choose the smallest binary number in Step 3 of Phase 1, but choose a binary number still satisfying the other conditions (i.e., (a) and (b)) in Step 3 of Phase 1, then we can still obtain a MOS network of the same minimum number of MOS logic gates as long as the bit length R of $L(00\ldots0)$ is kept the same. (For the top node also, we do not need to choose the smallest binary number as $L(1\,1\ldots)$, no matter whether f for the node is don't-care.)

This freedom may change the structure of the network, although the number of logic gates is still the minimum. Among all the networks obtained, there is a network that has a minimum number of logic gates as its primary objective, and a minimum number of interconnections as its secondary objective. (Generally, it is not easy to find such a network because there are too many possibilities.) □

Although the number of MOS logic gates in the network designed by Procedure 32.1 is always minimized, the networks designed by Procedure 32.1 may have the following problems: (1) the number

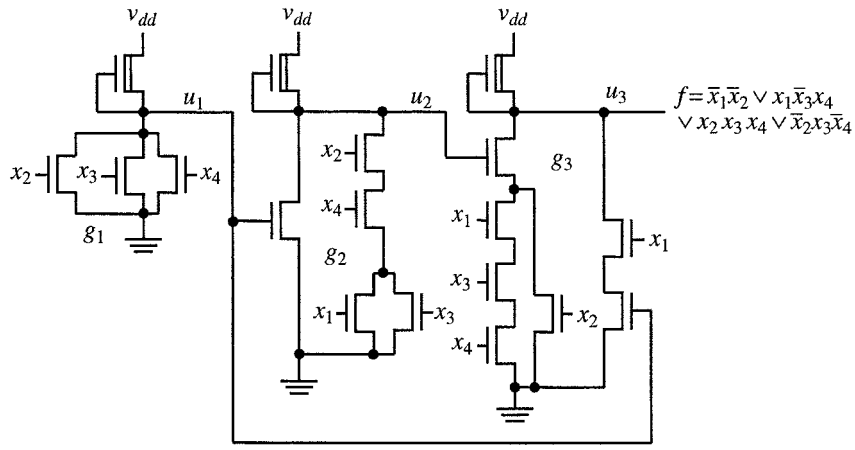


FIGURE 32.3 MOS network based on the labels $L(V)$ in Fig. 32.2.

of interconnections is not always minimized; (2) some logic gates may become very complex so that these logic gates may not work properly with reasonably small gate delay times. If so, we need to split these logic gates into a greater number of reasonably simple logic gates, giving up the minimality of the number of logic gates. Also, after designing several networks according to Phase 3, we may be able to find a satisfactory network.

32.2 Algorithm DIMN

Compared with the problem (1) of Procedure 32.1, problem (2) presents a far more serious difficulty. Thus, Algorithm DIMN (an acronym for Design of Irredundant Minimal Network) was developed to design a MOS network in single-rail input logic such that the number of gates is minimized and every connection among cells is irredundant (i.e., if any connection among logic gates is removed, the network output will be changed).[2,3] Algorithm DIMN is powerful but is far more complex than the minimal labeling procedure (i.e., Phases 1 and 2 of Procedure 32.1). So, let us only outline it.

Procedure 32.2: Outline of Algorithm DIMN

1. All the nodes of a lattice are labeled by the minimal labeling procedure (i.e., Phases 1 and 2 of Procedure 32.1), starting with the top node and moving downward. Let the number of bits of each label be R. Then all the nodes are labeled by a procedure similar to the minimal labeling procedure, starting with the bottom node which is now labeled with the largest binary number of R bits, and moving upward on the lattice. Then, the first negative gate with irredundant MOSFETs is designed after finding as many don't-cares as possible by comparing two labels at each node which are derived by these downward and upward labelings.
2. The second negative gate with irredundant MOSFETs is designed after downward and upward labelings to find as many don't-cares as possible. This process is repeated to design each gate until the network output gate with irredundant MOSFETs is designed. □

Unlike the minimal labeling procedure where the downward labeling is done only once and then the entire network is designed, Algorithm DIMN repeats the downward and upward labelings for designing each negative gate. The network derived by DIMN for the function $x_1x_2x_3x_4 \vee x_1x_2\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2x_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3x_4$, for example, has three negative gates, the same as that of the network derived by the minimal labeling procedure for the same function. The former, however, has only 12 n-channel MOSFETs, whereas the latter has 20 n-channel MOSFETs. DIMN usually yields networks that are substantially simpler than

those designed by the minimal labeling procedure, but it has the same problems as Procedure 32.1 when the number of input variables of the function becomes large.

The computational efficiency of DIMN was improved.[4] Also, a version of DIMN specialized to two-level networks was developed.[1]

References

1. Hu, K. C., Logic Design Methods for Irredundant MOS Networks, Ph.D. dissertation, Dept. Comput. Sci., University of Illinois, at Urbana-Champaign, Aug. 1978, Also Report. No. UIUCDCS-R-80-1053, 317, 1980.
2. Lai, H. C. and S. Muroga, "Automated logic design Of MOS networks," Chapter 5, in *Advances in Information Systems Science*, vol. 9, ed. by J. Tou, Plenum Press, New York, pp. 287-336, 1985.
3. Lai, H. C. and S. Muroga, "Design of MOS networks in single-rail input logic for incompletely specified functions," *IIEEE Tr. CAD*, 7, pp. 339-345, March 1988.
4. Limqueco, J. C., Algorithms for the Design of Irredundant MOS Networks, Master thesis, Dept. Comput. Sci., University of Illinois, Urbana, IL, 87, 1998.
5. Liu, T. K., "Synthesis of multilevel feed-forward MOS networks," *IEEE TC,* pp. 581-588, June 1977.
6. Liu, T. K., "Synthesis of feed-forward MOS network with cells of similar complexities," *IEEE TC*, pp. 826-831, Aug. 1977.
7. Muroga, S., *VLSI System Design*, John Wily & Sons, 1982.
8. Muroga, S., *Computer-Aided Logic Synthesis for VLSI Chips*, ed. by M. C. Yovits, vol. 32, Academic Press, pp. 1-103, 1991.
9. Nakamura, K., N. Tokura, and T. Kasami, "Minimal negative gate networks," *IEEE TC*, pp. 5-11, Jan. 1972.

Yoshikawa, K., et al. "Logic Synthesizer with Optimizations in Two Phases"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

33

Logic Synthesizer with Optimizations in Two Phases

Ko Yoshikawa
NEC Corporation

Saburo Muroga
University of Illinois at Urbana-Champaign

Logic networks and then their corresponding transistor circuits to be laid out on integrated chips have been traditionally designed manually, spending long time and repeatedly making mistakes and correcting them. As the cost and size of transistor circuits continue to decline, logic networks that are realized by transistor circuits have been designed with an increasingly large number of logic gates. Manual design of such large logic networks is becoming too time-consuming and prone to design mistakes, thus necessitating automated design. Logic synthesizers are automated logic synthesis systems used for this purpose and transform the given logic functions into technology-dependent logic circuits that can be easily realized as transistor circuits. The quality of technology-dependent logic circuits derived by logic synthesizers is not necessarily better than manually designed ones, at least for those with a small number of logic gates, but technology-dependent logic circuits for those with millions of logic gates cannot be designed manually for reasonably short times.

Automated design of logic networks has been attempted since the early 1960s.[3] Since the beginning of the 1960s, IBM has pushed research of design automation in logic design. In the 1970s, a different type of algorithm, called the Transduction method, was devised for automated design of logic networks, as described in Chapter 34. Since the beginning of the 1980s, the integration size of integrated chips has tremendously increased, research and development of automated logic synthesis has become very active at several places[1,2,4,9] and powerful logic synthesizers have become commercially available.[6]

There are different types of logic synthesizers. But here, let us describe a logic synthesizer that derives a technology-dependent logic circuit in two phases,[7] although there are logic synthesizers where, unlike the following **synthesizer in two phases**, only technology-dependent optimization is used throughout the transformation of the given logic functions into technology-dependent logic circuits that are easily realizable as transistor circuits:

> In the first phase, an optimum logic network that is not necessarily realizable as a transistor circuit is designed by Boolean algebraic approaches which are easy to use. This phase is called **technology-independent optimization**.
>
> In the second phase, the logic network derived in the first phase is converted into a technology-dependent logic circuit that is easily realizable as a transistor circuit. (Note that all logic synthesizers, including the logic synthesizer in two phases, must eventually have technology-dependent logic circuits that are easily realizable, as transistor circuits.) This phase is called **technology-dependent optimization**.

The logic synthesizer in two phases has variations, depending on what optimization algorithms are used, and some of them have the advantage of short processing time.

Suppose a logic network is designed for the given functions by some means, including design methods described in the previous chapters. Figure 33.1 is an example of such logic networks. Then the logic network is to be processed by many logic optimization algorithms in sequence, as shown in Fig. 33.2. As illustrated in Fig. 33.1, a subnetwork (i.e., several logic gates that constitute a small logic network by themselves) that has a single output logic function is called a **node,** and its output function is expressed as a sum-of-products form. Thus, the entire logic network is expressed as a network of the nodes. During the first phase of technology-independent optimization, each node does not correspond to a transistor circuit. But the second phase of technology dependent-optimization, as illustrated in Fig. 33.3, converts the logic network derived by the first phase into technology-dependent circuits, in which each node corresponds to a cell [i.e., a small transistor circuit (shown in each dot-lined loop in Fig. 33.3)] in a library of cells. The cell library consists of a few hundred cells where a cell is a small transistor circuit with good layout. These cells are designed beforehand and can be repeatedly used for logic synthesis.

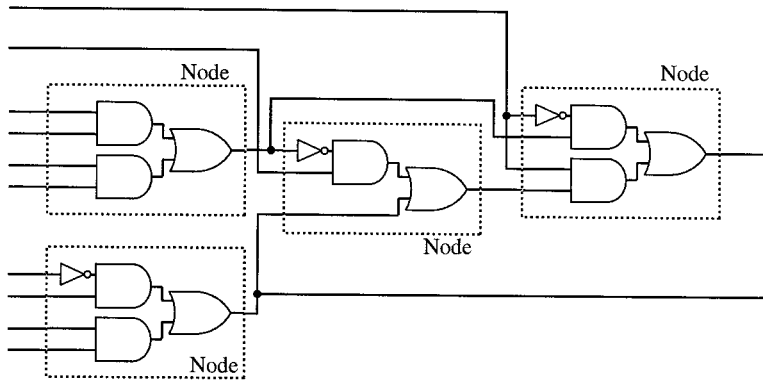


FIGURE 33.1 Logic network to be processed by a logic synthesizer.

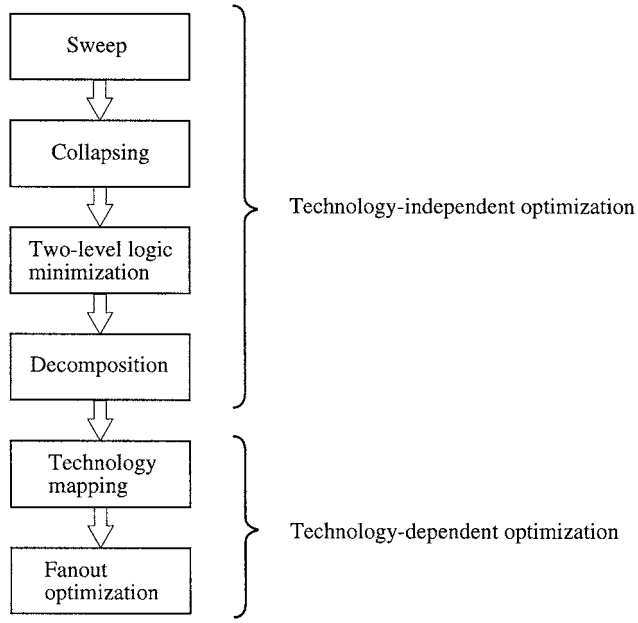
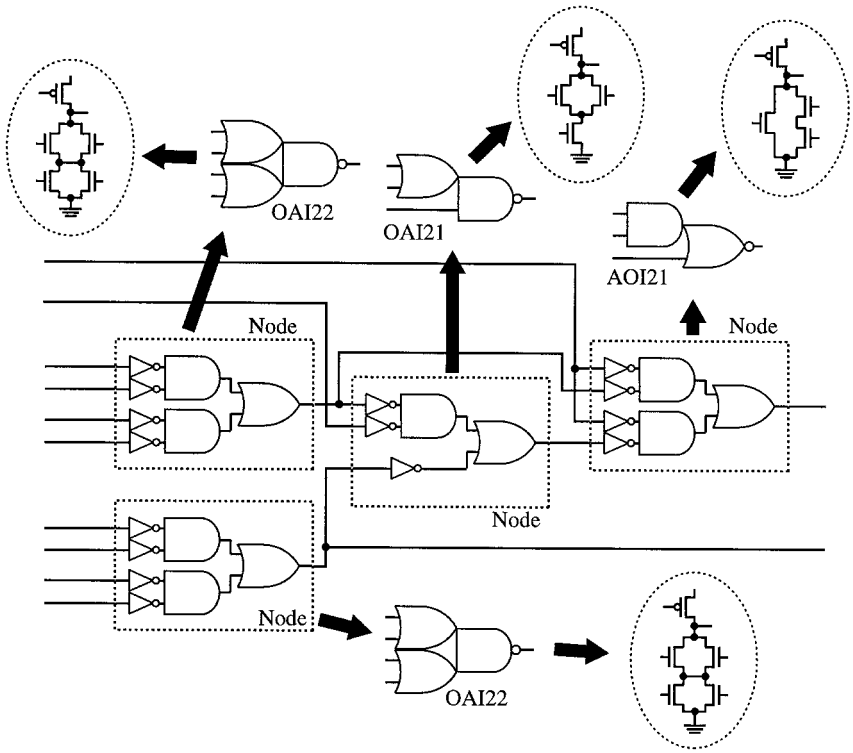


FIGURE 33.2 A logic optimization flow in the logic synthesizer in two phases.



OAI22, OAI21, and AOI21 are the cells in the cell library.

FIGURE 33.3 Technology-dependent logic circuit.

A typical flow of logic optimization in the logic synthesis in two phases works as follows and is illustrated in Fig. 33.2, although there may be variations:

1. Sweep, illustrated in Fig. 33.4: Nodes that do not have any fan-outs are deleted. A node that consists of only a buffer or an inverter node is merged into next nodes. The following rules optimize nodes that have inputs of constant values, 0 or 1:

$$A \cdot 1 = A,\ A \cdot 0 = 0,\ A \vee 1 = 1,\ A \vee 0 = A$$

2. Collapsing (also often called flattening), illustrated in the upper part of Fig.33.5: More than one node is merged into one node, so that it has a more complicated logic function than before.
3. Two-level logic minimization: Two-level logic minimization[2] is carried out at each node, as illustrated in the lower part of Fig. 33.5. (PLA minimization programs to be described in Chapter 42 can be used for this purpose.)
4. Decomposition, illustrated in Fig. 33.6: Decomposition[4] extracts common expressions from the sum-of-products forms of nodes, adds intermediate nodes whose logic functions are the same as the common expressions, and re-expresses the sum-of-products form at the original nodes, using the intermediate nodes. This transformation, called **weak division** (described in Chapter 29), reduces the area of the circuit.
5. Technology mapping,[5] illustrated in Fig. 33.7: Technology mapping finds an appropriate mapping of the network of the nodes onto the set of cells in the target cell library. Technology mapping works as follows:

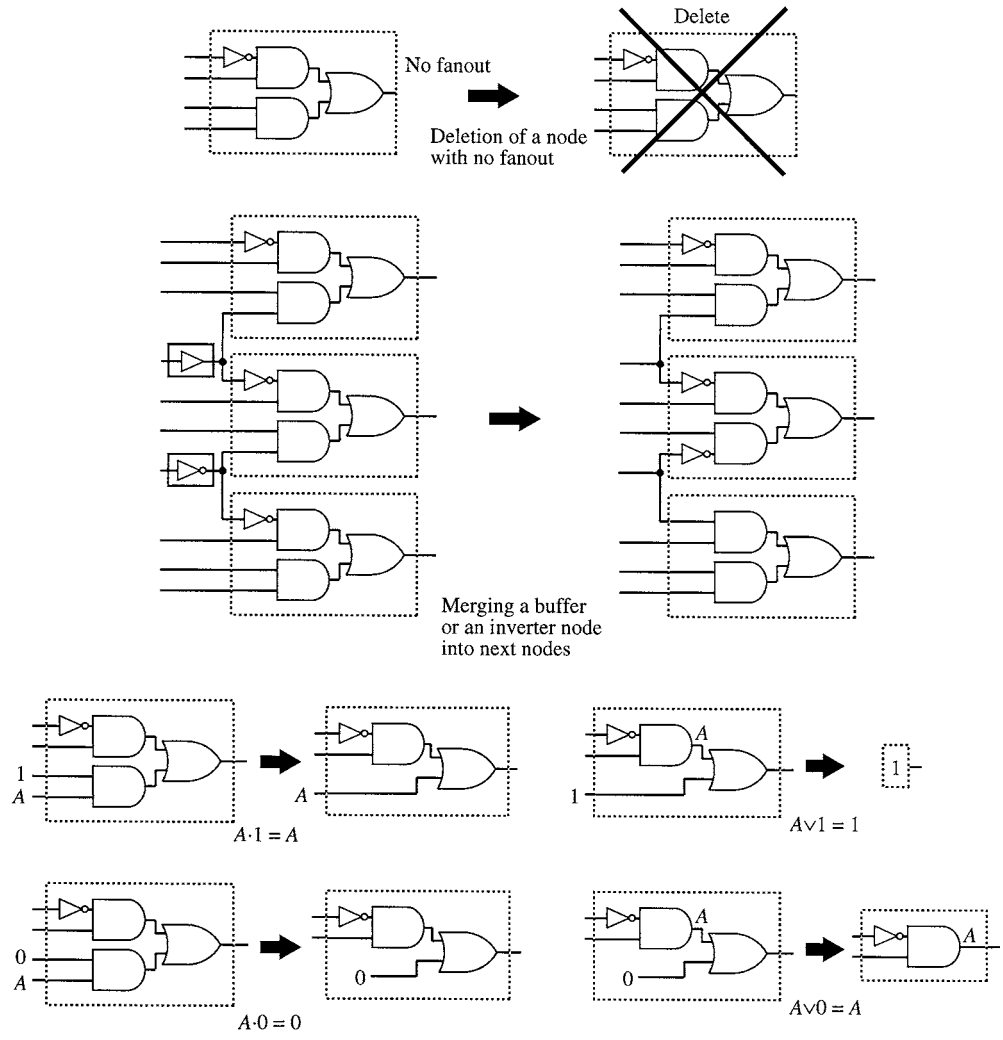


FIGURE 33.4 Sweep.

a. Select a tree structure: a tree structure where all nodes have one fan-out node is extracted from the circuit.

b. Decompose it to two-input NANDs and inverters: a selected tree is decomposed to two-input NANDs and inverters. The result of this decomposition is called a subject tree.

c. Matching: all cells in the target cell library are also decomposed to two-input NANDs and inverters, as shown in Fig. 33.8. Pattern matching between the subject tree and the cells in the cell library is carried out to get the best circuit.

Timing or power optimization, that is, minimization of delay time or power consumption under other conditions (e.g., minimization of delay time without excessively increasing power consumption, or minimization of power consumption without sacrificing speed too much) is also carried out by changing the structure of the nodes and their corresponding cells in the library.

6. Fanout optimization, illustrated in Fig. 33.9: Fanout optimization attempts to reduce delay time by the following transformations.

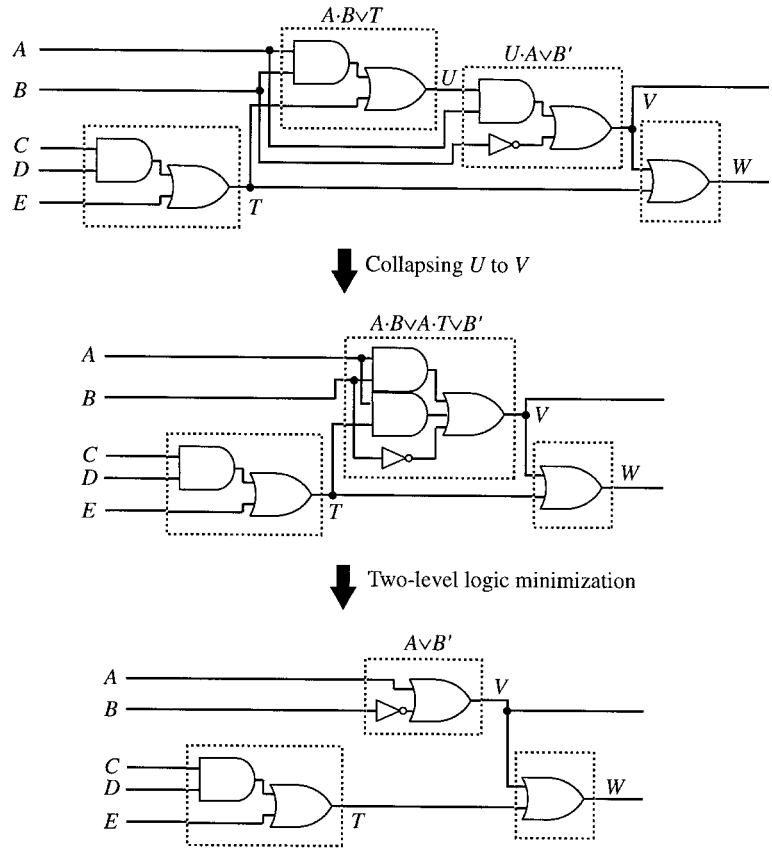


FIGURE 33.5 Collapsing and two-level logic minimization.

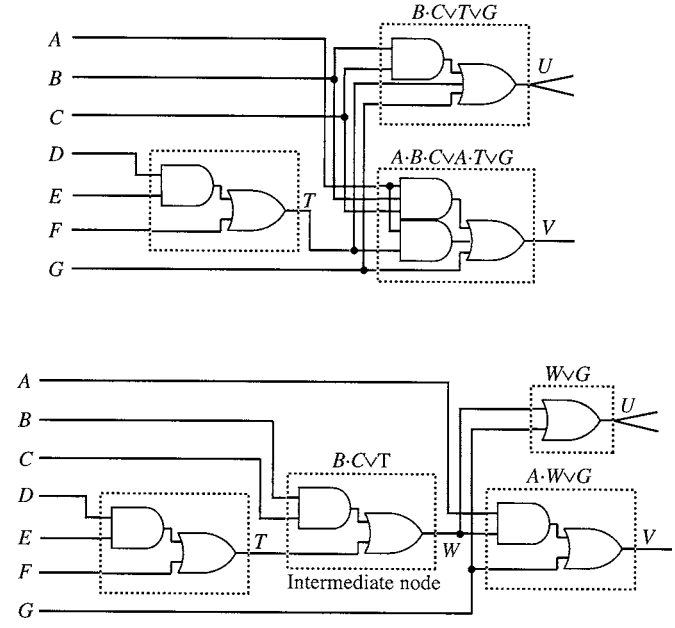


FIGURE 33.6 Decomposition.

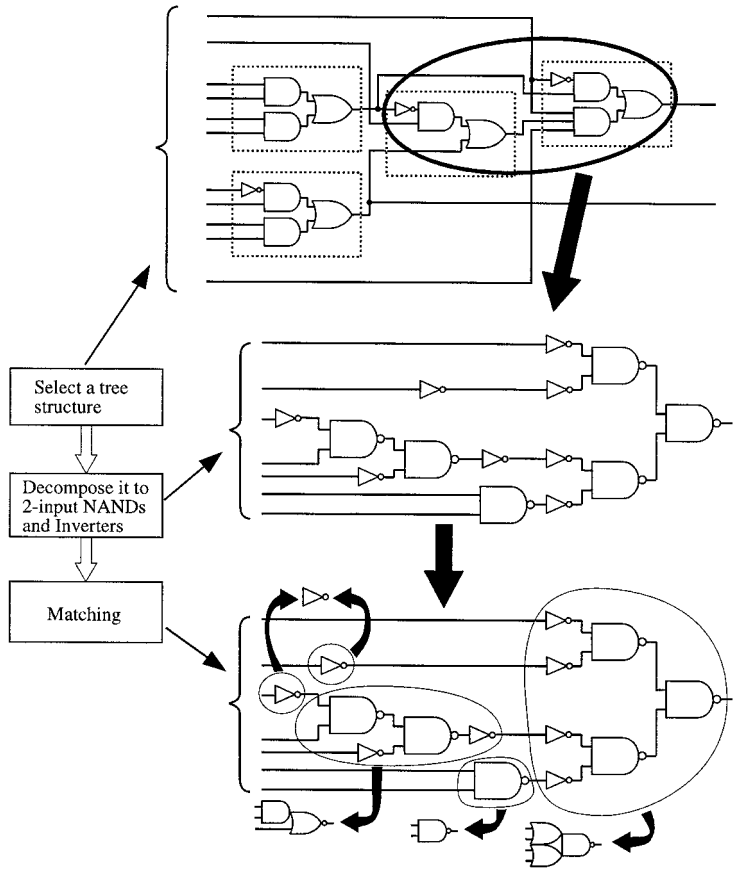


FIGURE 33.7 Technology mapping.

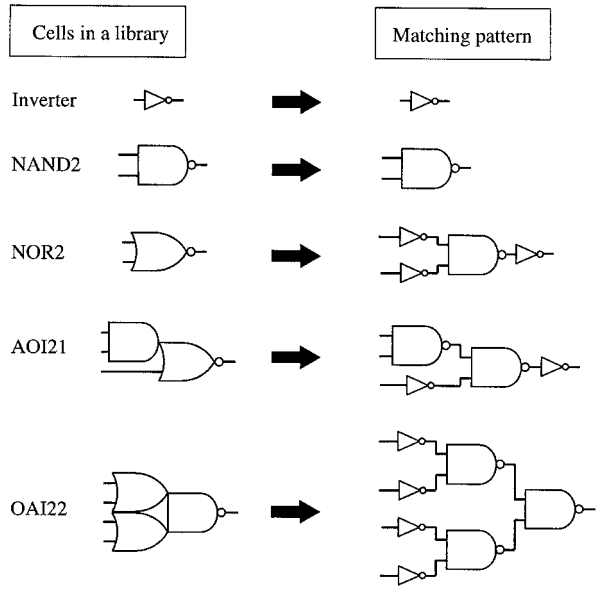


FIGURE 33.8 Pattern trees for a library.

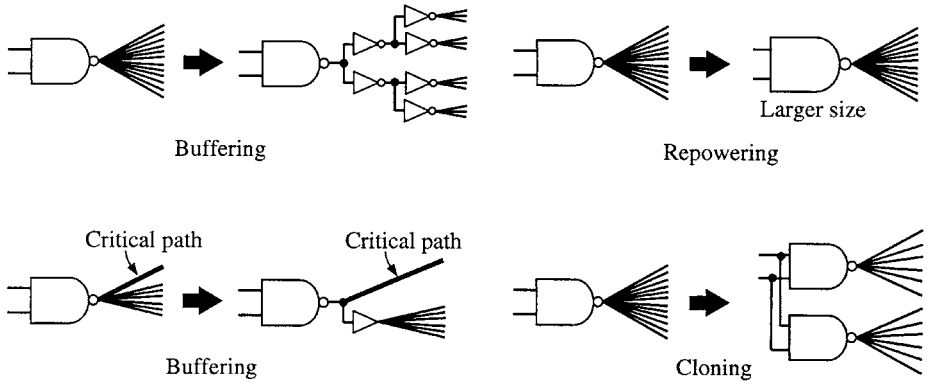


FIGURE 33.9 Fan-out optimization.

(a) Buffering: inserting buffers and inverters in order to reduce load capacitance of logic gates that have many fanouts. In particular, a critical path that is important for performance is separated from other fanouts.[8]
(b) Repowering: replacing a logic gate that has many fanouts, by another cell in the cell library that has greater output power (its cell is larger) and the same logic function as the original one. This does not change the structure of the circuit.[8]
(c) Cloning: creating a clone node for a logic gate that has many fanouts and distributing the fanouts among these nodes to reduce the load capacitance.

This is a typical logic optimization flow. Many other optimization algorithms can be incorporated within this flow to synthesize better circuits.

References

1. Bergamaschi, R. A. et al., "High-level synthesis in an industrial environment," *IBM Jour. Res. and Dev.*, vol. 39, pp. 131-148, Jan./March 1995.
2. Brayton R. K., G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis,* Kluwer Academic Publishers, Boston, 1984.
3. Breuer, M. A., "General survey of design automation of digital computers," *Proc. IEEE*, vol. 54, no. 12, pp. 1708-1721, Dec., 1966.
4. Brayton R. K., A. Sangiovanni-Vincentelli, and A. Wang, "MIS: A multiple-level logic optimization system," *IEEE Tr. CAD*, CAD-6, no. 6, pp. 1062-1081, Nov. 1987.
5. Detjens E., G., Gannot, R., Rudell, A., Sangiovanni-Vincentelli, and A. Wang, "Technology mapping in MIS," *Proc. Int'l Conf. CAD*, pp. 116-119, 1987.
6. Kurup, P. and T. Abbasi, Ed., *Logic Synthesis Using Synopsys*, 2nd ed., Kluwer Academic Publishers, 322, 1997.
7. Rudell R., "Tutorial: Design of a logic synthesis system," *Proc. 33rd Design Automation Conf.*, pp. 191-196, 1996.
8. Singh K. J. and A Sangiovanni-Vincentelli., "A heuristic algorithm for the fanout problem," *Proc. 27th Design Automation Conf.*, pp. 357-360, 1990.
9. Stok, L., et al. (IBM), "BooleDozer: Logic synthesis for ASICs," *IBM Jour. Res. Dev.*, vol. 40, no. 4, pp. 407-430, July 1996.

Muroga, S. "Logic Synthesizer by the Transduction Method"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

34

Logic Synthesizer by the Transduction Method

Saburo Muroga
University of Illinois at Urbana-Champaign

34.1 Technology-Dependent Logic Optimization

In this chapter, a logic optimization method called the transduction method, which was developed in the early 1970s, is described. Unlike the logic synthesizer in two phases described in Chapter 33, we can have logic synthesizers with totally technology-dependent optimization, based on the transduction method, and also can have transistor circuits with better quality, although it is more time-consuming to execute the method. Some underlying ideas in the method came from analyzing the minimal logic networks derived by the integer programming logic design method mentioned in Chapter 31, Section 31.5. The integer programming logic design method is very time-consuming, although it can design absolutely minimal networks. The processing time almost exponentially increases as the number of logic gates increases, and it appears to be excessively time consuming to design minimal logic networks with over about 10 logic gates. Unlike the integer programming logic design method, logic synthesizers based on the transduction method are heuristic, just like any other logic synthesizer for synthesizing large circuits, and the minimality of the transistor circuits derived is not guaranteed, but one can design circuits with far greater numbers of transistors.

34.2 Transduction Method for the Design of NOR Logic Networks

The transduction method can be applied on any types of gates, including negative gates (i.e., MOS logic gates), AND gates, OR gates, or logic gates for more complex logic operations; **but in the following, the transduction method is described with logic networks of NOR gates for the sake of simplicity**. The basic concept of the transduction method is "permissible functions," which will be explained later. This method, which is drastically different from any previously known logic design

method, is called the **transduction method** (**trans**formation and re**duction**) because it repeats the transformation and reduction as follows:

Procedure 34.1: Outline of the transduction method

Step 1. Design an initial NOR network by any known design method. Or, any network to be simplified can be used as an **initial network**.

Step 2. Remove the redundant part of the network, by use of permissible functions. (This is called the **pruning procedure**, as defined later).

Step 3. Perform **transformation** (which is local or global) of the current network, using permissible functions.

Step 4. Repeat Steps 2 and 3 until no further improvement is possible. □

Let us illustrate how Procedure 34.1 simplifies the initial network shown in Fig. 34.1(a), as an example. By a transformation (more precisely speaking, transformation called "connectable condition," which is explained later), we have the new network shown in Fig. 34.1(b), by connecting the output of gate v_6 to the gate v_8 (shown by a bold line). By the pruning procedure, the connection from gate v_6 to gate v_4 (shown by a dotted line) is deleted, deriving the new network shown in Fig. 34.1(c), and then the connection from gate v_9 to gate v_6 is deleted, deriving the new network shown in Fig. 34.1(d). Then, by a transformation ("connectable condition" again), we have the new network shown in Fig. 34.1(e), by connecting the output of gate v_6 to the gate v_7. By the pruning procedure, we can delete the output connection from gate v_{10} to gate v_7, and then we can delete gate v_{10}. During all these transformations and prunings, the network output f is not changed. Thus, the initial network with 7 gates and 13 connections, shown in Fig. 34.1(a), has been simplified to the one with 6 gates and 11 connections, shown in Fig. 34.1(f), by the transduction method.

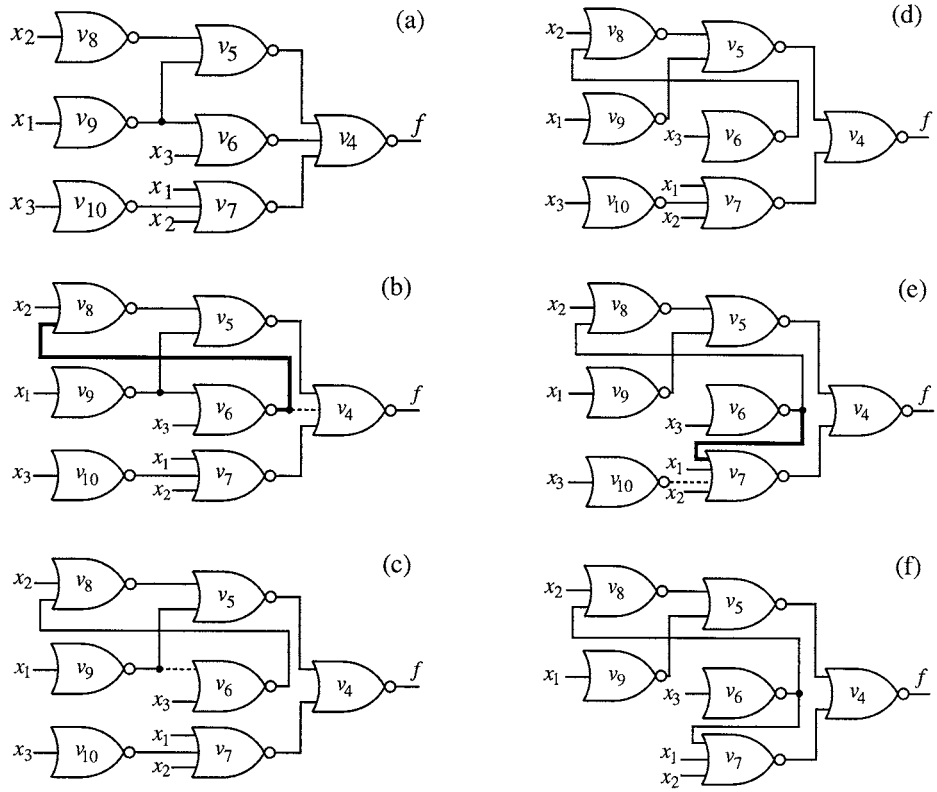


FIGURE 34.1 An example for the simplification of a logic network by the transduction method.

Many transformation procedures for Step 3 were devised to develop efficient versions of the transduction method. Reduction of a network is done in Step 2 and also sometimes in Step 3.

The initial network stated in Step 1 of Procedure 34.1 can be designed by any known method and need not be the best network. The transduction method is essentially an improvement of a given logic network, that is, the initial network by repeated transformation and reduction, as illustrated in the flow diagram in Fig. 34.2. In contrast, traditional logic design is the design of a logic network (i.e., the initial network in the case of the Transduction method) for given logic functions and once derived, no improvement is attempted. This was natural because it was not easy to find where to delete gates or connections in logic networks. For example, can we simplify the network in Fig. 34.1(a), when it is given? The transduction method shows how to do it.

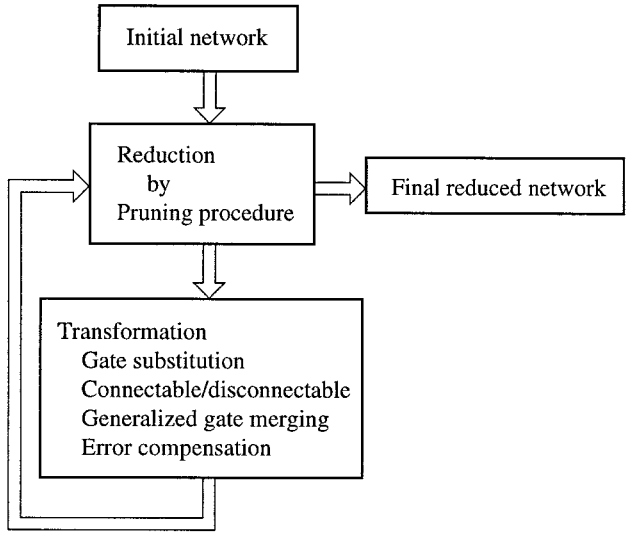


FIGURE 34.2 Basic structure of the transduction method.

The transduction method can reduce a network of many NOR gates into a new network with as few gates as possible. Also, for a given function, the transduction method can be used to generate networks of different configurations, among which we can choose those suitable for the most compact layouts on chips, since chip areas are greatly dependent on connection configurations of networks.

The transduction method was developed in the early 1970s with many reports and summarized in Refs. 9 and 10.

Permissible Functions

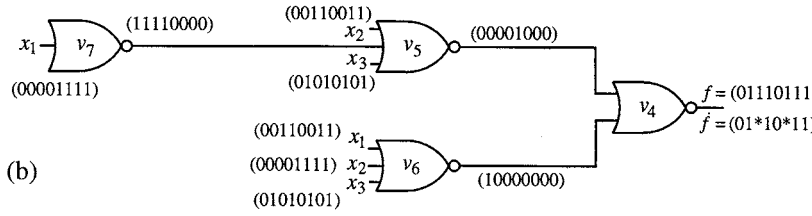
First, let us discuss the concept of permissible functions, which is the core concept for the transduction method.

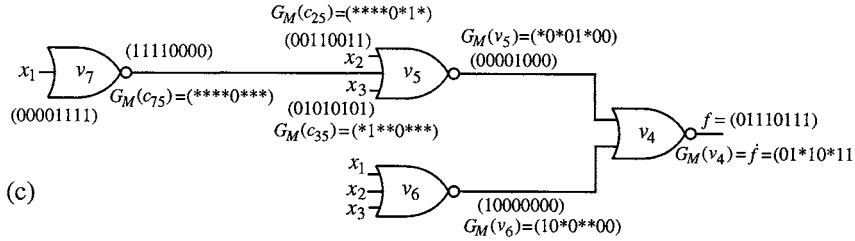
> **Definition 34.1**: If no output function of a network of NOR gates changes by replacing the function realized at a gate (or an input terminal) v_i, or a connection c_{ij}, with a function g, then function g is called a **permissible function** for that v_i, or c_{ij}, respectively. (Note that in this definition, changes in don't-care values, *, of the network output functions do not matter.)

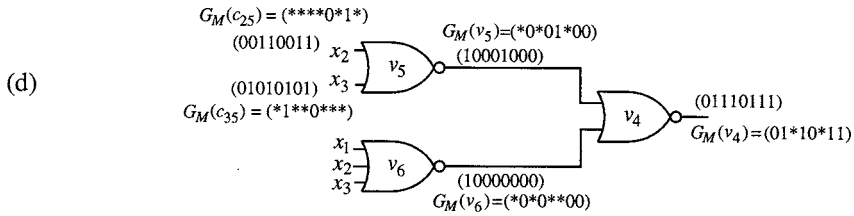
For example, we want to design a network for a function $\dot{f}(x_1, x_2, x_3)$ shown in the truth table in Fig. 34.3(a). Suppose the network in Fig. 34.3(b) is designed by some means to realize function $\dot{f}(x_1, x_2, x_3) = (01*10*11)$. The output function realized at each gate v_i in Fig. 34.3(b) is denoted by $f(v_i)$ in Fig. 34.3(a). The columns in Fig. 34.3(a) are shown horizontally as vectors in Fig. 34.3(b). For example, the

(a) Truth table

x_1	x_2	x_3	$f(v_7)$	$f(v_5)$	$f(v_6)$	$f(v_4)$	$\dot{f}$
0	0	0	1	0	1	0	0
0	0	1	1	0	0	1	1
0	1	0	1	0	0	1	*
0	1	1	1	0	0	1	1
1	0	0	0	1	0	0	0
1	0	1	0	0	0	1	*
1	1	0	0	0	0	1	1
1	1	1	0	0	0	1	1







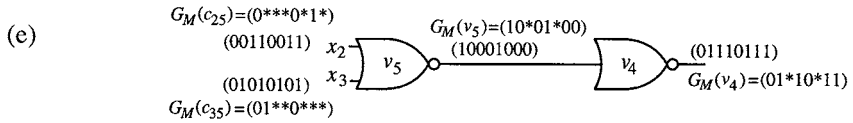


FIGURE 34.3 Permissible functions of a network.

output function of gate v_5, $f(v_5)$, is shown by vector (00001000) placed just outside gate v_5 in Fig. 34.3(b). (Henceforth, columns in a truth table will be shown by vectors in figures for logic networks.) Notice that the function $\dot{f}(x_1, x_2, x_3)$ = (01*10*11) (denoted by f with a dot on top of it) is a function to be realized by a network and generally is incompletely specified, containing don't cares (i.e., *'s), whereas the output function $f(v_4)$ (denoted by f without a dot on top of it) of gate v_4, for example, in the actually realized network in Fig. 34.3(b) is completely specified, containing no * because the value of the function $f(v_4)$ realized at the output of gate v_4 is 0 or 1 for each combination of values of input variables x_1 = (00001111), x_2 = (00110011), and x_3 = (01010101) (assuming x_i contains no *'s in its vector for each i).

Let us find permissible functions at the output of gate v_5. Because the first component of the output $\dot{f}$ of gate v_4 is 0, the first component of the output of gate v_6 is 1, and v_4 is a NOR gate, the first component

of a permissible function at the output of gate v_5 can be 0 or 1 (in other words, *). Because the second component of the output $\dot{f}$ of v_4 is 1 and v_4 is a NOR gate, the second component of a permissible function at the output of v_5 must be 0. Because the third component of $\dot{f}$ at v_4 is *, the third component of a permissible function at the output of gate v_5 can be 0 or 1 (i.e., *). Calculating every component in this manner, we will find that (00001000), (10001000), … are all permissible functions at the output of gate v_5, including the original vector (00001000) shown at v_5 in Fig. 34.3(b). In other words, even if we replace the function $f(v_5)$ realized at gate v_5 by any of these permissible functions, the network output at v_4 still realizes the given function $\dot{f}(x_1, x_2, x_3) = (01*10*11)$.

Notice that the value of the signal at a connection c_{ij} from gate v_i to v_j is always identical to the output value of v_i, but permissible functions for a connection c_{ij} are separately defined from those for a gate v_i. As we shall see later, this is important when gate v_i has more than one output connection. When v_i has only one output connection c_{ij}, permissible functions for v_i are identical to those for c_{ij} (the networks in Fig. 34.3 are such cases and the permissible functions for each c_{ij} are identical to those for v_i); but when v_i has more than one output connection, they are not necessarily identical, as we will see later.

Usually, there is more than one permissible function for each of v_i and c_{ij}. But when we find a set of permissible functions for a gate or connection, all the vectors representing these permissible functions can be expressed by one vector, as discussed in the following. This is convenient for processing.

For example, suppose the output function $f(v_i)$ of a gate v_i in a network has the values shown in the column labeled $f(v_i)$ in the truth table in Table 34.1, where the network has input variables $x_1, x_2, \ldots, x_n$. Let us write this column as a vector, $f(v_i) = (0011 \ldots)$. Suppose any outputs of the network do not change even if this function $f(v_i)$ in Table 34.1 is replaced by $g_1 = (0101 \ldots)$, $g_2 = (0001 \ldots)$, … , or $g_h = (0..1\ldots)$ shown in Table 34.1. In other words, $g_1, \ldots, g_h$ are permissible functions (they are not necessarily all the permissible functions at gate v_i). Then, permissible functions, g_1 through g_h, can be expressed by a single vector $G(v_i) = (0**1 \ldots)$ for the following reasons:

TABLE 34.1 Truth Table

x_1	…	x_{n-1}	x_n	$f(v_i)$	g_1	g_2	g_3	…	g_h	$G(v_i)$
0	…	0	0	0	0	0	0	…	0	0
0	…	0	1	0	1	0	1	…	…	*
0	…	1	0	1	0	0	1	…	…	*
0	…	1	1	1	1	1	1	…	1	1
…	…			…				…		…
…	…			…				…		…
…	…			…				…		…

1. Suppose permissible functions, g_1 and g_2, in Table 34.1 differ only in their second components. In other words, the second component is 1 in g_1 and 0 in g_2. Consequently, even if the second component of the output function of gate v_i is don't care (i.e., *), no network output will change. Thus, (0*01…) (i.e., the vector $g_1 = (0101\ldots)$ with the second component replaced by *) is another permissible function at gate v_i. This can be interpreted as follows. Original permissible functions, g_1 and g_2, can be replaced by the new permissible function, interpreting * to mean 0 or 1; in other words, (0*01…) means $g_1 = (0101\ldots)$ and $g_2 = (0001\ldots)$.
   When there is a permissible function g_j with * in the second component, any other permissible function, g_k, can be replaced by g_k itself with its second component replaced by *, even if other components of g_j may not be identical to those of g_k. For example, a permissible function, $g_3 = (0111\ldots)$, in Table 34.1 can be replaced by another permissible function (0*11…). This is because the value of each component of the output function is independent of any other component (in other words, the value of the output function at each gate for a combination of values of $x_1, x_2, \ldots, x_n$ is completely independent of those for other combinations). This means that if the second component of one permissible function, g_j, is *, then the second components of all other permissible functions can be *, where $1 \le j \le h$.

2. Suppose the first and fourth components are 0 and 1, respectively, in permissible functions, g_1 through g_h.
3. Then, the set of permissible functions, g_1, g_2, …, g_h, for gate v_i can be expressed as a single vector, $G(v_i) = (0{**}1\ldots)$, as shown in the last column in Table 34.1.

Pruning Procedure

From the basic properties of permissible functions just observed, we have the following.

Theorem 34.1: The set of all permissible functions at a gate (or an input terminal) or connection, or any subset of it, can be expressed by a single incompletely specified function.

Henceforth, let G denote the vector that this single incompletely specified function expresses. $G(v_i) = (0{**}1\ldots)$ in Table 34.1 is such an example.

When an arbitrary NOR gate network, such as Fig. 34.3(b), is given, it is generally very difficult to find which connections and/or gates are unnecessary (i.e., redundant) and can be deleted from the network without changing any output functions of the network. But, it is easy to do so by the following fundamental property based on the concept of permissible function:

If a set of permissible functions at a gate, a connection, or an input terminal in a given NOR gate network has no 1-component, in other words, consists of components, 0's, *'s, or a mixture of 0's and *'s, then that gate (and its output connections), connection, or input terminal, is redundant. In other words, it can be removed from the network without changing the network outputs, because the function on this gate, connection, or input terminal, can be the vector consisting of all 0-components (by changing every * to 0); and even if all the output connections of it are removed from the NOR gates whose inputs have these connections, the output functions of the network do not change. (This is because an input to a NOR gate that has more than one input is 0, and it will not affect the output of that NOR gate and hence is removable.)

Based on this property, we can simplify the network by the following procedure:

Procedure 34.2: Pruning procedure

We calculate a set of permissible functions for every gate, connection, and input terminal in a network of NOR gates, starting with the connections connected to output terminals of the network and moving toward the input terminals of the network. Whenever the single vector, G, that this set of permissible functions expresses, is found to consist of only 0 or * components, without 1-components during the calculation, we can delete that gate, connection, or input terminal without changing any network output. □

In this case, it is important to notice that we generally can prune only one of the gates, connections, and input terminals whose G's consist of only 0-components and *-components, as we will see later. If we want to prune more than one of them, we generally need to recalculate permissible functions of all gates, connections, and input terminals throughout the network after pruning one.

Calculation of Sets of Permissible Functions

Let us introduce simple terminology for later convenience. If there is a connection c_{ij} from the output of v_i to the input of v_j, v_i is called an **immediate predecessor** of v_j, and v_j is called an **immediate successor** of v_i. $IP(v_i)$ and $IS(v_i)$ denote the set of all immediate predecessors of v_i and the set of all immediate successors of v_i, respectively. If there is a path of gates and connections from the output of gate v_i to the input of v_j, then v_i is called a **predecessor** of v_j and v_j is called a **successor** of v_i. $P(v_i)$ and $S(v_i)$ denote the set of all predecessors of v_i and the set of all successors of v_i, respectively.

Thus far, we have considered only an arbitrary set of permissible functions at a gate, connection, or input terminal, v_i (or c_{ij}). Now let us consider the set of all permissible functions at v_i (or c_{ij}).

Definition 34.2: The set of all permissible functions at any gate, connection, or input terminal is called the **maximum set of permissible functions**, abbreviated as **MSPF**. This set can be expressed by a single vector, as already shown.

The MSPF of a gate, connection, or input terminal can be found by examining which functions change the network outputs. But MSPFs throughout the network can be far more efficiently calculated by the following procedure.

Procedure 34.3: Calculation of MSPFs in a network

MSPFs of every gate, connection, and input terminal in a network can be calculated, starting from the outputs of the network and moving toward the input terminals, as follows:

1. Suppose we want to realize a network that has input variables $x_1, x_2, \ldots, x_n$ and output functions $\dot{f}_1, \dot{f}_2, \ldots, \dot{f}_m$, that may contain don't-care components (i.e., $*$'s). Then, suppose we have actually realized a network with R NOR gates for these functions by some means and the output gates of this network realize $f_1, f_2, \ldots, f_m$, respectively. ($v_1, v_2, \ldots, v_n$ are regarded as input terminals where $x_1, x_2, \ldots, x_n$ are provided, respectively.) Notice that the output functions, $f_1, f_2, \ldots, f_m$, of this network do not contain any don't-care components because each network output function is 1 or 0 for each combination of values of $x_1, x_2, \ldots, x_n$. Calculate output function $f(v_j)$ at each gate v_j in this network.
   Then, the MSPF, $G_M(v_t)$, of each gate, v_t, whose output function is one of the network outputs, f_k, is set to $\dot{f}_k$ (not f_k, which is a completely specified function at v_t of the actually realized network); that is, $G_M(v_t) = \dot{f}_k$, where $k = 1, 2, \ldots, m$.
2. Suppose a gate v_j in the network has MSPF, $G_M(v_j)$. Then, the MSPF of each connection c_{ij} to the input of this gate can be calculated as follows:
   If a component of $G_M(v_j)$ is 0 and the disjunction (i.e., logical sum, or OR) of the values of the corresponding components of the input connections other than c_{ij} is 0, then the MSPF of input connection c_{ij} is 1. If they are 1 and 0, respectively, the MSPF of c_{ij} is 0. And so on. This operation, denoted by $\square$, is summarized in Table 34.2, where the first operand is a component of $G_M(v_j)$ and the second operand is the disjunction of the component values of the input connections other than c_{ij} and where "-" denotes "undefined" because v_j is a NOR gate and consequently these cases do not occur. Calculate each component of $G_M(v_j)$ in this manner.

TABLE 34.2 Definition of $\square$

	Second operand, $(\vee f(v))$ i.e., $v \in IP(v_j)$ $v \neq v_i$			
$\square$	0	1	*	
First operand, i.e., $G_M(v_j)$	0	1	*	–
	1	0	–	–
	*	*	*	–

– undefined

If this gate v_j has only one input connection c_{ij} (i.e., v_j becomes an inverter), the MSPF of c_{ij} is 1, 0, or $*$ according as $G_M(v_j)$ is 0, 1, or $*$.
This calculation can be formally stated by the following formula:

$$G_M(c_{ij}) = G_M(v_j) \square \left(\bigvee_{\substack{v \in IP(v_j) \\ v \neq v_i}} f(v) \right) \tag{34.1}$$

where if $IP(v_i) = \{v_i\}$ (in other words, gate v_j has only one input connection from v_i), then

$$\left(\bigvee_{\substack{v \in IP(v_j) \\ v \neq v_i}} f(v) \right)$$

is regarded as the function that is constantly 0.

For example, let us calculate MSPFs of the network shown in Fig. 34.3(b) which realizes the function $\dot{f}(x_1, x_2, x_3) = (01*10*11)$ given in Fig. 34.3(a). Because gate v_4 is the network output, this $\dot{f}$ is the MSPF of gate v_4 according to Step 1; that is, $G_M(v_4) = (01*10*11)$ as shown in Fig. 34.3(c). Gate v_4 has input connections realizing functions $f(c_{54}) = (00001000)$ and $f(c_{64}) = (10000000)$. Then, $G_M(c_{54})$, MSPF of c_{54}, can be calculated as follows.The first component of $G_M(c_{54})$ is $*$, using Table 34.2, because the first component of $G_M(v_4)$ is 0 and the first component of $f(c_{64})$ (which, in this case, is the disjunction of the first components of functions realized at connections other than c_{54} because c_{64} is the only connection other than c_{54}) is 1. Proceeding with the calculation based on Table 34.2, we have $G_M(c_{54}) = (*0*01*00)$. ($G_M(c_{54})$, and other $G_M(c_{ij})$'s are not shown in Fig. 34.3.)

3. The MSPF of a gate or input terminal, v_i, can be calculated from $G_M(c_{ij})$ as follows:
If a gate or input terminal, v_i, has only one output connection, c_{ij}, whose MSPF is $G_M(c_{ij})$, then $G_M(v_i)$, MSPF of v_i, is given by:

$$G_M(v_i) = G_M(c_{ij}) \tag{34.2}$$

Thus, we have $G_M(v_5) = G_M(c_{54})$ and this is shown in Fig. 34.3(c).
If v_i has more than one output connection to gates v_j's, then $G_M(v_i)$ is not necessarily identical to $G_M(c_{ij})$. In this case, the MSPF for any gate or input terminal, v_i, in the network is given by the following H:

$$H = (H^{(1)}, H^{(2)}, \ldots, H^{(2^n)}) \tag{34.3}$$

whose w-th component is given by:
$H^{(w)} = *$ if $f_k^{(w)} \supseteq f_k'^{(w)}$ for every k such that $1 \le k \le m$, and
$H^{(w)} = f^{(w)}(v_i)$ if $f_k^{(w)} \supseteq f_k'^{(w)}$ does not hold for some k such that $1 \le k \le m$,
where $f_k' = (f_k'^{(1)}, f_k'^{(2)}, \ldots, f_k'^{(2^n)})$ is the new k-th output function of the network, to which the k-th output of the original network, f_k, changes by the following reconfiguration: insert an inverter between gate or input terminal, v_i, and its immediate successor gates, v_j's (so the immediate successor gates receive $\bar{f}(v_i)$ instead of the original function $f(v_i)$ at v_i). Here, "$\supseteq$," which means $\supset$ or $=$, is used as an ordinary set inclusion, i.e., $* \supseteq 0$, $* \supseteq 1$, $1 \supseteq 1$, and $0 \supseteq 0$, interpreting $*$ as the set of 1 and 0. $f^{(w)}(v_i)$ is the w-th component of $f(v_i)$. Notice that the calculation of the MSPF for v_i, based on Eq. 34.3, is done by finding out the new values of the network outputs, $f_1, f_2, \ldots, f_m$, for the preceding reconfiguration, without using $G_M(c_{ij})$'s. Thus, this calculation is complex and time-consuming.

4. Repeat Steps 2 and 3 until finishing the calculation of MSPFs throughout the network.
Let us continue the example in Fig. 34.3(c), where input variables, x_1, x_2, and x_3, are at input terminals v_1, v_2, and v_3, respectively, and then for the input variables, $x_1 = f(v_1) = (0\,0\,0\,0\,1\,1\,1\,1)$, $x_2 = f(v_2) = (0\,0\,1\,1\,0\,0\,1\,1)$, and $x_3 = f(v_3) = (0\,1\,0\,1\,0\,1\,0\,1)$, the outputs of gates are $f(v_4) = (0\,1\,1\,1\,0\,1\,1\,1)$, $f(v_5) = (0\,0\,0\,0\,1\,0\,0\,0)$, $f(v_6) = (1\,0\,0\,0\,0\,0\,0\,0)$, and $f(v_7) = (1\,1\,1\,1\,0\,0\,0\,0)$. Because gate v_5 has only one output connection, $G_M(v_5) = G_M(c_{54})$, according to Step 3. The first component of $G_M(c_{25})$ (i.e., the MSPF of connection, c_{25}) from input terminal v_2 (which actually has input variable x_2), is $*$ by Table 34.2 because the first component of $G_M(v_5)$ is $*$. The second component of $G_M(v_2)$ is $*$ by Table 34.2 because the second component of $G_M(v_5)$ is 0 and the disjunction of the second components of $f(c_{75})$ and $f(c_{35})$ is $1 \vee 1 = 1$. And so on. □

Derivation of an Irredundant Network Using the MSPF

The following procedure provides a quick means to identify redundant gates, connections, or input terminals in a network. By repeating the calculation of MSPF at all gates, connections, and input terminals throughout the network and the deletion of some of them by the pruning procedure (Procedure 34.2), we can derive an irredundant logic network as follows, where "**irredundant network**" means that if any connection, gate, or input terminal is deleted, some network outputs will change; in other words, every connection, gate, or input terminal is not redundant.

Procedure 34.4: Derivation of an irredundant network using the MSPF

1. Calculate the MSPFs for all gates, connections, and input terminals throughout the network by Procedure 34.3, starting from each output terminal and moving toward input terminals.
2. During Step 1, we can remove a connection, gate, or input terminal, without changing any output function of the network, as follows:
   a. If there exists any input connection of a gate whose MSPF consists of 0's and $*$'s only, remove it. We can remove only one connection because if two or more connections are simultaneously removed, some network outputs may change.
   b. As a result of removing any connection, there may be a gate without any output connections. Then remove such a gate.
3. If a connection and possibly a gate are removed in Step 2, return to Step 1 with the new network. Otherwise, go to Step 4.
4. Terminate the procedure and we have obtained an irredundant network. □

This procedure does not necessarily yield a network with a minimum number of gates or connections, because an irredundant network does not necessarily have a minimum number of gates or connections. But the procedure is useful in many cases because every network with a minimum number of gates or connections must be irredundant and an obtained network may be minimal.

Example 34.1: Let us apply Procedure 34.4 to the network shown in Fig. 34.3(c). Because each gate has only one output connection, $G_M(v_i) = G_M(c_{ij})$ holds for every v_i. Thus, using Eq. 34.1 and $G_M(v_4) = \dot{f}(x_1, x_2, x_3)$, MSPFs are obtained as follows:

$$G_M(c_{54}) = G_M(v_5) = G_M(v_4) \square f(v_6) = (* \ 0 \ * \ 0 \ 1 \ * \ 0 \ 0),$$

$$G_M(c_{64}) = G_M(v_6) = G_M(v_4) \square f(v_5) = (1 \ 0 \ * \ 0 \ * \ * \ 0 \ 0), \text{ and}$$

$$G_M(c_{75}) = G_M(v_7) = G_M(v_5) \square (x_2 \vee x_3) = (* \ * \ * \ * \ 0 \ * \ * \ *)$$

Then we can remove connection c_{75} and consequently gate v_7 by the pruning procedure, as shown in Fig. 34.3(d). For this new network, we obtain the following:

$$f(v_4) = (0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 1 \ 1), f(v_5) = (1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0),$$

$$f(v_6) = (1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0),$$

$$G_M(v_4) = \dot{f}(x_1, x_2, x_3) = (0 \ 1 \ * \ 1 \ 0 \ * \ 1 \ 1),$$

$$G_M(c_{54}) = G_M(v_5) = (* \ 0 \ * \ 0 \ 1 \ * \ 0 \ 0), \text{ and}$$

$$G_M(c_{64}) = G_M(v_6) = G_M(v_4) \square f(v_5) = (* \ 0 \ * \ 0 \ * \ * \ 0 \ 0)$$

Then we can remove connection c_{64} and then gate v_6. Thus, the original network of four gates and nine connections is reduced to the network of two gates with three connections in Fig. 34.3(e), where G_M's are recalculated and no gate can be removed by the pruning procedure. □

Whenever we delete any one connection, gate, or input terminal by the pruning procedure, we need to calculate MSPFs of the new network from scratch. If we cannot apply the pruning procedure, the final logic network is irredundant. This means that if any gate, connection, or input terminal is deleted from the network, some of the network outputs change. It is important to notice that an irredundant network is completely testable; that is, we can detect, by providing appropriate values to the input terminals, whether or not the network contains faulty gates or connections. (A redundant network may contain gates or connections such that it is not possible to detect whether they are faulty.)

Use of MSPFs is time-consuming because the calculation of MSPFs by Eq. 34.3 in Step 3 of Procedure 34.3 is time-consuming and also because we must recalculate MSPFs whenever we delete a connection, gate, or input terminal by the pruning procedure.

Calculation of Compatible Sets of Permissible Functions

Reduction of calculation time is very important for developing a practical procedure. For this purpose, we introduce the concept of a compatible set of permissible functions which is a subset of an MSPF.

Definition 34.3: Let V be the set of all the gates and connections in a network, and e be a gate (or input terminal) v_i or a connection c_{ij} in this set. All the sets of permissible functions for all e's in V, denoted by $G_C(e)$'s, are called **compatible sets of permissible functions** (abbreviated as **CSPFs**), if the following properties hold with every subset T of V:

a. Replacement of the function at each gate or connection, t, in set T by a function $f(t) \in G_C(t)$ results in a new network where each gate or connection, e, such that $e \notin T$ and $e \in V$, realizes function $f(e) \in G_C(e)$.
b. The condition a holds, no matter which function $f(t)$ in $G_C(t)$ is chosen for each t. □

Consequently, even if the function of a gate or connection is replaced by $f(v_t) \in G_C(v_t)$, the function $f(v_u)$ at any other gate or connection still belongs to the original $G_c(v_u)$.

CSPFs at all gates and connections in a network can be calculated as follows.

Procedure 34.5: Calculation of CSPFs in a logic network

CSPFs of all gates, connections, and input terminals in a network can be calculated, starting from the network outputs and moving toward the input terminals, as follows:

1. Suppose we want to realize a network that has inputs $x_1, x_2 \ldots x_n$ and output functions $\dot{f}_1, \dot{f}_2, \ldots, \dot{f}_m$, which may contain don't-care components (i.e., *'s). Then, suppose we have actually realized a network with R NOR gates for these functions by some means.
   Calculate output function $f(v_j)$ at each gate v_j in this network.
   Then, CSPF of each gate, v_t, whose output function is one of the network outputs, f_k, is $G_C(v_t) = \dot{f}_k$ (not f_k), where $k = 1, 2, \ldots, m$.
2. Unlike MSPFs, the CSPF at each gate, connection, or input terminal is highly dependent on the order of processing gates, connections, and input terminals in a network. So, before starting the calculation of CSPFs, we need to determine an appropriate order of processing. Selection of a good processing **order**, denoted by r, is important for the development of an efficient transduction method. Let $r(v_i)$ denote the ordinal number of gates or connections, v_i in this order.
   Suppose a gate v_j in the network has CSPF, $G_C(v_j)$. Then, CSPF of each input connection c_{ij}, $G_C(c_{ij})$, of this gate can be calculated by the following formula which is somewhat different from Eq. 34.1:

$$G_C(c_{ij}) = \{G_C(v_j) \square (\underset{v \in IP(v_j)}{\underset{r(v) > r(v_i)}{\vee}} f(v))\} \# f(v_i) \tag{34.4}$$

by calculating the terms on the right-hand side in the following steps:

a. Calculate the disjunction (i.e., component-wise disjunction) of all the functions $f(v_i)$'s of immediate predecessor gates, v_i's, of the gate v_j, whose ordinal numbers $r(v_i)$'s are greater than the ordinal number $r(v_i)$ of the gate v_i. If there is no such gate v_i, the disjunction is 0.
b. Calculate the expression inside { } in Eq. 34.4 by using Table 34.2 (the definition of □). In this table, $G_C(v_j)$ is used as the first operand, and

$$\left(\bigvee_{\substack{v \in IP(v_j) \\ r(v) > r(v_i)}} f(v) \right)$$

(i.e., the disjunction calculated in Step a) is used as the second operand.
c. Calculate the right-hand side of Eq. 34.4 using Table 34.3, with the value calculated in Step b as the first operand and $f(v_i)$ as the second operand.

TABLE 34.3 Definition of #

First operand, i.e., $G_C(v_j) \square \left(\bigvee_{\substack{v \in IP(v_j) \\ r(v) > r(v_i)}} f(v) \right)$	Second operand, i.e., $f(v_i)$			
	#	0	1	*
	0	0	*	*
	1	*	1	*
	*	*	*	*

For example, suppose gate v_j in Fig. 34.4 has CSPF, $G_C(v_j) = (010^*)$, and input connections c_{ij}, c_{gj}, and c_{hj}, from gates v_i, v_g, and v_h, whose functions are $f(v_i) = (1001)$, $f(v_g) = (0011)$, and $f(v_h) = (0010)$. (Notice that $f(v_a) = f(c_{ab})$ always holds for any gate or input terminal, v_a, no matter whether v_a has more than one output connection.) Suppose we choose an order, $r(v_i) < r(v_g) < r(v_h)$. Then the first component of $G_C(c_{ij})$ is $\{ 0 \square (0 \vee 0) \}\#1$ because the first components of $G_C(v_j)$, $f(v_g)$, $f(v_h)$, and $f(v_i)$, which appear in this order in Eq. 34.4, are 0, 0, 0, and 1, respectively. Using Tables 34.2 and 34.3, this becomes $\{ 0 \square (0 \vee 0) \}\#1 = \{ 0 \square 0 \}\#1 = 1\#1 = 1$. Calculating other components similarly, we have $G_C(c_{ij}) = (10^{**})$. We have $G_C(c_{gj}) = (^*0^{**})$ because the fourth component of $G_C(c_{gj})$, for example, is $\{^* \square 0\}\#1$ since the fourth components of $G_C(v_j)$, $f(v_h)$, and $f(v_g)$ are *, 0, and 1, respectively. In this case, the value of $f(v_i)$ is not considered unlike the calculation of MSPF, $G_M(c_{gj})$, because $f(v_i)$ is not included in Eq. 34.4 due to the order, $r(v_i) < r(v_g)$. Also, we have $G_C(c_{hj}) = (^*01^*)$ because the fourth component of $G_C(c_{hj})$, for example, becomes $\{^* \square 0\}\#0$ since the fourth components of $G_C(v_j)$,

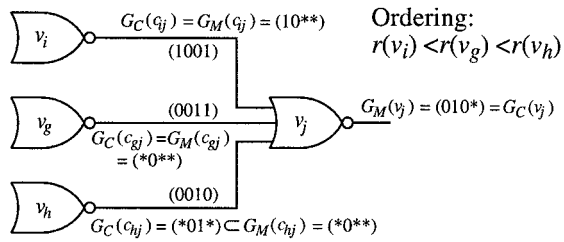


FIGURE 34.4 An example of the calculation of MSPFs and CSPFs.

$$\left(\bigvee_{v \in IP(v_j),\ r(v) > r(v_i)} f(v) \right),$$

and $f(v_h)$ are $*$, 0 (because no gate v such that $r(v) > r(v_h)$), and 0, respectively. In this case, $f(v_i)$ and $f(v_g)$ are not considered.

For comparison, let us also calculate MSPFs by Procedure 34.3. The MSPFs of connections c_{ij}, c_{gj}, and c_{hj} can be easily calculated as $G_M(c_{ij}) = (10^{**})$, $G_M(c_{gj}) = (^*0^{**})$, and $G_M(c_{hj}) = (^*0^{**})$, respectively, as shown in Fig. 34.4. Comparing with the CSPFs, we can find $G_C(c_{ij}) = G_M(c_{ij})$ and $G_C(c_{gj}) = G_M(c_{gj})$. But $G_C(c_{hj})$ is a subset of $G_M(c_{hj})$ (denoted as $G_C(c_{hj}) \subset G_M(c_{hj})$) because the third component of $G_C(c_{hj}) = (^*01^*)$ is 1, which is a subset of the third component, * (i.e., 0 or 1), of $G_M(c_{hj}) = (^*0^{**})$, while other components are identical.

The CSPF and MSPF of a gate, connection, or input terminal can be identical. For the gate with $G_C(v_j) = G_M(v_j) = (010^*)$ shown in Fig. 34.5, for example, we have $G_C(c_{ij}) = G_M(c_{ij}) = (10^{**})$, $G_C(c_{gj}) = G_M(c_{gj}) = (^*0^{**})$, and $G_C(c_{hj}) = G_M(c_{hj}) = (^*01^*)$ with the ordering $r(v_i) < r(v_g) < r(v_h)$.

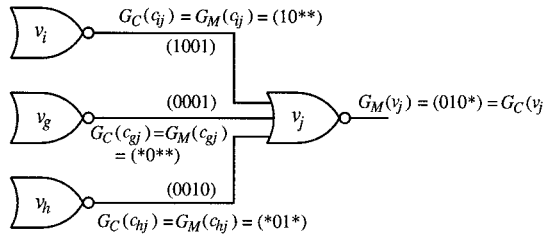


FIGURE 34.5 An example of the calculation of MSPFs and CSPFs.

As can be seen in the third components of G_M's in the example in Fig. 34.4, when gate v_j in a network has more than one input connection whose w-th component is 1 and we have $G_M^{(w)}(v_j) = 0$, the w-th components of MSPFs for all these input connections are $*$'s, as seen in the third components of $G_M(c_{ij})$, $G_M(c_{gj})$, and $G_M(c_{hj})$ in the example in Fig. 34.4. But the w-th components of CSPFs, however, are *'s except for one input connection whose value is required to be 1, as seen in the third components of $G_C(c_{hj})$ in the example in Fig. 34.4. Which input connection is such an input connection depends upon order r. Intuitively, an input connection to the gate v_j from an immediate predecessor gate that has a smaller ordinal number in order r will probably tend to have more *'s in its CSPF and, consequently, have a greater probability for this input connection to be removed.

3. The CSPF of a gate or input terminal, v_i, can be calculated from $G_C(c_{ij})$ as follows.
   If a gate or input terminal, v_i, has only one output connection, c_{ij}, whose CSPF is $G_C(c_{ij})$, then $G_C(v_i)$, CSPF of v_i, is given by

$$G_C(v_i) = G_C(c_{ij}) \tag{34.5}$$

If v_i has more than one output connection, then $G_C(v_i)$ is not necessarily identical to $G_C(c_{ij})$. In this case, the CSPF for any gate or input terminal, v_i, in a network is given by the following:

$$G_C(v_i) = \bigcap_{v_j \in IS(v_i)} G_C(c_{ij}) \tag{34.6}$$

where the right-hand side of Eq. 34.6 is the intersection of $G_C(c_{ij})$'s of output connections, c_{ij}'s, of gate v_i.

Unlike Eq. 34.3 for the case of MSPFs, Eq. 34.6 is simple and can be calculated in a short time.

4. Repeat Steps 2 and 3 until the calculation of CSPFs throughout the network is finished. □

A gate or connection may have different CSPFs if the order r of processing is changed. On the other hand, each gate or connection has a unique MSPF, independent of the order of processing.

Comparison of MSPF and CSPF

It is important to notice the difference in the ways of defining MSPF and CSPF. Any function $f(v_i)$ (or $f(c_{ij})$), that belongs to the MSPF of a gate, connection, or input terminal, v_i (or c_{ij}), can replace the original function realized at this v_i (or c_{ij}) without changing any network output, keeping the functions at all other gates, connections, or input terminals intact. If functions at more than one gate, connection, and/or input terminal are simultaneously replaced by permissible functions in their respective MSPFs, some network outputs may change. In the case of CSPF, simultaneous replacement of the functions at any number of gates, connections, and input terminals by permissible functions in their respective CSPFs does not change any network output.

Example 34.2: This example illustrates that if functions realized at more than one gate, connection, or input terminal are simultaneously replaced by permissible functions in their respective MSPFs, some network outputs may change, whereas simultaneous replacement by permissible functions in their respective CSPFs does not change any network outputs. Let us consider the network in Fig. 34.6(a) where all the gates have the same MSPFs as those in Fig. 34.4. In Fig. 34.6(a), let us simultaneously replace functions (1001), (0011), and (0010) realized at the inputs of gate v_j in Fig. 34.4 by (1000), (0001), and (1001), respectively, such that $(1000) \in G_M(c_{ij}) = (10^{**})$, $(0001) \in G_M(c_{gj}) = (*0^{**})$, and $(1001) \in G_M(c_{hj}) = (*0^{**})$ hold. Then the output function of gate v_j in Fig. 34.6(a) becomes (0110). But we have $(0110) \notin G_M(v_j)$ because the third component 1 is different from the third component 0 of $G_M(v_j)$. So, (0110) is not a permissible function in MSPF, $G_M(v_j)$. But if we replace the function at only one input to gate v_j by a permissible function of that input, the output function of v_j is still a permissible function in MSPF, $G_M(v_j)$. For example, if only (0011) at the second input of gate v_j in Fig. 34.4 is replaced by (0001), the output function of v_j becomes (0100), which is still a permissible function of $G_M(v_j)$, as shown in Fig. 34.6(b).

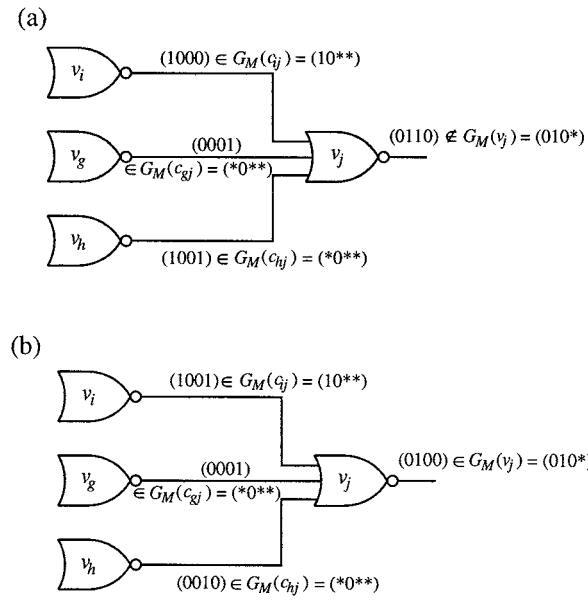


FIGURE 34.6 MSPFs.

If we use CSPF, we can replace more than one function. For example, let us consider the network in Fig. 34.7, where gate v_j has $G_C(v_j)$, the same as $G_C(v_j)$ in Fig. 34.4. The functions at the inputs of gate v_j in Fig. 34.7 belong to CSPFs calculated in Fig. 34.4; in other words, $(1000) \in G_C(c_{ij}) = (10{**})$, $(0001) \in G_C(c_{gj}) = (*0{**})$, and $(1010) \in G_C(c_{hj}) = (*01*)$. Even if all functions (1001), (0011), and (0010) in Fig. 34.4 are simultaneously replaced by these functions, function (0100) realized at the output of gate v_j is still a permissible function in $G_C(v_j)$. □

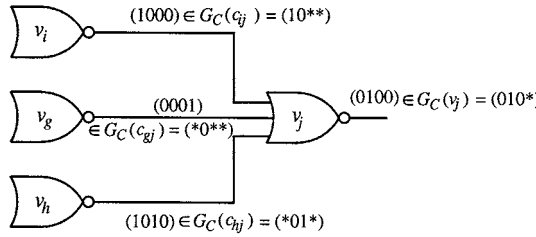


FIGURE 34.7 CSPFs.

Procedures based on CSPFs have the following advantages and disadvantages:

1. For the calculation of CSPFs, we need not use Eq. 34.3 which is time-consuming.
2. Even if a redundant connection is removed, we need not recalculate CSPFs for the new network. In other words, CSPFs at different locations in the network are independent of one another, whereas MSPFs at these locations may not be. Thus, using CSPFs, we can simultaneously remove more than one connection; whereas using MSPFs, we need to recalculate MSPFs throughout the network, whenever one connection is removed.
   If, however, we use CSPFs instead of MSPFs, we may not be able to remove some redundant connections by the pruning procedure because each CSPF is a subset of its respective MSPF and depends on processing order *r*. Because gates with smaller ordinal number in order, *r*, tend to have more ∗-components, the probabilities of removing these gates (or their output connections) are greater. Thus, if a gate is known to be irredundant, or hard to remove, we can assign a larger ordinal number in order *r* to this gate and this will help giving ∗-components to the CSPFs of other gates.
3. The property (2) is useful for developing network transformation procedures based on CSPFs, which will be discussed later.
4. Because each CSPF is a subset of a MSPF, the network obtained by the use of CSPFs is not necessarily irredundant. But if we use MSPFs for pruning after repeated pruning based on CSPFs, then the final network will be irredundant.

For these reasons, there is a tradeoff between the processing time and the effectiveness of procedures.

Transformations

We can delete redundant connections and gates from a network by repeatedly applying the pruning procedure (in other words, by repeating only Step 2, without using Step 3, in Procedure 34.1, the outline of the transduction method). In this case, if MSPF is used, as described in Procedure 34.4, the network that results is irredundant. However, to have greater reduction capability, we developed several transformations of a network. By alternatively repeating the pruning procedure (Step 2 in Procedure 34.1) and transformations (Step 3), we can reduce networks far more than by the use of only one of them.

The following gate substitution procedure is one of these transformations.

Procedure 34.6: Gate substitution procedure

If there exist two gates (or input terminals), v_i and v_j, satisfying the following conditions, all the output connections of v_j can be replaced by the output connections of v_i without changing network outputs. Thus, v_j is removable.

1. $f(v_i) \in G(v_j)$, where $G(v_j)$ is a set of permissible functions of v_j which can be an MSPF or a CSPF.
2. v_i is not a successor of v_j (no loop will be formed by this transformation). □

This is illustrated in Fig. 34.8. The use of MSPFs may give a better chance for a removal than their subsets such as CSPFs, although the calculation of MSPFs is normally time-consuming.

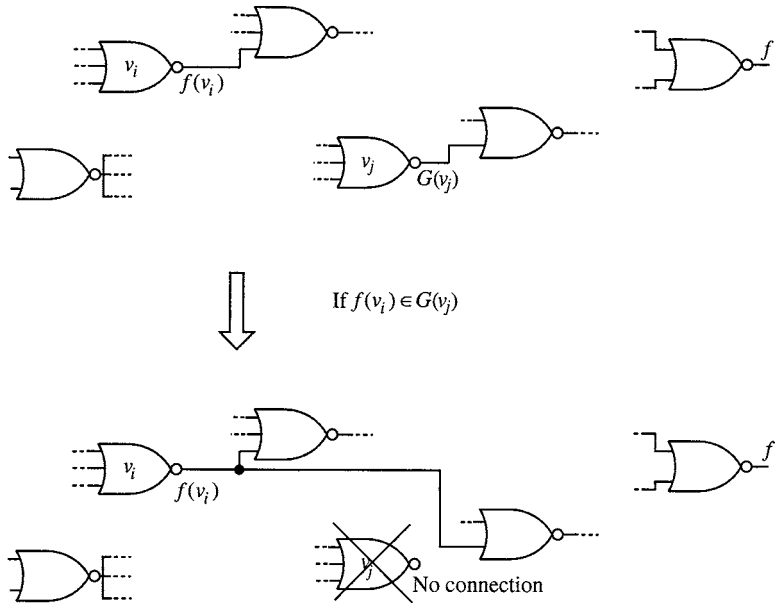


FIGURE 34.8 Gate substitution.

Example 24.3: Let us apply Procedure 34.6 to the network shown in Fig. 34.9(a), which realizes the function $f = \bar{x}_1 x_2 \vee \bar{x}_1 x_3 \vee x_2 x_3$. Functions realized at input terminals and gates are as follows:

$$x_1 = f(v_1) = (0\ 0\ 0\ 0\ 1\ 1\ 1\ 1),\ x_2 = f(v_2) = (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1),$$
$$x_3 = f(v_3) = (0\ 1\ 0\ 1\ 0\ 1\ 0\ 1),$$
$$f(v_4) = (0\ 1\ 1\ 1\ 0\ 0\ 0\ 1),\ f(v_5) = (0\ 0\ 0\ 0\ 1\ 0\ 1\ 0),$$
$$f(v_6) = (1\ 0\ 0\ 0\ 1\ 1\ 0\ 0),\ f(v_7) = (1\ 1\ 1\ 1\ 0\ 0\ 0\ 0),$$
$$f(v_8) = (0\ 1\ 1\ 1\ 0\ 0\ 0\ 0),\ \text{and}\ f(v_9) = (1\ 0\ 0\ 0\ 1\ 0\ 0\ 0)$$

Let us consider the following two different approaches.

1. Transformation by CSPFs: CSPFs for the gates are calculated as follows if at each gate, the input in the lower position has higher processing order than the input in the upper position:

$$G_C(v_4) = (0\ 1\ 1\ 1\ 0\ 0\ 0\ 1),\ G_C(v_5) = (*\ 0\ 0\ 0\ *\ *\ 1\ 0),$$
$$G_C(v_6) = (1\ 0\ 0\ 0\ 1\ 1\ *\ 0),\ G_C(v_7) = (*\ *\ 1\ *\ *\ *\ 0\ *),$$
$$G_C(v_8) = (0\ 1\ *\ *\ 0\ 0\ *\ *),\ \text{and}\ G_C(v_9) = (1\ 0\ *\ *\ *\ *\ *\ *)$$

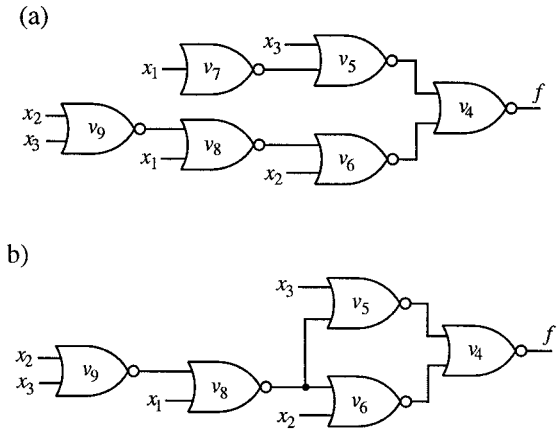


FIGURE 34.9 An example of gate substitution.

Because $f(v_8) \in G_C(v_7)$, the network in Fig. 34.9(b) is obtained by substituting connection c_{86} for c_{75}. Then, gate v_7 is removed, yielding a simpler network.

2. Transformation by MSPFs: In this case, the calculation of MSPFs is very easy because each gate in Fig. 34.9(a) has only one output connection. MSPFs for gates are as follows:

$$G_M(v_4) = (0\ 1\ 1\ 1\ 0\ 0\ 0\ 1),\ G_M(v_5) = (*\ 0\ 0\ 0\ *\ *\ 1\ 0),$$
$$G_M(v_6) = (1\ 0\ 0\ 0\ *\ 1\ *\ 0),\ G_M(v_7) = (*\ *\ 1\ *\ *\ *\ 0\ *),$$
$$G_M(v_8) = (0\ 1\ *\ *\ *\ 0\ *\ *),\ \text{and}\ G_M(v_9) = (1\ 0\ *\ *\ *\ *\ *\ *)$$

Here, $G_M(v_7) = G_C(v_7)$, and we get the same result in Fig. 34.9(b).

This result cannot be obtained by the gate merging to be discussed later. □

This gate substitution can be further generalized. In other words, a gate, v_j, can be substituted by more than one gate, instead of by only one gate v_i in Procedure 34.6.

If we connect a new input to a gate or disconnect an existing input from a gate, the output of the gate may be modified. But if the new output is still contained in the set of permissible functions at this gate, the modification does not change the network outputs. By applying this procedure, we can change the network configuration and possibly can remove connections and/or gates. Even if we cannot reduce the network, a modification of the network is useful for further applications of other transformations. We have such a procedure if the connectable condition stated in the following or the disconnectable condition stated after the following is satisfied.

Procedure 34.7: Connectable condition

Let $G(v_j)$ be a set of permissible functions for gate v_j which can be an MSPF or a CSPF. We can add a connection from input terminal or gate, v_i, to v_j without changing network outputs, if the following conditions are satisfied:

1. $f^{(w)}(v_i) = 0$ for all w's such that $G^{(w)}(v_j) = 1$.
2. v_i is not a successor of v_j (no loop will be formed by this transformation).

This is illustrated in Fig. 34.10.

This transformation procedure based on the connectable condition can be extended into the forms that can be used to change the configuration of a network. When we cannot apply any transformations to a given network, we may be able to apply those transformations after these extended transformations.

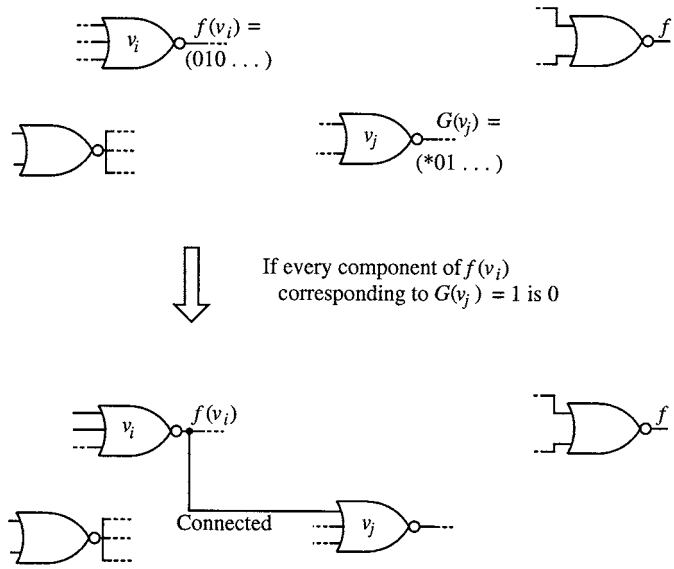


FIGURE 34.10 Connectable condition.

Example 34.4: If we add two connections to the network in Fig. 34.11(a), as shown in bold lines in Fig. 34.11(b), then the output connection (shown in a dotted line) of gate v_{12} becomes disconnectable and v_{12} can be removed. Then we have the network shown in Fig. 34.11(c). □

Procedure 34.8: Disconnectable condition

If we can find a set of inputs of gate v_k such that the disjunction of the w-th component of the remaining inputs of v_k is 1 for every w satisfying $G^{(w)}(v_k) = 0$, then this set of inputs can be deleted, as being redundant, without changing network outputs. □

Procedures 34.7 and 34.8 will be collectively referred as the **connectable/disconnectable conditions** (or **procedures**).

In the network in Fig. 34.12(a), x_2 is connectable to gate v_6, and x_1 is connectable to gate v_7. After adding these two connections, the outputs of v_6 and v_7 become identical, so v_7 can be removed as shown in Fig. 34.12(b). This transformation is called **gate merging**. This can be generalized, based on the concept of permissible functions, as follows.

Procedure 34.9: Generalized gate merging

1. Find two gates, v_i and v_j, such that the intersection, $G_C(v_i)G_C(v_j)$, of their CSPFs is not empty, as illustrated in Fig. 34.13.
2. Consider an imaginary gate, v_{ij}, whose CSPF is to be $G_C(v_i)G_C(v_j)$.
3. Connect all the inputs of gate v_i and v_j to v_{ij}. If v_{ij} actually realizes a function in $G_C(v_i)G_C(v_j)$, then v_{ij} can be regarded as a merged gate of v_i and v_j. Otherwise, v_i and v_j cannot be merged without changing network outputs in this generalized sense.
4. If v_{ij} can replace both v_i and v_j, then remove redundant inputs of v_{ij}. □

Next, let us outline another transformation, called the **error compensation procedure**. In order to enhance the gate removal capability of the transduction method, the concept of permissible function is generalized to "a permissible function with errors."[5] Because the transformation procedures based on the error compensation are rather complicated, we outline the basic idea of these procedures along with an example, as follows:

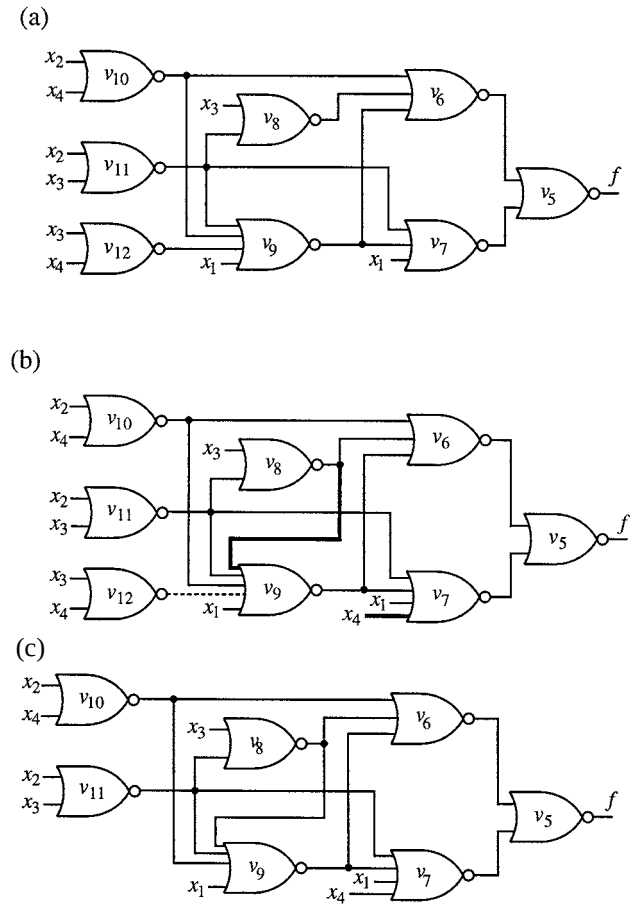


FIGURE 34.11 An example of the connectable/disconnectable conditions.

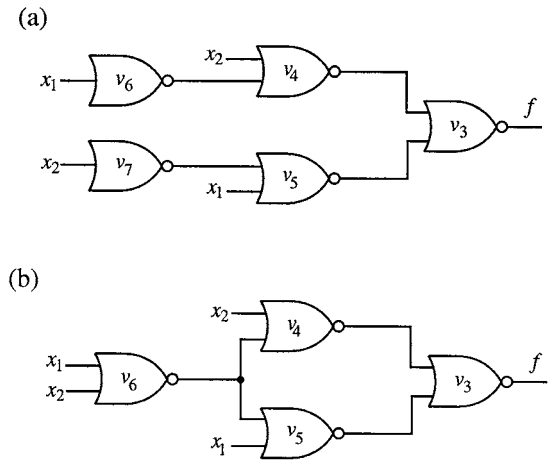


FIGURE 34.12 An example of gate merging.

1. Remove an appropriate gate or connection from the network.

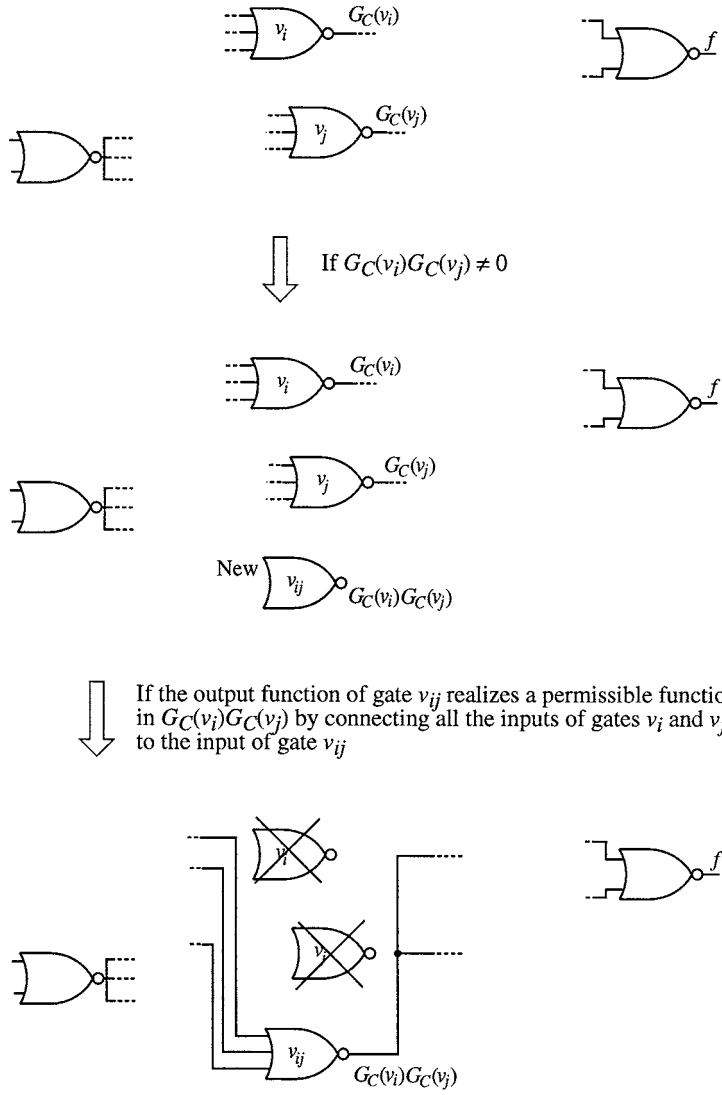


FIGURE 34.13 Generalized gate merging.

2. Calculate errors in components of functions at gates or connections that are caused by the removal of the gate or connection, and then calculate permissible functions with errors throughout the network. These permissible functions with errors represent functions with erroneous components (i.e., components whose values are erroneous) as well as ordinary permissible functions that have no erroneous components.
3. Try to compensate for the errors by changing the connection configuration of the network. In order to handle the errors, the procedures based on ordinary permissible functions are modified.

Example 34.5: Figure 34.14(a) shows a network whose output at gate v_5 realizes

$$(1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1)$$

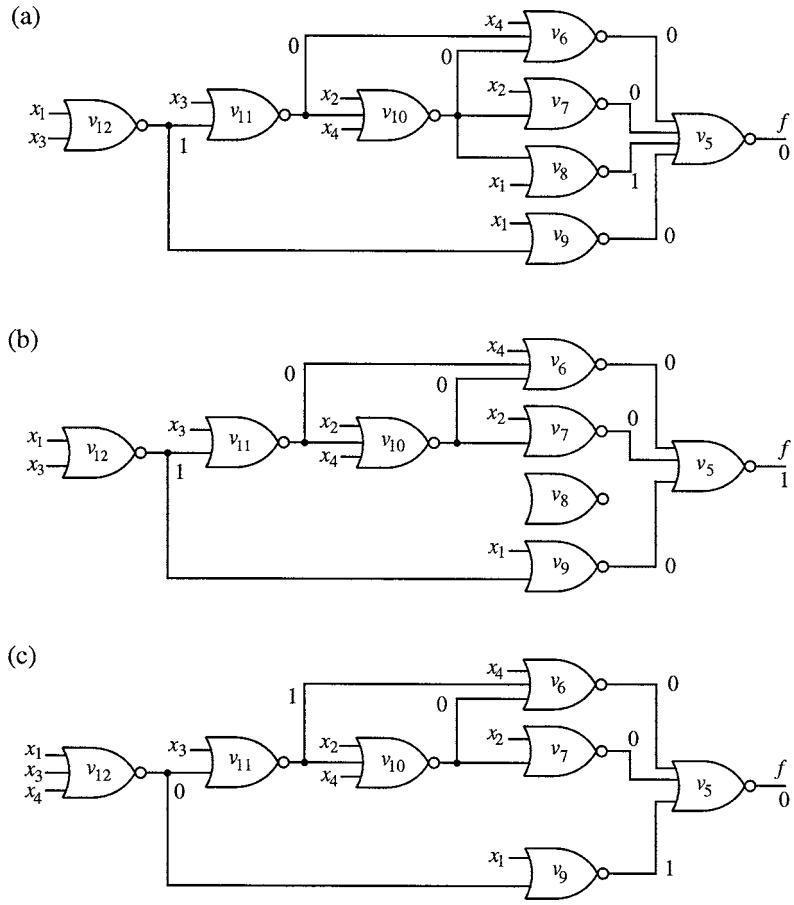


FIGURE 34.14 An example of error compensation.

In order to reduce the network, let us remove gate v_8, having the network in Fig. 34.14(b) whose output at gate v_5 is (1 0 0 0 0 $\underline{1}$ 0 0 0 0 1 0 1 1 0 1)

Note that the outputs of the two networks differ only in the 6-th components (underlined). We want to compensate for the value of the erroneous component of the latter network by adding connections. Functions realized at the input terminals v_1 through v_4 and gates v_5 through v_{12} in the original network in Fig. 34.14(a) are as follows:

$$
\begin{aligned}
x_1 = f(v_1) &= (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)\\
x_2 = f(v_2) &= (0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1)\\
x_3 = f(v_3) &= (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1)\\
x_4 = f(v_4) &= (0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1)\\
f(v_5) &= (1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1)\\
f(v_6) &= (0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0)\\
f(v_7) &= (0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 0)\\
f(v_8) &= (0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)\\
f(v_9) &= (0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)\\
f(v_{10}) &= (1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0)\\
f(v_{11}) &= (0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 0)\\
f(v_{12}) &= (1\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)
\end{aligned}
$$

The values of the 6-th component (i.e., the values corresponding to the input combination $x_1 = x_3 = 0$ and $x_2 = x_4 = 1$) are shown in Figs. 34.14(a) and (b). Components of vectors representing CSPFs can be calculated independently, so we calculate CSPFs for all components except the 6-th component (shown by "-") of the network in Fig. 34.14(b), as follows:

$$
\begin{aligned}
G_C(v_5) &= (1\ 0\ 0\ 0\ 0 - 0\ 0\ 0\ 0\ 1\ 0\ 1\ 1\ 0\ 1)\\
G_C(v_6) &= (0\ *\ *\ *\ 1 - 1\ *\ *\ *\ 0\ *\ 0\ 0\ 1\ 0)\\
G_C(v_7) &= (0\ 1\ *\ 1\ *\ - *\ *\ 1\ 1\ 0\ 1\ 0\ 0\ *\ 0)\\
G_C(v_9) &= (0\ *\ 1\ *\ *\ - *\ 1\ *\ *\ 0\ *\ 0\ 0\ *\ 0)\\
G_C(v_{10}) &= (1\ 0\ *\ 0\ 0 - 0\ *\ 0\ 0\ 1\ 0\ *\ *\ 0\ *)\\
G_C(v_{11}) &= (0\ *\ *\ *\ 0 - 0\ *\ 1\ *\ 0\ *\ 1\ *\ 0\ *)\\
G_C(v_{12}) &= (1\ *\ 0\ *\ 1 - *\ 0\ 0\ *\ *\ *\ 0\ *\ *\ *)
\end{aligned}
$$

If we can change the 6-th component of any of $f(v_6)$, $f(v_7)$, or $f(v_9)$ (i.e., immediate predecessors of gate v_5) from 0 to 1, the error in the network output can be compensated, as can be seen in Fig. 34.14(b) where v_8 is removed. The value 0 in the 6-th component of the output at gate, v_6, v_7, or v_9, is due to $x_4 = 1$, $x_2 = 1$, or $f(v_{12}) = 1$, respectively. If we want to change the output of v_9 from 0 to 1, the 6-th component of $f(v_{12})$ must be 0. If we can change the output of v_{12} to any function in the set of permissible functions

$$H = (1\ *\ 0\ *\ 1\ 0\ *\ 0\ 0\ *\ *\ *\ 0\ *\ *\ *)$$

that is $G_C(v_{12})$ except the 6-th component specified to 0, the error will be compensated. We can generate such a function by connecting x_4 to gate v_{12} and consequently by changing the output of v_{12} into

$$(1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$$

which is contained in H. The network obtained is shown in Fig. 34.14(c). Thus, the network with 8 gates and 20 connections shown in Fig. 34.14(a) is reduced to the network with 7 gates and 18 connections in Fig. 34.14(c). □

Let us describe the error compensation procedure illustrated by Example 34.5:

1. Remove a gate or a connection from a given network N, having a new network N'.
2. Calculate the erroneous components in the outputs of network N'.
3. Calculate the components of vectors representing MSPFs or CSPFs for the functions realized at the remaining gates and connections, corresponding to all error-free components of the outputs of N'.
4. Compensate for the errors by adding or removing connections.

This procedure can remove gates and connections in a more systematic manner than the other transformation procedures discussed so far.

34.3 Various Transduction Methods

In addition to **gate substitution**, **connectable/disconnectable conditions**, **generalized gate merging**, and **error compensation**, outlined thus far, some known transformations can be generalized for efficient processing, using permissible functions. In the gate merging procedure, for example, a permissible function which is common to two gates, v_i and v_j, can be easily found. Without permissible functions, the transformations would be excessively time-consuming.

We can have different transduction methods by combining different transformations and the pruning procedure. In other words, we can have different transduction methods based on different orders in processing gates, connections, and components of MSPFs or CSPFs.

These transduction methods can be realized in Fig. 34.2, which illustrates the basic structure of the transduction method outlined in Procedure 34.1. We can use these transduction methods in the following different manners:

1. An initial network can be designed by any conventional logic design method. Then we apply the transduction methods to such an initial network. The transduction methods applied to different initial networks usually lead to different final networks.
2. Instead of applying a transduction method only once, we can apply different transduction methods to an initial network in sequence. In each sequence, different or identical transduction methods can be applied in different orders. This usually leads to many different final networks.

Thus, if we want to explore the maximum potential of the transduction methods, we need to use them in many different ways, as explained in 1 and 2.[3,4]

Computational Example of the Transduction Method

Let us show a computational example of the transduction method. Suppose the initial network, which realizes a four-variable function, given as illustrated in Fig. 34.15(a), and this function has a minimal network shown in Fig. 34.15(b) (its minimality is proved by the integer programming logic design method). Beginning with the initial network of 12 gates shown in Fig. 34.15(a), the transduction method with error-compensation transformation (this version was called NETTRA-E3) produced the tree of solutions shown in Fig. 34.16. The size of the tree can be limited by the program parameter, NEPMAX.[6] (In Fig. 34.16, NEPMAX was set to 2. If set to 8, we will have a tree of 81 networks). The notation "*a*/*b*:*c*" in Fig. 34.16 means a network numbered *a* (numbered according to the order of generation), consisting of *b* gates and *c* connections, and a line connecting a larger network with a smaller one means that the smaller is derived, treating the larger as an initial network. In Fig. 34.16, it is important to notice that while some paths lead to terminal nodes representing minimal networks, others lead to terminal nodes representing networks not very close to the minimal. By comparing the numbers of gates and connections in the networks derived at the terminal nodes of this solution tree, a best solution can be found.

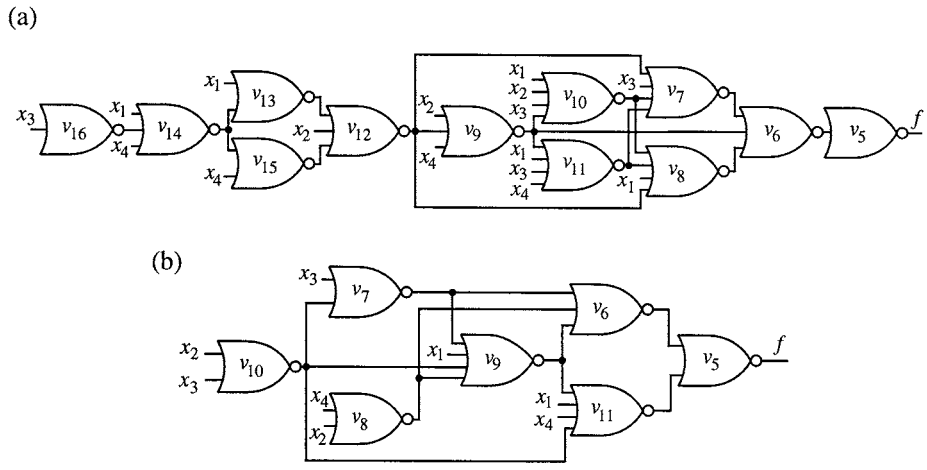


FIGURE 34.15 Initial and final networks for Fig. 34.6.

Intermediate solutions are logic networks with different connection configurations of negative gates, so some of them may be more appropriate for layout than others.

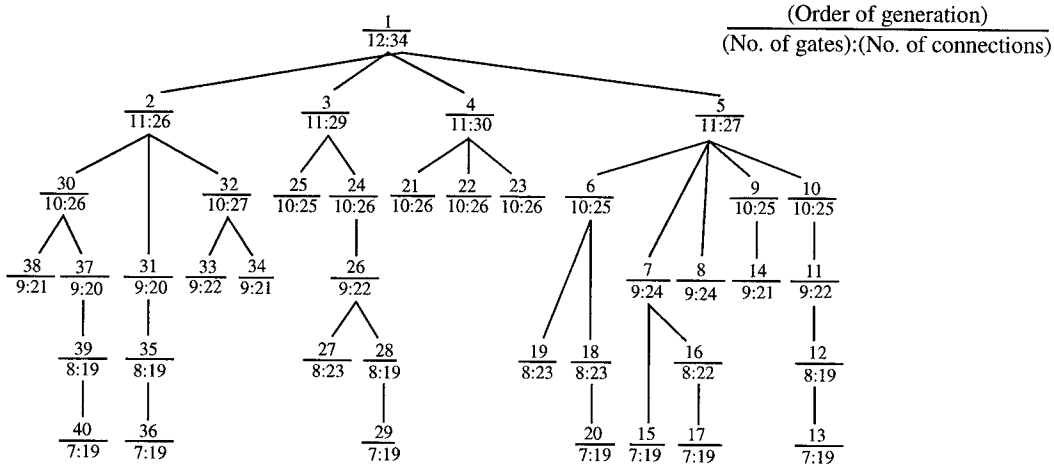


FIGURE 34.16 A tree of solutions generated by the transduction method based on error compensation.

34.4 Design of Logic Networks with Negative Gates by the Transduction Method

The transduction method has been described for the logic networks of NOR gates for the sake of simplicity, but it can be applied to logic networks of other types of gates, such as MOS logic gates and a mixture of AND gates and OR gates, tailoring its basic concepts (i.e., permissible functions and transformation). In this sense, it is important to understand what features different types of logic gates and consequently corresponding transistor circuits have in terms of logic operations and network transformations. In order to test the feasibility of design of logic networks with negative gates (MOS logic gates are negative gates) by the transduction method, a few synthesizers, called **SYLON** (an acronym for SYnthesis of LOgic Networks), were developed by modifying the transduction method.[1,2,7,8,11,12]

Some SYLON logic synthesizers consist of a mixture of technology-dependent optimization and technology-independent optimization. Here, let us outline **SYLON-REDUCE**,[7] a logic synthesizer which is of totally technology-dependent optimization and is more algorithmic, wherein a logic network is processed in its target technology throughout the execution of REDUCE. REDUCE reduces an initial network, using permissible functions, where in order to make each logic gate easily realizable as a MOS logic circuit, each logic gate throughout the execution of REDUCE is a negative gate that satisfies prespecified constraints on the maximum numbers of MOSFETs connected in series in each path and the maximum number of parallel paths. The reduction is done by repeatedly resynthesizing each negative gate. In other words, the outputs of some candidate gates or network inputs are connected to a gate under resynthesis and the connection configuration inside the gate is restructured, reducing the complexity of the gate and disconnecting unnecessary candidate gates or network inputs. The resynthesized cell is adopted if it has no more MOSFETs than the old gate and does not violate the constraints on the complexity (i.e., the specified maximum number of MOSFETs connected in series or the specified maximum number of parallel paths) otherwise, it is discarded, restoring the old gate. This resynthesis of each gate is repeated until no improvement can be done. Thus, the network transformation is done in a more subtle manner than the original transduction method. The result is a network where each gate still satisfies the same constraints on the complexity and contains no more MOSFETs than the corresponding gate in the original network and the connection configuration of the network may be changed.

References

1. Chen, K.-C., "Logic Synthesis and Optimization Algorithms," Ph.D. diss., Dept. of Comput. Sci., Univ. of Illinois, Urbana, 320, 1990.
2. Chen, K.-C. and S. Muroga, "SYLON-DREAM: A multi-level network synthesizer," *Proc. Int'l. Conf. on Computer-Aided Design*, pp. 552-555, 1989.
3. Hu, K. C., "Programming manual for the NOR network transduction system," UIUCDCS-R-77-887, Dept. Comp. Sci., Univ. of Illinois, Urbana, Aug. 1977.
4. Hu, K. C., and S. Muroga, "NOR(NAND) network transduction system (The principle of NETTRA system)," UIUCDCS-R-77-885, Dept. Comp. Sci., Univ. of Illinois, Urbana, Aug. 1977.
5. Kambayashi, Y., H. C. Lai, J. N. Culliney, and S. Muroga, "NOR network transduction based on error compensation (Principles of NOR network transduction programs NETTRA-E1, NETTRA-E2, NETTRA-E3)," UIUCDCS-R-75-737, Dept. of Comp. Sci., Univ. of Illinois, Urbana, June 1975.
6. Lai, H. C. and J. N. Culliney, "Program manual: NOR network transduction based on error compensation (Reference manual of NOR network transduction programs NETTRA-E1, NETTRA-E2, and NETTRA-E3)," UIUCDCS-R-75-732, Dept. Comp. Sci., Univ. of Illinois, Urbana, June 1975.
7. Limqueco, J. C. and S. Muroga, "SYLON-REDUCE: A MOS network optimization algorithm using permissible functions," *Proc. Int'l. Conf. on Computer Design*, Cambridge, MA, pp. 282-285, Sept. 1990.
8. Limqueco, J. C., "Logic Optimization of MOS Networks," Ph.D. thesis, Dept. of Comput. Sci., University of Illinois, Urbana, 250, 1992.
9. Muroga, S., "Computer-aided logic synthesis for VLSI chips," *Advances in Computers*, vol. 32, Ed. by M. C. Yovits, Academic Press, pp. 1-103, 1991.
10. Muroga, S., Y. Kambayashi, H. C. Lai, and J. N. Culliney, "The transduction method --- Design of logic networks based on permissible functions," *IEEE TC*, 38, 1404-1424, Oct. 1989.
11. Xiang, X. Q., "Multilevel Logic Network Synthesis System, SYLON-XTRANS, and Read-Only Memory Minimization Procedure, MINROM," Ph.D. diss., Dept. of Comput. Sci., Univ. of Illinois, Urbana, 286, 1990.
12. Xiang, X. Q. and S. Muroga, "Synthesis of multilevel networks with simple gates," *Int'l. Workshop on Logic Synthesis*, Microelectronic Center of North Carolina, Research Triangle Park, NC, May 1989.

Muroga, S. "Emitter-Coupled Logic"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

35

Emitter-Coupled Logic

Saburo Muroga
University of Illinois at Urbana-Champaign

35.1 Introduction

ECL, which stands for emitter-coupled logic, is based on bipolar transistors and is currently the logic family with the highest speed and a high output power capability, although power consumption is also the highest. ECL is more complicated to fabricate, covers a larger chip area, and is more expensive than any other logic family. As the speed of CMOS improves, ECL is less often used but is still useful for the cases where high speed, along with large output power, are necessary, such as high-speed transmission over communication lines. ECL has three types of transistor circuits: standard ECL logic gates, their modification with wired logic, and ECL series-gating.[4,6,10,13]

35.2 Standard ECL Logic Gates

A standard ECL logic gate is a stand-alone logic gate, and logic networks can be designed using many as building blocks. Its basic circuit is shown in Fig. 35.1. ECL has unique logic capability, as explained in the following.

The logic operation of the ECL gate shown in Fig. 35.1 is analyzed in Fig. 35.2, where the input z in Fig. 35.1 is eliminated for simplicity. The resistors connected to the bases of transistors, T_x and T_y, are for protecting these transistors from possible damage due to heavy currents through a transistor, but not for logic operations, and can be eliminated if there is no possibility for an excessively heavy current to flow. When input x has a high voltage representing logic value 1 and y has a low voltage representing logic value 0, transistor T_x becomes conductive, T_y becomes non-conductive, and a current flows through T_x and resistor R_1, as illustrated in Fig. 35.2(a). In this case the voltage at the emitter of transistor T_r becomes higher than –4 V due to the current through resistor R_p, as shown. Consequently, the voltage at this emitter becomes higher and the voltage at its base becomes not sufficiently high against its emitter to make T_r conductive, so there is no current through T_r and resistor R_2. Consequently, transistor T_f has a high voltage at its base, which makes T_f conductive, and output f has a high voltage representing logic value 1. On the other hand, transistor T_g is almost non-conductive (actually a small current flows, but let us ignore it for simplicity), since its base has a low voltage due to the voltage drop developed across resistor R_1 by the current shown. Thus, output g has a low voltage representing logic value 0.

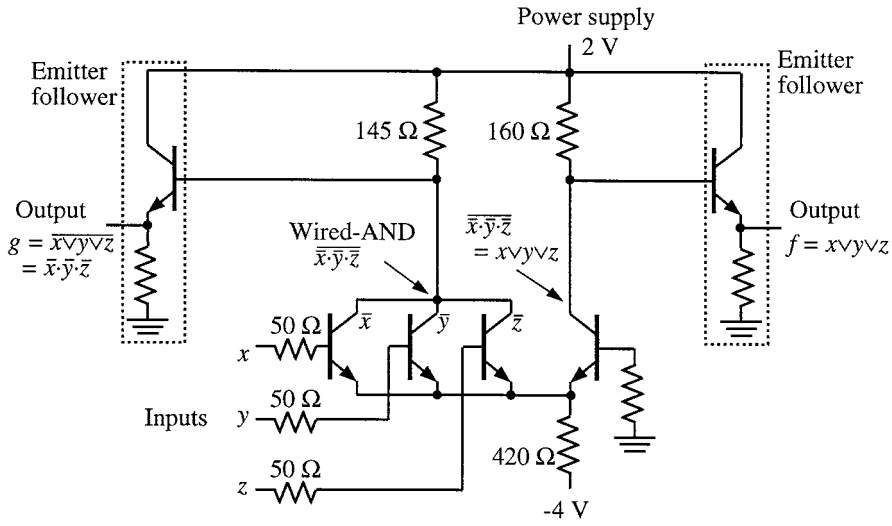


FIGURE 35.1 Basic circuit of standard ECL logic gate.

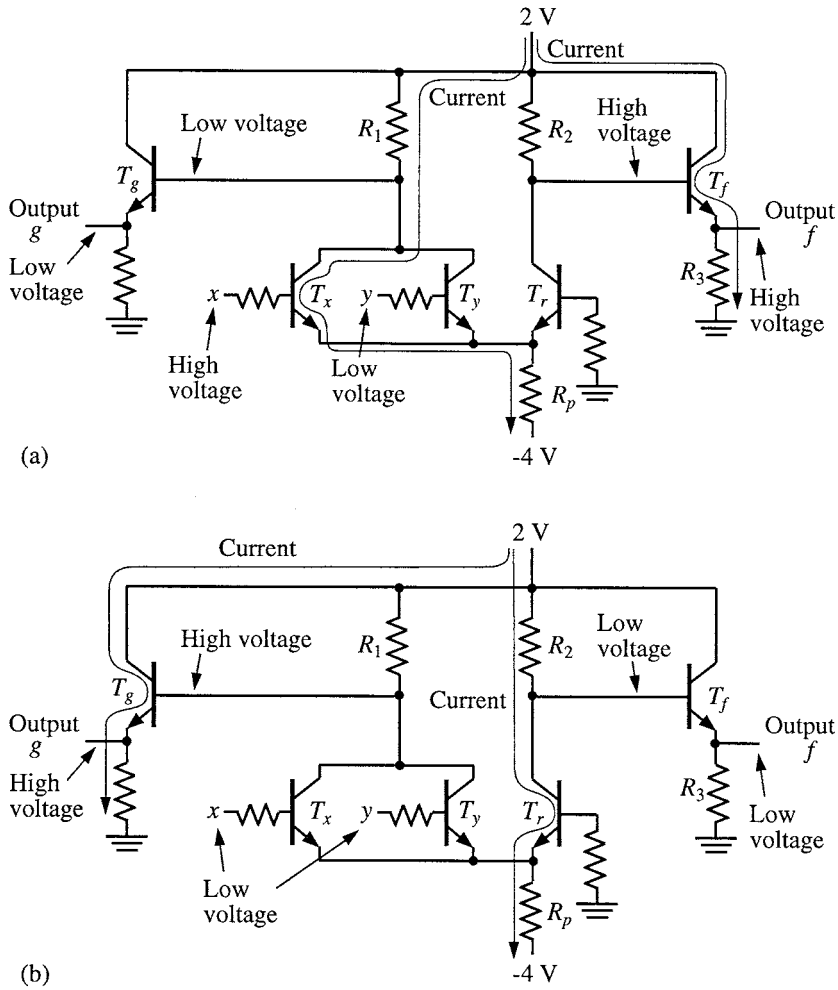


FIGURE 35.2 Logic operation of ECL gate in Fig. 35.1.

Even when y (instead of x), or both x and y, has a high voltage, the above situation is not changed, except for the current through T_y.

Next, suppose that both inputs x and y have low voltages, as shown in Fig. 35.2(b). Then there is no current through resistor R_1. Since the base of transistor T_r has a higher voltage than its emitter (0 V at the base and –0.8 V at the emitter), a current flows through R_2 and T_r, as illustrated in Fig. 35.2(b). Thus, T_f has a low voltage at its base and becomes almost non-conductive (more precisely speaking, less conductive). Output f has, consequently, a low voltage, representing logic value 0. Transistor T_g has a high voltage at its base and becomes conductive. Thus, output g has a high voltage, representing logic value 1.

Therefore, a current flows through only one of R_1 and R_2, switching quickly between these two paths. Notice that resistor R_p in Fig. 35.2 which is connected to a power supply of minus voltage is essential for this current steering because the voltage at the top end of R_p determines whether T_r becomes conductive or not. The emitter followers (shown in the dot-lined rectangles in Fig. 35.1) can deliver heavy output currents because an output current flows only through either transistor T_f or T_g and the on-resistance of the transistor is low.

The above analysis of Fig. 35.2 leads to the truth table in Table 35.1. From this table, the network in Fig. 35.2 has two outputs: $f = x \vee y$ and $g = \overline{x \vee y}$.

TABLE 35.1 Truth Table for Fig. 35.2

Inputs		Outputs	
x	y	f	g
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

In a similar manner, we can find that the ECL gate in Fig. 35.1 has two outputs: $f = x \vee y \vee z$ and $g = \overline{x \vee y \vee z}$.

The gate is denoted by the symbol shown in Fig. 35.3. The simultaneous availability of OR and NOR as the double-rail output logic, with few extra components, is the unique feature of the ECL gate, making its logic capability powerful.

FIGURE 35.3 Symbol for the standard ECL logic gate of Fig. 35.1.

Emitter-Dotting

Suppose we have the emitter follower circuit shown in Fig. 35.4(a) (also shown in the dot-lined rectangles in Fig. 35.1), as part of an ECL logic gate. Its output function, f, at the emitter of the bipolar transistor is 1 when the voltage at the emitter is high (i.e., the bipolar transistor in (a) is conductive), and f is 0 when the voltage at the output terminal is low (i.e., the bipolar transistor in (a) is non-conductive). Then, suppose there is another alike circuit whose output function at the emitter is g. If the emitters of these two circuits are tied together as shown in Fig. 35.4(b), the new output function at the tied point is $h = f \vee g$, replacing the original functions, f and g. This connection is called **emitter-dotting**, realizing **Wired-OR**. The tied point represents the new function $f \vee g$ because if both transistors, T_1 and T_2, are non-conductive, the voltage at the tied point is low; otherwise (i.e., if one of T_1 and T_2, or both, is conductive), the voltage at the tied point is high.

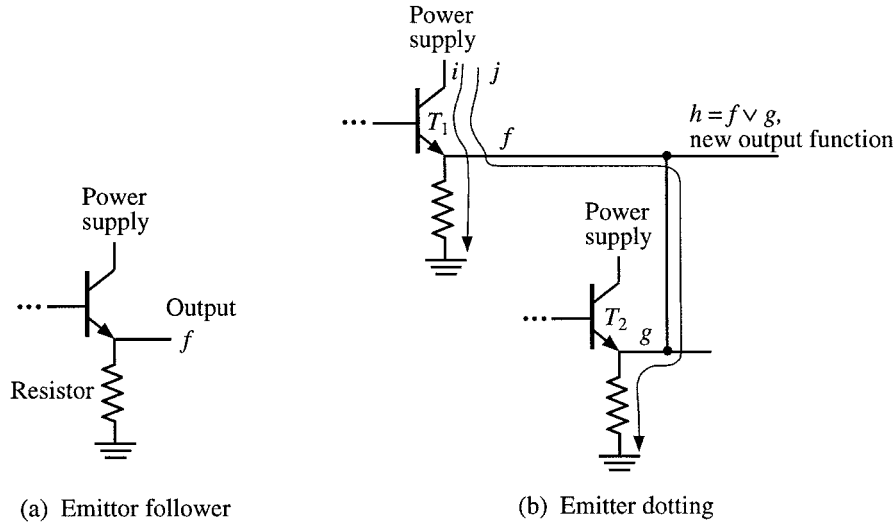


FIGURE 35.4 Emitter-dotting.

Wired-OR is an important feature of the ECL gate. The ECL gate in Fig. 35.1 has two emitter followers: one for f and the other for g. As shown in Fig. 35.5, the OR of the outputs can be realized without using an extra gate, simply by tying together these outputs. This is very convenient in logic design. **If one output is Wired-ORed with another ECL gate, it does not express the original function. And it cannot be further Wired-ORed to other gates if we want to realize Wired-OR with the original function.** But if the same output is repeated by adding an emitter follower inside the gate, as shown in Fig. 35.6 (i.e., the emitter follower inside a dot-lined rectangle in Fig. 35.1), then the new output can be Wired-ORed with another gate output or connected without Wired-OR to the succeeding gates. In the ECL gate at the top position in Fig. 35.6, for example, the first output $f = x \vee y \vee z$ is connected to gates in the next level, while the same f in the second output is used to produce the output $\overline{u \vee v} \vee x \vee y \vee z$ by Wired-ORing with the output of the second ECL gate.

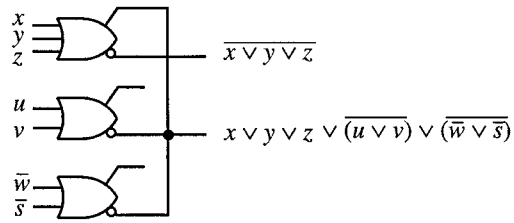


FIGURE 35.5 Wired-OR of ECL gates.

Design of a Logic Network with Standard ECL Gates

An ECL gate network can be designed, starting with a network of only NOR gates, for the following reason. Consider a logic network of ECL logic gates shown in Fig. 35.7(a), where Wired-ORs are included. This network can be converted into the network without Wired-OR shown in (b) by directly connecting connections in each Wired-OR to the inputs of a NOR gate without changing the outputs at gates 4 and 5, possibly sacrificing the maximum fan-in restriction. Then, two NOR outputs of gate 2 in (a), for example, can be combined into one in (b). Then, this network can be converted into the network shown

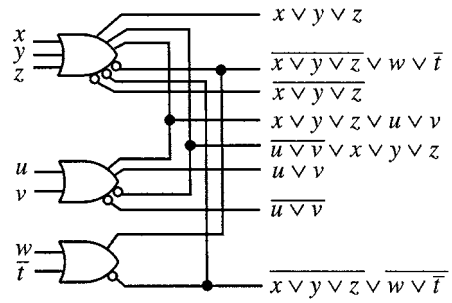


FIGURE 35.6 Multiple-output ECL gates for Wired-OR.

in (c) by eliminating OR outputs of all gates 1, 2, and 3 in (b), and connecting inputs of these gates directly to gates 4 and 5. Thus, the network in (c) that expresses the same outputs as the network in (a) consists of NOR gates only (i.e., the outputs of gates 4 and 5 in (c) are the same as those in (a)), possibly further sacrificing the maximum fan-in restriction at some gates. Notice that in this conversion, the number of gates does not change or decreases (if an ECL gate, like gate 1 in (c), has no outputs used, it can be deleted from (c)). Thus, from the given network of standard ECL gates with Wired-ORs, we can derive a NOR network of the same or fewer number of gates, possibly with greater fan-in at some gates, as shown in Fig. 35.7(d). Even if each gate has many NOR outputs or OR outputs, the situation does not change.

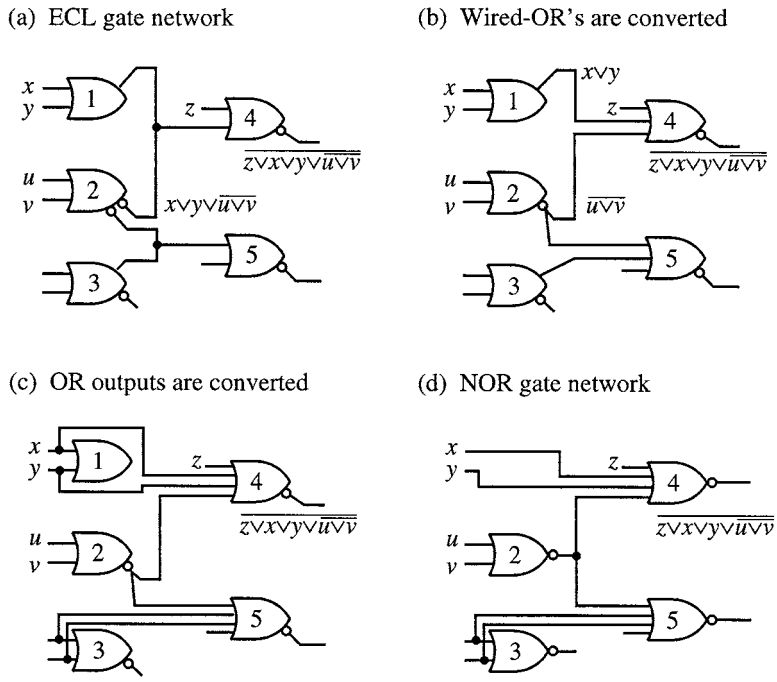


FIGURE 35.7 Conversion of an ECL gate network into a NOR gate network.

When we want to design a minimal standard ECL gate network for given functions f and $\bar{f}$, it can be designed by reversing the preceding conversion, as follows.

Procedure 35.1: Design of logic networks with standard ECL logic gates

1. Design a network for a given logic function f and another network for its complement, $\bar{f}$ using NOR gates only without considering maximum fan-in or fan-out restriction at each gate. Use a minimum number of NOR gates in each case. (The map-factoring method described in Chapter 31 is usually convenient for manual design of logic networks with a minimum number of NOR gates in single- or double-rail input logic.)
2. Choose one among the two logic networks obtained. Reduce the number of input connections to each gate, by providing Wired-ORs, or by using OR-outputs of other gates, if possible. In this case, extra NOR or OR outputs at each ECL gate must be provided whenever necessary (like the reverse conversion from Fig. 35.7(b) to Fig. 35.7(a), or from Fig. 35.7(c) to Fig. 35.7(b)). Thus, if any gate violates the maximum fan-in restriction, we can try to avoid it by using Wired-ORs or OR outputs.
3. This generally reduces fan-out of gates also; but if any gate still violates the maximum fan-out restriction, try to avoid it by using extra ECL gates (no simple good methods are known for doing this). The output ECL gate of this network presents f and $\bar{f}$.
   If no gate violates the maximum fan-in and fan-out restrictions in Steps 2 and 3, the number of NOR gates in the original NOR network chosen in Step 2 is equal to the number of ECL gates in the resultant ECL network. So, if we originally have a network with a minimum number of NOR gates, the designed ECL network also has the minimum number of standard ECL logic gates. But extra ECL gates have to be added if some gates violate the maximum fan-in restriction, maximum fan-out restriction, or other constraints.
4. Repeat Steps 2 and 3 for the other network. Choose the better one. □

Notice that the use of OR outputs and Wired-ORs generally reduces the number of connections or fan-ins (i.e., input transistors) and also reduces the total sum of connection lengths, thus saving chip area. For example, the total length of the connections for x and y in Fig. 35.7(c) can be almost twice the connection length between two gates in Fig. 35.7(b). Also, the total length of two connections in Fig. 35.7(b) can be almost twice the length for Wired-OR in Fig. 35.7(a). In Procedure 35.1, NOR networks with a minimum number of gates are important initial networks. It is known that when the number of gates is minimized, the number of connections in the networks also tends to be minimized.[11] (For the properties of wired logic, see Ref. 7.)

35.3 Modification of Standard ECL Logic Gates with Wired Logic

More complex logic functions than the output functions of the standard ECL logic gate shown in Fig. 35.1 can be realized by changing the internal structures of the standard ECL logic gates. In other words, if we connect points inside one ECL gate to some points of another ECL gate, we can realize a complex logic function with a simpler electronic circuit configuration. In other words, we can realize logic functions by freely connecting transistors, resistors, and diodes, instead of regarding the fixed connection configuration of transistors, resistors, and diodes as logic gates whose structure cannot be changed. This approach could be called **transistor-level logic design**. Wired logic is a powerful means for this, and collector-dotting and emitter-dotting are the basic techniques of wired logic.

Collector-Dotting

Collector-dotting is commonly used in bipolar transistor circuitry to realize the **Wired-AND operation**. Suppose we have the inverter circuit shown in Fig. 35.8(a) as part of an ECL logic gate. Its output function, f, at the collector of the bipolar transistor is 1 when the voltage at the collector is high, and f is 0 when the voltage at the collector is low. Then, suppose there is another like circuit whose output function at the collector is g. If the collectors of these two circuits, instead of the emitters for emitter-dotting in Fig. 35.4(b), are tied together as shown in Fig. 35.8(b), the new output function at the tied point is $h = f \cdot g$, replacing the original functions, f and g. This connection is called **collector-dotting**, realizing **Wired-**

AND. The tied point represents the new function $f \cdot g$ because if one of T_1 and T_2, or both is conductive in Fig. 35.8(b), the voltage at the tied point is low; otherwise (i.e., only when both transistors, T_1 and T_2, are non-conductive), the voltage at the tied point can be high.

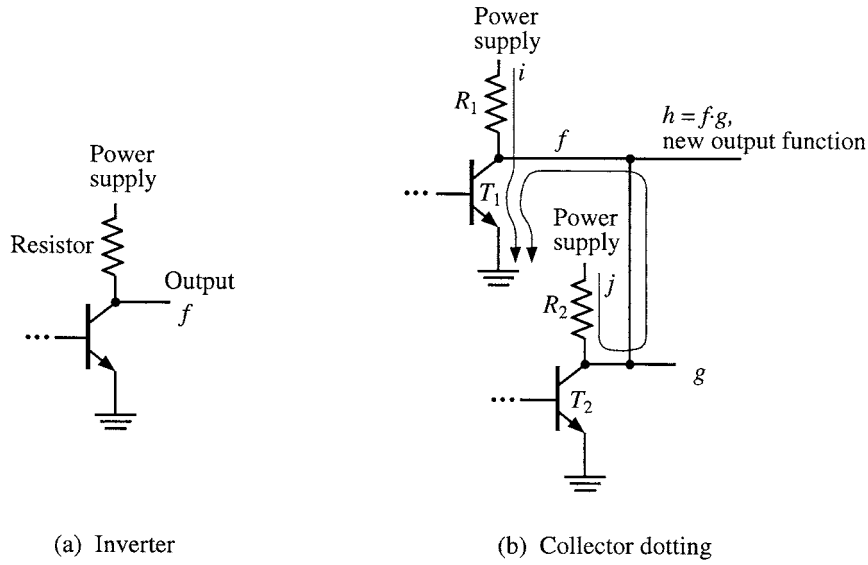


FIGURE 35.8 Collector-dotting.

In Fig. 35.9(a), Fig. 35.2 is repeated as gate 1 and gate 2. Transistor T_x has input x at its base, and its collector represents function $\bar{x}$ if T_y does not exist because when its base has a low voltage (i.e., logic value 0), its collector has a high voltage (i.e., logic value 1) and vice versa. Similarly, the collector of transistor T_y represents $\bar{y}$, if T_x does not exist. Then by tying together these collectors (i.e., collector-dotting), the tied point (i.e., point A) represents $\bar{x} \cdot \bar{y} = \overline{x \vee z}$, as already explained with respect to Fig. 35.2. Notice that the collector of T_x and the collector of T_y do not represent the original functions $\bar{x}$ and $\bar{y}$ respectively, after collector-dotting. Since the voltage level at B is always opposite to that at A, point B represents $x \vee y$.

We can use collector-dotting more freely. Point A in gate 1 in Fig. 35.9(a) can be connected to point A' or B' in gate 2. Point B can also be connected to point A' or B'. Such connections realize Wired-AND or collector-dotting. By collector-dotting points B and B' as shown in Fig. 35.9(b), point B 7 (also B') represents new function $(x \vee y) \cdot (z \vee w)$, which also appears at the emitter of transistor T. After this collector-dotting, points B and B' do not represent the original functions $x \vee y$ and $z \vee w$, respectively, anymore. Also, note that the function at any point that is not collector-dotted, such as A and A', is unchanged by collector-dotting of B and B'. In Fig. 35.9(b), two transistors, two diodes, and resistors (shown in the dotted line) are added for adjustment of voltage and current. But they have nothing to do with logic operations.

Another example is the parity function $\bar{x}y \vee x\bar{y}$ realized by connecting two ECL gates as shown in Fig. 35.10. The parity function requires four ECL gates if designed with the standard ECL logic gates as shown in Fig. 35.10(c), but can be realized by the much simpler electronic circuit of Fig. 35.10(b). In other words, $\bar{x}y$ and $x\bar{y}$ are realized by Wired-AND, then these two products are Wired-ORed in order to realize $\bar{x}y \vee x\bar{y}$. In Fig. 35.10 as well as Fig. 35.9, some resistors or transistors may be necessary for electronic performance improvement (since resistors R_1 and R_2 draw too much current in gate 1 in Fig. 35.10(a), new resistors are added in Fig. 35.10(b) in order to clamp the currents), and unnecessary resistors or transistors may be deleted, although such an addition or elimination of resistors or transistors has nothing to do with logic operations.

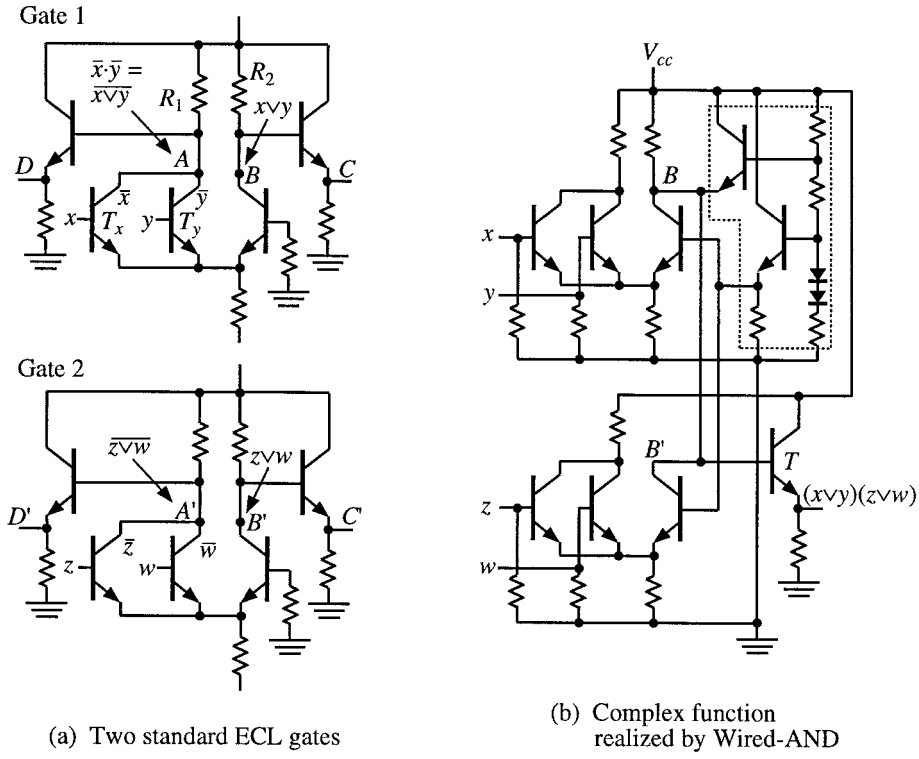


FIGURE 35.9 Example of Wired-AND.

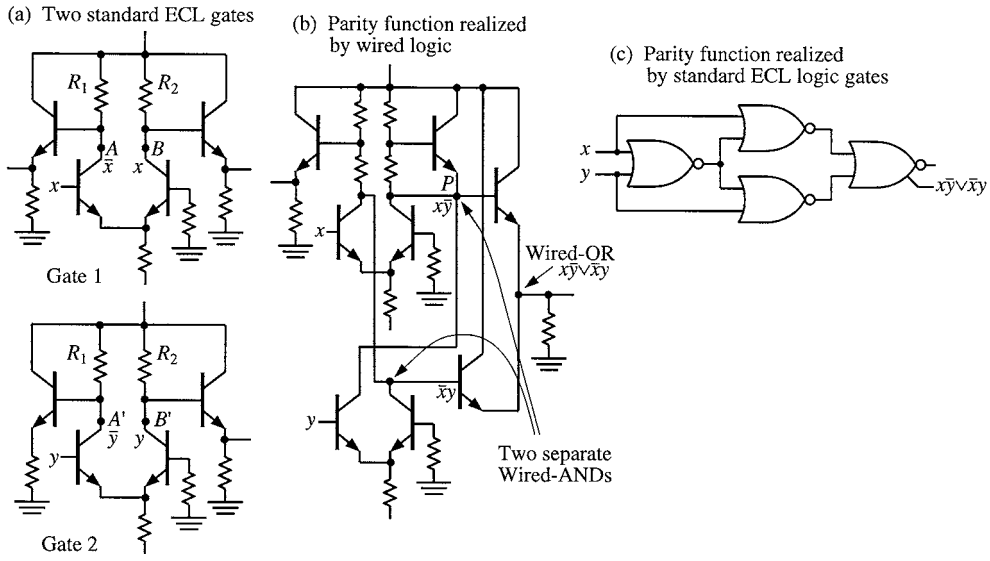


FIGURE 35.10 Parity function realized by ECL gate with Wired logic.

35.4 ECL Series-Gating Circuits

Figure 35.11(a) shows the basic pair of bipolar transistors in **series-gating ECL**, where A is connected to a power supply and B is connected to another power supply of minus voltage through a resistor. (Notice that this pair is T_x and T_r in Fig. 35.2, from which T_y is eliminated.) Transistor T_1 has an input x connected to its base and the other transistor T_2 has a constant voltage v_{ref} at its base. As illustrated in Fig. 35.2, v_{ref} is grounded through a resistor (i.e., $v_{ref} = 0$), where this resistor is for protection of transistor T_r from damage by a heavy current, and v_{ref} works as a reference voltage against changes of x. (The voltage at v_{ref} can be provided by a subcircuit consisting of resistors, diodes, and transistors, like the one in Fig. 35.9(b).) The collector of T_1 represents a logic function $\bar{x}$ because the collector of T_1 can have a high voltage (i.e., logic value 1) only when T_1 is non-conductive, that is, the input is a low voltage ($x = 0$).

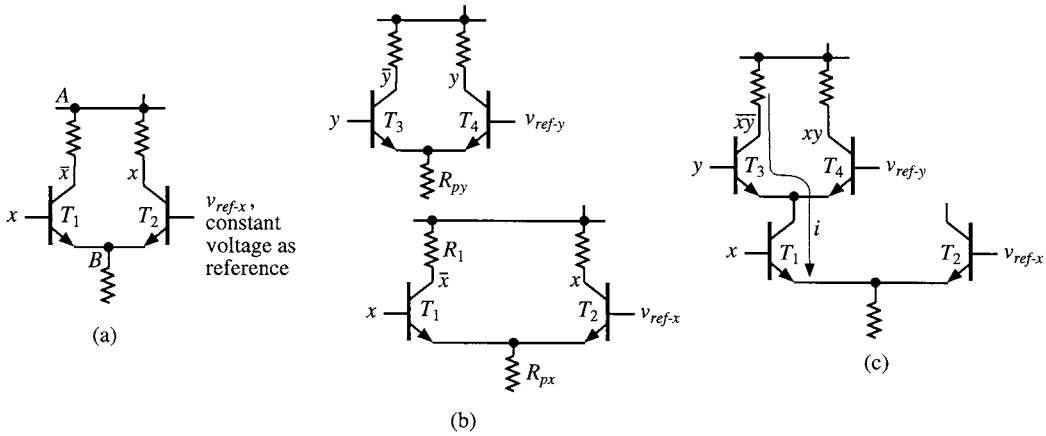


FIGURE 35.11 Series-gating.

The collector of T_2 represents function x because it becomes a high voltage only when the input x is a high voltage (i.e., when the input is a high voltage, T_1 becomes conductive and T_2 becomes non-conductive because a current flows at any time through exactly one of two transistors, T_1 and T_2. Thus, the collector of T_2 becomes a high voltage).

In Fig. 35.11(b), we have two pairs of transistors. In other words, we have the pair with input y, in addition to the pair with input x shown in (a). Then let us connect them in series without R_{py}, R_1 and the power supply for R_1, as shown in (c). The voltage at the collector of T_3 is low only when T_3 and T_1 are both conductive and, consequently, a current i flows through T_3 and T_1. The voltage at the collector of T_3 is high when either T_3 or T_1 is non-conductive (i.e., $x = 0$ or $y = 0$) and consequently no current (i.e., i) flows through T_3 and T_1. Thus, the collector of T_3 represents the function $\overline{xy}$, replacing the original function $\bar{y}$ shown in (b). This can be rewritten as $\overline{xy} = \bar{x} \vee \bar{y}$, so **series-gating can be regarded as the OR operation**.

Many of the basic pair of transistors shown in Fig. 35.11(a) are connected in a tree structure, as shown in Fig. 35.12, where inputs x, y, and z, as well as reference voltages, $v_{ref\text{-}1}$, $v_{ref\text{-}2}$, and $v_{ref\text{-}3}$, need to be at appropriate voltage levels. Then the complement of all minterms can be realized at the collectors of transistors in the top level of the series connections. Two of these complemented minterms (i.e., $\overline{\bar{x} \vee \bar{y} \vee z}$ and $\overline{\bar{x} \vee \bar{y} \vee \bar{z}}$) are shown with emitter followers, as examples at the far right end of Fig. 35.12.

Some of these collectors of transistors in the top level can be collector-dotted to realize the desired logic functions, as illustrated in Fig. 35.13. Notice that **once collectors are collector-dotted, these collectors do not express their respective original functions.**

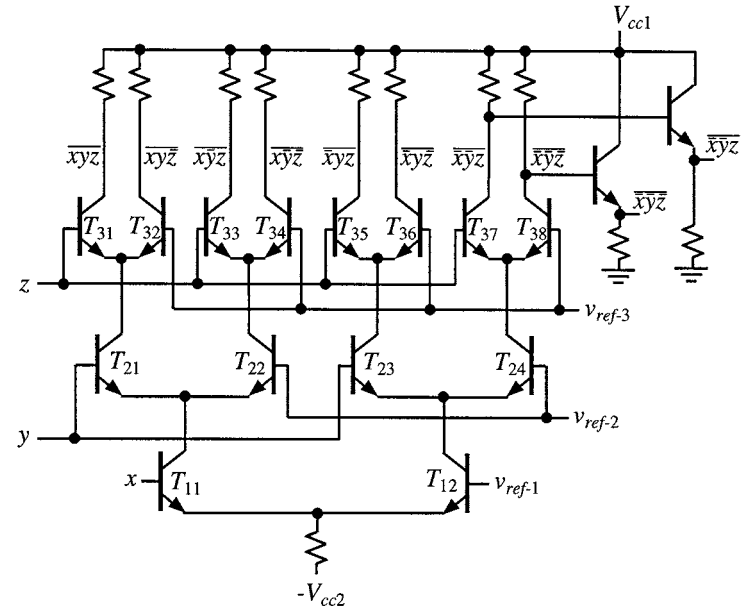


FIGURE 35.12 ECL series-gating.

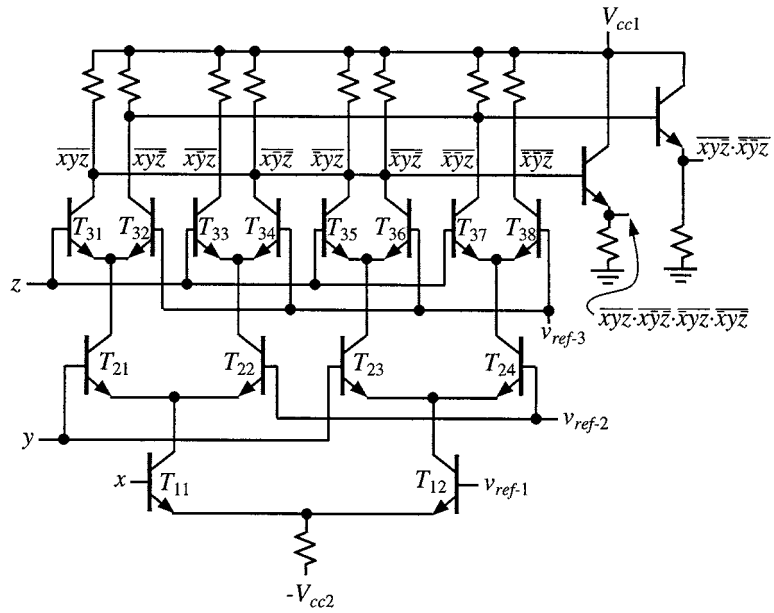


FIGURE 35.13 ECL series-gating.

The full adder in Fig. 35.14 is a more complex example of series-gating.[3] In this figure, we use collector-dotting by tying together some of collectors to realize Wired-AND, as explained already. For example, the voltage at the collector of transistor T_{31} represents function $\overline{xyc}$ because of series-gating with T_{11}, T_{21}, and T_{31}. Usually, at most, three transistors are connected in series (the two transistors in the bottom

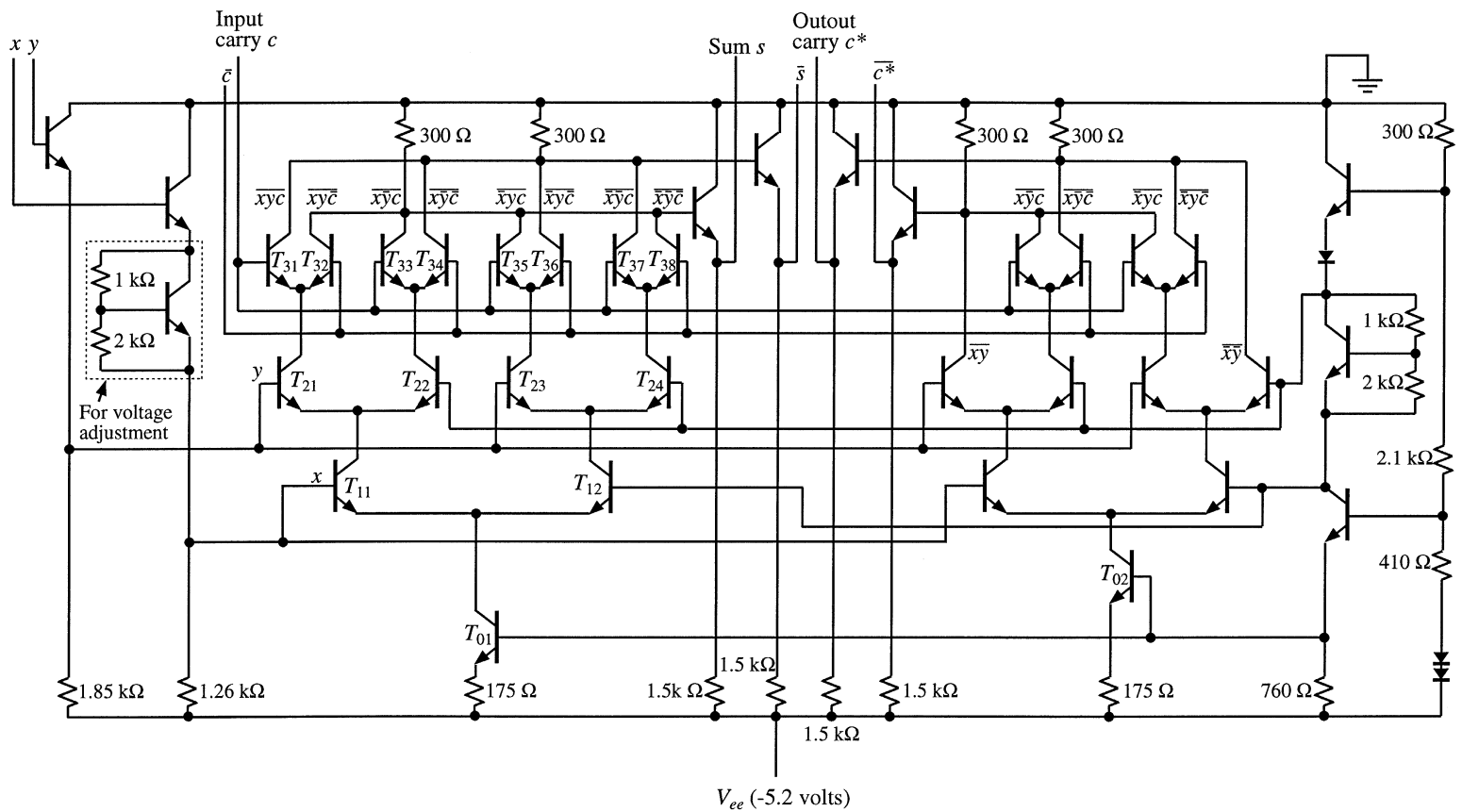


FIGURE 35.14 ECL full adder with series-gating. (From Ref. 3.)

level, T_{01} and T_{02}, in Fig. 35.14 are for controlling the amount of current as part of the power supply). This is because too many transistors in series tend to slow down the speed of the gate due to parasitic capacitances to ground. Then, by collector-dotting, some collectors of the transistors in the top level, sum s, and carry $c*$ are realized, as well as their complements, $\bar{s}$ and $\overline{c*}$.

Baugh and Wooley have designed a full adder in double-rail logic.[2] Ueda designed a full adder with ECL gates in single-rail logic with fewer transistors.[16]

The implementation of Wired-AND in this manner requires careful consideration of readjustments of voltages and currents. (Thus, transistors or resistors may be added or changed in order to improve electronic performance, but this is not directly related to logic operations.)

ECL series-gating can be extended as follows. Unlike the series-gating in Figs. 35.12, 35.13, and 35.14, the same input variables are not necessarily used in each level. For example, in the top level in Fig. 35.15, y and z are connected to the bases of transistors, instead of all y's. Then, collectors can be collector-dotted, although collector-dotting is not done in this figure. Complements of products, such as $\overline{xy}$ and $\overline{\bar{x}z}$, can be realized at collectors in the top level by the series-gating, as shown in Fig. 35.15. By this free connection of input variables, functions can be generally realized with fewer transistors.

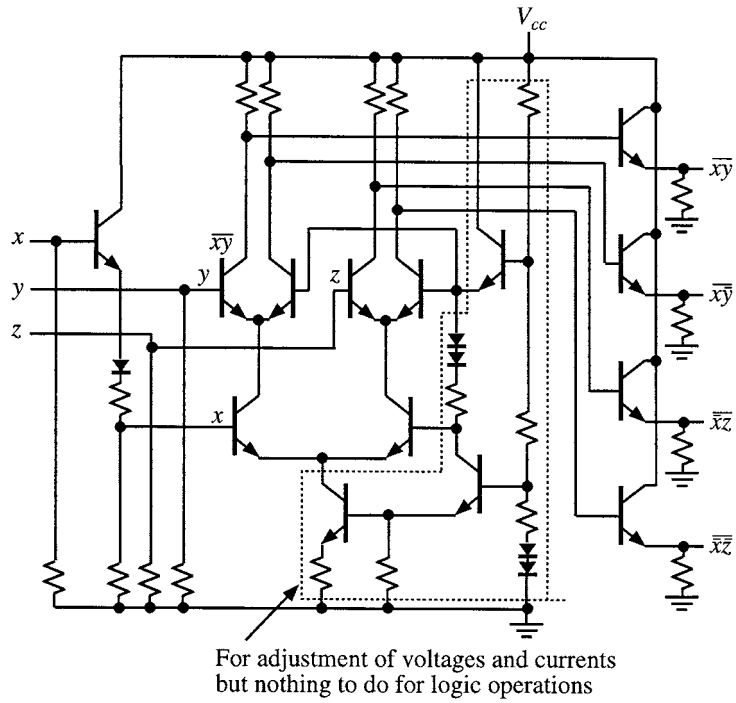


FIGURE 35.15 Series-gating.

CMOS has very low power consumption at low frequency but may consume more power than ECL at high speed (i.e., at high frequency). This is because the power consumption of CMOS is proportional to CFV^2, where C is parasitic capacitance, F is a switching frequency, and V is the power supply voltage. Thus, at high frequency, the power consumption of CMOS exceeds that of ECL, which is almost constant.

It is important to note that compared with the standard ECL logic gate illustrated in Fig. 35.1, series-gating ECL is faster with low power consumption for the following reasons:

- Because of speed-up of bipolar transistor (reduction of base thickness, and others), delay time over connections among standard ECL logic gates is greater than delay time inside logic gates and

then series-gating ECL, which can realize far more complex logic functions than NOR or OR realized by standard ECL gate and consequently eliminates long connections required among standard ECL logic gates, can have shorter delay.

- A series-gating ECL circuit has lower power consumption than a logic network with standard ECL logic gates because power supplies to all standard ECL logic gates are combined into one for the series-gating ECL circuit and a current flows in only one path at any time.[1,5,9,12]
- Then, in recent years, the power consumption of series-gating ECL is reduced with improved circuits.[8]
- The power consumption of series-gating ECL can also be reduced by active pull-down of some emitters.[14,15]

References

1. Abe, S., Y. Watanabe, M. Watanabe, and A. Yamaoka, "M parallel series computer for the changing market," *Hitachi Review*, vol. 45, no. 5, pp. 249-254,1996.
2. Baugh, C. R. and B. A. Wooley, "One bit full adder," U.S. Patent 3,978,329, August 31, 1976.
3. Garret, L. S., "Integrated-circuit digital logic families III — ECL and MOS devices," *IEEE Spectrum*, pp. 30-42, Dec. 1970.
4. Gopalan, K. G., *Introduction to Digital Microelectronic Circuits*, McGraw-Hill, 1996.
5. Higeta, K. et al., "A soft-error-immune 0.9-ns 1.15-Mb ECL-CMOS SRAM with 30-ps 120 k logic gates and on-chip test circuitry," *IEEE Jour. of Solid-State Circuits*, vol. 31, no. 10, pp. 1443-1450, Oct. 1996.
6. Jager, R. C., *Microelectronics Circuit Design*, McGraw-Hill, 1997.
7. Kambayashi, Y. and S. Muroga, "Properties of wired logic," *IEEE TC*, vol. C-35, pp. 550-563, 1986.
8. Kuroda, T., et al., "Capacitor-free level-sensitive active pull-down ECL circuit with self-adjusting driving capability," *Symp. VLSI Circuits*, pp. 29-30, 1993.
9. Mair, C. A., et al., "A 533-MHz BiCMOS superscaler RISC microprocessor," *IEEE JSSC*, pp. 1625-1634, Nov. 1997.
10. Muroga, S., *VLSI System Design*, John Wiley and Sons, 1982.
11. Muroga, S. and H.-C. Lai, "Minimization of logic networks under a generalized cost function," *IEEE TC*, pp. 893-907, Sept. 1976.
12. Nambu, H., et al., "A 0.65-ns, 72-kb ECL-CMOS RAM macro for a 1-Mb SRAM," *IEEE Jour. of Solid-State Circuits*, vol. 30, no. 4, pp. 491-499, April 1995.
13. Sedra, A. S. and K. C. Smith, *Microelectronic Circuits*, 4th ed., Oxford University Press, 1998.
14. Shin, H. J., "Self-biased feedback-controlled pull-down emitter follower for high-speed low-power bipolar logic circuits," *Symp. VLSI Circuits*, pp. 27-28, 1993.
15. Toh, K.-Y. et al., "A 23-ps/2.1-mW ECL gate with an AC-coupled active pull-down emitter-follower stage," *Jour. SSC*, pp. 1301-1306, Oct. 1989.
16. Ueda, T., Japanese Patent Sho 51-22779, 1976.

Muroga, S. "CMOS"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

36 CMOS

Saburo Muroga
University of Illinois at Urbana-Champaign

36.1 CMOS (Complementary MOS)

A CMOS logic gate consists of a pair of subcircuits, one consisting of nMOSFETs and the other pMOSFETs, where all MOSFETs are of enhancement mode described in Chapter 30, Section 30.3. **CMOS**, which stands for complementary MOS,[6,8–10] means that the nMOS and pMOS subcircuits are complementary. As a simple example, let us explain CMOS with the inverter shown in Fig. 36.1. A p-channel MOSFET is connected between the power supply of positive voltage V_{dd} and the output terminal, and an n-channel MOSFET is connected between the output terminal and the negative side, V_{ss}, of the above power supply, which is usually grounded. When input x is a high voltage, pMOS becomes non-conductive and nMOS becomes conductive. When x is a low voltage, pMOS becomes conductive and nMOS becomes non-conductive. This is the property of pMOS and nMOS when the voltages of the input and the power supply are properly chosen, as explained with Fig. 30.19. In other words, when either pMOS or nMOS is conductive, the other is non-conductive. When x is a low voltage (logic value 0), pMOS is conductive, with non-conductive nMOS, and the output voltage is a high voltage (logic value 1), which is close to V_{dd}. When x is a high voltage, nMOS is conductive, with non-conductive pMOS, and the output voltage is a low voltage. Thus, the CMOS logic gate in Fig. 36.1 works as an inverter. The pMOS subcircuit in this figure essentially works as a variable load.

When x stays at either 0 or 1, one of pMOS and nMOS subcircuits in Fig. 36.1 is always non-conductive, and consequently no current flows from V_{dd} to V_{ss} through these MOSFETs. In other words, when no input changes, the power consumption is simply the product of the power supply voltage V (if V_{ss} is grounded, V is equal to V_{dd}) and a very small current of a non-conductive MOSFET. (Ideally, there should be no current flowing through a non-conductive MOSFET, but actually a very small current which is less than 10 nA flows. Such an undesired, very small current is called a **leakage current**.) This is called the **quiescent power consumption**. Since the leakage current is typically a few nanoamperes, the quiescent power consumption of CMOS is less than tens of nW, which is very small compared with those for other logic families.

Whenever the input x of this CMOS logic gate changes to a low voltage (i.e., logic value 0), the parasitic capacitance C at the output terminal (including parasitic capacitances at the inputs of the succeeding

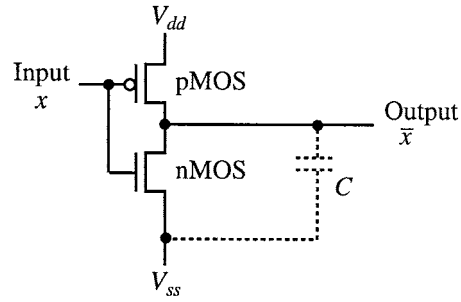


FIGURE 36.1 CMOS inverter.

CMOS logic gates, to which the output of this logic gate is connected) must be charged up to a high voltage through the conductive pMOS. (A current can be as large as 0.3 milliamperes or more.) Then, when the input x changes to a high voltage (i.e., logic value 1) at the next input change, the electric charge stored in the parasitic capacitance must be discharged through the conductive nMOS. Therefore, much larger power consumption than the quiescent power consumption occurs whenever the input changes. This dynamic power consumption due to the current during this transition period is given by CFV^2, where C is the parasitic capacitance, V is the power supply voltage, and F is the switching frequency of the input. Thus the power consumption of CMOS is a function of frequency. CMOS consumes very little power at low frequency, but it consumes more than ECL as the frequency increases. As the integration size increases, CMOS is being almost exclusively used in VLSI because of low power consumption. But even CMOS has difficulty in dissipation of the heat generated in the chip when switching frequency increases. In order to alleviate this difficulty, valiants, such as dynamic CMOS, have been used which will be described later.

Output Logic Function of a CMOS Logic Gate

Let us consider a CMOS logic gate in which many MOSFETs of the enhancement mode are connected in each of the pMOS and nMOS subcircuits (e.g., the CMOS logic gate in Fig. 36.2). By regarding the pMOS subcircuit as a variable load, the output function f can be calculated in the same manner as the one of an nMOS logic gate:

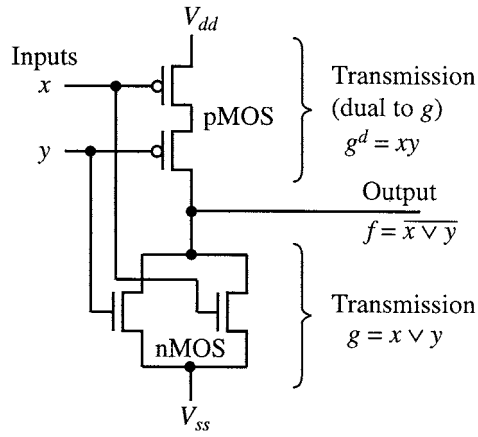


FIGURE 36.2 CMOS NOR gate.

1. Calculate the transmission between the output terminal and V_{ss} (or the ground), considering nMOSFETs as make-contacts of relays (transmission and relays are described in Chapter 30).
2. Then, its complement is the output function of this CMOS logic gate.

Thus, the CMOS logic gate in Fig. 36.2, for example, has the output function $f = \overline{x \vee y}$.

We can prove that if the pMOS subcircuit of any CMOS logic gate has the transmission between V_{dd} and the output terminal, calculated by regarding each pMOS as a make-contact relay, and this transmission is the dual of the transmission of the nMOS subcircuit, then one of the pMOS and nMOS subcircuits is always non-conductive, with the other conductive, for any combination of input values. (Note that regarding each pMOS as a make-contact relay, as we do each nMOS as a make-contact relay, means finding a relationship of connection configuration between nMOS subcircuit and pMOS subcircuit.) In the CMOS logic gate of Fig. 36.2, the pMOS subcircuit has transmission $g^d = xy$, which is dual to the transmission $g = x \vee y$ of the nMOS subcircuit, where the superscript d on g means "dual." Thus, any CMOS logic gate has the unique features of unusually low quiescent power consumption and dynamic power consumption CV^2F.

The input resistance of a CMOS logic gate is extremely high and at least 10^{14} **Ω**. This permits large fan-outs from a CMOS logic gate. Thus, if inputs do not change, CMOS has almost no maximum fan-out restriction. The practical maximum fan-out is 30 or more, which is very large compared with other logic families. If the number of fan-out connections from a CMOS logic gate is too many, the waveform of a signal becomes distorted. Also, fan-out increases the parasitic capacitance and consequently reduces the speed, so fan-out is limited to a few when high speed is required.

In addition to extremely low power consumption, CMOS has the unique feature that CMOS logic networks work reliably even if power supply voltage fluctuates, temperature changes over a wide range, or there is plenty of noise interference. This makes use of CMOS appropriate in rugged environments, such as for automobile, in factories, and weapons.

Problem of Transfer Curve Shift

Unfortunately, when a CMOS logic gate has many inputs, its transfer curve (which shows the relationship between input voltage and output voltage of a CMOS logic gate) shifts, depending on how many of the inputs change values. For example, in the two-input NAND gate shown in Fig. 36.3(a), the transfer curve for the simultaneous change of the two inputs (1 and 2) is different from that for the change of only input 1, with input 2 kept at a high voltage. This is different from an nMOS logic gate (or a pMOS logic gate) discussed in the previous sections, where every driver MOSFET in its conductive state must have a much lower resistance than the load MOSFET in order to have a sufficiently large voltage swing. But if only input 1 in Fig. 36.3(a), for example, changes, the resistance of the pMOS subcircuit is twice as large as that for the simultaneous change of the two inputs 1 and 2; so parasitic capacitance, C, is charged in a shorter time in the latter case. Other examples are shown in (b) and (c) in Fig. 36.3. Because of this problem of transfer curve shift, the number of inputs to a CMOS logic gate is practically limited to four if we want to maintain good noise immunity. If we need not worry about noise immunity, the number of inputs to a CMOS logic gate can be greater.

36.2 Logic Design of CMOS Networks

The logic design of CMOS networks can be done in the same manner as that of nMOS logic networks, because the nMOS subcircuit in each CMOS logic gate, with the pMOS subcircuit regarded as a variable load, essentially performs the logic operation, as seen from Fig. 36.2. The design procedures discussed for nMOS networks in Chapter 30, Section 30.3 can be used more effectively than in the case of nMOS networks, because more than four MOSFETs can be in series inside each logic gate, unless we are concerned about the transfer-curve shift problem or high-speed operation. Also, an appropriate use of transmission gates, discussed in the next paragraph, often simplifies networks.

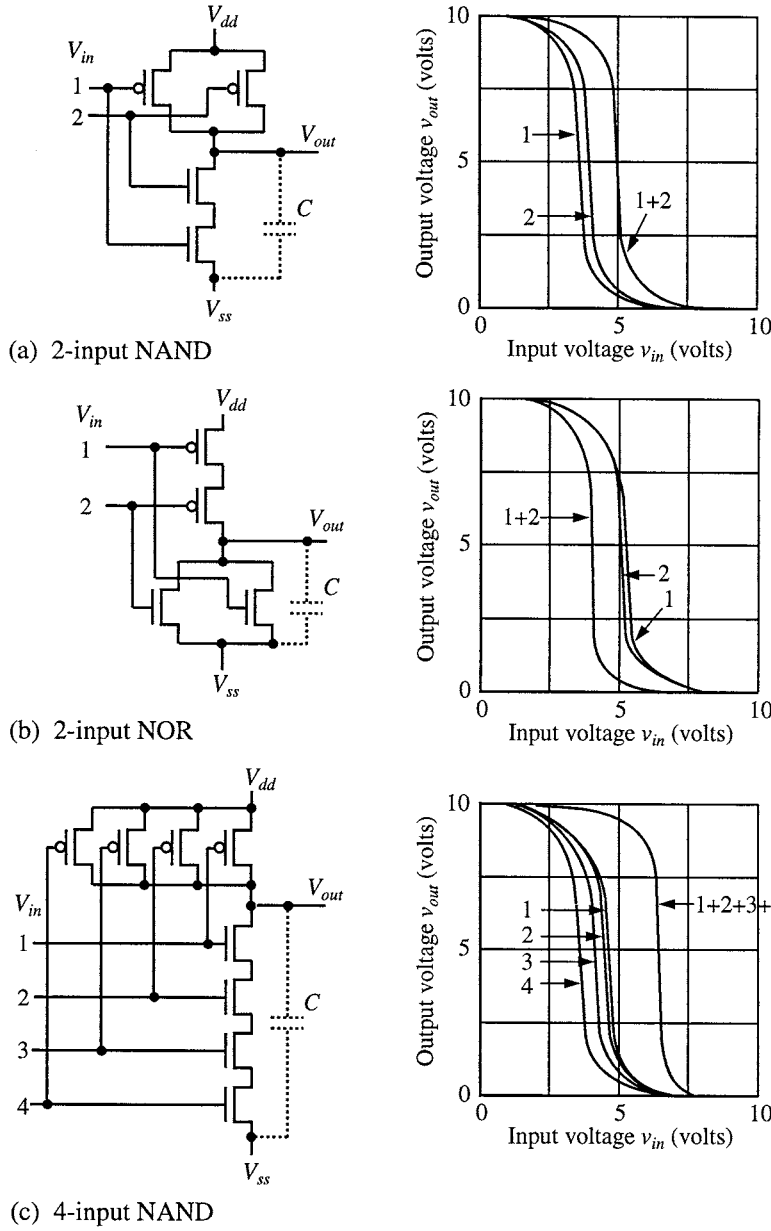


FIGURE 36.3 CMOS transfer curves for different numbers of inputs.

The **transmission gate** shown in Fig. 36.4 is a counterpart of the pass transistor (i.e., transfer gate) of nMOS, and is often used in CMOS network design. It consists of a pair of p-channel and n-channel MOSFETs. The control voltage d is applied to the gate of the n-channel MOSFET, and its complement $\bar{d}$ is applied to the gate of the p-channel MOSFET. If d is a high voltage, both MOSFETs become conductive, and the input is connected to the output. (Unlike a pass transistor with nMOS whose output voltage is somewhat lower than the input voltage, the output voltage of the transmission gate in CMOS is the same as the input voltage after the transition period.) If d is a low voltage, both MOSFETs become non-conductive, and the output is disconnected from the input, keeping the output

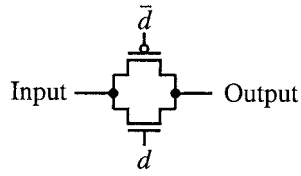


FIGURE 36.4 Transmission gate.

voltage (which gradually becomes low because of current leakage) at its parasitic capacitance, as it was before the disconnection. Since the input and output are interchangeable, the transmission gate is bidirectional. A D-type flip-flop is shown in Fig. 36.5, as an example of CMOS circuits designed with transmission gates.

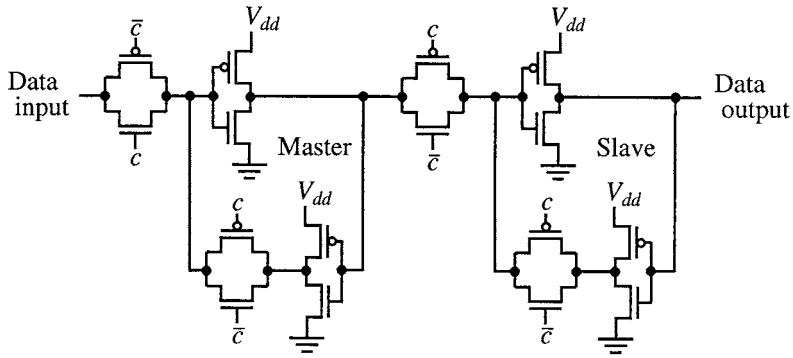


FIGURE 36.5 D-type flip-flop (c is a clock).

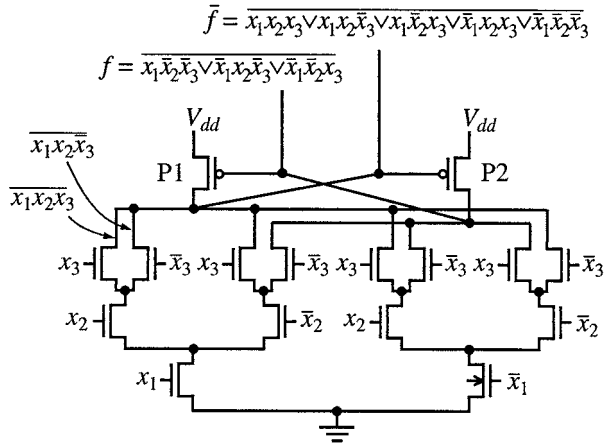
A full adder in CMOS with transmission gates is shown in Chapter 38. Also, pass transistors realized in nMOS can been used mixed with CMOS logic gates to reduce area or power consumption, as will be discussed in Chapter 37.

36.3 Logic Design in Differential CMOS Logic

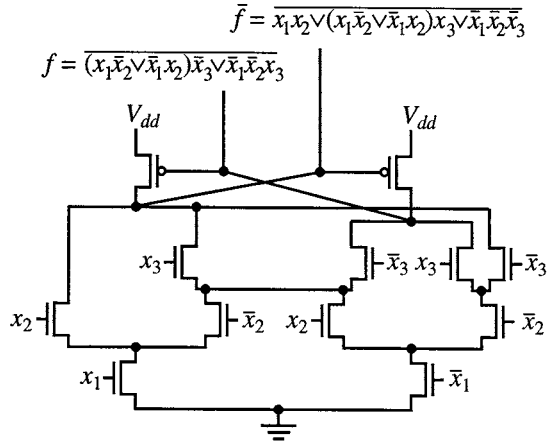
Differential CMOS logic is a logic gate that works very differently from the CMOS logic gate discussed so far. It has two outputs, f and its complement, $\bar{f}$, and works like a flip-flop such that when one output is a high voltage, it always makes the other output have a low voltage.

A logic gate in **cascode voltage switch logic**, which is abbreviated as **CVSL**, is illustrated in Fig. 36.6. CVSL is sometimes called **differential logic** because CVSL is a CMOS logic gate that realizes both an output function, f, and its complement, $\bar{f}$, switching their values quickly. The CVSL gate shown in Fig. 36.6(a) has a driver that is a tree consisting of nMOSFETs, where each pair of nMOSFETs in one level has input x_i and its complement $\bar{x}_i$. The top end of each path in the tree expresses the complement of a minterm (just like series-gating ECL described in Chapter 35). Then by connecting some of these top ends to both the gate of one pMOSFET (i.e., P1), and the drain of the other pMOSFET (i.e., P2), we can realize the complement of a sum-of-products. The connection of the remaining top ends to both the gate of P2 and the drain of P1 realizes its complement. The outputs in Fig. 36.6(a) realize

$$f = \overline{x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1x_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2x_3}$$



(a)



(b)

FIGURE 36.6 (a) Static CVSL; (b) Static CVSL for the same functions as in (a).

and

$$\bar{f} = \overline{x_1x_2x_3 \vee x_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3 \vee \bar{x}_1x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3}$$

When one path in the tree is conductive, the output of it, say the output f, has a lower voltage, and P1 (i.e., the pMOSFET whose gate is connected to f) becomes conductive. Then, the other output $\bar{f}$ has a high voltage. The driver tree in Fig. 36.6(a), which resembles series-gating ECL, can be simplified as shown in (b). (The tree structure in CSVL in (b) can be obtained from an ordered reduced binary decision diagram described in Chapters 23 and 26.) Notice that P1 and P2 in (a) are cross-connected for fast switching.

CVSL can be realized without tree structure. Figure 36.7 shows such a CSVL gate, for $f = \overline{xy}$ and its complement. The connection configuration of nMOSFETs connected to one output terminal is dual to that connected to the other output terminal (in Fig. 36.7, nMOSFETs for the output terminal for f are connected in series, while nMOSFETs for $\bar{f}$ are connected in parallel).

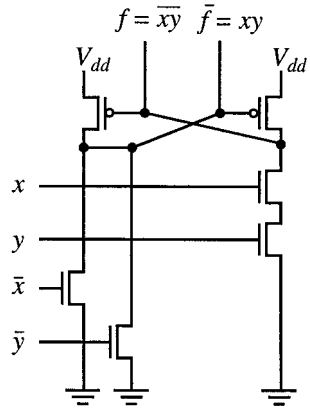


FIGURE 36.7 Static CVSL.

CVSL has a variant called dynamic CVSL, to be described later. In order to differentiate from this, CVSL here is usually called **static CVSL**.

36.4 Layout of CMOS

Layout of CMOS logic networks is more complex than logic networks where only nMOSFETs or pMOSFETs are used because in CMOS networks, nMOSFETs and pMOSFETs need to be fabricated with different materials. This makes layout of CMOS complicated. We often need to consider appropriate connection configuration of MOSFETs inside each logic gate and a logic network such that speed or area is optimized.

When we want to raise the speed, the switching speed of the pMOS subcircuit should be comparable to that of the nMOS subcircuit. Since the mobility of electrons is roughly 2.5 times higher than that of holes, the channel width W of p-channel MOSFETs must be much wider (often 1.5 to 2 times because the width can be made smaller than 2.5 times due to different doping levels) than that of n-channel MOSFETs (which work based on electrons) in order to compensate for the low speed of p-channel MOSFETs (which work based on holes) if the same channel length is used in each. Thus, for high-speed applications, NAND logic gates such as Fig. 36.3(a) are preferred to NOR logic gates, because the channel width of p-channel MOSFETs in series in a NOR logic gate such as Fig. 36.3(b) must be further increased such that the parasitic capacitance C can be charged up in a shorter time with low series resistance of these p-channel MOSFETs.

When designers are satisfied with low speed, the same channel width is chosen for every p-channel MOSFET as for n-channel MOSFETs, since a CMOS logic gate requires already more area than a pMOS or nMOS logic gate and a further increase by increasing the channel width is not desirable unless really necessary.

36.5 Pseudo-nMOS

In the case of the CMOS discussed so far, each CMOS logic gate consists of a pair of pMOS and nMOS subcircuits which realize dual transmission functions, as explained with Fig. 36.2. In design practice, however, some variations are often used. For example, Fig. 36.8 realizes NOR. The pMOS subcircuit consists of only a single MOSFET. Thus, the chip area is reduced and the speed is faster than static CMOS discussed so far because of a smaller parasitic capacitance. But more power is dissipated for some combinations of input values[1] because the pMOS subcircuit in (a) is always conductive.[9,10]

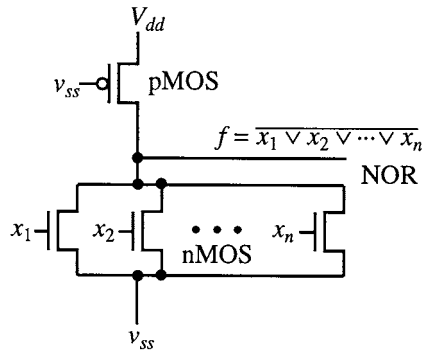


FIGURE 36.8 Peudo-nMOS.

36.6 Dynamic CMOS

Dynamic CMOS[4] is a CMOS logic gate that works in a manner very different from the CMOS logic gates discussed so far, which are usually called static CMOS. Using clock pulse, a parasitic capacitance is precharged, and then it is evaluated whether the parasitic capacitance is discharged or not, depending on the values of input variables. Dynamic CMOS has been used often for high speed because parasitic capacitance can be made small due to unique connection configuration of MOSFETs inside a logic gate, although good layout is not easy. Power consumption is not necessarily small. (In static CMOS, a current flows from the power supply to the ground during transition period. But in dynamic CMOS, there is no such current. But this does not mean that dynamic CMOS consumes less power because when input variables do not change their values, static CMOS does not consume power at all, whereas dynamic CMOS may consume power by repeating precharging. Dynamic CMOS and static CMOS have completely different power consumption mechanisms.)

Domino CMOS

Domino CMOS, illustrated in Fig. 36.9, consists of pairs of a CMOS logic gate and an inverter CMOS logic gate.[4] The first CMOS logic gate in each pair (such as logic gates 1, 3, and 5) has the pMOS subcircuit, consisting of a single pMOSFET with clock, and the nMOS subcircuit, consisting of many nMOSFETs with logic inputs and a single nMOSFET with a clock. The first CMOS logic gate is followed by an inverter CMOS logic gate (such as logic gates 2, 4, and 6). When a clock pulse is absent at all terminals labeled c, all parasitic capacitances (shown by dotted lines) are charged to value 1 (i.e., a high voltage) because all pMOSFETs are conductive. This process is called **precharging**. Thus, the outputs of all inverters become value 0. Suppose that $x = v = 1$ (i.e., a high voltage) and $y = z = u = 0$ (i.e., a low voltage). When a clock pulse appears, that is, $c = 1$, all pMOSFETs become non-conductive but the nMOS subcircuit in each of logic gates 1 and 5 becomes conductive, discharging parasitic capacitance. Then the outputs of logic gates 1, 2, 3, 4, 5, and 6 become 0, 1, 1, 0, 0, and 1, respectively. Notice that the output of logic gate 3 remains precharged because its nMOSFET for u remains non-conductive. Domino CMOS has the following advantages:

- It has a small area because the pMOS subcircuit in each logic gate consists of a single pMOSFET.
- It is faster (about twice) than the static CMOS discussed so far because parasitic capacitances are reduced by using a single pMOS in each logic gate and the first logic gate is buffered by an inverter. Also, an inverter, such as logic gate 2, has smaller parasitic capacitance at its output because it connects to only nMOSFET in logic gate 3, for example, compared to static CMOS where it connects to both pMOSFET and nMOSFET in each of next static CMOS logic gates, to which the output of this static CMOS is connected. This also makes domino CMOS faster.

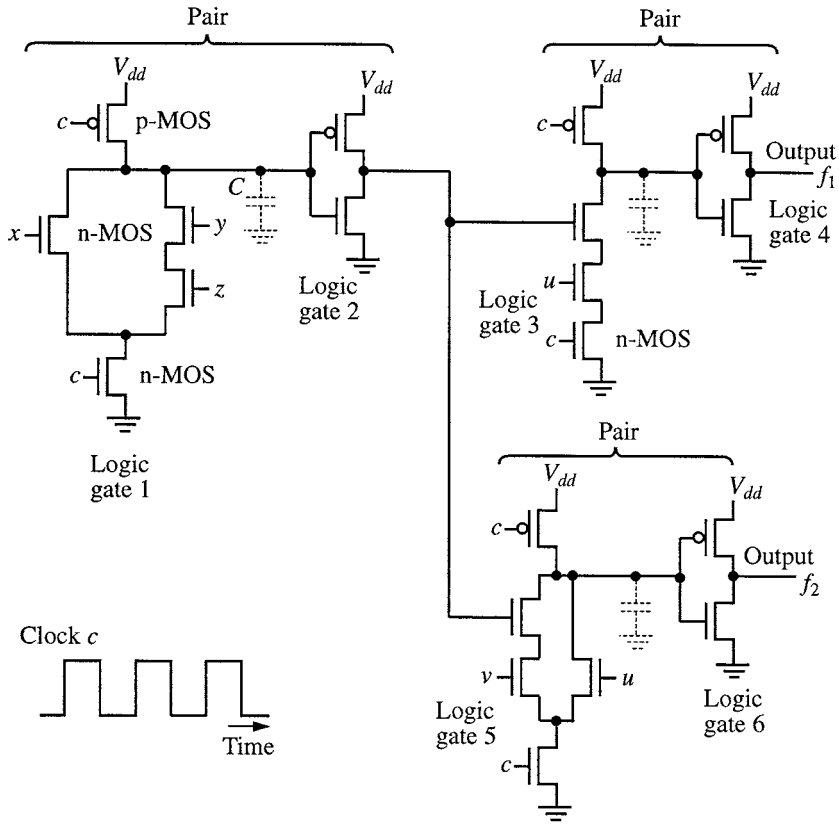


FIGURE 36.9 Domino CMOS.

- It is free of glitches (i.e., transition is smooth) because at the output of each logic gate, a high voltage remains or decreases, but no voltage increases from low to high.[7]

Domino CMOS has the following disadvantage:

- Only positive functions with respect to input variables can be realized. (If both x_i and $\bar{x}_i$ for each x_i is available as network inputs, the network can realize any function. But if only one of them, say x_i, is available, functions that are dependent on $\bar{x}_i$ cannot be realized by a domino CMOS logic network.)

So we have to have domino CMOS networks in double-rail input logic (e.g., Ref. 2), or to add inverters, whenever necessary. Thus, although the number of MOSFETs in domino CMOS networks in single-rail input logic, such as Fig. 36.9, is almost half of static CMOS networks, the number of MOSFETs in such domino CMOS networks to realize any logic functions may become comparable to the number of MOSFETs in static CMOS networks in single-rail input logic.

Dynamic CVSL

Static CVSL, which is previously described, can be easily converted into **dynamic CVSL** which is faster, as illustrated in Fig. 36.10. The parasitic capacitance of two output terminals are precharged through each of the two pMOSFETs during the absence of a clock pulse. Dynamic CVSL works in a similar manner to domino CMOS. Notice that two pMOSFETs are not cross-connected like static CVSL and

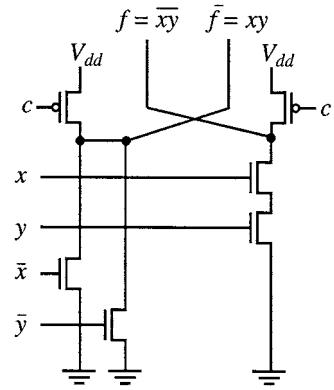


FIGURE 36.10 Dynamic CVSL.

we have essentially two independent logic gates. Dynamic CSVL with two outputs, f and $\bar{f}$, is in double-rail logic, so unlike domino CMOS, not only positive functions but also any logic functions can be realized. It is fast because the pMOS subcircuit of static CMOS is replaced by one pMOSFET and consequently parasitic capacitance is small and also because the output of a dynamic CSVL is connected only to the nMOS subcircuits, instead of to both the nMOS and pMOS of a next static CMOS logic gate. It is also free of glitches.

Problems of Dynamic CMOS

Dynamic CMOS, such as domino CMOS and differential CMOS logic, is increasingly important for circuits that require high speed, such as arithmetic/logic units,[5] although design and layout of appropriate distribution of voltages and currents are far trickier than static CMOS. Dynamic CMOS with a single-phase-clock has advantage of simple clock distribution lines.[3]

References

1. Cooper, J. A., J. A. Copland, and R. H. Krambeck, "A CMOS microprocessor for telecommunications applications," *ISSCC '77*, pp. 137-138.
2. Heikes, C., "A 4.5mm^2 multiplier array for a 200MFLOP pipelined coprocessor," *ISSCC '94*, pp. 290-291. (Double-rail domino CMOS.)
3. Ji-Ren, Y., I. Karlsson, and C. Svensson, "A true single-phase-clock dynamic CMOS circuit technique," *IEEE JSSC*, pp. 899-901, Oct. 1987.
4. Krambeck, R. H., C. M. Lee, and H.-F. S. Law, "High-speed compact circuits with CMOS," *IEEE JSSC*, pp. 614-619, June 1982. (First paper on domino CMOS)
5. Lu, F. and H. Samueli, "A 200-MHz CMOS pipelined multiplier-accumulator using quasi-domino dynamic full-adder cell design," *IEEE JSSC*, pp. 123-132, Feb. 1993.
6. Muroga, S., *VLSI System Design*, John Wiley & Sons, 1982.
7. Murphy, et al., "A CMOS 32b single chip microprocessor," *ISSCC '81*, pp. 230, 231, 276.
8. Shoji, M., *CMOS Digital Circuit Technology*, Prentice-Hall, 1988.
9. Vyemura, J. P., *CMOS Logic Circuit Design*, Kluwer Academic Publishers, 1999.
10. Weste, N. H. E. and K. Eshraghian, *Principles of CMOS VLSI Design*, 2nd ed., Addison Wesley, 1993.

Yano, K., Muroga, S. "Pass Transistors"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

37

Pass Transistors

Kazuo Yano
Hitachi Ltd.

Saburo Muroga
University of Illinois at Urbana-Champaign

37.1 Introduction

A MOSFET is usually used such that a voltage at the gate terminal of the MOSFET controls a current between its source and drain. When the voltages at the gate terminals of MOSFETs that constitute a logic gate are regarded as input variables x, y, and so on, the voltage at the output terminal represents the output function f. Here, $x = 1$ means a high voltage and $x = 0$ a low voltage. For example, MOSFETs in the logic gate in CMOS shown in Fig. 37.1(a) are used in this manner and the output function f represents $\overline{x \vee y}$. But a MOSFET can be used such that an input variable x applied at one of source and drain of the MOSFET is delivered to the other or not, depending on whether a voltage applied at the gate terminal of the MOSFET is high or low. For example, the n-channel MOSFET shown in Fig. 37.1(b) works in this manner and the MOSFET used in this manner is called a **transfer gate** or, more often, a **pass transistor**. When the control voltage c is high, the MOSFET becomes conductive and the input voltage x appears at the output terminal f (henceforth, let letters, f, x, and others represent terminals as well as voltages or signal values), no matter whether the voltage x is high or low. When the control voltage c is low, the MOSFET becomes non-conductive and the input voltage at x does not appear at the output terminal f.

Pass transistors have been used for simplification of transistor circuits. Logic functions can be realized with fewer MOSFETs than logic networks of logic gates where MOSFETs are used like those in Fig. 37.1(a). The circuit in Fig. 37.2, for example, realizes the even-parity function $\overline{x \oplus y}$[2]. This circuit works in the following manner. When x and y are both low voltages, n-channel MOSFETs, 1 and 2, are non-conductive and consequently no current flows from the power supply V_{dd} to the terminal x or y. Thus, the output voltage f is high because it is the same as the power supply voltage at V_{dd}. When x is a low voltage and y is a high voltage, MOSFET 1 becomes conductive and 2 is non-conductive. Consequently, a current flows from the power supply to x because x is a low voltage. Thus, the output voltage at f is low. Continuing the analysis, we have the truth table shown in Fig. 37.3(a). Then we can derive the truth table for function $f = \overline{x \oplus y}$ in Fig. 37.3(b) by regarding a low and high voltages as 0 and 1, respectively. A logic gate for this function in CMOS requires 8 MOSFETs, as shown in Fig. 37.4, whereas the circuit realized with pass transistors in Fig. 37.2 requires only three MOSFETs. Notice that inputs x and y are connected to the sources of MOSFETs 1 and 2, unlike MOSFET in ordinary logic gates. Signal x at the source of MOSFET 1 is either sent to the drain or not, according to whether or not its MOSFET gate has a high voltage.

Pass transistors, however, are sometimes used inside an ordinary logic gate, mixed with ordinary MOSFETs. MOSFETs 1, 2, and 3 in Fig. 37.5 are such pass transistors. (Actually, the pair of 1 and 2 is a transmission gate to be described in the following and also in Chapter 36, Section 36.2) Logic networks

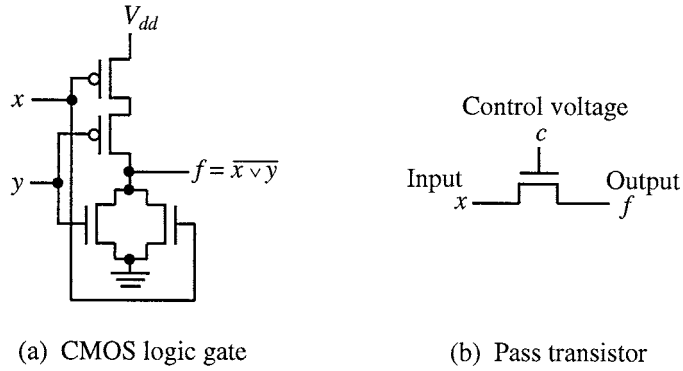


FIGURE 37.1 CMOS logic gate and pass-transistor circuit.

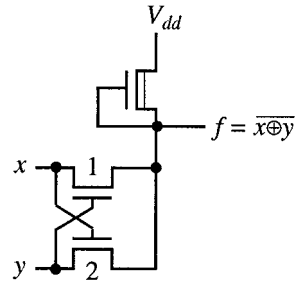


FIGURE 37.2 Circuit with pass transistors for the even-party function.

(a) Voltage relationship

x	y	f
Low volt.	Low volt.	High volt.
Low volt.	High volt.	Low volt.
High volt.	Low volt.	Low volt.
High volt.	High volt.	High volt.

(b) Logic function

x	y	f
0	0	1
0	1	0
1	0	0
1	1	1

FIGURE 37.3 Behavior of the circuit in Fig. 37.2.

in CMOS where MOSFETs are used in the same way as those in Fig. 37.1(a) can sometimes be simplified by an appropriate use of pass transistors, possibly with speed-up.

The wide use of a pass transistor is found in the DRAM memory cell, which consists of one capacitor and one pass transistor. The control of charging to or discharging from the memory capacitor is done through the pass transistor. This shows the impact of this technique in area reduction, power consumption reduction, and possibly also in speed-up. Pass transistors are also used in the arithmetic-logic unit of a computer, which requires speed and small area, such as fast adders (actually the logic gate in Fig. 37.5 is part of such an adder), multipliers and multiplexers.[3,7,11,12]

The circuit in Fig. 37.6(a) in double-rail input logic (i.e., x, $\bar{x}$, y, and $\bar{y}$ are available as inputs) realizes the odd-parity function $x \oplus y$. A circuit for the inverter shown by the triangle with a circle in Fig. 37.6(a) is shown in (b).

Pass transistors are often used for forming a demultiplexer, as illustrated in Fig. 37.7. Series connection of pass transistors has some resistance. So the number of control variables (here, x and y) is limited to at most four and the inverter shown in Fig. 37.6(b) is added after input g as a buffer.

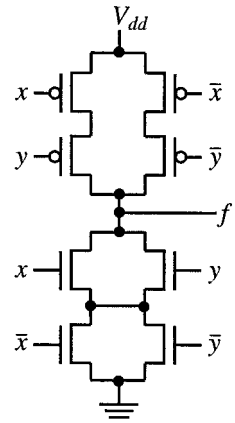


FIGURE 37.4 CMOS logic gate for function $f = \overline{x \oplus y}$

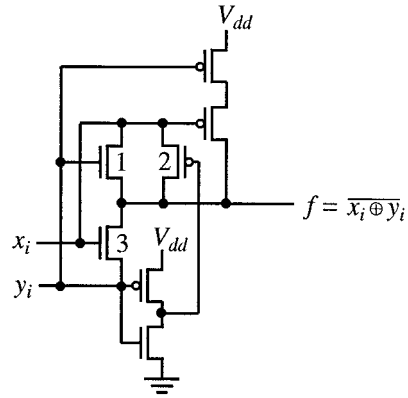


FIGURE 37.5 Pass transistors in a logic gate.

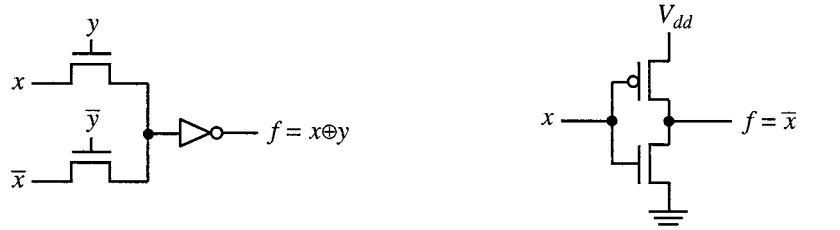


FIGURE 37.6 Parity function realized with pass transistors.

Using pass transistors, a latch (more precisely speaking, a D latch to store data) can be constructed as shown in Fig. 37.8. When control input c is a high positive voltage, the feedback loop is cut by the pass transistor with $\bar{c}$, and the input value is fed into the cascade of two inverters. When c becomes a low voltage, the input is cut off and the loop that consists of two inverters and one pass transistor retains the information.

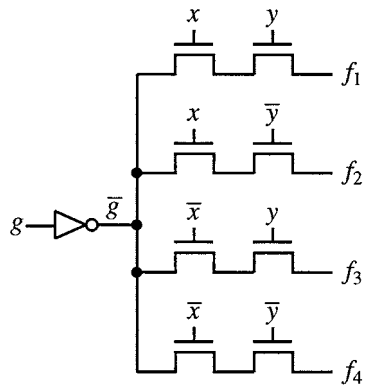


FIGURE 37.7 Demultiplexer with pass transistors.

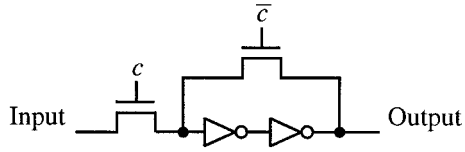


FIGURE 37.8 Latch with pass transistors.

The use of pass transistors in logic networks, however, has been very limited because of electronics problems to be discussed in the following. The majority of commercially available logic networks have been in static CMOS circuits. But as higher speed and smaller area are desired, this situation is gradually changing. The pass-transistor logic has recently attracted much attention under these circumstances and is anticipated to be widely used in the near future for its area/power saving and high-performance benefits.

Beside pass transistors, there are many other unconventional MOS networks. All these networks are useful for simplification of electronic realization of logic networks or for improvement of performance, although complex adjustments of voltages or currents are often required.

37.2 Electronic Problems of Pass Transistors

Suppose an n-channel MOSFET is used as a pass transistor, as shown in Fig. 37.9. Then, the MOSFET behaves electronically as follows. When the control voltage at *c* is high, the voltage at the input *x* is delivered to the output terminal *f*, no matter whether the voltage at the input *x* is high or low. But if the voltage at the input *x* is high, a current flows through the MOSFET to charge up the parasitic capacitance (shown in the dotted lines in Fig. 37.9) at *f* and stops when the difference between the voltage at *f* and the voltage at *c* reaches the threshold voltage, making the voltage at *f* somewhat lower than the

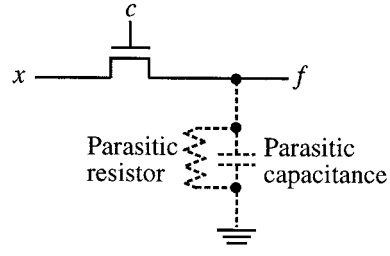
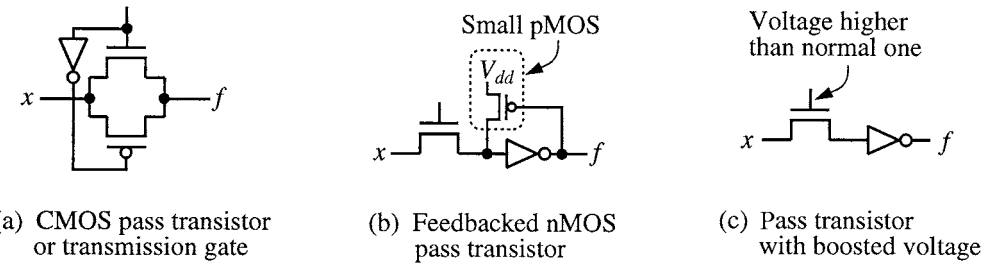


FIGURE 37.9 Electronic behavior of a pass transistor.

voltage at x. When the voltage at c becomes low, the pass transistor becomes non-conductive and the electric charge stored on the parasitic capacitance gradually leaks to the ground through the parasitic resistor. If the voltage at the input x is low when the control voltage at c is high, the electric charge stored on the parasitic capacitance flows to the ground through the pass transistor and input terminal x also if it has not completely leaked to the ground yet.

This complex electronic behavior of the pass transistor makes a circuit with pass transistors unreliable. The intermediate value of the voltage at f, which is lower by the threshold voltage than the voltage at x or partially leaked voltage, causes unpredictable operations of the logic network when the voltage at f is fed to ordinary CMOS logic gates in the next stage. Moreover, it degrades the switching speed of the CMOS logic gates. In the worst case, the circuit loses noise margin or it does not operate properly.

There are three techniques to avoid this drawback. The first one is to combine an nMOS pass transistor and a pMOS pass transistor in parallel, as shown in Fig. 37.10(a). With this technique, when the pass transistor is conductive, the output voltage at f reaches exactly the same value as the input voltage at x, no matter whether the input voltage is high or low. This pair of nMOS and pMOS pass transistors is sometimes called a **transmission gate**. Although this has better stability over the pass transistor circuit in Fig. 37.9, it consumes roughly twice as large an area.



(a) CMOS pass transistor or transmission gate

(b) Feedbacked nMOS pass transistor

(c) Pass transistor with boosted voltage

FIGURE 37.10 Techniques to avoid the influence of the low output voltage.

The second approach is to use a pMOS feedback circuit at the output of the nMOS pass transistor, as shown in Fig. 37.10(b). The gate of a p-channel MOSFET is driven by the CMOS inverter (shown as the triangle with a small circle), which works as an amplifier. When the CMOS inverter discharges the electric charge at the output, it also turns on the feedback pMOS to raise the pass transistor output to the power supply voltage, eliminating the unreliable operation. One limitation of this approach is that it does not solve the degradation of switching speed due to low voltage, because the speed is determined by the initial voltage swing before the pMOS turns on. Area increase with this approach is smaller than the transmission gate in Fig. 37.10(a).

The third approach is to raise the gate voltage swing up to the normal voltage plus threshold voltage, which is used in DRAM and referred to as "word boost," as shown in Fig. 37.10(c). This approach requires a boost driver every time the gate signal is generated, which is difficult to use in logic functions in general. In addition, a voltage that is higher than the power supply voltage is applied to the gate of a MOSFET, which requires special care against breakdown and reliability problems (these need to be solved by increasing the thickness of gate insulation).

Another important consideration for pass-transistor operation is how many pass transistors can be connected in series without buffers. Many pass transistors connected in series can be treated as a serially connected resister-capacitor circuit. The delay of this RC (resistor-capacitor) circuit, which is proportional to the product of R and C, becomes roughly four times larger when both R and C are doubled. Thus, the delay of this circuit is proportional to the square of the number of pass transistors. This means that it is not beneficial to increase the number of pass transistors too many. However, short-pitch insertion of CMOS inverters increases the delay overhead of the buffers themselves. Empirically, the optimal pass-transistor stages for delay minimization is known to be about two to three. In design practice, the number

of pass transistors cannot be arbitrarily chosen because designers want to have a certain number of fan-out connections and a buffer cannot support too many fan-outs and pass transistors. Also, the structure of a logic network cannot be arbitrarily chosen because of area size and power consumption.

37.3 Top-down Design of Logic Functions with Pass-Transistor Logic

After designing logic networks manually or by CAD programs, computer systems have been designed. This is called a bottom-up design approach. In the 1990s, so-called top-down design has been accepted as the mainstream design approach. In the top-down logic design, register-transfer-level functionality is described with a hardware-description language, such as Verilog-HDL and VHDL (Very High Speed Integrated Circuit Hardware Description Language) rather than directly designing gate-level structure of logic networks, or "netlist." And then, this is converted to logic networks of logic gates using a logic synthesizer (i.e., CAD programs for automated design of logic networks). This process resembles the compilation process of the software construction and it is sometimes referred to as "compile." Based on this netlist, placement and routing of transistors are done automatically on an IC chip. By using this top-down approach, a logic designer can focus on the functional aspect of the logic rather than the in-depth structural aspect. This enhances the productivity. Also, this enables one to easily port one design in one technology to another.

Automated design of logic networks with pass transistors has been difficult to realize because of complex electronic behavior. So, conventionally, pass-transistor logic has been manually designed, particularly in arithmetic modules as shown in this section. But as reduction of power consumption, speedup or area reduction is strongly desired, this is changing. Logic design based on selectors with pass-transistors can be done in this top-down manner.[10] Pass transistors have been used often as a selector by combing two pass transistors, as shown in Fig. 37.11(a). A selector is also called a multiplexer. The output f of the selector becomes input x when $c = 1$ and input y when $c = 0$. Figure 37.11(b) shows a selector realized in a logic gate and also in pass transistors. Compared with the selector in a CMOS logic gate shown on the left side of Fig. 37.11(b) which consists of ten MOSFETs, the selector in pass transistors shown on the right side of Fig. 37.11(b) consists of only four MOSFETs, reducing the number of MOSFETs to less than half, and consequently the area. A selector is known to be a universal logic element because

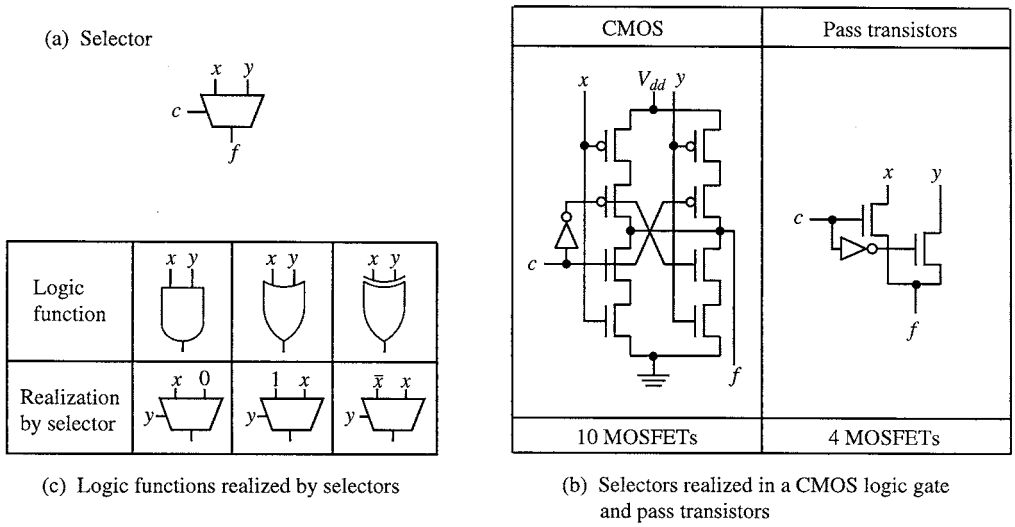


FIGURE 37.11 Selectors and various logic functions realized by pass transistors.

it can be used as an AND, an OR, and an XOR (i.e., Exclusive-OR) by changing its inputs, as shown in Fig. 37.11(c). This property is also useful in the top-down design approach discussed in the following. The speed of a logic network with pass transistors is sometimes improved up to 2 times better than a conventional CMOS logic network, depending on logic functions.

One limitation of this pass-transistor selector is that it suffers a relatively slow switching speed when the control signal arrives later than selected input signals. This is because an inverter is needed for the selector to have a complementary signal applied to the gate of a pass transistor.

To circumvent this limitation, CPL (which stands for the complementary pass-transistor logic) has been conceived.[11,12] In CPL, complementary signals are used for both inputs and outputs, eliminating the need for the inverter. The circuits that require complementary signals like CPL are sometimes categorized as dual-rail logics. Because of the need for complementary signal, CPL is sometimes twice as large as CMOS, but is sometimes surprisingly small if a designer succeeds in fully utilizing the functionality of a pass-transistor circuit. A very fast and compact CPL full adder, a multiplier, and a carry-propagate-chain circuit have been reported. A full adder realized with CMOS logic gates is compared with a full adder realized with selectors in pass transistors in Fig. 37.12. Speed and power consumption are significantly improved.

Variants of CPL have been also reported, including DPL,[8] SRPL,[5] and SAPL.[4]

However, conventional switching theory, on which widely used logic synthesizers are based, cannot be conveniently used for this purpose because it is very difficult to derive convenient logic expressions based on the output function $xc \vee y\bar{c}$ of the selector. Instead, BDD (i.e., binary decision diagrams) are used, as follows.

A simple approach to use a selector as the basic logic element is to build a binary tree of pass-transistor selectors, as shown in Fig. 37.13. The truth table shown in Fig. 37.13(a) is directly mapped into the tree structure shown in Fig. 37.13(b). When $x = 1$, $y = 0$, and $z = 1$, for example, the third 1 from the left in the top of Fig. 37.13(b) is connected to the output as $f = 1$. This original tree generally has redundancy, so it should be reduced to an irredundant form as shown in Fig. 37.13(c). This approach is simple and effective when the number of input variables is less than 5 or so. However, this does not work for functions with more input variables, because of the explosive increase of the tree size.

To solve this, a binary decision diagram (i.e., BDD), has been utilized.[10] Basic design flow of BDD-based pass-transistor circuit synthesis is shown in Fig. 37.14. The logic expressions for functions f_1 and f_2 shown in Fig. 37.14(a) are converted to the BDD in (b). Then, buffers (shown as triangles) are inserted in Fig. 37.14(c). In this case, only locations where the buffers should be inserted in Fig. 37.14(d) are specified and the nature of the BDD in Fig. 37.14(c) is not changed. In both Figs. 37.14(b) and (c), each solid line denotes the value 1 of a variable and each dotted line the value 0. For example, the downward solid line from the right-hand circle with w inside denotes $w = 1$. From f_1, if we follow dotted lines three times and then the solid line once in each of (b) and (c), we reach the 0 inside the rectangle in the bottom. This means that $f_1 = 0$ for $w = x = y = 0$ and $z = 1$.

Preparation of an appropriate cell library based on selectors is required, as shown in Fig. 37.15, which consists of a simple two-input selector (Cell 1) and its variants (Cells 2 and 3). The inverters shown with a dot inside the triangle in Fig. 37.15, which is different from the simple inverter shown in Fig. 37.6(b), is to keep the electric charge on the parasitic capacitance at its input. In Fig. 37.14(d), the inverters of this kind have to be inserted corresponding to the buffers shown in (c). But in this case, the insertion has to be done such that the outputs f_1 and f_2 have the same polarity in both (c) and (d) because the inverters change signal values from 1 to 0 or from 0 to 1.

In the design flow in Fig. 37.14, starting from the logic functions which are represented with logic equations or a truth table, the logic functions are then converted to a BDD. Each node of the BDD represents two-input selector logic, and, in this way, mapping to the above selector-based cells is straightforward, requiring only consideration of the fan-out and signal polarity.

One difficulty of this approach with BDD is optimization of the logic depth, that is, the number of pass transistors from an input to an output. One important desired capability of a logic synthesizer for

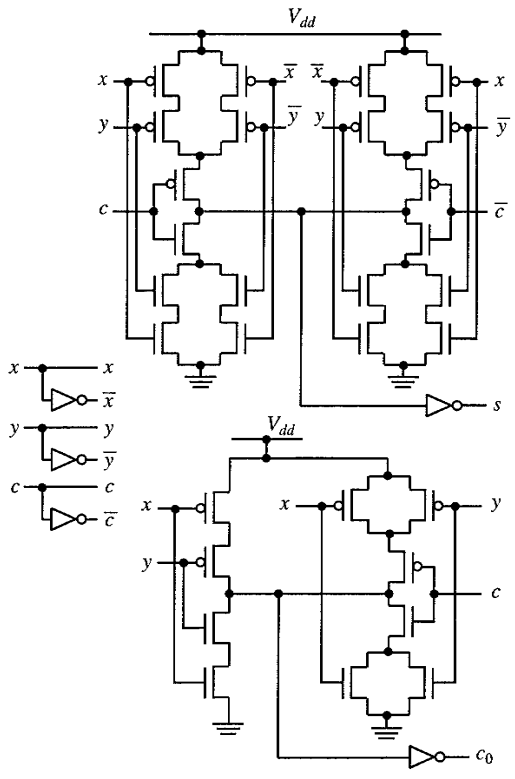
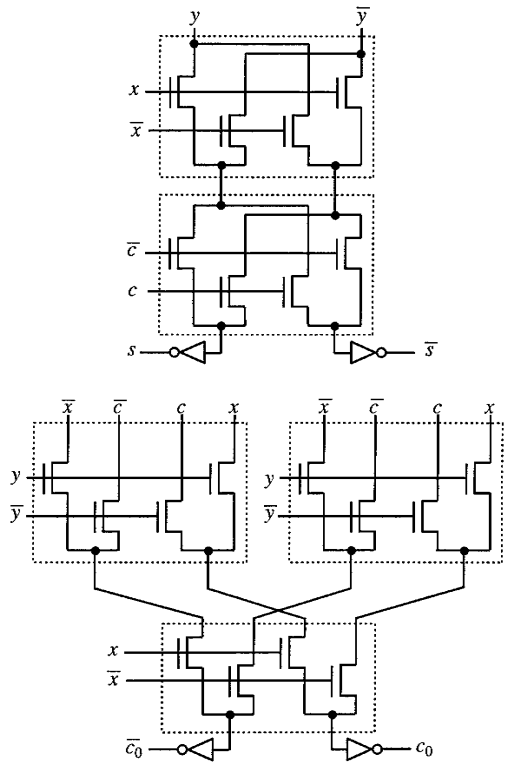
	CMOS	CPL
Full adder circuit		
Transistor count	40	28
Area	4730 μm^2	4218 μm^2
Delay at power supply with 4V	0.63 ns	0.26 ns
Power at clock with 100MHz	1.2 mW	0.86 mW

FIGURE 37.12 Full adder realized in CMOS logic gates and complementary pass-transistor logic (CPL).

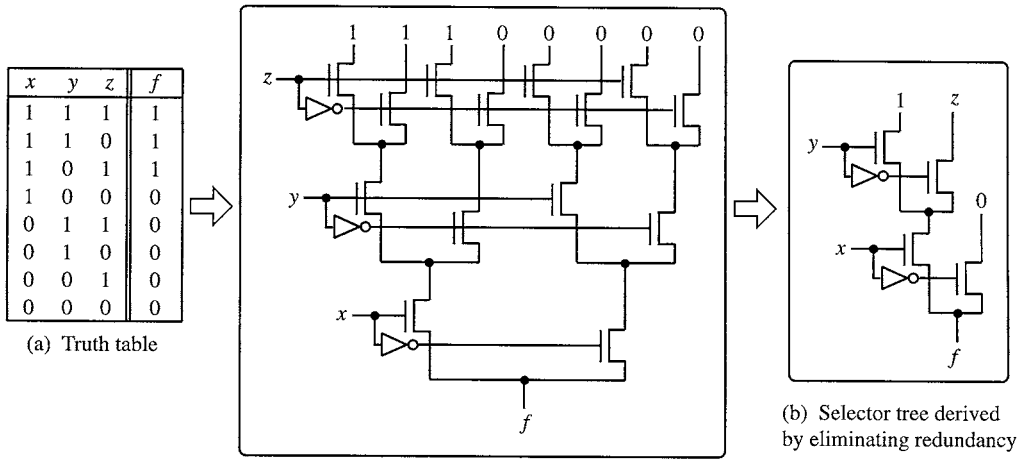


FIGURE 37.13 Binary-tree-based simple construction method of pass-transistor logic.

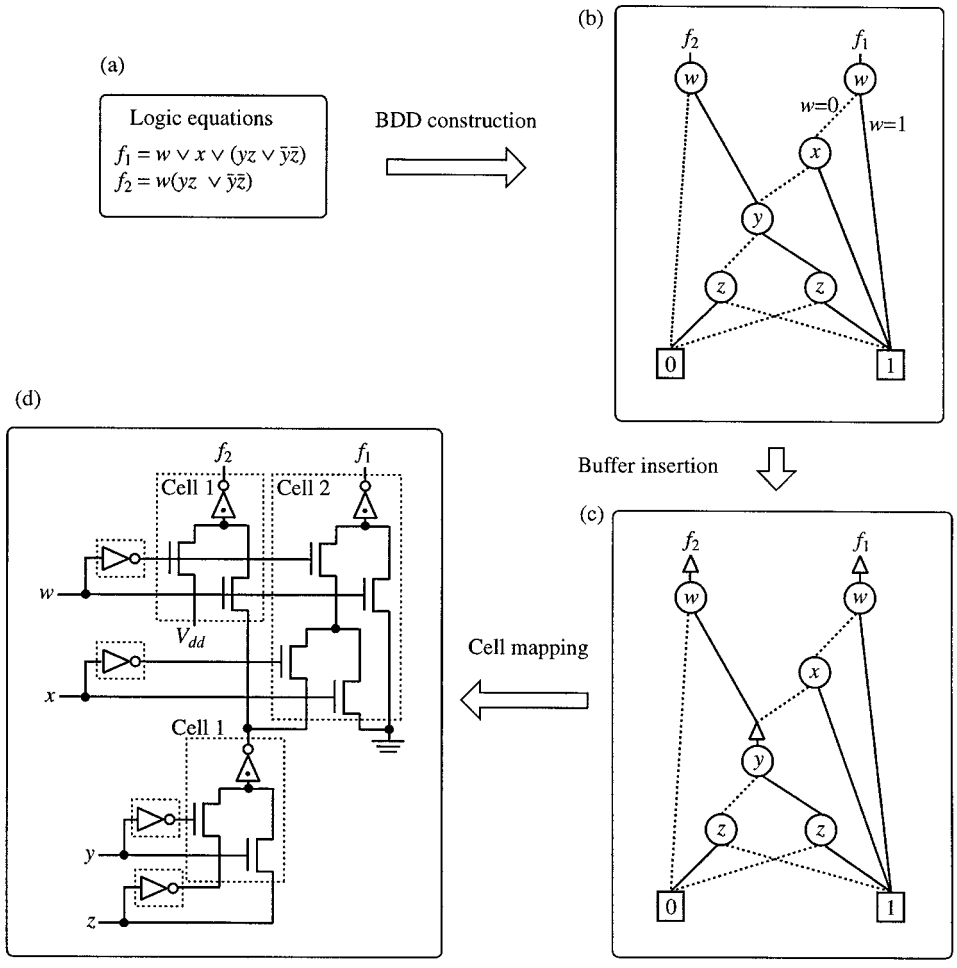


FIGURE 37.14 Design flow of BDD-based pass-transistor logic synthesis.

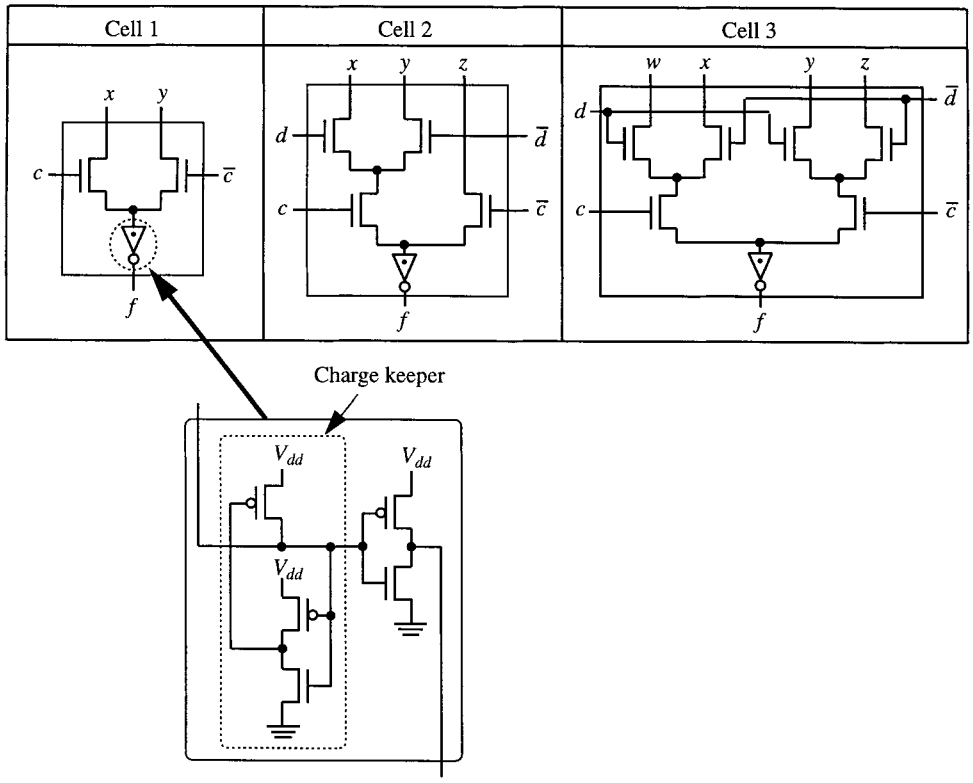


FIGURE 37.15 Pass-transistor-based cell library.

this approach is the control of the logic depth, for example, so that a designer can limit the delay time from an input to an output. It is difficult to incorporate this requirement in the framework of a BDD. Another difficulty of the BDD-based synthesis is that the number of pass transistors connected in series increases linearly as the number of inputs increases and this number may become excessive.

To solve these difficulties, MPL (which stands for multi-level pass-transistor logic) and its representation, multi-level BDD, have been proposed.[6] In the above simple BDD-based approach, the output of a pass-transistor selector is connected only to the source-drain path of another pass-transistor selector. This causes the above difficulty. In MPL, the output of a pass-transistor selector is flexibly connected to either a source-drain path or the gate of another MOSFET. Because of this freedom, the delay of the circuit can be flexibly controlled. It is known empirically that the delay, especially of a logic network having a large number of input variables, is reduced by a factor of 2, compared to the simple BDD approach.

Another important extension of pass-transistor logic is to incorporate CMOS circuits in a logic network.[9] Logic networks based on pass transistors are not always smaller than CMOS logic networks in area, delay, and power consumption. They are effective when selectors fit well to the target logic functions. Otherwise, conventional CMOS logic networks are a better choice. For example, a simple NAND function implemented in CMOS logic network has better delay, area, and power consumption than its pass-transistor-based counterpart. Combining pass-transistor logic and CMOS logic gives the best solution.

Pass-transistor logic synthesis is still not as well developed as CMOS-based logic synthesis. However, even at its current level of development, it has shown generally positive results. In other words, 10 to 30% power reduction is possible, as compared with pure CMOS,[9] showing enough potential[1] to be further exploited in future research.

References

1. A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, SC-27, pp. 473-484, 1992.
2. Frei, A. H., W. K. Hoffman, and K. Shepard, "Minimum area parity circuit building block," *ICCC 80*, pp. 680-684, 1980.
3. K. Kikuchi, Y. Nukada, Y. Aoki, T. Kanou, Y. Endo, and T. Nishitani, "A single-chip 16-bit 25ns realtime video/image signal processor," *1989 IEEE International Solid-State Circuits Conference*, pp. 170-171, 1989.
4. M. Matsui, H. Hara, Y. Uetani, L-S. Kim, T. Nagamatsu, Y. Watanabe, A. Chiba, K. Matsuda, and T. Sakurai, "A 200 MHz 13 mm^2 2-D DCT macrocell using sense-amplifying pipeline flip-flop scheme," *IEEE J. Solid-State Circuits*, vol. 29, pp. 1482-1489, 1994.
5. A. Parameswar, H. Hara, and T. Sakurai, "A high speed, low power, swing restored pass-transistor logic based multiply and accurate circuit for multimedia applications," *IEEE 1994 Custom Integrated Circuits Conference*, pp. 278-281, 1994.
6. Y. Sasaki, K. Yano, S. Yamashita, H. Chikata, K. Rikino, K. Uchiyama, and K. Seki, "Multi-level pass-transistor logic for low-power ULSIs," *IEEE Symp. Low Power Electronics*, pp. 14-15, 1995.
7. Y. Shimazu, T. Kengaku, T. Fujiyama, E. Teraoka, T. Ohno, T. Tokuda, O. Tomisawa, and S. Tsujimichi, "A 50MHz 24b floating-point DSP," *1989 IEEE International Solid-State Conference*, pp. 44-45, 1989.
8. M. Suzuki, N. Ohkubo, T. Yamanaka, A. Shimizu, and K. Sasaki, "A 1.5 ns 32 b CMOS ALU in double pass-transistor logic," *1993 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pp. 90, 91, 267, 1993.
9. S. Yamashita, K. Yano, Y. Sasaki, Y. Akita, H. Chikata, K. Rikino, and K. Seki, "Pass-transistor/CMOS collaborated logic: The best of both worlds," *1997 Symp. VLSI Circuits Digest of Technical Papers*, pp. 31-32, 1997.
10. K. Yano, Y. Sasaki, K. Rikino, and K. Seki, "Top-down pass-transistor logic design," *IEEE J. Solid-State Circuits*, vol. 31, pp. 792-803, 1996.
11. K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8ns CMOS 16×16 multiplier using complementary pass-transistor logic," IEEE *1989 Custom Integrated Circuits Conference*, 10.4.1-4, 1989.
12. K. Yano, T. Yamanaka, T. Nishida, M. Saito, K. Shimohigashi, and A. Shimizu, "A 3.8-ns CMOS 16×16-b multiplier using complementary pass-transistor logic," *IEEE J. Solid-State Circuits*, SC-25, pp. 388-395, 1990.

Takagi, N., et al."Adders"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

38 Adders

Naofumi Takagi
Nagoya University

Haruyuki Tago
Toshiba Semiconductor Company

Charles R. Baugh
C. R. Baugh and Associates

Saburo Muroga
University of Illinois at Urbana-Champaign

38.1 Introduction

Adders are the most common arithmetic circuits in digital systems. Adders are used to do subtraction and also are key components of multipliers and dividers, as described in Chapters 39 and 40. There are various types of adders with different speeds, areas, and configurations. We can select an appropriate one which satisfies given requirements. For the details of adders and addition methods, see Refs. 2, 4–6, 12, 15, 17, and 20.

38.2 Addition in the Binary Number System

Before considering adders, let us take a look at addition in the binary number system.

In digital systems, numbers are usually represented in the binary number representation, although the most familiar number representation to us is the decimal number representation. The binary number representation is with the radix (base) 2 and the digit set {0, 1}, while the decimal number representation is with the radix 10 and the digit set {0, 1, 2, …, 9}. For example, a binary number (i.e., a number in the binary number representation) [1101] represents $1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = 13$, whereas a decimal number (i.e., a number in the decimal number representation) [8093] for example, represents $8\cdot10^3 + 0\cdot10^2 + 9\cdot10^1 + 3\cdot10^0 = 8093$.

In the binary number representation, an integer is represented as $[x_{n-1}x_{n-2}\ldots x_0]$ where each binary digit, called a bit, x_i, is one of the elements of the digit set {0, 1}. The binary representation $[x_{n-1}x_{n-2}\ldots x_0]$ represents the integer $\sum_{i=0}^{n-1} x_i \cdot 2^i$.

By the binary number representation, we can represent not only an integer, but also a number that has a fractional part as well as an integral part, as by the decimal number representation. The binary representation $[x_{n-1}x_{n-2}\ldots x_0.x_{-1}x_{-2}\ldots x_{-m}]$ represents the number $\sum_{i=-m}^{n-1} x_i \cdot 2^i$. For example, [1101.101] represents 13.625. By a binary representation with n-bit integral part and m-bit fractional part, we can represent 2^{n+m} numbers in the range from 0 to $2^n - 2^{-m}$.

Let us consider addition of two binary numbers, $X = [x_{n-1}x_{n-2}\ldots x_0.x_{-1}x_{-2}\ldots x_{-m}]$ and $Y = [y_{n-1}y_{n-2}\ldots y_0.y_{-1}y_{-2}\ldots y_{-m}]$. We can perform addition by calculating the sum at the i-th position, s_i and the carry to the next higher position c_{i+1} from x_i, y_i, and the carry from the lower position c_i according to the truth

TABLE 38.1 Truth Table for One-bit Addition

x_i	y_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

table shown in Table 38.1, successively from the least significant bit to the most significant one, that is, from $i = -m$ to $n - 1$, where $c_{-m} = 0$. Then, we have a sum $S = [s_n s_{n-1} s_{n-2} \ldots s_0 . s_{-1} s_{-2} \ldots s_{-m}]$, where $s_n = c_n$. (i.e., s_n is actually a carry c_n from the $(n-1)$-th position).

There are two major methods for representing negative numbers in the binary number representation. One is **sign and magnitude representation**, and the other is **two's complement representation**.

In the sign and magnitude representation, the sign and the magnitude are represented separately, as in the usual decimal representation. The first bit is the sign bit and the remaining bits represent the magnitude. The sign bit is normally selected to be 0 for positive numbers and 1 for negative numbers. For example, 13.625 is represented as [01101.101] and –13.625 is represented as [11101.101]. The sign and magnitude binary representation $[x_{n-1} x_{n-2} \ldots x_0 . x_{-1} x_{-2} \ldots x_{-m}]$ represents the number $(-1)^{x_{n-1}} \cdot \sum_{i=-m}^{n-2} x_i \cdot 2^i$. By the sign and magnitude representation with n-bit integral part (including a sign bit) and m-bit fractional part, we can represent $2^{n+m} - 1$ numbers in the range from $-2^{n-1} + 2^{-m}$ to $2^{n-1} - 2^{-m}$.

In the two's complement representation, a positive number is represented in exactly the same manner as in the sign and magnitude representation. On the other hand, a negative number $-X$, where X is a positive number, is represented as $2^n - X$. For example, –13.625 is represented as [100000.000] – [01101.101], i.e., [10010.011]. The first bit of the integral part (the most significant bit) is 1 for negative numbers, indicating the sign of the number. The binary representation of $2^n - X$ is called the **two's complement of *X***. We can obtain the representation of $-X$, i.e., $2^n - X$ by complementing the representation of X bitwise and adding [0.00…001] (i.e., 1 in the position of 2^{-m} but 0 in all other positions). It is because when $X = [x_{n-1} x_{n-2} \ldots x_0 . x_{-1} x_{-2} \ldots x_{-m}] = \sum_{i=-m}^{n-1} x_i \cdot 2^i$, the negation of X, i.e., $-X$, becomes $2^n - X = (\sum_{i=-m}^{n-1} 2^i + 2^{-m}) - X = \sum_{i=-m}^{n-1} (1 - x_i) \cdot 2^i + 2^{-m}$, where $1 - x_i$ is 1 or 0, according to whether x_i is 0 or 1, i.e., $1 - x_i$ is the complement of x_i. For example, given a binary number [01101.101], we can obtain its negation [10010.011] by complementing [01101.101] bitwise and adding [0.001] to it, i.e., by [10010.010] + [0.001]. By a two's complement representation with n-bit integral part (including a sign bit) and m-bit fractional part, we can represent 2^{n+m} numbers in the range from -2^n to $2^n - 2^{-m}$.

Each of all the binary number representations described so far (i.e., the positive number representation, sign and magnitude representation, and two's complement representation) can express essentially the same numbers, that is, 2^{n+m} numbers, although the second case expresses $2^{n+m} - 1$ numbers (i.e., one number less) when a number is expressed with $n + m$ bits. Thus, these representations essentially do not lose the precision, no matter whether or not one of the $n + m$-bits is used as a sign bit, although the range of the numbers is different in each case.

When we add two numbers represented in the sign and magnitude representation, we calculate the sign and the magnitude separately. When the operands (the augend and the addend) are with the same sign, the sign of the sum is the same as that of the operands, and the magnitude of the sum is the sum of those of the operands. A carry, 1, from the most significant position of the magnitude part indicates overflow. On the other hand, when the signs of the operands are different, the sign of the sum is the same as that of the operand with larger magnitude, and the magnitude of the sum is the difference of those of the operands.

The addition of two numbers represented in the two's complement representation, $X + Y$, for $X = [x_{n-1}x_{n-2}\ldots x_0.x_{-1}x_{-2}\ldots x_{-m}]$ and $Y = [y_{n-1}y_{n-2}\ldots y_0.y_{-1}y_{-2}\ldots y_{-m}]$ where x_{n-1} and y_{n-1} are their sign bits, can be done as follows, no matter whether each of X and Y is a positive or negative number:

1. The sign bits are added in the same manner as the two bits, x_i and y_i, in any other bit position, that is, according to Table 38.1. As illustrated in Fig. 38.1(a), the sign bits, x_{n-1} and y_{n-1}, and the carry, c_{n-1}, from the $(n-2)$-th position are added, producing the sum bit s_{n-1} (in the sign bit position) and the carry c_n. Always, c_n is ignored, no matter whether it is 1 or 0.
2. When $x_{n-1} = y_{n-1}$ (i.e., X and Y have the same sign bit), an overflow occurs if s_{n-1} is different from x_{n-1} and y_{n-1} (i.e., $c_n \oplus c_{n-1} = 1$ means an overflow, as can be seen from Table 38.1). This case is illustrated in Fig. 38.1(b) because we have $s_{n-1} = 0$ while $x_{n-1} = y_{n-1} = 1$ and hence s_{n-1} is not equal to x_{n-1} or y_{n-1}.

(a)

Augend		①	1	1	0	1.	1	0	1	(13.625)
Addend	+	①	1	0	1	0.	1	1	0	(-5.25)
Sum		(1)	①	1	0	0	0.	0	1	1	(8.375)

(x_{n-1}, y_{n-1}, c_n, s_{n-1})

(b)

Augend			1	0	0	1	0.	0	1	1	(-13.625)
Addend	+		1	1	0	1	0.	1	1	0	(-5.25)
Sum		(1)	0	1	1	0	1.	0	0	1	(Overflow)

FIGURE 38.1 Examples of addition of numbers in two's complement representation.

Next let us consider the subtraction of two numbers represented in the two's complement representation, $X - Y$, that is, subtraction of Y from X, where each of X and Y is a positive or negative number. This can be done as addition as explained in the previous paragraph after taking the two's complement of Y (i.e., deriving $2^n - Y$), no matter whether Y is a negative or positive number. Actually, the subtraction, $X - Y$, can be realized by the addition of X and the bitwise complement of Y with a carry input of 1 to the least significant position. This is convenient for realizing a subtracter circuit, whether it is a serial or parallel adder (to be described later).

Henceforth, let us consider addition of n-bit positive binary integers (without the sign bit) for the sake of simplicity. Let the augend, addend, and sum be $X = [x_{n-1}x_{n-2}\ldots x_0]$, $Y = [y_{n-1}y_{n-2}\ldots y_0]$, and $S = [s_n s_{n-1} s_{n-2}\ldots s_0]$ with $s_n = c_n$, respectively, where each of x_i, y_i, and s_i assumes a value of 0 or 1.

38.3 Serial Adder

A serial adder operates similarly to manual addition. The serial adder, at each step, calculates the sum and carry at one bit position. It starts at the least significant bit position (i.e., $i = 0$) and each successive next step it sequentially moves to the next more significant bit position where it calculates the sum and carry. At the n-th step, it calculates the sum and carry at the most significant bit position (i.e., $i = n - 1$). In other words, the serial adder serially adds augend X and addend Y by adding x_i, y_i, and c_i at the i-th bit position from $i = 0$ to $n - 1$. From the truth table shown in Table 38.1, we have sum bit $s_i = x_i \oplus y_i \oplus c_i$ and carry to the next higher bit position $c_{i+1} = x_i y_i \vee c_i(x_i \vee y_i)$ (also $c_{i+1} = x_i \cdot y_i \vee c_i \cdot (x_i \oplus y_i)$), where "$\cdot$" is AND, "$\vee$" is OR, and "$\oplus$" is XOR, and henceforth, "$\cdot$" will be omitted. This serial addition can be realized by the logic network, called a **serial adder**, or **bit-serial adder**, shown in Fig. 38.2, where its operation is synchronized by a clock. The addition of each i-th bit is done at a rate of one bit per cycle of clock, producing sum bits, s_i's, at the same rate, from the least significant bit to the most significant one. In each cycle, s_i and c_{i+1}, are calculated from x_i, y_i, and the carry from the previous cycle, c_i. The core logic network, shown in the rectangle in Fig. 38.2, for this one-bit addition for the i-th bit position

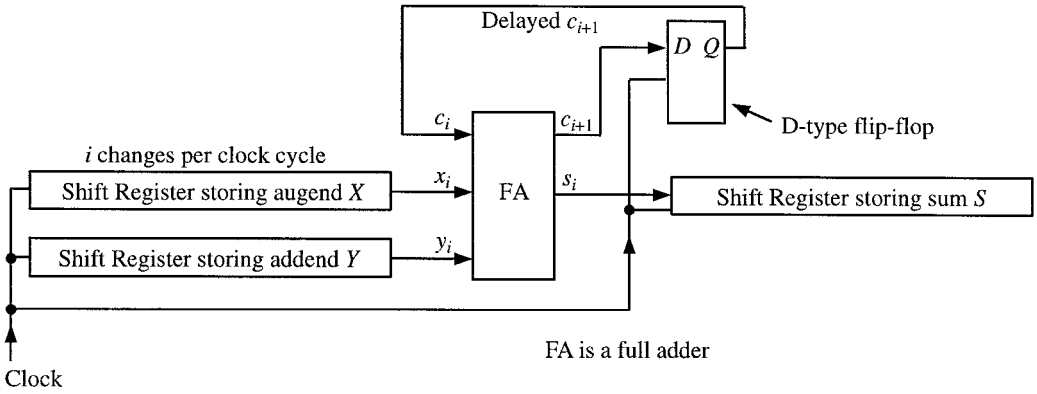


FIGURE 38.2 A serial adder.

is called a **full adder** (abbreviated as **FA**). We obtain a logic network for an FA shown in Fig. 38.3 using AND, OR, and XOR gates. A D-type flip-flop may be used as a delay element which stores the carry for a cycle. Full adders realized in ECL (emitter-coupled logic) are described in Chapter 35. FAs with a minimum number of logic gates are known for different types of logic gates.[10]

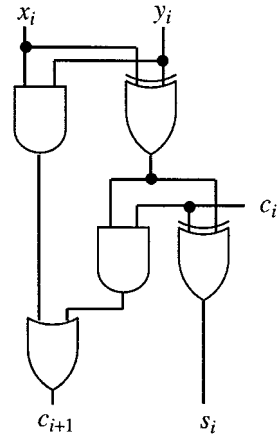


FIGURE 38.3 A full adder.

A serial subtracter can be constructed with a minor modification of a serial adder, as explained in the last paragraph of Section 38.2.

38.4 Ripple Carry Adder

A parallel adder performs addition at all bit positions simultaneously, so it is faster than serial adders.

The simplest parallel adder is a **ripple carry adder**. An n-bit ripple carry adder is constructed by cascading n full adders, as shown in Fig. 38.4. The carry output of each FA is connected to the carry input of the FA of the next higher position. The amount of its hardware is proportional to n. Its worst-case delay is proportional to n because of ripple carry propagation. In designing an FA for a fast ripple carry adder, it is critical to minimize the delay from the carry-in, c_i, to the carry-out, c_{i+1}.

An FA can be realized with logic gates, such as AND gates, OR gates, and XOR gates, as exemplified in Fig. 38.3, and also can be realized with MOSFETs, including pass transistors,[18,23] such that a carry

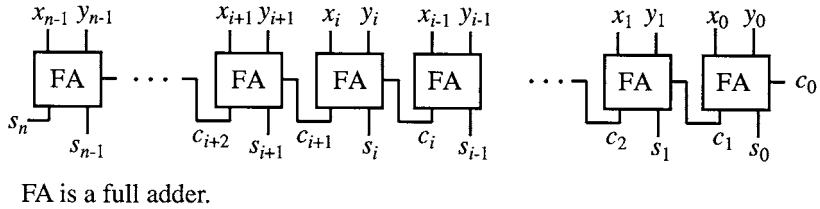
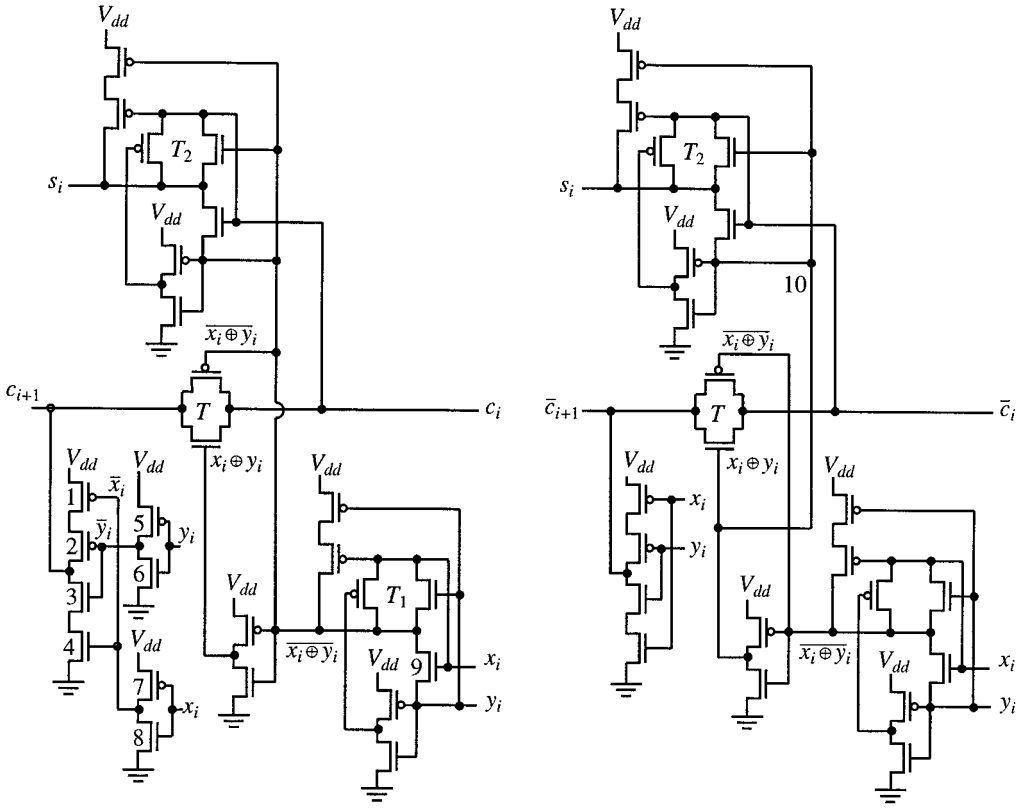


FIGURE 38.4 A ripple carry adder..

c_i goes through logic gates which have some delays. But the speed of adders is very important for the speed of the entire computer. So, FAs are usually realized with more sophisticated transistor circuits using MOSFETs such that a carry c_i can propagate fast to higher bit positions through pass transistors.[22] An example of such an adder is shown in Fig. 38.5, being realized in CMOS. In Fig. 38.5(a), a carry propagates through transmission gate, T (described in Chapter 36, Section 36.2). When we have $x_i = y_i = 0$, T becomes non-conductive and nMOSFETs, 3 and 4, become conductive. Then, the carry-out, c_{i+1}, becomes 0 because the carry-out terminal c_{i+1} is connected to the ground through 3 and 4. When $x_i = y_i = 1$, T becomes non-conductive and pMOSFETs, 1 and 2, become conductive. Then, the carry-out, c_{i+1}, becomes 1 because the carry-out terminal c_{i+1} is connected to the power supply V_{dd} through 1 and 2. When $x_i = 0$ and $y_i = 1$, or $x_i = 1$ and $y_i = 0$, T becomes conductive and the carry-out terminal is connected to neither V_{dd} nor the ground, so a carry-in c_i is sent to c_{i+1} as a carry-out. Thus, we have the values of c_{i+1} for different combinations of values of



(a) Full adder with non-complemented carries.

(b) Full adder with complemented carries.

FIGURE 38.5 A Manchester-type full adder in CMOS.

x_i, y_i, and c_i, as shown in Table 38.1. This carry-path is called a Manchester carry chain. (T_1 is another transmission gate, whereby a circuit on the carry path is simplified and carry propagation is sped up and nMOSFET 9 works as a pass transistor.) A ripple carry adder with Manchester carry chain is referred to as **Manchester adder**. This idea combines very well with the carry skip technique to be mentioned in section 38.5.

The FA in Fig. 38.5(a) cannot send a carry over many positions in a ripple carry adder. For speed-up, we need to insert an inverter in every few positions to send a high output power over many higher bit positions. In order to reduce the number of inverters which have delays in themselves, we can use the FA shown in Fig. 38.5(b) which works with complemented carries. An example with insertion of an inverter at the end of every four cascaded FAs is shown in Fig. 38.6, where a block of four of Fig. 38.5(a) and a block of four of Fig. 38.5(b) are alternated. In Fig. 38.5(b), inverters consisting of MOSFETS 5, 6, 7, and 8 are eliminated from (a), and the function $x_i \oplus y_i$ at point 10 in (b) is the complement of the function at the corresponding point in (a).

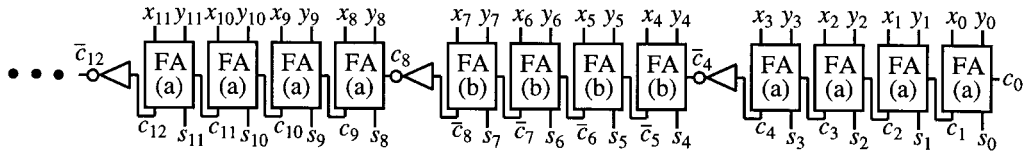
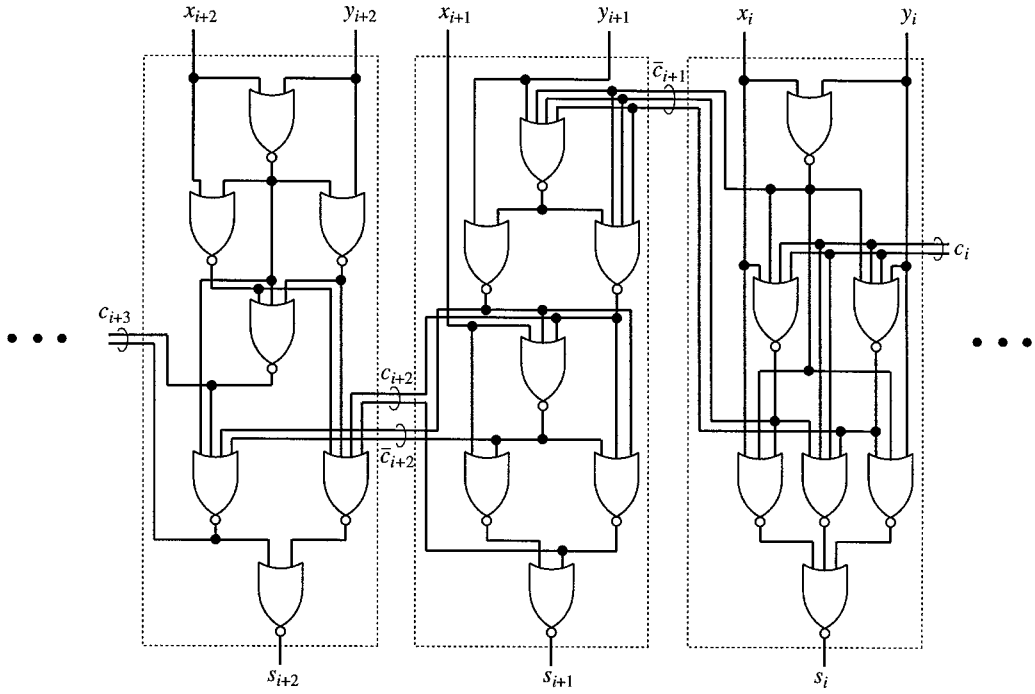


FIGURE 38.6 A ripple carry adder realized by connecting Figs. 38.5(a) and (b).

For high speed, a Manchester full adder realized in dynamic CMOS is used instead of the Manchester full adder shown in static CMOS, where dynamic CMOS and static CMOS are two different variations of CMOS. For example, a Manchester full adder in dynamic CMOS is used inside an adder (to be mentioned later) which is more complex but faster than ripple carry adders.[7]

In the simultaneous addition in all n-bit positions, a carry propagates n positions in the worst case, but on the average, it propagates only about $\log_2 n$ positions.[13] The average computation time of a ripple carry adder can be reduced by detecting the carry completion, because we need not always wait for the worst delay. An adder with such a mechanism is called a carry completion detection adder[3] and is useful for asynchronous systems.

When an FA is realized with ordinary logic gates, say NOR gates only, the total number of logic gates in the ripple carry adder is not minimized, even if the number of logic gates in each FA is minimized. But the number of logic gates in the ripple carry adder can be reduced by using modules that have more than one input for a carry-in (the carry-in c_i, for example in Fig. 38.7 is represented by two lines, instead of one line) and more than one output for a carry-out (the complemented carry-out $\bar{c}_{i+1}$ in Fig. 38.7 is represented by three lines), as shown in Fig. 38.7 (where modules are shown in dot-lined rectangles), instead of using FAs which have only one input for a carry-in and only one output for a carry-out. The number of NOR gates of such a module is minimized by the integer-programming logic design method[11] and it is found that there are 13 minimal modules. Different types of ripple carry adders can be realized by cascading such minimal modules. Some of these adders have carry propagation times shorter than that of the ripple carry adder realized with FAs with NOR gates. Besides, there is an adder that has a minimum number of NOR gates — when this adder is constructed by cascading the three consecutive minimal modules shown in dot-lined rectangles in Fig. 38.7, where the module for the least significant bit position is slightly modified (i.e., replacement of the two carry inputs by a single carry input) and one NOR gate is added to convert the carry out in multiple-lines from the adder into the carry out in a single line. Then it is proved that the total number of NOR gates in this ripple carry adder is minimum for any value of n.[8] Also, there is a ripple adder such that the number of connections, instead of the number of logic gates, is minimized.[14] Related adders are referred to in Section 2.4.3 of Ref. 11.



A ripple adder that has the minimal number of NOR gates for any arbitrary bit length can be realized by cascading these three consecutive minimal modules.

FIGURE 38.7 Three consecutive minimal modules for a ripple carry adder that has the minimal number of NOR gates for any arbitrary bit length.

38.5 Carry Skip Adder

In a ripple carry adder, a carry propagates through the i-th FA when $x_i \neq y_i$, i.e., $x_i \oplus y_i = 1$. Henceforth, we denote $x_i \oplus y_i$ as p_i. A carry propagates through a block of consecutive FAs, when all p_i's in the block are 1. This condition (i.e., all p_i's are 1) is called the carry propagation condition of the block.

A **carry skip adder** is a ripple carry adder that is partitioned into several blocks of FAs, attaching a carry skip circuit to each block, as shown in Fig. 38.8.[9] A carry skip circuit detects the carry propagation condition of the block and lets the carry from the next lower block bypass the block when the condition holds. In Fig. 38.8, carry skip circuits are not attached to the blocks at the most and least significant few positions because the attachment does not speed up the carry propagation much.

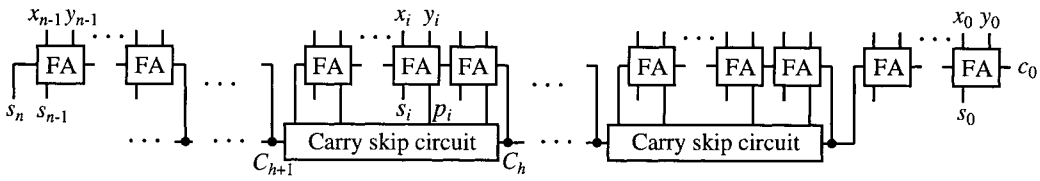


FIGURE 38.8 A carry skip adder.

In the carry skip circuit included in Fig. 38.8, the carry output, C_{h+1}, from block h that consists of k FAs starting from j-th position is calculated as:

$$C_{h+1} = c_{j+k} \vee P_h C_h$$

where c_{j+k} is the carry from the FA at the most significant position of the block,

$$P_h = p_{j+k-1}p_{j+k-2} \cdots p_j$$

is the formula for the carry propagation condition of the block, C_h is the carry from the next lower block, and p_i's are calculated in FAs. An example of carry skip circuit is shown in Fig. 38.9.

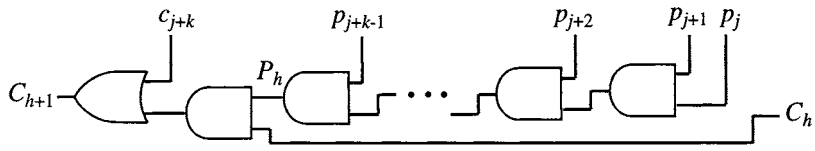


FIGURE 38.9 A carry skip circuit used in Fig. 38.8.

A carry may ripple through FAs in the block where it is generated, bypass the blocks where the carry propagation condition holds, and then, ripple through FAs in the block where the carry propagation condition does not hold. When all blocks are of the same size, k FAs, the worst case occurs when a carry is generated at the least significant position and propagate to the most significant position. The worst delay is the sum of the delay for rippling through $k-1$ FAs, the delay for bypassing $n/k-2$ blocks, and the delay for rippling through $k-1$ FAs. In the case that k is a constant independent of n, as well as in the case that k is proportional to n, the delay is proportional to n. We can reduce the worst delay to being proportional to $\sqrt{n}$, by letting k be proportional to $\sqrt{n}$. The amount of hardware for the entire adder is proportional to n in any case.

Applying the principle used to develop the carry skip adder borrowed from the ripple carry adder, we have a two-level carry skip adder from the basic carry skip adder, for further improvements. Recursive application of the principle yields a multi-level carry skip adder.[12]

38.6 Carry Look-Ahead Adder

As previously stated, the carry, c_{i+1}, produced at the i-th position is calculated as $c_{i+1} = x_i \cdot y_i \vee c_i \cdot (x_i \oplus y_i)$. This means that a carry is generated if both x_i and y_i are 1, or an incoming carry is propagated if one of x_i and y_i is 1 and the other is 0. Therefore, letting g_i denote $x_i y_i$, we have $c_{i+1} = g_i \vee c_i p_i$, where $p_i = x_i \oplus y_i$. Here, g_i is the formula for the carry generation condition at the i-th position, i.e., when g_i is 1, a carry is generated at this position. Substituting $g_{i-1} \vee p_{i-1}c_{i-1}$ for c_i, we get

$$c_{i+1} = g_i \vee p_i g_{i-1} \vee p_i p_{i-1} c_{i-1}$$

Recursive substitution yields

$$c_{i+1} = g_i \vee p_i g_{i-1} \vee p_i p_{i-1} g_{i-2} \vee \ldots \vee p_i p_{i-1} \ldots p_0 c_0$$

A **carry look-ahead adder** can be realized according to this expression, as illustrated in Fig. 38.10 for the case of four bits.[21] According to this expression, c_{i+1}'s are calculated at all positions in parallel.

It is hard to realize an n-bit carry look-ahead adder precisely according to this expression, unless n is small, because maximum fan-in and fan-out restriction is violated at higher positions. Large fan-out causes large delay, so the maximum fan-out is restricted. Also, the maximum fan-in is usually limited to 5 or less. There are some ways to alleviate this difficulty, as follows.

One way is the partition of carry look-ahead adders into several blocks such that each block consists of k positions, starting from the j-th one. In a block h, the carry at each position, c_{i+1}, where $j \le i \le j+k-1$, is calculated as

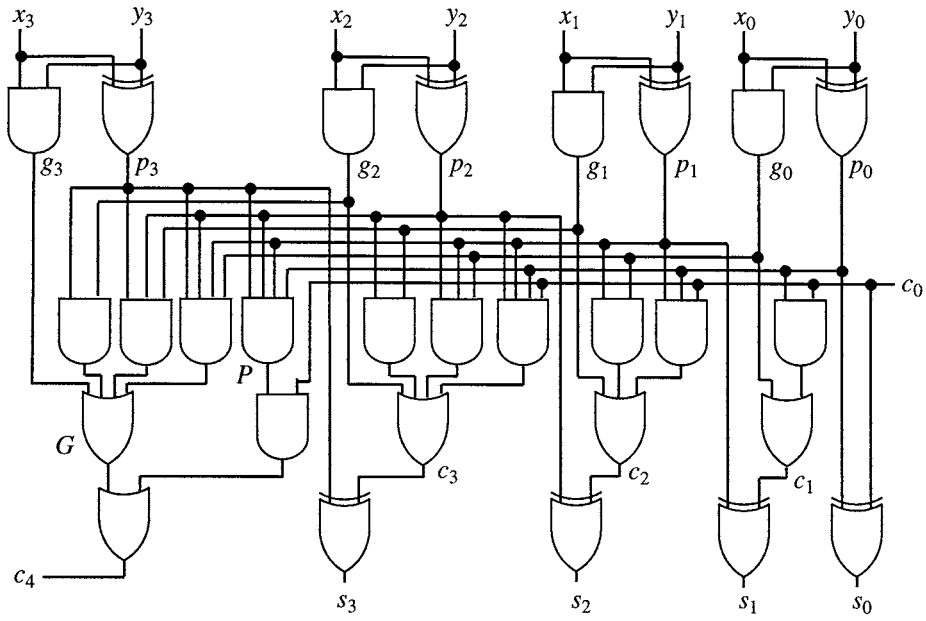


FIGURE 38.10 A 4-bit carry look-ahead adder.

$$c_{i+1} = g_i \vee p_i g_{i-1} \vee \ldots \vee p_i p_{i-1} \cdots p_{j+1} g_j \vee p_i p_{i-1} \cdots p_{j+1} p_j C_h$$

where C_h, i.e., c_j is the carry from the next lower block. The carry from the next lower block, C_h, goes to only the positions of the block h, so the fan-outs and fan-ins do not increase beyond a certain value. Therefore, we can form an n-bit adder by cascading n/k k-bit carry look-ahead adders, where k is a small constant independent of n, often 4. The worst delay of this type of adder and also the hardware amount are proportional to n.

Another way of alleviating the difficulty is recursively applying the principle of carry look-ahead to groups of blocks. The carry output of block h, C_{h+1} $(= c_{j+k})$, is calculated as

$$C_{h+1} = g_{j+k-1} \vee p_{j+k-1} g_{j+k-2} \vee \ldots \vee p_{j+k-1} p_{j+k-2} \cdots p_{j+1} g_j \vee p_{j+k-1} p_{j+k-2} \cdots p_{j+1} p_j C_h$$

This means that in the block, a carry is generated if

$$G_h = g_{j+k-1} \vee p_{j+k-1} g_{j+k-2} \vee \ldots \vee p_{j+k-1} p_{j+k-2} \cdots p_{j+1} g_j$$

is 1, and an incoming carry is propagated if $P_h = p_{j+k-1} p_{j+k-2} \cdots p_{j+1} p_j$ is 1. G_h is the formula for the carry generation condition of the block and P_h is the formula for the carry propagation condition of the block. (They are shown as P and G in Fig. 38.10) Let us consider a super-block, that is, a group of several consecutive blocks. The carry generation and the carry propagation condition of a super-block are detected from those of the blocks, in the same way that G_h and P_h are detected from g_i's and p_i's. Once the carry input to a super-block is given, carry outputs from the blocks in the super-block are calculated immediately. Consequently, we obtain a fast adder in which small carry look-ahead circuits which include carry calculation circuits are connected in a tree form. Figure 38.11 shows a 4-bit carry look-ahead circuit and Fig. 38.12 shows a 16-bit carry look-ahead adder using the 4-bit carry look-ahead circuits, where carry look-ahead circuits are shown as CLA in Fig. 38.12. The worst delay of this type of adder is proportional to log n. The number of logic gates is proportional to n.

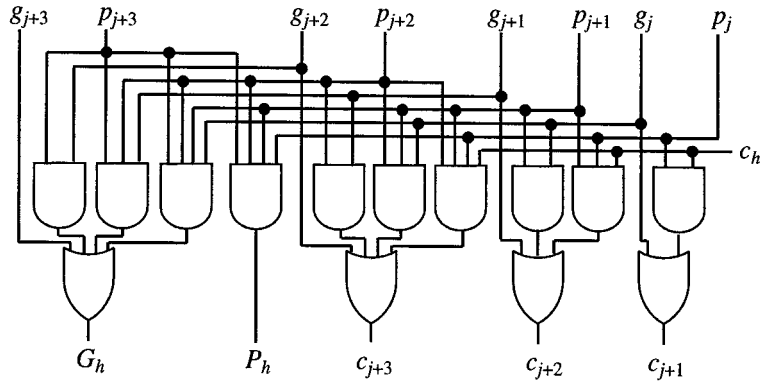


FIGURE 38.11 A 4-bit carry look-ahead circuit.

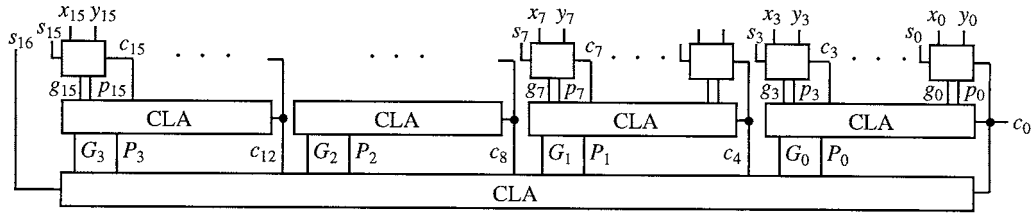


FIGURE 38.12 A 16-bit carry look-ahead adder. CLA stands for a carry look-ahead circuit which includes the carry calculation circuit shown in Fig. 38.10.

38.7 Carry Select Adder

We can reduce the worst delay of a ripple carry adder by partitioning the adder into two blocks: one for higher bit positions and the other for lower bit positions. In the block for higher bit positions, we calculate two candidate sums in parallel, one assuming a carry input of 0 from the block for lower bit positions and the other assuming a carry input of 1, then we select the correct sum based on the actual carry output from the block for lower bit positions. When we partition the adder into two blocks of the same size, the delay becomes about half because the calculations in these two blocks are carried out concurrently. An adder based on this principle is called a **carry select adder**.[1]

We can further reduce the delay by partitioning the adder into more blocks. Figure 38.13 shows a block diagram of a carry select adder. When all blocks are of the same size, k positions, the worst case occurs when a carry is generated at the least significant position and stops at the most significant position. The worst delay is the sum of the delay for rippling through $k-1$ FAs, and the delay for $n/k-1$ selectors.

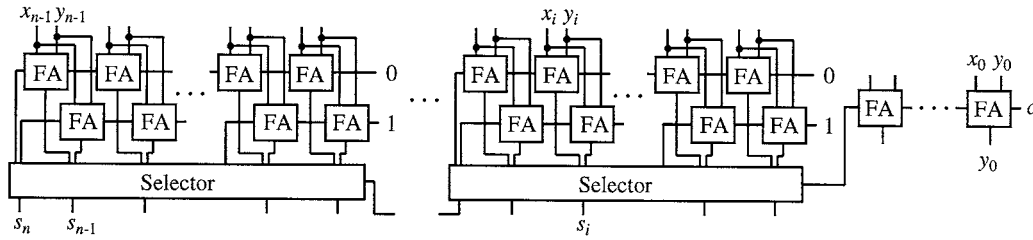


FIGURE 38.13 A carry select adder.

In the case that k is a constant independent of n, as well as in the case that k is proportional to n, the delay is proportional to n. We can reduce the worst delay to being proportional to $\sqrt{n}$, by letting k be proportional to $\sqrt{n}$. The amount of hardware is proportional to n in any case. It is to be noticed that a selector is unnecessary to the least significant few positions (probably less than k) in Fig. 38.13 because it is known whether the carry-in is 0 or 1.

We can reduce the amount of hardware by calculating two candidate sums using only one adder in each block.[19]

Applying the principle used to develop the carry select adder to each block, we can realize a two-level carry select adder. Recursive application of the principle yields a multi-level carry select adder. A conditional sum adder[16] can be regarded as the extreme case.

A carry select adder is used in a microprocessor with high performance.[7]

38.8 Carry Save Adder

When we add up several numbers sequentially, it is not necessary to propagate the carries during each addition. Instead, the carries generated during an addition may be saved as partial carries and added with the next operand during the next addition. Namely, we can accelerate each addition by postponing the carry propagation. This leads to the concept of carry save addition. We may add up numbers by a series of carry save additions, followed by a carry propagate addition. Namely, for multiple-operand addition, only one carry propagate addition is required.

An adder for carry save addition is referred to as a **carry save adder**, while the adders mentioned in the previous section are called **carry propagate adders**. A carry save adder sums up a partial sum and a partial carry from the previous stage as well as an operand and produces a new partial sum and partial carry. An n-bit carry save adder consists of just n full adders without interconnections among them.

References

1. Bedrij, O. J., "Carry-select adder," *IRE Trans. Elec. Comput.*, vol. EC-11, pp. 340–346, June 1962.
2. Cavanagh, J. J. F., *Digital Computer Arithmetic — Design and Implementation*, McGraw-Hill, 1984.
3. Gilchrist, B., J. H. Pomerene, and S. Y. Wong, "Fast carry logic for digital computers," *IRE Trans. Elec. Comput.*, vol. EC-4, pp. 133–136, 1955.
4. Hennessy, J. L. and D. A. Patterson, *Computer Architecture — A Quantitative Approach*, Appendix A, Morgan Kaufmann Publishers, 1990.
5. Hwang, K., *Computer Arithmetic — Principles, Architecture, and Design*, John Wiley & Sons, 1979.
6. Koren, I., *Computer Arithmetic Algorithms*, Prentice Hall, 1993.
7. Kowaleski, J. A. Jr. et al., "A dual-execution pipelined floating-point CMOS processor," *ISSCC Digest of Technical Papers*, pp. 358–359, Feb. 1996.
8. Lai, H.-C. and S. Muroga, "Minimum parallel binary adders with NOR (NAND) gates," *IEEE TC*, pp. 648–659, Sept. 1979.
9. Lehman, M. and N. Burla, "Skip techniques for high-speed carry propagation in binary arithmetic units," *IRE Trans. Elec. Comput.*, vol. EC-10, pp. 691–698, Dec. 1961.
10. Liu, T.-K., K. Hohulin, L.-E., Shiau, and S. Muroga, "Optimal one-bit full adders with different types of gates," *IEEE TC*, pp. 63–70, Jan. 1974.
11. Muroga, S., "Computer-aided logic synthesis for VLSI chips," pp. 1-103, *Advances in Computers*, vol. 32, Ed. by M. C. Yovits, Academic Press, 1991.
12. Omondi, A. R., *Computer Arithmetic Systems — Algorithms, Architecture and Implementations*, Prentice-Hall, 1994.
13. Reitwiesner, G. W., "The determination of carry propagation length for binary addition," *IRE Trans. Elec. Comput.*, vol. EC-9, pp. 35–38, 1960.

14. Sakurai, A. and S. Muroga, "Parallel binary adders with a minimum number of connections," *IEEE Trans. Comput.*, C-32, pp. 969–976, Oct. 1983. (Correction: In Fig. 7, labels, a_0 and $\bar{c}_0$, should be interchanged.)
15. Scott, N. R., *Computer Number Systems & Arithmetic*, Prentice-Hall, 1985.
16. Slansky, J., "Conditional sum addition logic," *IRE Trans. Elec. Comput.*, vol. EC-9, pp. 226–231, June 1960.
17. Spaniol, O., *Computer Arithmetic Logic and Design*, John Wiley & Sons, 1981.
18. Suzuki, M. et al., "A 1.5-ns 32-b CMOS ALU in double pass-transistor logic," *IEEE Jour. of Solid-State Circuits*, pp. 1145–1151, Nov. 1993.
19. Tyagi, A., "A reduced-area scheme for carry-select adders," *IEEE Trans. Comput.*, vol. 42, pp. 1163–1170, Oct. 1993.
20. Waser, S. and M. J. Flynn, *Introduction to Arithmetic for Digital Systems Designers*, Holt, Rinehart and Winston, 1982.
21. Weinberger, A. and J. L. Smith, "A one-microsecond adder using one-megacycle circuitry," *IRE Trans. Elec. Comput.*, vol. EC-5, pp. 65–73, 1956.
22. Weste, N. H. E. and K. Eshraghian, *Principles of CMOS VLSI Design*, 2nd ed., Addison Wesley, 1993.
23. Zhuang, N. and H. Wu, "A new design of the CMOS full adder," *IEEE JSSC*, pp. 840–844, May 1992. (Full adder with transfer gates.)

Takagi, N., et al. "Multipliers"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

39 Multipliers

Naofumi Takagi
Nagoya University

Charles R. Baugh
C. R. Baugh and Associates

Saburo Muroga
University of Illinois at Urbana-Champaign

39.1 Introduction

Many microprocessors and digital signal processors now have fast multipliers in them. There are several types of multipliers with different speeds, areas, and configurations. For the details of multipliers and multiplication methods, see Refs. 3–5, 7–10, and 14.

Here, let us consider multiplication of n-bit positive binary integers (without the sign bit) where a multiplicand $X = [x_{n-1}x_{n-2}\ldots x_0]$ is multiplied by a multiplier $Y = [y_{n-1}y_{n-2}\ldots y_0]$ to derive the product $P = [p_{2n-1}p_{2n-2}\ldots p_0]$, where each of x_i, y_i, and p_i takes a value of 0 or 1.

39.2 Sequential Multiplier

A **sequential multiplier** works in a manner similar to manual multiplication of two decimal numbers, although two binary numbers are multiplied in this case. A multiplicand $X = [x_{n-1}x_{n-2}\ldots x_0]$ is multiplied by each bit of a multiplier $Y = [y_{n-1}y_{n-2}\ldots y_0]$, forming the multiplicand-multiple $Z = [z_{n-1}z_{n-2}\ldots z_0]$, where $z_i = x_i y_j$ for each $i = 0, \ldots, n-1$. Then, Z is shifted left by j bit positions and is added, in all digit positions in parallel, to the partial product P_{j-1} which has been formed by the previous steps, to generate the partial product P_j. Repeating this step for $j = 0$ to $n-1$, the product $P = [p_{2n-1}p_{2n-2}\ldots p_0]$ of $2n$ bits is derived. The only difference of this sequential multiplier from the manual multiplication is the repeated addition of each multiplicand-multiple, instead of one-time addition of all multiplicand-multiples at the end.

This sequential multiplier is realized, as shown in Fig. 39.1, which consists of a Multiplicand Register of n-bits for storing multiplicand X, a Shift Register of $2n$-bits for storing multiplier Y and partial product P_{j-1}, a Multiplicand-Multiple Generator (denoted as MM Generator) for generating a multiplicand-multiple $y_j \cdot X$, and a Parallel Adder of n-bits. Initially, X is stored in the Multiplicand Register and Y is stored in the lower half (i.e., the least significant bit positions) of the Shift Register where the upper half of the Shift Register stores 0. This sequential multiplier performs one iteration step described above in each clock cycle. In other words, in each clock cycle, a multiplier bit y_j is read from the right-most position of Shift Register. A multiplicand-multiple $y_j \cdot X$ is produced by Multiplicand-Multiple Generator, which is X or 0 based on whether y_j is 1 or 0, and is fed to Parallel Adder. The upper n-bit of the partial product is read from the upper half of Shift Register and also fed to Parallel Adder. The content of Shift Register is shifted one position to the right. The $(n+1)$-bit output of Parallel Adder including the carry output, which is the upper part of the updated partial product, is stored into the upper $(n+1)$ positions of Shift Register. After n cycles, Shift Register holds the $2n$-bit product, P.

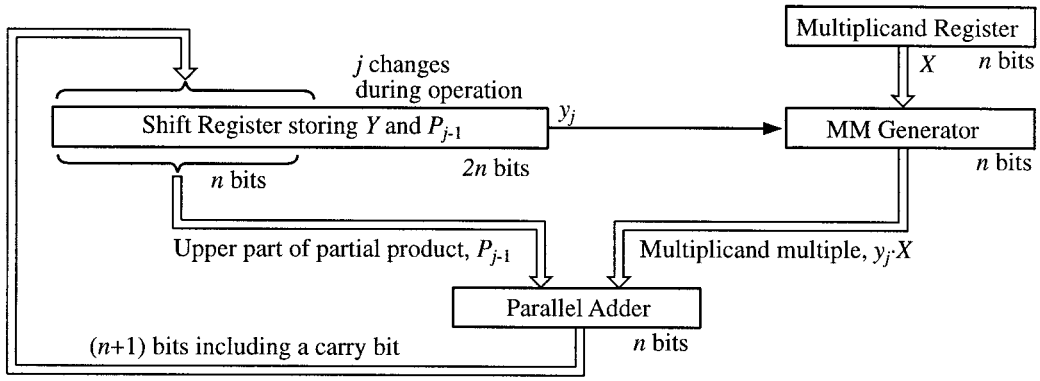


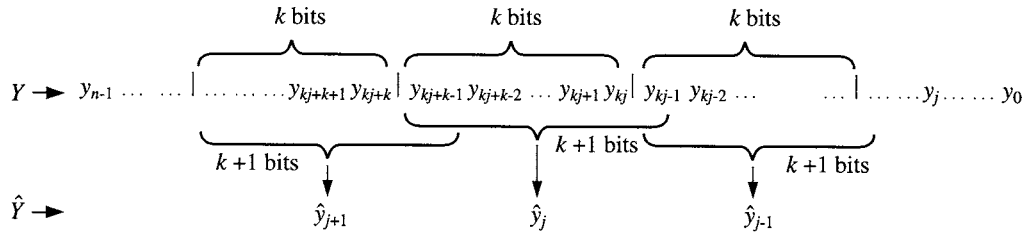
FIGURE 39.1 A sequential multiplier.

We can use any adder described in Chapter 38. The faster the adder, the shorter the clock cycle and hence the faster the multiplier. When we use a carry save adder as Parallel Adder, we have to modify Shift Register so that its upper half stores two binary numbers (i.e., a partial sum and a partial carry). Besides the carry save adder, we need a carry propagate adder for summing up the final partial sum and carry.

We can accelerate sequential multiplication by processing multiple-bits of the multiplier per clock cycle. When k-bits of multiplier Y are processed per cycle, an n-bit multiplication is performed through n/k clock cycles. There are two methods for processing multiple-bits of the multiplier per cycle. One is generating several candidate multiplicand-multiples and then choosing an appropriate one among them. The other is generating k multiplicand-multiples and summing them up in one cycle. Also, these two methods can be combined.

In the first method, Multiplicand-Multiple Generator generates 2^k different multiplicand-multiples, 0, X, $2X$, $3X$, …, $(2^k - 1)X$. For example, when $k = 2$, Multiplicand-Multiple Generator generates $2X$ and $3X$, as well as X and 0. Multiplicand-Multiple Generator consists of a look-up table containing the necessary multiples of the multiplicand and a selector for selecting an appropriate multiple. The look-up table need not hold all multiples, because several of them can be generated from others by shifting whenever necessary. For example, when $k = 2$, only $3X$ must be pre-computed and be held. **Extended Booth's method**[13] is useful for reducing the number of pre-computed multiples. When k-bits are processed per cycle, k-bit Booth's method is applied, which recodes multiplier Y into radix-2^k redundant signed-digit representation with the digit set $\{-2^{k-1}, -2^{k-1}+1, \ldots, 0, \ldots, 2^{k-1} - 1, 2^{k-1}\}$, where each digit in radix 2^k takes a value among those in this digit set. Y is recoded into $\hat{Y}$ by considering $k + 1$ bits of Y, i.e., $y_{kj+k-1}, y_{kj+k-2}, \ldots, y_{kj+1}, y_{kj}, y_{kj-1}$, (i.e., $k + 1$ bits among $y_{n-1}, y_{n-2}, \ldots, y_0$ of Y) per cycle, instead of only a single bit of Y (say, y_j) per cycle, as illustrated in Fig. 39.2(a). More specifically, the j-th digit $\hat{y}_j$ of the recoded multiplier, where $j = 0, 1, \ldots, \lceil (n+1)/k \rceil - 1$, is calculated as $\hat{y}_j = -2^{k-1} \cdot y_{kj+k-1} + 2^{k-2} \cdot y_{kj+k-2} + \ldots + 2 \cdot y_{kj+1} + y_{kj} + y_{kj-1}$. In this case, since all components of $Y = [y_{n-1} y_{n-2} \ldots y_0]$ are recoded for every k components at a time, the recoded number becomes $\hat{Y} = [\hat{y}_{\lceil (n+1)/k \rceil - 1} \hat{y}_{\lceil (n+1)/k \rceil - 2} \ldots \hat{y}_0]$ with $\lceil (n+1)/k \rceil$ components, in contrast to multiplier $Y = [y_{n-1} y_{n-2} \ldots y_0]$ which has n components. Then we have a multiplicand-multiple $\hat{y}_j \cdot X$. Since the negation of a multiple can be produced by complementing it bitwise and adding 1 at the least significant position (this 1 is treated as a carry into the least significant position of Parallel Adder), the number of multiples to be held is reduced. For example, when $k = 2$, the multiplier is recoded to radix-4 redundant signed-digit representation with the digit set $\{-2, -1, 0, 1, 2\}$ by means of the 2-bit Booth's method (i.e., the extended Booth's method with $k = 2$) as $\hat{y}_j = -2y_{2j+1} + y_{2j} + y_{2j-1}$ and all multiples are produced from X by shift and/or complementation. In 2-bit Booth recoding, multiplier Y is partitioned into 2-bit blocks, and then at the j-th block, the recoded multiplier digit $\hat{y}_j$ is calculated from the two bits of the block, i.e., y_{2j+1} and y_{2j}, and the higher bit of the next lower block, i.e., y_{2j-1}, according to the rule shown in Table 39.1. For example, 11110001010 is recoded to $2\ 0\ \bar{2}\ 1\ \bar{1}\ \bar{2}$, where $\bar{1}$ and $\bar{2}$ denote –1

(a) Recoding for k-bit Booth's method



(b) An example of the 2-bit Booth's method based on Table 39.1

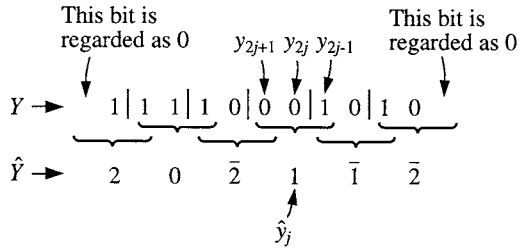


FIGURE 39.2 Recoding in the extended Booth's method.

TABLE 39.1 The Recording Rule of 2-bit Booth's Method

y_{2j+1}	y_{2j}	y_{2j-1}	$\hat{y}_j$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	–2
1	0	1	–1
1	1	0	–1
1	1	1	0

and –2, respectively, as illustrated in Fig. 39.2(b). (In this case, whenever a bit is not available in Y, such as the next lower bit of the least significant bit, it is regarded as 0.) When the extended Booth's method is employed, Parallel Adder is modified such that negative numbers can be processed.

In the second method for processing multiple-bits of the multiplier per clock cycle, Parallel Adder sums up $k + 1$ numbers, using k adders. Any adder can be used, but usually carry save adders are used for the sake of speed and cost. Carry save adders can be connected either in series or in tree form. Of course, the latter is faster, but the structure of Parallel Adder becomes more complicated because of somewhat irregular wire connections. By letting k be n, we have a parallel multiplier, processing the whole multiplier-bits in one clock cycle, as will be mentioned in the following section.

39.3 Array Multiplier

The simplest parallel multiplier is an **array multiplier**[2] in which the multiplicand-multiples (i.e., $(y_j \cdot X)$'s) are summed up one by one by means of a series of carry save adders. It has a two-dimensional array structure of full adders as shown in Fig. 39.3. Each row of full adders except the bottom one forms

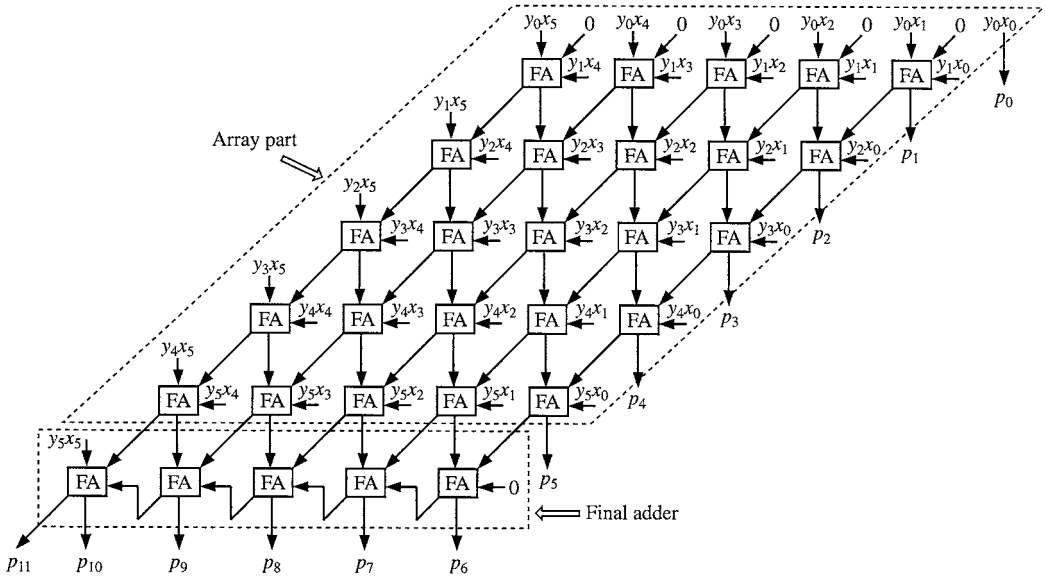


FIGURE 39.3 An array multiplier.

a carry save adder. The bottom row forms a ripple carry adder for the final carry propagate addition. An array multiplier is suited for VLSI realization because of its regular cellular array structure. The number of logic gates is proportional to n^2. The delay is proportional to n.

We can reduce the delay in the final adder by using a faster carry propagate adder such as a carry select adder. Also, we can reduce the delay in the array part in Fig. 39.3 by means of 2-bit Booth's method mentioned in Section 39.2. Since 2-bit Booth's method reduces the number of multiplicand-multiples to about half, the number of necessary carry save additions is also reduced to about half, and hence, the delay in the array part is reduced to about half. But the amount of hardware is not reduced much because a 2-bit Booth recoder and multiplicand-multiple generators, which essentially work as selectors, are required.

Another method to reduce the delay in the array part is to double the accumulation stream.[6] Namely, we divide the multiplicand-multiples into two groups, sum up the members of each group by a series of carry save adders independently of the other group, and then sum up the two accumulations into one. The delay in the array part is reduced to about half. The 2-bit Booth's method can be combined with this method. We can further reduce the delay by increasing the number of accumulation streams, although it complicates the circuit structure.

39.4 Multiplier Based on Wallace Tree

In the **multiplier based on Wallace tree**,[13] the multiplicand-multiples are summed up in parallel by means of a tree of carry save adders. A carry save adder sums up three binary numbers and produces two binary numbers (i.e., a partial sum and a partial carry). Therefore, using $n/3$ carry save adders in parallel, we can reduce the number of multiplicand-multiples from n to about $2n/3$. Then, using about $2n/9$ carry save adders, we can further reduce it to $4n/9$. Applying this principle about $\log_{3/2} n$ times, the number of multiplicand-multiples can be reduced to only two. Finally, we sum up these two multiplicand-multiples by means of a fast carry propagate adder.

Figure 39.4 illustrates a block diagram of a multiplier based on Wallace tree. This consists of full adders, just like the array multiplier described previously. (Recall that a carry save adder consists of full

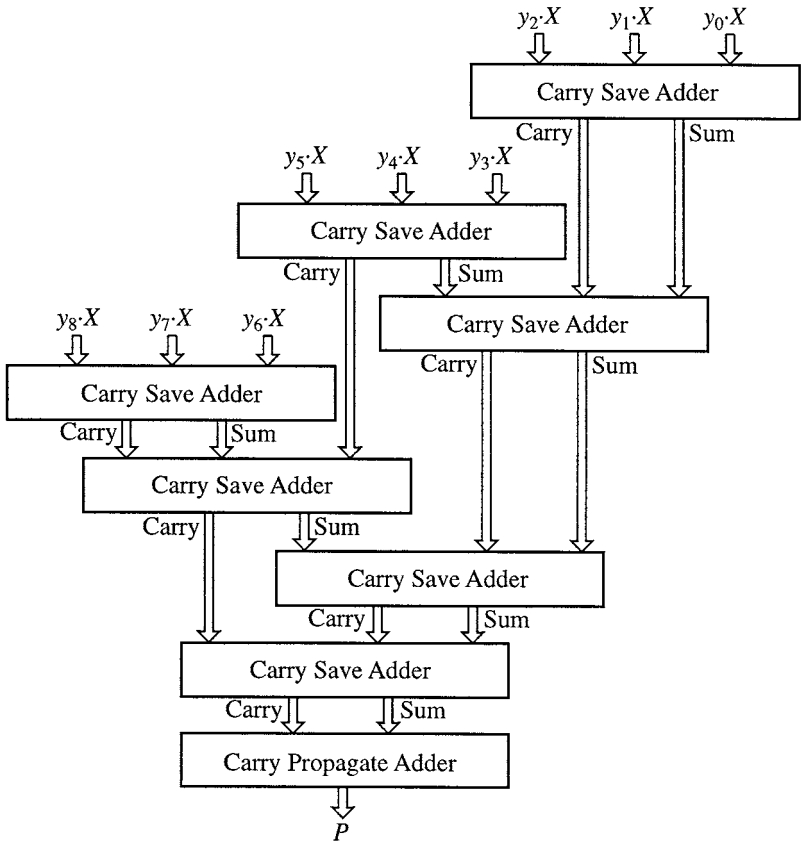


FIGURE 39.4 A multiplier with Wallace tree.

adders.) The delay is small. It is proportional to log n when a fast carry propagate adder with $O(\log n)$ delay is used for the final addition. The number of logic gates is about the same as that of an array multiplier and is proportional to n^2. However, its circuit structure is not suited for VLSI realization because of the complexity.

The 2-bit Booth's method can be also applied to this type of multiplier. However, it is not as effective as in the array multiplier, because the height of the Wallace tree decreases by only one or two, even though the number of the multiplicand-multiples is reduced to about half.

A full adder, which is used as the basic cell in a multiplier based on Wallace tree, can be regarded as a counter which counts up 1's in the three-input bits and outputs the result as a 2-bit binary number. Namely, a full adder can be regarded as a 3-2 counter. We can also use larger counters, such as 7-3 counters and 15-4 counters, as the basic cells, instead of full adders.

We can increase the regularity in the circuit structure by replacing Wallace tree with a 4-2 adder tree,[12] where a 4-2 adder is formed by connecting two carry save adders, shown in the dot-lined rectangles, in series, as shown in Fig. 39.5. A 4-2 adder is an adder that sums up four binary numbers, $A = [a_{n-1} \ldots a_0]$, $B = [b_{n-1} \ldots b_0]$, $C = [c_{n-1} \ldots c_0]$, and $D = [d_{n-1} \ldots d_0]$, and produces two binary numbers, $E = [e_n \ldots e_0]$ and $F = [f_n \ldots f_0]$, where $f_0 = 0$. We can form a 4-2 adder tree by connecting 4-2 adders in a binary tree form. The delay of a multiplier based on a 4-2 adder tree is slightly larger than that of a multiplier with Wallace tree but is still proportional to log n. The number of logic gates is about the same as a multiplier based on Wallace tree, and is proportional to n^2. It is more suited for VLSI realization than the Wallace tree.

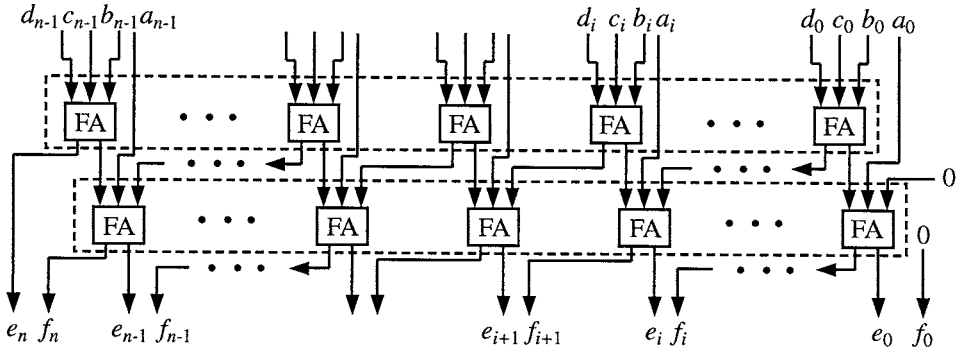


FIGURE 39.5 A 4-2 adder.

39.5 Multiplier Based on a Redundant Binary Adder Tree

There is another fast multiplier based on a rather regular structure called a **multiplier based on a redundant binary adder tree**.[11] In it, multiplicand-multiples are generated first as in other parallel multipliers, being regarded as redundant binary numbers, and are summed up pairwise by means of redundant binary adders connected in binary tree form. Then, finally, the product represented in the redundant binary representation is converted into the ordinary binary representation. The redundant binary representation, also called the binary signed-digit representation, is a binary representation with a digit set $\{\bar{1}, 0, 1\}$, where $\bar{1}$ denotes -1.[1] An n-digit redundant binary number $A = [a_{n-1}a_{n-2}\ldots a_0]$ has the value $\sum_{i=0}^{n-1} a_i \cdot 2^i$, where a_i takes a value $\bar{1}$, 0, or 1. There may be several redundant binary numbers which have the same value. For example, [0101], $[0 1 1 \bar{1}]$, $[1 \bar{1} 0 1]$, $[1 \bar{1} 1 \bar{1}]$, and $[1 0 \bar{1} \bar{1}]$ all represent 5. Because of this redundancy, we can add two redundant binary numbers without carry propagation.

Let us consider the addition of two redundant binary numbers, that is, the augend, $A = [a_{n-1}a_{n-2}\ldots a_0]$, and the addend, $B = [b_{n-1}b_{n-2}\ldots b_0]$ to derive the sum, $S = [s_n s_{n-1}\ldots s_0]$, where each of a_i, b_i, and s_i takes a value -1, 0, or 1. The addition without carry propagation is done in two steps. In the first step, an intermediate carry c_{i+1}, and intermediate sum d_i in the i-th position are determined such that $a_i + b_i = 2c_{i+1} + d_i$ is satisfied (the 2 of $2c_{i+1}$ means shifting c_{i+1} to the next higher digit position as a carry), where each of c_{i+1} and d_i is -1, 0, or 1. In this case, c_{i+1} and d_i are determined such that a new carry will not be generated in the second step. In the second step, in each digit position, sum s_i is determined by adding intermediate sum d_i and intermediate carry c_i from the next lower position, where c_i takes a value, -1, 0, or 1.

Suppose one of addend digit a_i and addend digit b_i is 1 and the other is 0. If $c_{i+1} = 0$ and $d_i = 1$ in the first step, a new carry will be generated for $c_i = 1$ from the next lower digit position in the second step. So, if there is a possibility of $c_i = 1$, we choose $c_{i+1} = 1$ and $d_i = \bar{1}$. On the other hand, if there is a possibility of $c_i = \bar{1}$, we choose $c_{i+1} = 0$ and $d_i = 1$. This makes use of the fact that 1 can be expressed by [01] and $[1 \bar{1}]$ in the redundant binary number representation. Whether c_i becomes 1 or $\bar{1}$ can be detected by examining a_{i-1} and b_{i-1} in the next lower digit position but not further lower digit positions. For other combinations of the values of a_i, b_i, and c_i, c_{i+1} and d_i can be similarly determined.

In the second step, s_i is determined by adding only two digits, c_i and d_i. Suppose $c_i = 1$. Then s_i is 0 or 1, based on whether d_i is $\bar{1}$ or 0. Notice that two combinations, $c_i = d_i = 1$ and $c_i = d_i = \bar{1}$, never occur. For other combinations of the values of c_i and d_i, s_i is similarly determined. Consequently, sum digit s_i at each digit position can be determined by the three digit positions of the angend and addend, a_i, a_{i-1} and a_{i-2}, and b_i, b_{i-1}, and b_{i-2}.

A binary number is a redundant binary number as it is, so there is no need for converting it to the redundant binary number representation. But conversion of a redundant binary number to a binary number requires an ordinary binary subtraction. Namely, we subtract the binary number that is derived by replacing every 1 by 0 and $\bar{1}$ by 1 in the redundant binary number, from the binary number that is

derived by replacing $\bar{1}$ by 0. For example, $[1\bar{1}0\bar{1}1]$ is converted to [00111] by the subtraction [10001] – [01010]. We need borrow propagate subtraction for this conversion. This conversion in a multiplier based on a redundant binary adder tree corresponds to the final addition in the ordinary multipliers.

References

1. Avizienis, A., "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Elec. Comput.*, vol. EC-10, pp. 389–400, Sep. 1961.
2. Baugh, C. R. and B. A. Wooly, "A two's complement parallel array multiplier," *IEEE Trans. Computers*, vol. C-22, no. 12, pp. 1045–1047, Dec. 1973.
3. Cavanagh, J. J. F., *Digital Computer Arithmetic — Design and Implementation*, McGraw-Hill, 1984.
4. Hennessy, J. L. and D. A. Patterson, *Computer Architecture —A Quantitative Approach, Appendix A*, Morgan Kaufmann Publishers, 1990.
5. Hwang, K., *Computer Arithmetic — Principles, Architecture, and Design*, John Wiley & Sons, 1979.
6. Iwamura, J., et al., "A 16-bit CMOS/SOS multiplier-accumulator," *Proc. IEEE Intnl. Conf. on Circuits and Computers*, 12.3, Sept. 1982.
7. Koren, I., *Computer Arithmetic Algorithms*, Prentice-Hall, 1993.
8. Omondi, A. R., *Computer Arithmetic Systems — Algorithms, Architecture and Implementations*, Prentice-Hall, 1994.
9. Scott, N. R., *Computer Number Systems & Arithmetic*, Prentice-Hall, 1985.
10. Spaniol, O., *Computer Arithmetic Logic and Design*, John Wiley & Sons, 1981.
11. Takagi, N., H. Yasuura, and S. Yajima, "High-speed VLSI multiplication algorithm with a redundant binary addition tree," *IEEE Trans. Comput.*, vol. C-34, no. 9, pp. 789–796, Sep. 1985.
12. Vuillemin, J. E., "A very fast multiplication algorithm for VLSI implementation," *Integration, VLSI Journal*, vol. 1, no. 1, pp. 39–52, Apr. 1983.
13. Wallace, C. S., "A suggestion for a fast multiplier," *IEEE Trans. Elec. Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
14. Waser, S. and M. J. Flynn, *Introduction to Arithmetic for Digital Systems Designers*, Holt, Rinehart and Winston, 1982.

Takagi, N., Muroga, S. "Dividers"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

40 Dividers

Naofumi Takagi
Nagoya University

Saburo Muroga
University of Illinois at Urbana-Champaign

40.1 Introduction

There are two major classes of division methods: subtract-and-shift methods and multiplicative methods. For details of division methods and dividers, see Refs. 2, 5–11, and also Ref. 3 for subtract-and-shift division.

Suppose two binary numbers, X and Y, are normalized in the range equal to or greater than 1/2 but smaller than 1 and also $X < Y$ holds for the sake of simplicity. Then, a dividend $X = [0.1x_2 \ldots x_n]$, which is an n-bit normalized binary fraction, is to be divided by a divisor, $Y = [0.1y_2 \ldots y_n]$, which is another n-bit normalized binary fraction, where each of x_i and y_i takes a value of 0 or 1. We assume to calculate the quotient $Z = [0.1z_2 \ldots z_n]$ which satisfies $|X/Y - Z| < 2^{-n}$, where z_i takes a value of 0 or 1.

40.2 Subtract-And-Shift Dividers

The **subtract-and-shift divider** works in a manner similar to manual division of one decimal number by another, using paper and pencil. In the case of manual division of decimal numbers, each x_i of dividend X and y_i of divisor Y is selected from {0, 1, …, 9} (i.e., the radix is 10). Each z_i of quotient Z is selected from {0, 1, …, 9}. In the following case of dividers for a digital system, dividend X and divisor Y are binary numbers, so each x_i of X and y_i of Y is selected from {0, 1}, but the quotient (which is not necessarily represented as a binary number) is expressed in radix r. The radix r, is usually chosen to be 2^k (i.e., 2, 4, 8, and so on.) So, a quotient is denoted with $Q = [0.q_1q_2 \ldots q_{\lceil n/k \rceil}]$, to be differentiated from $Z = [0.1z_2 \ldots z_n]$ expressed in binary numbers, although both Q and Z will henceforth be called quotients.

The subtract-and-shift method with a radix r iterates the recurrence step of replacing R_j by $r \cdot R_{j-1} - q_j \cdot Y$, where q_j is the j-th quotient digit and R_j is the partial remainder after the determination of q_j. Initially, R_{j-1} for $j - 1 = 0$; that is, R_0 is X. Each recurrence step consists of the following four substeps. Suppose that $r = 2^k$.

1. Shift of the partial remainder R_{j-1} to the left by k bit positions to produce $r \cdot R_{j-1}$.
2. Determination of the quotient digit q_j by quotient-digit selection.
3. Generation of the divisor multiple $q_j \cdot Y$.
4. Subtraction of $q_j \cdot Y$ from $r \cdot R_{j-1}$ to calculate R_j.

The dividers for a digital system have many variations, depending on the methods in choosing the radix, the quotient-digit set from which q_j is chosen (q_j is not necessarily 0 or 1, even if it is in radix 2),

and the representation of the partial remainder. The simplest cases are with a radix of 2 and the partial remainder represented in the non-redundant form. When $r = 2$, the recurrence is $R_j = 2R_{j-1} - q_j \cdot Y$. There are three methods: the restoring, the non-restoring, and the SRT methods.

Restoring Method

In the radix-2 restoring method, a quotient digit, q_j, is chosen from the quotient-digit set {0, 1}. When $2R_{j-1} - Y \geq 0$ ($2R_{j-1}$ means shift of R_{j-1} by one bit position to the left), 1 is selected, and otherwise 0 is selected. Namely, $R_j' = 2R_{j-1} - Y$ is calculated first, and then, when $R_j' \geq 0$ holds, we set $q_j = 1$ and $R_j = R_j'$, and otherwise $q_j = 0$ and $R_j = R_j' + Y$ (i.e., Y is added back to R_j'). For every j, R_j is kept in the range, $0 \leq R_j < Y$. The j-th bit of the quotient in binary number $Z = [0.1z_2 \ldots z_n]$, z_j, is equal to q_j (i.e., z_j is the same as q_j in radix 2 in this case of the restoring method). This method is called the **restoring method**, because Y is added back to R_j' when $R_j' < 0$. For speed-up, we can use $2R_{j-1}$ as R_j by keeping R_{j-1}, instead of adding back Y to R_j', when $R_j' < 0$. Figure 40.1 shows an example of radix-2 restoring division.

Dividend: $X = [0.100111]$, Divisor: $Y = [0.110001]$

		A	B	C	D	E								
	R_0	⓪					1	0	0	1	1	1		
	$2R_0$		⓪				1.	0	0	1	1	1		
	$+(-Y)$			①			1.	0	0	1	1	1	0	+1
	R_1'				○	⓪	0.	1	1	1	0	1	←	Positive
$z_1 = q_1 = 1$	R_1						0.	1	1	1	0	1	←	$R_1 = R_1'$
	$2R_1$						0	0.	1	1	1	0	1	
	$+(-Y)$						1	1.	0	0	1	1	1	0	+1
	R_2'							0.	0	0	1	0	0	1	←	Positive
$z_2 = q_2 = 1$	R_2							0.	0	0	1	0	0	1	←	$R_2 = R_2'$
	$2R_2$						0	0.	0	1	0	0	1	
	$+(-Y)$						1	1.	0	0	1	1	1	0	+1
	R_3'							1.	1	0	0	0	0	1	←	Negative
$z_3 = q_3 = 0$	R_3							0.	0	1	0	0	1	0	←	$R_3 = R_3' + Y = 2R_2$
	$2R_3$						0	0.	1	0	0	1	0	
	$+(-Y)$						1	1.	0	0	1	1	1	0	+1
	R_4'							1.	1	1	0	0	1	1	←	Negative
$z_4 = q_4 = 0$	R_4							0.	1	0	0	1	0	0	←	$R_4 = R_4' + Y = 2R_3$
	$2R_4$						0	1.	0	0	1	0	0	
	$+(-Y)$						1	1.	0	0	1	1	1	0	+1
	R_5'							0.	0	1	0	1	1	1	←	Positive
$z_5 = q_5 = 1$	R_5							0.	0	1	0	1	1	1	←	$R_5 = R_5'$
	$2R_5$						0	0.	1	0	1	1	1	
	$+(-Y)$						1	1.	0	0	1	1	1	0	+1
	R_6'							1.	1	1	1	1	0	1	←	Negative
$z_6 = q_6 = 0$														

Quotient: $Z = [0.110010]$

FIGURE 40.1 An example of radix-2 restoring division. (Each of *A*, *B*, and *C* is a sign bit. Notice that *D* is always equal to *E* and is ignored.)

Non-Restoring Method

In the radix-2 non-restoring method, a quotient digit, q_j is chosen from the quotient-digit set {–1, 1}. Quotient digit q_j is chosen according to the sign of R_{j-1}. In other words, we set $q_j = 1$ when $R_{j-1} \geq 0$ and, otherwise, we set $q_j = -1$, abbreviated as $q_j = \bar{1}$. Then, $R_j = 2R_{j-1} - q_j \cdot Y$ is calculated. Even if R_j is

negative, Y is not added back in this method, so this method is called the **non-restoring method**. Note that since we have $R_0 = X > 0$ and $(1/2) \le X < Y < 1$ by the assumption, we always have $q_1 = 1$ and $R_1 = 2R_0 - Y = 2X - Y > 0$, and hence, $q_2 = 1$. For every j, R_j is kept in the range, $-Y \le R_j < Y$. The j-th bit of the quotient in binary number $Z = [0.1z_2 \ldots z_n]$, z_j, is 0 or 1, based on whether q_{j+1} is -1 or 1. And we have always $z_n = 1$. For example, when $Q = [0.111\bar{1}\bar{1}1]$ where $\bar{1}$ denotes -1, we have $Z = [0.110011]$. (The given number $[0.111\bar{1}\bar{1}1]$ is calculated as $[0.111001] - [0.000110] = [0.110011]$. In other words, the number derived by replacing all 1's by 0's and all $\bar{1}$'s by 1's in the given number is subtracted from the number derived by replacing all $\bar{1}$'s by 0's in the given number. This turns out to be a simple conversion between z_j and q_{j+1}, as stated above, without requiring the subtraction.) Combining this conversion with the recurrence on R_j yields the method in which $R_1 = 2X - Y$, and for $j \ge 2$, when $R_{j-1} \ge 0$, $z_{j-1} = 1$ and $R_j = 2R_{j-1} - Y$ and, otherwise, $z_{j-1} = 0$ and $R_j = 2R_{j-1} + Y$. Figure 40.2 shows an example of radix-2 non-restoring division. Note that the remainder for the restoring method is always negative, whereas the remainder for the non-storing method (also the SRT method to be described in the following) can be negative, and consequently when the remainder is negative, the quotient of the latter is greater by 2^{-n} than the former. (This explains the difference between the quotients in Figs. 40.1 and 40.2, where $R_6 = 1$ in Fig. 40.2 indicates that the remainder is negative.)

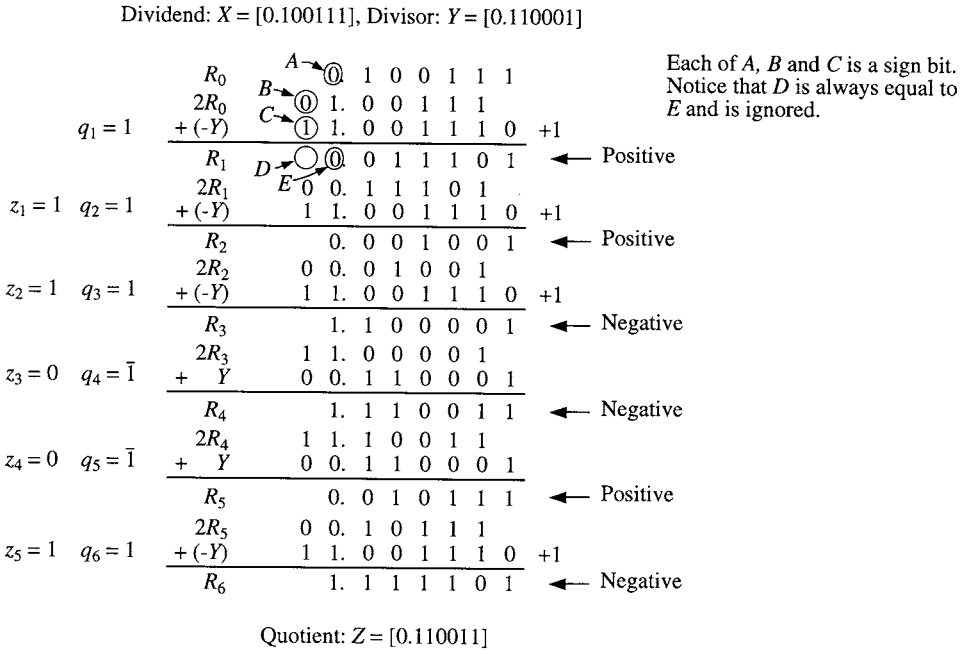


FIGURE 40.2 An example of radix-2 non-restoring division.

Figure 40.3 shows a radix-2 non-restoring divider that performs one recurrence step in each clock cycle. Register For Partial Remainder R_{j-1} initially stores the dividend X and then, during the division, stores a partial remainder R_{j-1} which may become negative. Divisor, Y, in Register For Divisor Y is added to or subtracted from the twice of the R_{j-1} stored in Register For Partial Remainder R_{j-1} by Adder/Subtracter, based on whether the left-most bit of Register For Partial Remainder R_{j-1} (i.e., the sign bit of R_{j-1}) is 1 or 0, and then the result (i.e., R_j) is stored back into Register For Partial Remainder R_{j-1}. Concurrently, the complement of the sign bit of R_{j-1} (i.e., z_{j-1}) is fed to Shift Register For Z_{j-2} (which stores the partial quotient $Z_{j-2} = [0.1z_2 \ldots z_{j-2}]$) from the right end, and the partial quotient stored in Shift Register For Z_{j-2} is shifted one position to the left. The divider performs one recurrence step in each clock cycle. After n cycles, Shift Register For Z_{j-2} holds the quotient Z. We can use any carry propagate adder (subtracter) as the Adder/Subtracter. The faster the adder, the shorter the clock cycle, and hence, the faster the divider.

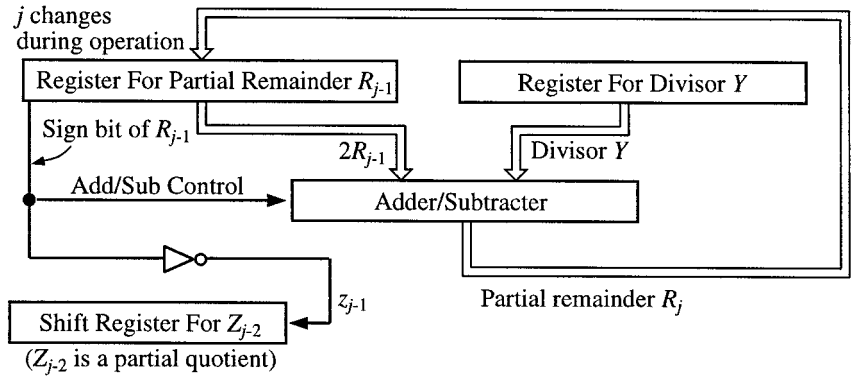


FIGURE 40.3 A radix-2 non-restoring divider.

SRT Method

In the radix 2 SRT method, the quotient-digit set is {–1, 0, 1}. In this case, –1 or 0 or 1 is selected as q_j, based on whether $2R_{j-1} < -(1/2)$ or $-(1/2) \le 2R_{j-1} < (1/2)$ or $(1/2) \le 2R_{j-1}$. When 0 is selected as q_j, no addition or subtraction is performed for the calculation of R_j, and hence, the computation time may be shortened. Quotient digit q_j is determined from the values of the three most significant bits (shown in each of the dot-lined rectangles in Fig. 40.4) of $2R_{j-1}$, as the radix-2 SRT division is exemplified in Fig. 40.4. R_{j-1} satisfies $-Y \le R_{j-1} < Y$ and is represented in a two's complement representation with 1-bit integral part (and n-bit fractional part).

Dividend: $X = [0.100111]$, Divisor: $Y = [0.110001]$

	R_0	A→	0.	1	0	0	1	1	1	
	$2R_0$	B→	0	1.	0	0	1	1	1	← $1/2 \le 2R_0$ (actually, $1 < 2R_0$)
$q_1 = 1$	+ (-Y)	C→	1	1.	0	0	1	1	1	0 +1
	R_1	D→	0	0.	0	1	1	1	0	1
	$2R_1$	E→	0	0.	1	1	1	0	1	← $1/2 \le 2R_1$
$q_2 = 1$	+ (-Y)		1	1.	0	0	1	1	1	0 +1
	R_2			0.	0	0	1	0	0	1
	$2R_2$		0	0.	0	1	0	0	1	← $-1/2 \le 2R_2 < 1/2$
$q_3 = 0$										← No addition
	R_3			0.	0	1	0	0	1	0
	$2R_3$		0	0.	1	0	0	1	0	← $1/2 \le 2R_3$
$q_4 = 1$	+ (-Y)		1	1.	0	0	1	1	1	0 +1
	R_4			1.	1	1	0	0	1	1
	$2R_4$		1	1.	1	0	0	1	1	← $-1/2 \le 2R_4 < 1/2$
$q_5 = 0$										← No addition
	R_5			1.	1	0	0	1	1	0
	$2R_5$		1	1.	0	0	1	1	0	← $2R_5 < -1/2$
$q_6 = \bar{1}$	+ Y		0	0.	1	1	0	0	0	1
	R_6			1.	1	1	1	1	0	1

Quotient: $Q = [0.11010\bar{1}] \rightarrow Z = [0.110011]$

FIGURE 40.4 An example of radix-2 SRT division. (Each of *A*, *B*, and *C* is a sign bit. Notice that *D* is always equal to *E* and is ignored.)

In the above three methods, carry (or borrow) propagates in the calculation of R_j in each recurrence step because the partial remainder is represented in the non-redundant form. We can accelerate the SRT method by using a redundant form for representing the partial remainder, and performing the calculation of R_j without carry (or borrow) propagation. Let us consider the use of the carry save form. R_j, which satisfies $-Y \le R_j < Y$, is represented in a two's complement carry save form with 1-bit integral part (and n-bit fractional part.) (In the two's complement carry save form, both the partial carry and the partial sum are represented in the two's complement representation.) In this case, –1 or 0 or 1 is selected as q_j, based on whether $2\hat{R}_{j-1} \le -1$ or $2\hat{R}_{j-1} = -(1/2)$, or $2\hat{R}_{j-1} \ge 0$, where $2\hat{R}_{j-1}$ denotes the value of the three most significant digits (shown in each of the dot-lined rectangles in Fig. 40.5), i.e., down to the first binary position, of $2R_{j-1}$. Note that $2\hat{R}_{j-1} \le 2R_{j-1} < 2\hat{R}_{j-1} + 1$. Figure 40.5 shows an example of radix-2 SRT division with the partial remainder represented in the carry save form.

Dividend: $X = [0.100111]$, Divisor: $Y = [0.110001]$

R_0	RS_0		0.	1	0	0	1	1	1	← Partial sum
	RC_0		0.	0	0	0	0	0		← Partial carry
$2R_0$	$2RS_0$		0	1.	0	0	1	1	1	
	$2RC_0$		0	0.	0	0	0	0		← B
$q_1 = 1$	+ (-Y)		1	1.	0	0	1	1	1	0 +1
	RS_1			0.	0	0	0	0	0	1
	RC_1			0.	0	1	1	1	0	← A
	$2RS_1$		0	0.	0	0	0	0	1	
	$2RC_1$		0	0.	1	1	1	0		
$q_2 = 1$	+ (-Y)		1	1.	0	0	1	1	1	0 +1
	RS_2			1.	1	1	0	1	0	1
	RC_2			0.	0	1	0	1	0	
	$2RS_2$		1	1.	1	0	1	0	1	
	$2RC_2$		0	0.	1	0	1	0		
$q_3 = 1$	+ (-Y)		1	1.	0	0	1	1	1	0 +1
	RS_3			0.	0	0	1	1	0	1
	RC_3			1.	0	1	0	1	0	
	$2RS_3$		0	0.	0	1	1	0	1	
	$2RC_3$		1	0.	1	0	1	0		
$q_4 = \bar{1}$	+ Y		0	0.	1	1	0	0	0	1
	RS_4			0.	0	0	0	0	1	1
	RC_4			1.	1	1	0	0	0	
	$2RS_4$		0	0.	0	0	0	1	1	
	$2RC_4$		1	1.	1	0	0	0		
$q_5 = 0$	+ 0		0	0.	0	0	0	0	0	0
	RS_5			1.	1	0	0	1	1	0
	RC_5			0.	0	0	0	0	0	
	$2RS_5$		1	1.	0	0	1	1	0	
	$2RC_5$		0	0.	0	0	0	0		
$q_6 = \bar{1}$	+ Y		0	0.	1	1	0	0	0	1
	RS_6			1.	1	1	1	1	0	1
	RC_6			0.	0	0	0	0	0	

Quotient: $Q = [0.111\bar{1}0\bar{1}] \rightarrow Z = [0.110011]$

FIGURE 40.5 An example of radix-2 SRT division with partial remainder represented in the carry save form. (The bits shown inside *A* are calculated from the bits shown inside *B*.)

In the radix-2 SRT method with the partial remainder represented in the carry save form, as in the original SRT method, the conversion of the quotient into the ordinary binary representation is required because the quotient is represented in the redundant binary representation with digit set {–1, 0, 1}. There is a method called the **on-the-fly conversion**[4] that performs the conversion without carry propagate addition. It converts the quotient on-the-fly as quotient digits are produced. Expressing the bits up to the j-th of Q and Z as $Q_j = [0.q_1q_2 \ldots q_j]$ and $Z_j = [0.z_1z_2 \ldots z_j]$ respectively, Z_j is the ordinary binary representation of Q_j or that of $Q_j - 2^{-j}$. The latter is the case when the remaining (lower) part of Q_n is negative. Therefore, we can obtain Z immediately after q_n is determined (i.e., all quotient digits are determined), by holding the ordinary binary representation of Q_j and that of $Q_j - 2^{-j}$ at each recurrence step. Let them be ZP_j and ZN_j, respectively. At each step, ZP_j and ZN_j are calculated as follows. When $q_j = -1$, $ZP_j = ZN_{j-1} + 2^{-j}$ and $ZN_j = ZN_{j-1}$. When $q_j = 0$, $ZP_j = ZP_{j-1}$ and $ZN_j = ZN_{j-1} + 2^{-j}$. When $q_j = 1$, $ZP_j = ZP_{j-1} + 2^{-j}$ and $ZN_j = ZP_{j-1}$. In any case, each calculation of ZP_j and ZN_j is performed by a selection of ZP_{j-1} or ZN_{j-1} and a concatenation of 0 or 1. Figure 40.6 shows an example of on-the-fly conversion.

$$
\begin{array}{lll}
q_1 = 1 & ZP_1 & 0.\ 1 \\
 & ZN_1 & 0.\ 0 \\
q_2 = 1 & ZP_2 & 0.\ 1\ 1 \\
 & ZN_2 & 0.\ 1\ 0 \\
q_3 = 1 & ZP_3 & 0.\ 1\ 1\ 1 \\
 & ZN_3 & 0.\ 1\ 1\ 0 \\
q_4 = \bar{1} & ZP_4 & 0.\ 1\ 1\ 0\ 1 \\
 & ZN_4 & 0.\ 1\ 1\ 0\ 0 \\
q_5 = 0 & ZP_5 & 0.\ 1\ 1\ 0\ 1\ 0 \\
 & ZN_5 & 0.\ 1\ 1\ 0\ 0\ 1 \\
q_6 = \bar{1} & ZP_6 & 0.\ 1\ 1\ 0\ 0\ 1\ 1 \\
 & ZN_6 & 0.\ 1\ 1\ 0\ 0\ 1\ 0
\end{array}
$$

$$\text{Quotient: } Z = ZP_6 = [0.110011]$$

FIGURE 40.6 An example of on-the-fly conversion.

Figure 40.7 shows a radix-2 SRT divider with a carry save adder. Register For Partial Carry For R_{j-1} and Register For Partial Sum For R_{j-1} together store partial remainder R_{j-1} represented in the carry save form, i.e., by two binary numbers, partial carry, and partial sum. The three most significant bits of these two registers are fed to the Quotient-Digit Selector, which produces q_j. Divisor Multiple Generator generates Y, 0, or $-Y$, based on whether q_j is –1, 0, or 1. Shift Registers For Z_{j-1} consists of two shift registers for storing ZP_{j-1} and ZN_{j-1} and two selectors controlled by q_j.

40.3 Higher Radix Subtract-and-Shift Dividers

In the previous subsection, we considered radix-2 dividers. In this subsection, let us consider higher radix dividers. When the radix r is 2^k, the number of iterations of the recurrence step is n/k. The larger the k, the fewer the iterations but the longer the time of each recurrence step, because of the additional complexity in the quotient-digit selection and the generation of the divisor multiples.

A redundant digit set, especially a redundant symmetric signed-digit set $\{-a, -a+1, \ldots, -1, 0, 1, \ldots, a-1, a\}$, where $a \geq r/2$, is often used to obtain fast algorithms. A larger a reduces complexity of the quotient-digit selection but increases the complexity of the generation of the divisor multiples. The use of the signed-digit set makes the conversion of the quotient into the ordinary binary representation necessary. The partial remainder can be represented in either non-redundant form, or redundant form

(e.g., carry save form). By the use of a redundant form, we can perform addition/subtraction without carry/borrow propagation, and hence fast. However, it slightly complicates the quotient-digit selection and doubles the number of register bits required for storing the partial remainder.

Among the radix-4 methods, the method with the quotient-digit set of {–2, –1, 0, 1, 2} and the redundant partial remainder is the most popular, because the generation of the divisor multiples is easy and carry save addition can be used for speed-up. In this method, R_j, which satisfies $-(2/3)Y \le R_j < (2/3)Y$, is represented in the carry save form (or redundant binary representation with the digit set {–1, 0, 1}) with 1-bit integral part (and n-bit fractional part). ($R_0 = X$ and $X < (2/3)Y$ must hold.) Quotient digit q_j is determined from the seven most significant digits of R_{j-1} and the five most significant bits of Y. (Actually not five but four bits of Y are required, since its most significant bit is always 1.) The on-the-fly conversion of the quotient can be extended to radix-4. The essential part of a divider based on this method is very similar to that shown in Fig. 40.7. The seven most significant bits of the content of Register For Partial Carry For R_{j-1} and Register For Partial Sum For R_{j-1}, as well as the five (actually four) significant bits of the content of Register For Divisor Y are fed to Quotient Digit Selector. Divisor Multiple Generator generates $-2Y, -Y, 0, Y$, and $2Y$. This type of divider is used in floating point arithmetic units of several microprocessors.

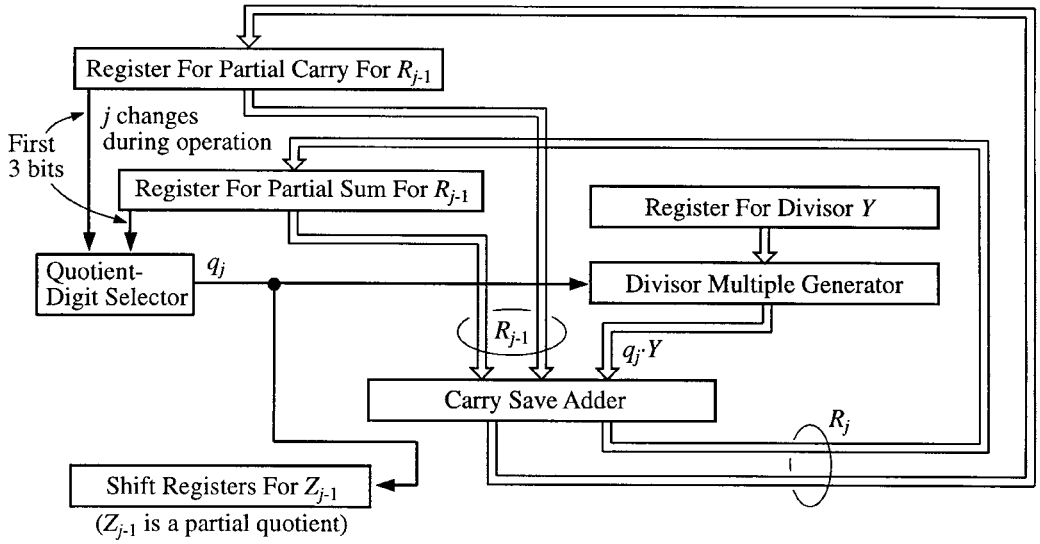


FIGURE 40.7 A radix-2 SRT divider with carry save adder.

An efficient radix-8 method is with the quotient-digit set of {–7, –6, … , –1, 0, 1, … , 6, 7} and the redundant partial remainder. Quotient digit q_j is determined from the eight most significant digits of R_{j-1} and the four most significant bits of Y. Quotient digit q_j is decomposed into two components, $qh_j \in \{-8, -4, 0, 4, 8\}$ and $ql_j \in \{-2, -1, 0, 1, 2\}$, and R_j is calculated through two carry save additions.

As the radix increases, the quotient-digit selection becomes more complex. This complexity can be reduced by restricting the divisor to a range close to 1, by prescaling the divisor. We can preserve the value of the quotient by prescaling the dividend with the same factor as the divisor. For example, we can design an efficient radix-16 method with the quotient-digit set of {–10, –9, … , –1, 0, 1, … , 9, 10}, where q_j is determined from the ten most significant digits of R_{j-1}, by scaling the divisor into $(8107/8192) \le Y \le (8288/8192)$. In this method, q_j is decomposed into two components, $qh_j \in \{-8, -4, 0, 4, 8\}$ and $ql_j \in \{-2, -1, 0, 1, 2\}$, and R_j is calculated through two carry save additions.

Since the recurrence step becomes more complex as the radix increases, a higher radix division step is implemented by several lower radix (i.e., radix of 2 or 4) stages. We can design a more efficient divider by not merely concatenating lower radix stages but overlapping them so that the delay and the cost of the hardware for the recurrence step are reduced.

40.4 Even Higher Radix Dividers with a Multiplier

Division with even higher radix, say radix-2^8, based on the iteration of calculating R_j by $r \cdot R_{j-1} - q_j \cdot Y$, requires a multiplier for the generation of the divisor multiple $q_j \cdot Y$. This multiplier is also used for prescaling the divisor very close to 1, so that the quotient digit is produced by truncation or rounding of the shifted partial remainder to the left of the radix point.

Many variations of the method are feasible, depending on the choice of the radix, the quotient-digit set, the representation of the partial remainder (non-redundant or redundant), quotient-digit selection (by truncation or rounding), calculation of the scaling factor, and scaling. The scaling factor can be calculated by any method that produces an approximation of the reciprocal of the divisor. Direct approximation by table look-up and linear interpolation by table look-up followed by multiply-addition are examples. There are three ways of performing scaling: scaling the divisor and the dividend, scaling the divisor and the quotient, and scaling the partial remainder in each iteration. A version of this method is realized in a math-coprocessor.[1]

40.5 Multiplicative Dividers

Multiplicative methods perform division through iterative multiplications and are employed in systems that have fast multipliers. The Newton method and Goldschmidt's algorithm are well-known.

The Newton method calculates the reciprocal of the divisor Y by the Newton-Raphson method. Beginning with an approximation to the reciprocal of the divisor, U_0, $1/Y$ is obtained through the iterative calculation of $U_{i+1} = U_i \cdot (2 - U_i \cdot Y)$. Multiplying the dividend X with the obtained $1/Y$, we can obtain the quotient.

Goldschmidt's algorithm is based on the fact that the value of a fraction is unchanged by multiplying both the numerator and the denominator by the same number. Namely, $X/Y = (X \cdot D_0 \cdot D_1 \cdot D_2 \ldots)/(Y \cdot D_0 \cdot D_1 \cdot D_2 \ldots)$ holds. When D_i's are selected so that $Y \cdot D_0 \cdot D_1 \cdot D_2 \ldots$ approaches to 1, $X \cdot D_0 \cdot D_1 \cdot D_2 \ldots$ approaches X/Y. We can obtain the quotient through the iterative calculations of $D_i = 2 - Y_i$, $Y_{i+1} = Y_i \cdot D_i$, and $X_{i+1} = X_i \cdot D_i$. We use an approximation to $1/Y$ as D_0.

In either method, an approximation to the reciprocal of the divisor is required. Direct approximation by table look-up or linear interpolation by table look-up followed by multiply-addition can be used. When the precision of the approximation is m-bits, n-bit division can be done through about $\log_2(n/m)$ iterations. Two multiplications and a complementation are performed in each iteration. The required calculations are almost the same in both methods. The two multiplications in the Goldschmidt's algorithm are independent of each other, whereas those in the Newton method are performed serially.

Multiplicative division methods are employed in mainframe computers as well as some microprocessors.

References

1. Briggs, W. S. and D. T. Matula, "A 17 × 19 bit multiply and add unit with redundant binary feedback and single cycle latency," *Proc. 11th Symposium on Computer Arithmetic*, pp. 163–170, June 1993.
2. Cavanagh, J. J. F., *Digital Computer Arithmetic — Design and Implementation*, McGraw-Hill, 1984.
3. Ercegovac, M. D. and T. Lang, *Division and Square Root — Digit-Recurrence Algorithms and Implementations*, Kluwer Academic Publishers, 1994.
4. Ercegovac, M. D. and T. Lang, "On-the-fly conversion of redundant into conventional representations," *IEEE Trans. Comput.*, vol. C-36, no. 7, pp. 895–897, July 1987.
5. Hennessy, J. L. and D. A. Patterson, *Computer Architecture — A Quantitative Approach*, *Appendix A*, Morgan Kaufmann Publishers, 1990.
6. Hwang, K., *Computer Arithmetic — Principles, Architecture, and Design*, John Wiley & Sons, 1979.
7. Koren, I., *Computer Arithmetic Algorithms*, Prentice-Hall, 1993.
8. Omondi, A. R., *Computer Arithmetic Systems — Algorithms, Architecture and Implementations*, Prentice-Hall, 1994.

9. Scott, N. R., *Computer Number Systems & Arithmetic*, Prentice-Hall, 1985.
10. Spaniol, O., *Computer Arithmetic Logic and Design*, John Wiley & Sons, 1981.
11. Waser, S. and M. J. Flynn, *Introduction to Arithmetic for Digital Systems Designers*, Holt, Rinehart and Winston, 1982.

Muroga, S. "Full-Custom and Semi-Custom Design"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

41

Full-Custom and Semi-Custom Design

Saburo Muroga
University of Illinois at Urbana-Champaign

41.1 Introduction

As integrated circuits become more inexpensive and compact, many new types of products, such as digital cameras, digital camcorders, and digital television,[2] are being introduced, based on digital systems. Consequently, logic design must be done under many different motivations. Since each case is different, we have different design problems. For example, we have to choose an appropriate IC (integrated circuit) logic family, since these cases have different performance requirements (scientific computers require high speed, but wristwatches require very low power consumption), although in recent years, CMOS has been more widely used than other IC logic families, such as ECL, which has been used for fast computers.

Logic functions that are frequently used by many designers, such as a full adder, are commercially available as off-the-shelf IC packages. (A package means an IC chip or a discrete component encased in a container.) Logic networks that realize such logic networks are often called **standard (logic) networks**. A single component, such as a resistor and a capacitor, is also commercially available as an off-the-shelf discrete component package. Logic networks can be assembled with these off-the-shelf packages. In many cases, not only performance requirements but also compactness and low cost are very important for products such as digital cameras. So, digital systems must accordingly be realized in IC packages that are designed, being tailored to specific objectives, rather than assembling many of these off-the-shelf packages on pc-boards, although assembling with these off-the-shelf packages has the advantage of ease of partial design changes.

Here, however, let us consider two important cases of designing an IC chip inside such an IC package, which is not off-the-shelf, that leads to two sharply contrasting logic design approaches: quick design and high-performance design. Quick design of IC chips is called **semi-custom design** (recently called **ASIC design**, abbreviating Application Specific Integrated Circuit design), whereas deliberate design for high performance is called **full-custom design** because full-custom design is fully customized to high performance. Full-custom design is discussed in this chapter, and different approaches of semi-custom design will be discussed in the succeeding chapters.

Semi-Custom Design

When manufacturers introduce new products or computers, it is ideal to introduce them with the highest performance in the shortest time. But it is usually very difficult to attain both, so one of them must be

emphasized, based on the firm's marketing strategy against competitors. Often, quick introduction of new computers or new merchandise with digital systems is very important for a manufacturer in terms of profits. (In some cases, introduction of a new product one year earlier than a competitor's generates more than twice the total income that the competitor gets.[1]) This is because the firm that introduces the product can capture all initial potential customers at highest prices, and latecomers are left with only the remaining fewer customers, selling at lower prices. This difference in timing often means a big difference in profits. In other words, the profits due to faster introduction of new products on a market often far exceed the profits due to careful design. The use of off-the-shelf IC packages, including off-the-shelf microprocessors, is used to be a common practice for shortening design time. But recent progress enables us to design digital systems in an IC chip more compactly with higher performance than before by curtailing time-consuming layout of transistor circuits on chips and by extensively using CAD programs. Thus, in the case of small volume production, the design cost, part of the product cost, is reduced. This makes semi-custom design appropriate for debugging or prototyping of new design. But extensive use of CAD programs tends to sacrifice the performance or compactness of the semi-custom-designed IC chips. Semi-custom design, or ASIC design, has several different approaches, as will be discussed in later chapters.[3,4] Design of logic networks with the highest performance requires deliberate design of logic networks, design of transistor circuits, layout of these transistor circuits most compactly, and manufacturing of them. Such logic networks are called **random-logic gate networks** and are realized by full-custom design. In contrast to full-custom design, semi-custom design simplifies design and layout of transistor circuits to save expenses and design time. Depending on how design and layout of transistor circuits are simplified (e.g., repetition of small transistor subcircuit, or not so compact layout) and even how logic design is simplified, we have variants of semi-custom design.

Full-Custom Design

Full-custom design is logic design to attain the highest performance or smallest size, utilizing the most advanced technology. Designers usually try to improve the economic aspect, that is, performance per cost, at the same time. Full-custom design with the most advanced technology usually takes many years to achieve final products, because new technology must often be explored at the same time. Hence, this is the other extreme to the above quick design in terms of design time. Every design stage is carefully done for the maximum performance, and transistor circuits are deliberately laid out on chips most compactly, spending months by many draftpeople and engineers. CAD programs are used but not extensively as in the case of semi-custom design. When CAD programs for high performance are not available, for example, for the most compact layout of transistor circuits to which is required for high performance — manual design is used, possibly mixed with the use of some CAD programs. Also, once mistakes sneak into some stages in the long sequence of design, designers have to repeat at least part of the long sequence of design stages to correct them. So, every stage is deliberately tested with substantial effort.

Motivation for Semi-Custom Design

It should be noticed that the cost of a digital system highly depends on the production volume of a chip. The cost of an IC package can be approximately calculated by the following formula:

$$\begin{bmatrix}\text{Total cost}\\ \text{of an IC}\\ \text{package}\end{bmatrix} = \frac{[\text{Design expenses}]}{[\text{Production volume}]} + \begin{bmatrix}\text{Manufacturing}\\ \text{cost per IC}\\ \text{package}\end{bmatrix} \tag{41.1}$$

The second term on the right-hand side of Eq. 41.1, [Manufacturing cost per IC package], is fairly proportional to the size of each chip when the complexity of manufacturing is determined, being usually on the order of dollars, or tens of dollars in the case of commercial chips. In the case of full-custom design, chips are deliberately designed by many designers spending many months. So, [Design

expenses], the first term on the right-hand side of Eq. 41.1 is very high and can easily be on the order of tens of millions of dollars. Thus, the first term is far greater than the second term, making [Total cost of an IC package] very expensive, unless [Production volume] is very large, being on the order of more than tens of millions. Many digital systems that use IC chips are produced in low volume and [Design expenses] must be very low. Semi-custom design is for this purpose and CAD programs need to be used extensively for shortening design time and manpower in order to reduce [Design expenses]. In this case, [Manufacturing cost per IC chip] is higher than that in the case of full-custom design because the size of each chip is larger.

Thus, we can see the following from the formula in Eq. 41.1: chips by semi-custom design are cheaper in small production volume than those by full-custom design, but more expensive in high production volume. But chips by full-custom design are cheaper in the case of high volume production, and are expensive for low volume production.

41.2 Full-Custom Design Sequence of a Digital System

Full-custom design flow of a digital system follows a long sequence of different design stages, as follows.

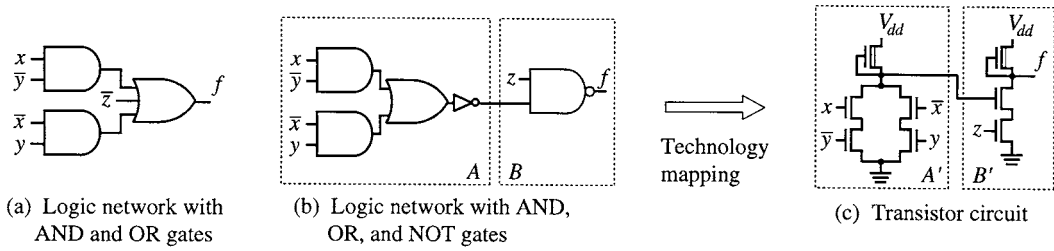
First, the architecture of a digital system is designed by a few people. The performance or cost of the entire system is predominantly determined by architectural design, which must be done based on good knowledge of all other aspects of the system, including logic design and also software to be run. If an inappropriate architecture is chosen, the best performance or lowest cost of the system cannot be achieved, even if logic networks, or other aspects like software, are designed to yield the best results. For example, if microprogramming is chosen for the control logic of a microcomputer based on ROM, it occupies too much of the precious chip area, sacrificing performance and cost, although we have the advantages of short design time and design flexibility. Thus, if performance or manufacturing cost is important, realization of control logic by logic networks (i.e., hard-wired control logic) is preferred. Actually, every design stage is important for the performance of the entire system. **Logic design is also one of key factors for computer performance, such as architecture design, transistor circuit design, layout design, compilers, and application programs. Even if other factors are the same, computer speed can be significantly improved by deliberate logic design.**

Next, appropriate IC logic families and the corresponding transistor circuit technology are chosen for each segment of the system. Other aspects such as memories are simultaneously determined in greater detail. We do not use expensive, high-speed IC logic families where speed is not required.

Architecture and transistor circuits are outside the scope of this handbook, so they are not discussed here further.

The next stage in the design sequence is the design of logic networks, considering cost reduction and the highest performance, realizing functions for different segments of the digital system. Logic design requires many engineers for a fairly long time.

Then, logic networks are converted into transistor circuits. This conversion is called **technology mapping**. It is difficult to realize the functions of the digital system with transistor circuits directly, skipping logic design, although experienced engineers can design logic networks and technology mapping at the same time, at least partly. Logic design with AND, OR, and NOT gates, using conventional switching theory, is convenient for human minds because AND, OR, and NOT gates in logic networks directly correspond, respectively, to basic logic operations, AND, OR, and NOT in logic expressions. Thus, logic design with AND, OR, and NOT gates is usually favored for manual design by designers and then followed by technology mapping. For example, the logic network with AND and OR gates shown in Fig. 41.1(a) is technology-mapped into the MOS circuit shown in Fig. 41.1(c). A variety of IC logic families, such as static MOS circuits and dynamic MOS circuits, are now used to realize logic gates. Thus, the relationships between logic networks with AND, OR, and NOT gates and those in transistor circuits are complex because logic gates realized in transistor circuits do not have one-to-one correspondence with AND, OR, and NOT gates, as illustrated in Fig. 41.1. The function $f = x\bar{y} \vee \bar{x}y \vee \bar{z}$ in Fig. 41.1 can be realized with two AND gates realizing $x\bar{y}$ and $\bar{x}y$, and then with an OR gate which has inputs from the AND



(a) Logic network with AND and OR gates

(b) Logic network with AND, OR, and NOT gates

(c) Transistor circuit

FIGURE 41.1 Technology mapping.

gates and $\bar{z}$, as shown in Fig. 41.1(a). But if the function is realized with AND, OR, and NOT gates, as shown in Fig. 41.1(b), then conversion of it into the MOS circuit shown in Fig. 41.1(c) is easier because correspondence between subnetworks in (b) and transistor logic gates in (c) is clear, where dot-lined rectangles, A and B, in (b) correspond A' and B' in (c), respectively.

Then, after technology mapping, these transistor circuits are laid out on a chip. Layout is also a painstaking endeavor for many draftpersons. The above design stages are highly interactive and iterative because, if bad design is made in a certain stage, good design in other stages cannot compensate for it, thus yielding poor performance or cost increase of the entire chip. In particular, logic network design and layout design are highly interactive.

In this case, it is important to notice the difference of delay time of signal propagation in logic networks from that in transistor circuits laid out on a chip. In previous chapters, we have assumed for the sake of simplicity that signal propagation has delay time only on gates (for the sake of simplicity, equal delay time is assumed on every gate) but not on connections. But in the case of transistor circuits, signal propagation on each connection has significant delay time, which can be greater than delay time of gates. The longer the connection, the greater the delay time on that connection. The larger the number of connections (i.e., fan-out connections) from the output of a gate, the greater the delay time of each connection. Also, each logic gate realized in transistor circuit may have a different delay time. The greater the fan-in of a gate, the greater the delay time. Restrictions on maximum fan-out and maximum fan-in are very important for the performance of logic networks. Thus, if we want to have fast transistor circuits, we need to consider these relationships in designing logic networks with AND and OR gates. Consideration of only the number of levels is not sufficient.

Then, IC chips are manufactured and are assembled with pc-boards into a digital system.

In the case of the high-performance design discussed above, every effort is made to realize digital systems with the best performance (usually speed), while simultaneously considering the reduction of cost.

When we want to develop digital systems of high performance, using the most advanced technology, much greater manpower and design time are required than that needed for semi-custom design approaches. The actual design time requirement depends on how ambitious the designers are. High-performance microprocessor chips are usually designed by full-custom design, typically taking 3 to 5 years with a large number of people, perhaps several dozen engineers. If the digital system is not drastically different from previous models, design time can be shorter with fewer people; but if the system is based on many new ideas, it may be longer.

As we become able to pack an increasingly large number of networks in a single IC chip every year, the full-custom design of VLSI chips (including microcomputers) of high performance with the most advanced technology is beginning to require far greater design effort and longer time. Thus, more extensive use of improved CAD programs is inevitable. This is because a new generation of microprocessor chips has been introduced every 4 years, having a few times as many transistors on a chip. Compared with systems of 10 years ago, contemporary systems consist of two or three order more transistors, although the physical size of these systems are far smaller. For example, IBM's first personal computer, introduced in 1981, was installed with only 16 kilobytes RAM (expandable to 64 kilobytes) and Intel's

microprocessor 8080, which consists of 4800 transistors. But Intel's microprocessor, Pentium III with 500 MHz, introduced in 1999 consists of about 9,500,000 transistors (an approximate number of logic gates can be obtained by dividing the number of transistors by a number between 3 and 5).

In addition to the use of transistor circuits as logic gates, memories are becoming widely used to implement logic networks, being mixed with gates. Also, software is often implemented with ROMs (read-only memories) as firmware, since ROMs are cheaper and smaller than RAMs (random-access memories). Because of these developments, we have complex problems in designing logic networks with a mixture of gates, software, and memories. Essentially, boundaries among logic design, transistor circuits, software, and architecture have disappeared. The number of transistors, or logic gates, used in digital systems is increasing all the time. In designing such gigantic digital systems, it is becoming extremely important to design without errors, necessitating extensive testing in every design stage. To cope with these complex problems, CAD programs with new logic design methods have been developed in recent years. For example, recent CAD programs for logic design can synthesize far larger logic networks than manual design can, and appropriate logic expressions can be derived for functions with a large number of variables by using BDDs.

References

1. Davidow, W., "How microprocessors boost profits," *Electronics,* pp. 105–108, July 11, 1974; p. 92, Jan. 23, 1975.
2. Jurgen, R. K., *Digital Consumer Electronics Handbook*, McGraw-Hill, 1997.
3. Muroga, S., *VLSI System Design*, John Wiley & Sons, 1982.
4. Weste, N. H. E. and K. Eschraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, 2nd ed., Addison-Wesley, 1993.

Muroga, S. "Programmable Logic Devices"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

42

Programmable Logic Devices

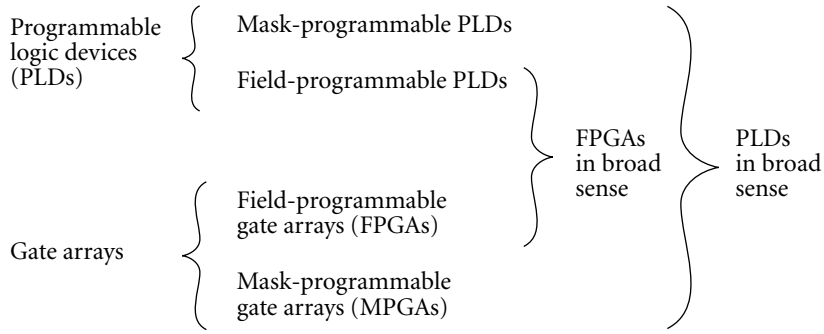
Saburo Muroga
University of Illinois at Urbana-Champaign

42.1 Introduction

Hardware realization of logic networks is generally very time-consuming and expensive. Also, once logic functions are realized in hardware, it is difficult to change them. In some cases, we need logic networks that are easily changeable. One such case is logic networks whose output functions need to be changed frequently, such as control logic in microprocessors, or logic networks whose outputs need to be flexible, such as additional functions in wrist watches and calculators. Another case is logic networks that need to be debugged before finalizing. **Programmable logic devices** (i.e., **PLD**s) are for this purpose. On these PLDs, all transistor circuits are laid out on IC chips prior to designers' use, considering all anticipated cases. With PLDs, designers can realize logic networks on an IC chip, by only deriving concise logic expressions such as minimal sums or minimal products, and then making connections among pre-laid logic gates on the chip. So, designers can realize their own logic networks quickly and inexpensively using these pre-laid chips, because they need not design logic networks, transistor circuits, and layout for each design problem. Thus, designers can skip substantial time of months for hardware design. CAD programs for deriving minimal sums or minimal products are well developed,[1] so logic functions can be realized very easily and quickly as hardware, using these CAD programs. The ease in changing logic functions without changing hardware is just like programming in software, so the hardware in this case is regarded as "programmable." Programmable logic arrays (i.e., PLAs) and FPGAs are typical programmable logic devices.

PLDs consists of mask-programmable PLDs and field-programmable PLDs. **Mask-programmable PLD**s (i.e., **MPLD**s) can be made only by semiconductor manufacturers because connections are made by custom masks. Manufacturers need to make few masks for connections out of all of more than 20 masks, according to customer's specification on what logic functions are to be realized. Unlike mask-programmable PLDs, **field-programmable PLD**s (i.e., **FPLD**s) can be programmed by users and are economical only for small production volume, whereas MPLDs are economical for high production volume. Logic functions can be realized quicker on FPLDs than on MPLDs, saving payment of charges for custom masks for connections to semiconductor manufacturers, but they are larger, more expensive, and slower because of addition of electronic circuits for programmability.

Classification of PLDs is somewhat confusing in publications. Gate arrays to be described in Chapter 43 are regarded as PLDs in a broad sense in some publications, as shown in the following table, though PLDs may not include **field-programmable gate array**s (i.e., **FPGAs**) in some publications. Field-programmable PLDs and FPGAs, are regarded as FPGAs in a broad sense. FPGAs in some cases have arrays of PLDs inside and are sometimes called **complex PLD**s to differentiate them from PLDs which are simpler.



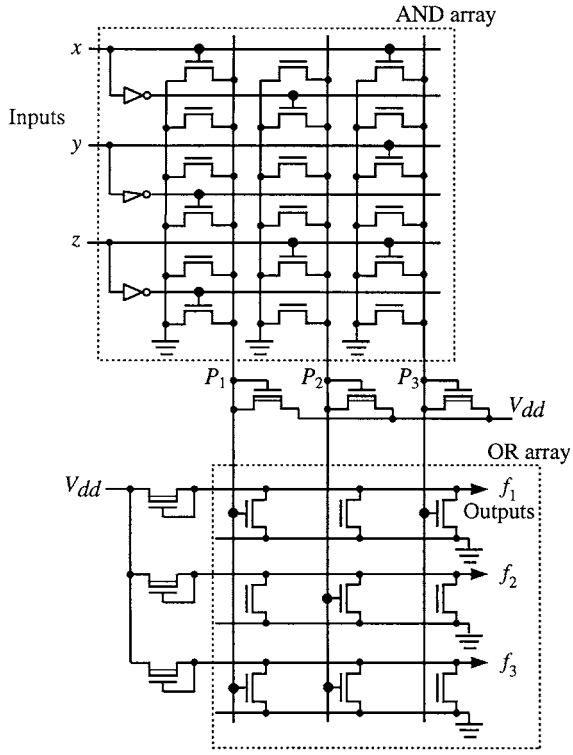
42.2 PLAs and Variations

Programmable logic arrays (i.e., PLAs) are one of programmable logic devices. Logic functions realized in PLAs can be as easily changed as software is.

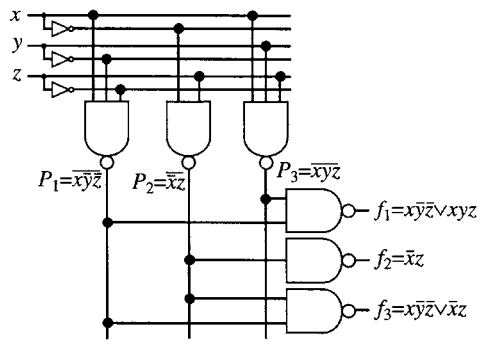
A **programmable logic array** (abbreviated as **PLA**), which was devised by Proebsting,[9] is a special type of ROM (which stands for **Read-Only Memory**), although its usage is completely different from that of ROMs. MOSFETS are arranged in a matrix on a chip, as illustrated in Fig.42.1(a). A PLA consists of an **AND array** and an **OR array**. In order to store logic expressions, connections between the MOSFET gates and the vertical lines in the AND array and also connections between the MOSFET gates and the horizontal lines in the OR array are set up by semiconductor manufacturers during fabrication according to customer' specifications. Since for these connections only one mask out of many necessary masks needs to be custom-made, PLAs are inexpensive when the production volume is high enough to make the custom preparation cost of the connection mask negligibly small. Because of low cost and design flexibility, PLAs are extensively used in VLSI chips, such as microprocessor chips for general computation and microcontroller chips for home appliances, toys, and watches.

When MOSFET gates are connected, as denoted by the large dots in Fig. 42.1(a), we have, $\overline{x\bar{y}\bar{z}}$, $\overline{\bar{x}z}$, $\overline{xyz}$, at the outputs P_1, P_2, P_3 of the AND array, respectively, since P_1, P_2, and P_3 represent the outputs of NAND gates, if negative logic is used with n-MOS. Here, **negative logic, where a high voltage and a low voltage are regarded as signal 0 and 1, respectively, is used for the sake of the convenience in deriving sums-of-products, i.e., disjunctive forms for the output functions f_1, f_2, and f_3.** (If positive logic is used, where a high voltage and a low voltage are regarded as signal 1 and 0, respectively, then P_1, P_2, and P_3 represent the outputs of NOR gates, and f_1, f_2, and f_3 are expressed in the forms of product-of-sums, i.e., in conjunctive forms. Since most people prefer disjunctive forms, negative logic is usually used in the case of PLAs.) Then, the outputs f_1, f_2, and f_3 of the OR-array also represent the outputs of NAND gates with P_1, P_2, P_3 as their inputs. Thus,

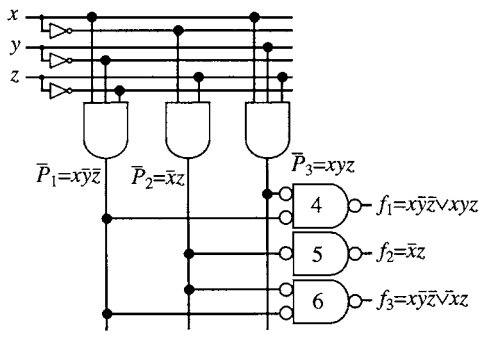
$$f_1 = \overline{P_1 P_3} = \bar{P}_1 \vee \bar{P}_3 = x\bar{y}\bar{z} \vee xyz$$
$$f_2 = \overline{P_2} = \bar{x}z$$
$$f_3 = \overline{P_1 P_2} = \overline{P_1} \vee \overline{P_2} = x\bar{y}\bar{z} \vee \bar{x}z$$



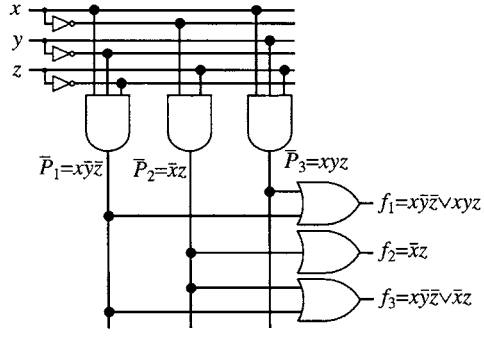
(a) PLA without flip-flops.



(b) Two-level NAND network which is equivalent to (a).



(c) Network converted from (b).



(d) Two-level AND-OR network which is equivalent to (a).

FIGURE 42.1 A PLA and equivalent logic networks.

(It is to be noted that in the AND array, the inputs x, y, and z have thier complements by inverters.) Therefore, the two arrays in Fig.42.1(a) represent a network of NAND gates in two levels, as illustrated in Fig. 42.1(b). This can be redrawn in Fig. 42.1(c) by moving the bubbles (i.e., inverters) to the inputs of the NAND gates 4, 5, and 6 without changing the outputs f_1, f_2, and f_3. Then, gate 4, for example, can be replaced by an OR gate because f_1 is

$$f_1 = \overline{P_1 P_3} = \bar{P}_1 \vee \bar{P}_3$$

by De Morgan's theorem. Thus, this is interpreted as a network of AND gates in the first level and OR gates in the second (output) levels, as illustrated in Fig. 42.1(d). This is the reason why the upper and lower matrices in Fig. 42.1(a) are called AND and OR arrays, respectively. The vertical lines which run through the two arrays in Fig. 42.1(a) are called the **product lines**, since they correspond to the product terms in disjunctive forms for the output functions f_1, f_2, and f_3 . Thus, any combinational network (or networks) of AND and OR gates in two levels can be realized by a PLA. The connections of MOSFET gates to horizontal or vertical lines are usually denoted by dots, as shown in Fig. 42.2.

Sequential networks can also be easily realized on a PLA, as shown in Fig. 42.2. Some outputs of the OR array are connected to the inputs of master-slave flip-flops (usually *J-K* master-slave flip-flops), whose outputs are in turn connected to the AND array as its inputs. More than one sequential network can be realized on a single PLA, along with many combinational networks. Flip-flops can be also realized inside the AND and OR arrays without providing them outside the arrays.

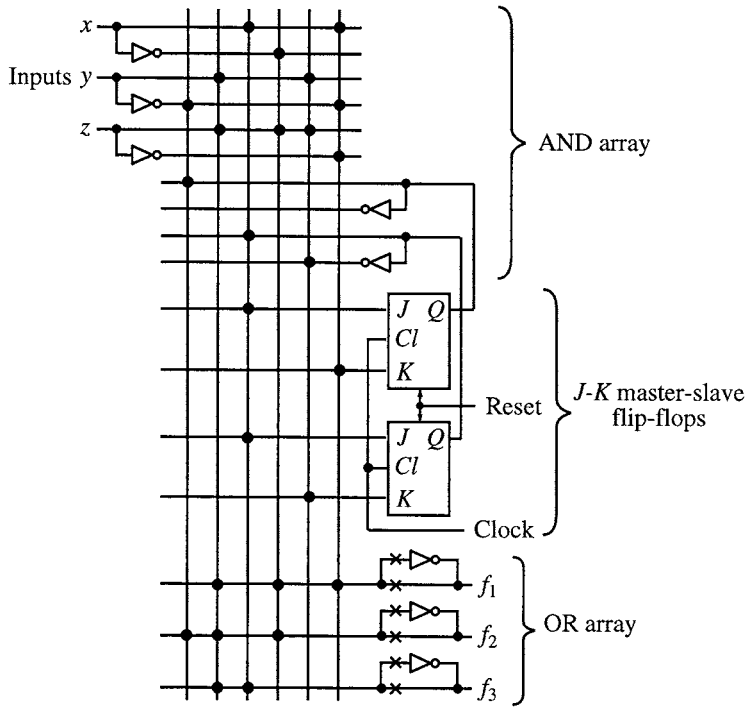


FIGURE 42.2 PLA with flip-flops and output-complementation choice.

In many PLAs, the option of an output f_1 or its complement $\bar{f}_1$ is provided in order to give flexibility, as illustrated in the lower right-hand corner of Fig. 42.2. By disconnecting one of the two ×'s at each output, we can have either f_1 or $\bar{f}_1$ as output, as illustrated in Fig. 42.3. When f_1 has too many products in its disjunctive form and cannot be realized on a PLA, its complement $\bar{f}_1$ may have a sufficiently small number of terms to be realizable on the PLA, or vice versa.

If the number of product lines in a PLA is too many, each horizontal line gets too long with a significant increase in parasitic capacitance. Then, if the majority of the MOSFET gates provided are connected to this horizontal line, the input or its inverter has too many fan-out connections on this horizontal line. Similarly, the total number of horizontal lines cannot be too large. In other words, the array size of a PLA is limited because of speed considerations. In contrast, the size of a ROM can be much larger, since we can use more than one decoder, or use a complex decoding scheme.

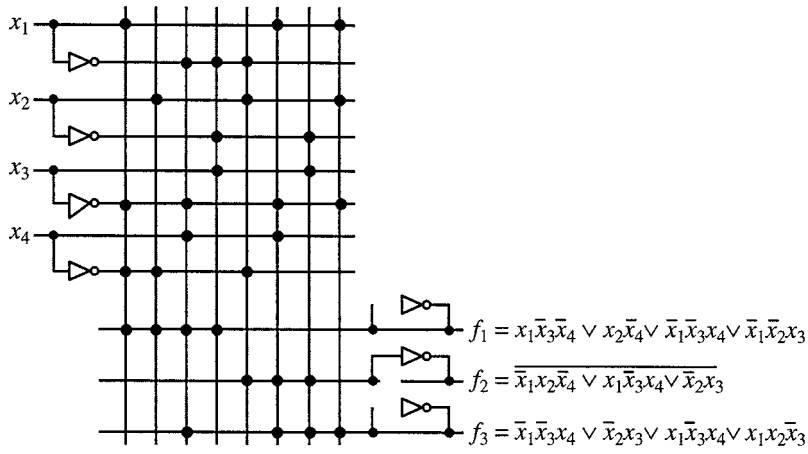


FIGURE 42.3 A PLA minimized for the given functions, f_1, f_2, and f_3.

The PLAs whose connections of some MOSFETs to lines are made by custom masks by semiconductor manufacturers are called **mask-programmable PLAs** (abbreviated as **MPLAs**).

A **field-programmable PLA** (abbreviated as **FPLA**) is also available, using fuses or anti-fuses, where unlike fuses, anti-fuses are initially not connected but can be connected by applying a voltage that is higher than a normal voltage. In an FPLA, a user can set up a dot pattern by blowing fuses or connecting anti-fuses connected to some MOSFETs by temporarily feeding excessively high voltages. In this realization, a special electronic circuit to blow fuses or to connect anti-fuses must be provided in addition to the PLA arrays, and this adds extra area to the entire area. In large-volume production, FPLAs are more expensive due to this extra size than PLAs, but when users need a small number of PLAs, FPLAs are much cheaper and convenient, since users can program FPLAs by themselves, inexpensively and quickly, instead of waiting weeks for the delivery of MPLAs from semiconductor manufacturers.

In contrast to the above FPLAs based on fuses, FPLAs whose undesired connections are disconnected by laser beam are also available. In this case, the chip size is smaller than that of the above FPLAs, since the electronic circuits to blow fuses are not necessary, but special laser equipment is required.

FPLAs are less expensive than MPLAs for small production volumes, although for high production volumes, MPLAs are much less expensive. In particular, when designers want to use MPLAs but their design is not completely debugged, they should try their design ideas with FPLAs and then switch to MPLAs only after debugging is complete, because if a semiconductor manufacturer is already working on MPLAs, sudden interruption of the work due to the discovery of design mistakes is unprofitable for both the manufacturer and the customer.

42.3 Logic Design with PLAs

Minimization techniques for multiple-output logic functions discussed in Chapter 27 can be used to minimize the size of a PLA. If the number of AND gates in a two-level AND-OR network (i.e., the number of distinct multiple-output prime implicants in disjunctive forms) for the given output functions is minimized, we can minimize the number of product lines, *t*. thus, the array size $(2n + m)t$ of a PLA is minimized when the PLA has n inputs, m outputs, and t product lines, where n and m are given. Also, if the total number of connections in a two-level AND-OR network is minimized as the secondary objective, as we do in the minimization of a multiple-output logic function, then the number of dots (i.e., connected intersections) of the product lines and the horizontal lines) in the PLA is minimized. Therefore, the derivation of a minimal two-level network with AND and OR gates by the minimization techniques known in switching theory is very important for the minimal and reliable design of PLAs.

The PLA show in Fig. 42.3, for example, is minimized for the given functions f_1, f_2, and f_3, with 8 product lines and array size, $(2 \times 4 + 3) \times 8 = 88$.

However, the minimization of the number of connections in a minimal two-level AND-OR network may not be as important as the minimization of the number of AND gates, although it tends to reduce the power consumption, because the chances of faulty PLAs can be greatly reduced by careful fabrication of chips. But the PLA size is determined by the number of AND gates and cannot be changed by any other factors. Also, instead of making connections (i.e., dots) as they become necessary on a PLA, a PLA is sometimes prepared by disconnecting unnecessary connections by laser beam or by blowing fuses after it has been manufactured with all MOSFET gates connected to the lines. In this case, the chances of faults can be reduced by increasing the number of connections (i.e., the number of dots) in the two-level AND-OR network.

For comparison with a PLA, the MOS realization of a ROM is shown in Fig. 42.4. The upper matrix is a decoder which has 2^n vertical lines if there are n input variables. The lower matrix stores information by connecting or not connecting MOSFET gates. Figure 42.4 actually realizes the same output functions (in negative logic) as those in Fig. 42.1(a). The AND array in Fig. 42.1(a) is essentially a counterpart of the decoder in Fig. 42.4, or the decoder may be regarded as a fixed AND array with 2^n product lines, which is the maximum number of the product lines in a PLA. The AND array in Fig. 42.1(a) has only three vertical lines, whereas the decoder in Fig. 42.4 has eight fixed vertical lines. This indicates the compact information packing capability of PLAs. PLAs are smaller than ROMs, although the packing advantage of PLAs varies, depending on functions. For example, if we construct a ROM that realizes the functions of the PLA of Fig. 42.3, in a manner similar to Fig. 42.4, the decoder consists of 8 horizontal lines and 16 vertical lines, and the lower matrix for information storage consists of 16 vertical lines and 3 horizontal lines. Thus, the ROM requires the array size of $16 \times (8 + 3) = 176$, compared with 88 in Fig 42.3.

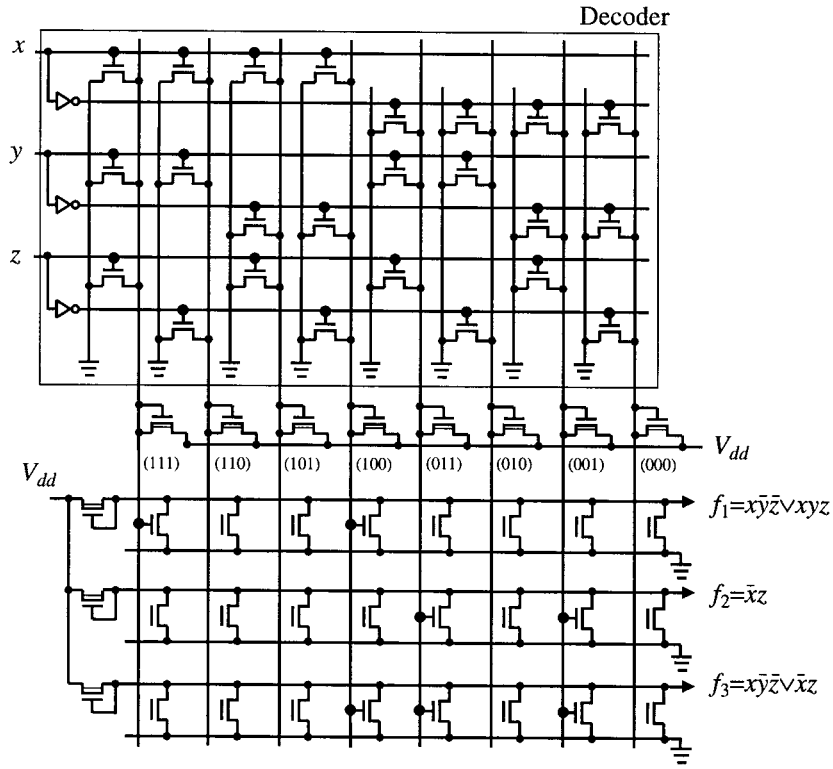


FIGURE 42.4 ROM that corresponds to the PLA in Fig. 42.1.

Generally, the size difference between PLAs and ROMs sharply increases as the number of input variables increases.

A PLA, however, cannot store some functions, such as $x_1 \oplus x_2 \oplus \ldots \oplus x_n$ if n is large, because 2^{n-1} product lines are required and the number of these lines is excessively large for a PLA. (The horizontal lines become too long with excessive fan-out and parasitic capacitance.) However, we can store these functions in a ROM with an appropriate decoding scheme.

Of course, in the case of ROMs, storing a truth table without worrying about conversion of given logic functions into a minimal sum is convenient, although it makes the ROM size bigger than the PLA size.

Minimal two-level networks of AND and OR gates for the absolute minimization of the PLA size can be derived by the minimization methods discussed in earlier chapters, if a function to be minimized has either at most several variables, or many more variables but with a simple relationship among its prime implicants.[8] But otherwise, we have to be content with near-minimal networks instead of minimal networks. In many cases, efforts to reduce the PLA size, even without reaching an absolute minimum, result in significant size reduction. Also, CAD programs have been developed with heuristic minimization methods,[12,13] such as the one by Hong et al.,.[7] which was the first powerful heuristic procedure drastically different from conventional minimization procedures. MINI, PLA minimization program of Hong, et al., was later improved to ESPRESSO by Rudell, Brayton, et al.[1,10,11] Recently, however, Coudert and Madre[2–6] developed a new method for absolute minimization by implicitly expressing prime implicants and minterms using BDDs described in Chapter 26. By this method, absolute minimization of functions with greater numbers of variables is more feasible than before, although it is still time-consuming.

42.4 Dynamic PLA

If we want to realize a PLA in CMOS, instead of static nMOS circuit that has been discussed in Chapter 30, Section 30.3, in order to save power consumption, then a PLA in CMOS requires a large area because we need pMOS and nMOS subcircuits. Thus, instead of static CMOS, the dynamic CMOS illustrated in Fig. 42.5(a) is usually used. During the absence of a clock pulse of the first- and second-phase clocks, ϕ_1 and ϕ_2 (i.e., during $\phi_1 = \phi_2 = 0$ (low voltage, using positive logic)) shown in Fig. 42.5(b), pMOSFETs, T_1, T_2, and T_3, become conductive and nMOSFETs, T_4, T_5, and T_6 become non-conductive precharging vertical lines, P_1, P_2, and P_3. When a clock pulse of the first-phase clock, ϕ_1, appears but a clock-pulse of the second-phase clock, ϕ_2, does not appear yet, i.e., when $\phi_1 = 1$ (high voltage) and $\phi_2 = 0$, pMOSFETs, T_1, T_2, and T_3, become non-conductive and nMOSFETs, T_4, T_5, and T_6, become conductive. Then, depending on the values of x, y, and z, some verticle lines, P_1, P_2, and P_3 are discharged through some of the nMOSFETs in the AND array. (For example, if y = 0 (low voltage), P_1 is discharged through nMOSFETs A.) A clock pulse of the second-phase clock, ϕ_2, is still absent (i.e., $\phi_2 = 0$), so pMOSFETs, T_7, T_8, and T_9, become conductive and nMOSFETs T_{10}, T_{11}, and T_{12}, become non-conductive, precharging horizontal lines, f_1, f_2, and f_3. When a clock pulse of the first-phase clock, ϕ_1, is still present, and a clock pulse of the second-phase clock, ϕ_2, appears, i.e., when $\phi_1 = \phi_2 = 1$, pMOSFETs, T_7, T_8, and T_9, become non-conductive and nMOSFETs, T_{10}, T_{11}, and T_{12}, become conductive. Then, some of horizontal lines, f_1, f_2, and f_3, are discharged through some of the nMOSFETs in the OR array, depending on which of the vertical lines, P_1, P_2, and P_3, are still charged.

42.5 Advantages and Disadvantages of PLAs

PLAs, like ROMs which are more general, have the following advantages over random-logic gate networks, where random-logic gate networks are those that are compactly laid out on an IC chip:

1. There is no neeed for the time-consuming logic design of random-logic gate networks and even more time-consuming layout.
2. Design checking is easy, and design change is also easy.
3. Layout is far simpler than that for random-logic gate networks, and thus is far less time-consuming.
4. When new IC fabrication technology is introduced, we can use previous design information with ease but without change, making adoption of the new technology quick and easy.

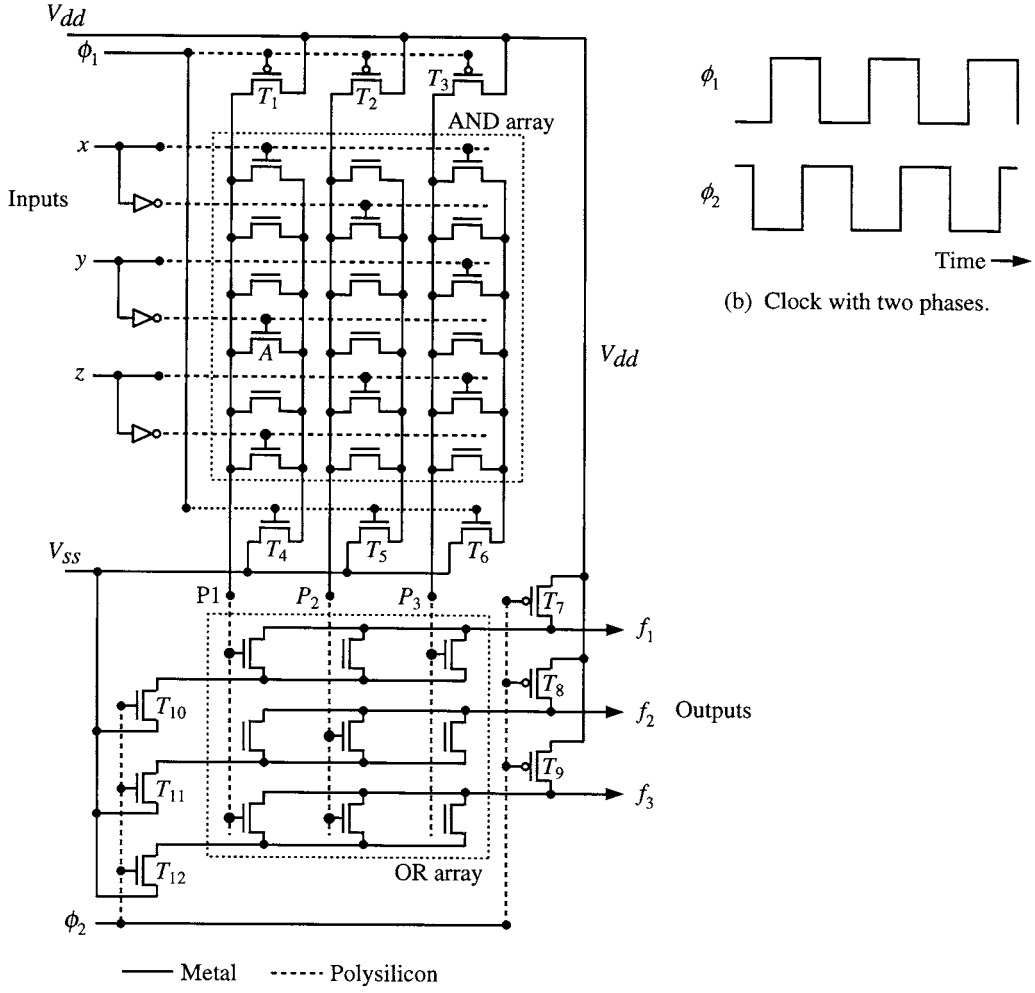


FIGURE 42.5 Dynamic PLA.

5. Only the connection mask needs to be custom-made.
6. Considering all these, PLA is a very inexpensive approach, greatly shortening desing time.

PLAs have the following disadvantages compared with random-logic gate networks:

1. Random-logic gate networks have higher speed than PLAs or ROMs.
2. Random-logic gate networks occupy smaller chip areas than PLAs or ROMs, although the logic design and the layout of random-logic gate networks are far more tedious and time-consuming.
3. Also, with large production volumes, random-logic gate networks are cheaper than PLAs or ROMs.

PLAs have the following advantage and disadvantage, compared with ROMs:

- For storing the same functions or tasks, PLAs can be smaller than ROMs; generally, the size difference sharply increases as the number of input variables increases.
- The small size advantages of PLAs diminishes as the number of terms in a disjunctive form increases. Thus, PLAs cannot store complex functions, i.e., functions whose disjunctive forms consist of many product terms.

Applications of PLAs

Considering the above advantages and disadvantages, PLAs have numerous unique applications. A microprocessor chip uses many PLAs because of easy of design change and check. In particular, PLAs are used in its control logic, which is complex and requires many changes, even during its design. Also, PLAs are used for code conversions, microprogram address conversions, decision tables, bus priority resolvers, and memory overlay.

When a new product is to be manufactured in small volume or test-marketed, PLAs is a choice. When the new product is well received in the market and does not need further changes, PLAs can be replaced by random-logic gate networks for low cost for high volume production and high speed. Also, a full-custom design approach is very time-consuming, probably taking months or years, but if PLAs are used in the control logic, a number of different custom-design chips with high performance can be made quickly by changing only one connection mask for the PLAs, although these chips cannot have drastically different performance and functions.

42.6 Programmable Array Logic

A **programmable array logic** (**PAL**) is a special type of a PLA where the OR array is not programmable. In other words, in a PAL, the AND array is programmable but the OR array is fixed; whereas in a PLA, both arrays are programmable. The advantage of PALs is the elimination of fuses in the OR array in Fig. 42.1(a) and special electronic circuits to blow these fuses. Since these special electronic circuits and programmable OR array occupy a very large area, the area is significantly reduced in PAL. Since single-output, two-level networks (i.e., many AND gates in the first level and one OR gate as the network output) are needed most often in desing practice, many single-output two-level networks which are mutually unconnected are placed in some PAL packages.

In digital systems, many non-standard networks are still used because designers want to differentiate their computers from competitors'. But logic functions that designers want to have are too diverse to be standardized by semiconductor manufacturers. When off-the-shelf IC packages for standard networks, including microprocessors and their peripheral networks, are assembled on pc boards, many non-standard networks are usually required for interfacing them to other key networks or for minor modifications. So, they require many discrete components and IC packages, each of which has a smaller number of transistors, in addition to a microprocessor package with millions of gates, occupying a significant share of the areas on pc boards. Now, we can make connections inside PALs, instead of custom-making pc boards. Custom-made pc boards are expensive and time-consuming because connection patterns on pc boards need to be designed, these pc boards need to be manufactured and then the holes of pc boards have to be soldered to the pins of IC packages. The replacement by PAL packages can substantially reduce the area, time, and cost. If we consider related factors such as reductions of cabinet size, power consumption, and fans, the significance of this reduction is further appreciated.

There are mask-programmable PALs and field-programmable PALs (i.e., FPALs). When logic design is not finalized and needs to be changed often, FPAL packages can reduce expense and time for repeatedly redesigning and remaking pc boards.

References

1. Brayton, R. K., G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, 1984.
2. Coudert, O., "Two-level logic minimization: an overview," *Integration, The VLSI Jour.*, pp. 97–140, Oct. 1994.
3. Coudert, O., "Doing two-level logic minimization 100 times faster," *Symposium on Discrete Algorithms (SODA)*, San Francisco, pp. 112–121, Jan. 1995.
4. Coudert, O., "On solving covering problems," *33rd DAC*, pp. 197–202, June 1996.

5. Coudert, O., J. C. Madre, H. Fraisse, and H. Touati, "Implicit prime cover computation: An overview," *Proc. SASIMI'93 (Synthesis and Simulation Meeting and Int'l Interchange)*, Nara, Japan, Oct. 1993.
6. Coudert, O. and J. C. Madre, "New ideas for solving covering problems," *32nd DAC*, San Francisco, pp. 641–646, June 1995.
7. Hong, S. J., R. G. Cain, and D. L. Ostapko, "Mini: A heuristic approach for logic minimization," *IBM JRD*, pp. 443–458, Sept. 1974.
8. Muroga, S., *Logic Design and Switching Theory*, John Wily & Sons, 1979. (Now available from Krieger Publishing Co.)
9. Proebsting, R., *Electronics*, p. 82, Oct. 28, 1976.
10. Rudell, R. L. and A. L. Sangiovanni-Vincentelli, "Multiple valued minimization for PLA optimization," *IEEE Transactions on CAD*, vol 6, no 5, pp. 727–750, Sept. 1987.
11. Sasao, T., "An application of multiple-valued logic to a design of programmable logic arrays," *Proc. of Int'l Symposium on Multiple-Valued Logic*, 1978.
12. Smith, M. J. S., *Application-Specific Integrated Circuits*, Addison-Wesley, 1997.
13. Venkateswaran, R. and P. Mazumder, "A survey of DA techniques for PLD and FPGA based systems," *Integration, the VLSI Jour.*, vol. 17, no. 3, pp. 191–240, Nov. 1994.

Muroga, S. "Gate Arrays"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

43

Gate Arrays

Saburo Muroga
University of Illinois at Urbana-Champaign

43.1 Mask-Programmable Gate Arrays

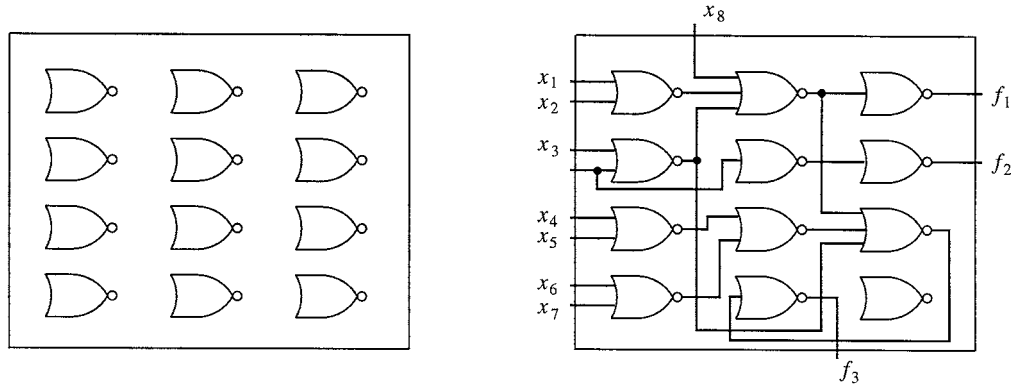
Among all ASIC chips, gate arrays are most widely used because with gate arrays, we can easily realize logic functions that require a far more logic gates than with PLAs. If PLAs are used, such logic functions would require far larger area and delay time. Also, design time with gate arrays is shorter than with the standard cell design approach to be described in Chapter 45.

A gate array is an IC chip on which gates are placed in matrix form without connections among the gates, as illustrated in Fig. 43.1(a). By connecting gates, we can realize logic networks, as exemplified in Fig. 43.1(b). But actually, logic gates are not realized in a gate array. Instead of gates, cells, each of which consists of unconnected components, are arranged in matrix form, and each cell can realize one of a few different types of gates by connecting these components. Then, by connecting these gates as illustrated in Fig. 43.1(b), networks can be realized. Only two or three masks for connections and contact windows (i.e., small metallic areas between MOSFETs and connections) have to be custom-made, instead of all two dozen masks required for full-custom design. Also, because only the connection layout, along with the placement of gates, needs to be considered, CAD can be effectively used, thereby greatly reducing the layout time. Thus, the design with gate arrays is very inexpensive and quick, compared with full-custom design. Gate arrays of CMOS have been extensively used in many computers.[1–3]

In Fig. 43.1(b), connections among gates are run in narrow strips of space between columns or rows of gates. These strips of space are called **routing channels**. In gate arrays that were commercially available for the first time, routing channels were provided between columns or rows of logic gates. Now, gate arrays without routing channels are also available, running connections over gates, as it becomes easy to do so because many metal layers are available. Such gate arrays are called **sea-of-gate array**s. Gate arrays with a large number of gates are usually sea-of-gates without routing channels. But gate arrays with routing channels are still used for those with a small number of gates. A relatively large number of pads are necessary, even for such small gate arrays. Then, the number of pads determines the area size of gate arrays and it does not matter whether or not routing channels are provided. If routing channels are provided, then two metal layers are sufficient for a higher yield. Gate arrays in sea-of-gates require three or more metal layers, so they are expensive.

43.2 CMOS Gate Arrays

CMOS gate arrays are commercially available from many manufacturers in slightly different layout forms. As an example, Fig. 43.2 shows a cell of a CMOS gate array, where a pair of pMOSFETs and a pair of nMOSFETs are placed on the left and right, respectively, without connections between them. The NAND



(a) Before making connections

(b) After connections made

FIGURE 43.1 Gate array.

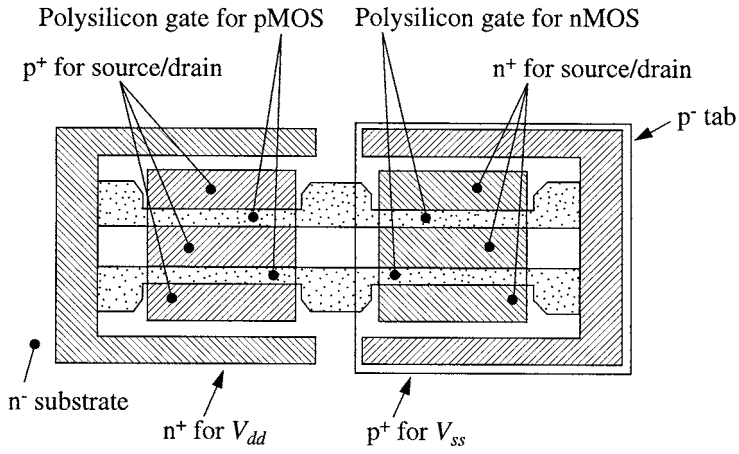


FIGURE 43.2 A cell of CMOS gate array. (Courtesy of Fujitsu Ltd. With permission.)

gate shown in Fig. 43.3(a) can be realized by connecting the components shown in Fig. 43.2 by two metal layers as shown in Fig. 43.3(b). These two metal layers are formed by forming the first metal layer shown in Fig. 43.3(c), the insulation layer (not shown), and then the second metal layer shown in (d). The inverter shown in Fig. 43.4(a) can be realized by connections as shown in Fig. 43.4(b).

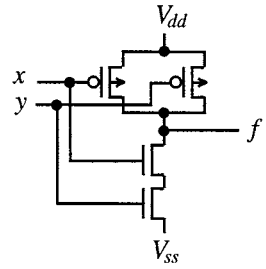
Many different patterns other than that in Fig. 43.2 are available for the components of a cell.

43.3 Advantages and Disadvantages of Gate Arrays

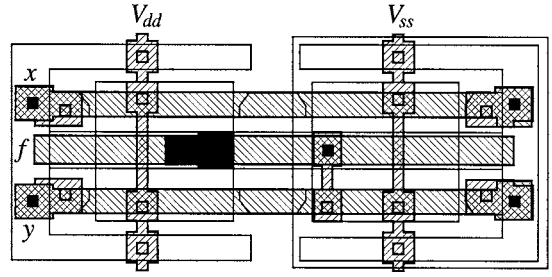
Gate arrays have the following advantages:

1. Even when fabrication technology or electronic circuitry changes, gate arrays can be designed in a short time. In other words, only one cell of transistors (or few different cells) needs to be carefully designed and laid out, and this layout can be repeated on a chip.
2. After designers design logic gate networks (although this is still very time-consuming, the minimization of the number of gates or delays, under constraints such as maximum fan-out, would be a designer's primary concern, but connections are a less significant problem), CAD programs

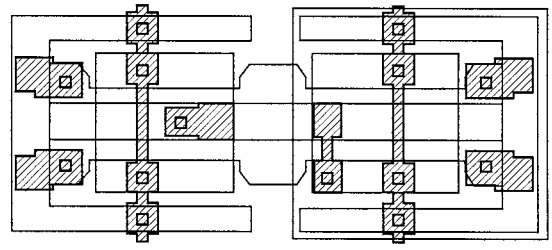
(a) NAND gate.



(b) The connections consist of two metal layers shown in (c) and (d).



(c) First metal connection layer.



(d) Second metal connection layer.

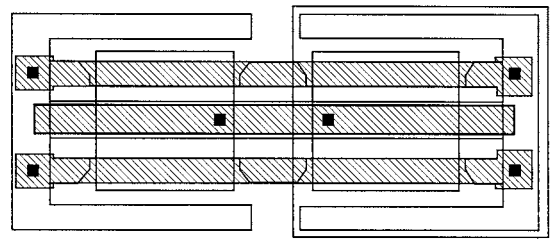
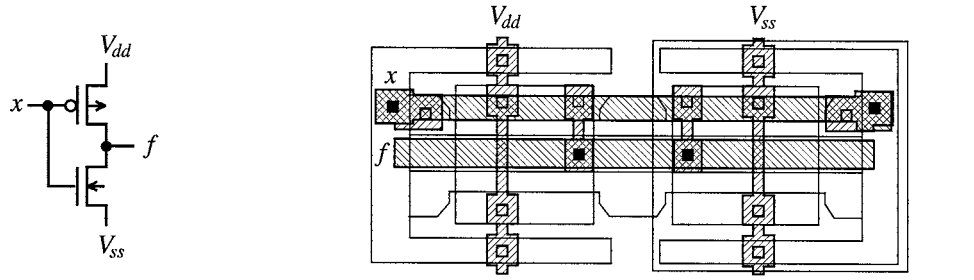


FIGURE 43.3 Connection layout example of the cell in Fig. 43.2. (Courtesy of Fijitsu Ltd. With permission.)

automatically do the placement of logic gates and the routing of connections on a chip, although in complex cases, placement, which is more difficult than routing, must be done manually by designers, possibly using a placement library of standard networks. (Often, a small percent of the connections cannot be processed well by CAD programs and must be rerouted manually. Thus, when the number of connections is very large, even a few percent means a large number of connections need to be rerouted. So, it is important to reduce this percentage.) It is to be noted that because the gate positions are prefixed on the array, CAD for placement and routing becomes much easier to use than other cases. For the above reasons, the layout time is greatly reduced, shortening the design time, and consequently design expenses. (Delivery time by vendors is usually at least a few weeks.)



(a) Inverter. (b) Connection layout to realize (a) (Courtesy of Fujitsu Ltd.)

FIGURE 43.4 Connection layout example of the cell in Fig. 43.2.

3. Only a few masks for connections and contact windows must be custom-made for each design case, and all other masks are common to all cases, spreading the initial investment for the preparation of all these common masks over all cases. Thus, gate arrays are cost-effective in low production volumes of hundreds or thousands.
4. Speed is improved over logic networks realized with off-the-shelf packages or PLAs because interconnections among gates are shortened on the average (most interconnections are inside gate array chips rather than on pc boards).
5. The power consumption is reduced, compared with logic networks realized with off-the-shelf packages or PLAs.
6. Logic gates and connections are placed in a very different manner from full-custom-designed networks, where logic gates and connections that are functionally closely related are placed in nearby places. Thus, even if competitors look at the layout of the gate array chips, the layouts are too cryptic to understand the nature of logic networks. In this sense, gate arrays are good for protection of proprietary logic design.

On the other hand, gate arrays have the following disadvantages.

1. Chip size is large because logic gates are not compactly laid out. For example, each pair of nMOSFETs is placed in an individual p^--tab without sharing p^--tab with many other nMOSFETs, as can be seen in Fig. 43.2.
2. The percentage of unused gates is possibly high, for the following reasons. Depending on the types of networks or which parts of a large network are placed in a gate array chip, all spacings provided for connections can be used up (by taking a detour if the shortest paths are filled up by other connections), or all the pins of the chip can be used up by incoming and outgoing connections. In either case, many gates may not be used at all, and fewer than half the gates on a chip are used in some cases. Because of these disadvantages, the average size of a gate array chip is easily four or five times as large as that of a full-custom-designed chip, or it can be even greater, for the same logic networks. The cost difference would be greater (the cost is not necessarily linearly proportional to chip size) for the same production volume.
3. It is difficult to keep gate delays uniform. As the number of fan-outs and the length of fan-out connections increase, delays increase dramatically. (If delay times of gates are not uniform, the network tends to generate spurious output signals.) In the case of full-custom design, the increase of gate delay by long or many-output connections of a gate can be reduced by redesigning the transistor circuit (e.g., increasing transistor size for delivering greater output power and accordingly reducing the delay). But such a precise adjustment is not possible in the case of gate arrays.

Responding to a variety of different user needs in terms of speed, power consumption, cost, design time, ease of change, and possibly others, a large number of different gate arrays are commercially available from semiconductor manufacturers or are used in-house by computer manufacturers. Different numbers of gates are placed on a chip, with different configuration capabilities. Some gate arrays contain memories, for example.

References

1. Okabe, M. et al., "A 400k-transistor CMOS sea-of-gate array with continuous track allocation," *IEEE J. Solid-State Circuits*, pp. 1280-1286, Oct. 1989.
2. Muroga, S., *VLSI System Design*, John Wiley & Sons, 1982.
3. Price, J. E., "VLSI chip architecture for large computers," in *Hardware and Software Concepts in VLSI*, Edited by G. Rabbat, Van Nostrand Reinhold Co., pp. 95-115, 1983.

Muroga, S. "Field-Programmable Gate Arrays"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

44

Field-Programmable Gate Arrays

Saburo Muroga
University of Illinois at Urbana-Champaign

44.1 Introduction

Field-programmable gate arrays or **FPGAs** are programmable by users. In other words, users easily and inexpensively realize their own logic networks in hardware, using FPGAs. The change of the logic networks is as easy as software is. FPGAs, however, have a greater variety of hardware and architecture than PLAs or gate arrays. If fuses or anti-fuses are used for hardware programmability, logic functions realized on an FPGA cannot be changed, once realized. But the addition of random-access memory (RAM) to hardware of FPGAs by Freeman[6,7] such that logic functions can be easily changed by changing information stored in the RAM substantially has enhanced the usefulness of FPGAs. By storing information into RAMs of FPGAs, logic functions can be rewritten in a short time as frequently as we need. In this sense, FPGAs with RAMs are not a straightforward extension of the gate arrays in Chapter 43, which are mask-programmable, and are very different from the gate arrays. With the changeability of the logic functions on an FPGA by changing information in its RAMs, the nature of programmability of an FPGA is essentially the same as that of software which is stored in the RAM of a general-purpose computer. With FPGAs, debugging or prototyping of new design can be done as easily and quickly as software. But an FPGA performs much faster than software on computer.

FPGAs have other types of hardware, in addition to those with RAMs. Thus, hardware of FPGAs consists of PLDs, logic gates, random-access memory, and often other types of components such as non-volatile memory. FPGAs from different manufacturers have different organization of PLDs, logic gates, random-access memory, and other types of components. In other words, different manufacturers have FPGAs in different architectures, but all of them have the same common feature: that the layout of a unit is repeated in matrix form. In this case, the unit is a circuit consisting of PLDs, logic gates, random-access memory, and other types of components that is far more complex than "gates" which are repeated in matrix in gate arrays. Logic networks realized in FPGAs are slower by two or three orders of magnitude than those realized in full-custom design, but are much faster by several orders than simulation of logic functions by software. Even application programs can be run on FPGAs and perform much faster than on general-purpose computer in many cases.

As the price of FPGAs goes down with higher speed, FPGAs are replacing other semi-custom design approaches in many applications.

44.2 Basic Structures of FPGAs

In the case of mask-programmable gate arrays, designers have to wait a few weeks for delivery of finished gate arrays from semiconductor manufacturers because the semiconductor manufacturers must prepare custom masks (although the number of custom masks for gate arrays is fewer than the case of the standard-cell library approach described in Chapter 45). With FPGAs, designers can realize their design on FPGA chips by themselves in minutes. Thus, FPGAs are becoming popular.[1,2,8–10]

Several different types of structures for FPGAs are available commercially. All of them have a basic structures that consists of many logic blocks or logic cells, accompanied by a large number of pre-laid lines for connecting these logic blocks. So, some manufacturers call FPGAs **logic block arrays** (**LBA**s). One has a structure similar to a gate array with routing channels where each logic cell in a gate array is replaced with a logic block, as shown in Fig. 44.1. Another one is similar to sea-of-gate array, as shown in Fig. 44.2 illustrated with 16 logic blocks. Also, there is a structure similar to standard cells (to be discussed in the next chapter) where there are routing channels between a pair of rows of logic blocks, as shown in Fig. 44.3. There is a structure where outputs of logic blocks are connected to the inputs of other logic blocks through bus lines, as shown in Fig. 44.4.

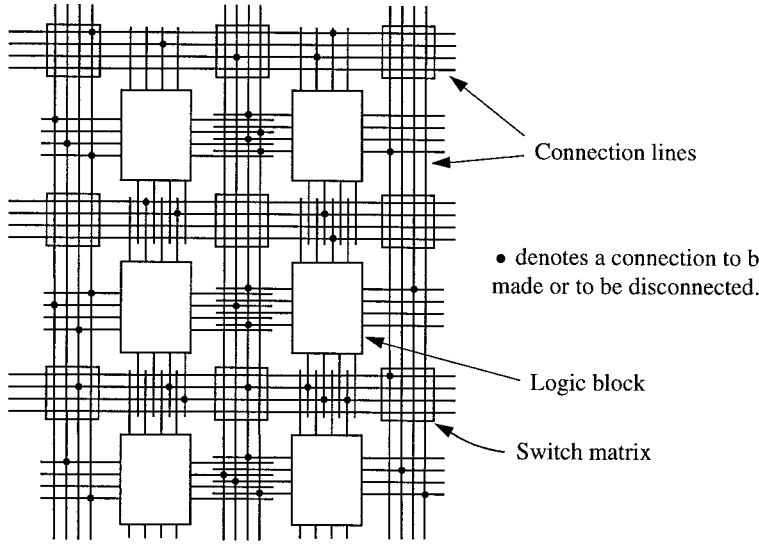


FIGURE 44.1 FPGA type of gate array with routing channels.

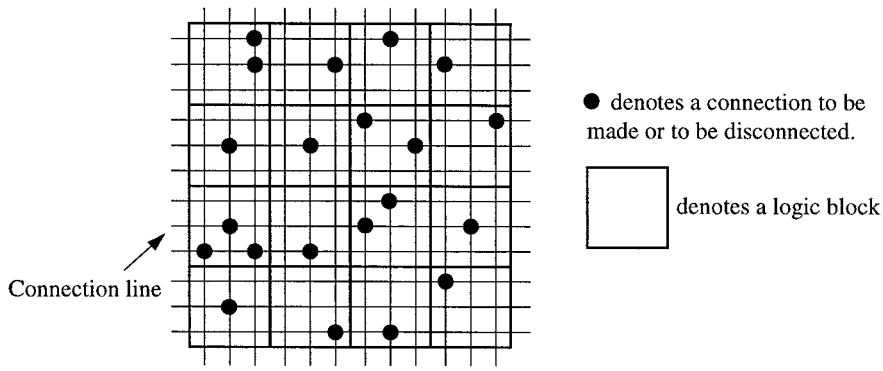


FIGURE 44.2 FPGA type of sea-of-gate array.

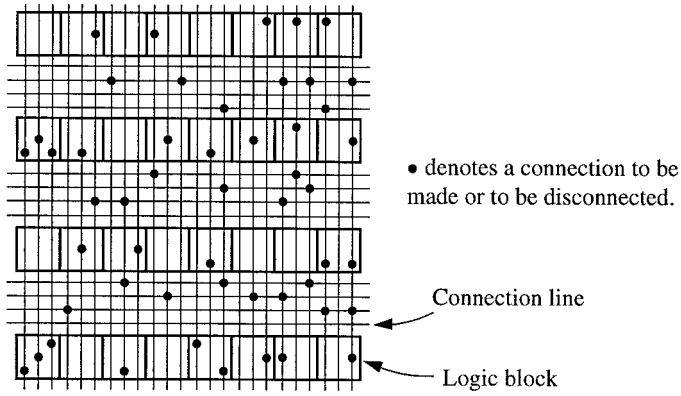


FIGURE 44.3 FPGA type of row-based array.

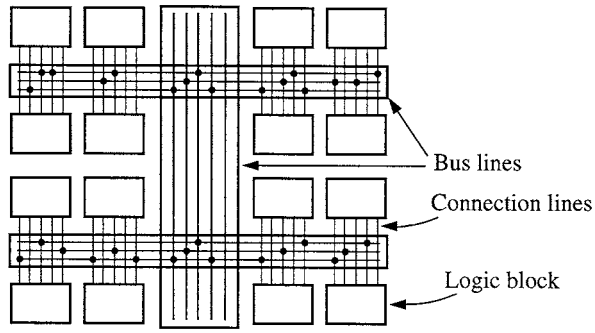


FIGURE 44.4 FPGA type of bus-based array.

The internal structure of logic blocks or logic cells differs, depending on the manufacturer. A logic block consists of SRAMs (used as look-up tables), PALs, NAND gates, along with multiplexers, flip-flops, and others. Lines are pre-laid horizontally and vertically and are connected to the inputs and outputs of logic blocks by **programmable switches**. Various programmable switches, such as fuses, anti-fuses, RAMs, and non-volatile memories, are provided by different manufacturers. Each line actually consists of many short line segments and only necessary line segments are connected in order not to add unnecessary delay due to parasitic capacitance by using an excessive number of line segments. Line segments are also connected by programmable switches.

In addition to these logic blocks and pre-laid lines, there are different types of input/output control blocks, that is, blocks for inputs for receiving signals from the outside of the FPGA and for outputs for sending signals to the outside.

Each programmable switch consists of many switching elements. A typical FPGA contains hundreds of thousands or more switching elements and how these elements are realized is essential for performance, costs, and size of the FPGA. Fuses, anti-fuses, SRAMs, and non-volatile memories are used as switching elements. They are either volatile or non-volatile, either rewritable or non-rewritable, and all have different programming times (or writing times). Thus, depending on which of these are used, different FPGAs have significantly different applications.

44.3 Various Field-Programmable Gate Arrays

FPGAs have different basic structures, as already discussed. If we look at their details beyond this, there are different types of FPGAs, as follows.

FPGAs Based on SRAMs

FPGAs based on SRAMs connect lines by controlling a pass transistor, a transmission gate, or a multiplexer, each of which is controlled by a flip-flop (i.e., one memory cell of SRAM) as shown in Fig. 44.5. Any horizontal line and a vertical line shown in Fig. 44.1 can be connected at some of their cross points (shown with dots in Fig. 44.1). Suppose a flip-flop is used. If the flip-flop is on, these two connection lines are connected; if it is off, they are not. By connecting horizontal connection lines and vertical connection lines at appropriate cross points in this manner, the outputs of one logic block can be connected to the inputs of other blocks.

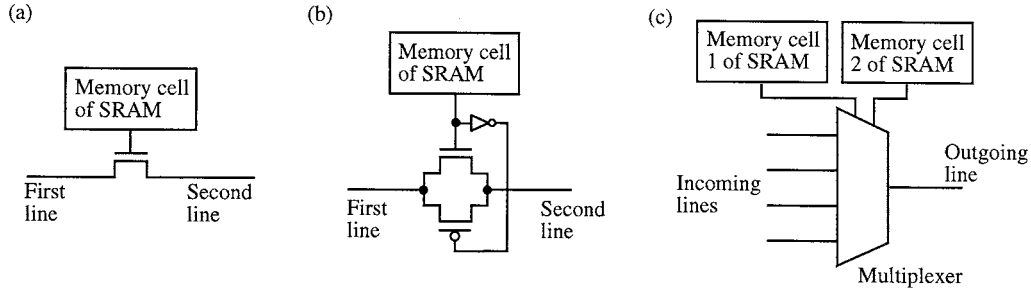


FIGURE 44.5 Connection of lines controlled by SRAM.

FPGAs based on SRAMs are available from Xilinx, Atmel (previously Concurrent Logic), and others. Lines are connected by transfer gates (or pass transistors) in Fig. 44.5(a), transmission gates in (b), or multiplexers in (c) that are controlled by ON or OFF states of memory cells of SRAM. A multiplexer is typically used to connect one of several incoming lines to one outgoing line.

FPGAs based on SRAMs were started by Xilinx.[6,10] Many logic blocks that contain SRAMs and flip-flops are provided in FPGAs of this type, along with many pre-laid connection lines, as illustrated in Fig. 44.1. Each block is connected to nearby one-line segments, as shown in Fig. 44.6. A switch matrix shown with a square is a set of multiplexers, where any one outgoing line segment can be connected to any of many incoming line segments. Besides these one-line segments, where by connecting many of them through switch matrices, long lines can be formed, there are global long lines that can be formed by

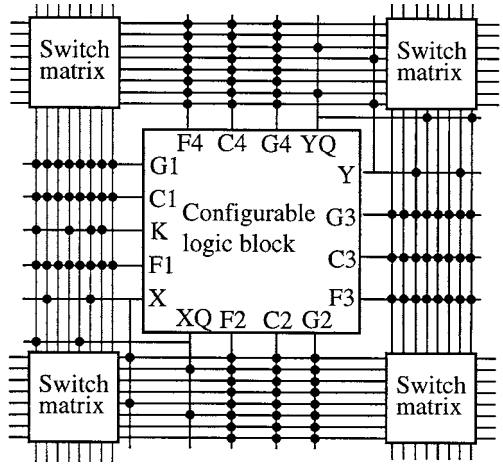


FIGURE 44.6 Typical connection of a configurable logic block to single line sections.

connecting fewer segments, each of which has longer segment length. (Note that a long line formed by connecting many one-line segments has a long delay because delays of many switch matrices are added to the delay of the lines due to parasitic capacitance.) In this sense, delay times on lines that consist of many line segments are unpredictable. Each logic block consists of a RAM along with several flip-flops, as shown in Fig. 44.7. Xilinx calls logic blocks **configurable logic block**s (**CLB**s). Users can store logic functions in these SRAMs and connect the blocks by lines as they want.

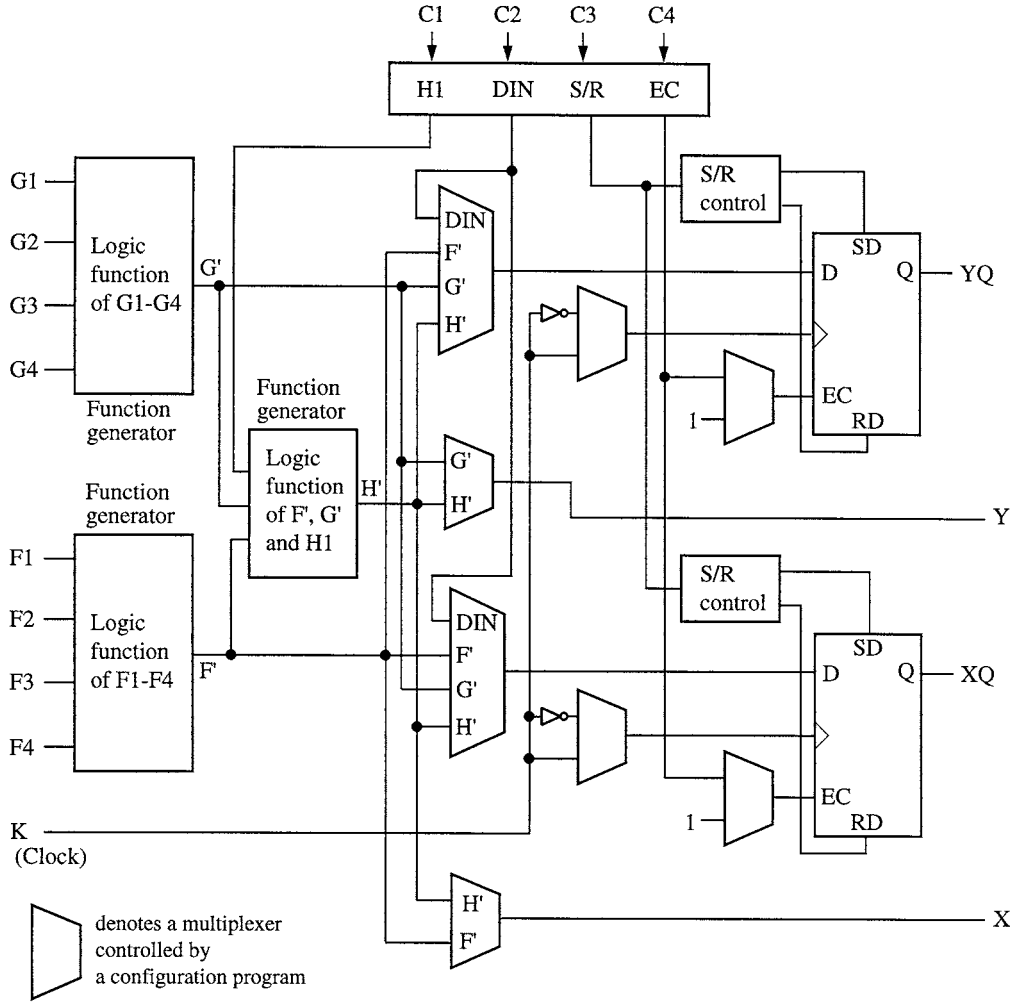


FIGURE 44.7 Simplified block diagram of configurable logic block of XC4000, FPGA of Xilinx, Inc.

Each CLB packs a pair of flip-flops and two independent four-input function generators, as illustrated in Fig. 44.7. The two function generators whose outputs are labeled F′ and G′ are realized with RAMs and are each capable of realizing any logic function of their four inputs, offering designers sufficient flexibility, because most combinational logic functions doe not need more than four inputs. Thirteen CLB inputs provide access to the function generators and flip-flops. These inputs and four CLB outputs connect to the programmable interconnect resources outside the block. Four independent inputs, F1–F4 or G1–G4, are provided to each of the two function generators. The function generators are used as look-up table memories; therefore, the propagation delay is independent of the function being realized. A third function generator, labeled H′, can realize any logic function of its three inputs, F′ and G′, and a

third input from outside the block (H1). Signals from the function generators can exit the CLB on two outputs; in other words, F′ or H′ can be connected to the X output, and G′ or H′ can be connected to the Y output. Thus, a CLB can realize any two independent functions of up to four variables, or any single function of five variables, or any function of four variables together with some functions of five variables, or it can even realize some functions of up to nine variables. Realizing a variety of functions in a single logic block reduces both the number of blocks required and the delay in the signal path, thus increasing both density and speed.

The CLB contains also edge-triggered D-type flip-flops with common clock (K) and clock enable (EC) inputs. A third common input (S/R) can be programmed as either an asynchronous set- or reset-signal independently for each of the two flip-flops; this input can also be disabled for either flip-flop. Each flip-flop can be triggered on either the rising or falling clock edge. The source of a flip-flop data input is programmable; it is driven either by the functions F′, G′, and H′, or the Direct In (DIN) block input. The flip-flops drive the CLB outputs, XQ and YQ.

In addition, each CLB function generator, F′ or G′, contains dedicated arithmetic/logic unit (not shown in Fig. 44.7) for the fast generation of carry and borrow signals, greatly increasing the efficiency and performance of adders, subtracters, accumulators, comparators, and counters. Multiplexers in the CLB map the four control inputs, labeled C1 through C4 in Fig. 44.7, into the four internal control signals (H1, DIN, S/R, and EC) in any arbitrary manner.

This FPGA has the following advantages:

- Once logic design is finished, logic functions can be realized by this FPGA in minutes by writing design information into SRAMs and flip-flops, as necessary.
- Quick realization of complex logic functions that require a large number of logic gates and therefore cannot be realized with FPLAs or PALs is the great advantage of this type of FPGA.

The disadvantages include the following:

- If the power supply to this FPGA is turned off, the information is lost and, consequently, design information must be written each time the power supply is turned on.
- Also, compared with mask-programmable gate arrays, its chip size is roughly 10 times larger and its speed is far slower. But its speed is still much faster than software run on a general-purpose computer.

If the number of CLBs is to be minimized, this type of FPGA presents a totally new type of logic design problem and there is no good design algorithm known. Traditional logic design problems are realization of given functions with a minimum number of simple logic gates of certain types (e.g., NOR gates or negative gates). But with this type of FPGA, we have to realize the given functions with logic functions that can fit in the SRAMs provided, where these functions can be far more complex than the NOR functions or negative functions that logic gates represent. But the number of CLBs is not necessarily to be minimized because after debugging logic design, final logic networks to be manufactured in large volume are usually realized with logic gates but without RAMs.

Using FPGAs of this type, the development time of mask-programmable gate arrays that are completely verified can be often shortened to less than half, and new computers can be introduced into the market quickly. In other words, when logic design is finished, computers can be shipped immediately in FPGAs to major customers for testing. After these customers find bugs, the designers fix them and the production of mask-programmable gate arrays can be started. Without FPGAs, the computers have to be shipped to customers for testing after manufacturing mask-programmable gate arrays — which takes several weeks. Then, when the customers find bugs, the designers start the production of the second mask-programmable gate arrays. Thus, when mask-programmable gate arrays are used, a new gate array must be manufactured, spending a few weeks, for each correction of bugs and testing by customers. The completion of the computer will be significantly delayed and with far greater expense. If the design is complex, the designers need repetition of tests by the customers. In contrast to this, if FPGAs are used,

the designers can send the debugged design to the customers online, and customers can start the second test instantly by revising the contents of the FPGAs.

Atmel's FPGA uses SRAMs, but logic blocks and architecture are similar to Actel's, unlike Xilinx's whose logic blocks contain SRAMs which are used as look-up tables. FPGAs of Plessey are based on SRAMs, but the architecture looks like the sea-of-gate array shown in Fig. 44.2. Altera also has FPGAs based on SRAMs, using the architecture with non-volatile memory.[10] FPGAs of Algotronix use SRAMs, but logic blocks are mostly multiplexers. AT&T Microelectronics also has FPGAs based on SRAMs similar to Xilinx's.

FPGAs Based on ROMs with Anti-fuses

There are FPGAs based on ROMs that are programmable with anti-fuses. These FPGAs are available from Actel, QuickLogic, and Crosspoint Solutions. Connection of lines in this type of FPGA can be made by the use of anti-fuses. Once an anti-fuse becomes conductive, it cannot be non-conductive again, unlike FPGAs based on SRAMs. Typical applications utilizing 85% of the available logic gates, however, need to use only 2 to 3% of these anti-fuses (i.e., to be changed to conductive state). Note that if fuses, instead of antifuses, are used, 97% or 98% of fuses need to be disconnected, requiring disconnection of a huge number of fuses.

Actel's FPGAs are of row-based structure, as illustrated in Fig. 44.3.[3–5,10] The actual logic block array of Actel is shown in Fig. 44.8 A logic block (Actel calls it a logic module) for a simple model of Actel's FPGA consists of three multiplexers and one OR gate, as shown in Fig. 44.9, where some of their inputs and outputs can be complemented. This logic module, which has eight inputs and a single output, can realize many different logic gates with some inputs inverted, as follows: four basic logic gates (i.e., NAND, AND, OR, and NOR gates with 2, 3, or 4 inputs); EXCLUSIVE-OR gates, EXCLUSIVE-OR-OR gates, EXCLUSIVE-OR-AND gates, OR-EXCLUSIVE-OR gates, AND-EXCLUSIVE-OR gates, AND-OR gates, and OR-AND gates; and a variety of D latches. Other models of Actel have more complex logic modules that contain latches.

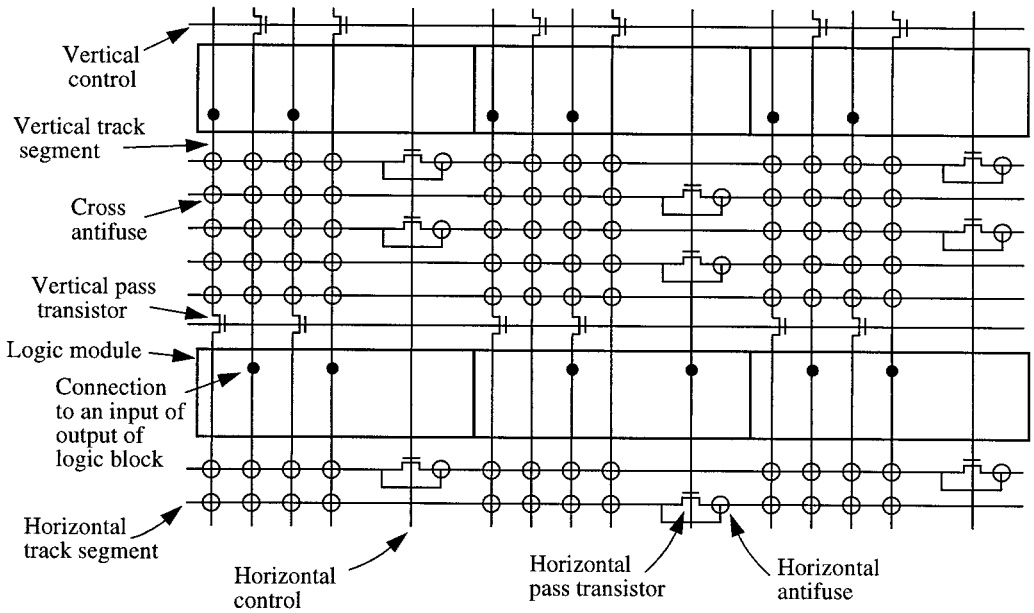


FIGURE 44.8 Logic module array of Actel.

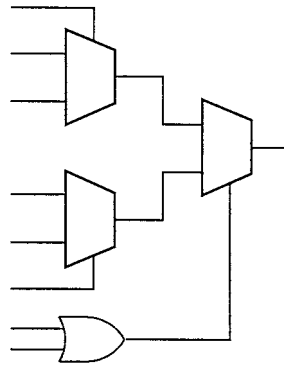


FIGURE 44.9 Logic block of Actel.

FPGAs Based on Non-volatile Memories

Altera has FPGAs that are based on non-volatile memory for connecting lines.[10] This FPGA has bus-based structure illustrated in Fig. 44.10 (i.e., Fig. 44.4). The inputs and outputs of each logic block can be connected to bus lines, as shown in Fig. 44.10. Altera calls these bus lines PIA (Programmable Interconnect Array). Altera claims that delay times on PIAs are more predictable than those on lines that consist of many line segments. A logic block contains of 16 logic macrocells, such as the one shown in Fig. 44.11.

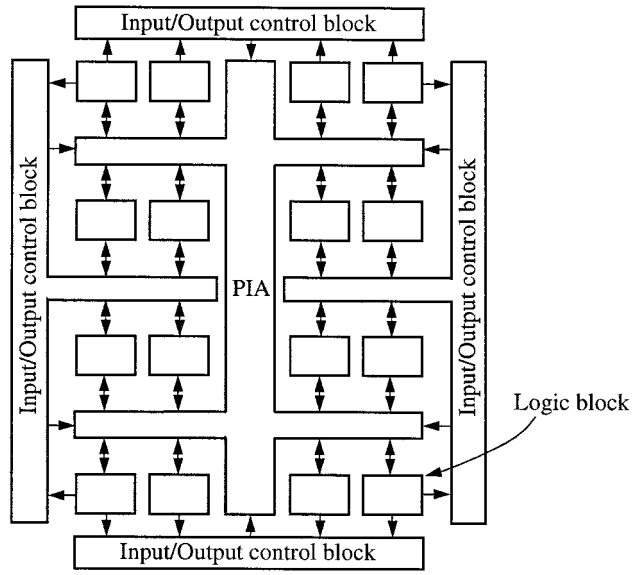


FIGURE 44.10 FPGA of Altera with non-volatile memory.

FPGAs of Advanced Micro Devices and Lattice Semiconductor Corp. are based on a different type of non-volatile memory, using PALs as logic blocks that can be connected with switch matrixes similar to Altera's.

44.4 Features of FPGAs

If we order semiconductor manufacturers to make mask-programmable gate arrays, we have to wait several weeks and pay twenty-thousand to hundreds of thousands of dollars. But with FPGAs, we can

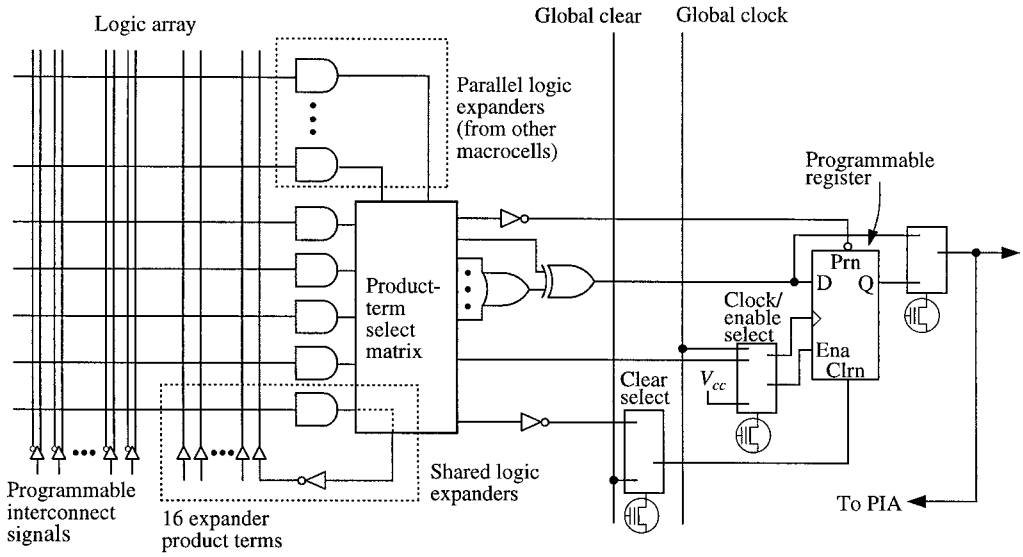


FIGURE 44.11 Macrocell of FPGA of Altera.

program FPGAs in minutes by ourselves and need to pay in the tens of dollars. But a FPGA can pack only about one-tenth the number of logic gates in a mask-programmable gate array because devices for user programmability, such as SRAMs, non-volatile memory, and anti-fuses, take up large areas. Mask-programmable gate arrays are faster by two orders of magnitude and far cheaper for large production volume. Thus, for debugging or verifying logic design that needs to be done quickly, FPGAs are used, and then mask-programmable gate arrays are used for large volume production after completing debugging or verification.

As seen already, FPGAs are classified into two types, depending on the types of devices used for programmability: rewritable FPGAs and non-rewritable FPGAs. Then, they are accordingly used for completely different purposes. Rewritable FPGAs, such as Xilinx's based on SRAMs and Altera's based on non-volatile memory, can be repeatedly rewritten in minutes. Non-rewritable FPGAs cannot be changed once programmed, but still have the advantages of realizing inexpensive logic chips with faster speed.

The area size of different devices for programmability also gives different advantages and disadvantages. A non-volatile memory cell is roughly four to five times larger than anti-fuse; a memory cell of SRAM is two times larger than a non-volatile memory cell. Anti-fuses are much smaller. So, because of smaller parasitic capacitance, FPGAs based on anti-fuses tend to be faster than those based on non-volatile memory or SRAMs.

References

1. Brown, S. D., et al., *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.
2. Chan, P. K. and S. Mourad, *Digital Design Using Field Programmable Gate Arrays*, Prentice-Hall, 1994.
3. El-Ayat, K., et al., "A CMOS electrically configurable gate array," *ISSCC Dig. Tech. Papers*, pp. 76-77, Feb. 1988.
4. El Gamal, A., K. A. El-Ayat, and A. Mohsen. "Programmable interconnect architecture," U.S. patent 5600265, Feb. 4 1997.
5. El Gamal, A., J. Greene, J. Reyneri, E. Rogoyski, K. A. El-Ayat, and A. Mohsen, "An architecture for electrically configurable gate arrays," *IEEE Jour. Solid-State Circuits*, vol. 24, no. 2, pp. 394-398, April 1989.

6. Freeman, R., "User-programmable gate arrays," *IEEE Spectrum*, pp. 32-35, Dec. 1988.
7. Freeman, R., "Configurable electrical circuit having configurable logic elements and configurable interconnects", U. S. Patent 4,870,302, Sep. 26, 1989.
8. Salcic, Z. and A. Smailagic, *Digital Systems Design and Prototyping Using Field Programmable Logic*, Kluwer Academic Publisher, 1997.
9. Smith, M. J. S., *Application-Specific Integrated Circuits*, Addison-Wesley, 1997.
10. Trimberger, S. M., edited, *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers, 1994.

Muroga, S. "Cell-Library Design Approach"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

45

Cell-Library Design Approach

Saburo Muroga
University of Illinois at Urbana-Champaign

45.1 Introduction

Compact layouts for basic logic gates (such as NOR gates) and small networks that are frequently used (such as flip-flops and a full adder) are designed by professional designers with time-consuming efforts and are stored in computer memories (i.e., magnetic disks). Each layout is called a **cell** and a collection of these layouts is called a **cell library**. Once a cell library is ready, any other designer can call up specific cells on the monitor of a computer. By arranging them and adding connections among them by the use of a mouse, the designer can make a layout of the entire logic network. When the layout is complete, photomasks are automatically prepared by computer. Such a design approach is called the **cell-library design approach**.[1–3]

45.2 Polycell Design Approach

When the layout of every cell has the same height, although its width may be different, this approach is called the **polycell design approach** (or **standard-cell design approach).** It is often called the cell-library design approach, but it is also called the polycell or standard-cell design approach in order to avoid confusion with the words "cell" and "library" from those which are used in the gate arrays. The height of all cells are chosen the same, even though area is wasted inside a cell, such that many cells are connected into rows, as exemplified in Fig. 45.1, which shows an example with three rows of cells.

Routing (i.e., connections among cells) is mainly done in space between adjacent rows of cells, as shown in Fig. 45.1. This space for routing is called a **routing channel**. Routing, in addition to design of logic networks, is the major task in the polycell design approach and is greatly facilitated by CAD programs.

Connections from one routing channel to others can be done through narrow passages, which are provided between cells in each row. In some cases, connections are allowed to go through cells horizontally or vertically (without going through the narrow passages provided) in order to reduce wiring areas. In this case, detailed routing rules on where in each cell connections are allowed to go through must be incorporated in the CAD programs.

The preparation of a cell library takes time (probably more than a dozen man-months for a small library), but layout time of chips, using a cell library, is greatly reduced by the polycell design approach (probably by an order of magnitude), compared with full-custom design which requires the most compact layout of gates and connections. This is a great advantage. When new fabrication technology is to be adopted, only cells have to be newly laid out; whereas in the case of full-custom design, the entire chip has to be laid out

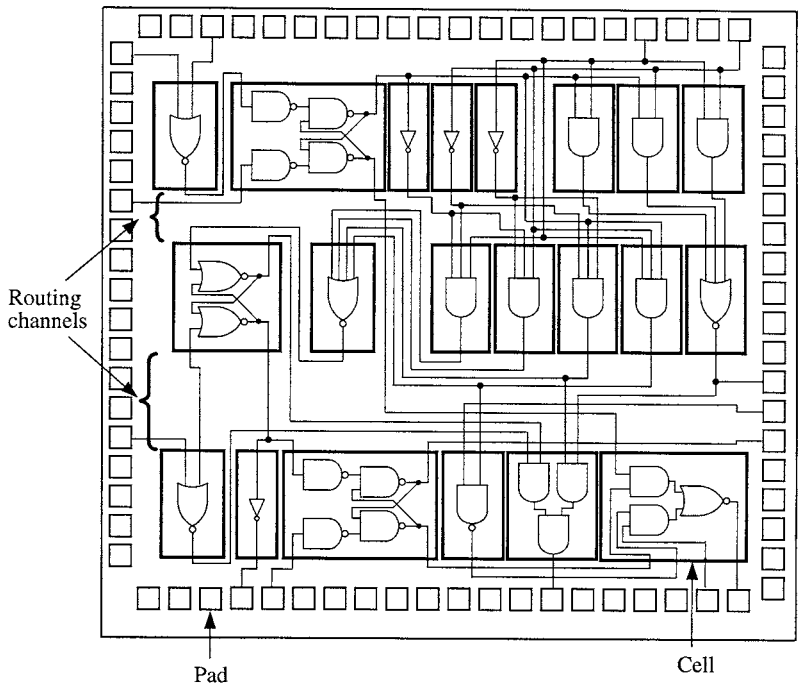


FIGURE 45.1 Polycell layout.

compactly with new technology. Thus, the cell library approach can adapt to a new technology in a shorter time than the full-custom design, but chip size and performance are sacrificed. Compared with the full-custom layout of random-logic gate networks, chip size is many times larger, although it is much smaller than the size realized with a gate array. In order to facilitate routing among cells laid out, all cells are prepared with equal height, keeping their inputs and outputs at the sides that face routing channels. Then, cells for simple functions become very thin and those for complex functions become squat. Thus, the area in each cell is not efficiently utilized. Also, interconnections among cells take up a significant share of the chip area.

Unlike gate arrays where only masks for connections and contact windows are to be custom-made, here all masks must be custom-made, so the polycell design approach needs expensive initial investment, despite its advantage of smaller chip size than gate arrays. However, since the layout is highly automated with interactive CAD programs for placement and routing, the layout of the entire chip is not as time-consuming as those by full-custom design, and consequently the polycell design approach is favored in many cases. In this sense, the polycell design approach is not as purely semi-custom design as the gate array design approach. But the approach is not purely full-custom design either, since the chips are not as compact as those by the full-custom design. Thus, it might be called a pseudo-full-custom design approach, although some people call it a full-custom design approach.

The polycell design approach is cost-effective when the production volume is in hundreds of thousands of dollars, since the initial investment costs $20,000 or more, which is much lower than tens of millions dollars for the full-custom design. The polycell design approaches have been extensively used when the chip compactness attained by the full-custom design approach is not required.

45.3 Hierarchical Design Approach

The cell library design approaches, using cells of different shapes and sizes, can reduce the chip size more than the polycell design approach, because by keeping the same height, a large portion of the area of

each cell is wasted, and by keeping all connections among cells in routing channels, the connection area may not be minimized. Moreover, by using a **hierarchical approach** based on cells of different shapes and sizes — in other words, by treating many cells as a building block in a higher level, and many such building blocks as a building block in a next higher level, and so on — we can further reduce the chip area, as illustrated in Fig. 45.2, because global area minimization can be treated better, even though this is done on the monitor. In other words, cells A, B, C, and D are assembled into a block R (shown in a dot-lined rectangle), as shown in Fig. 45.2. Then, such blocks, R, S, T and U, shown in dot-lined rectangles are assembled into a bigger block W, which is a block in a higher level than blocks R, S, T, and U, as shown in Fig. 45.2. But this is much more time-consuming than the polycell design approach, and the development of efficient CAD programs is harder. It appears to be difficult to make the difference of chip area from full-custom designed chips within about 20%, although the areas of full-custom designed chips vary greatly with designers and, accordingly, comparison is not simple.

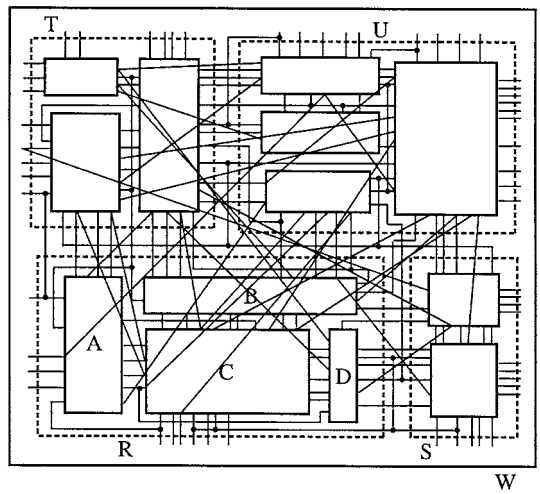


FIGURE 45.2 Hierachical design approach.

References

1. Lauther, U., "Cell based VLSI design system," in *Hardware and Software Concepts in VLSI*, Ed. by G. Rabbat, Van Nostrand Reinhold, pp. 480–494, 1983.
2. Kick, B. et al. "Standard-cell-based design methodology for high-performance support chips," *IBM Jour. Res. Dev.*, pp. 505–514, July/Sept. 1997.
3. Muroga, S., *VLSI System Design*, John Wiley & Sons, 1982.

Muroga, S. "Comparison of Different Design Approaches"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

46

Comparison of Different Design Approaches

Saburo Muroga
University of Illinois at Urbana-Champaign

46.1 Introduction

As discussed so far, there is an almost continuous spectrum of different design approaches from design approaches with off-the-shelf packages, semi-custom design approaches, to the full-custom design approach, depending on the degree of the regularity of transistor arrangement in the layout and the simplification of realization of logic functions (i.e., whether logic functions are realized by a minimal sum or a logic network). Naturally, comparison of these approaches by many different criteria such as performance, design time, and chip size is very complex. Also, different design approaches are often mixed on the same chip, further complicating the comparison. Reliable comparison data are rarely available. But here we will try to give some idea of the advantages and disadvantages of the different design approaches.[1-4]

46.2 Design Approaches with Off-the-Shelf Packages

Here, let us compare different design approaches from the viewpoint of design of logic networks or digital systems. This comparison includes off-the-shelf packages (where a package means a component or an IC chip encased in a container). But among all off-the-shelf packages, let us compare only off-the-shelf discrete component packages and off-the-shelf IC packages (many commonly used logic functions are commercially available as off-the-shelf IC packages), excluding off-the-shelf processors that are fully programmable. We exclude off-the-shelf fully programmable processor packages, such as microprocessor or microcontroller packages, because these off-the-shelf packages are essentially general-purpose computers that are used by writing appropriate software and accordingly do not require design of logic networks.

The major advantage of the off-the-shelf packages is the ease of partial changes of the entire computer or replacement of faulty parts. But there are many disadvantages, such as low reliability (due to connections outside these packages), high cost, and bulkiness. When off-the-shelf packages are assembled on pc boards and further into cabinets, the overall system costs make a substantial difference because of additional costs. In the case of custom-designed IC packages of large integration size (the integration size of an integrated-circuit chip means the number of transistors packed in a chip), all additional costs, such as those for pc boards and fans, become zero or insignificant, but we still have to consider test costs, which are even higher because of more stringent test requirements. Thus, custom-designed IC packages of large integration size are cost-effective for high volume production, which justifies high initial investment.

46.3 Full- and Semi-Custom Design Approaches

An approximate comparison of different design approaches in terms of design time and chip area is given in Fig. 46.1 in logarithmic scale. For each design situation, designers must choose the most appropriate approach, considering tradeoffs between design time (which is closely related to design cost) and chip area (which is related to manufacturing cost and performance). In this comparison, a design approach in a higher position in Fig. 46.1 takes less time to finish the design, but the finished chip is larger and slower in speed than those in lower positions.

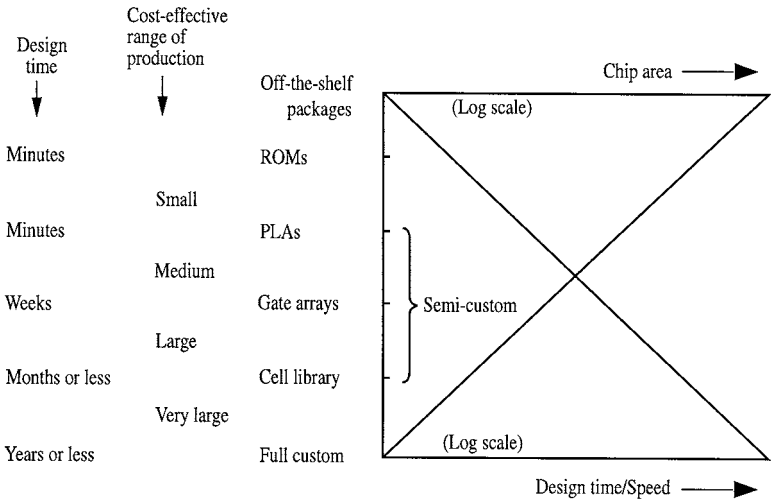


FIGURE 46.1 Chip area versus design time.

Logic functions can be realized in ROMs (read-only-memory) as a truth table. As the number of variables increases, the required memory size exponentially increases. So, its use is limited.

Among all custom (semi- and full-custom) design approaches, PLAs are the quickest to realize logic functions because we can do so simply by deriving minimum sums or minimum products with the use of CAD (computer-aided design) programs, skipping designing logic networks, and also layout and customization of many masks (where manufacturing cost is affected by how many masks need to be customized). But performance, size, and cost are sacrificed. PLAs require only one custom-made mask for connections, while the other custom design approaches require two or more custom-made masks among all of about two dozen required masks. Field-programmable PLAs have larger chip areas than mask-programmable ones, although no custom masks are required.

In using gate arrays, users need to design logic networks but can realize logic networks by the layout of connections among logic gates with CAD programs. Layout and customization of many masks can be skipped. Performance is much higher than PLAs. Gate arrays require two custom-made masks for connections among all of about two dozens masks. Field-programmable gate arrays have a variety. Among them, non-rewritable FPGAs have larger chip areas than mask-programmable ones, although no custom masks are required. Rewritable FPGAs (e.g., those with RAMs) are so different from the other custom design approaches that they cannot be properly compared in Fig. 46.1.

The cell-library design approach requires logic design and layout of connections with CAD, using a cell library which requires a one-time design effort. But the layout of logic gates and connections inside cells and layout of transistor circuits can be skipped, though all masks need to be customized. All of about two dozen masks need to be custom-made, making the cell-library design approach more expensive than gate arrays and PLAs.

The full-custom design approach requires deliberate design of all aspects of a digital system, although semi-custom design approaches along with appropriate CAD programs are used wherever speed is not critical for the speed of the entire system or frequent changes are expected, in the entire system. For example, PLAs are used in the control unit of the full-custom design system. All masks need to be custom-made. The full-custom design approach is the most expensive, taking the longest design time, but the approach yields chips with the highest performance and the lowest cost for high volume production.

A crude estimation of design time is of the order of minutes for ROMs and PLAs, weeks for gate arrays, months for cell-library approach, and years for full-custom design approaches, with the appropriate number of engineers assigned in each case, as shown in Fig. 46.1. A crude estimation of the cost-effective range of production, as shown in Fig. 46.1, is small volume production for the upper range of the cost-effective production of ROMs (i.e., the lower range of the cost-effective production of PLAs), and so on. And the full-custom design approach cannot be more cost-effective than the cell-library approach, unless the production volume is very high.

Rewritable FPGAs have a unique feature that is very different from the other custom design approaches: the ease of changing information in the memories of the FPGAs. With this feature, a designer can send new information to customers over communication lines, instead of sending hardware. So, the customers can change the information on their FPGAs instantly and very inexpensively. Thus, rewritable FPGAs are replacing other custom design approaches in many applications.

An approximate relationship between cost per package and production volume is illustrated in Fig. 46.2, although this may change depending on many factors, such as fabrication technology, logic families, system size, and performance. For each production volume, there is the most economical design approach. But the comparison is difficult and has to be done carefully in each case, because each approach has variations and it makes a difference whether or not libraries of cells or macrocells are prepared from scratch. (Notice that in Fig. 46.2, design approaches are shown in thin-line curves for the sake of simplicity, but actually they should be represented in very broad lines.) The cost per package for the off-the-shelf package design approach is fairly uniform over the entire range, but it increases for low production volumes because the development cost becomes significant as initial investment in the overall package

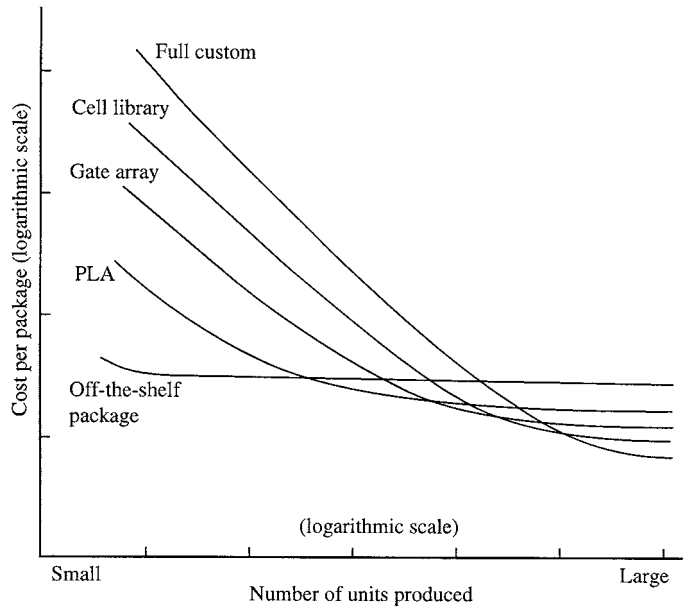


FIGURE 46.2 Package cost versus production volume.

cost. The relationship shown in this figure will change as the integration size of an IC chip increases, because the dependence on CAD will inevitably increase.

46.4 Comparison of All Different Design Approaches

As discussed so far, we have a very wide spectrum of different design approaches, from full-custom design approaches to the design approaches with off-the-shelf packages, as illustrated in Table 46.1. Digital systems can be designed by combining them. Depending upon different criteria imposed by different design motivations, such as speed, power consumption, size, design time, ease of changes, and reliability, designers can use the following approaches:

TABLE 46.1 Comparison of Different Task-Realization Approaches

	Full-Custom	Semi-Custom	Off-the-Shelf IC Package	Off-the-Shelf Microcomputer
Speed	Fastest	Fast	Medium	Slowest
Size	Smallest (chip size)	Small (chip size)	Large (many chips)	Medium (many chips)
Development time	Longest (layout)	Long (layout)	Medium (logic design)	Short (programming)
Flexibility	Lowest	Low	Medium	High
Initial investment	Highest (layout)	High (layout)	Medium (logic design)	Low (programming)
Unit Cost				
High volume	Lowest	Low	Medium	Highest
Low volume	Highest	High	Medium	Lowest
Reliability	Highest	High	Low	Medium

1. Custom-design full- and semi-custom approaches
2. Off-the-shelf discrete components and off-the-shelf IC packages, along with memory packages
3. Off-the-shelf microcomputers along with off-the-shelf IC packages

The full-custom design approaches give us the highest performance and reliability or the smallest chip size, although they are most time-consuming. (Even in the case of microcomputers, the full-custom designed microcomputers have better performance and smaller size than off-the-shelf microcomputers, by being tailored to the users' specific needs.) This is one end of the wide spectrum of different design approaches. At the other end, the off-the-shelf microcomputers give us a design approach where the development time is shortest, by programming rather than by chip design (including logic design), and the design changes are the easiest. The off-the-shelf discrete components and off-the-shelf IC packages give us logic networks tailored to specific needs with less programming than the off-the-shelf microcomputers.

Custom design approaches, in particular the full-custom design approaches, are the most economical for very high production volumes (on the order of a few hundred thousand) but the least economical for low production volumes.

When the production volume is low, the off-the-shelf discrete components and off-the-shelf IC packages give us the most economical approaches for simple tasks, but the off-the-shelf microcomputers are more economical for complex tasks, although performance is usually sacrificed.

References

1. Fey, C. F. and D. E. Paraskevopoulos, "A techno-economic assessment of application-specific integrated circuits: current and future trends," *Proc. IEEE*, pp. 829–841, June 1987.
2. Fey, C. F. and D. E. Paraskevopoulos, "Economic aspects of technology selection: Level of integration, design productivity, and development schedules," in *VLSI Handbook*, Ed. by J. Di Jacomo, McGraw-Hill, pp. 25.3–25.27, 1989.

3. Fey, C. F. and D. E. Paraskevopoulos, "Economic aspects of technology selection: Costs and risks," in *VLSI Handbook*, Ed. by J. Di Jacomo, McGraw-Hill, pp. 26.1–26.21, 1989.
4. Muroga, S., *VLSI System Design*, John Wiley & Sons, 1982.

Kourtev, I.S., Friedman, E.G."System Timing"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

47

System Timing

Ivan S. Kourtev
University of Pittsburgh

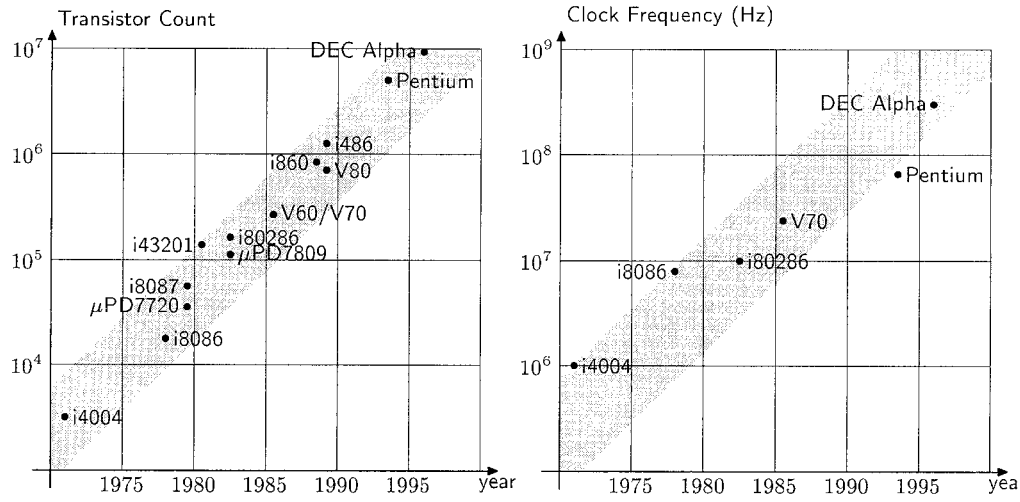
Eby G. Friedman
University of Rochester

47.1 Introduction

The concept of *data* or *information processing* arises in a variety of fields. Understanding the principles behind this concept is fundamental to computer design, communications, manufacturing process control, biomedical engineering, and an increasingly large number of other areas of technology and science. It is impossible to imagine modern life without computers for generating, analyzing, and retrieving large amounts of information, as well as communicating information to end users regardless of their location.

Technologies for designing and building microelectronics-based computational equipment have been steadily advancing ever since the first commercial *discrete integrated circuits* were introduced[1] in the late 1950s.[1] As predicted by *Moore's law* in the 1960s,[2] integrated circuit (IC) density has been doubling approximately every 18 months, and this doubling in size has been accompanied by a similar exponential increase in circuit speed (or, more precisely, clock frequency). These trends of steadily increasing circuit size and clock frequency are illustrated in Fig. 47.1(a) and (b), respectively. As a result of this amazing revolution in semiconductor technology, it is not unusual for modern integrated circuits to contain over ten million switching elements (i.e., transistors) packed into a chip area as large as 500 mm^2.[3-5] This truly exceptional technological capability is due to advances in both design methodologies and physical manufacturing technologies. Research and experience demonstrate that this trend of exponentially increasing integrated circuit computational power will continue into the foreseeable future.

[1] Monolithic integrated circuits (ICs) were introduced in the 1960s.



(a) Evolution of the number of transistors per integrated circuit; and (b) Evolution of clock frequency.

FIGURE 47.1 Moore's law: exponential increase in circuit integration and clock frequency. (Ref. 2. With permission.)

Integrated circuit performance is typically characterized[6] by the *speed of operation*, the available *circuit functionality*, and the *power consumption*, and there are multiple factors which directly affect these performance characteristics. While each of these factors is significant, on the technological side, increased circuit performance has been largely achieved by the following approaches:

- Reduction in feature size (technology scaling); that is, the capability of manufacturing physically smaller and faster device structures
- Increase in chip area, permitting a larger number of circuits and therefore greater on-chip functionality
- Advances in packaging technology, permitting the increasing volume of data traffic between an integrated circuit and its environment as well as the efficient removal of heat created during circuit operation.

The most complex integrated circuits are referred to as VLSI circuits, where the term "VLSI" stands for Very Large-Scale Integration. This term describes the complexity of modern integrated circuits consisting of hundreds of thousands to many millions of active transistor elements. Presently, the leading integrated circuit manufacturers have a technological capability for the mass production of VLSI circuits with feature sizes as small as 0.12 μm.[7] These sub-1/2-micrometer technologies are identified with the term *deep submicrometer* (DSM) since the minimum feature size is well below the one micrometer mark.

As these dramatic advances in fabricating technologies take place, integrated circuit performance is often limited by effects closely related to the very reasons behind these advances, such as small geometry interconnect structures. Circuit performance has become strongly dependent and limited by electrical issues that are particularly significant in deep submicrometer integrated circuits. *Signal delay* and related *waveform effects* are among those phenomena that have a great impact on high-performance integrated circuit design methodologies and the resulting system implementation. In the case of fully synchronous VLSI systems, these effects have the potential to create catastrophic failures due to the limited time available for signal propagation among gates.

Synchronous systems in general are reviewed in Section 47.2, followed by a more detailed description of these systems and the related timing constraints in Section 47.3. The timing properties of the storage elements are discussed in Section 47.4 closing with an appendix containing a glossary of the many terms used throughout this chapter.

47.2 Synchronous VLSI Systems

General Overview

Typically, a digital VLSI system performs a complex computational algorithm, such as a Fast Fourier Transform or a RISC[2] architecture microprocessor. Although modern VLSI systems contain large number of components, these systems normally employ only a limited number of different kinds of *logic elements* or *logic gates*. Each logic element accepts certain input signals and computes an output signal to be used by other logic elements. At the logic level of abstraction, a VLSI system is a *network* of tens of thousands or more logic gates whose terminals are *interconnected* by wires in order to implement the target algorithm.

The switching variables acting as inputs and outputs of a logic gate in a VLSI system are represented by tangible physical qualities,[3] while a number of these devices are interconnected to yield the desired function of each logic gate. The specifiics of the physical characteristics are collectively summarized with the term *technology*, which encompasses such detail as the type and behavior of the devices that can be built, the number and sequence of the manufacturing steps, and the impedance of the different interconnect materials used. Today, several technologies make possible the implementation of high-performance VLSI systems — these are best exemplified by CMOS, bipolar, BiCMOS, and Gallium Arsenide.[2,8] CMOS technology in particular exhibits many desirable performance characteristics, such as low power consumption, high density, ease of design, and reasonable to excellent speed. Due to these excellent performance characteristics, CMOS technology has become the dominant VLSI technology used today.

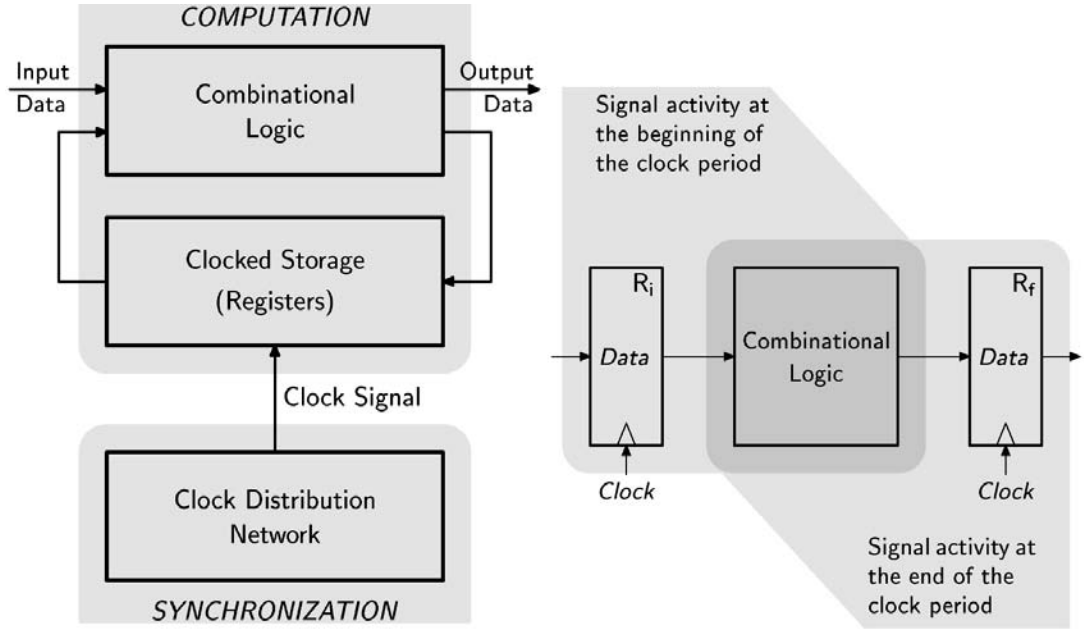
The design of a digital VLSI system may require a great deal of effort in order to consider a broad range of architectural and logic issues; that is, choosing the appropriate gates and interconnections among these gates to achieve the required circuit function. No design is complete, however, without considering the *dynamic* (or transient) characteristics of the signal propagation, or, alternatively, the changing behavior of signals within *time*. Every computation performed by a switching circuit involves multiple signal transitions between logic states and requires a *finite* amount of time to complete. The voltage at every circuit node must reach a specific value for the computation to be completed. Therefore, state-of-the-art integrated circuit design is largely centered around the difficult task of predicting and properly interpreting signal waveform shapes at various points in a circuit.

In a typical VLSI system, millions of signal transitions determine the individual gate delays and the overall speed of the system. Some of these signal transitions can be executed *concurrently,* while others must be executed in a strict *sequential* order.[9] The sequential occurence of the latter operations — or signal transition *events* — must be properly coordinated in time so that logically correct system operation is guaranteed and its results are reliable (in the sense that these results can be repeated). This coordination is known as *synchronization* and is critical to ensuring that any pair of logical operations in a circuit with a precedence relationship proceed in the proper order. In modern digital integrated circuits, synchronization is achieved at all stages of system design and system operation by a variety of techniques, known as a *timing discipline* or *timing scheme*.[8, 10-12] With few exceptions, these circuits are based on a *fully synchronous* timing scheme, specifically developed to cope with the finite speed required by the physical signals to propagate through the system.

An example of a *fully synchronous* system is shown in Fig. 47.2(a). As illustrated in Fig. 47.2(a), there are three recognizable components in this system. The first component — the logic gates, collectively referred to as the *combinational logic* — provides the range of operations that a system executes. The second component — the *clocked storage* elements or simply the *registers* — are elements that store the results of the logical operations. Together, the combinational logic and registers constitute the *computational* portion of the synchronous system and are interconnected in a way that implements the

[2] RISC = Reduced Instruction Set Computer.

[3] Such quantities as the *electical voltages* and *currents* in the electronic devices.



(a) Finite-state machine model of a sychronous system; and (b) A local data path.

FIGURE 47.2 A synchronous system.

required system function. The third component of the synchronous system — known as the *clock distribution network* — is a highly specialized circuit structure which does not perform a computational process, but rather provides an important control capability. The clock generation and distribution network controls the overall synchronization of the circuit by *generating* a time reference and properly *distributes* this time reference to every register.

The normal operation of a system, such as the example shown in Fig. 47.2(a), consists of the iterative execution of computations in the combinational logic, followed by the storage of the processed results in the registers. The actual process of storage is temporally controlled by the clock signal and occurs once the signal transients in the logic gate outputs are completed and the outputs have settled to a valid state. At the beginning of each computational cycle, the inputs of the system, together with the data stored in the registers, initiate a new switching process. As time proceeds, the signals propagate through the logic, generating results at the logic output. By the end of the clock period, these results are stored in the registers and are operated upon during the following clock cycle.

Therefore, the operation of a digital system can be thought of as the sequential execution of a large set of simple computations that occur concurrently in the combinational logic portion of the system. The concept of a *local data path* is a useful abstraction for each of these simple operations and is shown in Fig. 47.2(b). The magnitude of the delay of the combinational logic is bound by the requirement of storing data in the registers within a clock period. The initial register R_i is the storage element at the beginning of the local data path and provides some or all of the input signals for the combinational logic at the beginning of the computational cycle (defined by the beginning of the clock period). The *combinational path* ends with the data successfully latching within the final register R_f, where the results are stored at the end of the computational cycle. Each register acts as a *source* or *sink* for the data, depending upon which phase the system is currently operating in.

Advantages and Drawbacks of Synchronous Systems

The behavior of a fully synchronous system is well-defined and controllable as long as the *time window* provided by the clock period is sufficiently long to allow every signal in the circuit to propagate through

the required logic gates and interconnect wires and successfully latch within the final register. In designing the system and choosing the proper clock period, however, two contradictory requirements must be satisfied. First, the smaller the clock period, the more computational cycles can be performed by the circuit in a given amount of time. Alternatively, the time window defined by the clock period must be sufficiently long so that the slowest signals reach the destination registers before the current clock cycle is concluded and the following clock cycle is initiated.

This way of organizing computation has certain clear advantages that have made a fully synchronous timing scheme the primary choice for digital VLSI systems:

- It is easy to understand and its properties and variations are well-understood.
- It eliminates the nondeterministic behavior of the propagation delay in the combinational logic (due to environmental and process fluctuations and the unknown input signal pattern) so that the system as a whole has a completely deterministic behavior corresponding to the implemented algorithm.
- The circuit design does *not* need to be concerned with glitches in the combinational logic outputs so the only relevant dynamic characteristic of the logic is the *propagation delay*.
- The state of the system is completely defined within the storage elements; this fact greatly simplifies certain aspects of the design, debug, and test phases in developing a large system.

However, the synchronous paradigm also has certain limitations that make the design of synchronous VLSI systems increasingly challenging:

- This synchronous approach has a serious drawback in that it requires the overall circuit to operate as slow as the *slowest* register-to-register path. Thus, the global speed of a fully synchronous system depends upon those paths in the combinational logic with the largest delays; these paths are also known as the *worst-case* or *critical* paths. In a typical VLSI system, the propagation delays in the combinational paths are distributed unevenly so there may be many paths with delays much smaller than the clock period. Although these paths could take advantage of a lower clock period — higher clock frequency — it is the paths with the largest delays that bound the clock period, thereby imposing a limit on the overall system speed. This imbalance in propagation delays is sometimes so dramatic that the system speed is dictated by only a handful of very slow paths.
- The clock signal has to be distributed to tens of thousands of storage registers scattered throughout the system. Therefore, a significant portion of the system area and dissipated power is devoted to the clock distribution network — a circuit structure that does not perform any computational function.
- The reliable operation of the system depends upon the assumptions concerning the values of the propagation delays which, if not satisfied, can lead to catastrophic timing violations and render the system unusable.

47.3 Synchronous Timing and Clock Distribution Networks

Background

As described in Section 47.2, most high-performance digital integrated circuits implement data processing algorithms based on the iterative execution of basic operations. Typically, these algorithms are highly *parallelized* and *pipelined* by inserting clocked registers at specific locations throughout the circuit. The synchronization strategy for these clocked registers in the vast majority of VLSI/ULSI-based digital systems is a fully synchronous approach. It is not uncommon for the computational process in these systems to be spread over hundreds of thousands of functional logic elements and tens of thousands of registers.

For such synchronous digital systems to function properly, the many thousands of switching events require a strict temporal ordering. This strict ordering is enforced by a global synchronization signal known as the *clock signal*. For a fully synchronous system to operate correctly, the clock signal must be delivered to every register at a precise *relative* time. The delivery function is accomplished by a circuit and interconnect structure known as a *clock distribution network*.[13]

Multiple factors affect the propagation delay of the data signals through the combinational logic gates and the interconnect. Since the clock distribution network is composed of logic gates and interconnection wires, the signals in the clock distribution network are also delayed. Moreover, the dependence of the correct operation of a system on the signal delay in the clock distribution network is far greater than on the delay of the logic gates. Recall that by delivering the clock signal to registers at precise times, the clock distribution network essentially quantizes the time of a synchronous system (into clock periods), thereby permitting the simultaneous execution of operations.

The nature of the on-chip clock signal has become a primary factor limiting circuit performance, causing the clock distribution network to become a performance bottleneck for high-speed VLSI systems. The primary source of the load for the clock signals has shifted from the logic gates to the *interconnect*, thereby changing the physical nature of the load from a lumped capacitance (C) to a distributed *resistive-capacitive* (RC) load.[6,7] These interconnect impedances degrade the on-chip signal waveform shapes and increase the path delay. Furthermore, statistical variations in the parameters characterizing the circuit elements along the clock and data signal paths, caused by the imperfect control of the manufacturing process and the environment, introduce ambiguity into the signal timing that cannot be neglected. All of these changes have a profound impact on both the choice of synchronous design methodology and on the overall circuit performance. Among the most important consequences are increased power dissipated by the clock distribution network, as well as the increasingly challenging timing constraints that must be satisfied in order to avoid timing violations.[3-5,13,14] Therefore, the majority of the approaches used to design a clock distribution network attempts to simplify the performance goals by targeting minimal or zero global clock skew,[15-17] which can be achieved by different routing strategies,[18-21] buffered clock tree synthesis, symmetric n-ary trees[3] (most notably H-trees), or a distributed series of buffers connected as a mesh.[13,14]

Definitions and Notation

A synchronous digital system is a network of logic gates and registers whose input and output terminals are interconnected by wires. A sequence of connected logic gates (no registers) is called a *signal path*. Signal paths bounded by registers are called *sequentially-adjacent paths* and are defined next:

Definition 47.1: Sequentially-adjacent pair of registers. For an arbitrary ordered pair of registers $\langle R_i, R_f \rangle$ in a synchronous circuit, one of the following two situations can be observed. Either there exists at least one signal path[4] that connects some output of R_i to some input of R_f or any input of R_f cannot be reached from any output of R_i by propagating through a squence of logic elements only. In the former case — denoted by $R_1 \rightsquigarrow R_2$ — the pair of registers $\langle R_i, R_f \rangle$ is called a sequentially-adjacent pair of registers and switching events at the output of R_i can possibly affect the input of R_f during the same clock period. A sequentially-adjacent pair of registers is also referred to as a local data path.[13]

Examples of local data paths with flip-flops and latches are shown in Figs. 47.14 and 47.17, respectively. The clock signal C_i driving the initial register R_i of the local data path and the clock signal C_f driving the final register R_f are shown in Figs. 47.14 and 47.17, respectively.

A fully synchronous digital circuit is formally defined as follows:

[4] Consecutively connected logic gates.

Definition 47.2: A fully synchronous digital circuit $S = \langle G, R, C \rangle$ is an ordered triple, where:

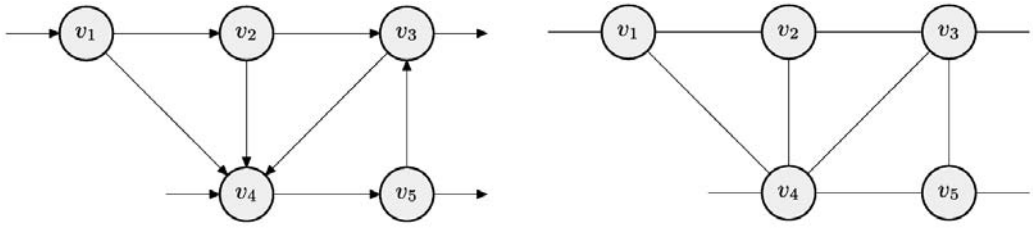
- $G = \{g_1, g_2, \ldots, g_M\}$ is the set of all combinational logic gates,
- $R = \{R_1, R_2, \ldots, R_N\}$ is the set of all registers, and
- $C = \|c_{i \times j}\|_{N \times N}$ is a matrix describing the connectivity of G where for every element $C_{i,j}$ of C

$$c_{i,j} = \begin{cases} 0, & \text{if}(R_i \rightsquigarrow R_j) \\ 1, & \text{if}(R_i \not\rightsquigarrow R_j) \end{cases}$$

Note that in a fully synchronous digital system there are no purely combinational signal *cycles*, that is, it is impossible to reach the input of any logic gate g_k by starting at the same gate and going through a sequence of combinational logic gates only.[13,22]

Graph Model of a Fully Synchronous Digital Circuit

Certain properties of a synchronous digital circuit may be better understood by analyzing a graph model of a circuit. A synchronous digital circuit can be modeled as a *directed graph*[23,24] G with a *vertex set* $V = \{v_1, \ldots, v_N\}$ and an *edge set* $E = \{e_1, \ldots, e_{N_p}\} \subseteq V \times V$. An example of a circuit graph G is illustrated in Fig. 47.3(a). The number of registers in the circuit is $|V| = N$ where the vertex v_k corresponds to the register R_k. The number of local data paths in the circuit is $|E| = N_p = 11$ for the example shown in Fig. 47.3. An edge is directed from v_i to v_j *iff* $R_i \rightsquigarrow R_j$. In the case where multiple paths between a sequentially-adjacent pair of registers $R_i \rightsquigarrow R_j$ exist, only *one* edge connects v_i to v_j. The *underlying* graph G_u of the graph G is a *non-directed* graph that has the same vertex set V, where the directions have been removed from the edges. The underlying graph G_u of the graph G depicted in Fig. 47.3(a) is shown in Fig. 47.3(b). Furthermore, an input or an output of the circuit is indicated in Fig. 47.3 by an edge incident to only one vertex.



(a) The directed graph G. (b) The underlying graph Gu of G in(a).

FIGURE 47.3 Graphs G and its underlying graph G_u of the graph $N = 5$ registers.

The timing constraints of a local data path are derived in Section 47.4 for paths consisting of flip-flops and latches. The concept of clock skew used in these timing constraints is formally defined next.

Definition 47.3: Let $S = \langle G, R, C \rangle$ be a fully synchronous digital circuit as defined in Definition 47.2. For any ordered pair of registers $\langle R_i, R_j \rangle$ driven by the clock signals C_i and C_j, respectively, the clock skew $T_{Skew}(i,j)$ is defined as the difference:

$$T_{Skew}(i,j) = t_{cd}^{i} - t_{cd}^{j} \tag{47.1}$$

where t_{cd}^{i} and t_{cd}^{j} are the clock delays of the clock signals C_i and C_j, respectively.

In Definition 47.3, the clock delays t_{cd}^{i} and t_{cd}^{j} are with respect to some reference point. A commonly used reference point is the source of the clock distribution network on the chip. Note that the clock skew $T_{Skew}(i,j)$ as defined in Definition 47.3 obeys the antisymmetric property,

$$T_{Skew}(i,j) = -T_{Skew}(j,i) \tag{47.2}$$

The clock skew $T_{Skew}(i,j)$ as defined in Definition 47.3 is a component in the timing constraints of a local data path (see inequalities 47.19, 47.24, 47.34, 47.35, and 47.40). Therefore, clock skew is defined and is only of practical use for sequentially-adjacent registers R_i and R_j[5] (i.e., only for local data paths).

The following substitutions are introduced for notational convenience:

Definition 47.4: Let $S = \langle G, R, C \rangle$ be a fully synchronous digital circuit where the registers $\mathsf{R}_i, \mathsf{R}_f \in R$ and $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$. The long path delay $\hat{D}^{i,f}_{PM}$ of the local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ is defined as

$$\hat{D}^{i,f}_{PM} = \begin{cases} (D^{Fi}_{CQM} + D^{i,f}_{PM} + \delta^{Ff}_{S} + 2\Delta^{F}_{L}), & \text{if } \mathsf{R}_i, \mathsf{R}_f \text{ are flip flops} \\ (D^{Li}_{CQM} + D^{i,f}_{PM} + \delta^{Lf}_{S} + \Delta^{L}_{L} + \Delta^{L}_{T}), & \text{if } \mathsf{R}_i, \mathsf{R}_f \text{ are latches} \end{cases} \tag{47.3}$$

Similarly, the short delay $\hat{D}^{i,f}_{Pm}$ of the local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ is defined as

$$\hat{D}^{i,f}_{Pm} = \begin{cases} (D^{i,f}_{Pm} + D^{Fi}_{CQ} - \delta^{Ff}_{H} - 2\Delta^{F}_{L}), & \text{if } \mathsf{R}_i, \mathsf{R}_f \text{ are flip flops} \\ (D^{Li}_{CQm} + D^{i,f}_{Pm} - \delta^{Lf}_{H} - \Delta^{L}_{L} - \Delta^{L}_{T}), & \text{if } \mathsf{R}_i, \mathsf{R}_f \text{ are latches} \end{cases} \tag{47.4}$$

For example, using the notations described in Definition 47.4, the timing constraints of a local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ with flip-flops (Eqs. 47.19 and 47.24) become

$$T_{Skew}(i,f) \le T_{CP} - \hat{D}^{i,f}_{PM} \tag{47.5}$$

$$-\hat{D}^{i,f}_{Pm} \le T_{Skew}(i,f) \tag{47.6}$$

For a local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ consisting of the flip-flows R_i and R_f, the setup and hold time violations are avoided if Eqs. 47.5 and 47.6, respectively, are satisfied.

The clock skew $T_{Skew}(i,f)$ for a local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ can be either *positive* or *negative*, as illustrated in Figs. 47.15 and 47.16, respectively. Negative clock skew may be used to effectively speed up a local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ by allowing an additional $T_{Skew}(i,f)$ amount of time for the signal to propagate from R_i to R_f. However excessive negative skew may create a hold time violation, thereby creating a lower bound on $T_{Skew}(i,f)$ as described by Eq. 47.6. A hold time violation is a *clock hazard* or a *race condition*, also known as *double clocking*.[13, 25] Similarly, positive clock skew effectively decreases the clock period T_{CP} by $T_{Skew}(i,f)$, thereby limiting the maximum clock frequency.[6] In this case, a clocking hazard known as *zero clocking* may be created.[13,25]

Clock Scheduling

Examining the constraints of Eqs. 47.5 and 47.6 reveals a procedure for preventing clock hazards. Assuming Eq. 47.5 is not satisfied, a suitably large value of T_{CP} can be chosen to satisfy constraint Eq. 47.5 and prevent zero clocking. Also note that, unlike Eq. 47.5, Eq. 47.6 is independent of T_{CP}. Therefore, T_{CP} cannot be varied to correct a double clocking hazard, but rather a redesign of the clock distribution network may be required.[17]

Both double and zero clocking hazards can be eliminated if two simple choices characterizing a fully synchronous digital circuit are made. Specifically, if equal values are chosen for all clock delays, then the clock skew $T_{Skew}(i,f) = 0$ for each local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$,

[5] Note that technically, however, $T_{Skew}(i,j)$ can be calculated for any ordered pair of registers $\langle \mathsf{R}_i, \mathsf{R}_j \rangle$.

[6] Positive clock skew may also be thought of as increasing the path delay. In either case, positive clock skew $T_{Skew} > 0$ makes it more difficult to satisfy Eq. 47.5.

$$\forall \langle R_i, R_f \rangle : t_{cd}^{i} = t_{cd}^{f} \Rightarrow T_{Skew}(i, f) = 0 \tag{47.7}$$

Therefore, Eqs. 47.5 and 47.6 become

$$T_{Skew}(i, f) = t_{cd}^{i} - t_{cd}^{f} = 0 \leq T_{CP} - \hat{D}_{PM}^{i,f} \tag{47.8}$$

$$-\hat{D}_{Pm}^{i,f} \leq 0 = T_{Skew}(i, f) = t_{cd}^{i} - t_{cd}^{f} \tag{47.9}$$

Note that Eq. 47.8 can be satisfied for each local data path $R_i \rightsquigarrow R_f$ in a circuit if a sufficiently large value — larger than the greatest value $\hat{D}_{PM}^{i,f}$ in a circuit — is chosen for T_{CP}. Furthermore, Eq. 47.9 can be satisfield across an entire circuit if it can be ensured that $\hat{D}_{Pm}^{i,f} \geq 0$ for each local data path $R_i \rightsquigarrow R_f$ in the circuit. The timing constraint Eqs. 47.8 and 47.9 can be satisfield since choosing a sufficiently large clock period T_{CP} is always possible and $\hat{D}_{Pm}^{i,f}$ is positive for a properly designed local data path $R_i \rightsquigarrow R_f$. The application of this zero clock skew methodology (Eqs. 47.7, 47.8, and 47.9) has been central to the design of fully synchronous digital circuits for decades.[13, 26] By requiring the clock signal to arrive at each register R_j with approximately the same delay t_{cd}^{j},[7] these design methods have become known as *zero* clock skew methods.

As shown by previous research,[13,15-17,27-29] both double and zero clocking hazards may be removed from a synchronous digital circuit even when the clock skew is *non-zero;* that is, $T_{Skew}(i, f) \neq 0$ for some (or *all*) local data paths $R_i \rightsquigarrow R_f$. As long as Eqs. 47.5 and 47.6 are satisfied, a synchronous digital system can operate reliably with *non-zero* clock skews, permitting the system to operate at higher clock frequencies while removing all race conditions.

The vector column of clock delays $\mathbf{T_{CD}} = [\, t_{cd}^{1}, t_{cd}^{2}, \ldots]^T$ is called a *clock schedule.*[13,25] If $\mathbf{T_{CD}}$ is chosen such that Eqs. 47.5 and 47.6 are satisfied for every local data path $R_i \rightsquigarrow R_f$, $\mathbf{T_{CD}}$ is called a *consistent* clock schedule. A clock schedule that satisfies Eq. 47.7 is called a *trivial* clock schedule. Note that a trivial clock schedule $\mathbf{T_{CD}}$ implies global *zero* clock skew since for any i and f, $t_{cd}^{i} = t_{cd}^{f}$, thus, $T_{Skew}(i, f) = 0$.

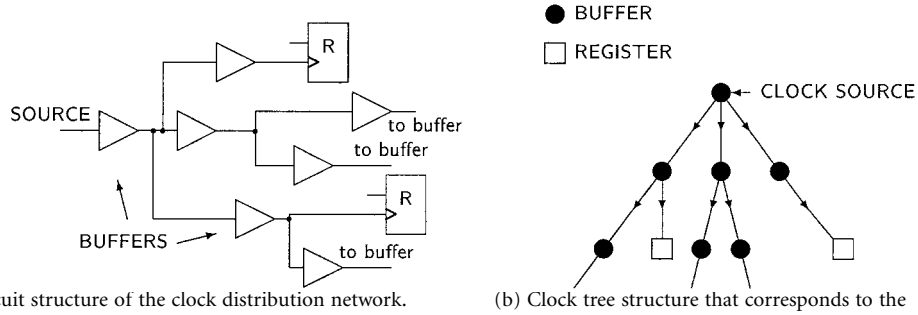
Fishburn[25] first suggested an algorithm for computing a consistent clock schedule that is *non-trivial.* Furthermore, Fishburn showed[25] that by exploiting negative and positive clock skew within the local data paths $R_i \rightsquigarrow R_f$, a circuit can operate with a clock period T_{CP} less than the clock period achievable by a trivial (or zero skew) clock schedule that satisfies the conditions specified by Eqs. 47.5 and 47.6. In fact, Fishburn[25] determined an *optimal* clock schedule by applying linear programming techniques to solve for $\mathbf{T_{CD}}$ so as to satisfy Eqs. 47.5 and 47.6 while minimizing the objective function $F_{\text{objective}} = T_{CP}$.

The process of determining a consistent clock schedule $\mathbf{T_{CD}}$ can be considered as the mathematical problem of minimizing the clock period T_{CP} under the constraints Eqs. 47.5 and 47.6. However, there are important practical issues to consider before a clock schedule can be properly implemented. A clock distribution network must be synthesized such that the clock signal is delivered to each register with the proper delay so as to satisfy the clock skew schedule $\mathbf{T_{CD}}$. Furthermore, this clock distribution network must be constructed so as to minimize the deleterious effects of *interconnect impedances* and *process parameter variations* on the implemented clock schedule. Synthesizing the clock distribution network typically consists of determining a *topology* for the network, together with the circuit design and physical layout of the *buffers* and *interconnect* within the clock distribution network.[13]

Structure of the Clock Distribution Network

The clock distribution network is typically organized as a rooted tree structure,[13,15,23] as illustrated in Fig. 47.4, and is often called a *clock tree.*[13] A circuit schematic of a clock distribution network is shown in Fig. 47.4(a). An abstract graphical representation of the tree structure depicted in Fig. 47.4(a) is shown in Fig. 47.4(b). The unique source of the clock signal is at the root of the tree. This signal is distributed

[7] Equivalently, it is required that the clock signal arrive at each register at approximately the same time.



(a) Circuit structure of the clock distribution network.

(b) Clock tree structure that corresponds to the circuit shown in (a).

FIGURE 47.4 Tree structure of a clock distribution network.

from the source to every register in the circuit through a sequence of buffers and interconnects. Typically, a buffer in the network drives a combination of other buffers and registers in the VLSI circuit. An interconnection network of wires connects the output of the driving buffer to the inputs of these driven buffers and registers. An *internal node* of the tree corresponds to a buffer, and a *leaf node* of the tree corresponds to a register. There are N leaves[8] in the clock tree labeled F_1 through F_N, where leaf F_j corresponds to register R_j. A clock tree topology that implements a given clock schedule $\mathbf{T_{CD}}$ must enforce a clock skew $T_{Skew}(i, f)$ for each local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ of the circuit in order to ensure that both Eqs. 47.5 and 47.6 are satisfied. This topology, however, can be affected by three important issues relating to the operation of a fully synchronous digital system.

Linear Dependency of the Clock Skews

An important corollary related to the *conservation property*[13] of clock skew is that there is a *linear dependency* among the clock skews of a global data path that form a cycle in the underlying graph of the circuit. Specifically, if $v_0, e_1, v_1 \neq v_0, \ldots, v_{k-1}, e_k, v_k \equiv v_0$ is a cycle in the underlying graph of the circuit, then

$$\begin{aligned} 0 &= [t_{cd}^0 - t_{cd}^1] + [t_{cd}^1 - t_{cd}^2] + \ldots \\ &= \sum_{i=0}^{k-1} T_{Skew}(i, i+1) \end{aligned} \tag{47.10}$$

The property described by 47.10 is illustrated in Fig. 47.3 for the undirected cycle v_1, v_4, v_3, v_2, v_1. Note that

$$\begin{aligned} 0 &= (t_{cd}^1 - t_{cd}^4) + (t_{cd}^4 - t_{cd}^3) + (t_{cd}^3 - t_{cd}^2) + (t_{cd}^2 - t_{cd}^1) \\ &= T_{Skew}(1, 4) + T_{Skew}(4, 3) + T_{Skew}(3, 2) + T_{Skew}(2, 1) \end{aligned} \tag{47.11}$$

The importance of this property is that Eq. 47.10 describes the inherent correlation among certain clock skews within a circuit. Therefore, these correlated clock skews *cannot* be optimized independently of each other. Returning to Fig. 47.3, note that it is not necessary that a directed cycle exists in the directed graph G of a circuit for Eq. 47.10 to hold. For example, v_2, v_3, v_4 is not a cycle in the directed circuit graph G in Fig. 47.3(a) but v_2, v_3, v_4 is a cycle in the undirected circuit graph G_u in Fig. 47.3(b). In addition, $T_{Skew}(2, 3) + T_{Skew}(3, 4) + T_{Skew}(4, 2) = 0$; that is, the skews $T_{Skew}(2, 3)$, $T_{Skew}(3, 4)$, and $T_{Skew}(4, 2)$ are linearly dependent. A maximum of $(|V| - 1) = (N - 1)$ clock skews can be chosen independently of each other in a circuit, which is easily proven by considering a spanning tree of the underlying circuit graph G_u.[23,24] Any spanning tree of G_u will contain $(N - 1)$ edges — each edge corresponding to a local

[8] The number of registers N in the circuit.

data path — and the addition of any other edge of G_u will form a cycle such that Eq. 47.10 holds for this cycle. Note, for example, that for the circuit modeled by the graph shown in Fig. 47.3, *four* independent clock skews can be chosen such that the remaining three clock skews can be expressed in terms of the independent clock skews.

Permissible Ranges

Previous research[17,29] has indicated that tight control over the clock skews rather than the clock delays is necessary for the circuit to operate reliably. The relationships in Eqs. 47.5 and 47.6 are used in Ref. 29 to determine a *permissible range* of the allowed clock skew for each local data path. The concept of a permissible range for the clock skew $T_{Skew}(i, f)$ of a local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ is illustrated in Fig. 47.5. When $T_{Skew}(i, f) \in [-\hat{D}^{i,f}_{Pm}, T_{CP} - \hat{D}^{i,f}_{PM}]$ — as shown in Fig. 47.5 — Eqs. 47.5 and 47.6 are satisfied. The clock skew $T_{Skew}(i, f)$ is *not permitted* to be in either the interval $(-\infty, -\hat{D}^{i,f}_{Pm})$ because a race condition will be created or the interval $(T_{CP} - \hat{D}^{i,f}_{PM}, +\infty)$ because the minimum clock period will be limited.

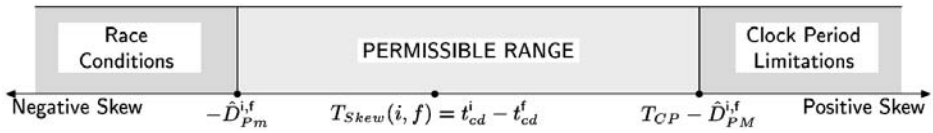


FIGURE 47.5 The permissible range of the clock skew of a local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$. A timing violation exists if $T_{Skew}(i, f) \notin [-\hat{D}^{i,f}_{Pm}, T_{CP} - \hat{D}^{i,f}_{PM}]$.

Also, note that the reliability of the circuit is related to the probability of a timing violation occurring for any local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$. Therefore, the reliability of any local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ of the circuit (and therefore of the entire circuit) is increased in two ways:

1. By choosing the clock skew $T_{Skew}(i, f)$ for a local data path as far as possible from the borders of the interval $[-\hat{D}^{i,f}_{Pm}, T_{CP} - \hat{D}^{i,f}_{PM}]$, that is, by (ideally) positioning the clock skew $T_{Skew}(i, f)$ in the middle of the permissible range, that is, $T_{Skew}(i, f) = 1/2\,[T_{CP} - (\hat{D}^{i,f}_{PM} + \hat{D}^{i,f}_{Pm})]$,
2. By increasing the width $T_{CP} - (\hat{D}^{i,f}_{PM} - \hat{D}^{i,f}_{Pm})$ of the permissible range of the local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$

Due to the linear dependence of the clock skews shown previously, however, it is *not* possible to build a typical circuit such that for each local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$, the clock skew $T_{Skew}(i, f)$ is in the middle of the permissible range.

Differential Character of the Clock Tree

In a given circuit, the clock signal delay t^j_{cd} from the clock source to the register R_j is equal to the sum of the propagation delays of the buffers on the unique path that exists between the root of the clock tree and the leaf F_j corresponding to the j-th register. Furthermore, if $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ is a sequentially-adjacent pair of registers, there is a portion of the two paths — denoted P^*_{if} — between the root of the clock tree and R_i and R_f, respectively, that is common to both paths. This concept is illustrated in Fig. 47.6. A portion of a clock tree is shown in Fig. 47.6 where each of the vertices 1 through 10 corresponds to a buffer in the clock tree. The vertices 4, 5, and 9 are leaves of the tree and correspond to the registers R_4, R_5, and R_9, respectively.[9] The local data paths $\mathsf{R}_4 \rightsquigarrow \mathsf{R}_5$ and $\mathsf{R}_5 \rightsquigarrow \mathsf{R}_9$ are indicated with arrows in Fig. 47.6, while the paths of the clock signals to each of the registers R_4, R_5, and R_9 are shown in Fig. 47.6 lightly shaded. The portion of the clock signal paths common to both registers of a local data path is shaded darker in Fig. 47.6; note the segments $1 \rightarrow 2 \rightarrow 3$ for $\mathsf{R}_4 \rightsquigarrow \mathsf{R}_5$ and $1 \rightarrow 2$ for $\mathsf{R}_5 \rightsquigarrow \mathsf{R}_9$.

Similarly, there is a portion of the clock signal path to any of the registers R_i and R_f in a sequentially-adjacent pair of registers $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$, denoted by P^i_{if} and P^f_{if}, respectively, that is unique to this register.

[9] Note that *not* all of the vertices correspond to registers.

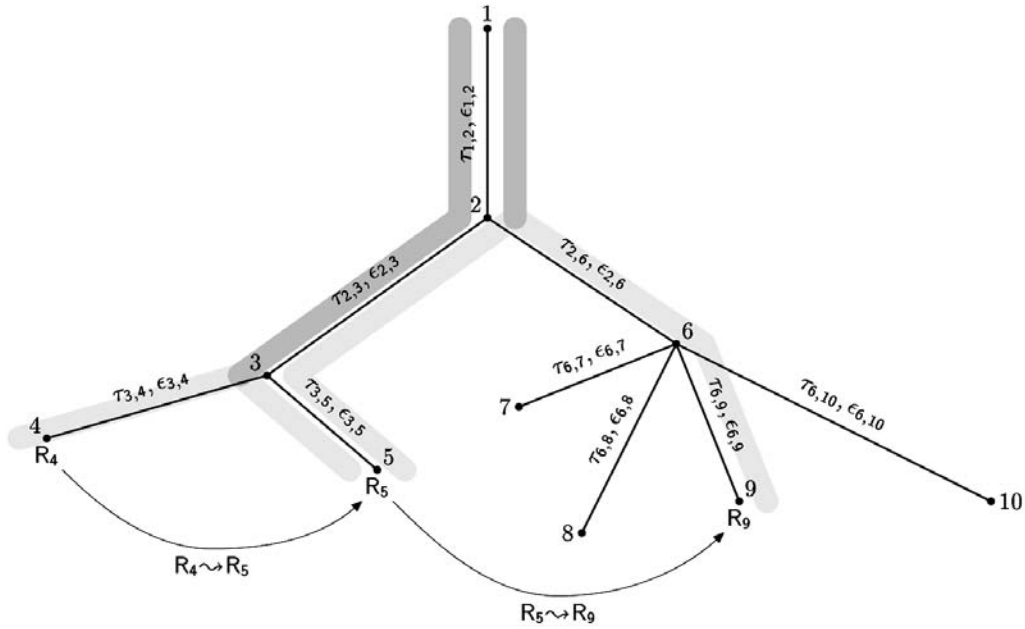


FIGURE 47.6 Illustration of the differential nature of the clock tree.

Returning to Fig. 47.6, the segments $3 \rightarrow 4$ and $3 \rightarrow 5$ are unique to the clock signal paths to the registers R_4 and R_5, while the segments $2 \rightarrow 3 \rightarrow 5$ and $2 \rightarrow 6 \rightarrow 9$ are unique to the clock signal paths to the registers R_5, and R_9, respectively.

Note that the clock skew $T_{Skew}(i, f)$ between the sequentially adjusted pair of registers $R_i \rightsquigarrow R_f$ is equal to the difference between the accumulated buffer propagation delays between P^i_{if} and P^f_{if}, that is, $T_{Skew}(i, f)$ = Delay (P^i_{if}) − Delay (P^f_{if}). Therefore, any variations of circuit parameters over P^*_{if} will not affect the value of the clock skew $T_{Skew}(i, f)$. For the example shown in Fig. 47.6, $T_{Skew}(4,5)$ = Delay $(P^4_{4,5})$ − Delay $(P^5_{4,5})$ and $T_{Skew}(5,9)$ = Delay $(P^5_{5,9})$ − Delay $(P^9_{5,9})$.

The differential feature of the clock tree suggests an approach for minimizing the effects of process parameter variations on the correct operation of the circuit. To illustrate this approach, each branch $p \rightarrow q$ of the clock tree shown in Fig. 47.6 is labeled with two numbers: $\tau_{p,q} > 0$ is the intended delay of the branch and $\varepsilon_{p,q} \geq 3\ 0$ is the maximum error (deviation) of this delay.[10] In other words, the actual delay of the branch $p \rightarrow q$ is in the interval $[\tau_{p,q} - \varepsilon_{p,q}, \tau_{p,q} + \varepsilon_{p,q}]$. With this notation, the *target* clock skew values for the local data paths $R_4 \rightsquigarrow R_5$ and $R_5 \rightsquigarrow R_9$ are shown in the middle column in Table 47.1. The bounds of the actual clock skew values for the local data paths $R_4 \rightsquigarrow R_5$ and $R_5 \rightsquigarrow R_9$ (considering the ε variations) are shown in the right-most column in Table 47.1.

TABLE 47.1 Target and Actual Values of the Clock Skews for the Local Data Paths $R_4 \rightsquigarrow R_5$ and $R_5 \rightsquigarrow R_9$ Shown in Fig. 47.6

	Target Skew	Actual Skew Bounds
$T_{Skew}(4, 5)$	$\tau_{3,4} - \tau_{3,5}$	$\tau_{3,4} - \tau_{3,5} \pm (\varepsilon_{3,4} + \varepsilon_{3,4})$
$T_{Skew}(5, 9)$	$\tau_{2,3} + \tau_{3,5} - \tau_{2,6} - \tau_{6,9}$	$\tau_{2,3} + \tau_{3,5} - \tau_{2,6} - \tau_{6,9} \pm (\varepsilon_{2,3} + \varepsilon_{3,5} + \varepsilon_{2,6} + \varepsilon_{6,9})$

[10] The deviation ε is due to parameter variations during circuit manufacturing as well as to environmnetal conditions during operation of the circuit.

As the results in Table 47.1 demonstrate, it is advantageous to maximize P^*_{if} for any local data path $R_i \rightsquigarrow R_f$ with a relatively narrow permissible range, such that the parameter variations on P^*_{if} do not affect $T_{Skew}(i, f)$ Similarly, when the permissible range $[-\hat{D}^{i,f}_{Pm}, T_{CP} - \hat{D}^{i,f}_{PM}]$ is wider, P^*_{if} may be permitted to be only a small franction of the total path from the root to R_i and R_f, respectively. Future research work will explore this approach of synthesizing a clock tree based on choosing a tree structure which restricts the possible variations of those local data paths with narrow permissible ranges, and tolerates larger delay variations for those local data paths with wider permissible ranges.

47.4 Timing Properties of Synchronous Storage Elements

Common Storage Elements

The general structure and principles of operation of a fully synchronous digital VLSI system were described in Section 47.2. In this section, the timing constraints due to the combinational logic and the storage elements within a synchronous system are reviewed. The clock distribution network provides the time reference for the storage elements — or registers — thereby enforcing the required logical order of operations. This time reference consists of one or more *clock signals* that are delivered to each and every register within the integrated circuit. These clock signals control the order of computational events by controlling the exact times the register data inputs are sampled.

The data signals are inevitably delayed as these signals propagate through the logic gates and along interconnections within the local data paths. These propagation delays can be evaluated within a certain accuracy and used to derive timing relationships among signals in a circuit. In this section, the properties of commonly used types of registers and their local timing relationships for different types of local data paths are described. After discussing registers in general in the next subsection, the properties of level-sensitive registers (latches) and the significant timing parameters of these registers are reviewed. Edge-sensitive registers (flip-flops) and their timing parameters are also analyzed. Properties and definitions related to the clock distribution network are reviewed, and finally, the mathematical foundation for analyzing timing violations in both flip-flops and latches is discussed.

Storage Elements

The storage elements (registers) encountered throughout VLSI systems vary widely in their function and temporal relationships. Independent of these differences, however, all storage elements share a common feature — the existence of two groups of signals with largely different purposes. A generalized view of a register is depicted in Fig. 47.7. The I/O signals of a register can be divided into two groups as shown in Fig. 47.7. One group of signals — called the *data signals* — consists of input and output signals of

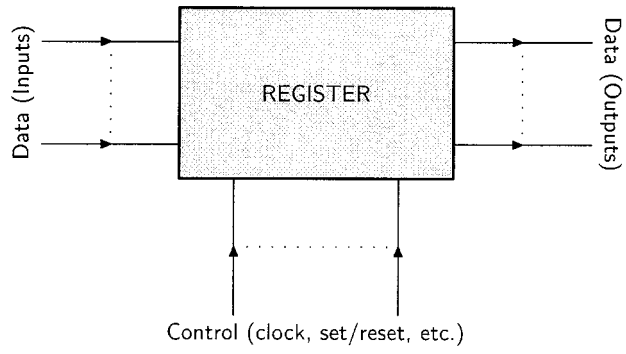


FIGURE 47.7 A general view of a register.

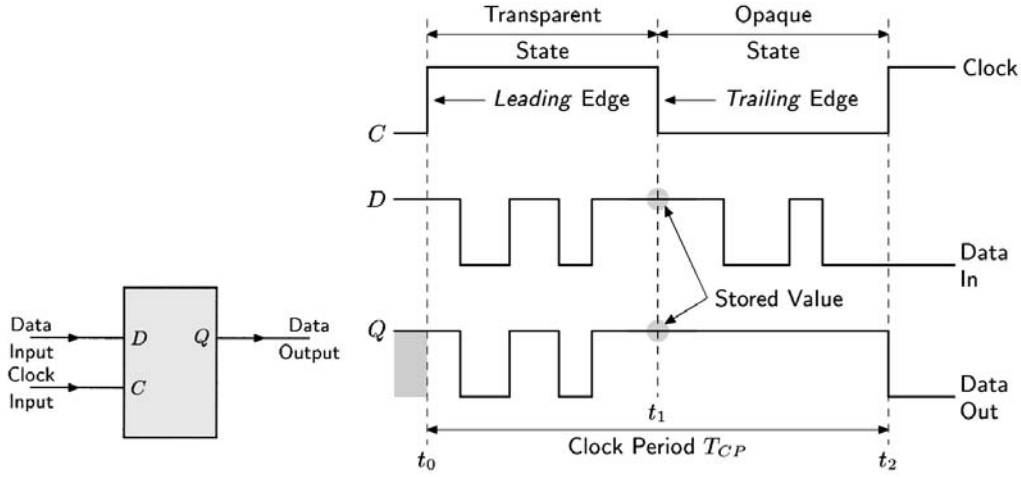
the storage element. These input and output signals are connected to the *data* signal terminals of other storage elements as well as to the terminals of ordinary logic gates. Another group of signals — identified by the name *control* signals — are those signals that control the storage of the data signals in the registers but do not participate in the logical computation process.

Certain control signals enable the storage of a data signal in a register independently of the values of any data signals. These control signals are typically used to initialize the data in a register to a specific well-known value. Other control signals — such as a *clock* signal — control the process of storing a data signal within a register. In a synchronous circuit, each register has at least one clock (or control) signal input.

The two major groups of storage elements (registers) are considered in the following sections based on the type of relationship that exists among the data and clock signals of these elements. In *latches*, it is the specific value or level of a control signal[11] that determines the data storage process. Therefore, latches are also called *level-sensitive registers*. In contrast to latches, a data signal is stored in *flip-flops* as controlled by an *edge* of a control signal. For that reason, flip-flops are also called *edge-triggered registers*. The timing properties of latches and flip-flops are described in detail in the following two sections.

Latches

A *latch* is a register whose behavior depends upon the value or level of the clock signal.[8,30-36] Therefore, a latch is often referred to as a *transparent* latch, a *level-sensitive* register, or a *polarity hold* latch. A simple type of latch with a clock signal C and an input signal D is depicted in Fig. 47.8(a)—the output of the latch is typically labeled Q. This type of latch is also known as a D latch and its operation is illustrated in Fig. 47.8(b).



(a) A level-sensitive register or latch. (b) Idealized operation of the latch shown in (a).

FIGURE 47.8 Schematic representation and principle of operation of a level-sensitive register (latch).

The register illustrated in Fig. 47.8 is a *positive-polarity*[12] latch since it is transparent during that portion of the clock period for which C is high. The operation of this positive latch is summarized in Table 47.2.

As described in Table 47.2 and illustrated in Fig. 47.8(b), the output signal of the latch follows the data input signal while the clock signal remains high, that is, $C = 1 \Rightarrow Q = D$. Therefore, the latch is said

[11] This signal is most frequently the clock signal.

[12] Or simply a *positive* latch.

TABLE 47.2 Operation of the Positive-Polarity D Latch

Clock	Output	State
High	Passes input	Transparent
Low	Maintains output	Opaque

to be in a *transparent* state during the interval $t_0 < t < t_1$ shown in Fig. 47.8(b). When the clock signal C changes from 1 to 0, the current value of D is stored in the register and the output Q remains fixed to that value regardless of whether the data input D changes. The latch does *not* pass the input data signal to the output, but rather holds onto the last value of the data signal when the clock signal made the high-to-low transition. By analogy with the term *transparent* introduced above, this state of the latch is called *opaque* and corresponds to the interval $t_1 < t < t_2$ shown in Fig. 47.8(b) where the input data signal is isolated from the output port. As shown in Fig. 47.8(b), the clock period is $T_{CP} = t_2 - t_0$.

The edge of the clock signal that causes the latch to switch to its transparent state is identified as the *leading edge* of the clock pulse. In the case of the positive latch shown in Fig. 47.8(a), the leading edge of the clock signal occurs at time t_0. The opposite direction edge of the clock signal is identified as the *trailing edge* — the falling edge at time t_1 shown in Fig. 47.8(b). Note that for a negative latch, the leading edge is a high-to-low transition and the trailing edge is a low-to-high transition.

Parameters of Latches

Registers such as the D latch illustrated in Fig. 47.8 and the flip-flops described later are built of discrete transistors. The exact relationships among signals on the terminals of a register can be presented and evaluated in analytical form.[37–39] In this section, however, registers are considered at a higher level of abstraction in order to hide the details of the specific electrical implementation. The latch parameters are briefly introduced next.

Note: The remaining portion of this section uses an extensive notation for various parameters of signals and storage elements. A glossary of terms used throughout this chapter is listed in the appendix.

Minimum Width of the Clock Pulse

The *minimum width of the clock pulse* C^L_{Wm} is the *minimum* permissible width of this portion of the clock signal during which the latch is transparent. In other words, C^L_{Wm} is the length of the time interval between the leading and the trailing edge of the clock signal such that the latch will operate properly. Increasing the value of C^L_{Wm} any further will *not* affect the values of D^L_{DQ}, δ^L_S, and δ^L_H (defined later). The minimum width of the clock pulse, $C^L_{Wm} = t_6 - t_1$, is illustrated in Fig. 47.9. The clock period is $T_{CP} = t_8 - t_1$.

Latch Clock-to-Output Delay

The *clock-to-output delay* D^L_{CQ} (typically called the clock-to-Q delay) is the propagation delay of the latch from the *clock* signal terminal to the output terminal. The value of $D^L_{CQ} = t_2 - t_1$ is depicted in Fig. 47.9 and is defined assuming that the data input signal has settled to a stable value sufficiently early, that is, setting the data input signal earlier with respect to the leading clock edge will not affect the value of D^L_{CQ}.

Latch Data-to-Output Delay

The *data-to-output delay* D^L_{DQ} (typically called the data-to-Q delay) is the propagation delay of the latch from the data signal terminal to the output terminal. The value of D^L_{DQ} is defined assuming that the clock signal has set the latch to its transparent state sufficiently early, that is, making the leading edge of the clock signal occur earlier will not change the value of D^L_{DQ}. The data-to-output delay $D^L_{DQ} = t_4 - t_3$ is illustrated in Fig. 47.9.

Latch Setup Time

The *latch setup time* $\delta^L_S = t_6 - t_5$, shown in Fig. 47.9, is the *minimum* time between a change in the data signal and the trailing edge of the clock signal such that the new value of D would propagate to the output Q of the latch and be stored within the latch during its opaque state.

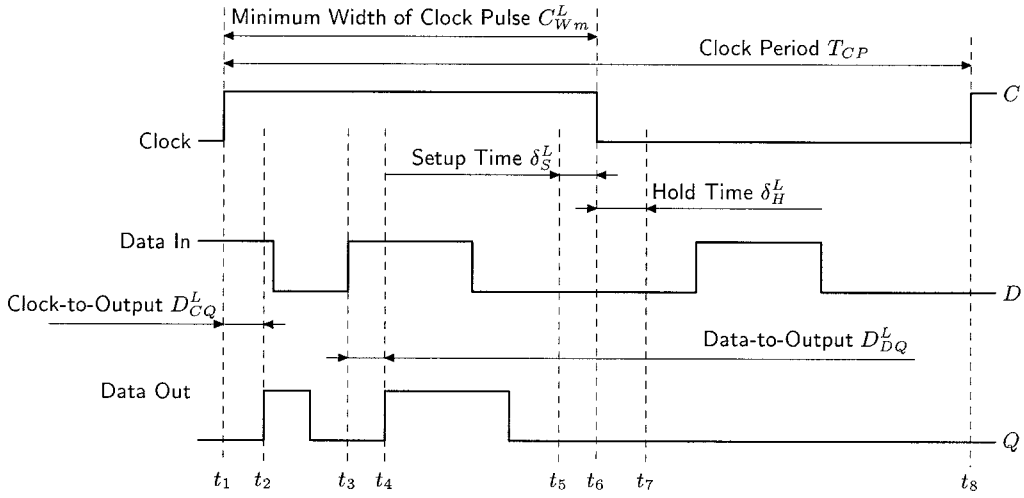


FIGURE 47.9 Parameters of a level-sensitive register.

Latch Hold Time

The *latch hold time* δ^L_H is the minimum time after the trailing clock edge that the data signal must remain constant so that this value of D is successfully stored in the latch during the opaque state. This definition of δ^L_H assumes that the last change of the value of D has occurred to later than δ^L_S before the trailing edge of the clock signal. The term $\delta^L_H = t_7 - t_6$ is shown in Fig. 47.9.

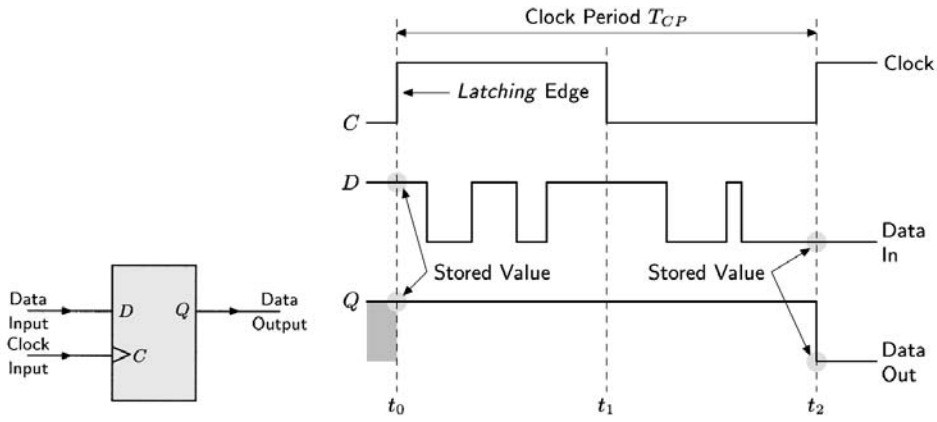
Note: The latch parameters previously introduced are used to refer to any latch in general, or to a specific instance of a latch when this instance can be unambiguously identified. To refer to a specific instance i of a latch explicitly, the parameters are additionally shown with a superscript. For example, D^{Li}_{CQ} refers to the clock-to-output delay of latch i. Also, adding m and M to the subscript of D^L_{CQ} and D^L_{DQ} can be used to refer to the *minimum* and *maximum* values of D^L_{CQ} and D^L_{DQ}, respectively.

Flip-Flops

An *edge-triggered register* or *flip-flop* is a type of register which, unlike the latches described previously, is never transparent with respect to the input data signal.[8, 30-36] The output of a flip-flop normally does not follow the input data signal at any time during the register operation, but rather holds onto a previously stored data value until a new data signal is stored in the flip-flop. A simple type of flip-flop with a clock signal C and an input signal D is shown in Fig. 47.10(a); similar to latches, the output of a flip-flop is usually labeled Q. This specific type of register, shown in Fig. 47.10(a), is called a D flip-flop and its operation is illustrated in Fig. 47.10(b).

In typical flip-flops, data is stored either on the rising edge (low-to-high transition) or on the falling edge (high-to-low transition) of the clock signal. The flip-flops are known as *positive-edge-triggered* and *negative-edge-triggered* flip-flops, respectively. The terms *latching, storing,* or *positive* edge is used to identify the edge of the clock signal on which storage in the flip-flop occurs. For the sake of clarity, the latching edge of the clock signal for flip-flops will also be called the *leading edge* (compare with the previous discusion of latches). Also, note that certain flip-flops — known as *double-edged-triggered* (DET) flip-flops[40-44] — can store data at either edge of the clock signal. The complexity of these flip-flops, however, is significantly higher and these registers are therefore rarely used.

As shown in the timing diagram in Fig. 47.10(b), the output of the flip-flop remains unchanged most of the time, regardless of the transitions in the data signal. Only values of the data signal in the vicinity of the storing edge of the clock signal can affect the output of the flip-flop. Therefore, changes in the



(a) An edge-triggered register or flip-flop. (b) Idealized operation of the flip-flop shown in (a).

FIGURE 47.10 Schematic representation and principle of operation of an edge-triggered register (flip-flop).

output will only be observed when the currently stored data has a logic value x, and the storing edge of the clock signal occurs while the input data signal has a logic value of $\bar{x}$.

Parameters of Flip-Flops

The significant timing parameters of an edge-triggered register are similar to those of latches and are presented next. These parameters are illustrated in Fig. 47.11.

Minimum Width of the Clock Pulse

The *minimum width of the clock pulse* C^F_{Wm} is the *minimum* permissible width of the time interval between the latching edge and the non-latching edge of the clock signal. The minimum width of the clock pulse $C^F_{Wm} = t_6 - t_3$ is shown in Fig. 47.11 and is defined as the minimum interval between the latching and non-latching edges of the clock pulse such that the flip-flop will operate correctly. Further increasing C^F_{Wm} will *not* affect the values of the setup time δ^F_S and hold time δ^F_H (defined later). The clock period $T_{CP} = t_6 - t_1$ is also shown in Fig. 47.11.

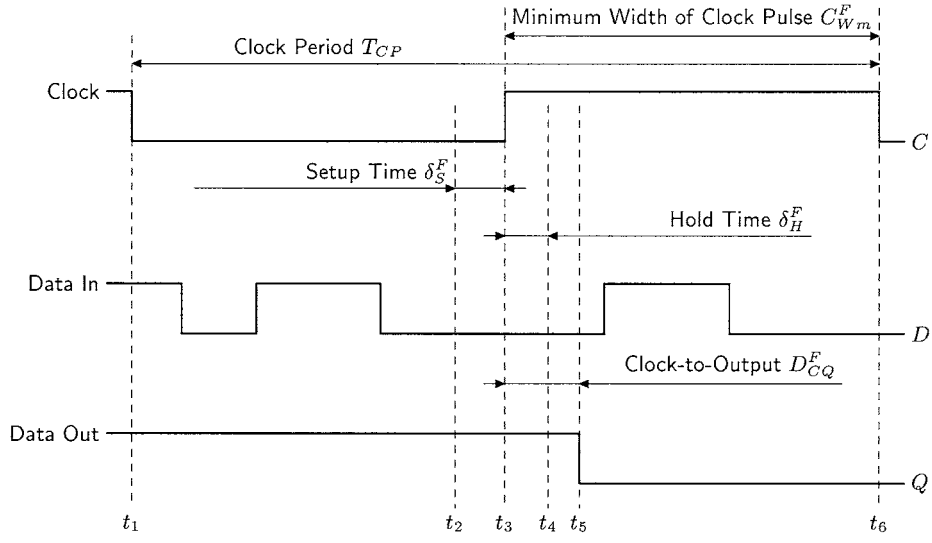


FIGURE 47.11 Parameters of an edge-triggered register.

Flip-Flop Clock-to-Output Delay

As shown in Fig. 47.11, the *clock-to-output delay* D^F_{CQ} of the flip-flop is $D^F_{CQ} = t_5 - t_3$. This propagation delay parameter — typically called the clock-to-Q delay — is the propagation delay from the clock signal terminal to the output terminal. The value of D^F_{CQ} is defined assuming that the data input signal has settled to a stable value sufficiently early, that is, setting the data input any earlier with respect to the latching clock edge will not affect the value of D^F_{CQ}.

Flip-Flop Setup Time

The flip-flop *setup time* δ^F_S is shown in Fig. 47.11 — $\delta^F_S = t_3 - t_2$. The parameter δ^F_S is defined as the *minimum* time between a change in the data signal and the latching edge of the clock signal such that the new value of D propagates to the output Q of the flip-flop and is successfully latched within the flip-flop.

Flip-Flop Hold Time

The flip-flop *hold time* δ^F_H is the minimum time after the arrival of the latching clock edge in which the data signal must remain constant in order to successfully store the D signal within the flip-flop. The hold time $\delta^F_H = t_4 - t_3$ is illustrated in Fig. 47.11. This definition of the hold time assumes that the last change of D has occurred no later than δ^F_S before the arrival of the latching edge of the clock signal.

Note: Similar to latches, the parameters of these edge-triggered registers refer to any flip-flop in general, or to a specific instance of a flip-flop when this instance is uniquely identified. To refer to a specific instance i of a flip-flop explicitly, the flip-flop parameters are additonally shown with a superscript. For example, δ^{Fi}_S refers to the setup time parameter flip-flop i. Also, adding m and M to the subscript of D^F_{CQ} can be used to refer to the *minimum* and *maximum* values of D^F_{CQ}, respectively.

The Clock Signal

The clock signal is typically delivered to each storage element within a circuit. This signal is crucial to the correct operation of a fully synchronous digital system.The storage elements serve to establish the relative sequence of events within a system so that those operations that cannot be executed concurrently operate on the proper data signals.

A typical clock signal $c(t)$ in a synchronous digital system is shown in Fig. 47.12. The *clock period* T_{CP} of $c(t)$ is indicated in Fig. 47.12. In order to provide the highest possible clock frequency, the objective is for T_{CP} to be the smallest number such that

$$\forall t: c(t) = c(t + nT_{CP}) \tag{47.12}$$

where n is an integer. The width of the clock pulse C_W is shown in Fig. 47.12 where the meaning of C_W has been previously explained.

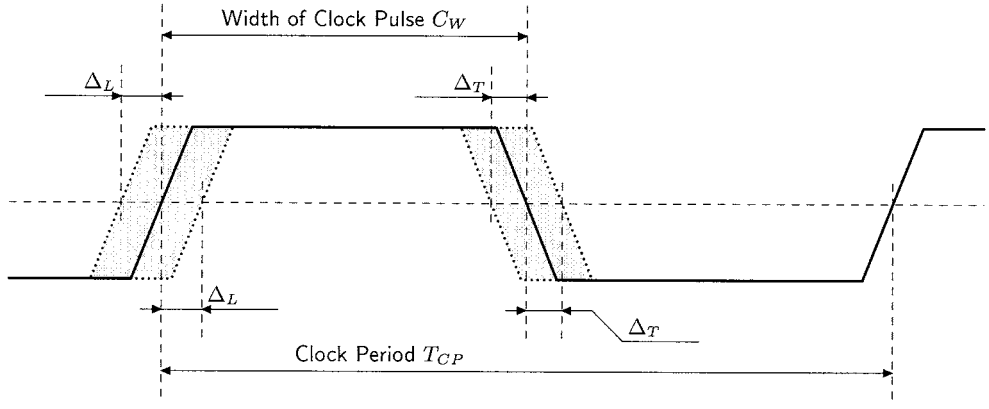


FIGURE 47.12 A typical clock signal.

Typically, the period of the clock signal T_{CP} is a constant,that is, $\partial T_{CP}/\partial t = 0$. If the clock signal $c(t)$ has a delay τ from some reference point, then the leading edges of $c(t)$ occur at times

$$\tau + mT_{CP} \qquad \text{for} \qquad m \in \{\ldots, -2, -1, 0, 1, 2, \ldots\} \tag{47.13}$$

and the trailing edges of c(t) occur at times

$$\tau + C_W + mT_{CP} \qquad \text{for} \qquad m \in \{\ldots, -2, -1, 0, 1, 2, \ldots\} \tag{47.14}$$

In practice, however, it is possible for the edges of a clock signal to fluctuate in time, that is, *not* to occur precisely at the times described by Eqs. 47.13 and 47.14 for the leading and trailing edges, respectively. This phenomenon is known as *clock jitter* and may be due to various causes, such as variations in the manufacturing process, ambient temperature, power supply noise, and oscillator characteristics.

To account for this clock jitter, the following parameters are introduced:

- The maximum deviation $\triangle_L$ of the leading edge of the clock signal: that is, the leading edge is guaranteed to occur anywhere in an interval $(\tau + kT_{CP} - \triangle_L, \tau + kT_{CP} + \triangle_L)$
- The maximum deviation $\triangle_T$ of the trailing edge of the clock signal: that is, the trailing edge is guaranteed to occur anywhere in the interval $(\tau + C_W + kT_{CP} - \triangle_T, \tau + C_W + kT_{CP} + \triangle_T)$,

Clock Skew

Consider a local data path such as the path shown in Fig. 47.2(b). Without loss of generality, assume that the registers shown in Fig. 47.2(b) are flip-flops. The clock signal with period T_{CP} is delivered to each of the registers R_i and R_f. Let the clock signal driving the register R_i be denoted as C_i. and the clock signal driving the registerR_f be denoted by C_f. Also, let t^{i}_{cd} and t^{f}_{cd} be the delays of C_i and C_f to the registers R_i and R_f, respectivly.[13] As described by Eq. 47.13, the latching or leading edges of C_i. occur at times

$$\ldots, \tau + t^{\mathsf{i}}_{cd} - T_{CP}, \tau + t^{\mathsf{i}}_{cd}, \tau + t^{\mathsf{i}}_{cd} + T_{CP}, \ldots$$

Similarly, the latching or leading edges of C_f. occur at times

$$\ldots, \tau + t^{\mathsf{f}}_{cd} - T_{CP}, \tau + t^{\mathsf{f}}_{cd}, \tau + t^{\mathsf{f}}_{cd} + T_{CP}, \ldots$$

as described by Eq. 47.14.

The *clock skew* $T_{Skew}(i, f) = t^{\mathsf{i}}_{cd} - t^{\mathsf{f}}_{cd}$ between $C_i$and C_f is introduced next as the difference of the arrival times of $C_i$and C_f.[13] This concept is illustrated by Fig. 47.13. Note that depending on the values of t^{i}_{cd}

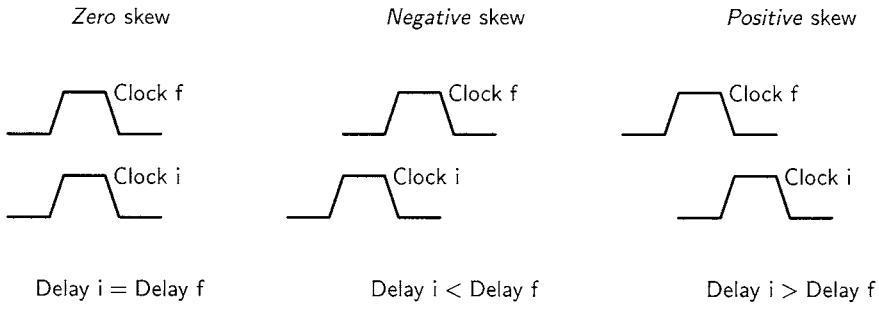


FIGURE 47.13 Lead/lag relationships causing clock skew to be zero, negative, or positive.

[13] Note that these delays t^{i}_{cd} and t^{f}_{cd} are measured with respect to the same reference point.

and t_{cd}^{f}, the skew can be *zero* ($t_{cd}^{i} = t_{cd}^{f}$), *negative* ($t_{cd}^{i} < t_{cd}^{f}$), or *positive* ($t_{cd}^{i} > t_{cd}^{f}$). Furthermore, note that the clock skew as defined above is only defined for sequentially-adjacent registers, that is, a local data path (such as the path shown in Fig. 47.2(b)).

Analysis of a Single-Phase Local Data Path with Flip-Flops

A local data path composed of two flip-flops and combinational logic between the flip-flops is shown in Fig. 47.14. Note the initial flip-flop R_i, which is the origin of the data signal, and the final flip-flop R_f, which is the destination of the data signal. The combinational logic block L_{if} between R_i and R_f accepts the input data signals supplied by R_i and other registers and logic gates and transmits the operated upon data signals to R_f. The period of the clock signal is denoted by T_{CP} and the delays of the clock signal C_i and C_f to the flip-flops R_i and R_f are denoted by t_{cd}^{i} and t_{cd}^{f}, respectively. The input and output data signals to R_i and R_f are denoted by D_i, Q_i, D_f, and Q_f, respectively.

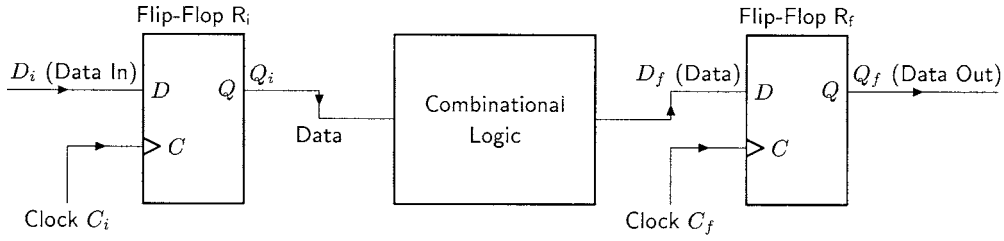


FIGURE 47.14 A single-phase local data path.

An analysis of the timing properties of the local data path shown in Fig. 47.14 is offered in the following sections. First, the timing relationships to prevent the late arrival of data signals to R_f are examined in the next subsection. The timing relationships to prevent the early arrival of signals to the register R_f are then described followed by analyses that borrow some notation from Refs. 11 and 12. Similar analyses of synchronous circuits from the timing perspective can be found in Refs. 45 through 49.

Preventing the Late Arrival of the Data Signal in a Local Data Path with Flip-Flops

The operation of the local data path $R_i \rightsquigarrow R_f$ shown in Fig. 47.14 requires that any data signal that is being stored in R_f arrives at the data input D_f of R_f no later than δ_S^{Ff} before the latching edge of the clock signal C_f. It is possible for the opposite event to occur, that is, for the data signal D_f not to arrive at the register R_f sufficiently early in order to be stored successfully within R_f. If this situation occurs, the local data path shown in Fig. 47.14 fails to perform as expected and it is said that a *timing failure* or *violation* has been created. This form of timing violation is typically called a *setup* (or *long path*) *violation*. A setup violation is depicted in Fig. 47.15 and is used in the following discussion.

The identical clock periods of the clock signals C_i and C_f are shaded for identification in Fig. 47.15. Also shaded in Fig. 47.15 are those portions of the data signals D_i, Q_i, and D_f that are relevant to the operation of the local data path shown in Fig. 47.14. Specifically, the shaded portion of D_i corresponds to the data to be stored in R_i at the beginning of the k-th clock period. This data signal propagates to the output of the register R_i and is illustrated by the shaded portion of Q_i shown in Fig. 47.15. The combinational logic operates on Q_i during the k-th clock period. The result of this operation is the shaded portion of the signal D_f which must be stored in R_f during the next $(k+1)$-st clock period.

Observe that, as illustrated in Fig. 47.15, the leading edge of C_i that initiates the k-th clock period occurs at time $t_{cd}^{i} + kT_{CP}$. Similarly, the leading edge of C_f that initiates the $(k+1)$-th clock period occurs at time $t_{cd}^{f} + (k+1)\,T_{CP}$. Therefore, the *latest arrival time* t_{AM}^{Ff} of D_f at R_f must satisfy.

$$t_{AM}^{Ff} \leq [\,t_{cd}^{f} + (k+1)\,T_{CP} - \Delta_L^{F}] - \delta_S^{Ff} \tag{47.15}$$

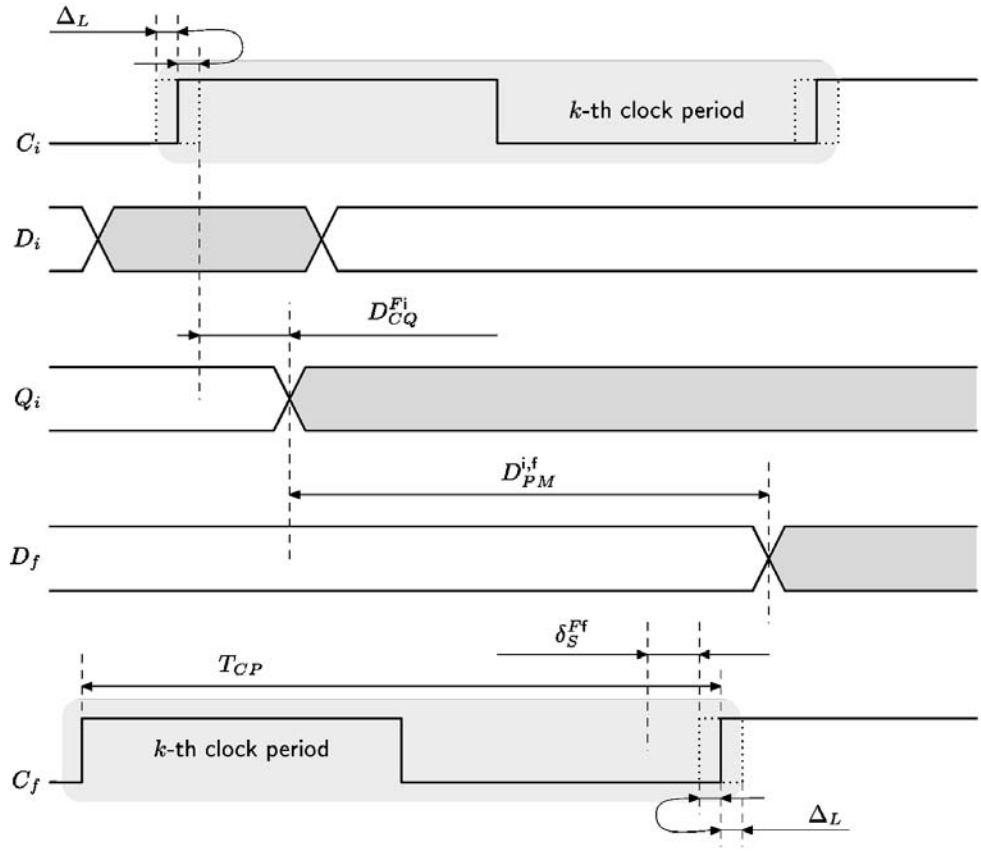


FIGURE 47.15 Timing diagram of a local data path with flip-flops with violation of the setup constraint.

The term $[\,t_{cd}^{f} + (k+1)T_{CP} - \Delta_L^F\,]$ on the right-hand side of Eq. 47.15 corresponds to the critical situation of the leading edge of C_f arriving earlier by the maximum possible deviation Δ_L^F. The $-\delta_S^{Ff}$ term on the right-hand side of Eq. 47.15 accounts for the setup time of R_f (recall the definition of δ_s^F). Note that the value of t_{AM}^{Ff} in Eq. 47.15 consists of two components:

1. The latest arrival time t_{QM}^{Fi} that a valid data signal Q_i appears at the output of R_i: that is, the sum $t_{QM}^{Fi} = t_{cd}^{i} + kT_{CP} + \Delta_L^F + D_{CQM}^{Fi}$ of the latest possible arrival time of the leading edge of C_i and the maximum clock-to-Q delay of R_i,
2. The maximum propagation delay $D_{PM}^{i,f}$ of the data signals through the combinational logic block L_{if} and interconnect along the path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$.

Therefore, t_{AM}^{Ff} can be described as

$$t_{AM}^{Ff} = t_{QM}^{Fi} + D_{PM}^{i,f} = (t_{cd}^{i} + kT_{CP} + \Delta_L^F + D_{CQM}^{Fi}) + D_{PM}^{i,f}. \tag{47.16}$$

By substituting Eq. 47.16 into Eq. 47.15, the timing condition guaranteeing correct signal arrival at the data input D of R_f is

$$(t_{cd}^{i} + kT_{CP} + \Delta_L^F + D_{CQM}^{Fi}) + D_{PM}^{i,f} \le [\,t_{cd}^{f} + (k+1)T_{CP} - \Delta_L^F\,] - \delta_S^{Ff}. \tag{47.17}$$

The above inequality can be transformed by subtracting the kT_{CP} terms from both sides of Eq. 47.17. Furthermore, certain terms in Eq. 47.17 can be grouped together and, by noting that $t_{cd}^{i} - t_{cd}^{f} = T_{Skew}(i, f)$ is the clock skew between the registers R_i and R_f,

$$T_{Skew}(i, f) + 2\Delta_L^F \leq T_{CP} - (D_{CQM}^{Fi} + D_{PM}^{i,f} + \delta_S^{Ff}) \tag{47.18}$$

Note that a violation of Eq. 47.18 is illustrated in Fig. 47.15.

The timing relationship Eq. 47.18 represents three important results describing the late arrival of the signal D_f at the data input of the final register R_f in a local data path $R_i \rightsquigarrow R_f$:

1. Given *any* values of $T_{Skew}(i, f)$ Δ_L^F, $D_{PM}^{i,f}$, δ_S^{Ff}, and D_{CQM}^{Fi}, the late arrival of the data signal at R_f can be prevented by controlling the value of the clock period T_{CP}. A sufficiently large value of T_{CP} can always be chosen to relax Eq. 47.18 by increasing the upper bound described by the right-hand side of Eq. 47.18.
2. For correct operation, the clock period T_{CP} does *not* necessarily have to be larger than the term $(D_{CQM}^{Fi} + D_{PM}^{i,f} + \delta_S^{Ff})$. If the clock skew $T_{Skew}(i, f)$ is properly controlled, choosing a particular negative value for the clock skew will relax the left side of Eq. 47.18, thereby permitting Eq. 47.18 to be satisfied despite $T_{CP} - (D_{CQM}^{Fi} + \hat{D}_{PM}^{i,f} + \delta_S^{Ff}) < 0$.
3. Both the term $2\Delta_L^F$ and the term $(D_{CQM}^{Fi} + \hat{D}_{PM}^{i,f} + \delta_S^{Ff})$ are harmful in the sense that these terms impose a *lower bound* on the clock period T_{CP} (as expected). Although negative skew can be used to relax the inequality of Eq. 47.18, these two terms work against relaxing the values of T_{CP} and $T_{Skew}(i, f)$

Finally, the relationship in Eq. 47.18 can be rewritten in a form that clarifies the upper bound on the clock skew $T_{Skew}(i, f)$ imposed by Eq. 47.18:

$$T_{Skew}(i, f) \leq T_{CP} - (D_{CQM}^{Fi} + D_{PM}^{i,f} + \delta_S^{Ff}) - 2\Delta_L^F \tag{47.19}$$

Preventing the Early Arrival of the Data Signal in a Local Data Path with Flip-Flops

Late arrival of the signal D_f at the data input of R_f (see Fig. 47.14) was analyzed in the previous subsection. In this section, the analysis of the timing relationships of the local data path $R_i \rightsquigarrow R_f$ to prevent early data arrival of D_f is presented. To this end, recall from previous discussion that any data signal D_f being stored in R_f must lag the arrival of the leading edge of C_f by at least δ_H^{Ff}. It is possible for the opposite event to occur, that is, for a new data D_f^{new} to overwrite the value of D_f and be stored within the register R_f. If this situation occurs, the local data path shown in Fig. 47.14 will not perform as desired because of a catastrophic timing violation known as a *hold* (or *short path*) violation.

In this section, hold timing violations are analyzed. It is shown that a hold violation is more dangerous than a setup violation since a hold violation cannot be removed by simply adjusting the clock period T_{CP} (unlike the case of a data signal arriving late where T_{CP} can be increased to satisfy Eq. 47.18). A hold violation is depicted in Fig. 47.16, which is used in the following discussion.

The situation depicted in Fig. 47.16 is different from the situation depicted in Fig. 47.15 in the following sense. In Fig. 47.15, a data signal stored in R_i during the k-th clock period arrives too late to be stored in R_f during the $(k + 1)$-th clock period. In Fig. 47.16, however, the data stored in R_i during the k-th clock period arrives at R_f too early and *destroys* the data that had to be stored in R_f during the same k-th clock period. To clarify this concept, certain portions of the data signals are shaded for easy identification in Fig. 47.16. The data D_i being stored in R_i at the beginning of the k-th clock period is shaded. This data signal propagates to the output of the register R_i and is illustrated by the shaded portion of Q_i shown in Fig. 47.16. The output of the logic (left unshaded in Fig. 47.16) is being stored within the register R_f at the beginning of the $(k + 1)$-th clock period. Finally, the shaded portion of D_f corresponds to the data that must be stored in R_f at the beginning of the k-th clock period.

Note that, as illustrated in Fig. 47.16, the leading (or latching) edge of C_i that initiates the k-th clock period occurs at time $t_{cd}^{i} + kT_{CP}$. Similarly, the leading (or latching) edge of C_f that initiates the k-th clock period occurs at time $t_{cd}^{f} + kT_{CP}$. Therefore, the earliest arrival time t_{Am}^{Ff} of the data signal D_f at the register R_f must satisfy the following condition:

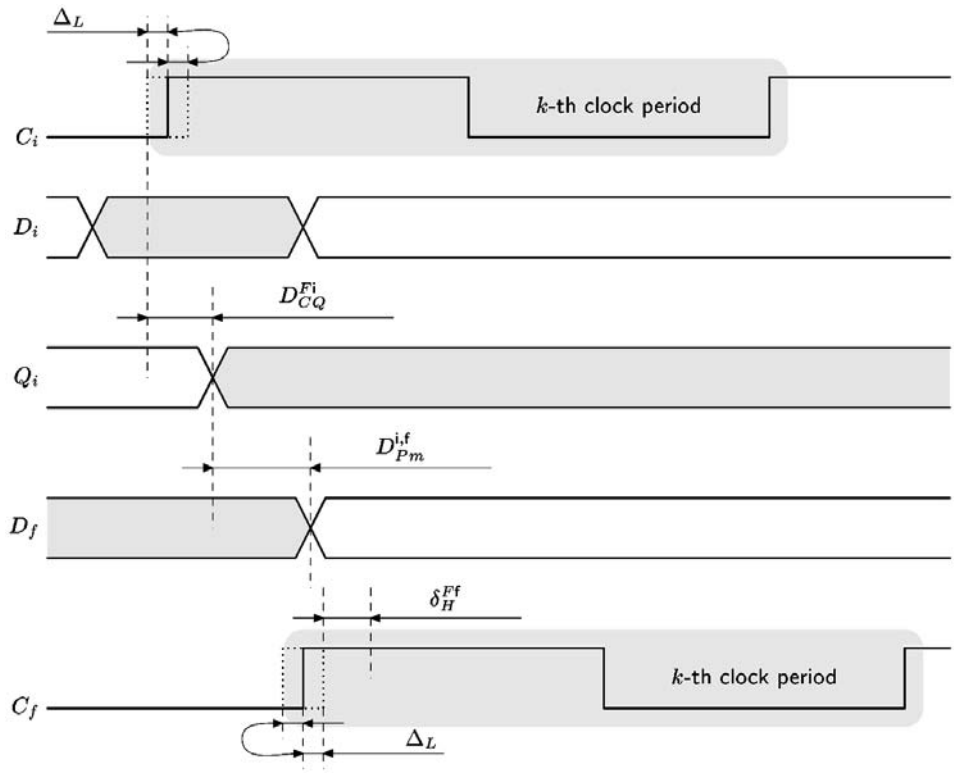


FIGURE 47.16 Timing diagram of a local data path with flip-flops with a violation of the hold constraint.

$$t_{Am}^{Ff} \geq (t_{cd}^{f} + kT_{CP} + \Delta_L^F) + \delta_H^{Ff} \tag{47.20}$$

The term $(t_{cd}^{f} + kT_{CP} + \Delta_L^F)$ on the right-hand side of Eq. 47.20 corresponds to the critical situation of the leading edge of the k-th clock period of C_f arriving late by the maximum possible deviation Δ_L^F. Note that the value of t_{Am}^{Ff} in Eq. 47.20 has two components:

1. The earliest arrival time t_{Qm}^{Fi} that a valid data signal Q_i appears at the output of $\mathsf{R_i}$: that is, the sum $t_{Qm}^{Fi} = t_{cd}^{i} + kT_{CP} - \Delta_L^F + D_{CQm}^{Fi}$ of the earliest arrival time of the leading edge of C_i and the minimum clock-to-Q delay of $\mathsf{R_i}$
2. The minimum propagation delay $D_{Pm}^{i,f}$ of the signals through the combinational logic block $\mathsf{L_{if}}$ and interconnect wires along the path $\mathsf{R_i} \rightsquigarrow \mathsf{R_f}$

Therefore, t_{Am}^{Ff} can be described as

$$t_{Am}^{Ff} = t_{Qm}^{Ff} + D_{Pm}^{i,f} = (t_{cd}^{i} + kT_{CP} - \Delta_L^F + D_{CQM}^{Fi}) + D_{Pm}^{i,f} \tag{47.21}$$

By substituting Eq. 47.21 into Eq. 47.20, the timing condition that guarantees that D_f does *not* arrive too early at $\mathsf{R_f}$ is

$$(t_{cd}^{i} + kT_{CP} - \Delta_L^F + D_{CQm}^{Fi}) + D_{Pm}^{i,f} \geq (t_{cd}^{f} + kT_{CP} + \Delta_L^F) + \delta_H^{Ff} \tag{47.22}$$

The inequality Eq. 47.22 can be further simplified by regrouping terms and noting that $t_{cd}^{i} - t_{cd}^{f} = T_{Skew}(i, f)$ is the clock skew between the registers $\mathsf{R_i}$ and $\mathsf{R_f}$:

$$T_{Skew}(i, f) - 2\Delta_L^F \geq -(D_{CQm}^{Fi} + D_{Pm}^{i,f}) + \delta_H^{Ff} \tag{47.23}$$

Recall that a violation of Eq. 47.23 is illustrated in Fig. 47.16.

The timing relationship described by Eq. 47.23 provides certain important facts describing the early arrival of the signal D_f at the data input of the final register R_f of a local data path:

1. Unlike Eq. 47.18, the inequality Eq. 47.23 does not depend on the clock period T_{CP}. Therefore, a violation of Eq. 47.23 *cannot* be corrected by simply manipulating the value of T_{CP}. A synchronous digital system with hold violations is non-functional, while a system with setup violations will still operate correctly at a reduced speed.[14] For this reason, hold violations result in catastrophic timing failure and are considered significantly more dangerous than the setup violations previously described.
2. The relationship in Eq. 47.23 can be satisfied with a sufficiently large value of the clock skew $T_{Skew}(i, f)$. However, both the term $2\Delta_L^F$ and the term δ_H^{Ff} are harmful in the sense that these terms impose a lower bound on the clock skew $T_{Skew}(i, f)$ between the register R_i and R_f. Although positive skew may be used to relax Eq. 47.23, these two terms work against relaxing the values of $T_{Skew}(i, f)$ and $(D_{CQm}^{Fi} + D_{Pm}^{i,f})$.

Finally, the relationship in Eq. 47.23 can be rewritten to stress the lower bound imposed on the clock skew $T_{Skew}(i, f)$ by Eq. 47.23:

$$T_{Skew}(i, f) \geq -(D_{Pm}^{i,f} + D_{CQ}^{Fi}) + \delta_H^{Ff} + 2\Delta_L^F \tag{47.24}$$

Analysis of a Single-Phase Local Data Path with Latches

A local data path consisting of two level-sensitive registers (or latches) and the combinational logic between these registers (or latches) is shown in Fig. 47.17. Note the initial latch R_i, which is the origin of the data signal, and the final latch R_f, which is the destination of the data signal. The combinational logic block L_if between R_i and R_f accepts the input data signals sourced by R_i and other registers and logic gates and transmits the data signals that have been operated on to R_f . The period of the clock signal is denoted by T_{CP} and the delays of the clock signals C_i and C_f to the latches R_i and R_f are denoted by t_{cd}^{i} and t_{cd}^{f}, respectively. The input and output data signals to R_i and R_f are denoted by D_i, Q_i, D_f, and Q_f, respectively.

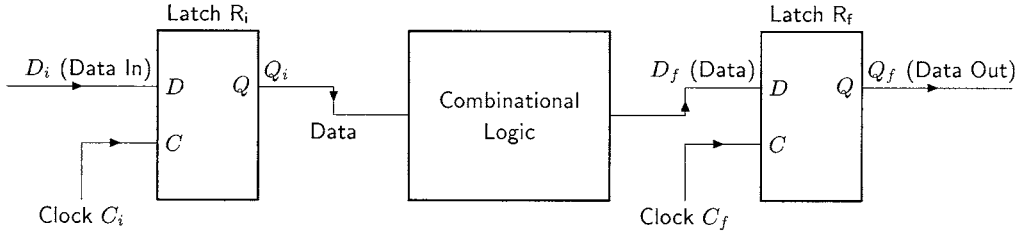


FIGURE 47.17 A single-phase local data path with latches.

An analysis of the timing properties of the local data path shown in Fig. 47.17 is offered in the following sections. The timing relationships to prevent the late arrival of the data signal at the latch R_f are examined, as well as the timing relationships to prevent the early arrival of the data signal at the latch R_f.

[14] Increasing the clock period T_{CP} in order to satisfy Eq. 47.18 is equivalent to reducing the frequency of the clock signal.

The analyses presented in this section build on assumptions regarding the timing relationships among the signals of a latch similar to those assumptions used in the previous chapter section. Specifically, it is guaranteed that every data signal arrives at the data input of a latch no later than δ_S^L time before the trailing clock edge. Also, this data signal must remain stable at least δ_H^L time after the trailing edge, that is, no new data signal should arrive at a latch δ_H^L time after the latch has become opaque.

Observe the differences between a latch and a flip-flop.[45,50] In flip-flops, the setup and hold requirements described in the previous paragraph are relative to the *leading* — not to the trailing — *edge* of the clock signal. Similar to in flip-flops, the late and early arrival of the data signal to a latch give rise to timing violations known as setup and hold violations, respectively.

Preventing the Late Arrival of the Data Signal in a Local Data Path with Latches

A similar signal setup to the example illustrated in Fig. 47.15 is assumed in the following discussion. A data signal D_i, is stored in the latch $\mathsf{R_i}$ during the k-th clock period. The data Q_i, stored in $\mathsf{R_i}$ propagates through the combinational logic $\mathsf{L_{if}}$ and the interconnect along the path $\mathsf{R_i} \rightsquigarrow \mathsf{R_f}$. In the $(k+1)$-th clock period, the result D_f of the computation in $\mathsf{L_{if}}$ is stored within the latch $\mathsf{R_f}$. The signal D_f must arrive at least δ_S^L time before the trailing edge of C_f in the $(k+1)$-th clock period.

Similar to the discussion presented in the previous section, the *latest arrival time* $t_{AM}^{L\mathsf{f}}$ of D_f at the D input of $\mathsf{R_f}$ must satisfy

$$t_{AM}^{L\mathsf{f}} \leq [\, t_{cd}^{\mathsf{f}} + (k+1)\, T_{CP} + C_{Wm}^{L} - \Delta_T^L] - \delta_S^{L\mathsf{f}} \tag{47.25}$$

Note the difference between Eqs. 47.25 and 47.15. In Eq. 47.15, the first term on the right-hand side is $[\, t_{cd}^{\mathsf{f}} + (k+1)\, T_{CP} - \Delta_L^F\,]$, while in Eq. 47.25, the first term on the right-hand side has an additional term C_{Wm}^L. The addition of C_{Wm}^L corresponds to the concept that, unlike flip-flops, a data signal is stored in a latch, shown in Fig. 47.17, at the trailing edge of the clock signal (the C_{Wm}^L term). Similar to the case of flip-flops, the term $[\, t_{cd}^{\mathsf{f}} + (k+1)\, T_{CP} + C_{Wm}^L - \Delta_T^L\,]$ in the right-hand side of Eq. 47.25 corresponds to the critical situation of the trailing edge of the clock signal C_f arriving earlier by the maximum possible deviation Δ_T^L.

Observe that the value of $t_{AM}^{L\mathsf{f}}$ in Eq. 47.25 consists of two components:

1. The latest arrival time $t_{QM}^{L\mathsf{i}}$ when a valid data signal Q_i appears at the output of the latch $\mathsf{R_i}$,
2. The maximum signal propagation delay through the combinational logic block $\mathsf{L_{if}}$ and the interconnect along the path $\mathsf{R_i} \rightsquigarrow \mathsf{R_f}$

Therefore, $t_{AM}^{L\mathsf{f}}$ can be described as

$$t_{AM}^{L\mathsf{f}} \;=\; D_{PM}^{\mathsf{i},\,\mathsf{f}} + t_{QM}^{L\mathsf{i}} \tag{47.26}$$

However, unlike the situation of flip-flops discussed previously, the term $t_{Qm}^{L\mathsf{i}}$ on the right-hand side of Eq. 47.26 is not the sum of the delays through the register $\mathsf{R_i}$. The reason is that the value of $t_{QM}^{L\mathsf{i}}$ depends on whether the signal D_i arrived *before* or *during* the transparent state of $\mathsf{R_i}$ in the k-th clock period. Therefore, the value of $t_{Qm}^{L\mathsf{i}}$ in Eq. 47.26 is the greater of the following two quantities:

$$t_{QM}^{L\mathsf{i}} \;=\; \max[(t_{AM}^{L\mathsf{i}} + D_{DQM}^{L\mathsf{i}}),\, (t_{cd}^{\mathsf{i}} + kT_{CP} + \Delta_L^L + D_{CQM}^{L\mathsf{i}})] \tag{47.27}$$

There are two terms in the right-hand side of Eq. 47.27:

1. The term $(\, t_{AM}^{L\mathsf{i}} + D_{DQM}^{L\mathsf{i}}\,)$ corresponds to the situation in which D_i arrives at $\mathsf{R_i}$ after the leading edge of the k-th clock period
2. The term $(\, t_{cd}^{\mathsf{i}} + kT_{CP} + \Delta_L^L + D_{CQM}^{L\mathsf{i}}\,)$ corresponds to the situation in which D_i arrives at $\mathsf{R_i}$ before the leading edge of the k-th clock pulse arrives

By substituting Eq. 47.27 into Eq. 47.26, the latest time of arrival t_{AM}^{Lf} is:

$$t_{AM}^{Lf} = D_{PM}^{i,f} + \max[(t_{AM}^{Li} + D_{DQM}^{Li}), (t_{cd}^{i} + kT_{CP} + \Delta_L^L + D_{CQM}^{Li})] \tag{47.28}$$

which is in turn substituted into Eq. 47.25 to obtain

$$\begin{aligned} &D_{PM}^{i,f} + \max[(t_{AM}^{Li} + D_{DQM}^{Li}), (t_{cd}^{i} + kT_{CP} + \Delta_L^L + D_{CQM}^{Li})] \\ &\leq [t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^{L} - \Delta_T^L] - \delta_S^{Lf} \end{aligned} \tag{47.29}$$

Equation Eq. 47.29 is an expression for the inequality that must be satisfied in order to prevent the late arrival of a data signal at the data input D of the register R_f. By satisfying Eq. 47.29, setup violations in the local data path with latches shown in Fig. 47.17 are avoided. For a circuit to operate correctly, Eq. 47.29 must be enforced for any local data path $\mathsf{R}_i \rightsquigarrow \mathsf{R}_f$ consisting of the latches R_i and R_f.

The max operation in Eq. 47.29 creates a mathematically difficult situation since it is unknown which of the quantities under the max operation is greater. To overcome this obstacle, this max operation can be split into two conditions:

$$D_{PM}^{i,f} + (t_{AM}^{Li} + D_{DQM}^{Li}) \leq [t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^{L} - \Delta_T^L] - \delta_S^{Lf} \tag{47.30}$$

$$D_{PM}^{i,f} + (t_{cd}^{i} + kT_{CP} + \Delta_L^L + D_{CQM}^{Li}) \leq [t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^{L} - \Delta_T^L] - \delta_S^{Lf} \tag{47.31}$$

Taking into account that the clock skew $T_{Skew}(i,f) = t_{cd}^{i} - t_{cd}^{f}$, Eqs. 47.30 and 47.31 can be rewritten as

$$D_{PM}^{i,f} + (t_{AM}^{Li} + D_{DQM}^{Li}) \leq [t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^{L} - \Delta_T^L] - \delta_S^{Lf} \tag{47.32}$$

$$T_{Skew}(i,f) + (\Delta_L^L + \Delta_T^L) \leq (T_{CP} + C_{Wm}^{L}) - (D_{CQM}^{Li} + D_{PM}^{i,f} + \delta_S^{Lf}) \tag{47.33}$$

Equation 47.33 can be rewritten in a form that clarifies the upper bound on the clock skew $T_{Skew}(i,f)$ imposed by Eq. 47.33:

$$D_{PM}^{i,f} + (t_{AM}^{Li} + D_{DQM}^{Li}) \leq [t_{cd}^{f} + (k+1)T_{CP} + C_{Wm}^{L} - \Delta_T^L] - \delta_S^{Lf} \tag{47.34}$$

$$T_{Skew}(i,f) \leq (T_{CP} + C_{Wm}^{L} - \Delta_L^L - \Delta_T^L) - (D_{CQM}^{Li} + D_{PM}^{i,f} + \delta_S^{Lf}) \tag{47.35}$$

Preventing the Early Arrival of the Data Signal in a Local Data Path with Latches

A similar signal setup to the example illustrated in Fig. 47.16 is assumed in the discussion presented in this section. Recall the difference between the late arrival of a data signal at R_f and the early arrival of a data signal at R_f. In the former case, the data signal stored in the latch R_i during the k-th clock period arrives too late to be stored in the latch R_f during the $(k+1)$-th clock period. In the latter case, the data signal stored in the latch R_i during the k-th clock period propagates to the latch R_f too early and overwrites the data signal that was already stored in the latch R_f during the same k-th clock period.

In order for the proper data signal to be successfully latched within R_f during the k-th clock period, there should not be any changes in the signal D_f until at least the hold time after the arrival of the storing (trailing) edge of the clock signal C_f. Therefore, the earliest arrival time t_{Am}^{Lf} of the data signal D_f at the register R_f must satisfy the following condition:

$$t_{Am}^{Lf} \geq (t_{cd}^{f} + kT_{CP} + C_{Wm}^{L} + \Delta_T^L) + \delta_H^{Lf} \tag{47.36}$$

The term $(t_{cd}^{\mathrm{f}} + \mathrm{k}T_{CP} + C_{Wm}^{L} + \Delta_{T}^{L})$ on the right-hand side of Eq. 47.36 corresponds to the critical situation of the trailing edge of the k-th clock period of the clock signal C_f arriving late by the maxiumum possible deviation Δ_{T}^{L}. Note that the value of t_{Am}^{Lf} in Eq. 47.36 consists of two components:

1. The earliest arrival time t_{Qm}^{Li} that a valid data signal Q_i appears at the output of the latch R_{i}: that is, the sum $t_{Qm}^{Li} = t_{cd}^{\mathrm{i}} + \mathrm{k}T_{CP} - \Delta_{L}^{L} + D_{CQm}^{Li}$ of the earliest arrival time of the leading edge of the clock signal C_i and the minimum clock-to-Q delay D_{CQm}^{Li} of R_{f},
2. The minimum propagation delay $D_{Pm}^{\mathrm{i,f}}$ of the signal through the combinational logic L_{if} and the interconnect along the path $\mathsf{R}_{\mathsf{i}} \rightsquigarrow \mathsf{R}_{\mathsf{f}}$.

Therefore, t_{Am}^{Lf} can be described as

$$t_{Am}^{Lf} = t_{Qm}^{Li} + D_{Pm}^{\mathrm{i,f}} = (t_{cd}^{\mathrm{i}} + kT_{CP} - \Delta_{L}^{L} + D_{CQm}^{Li}) + D_{Pm}^{\mathrm{i,f}} \tag{47.37}$$

By substituting Eq. 47.37 into Eq. 47.36, the timing condition guaranteeing that D_f does *not* arrive too early at the latch R_{f} is

$$(t_{cd}^{\mathrm{i}} + kT_{CP} - \Delta_{L}^{L} + D_{CQm}^{Li}) + D_{Pm}^{\mathrm{i,f}} \geq (t_{cd}^{\mathrm{f}} + kT_{CP} + C_{Wm}^{L} + \Delta_{T}^{L}) + \delta_{H}^{Lf} \tag{47.38}$$

The inequality Eq. 47.38 can be further simplified by reorganizing the terms and noting that $t_{cd}^{\mathrm{i}} - t_{cd}^{\mathrm{f}} = T_{Skew}(i, f)$ is the clock skew between the registers R_{i} and R_{f}:

$$T_{Skew}(i, f) - (\Delta_{L}^{L} + \Delta_{T}^{L}) \geq -(D_{CQm}^{Li} + D_{Pm}^{\mathrm{i,f}}) + \delta_{H}^{Lf} \tag{47.39}$$

The timing relationship described by Eq. 47.39 represents two important results describing the early arrival of the signal D_f at the data input of the final latch R_{f} of a local data path:

1. The relationship in Eq. 47.39 does not depend on the value of the clock period T_{CP}. Therefore, if a hold timing violation in a synchronous system has occurred,[15] this timing violation is catastrophic.
2. The relationship in Eq. 47.39 can be satisfied with a sufficiently large value of the clock skew $T_{Skew}(i, f)$. Furthermore, both the term $(\Delta_{L}^{L} + \Delta_{T}^{L})$ and the term δ_{H}^{Lf} are harmful in the sense that these terms impose a lower bound on the clock skew $T_{Skew}(i, f)$ between the latches R_{i} and R_{f}. Although positive skew $T_{Skew}(i, f) > 0$ can be used to relax Eq. 47.39, these two terms make it difficult to satisfy the inequality in Eq. 47.39 for specific values of $T_{Skew}(i, f)$ and $(D_{CQm}^{Li} + D_{Pm}^{\mathrm{i,f}})$.

Furthermore, Eq. 47.39 can be rewritten to emphasize the lower bound on the clock skew $T_{Skew}(i, f)$ imposed by Eq. 47.39:

$$T_{Skew}(i, f) \geq (\Delta_{L}^{L} + \Delta_{T}^{L}) - (D_{CQm}^{Li} + D_{Pm}^{\mathrm{i,f}}) + \delta_{H}^{Lf} \tag{47.40}$$

47.5 A Final Note

The properties of registers and local data paths were described in this chapter. Specifically, the timing relationships to prevent setup and hold timing violations in a local data path consisting of two positive edge-triggered flip-flops were analyzed. The timing relationships to prevent setup and hold timing violations in a local data path consisting of two positive-polarity latches were also analyzed.

In a fully synchronous digital VLSI system, however, it is possible to encounter types of local data paths different from those circuits analyzed in this chapter. For example, a local data path may begin

[15] As described by the inequality Eq. 47.39 not being satisfied.

with a *positive*-polarity, edge-sensitive register R_i, and end with a *negative*-polarity, edge-sensitive register R_f. It is also possible that different types of registers are used, for example, a register with more than one data input. In each individual case, the analyses described in this chapter illustrate the general methodology used to derive the proper timing relationships specific to that system. Furthermore, note that for a given system, the timing relationships that must be satisfied for the system to operate correctly — such as Eqs. 47.19, 47.24, 47.34, 47.35, and 47.40 — are collectively referred to as the overall *timing constraints* of the synchronous digital system.[13,51–55]

References

1. Kilby, J. S., "Invention of the Integrated Circuit," *IEEE Transactions on Electron Devices,* vol. ED-23, pp. 648-654, July 1976.
2. Rabaey, J. M., *Digital Integrated Circuits: A Design Perspective.* Prentice-Hall, Inc., Upper Saddle River, NJ, 1995.
3. Gaddis, N. and Lotz, J., "A 64-b Quad-Issue CMOS RISC Microprocessor," *IEEE Journal of Solid-State Circuits,* vol. SC-31, pp. 1697-1702, Nov. 1996.
4. Gronowski, P. E. et al., "A 433-MHz 64-bit Quad-Issue RISC Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. SC-31, pp. 1687-1696, Nov. 1996.
5. Vasseghi, N., Yeager, K., Sarto, E., and Seddighnezhad, M., "200-Mhz Superscalar RISC Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. SC-31, pp. 1675-1686, Nov. 1996.
6. Bakoglu, H. B., *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley Publishing Company, 1990.
7. Bothra, S., Rogers, B., Kellam, M., and Osburn, C. M., "Analysis of the Effects of Scaling on Interconnect Delay in ULSI Circuits," *IEEE Transactions on Electron Devices,* vol. ED-40, pp. 591-597, Mar. 1993.
8. Weste, N. W. and Eshraghian, K., *Principles of CMOS VLSI Design: A Systems Perspective.* Addison-Wesley Publishing Company, Reading, MA, 2nd ed., 1992.
9. Mead, C. and Conway, L., *Introduction to VLSI Systems.* Addison-Wesley Publishing Company, Reading, MA, 1980.
10. Anceau, F., "ASynchronous Approach for Clocking VLSI Systems," *IEEE Journal of Solid-State Circuits,* vol. SC-17, pp. 51-56, Feb. 1982.
11. Afghani M., and Svensson, C., "A Unified Clocking Scheme for VLSI Systems," *IEEE Journal of Solid State Circuits,* vol. SC-25, pp. 225-233, Feb. 1990.
12. Unger, S. H. and Tan, C.-J., "Clocking Schemes for High-Speed Digital Systems," *IEEE Transactions on Computers,* vol. C.-35, pp. 880-895, Oct. 1986.
13. Friedman, E. G., *Clock Distribution Networks in VLSI Circuits and Systems.* IEEE Press, 1995.
14. Bowhill, W. J. et al., "Circuit Implementation of a 300-MHz 64-bit Second-generation CMOS Alpha CPU," *Digital Technial Journal*, vol. 7, no. 1, pp. 100-118, 1995.
15. Neves, J. L. and Friedman, E. G., "Topological Design of Clock Distribution Networks Based on Non-Zero Clock Skew Specification," *Proceedings of the 36th IEEE Midwest Symposium on Circuits and Systems*, pp. 468-471, Aug. 1993.
16. Xi, J. G. and Dai, W. W.-M., "Useful-Skew Clock Routing With Gate Sizing for Low Power Design," *Proceedings of the 33rd ACM/IEEE Design Automation Conference,* pp. 383-388, June 1996.
17. Neves, J. L. and Friedman, E. G., "Design Methodology for Synthesizing Clock Distribution Networks Exploiting Non-Zero Localized Clock Skew," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. VLSI-4, pp. 286-291, June 1996.
18. Jackson, M. A. B., Srinivasan, A., and Kuh, E. S., "Clock Routing for High-Performance ICs," *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pp. 573-579, June 1990.
19. Tsay, R.-S., "An Exact Zero-Skew Clock Routing Algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. CAD-12, pp. 242-249, Feb. 1993.

20. Chou, N.-C. and Cheng, C.-K., "On General Zero-Skew Clock New Construction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. VLSI-3, pp. 141-146, Mar. 1995.
21. Ito, N., Sugiyama, H., and Konno, T., "ChipPRISM: Clock Routing and Timing Analysis for High-Performance CMOS VLSI Chips," *Fujitsu Scientific and Technical Jornal*, vol. 31, pp. 180-187, Dec. 1995.
22. Leiserson, C. E. and Saxe, J. B., "A Mixed-Integer Linear Programming Problem Which is Efficiently Solvable," *Journal of Algorithms*, vol. 9, pp. 114-128, Mar. 1988.
23. Cormen, T. H., Leiserson, C. E., and Rivest, R. L., *Introduction to Algorithms.* MIT Press, 1989.
24. West, D. B., *Introduction to Grpah Theory.* Prentice-Hall, 1996.
25. Fishburn, J. P., "Clock Skew Optimization," *IEEE Transactions on Computers,* vol. C-39, pp. 945-951, July 1990.
26. Lee, T.-C. and Kong, J., "The New Line in IC Design," *IEEE Spectrum,* pp. 52-58, Mar. 1997.
27. Friedman, E. G., "The Application of Localized Clock Distribution Design to Improving the Performance of Retimed Sequential Circuits," *Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 12-17, Dec. 1992.
28. Kourtev, I. S. and Friedman, E. G., "Simultaneous Clock Scheduling and Buffered Clock Tree Synthesis," *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1812-1815, June 1997.
29. Neves, J. L. and Friedman, E. G., "Optimal Clock Skew Scheduling Tolerant to Process Variations," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 623-628, June 1996.
30. Glasser, L. A. and Dobberpuhl, D. W., *The Design and Analysis of VLSI Circuits.* Addison-Wesley Publishing Company, 1985.
31. Uyemura, J. P., *Circuit Design for CMOS VLSI.* Kluwer Academic Publishers, 1992.
32. Kang, S. M. and Leblebici, Y., *CMOS Digital Integrated Circuits: Analysis and Design.* The McGraw-Hill Companies, Inc., 1996.
33. Sedra, A. S. and Smith, K. C., *Microelectronic Circuits.* Oxford University Press, 4th ed., 1997.
34. Kohavi, Z., *Switching and Finite Automata Theory.* McGraw-Hill Book Company, New York, NY, 2nd ed., 1978.
35. Mano, M. M. and Kime, C. R., *Logic and Computer Design Fundamentals.* Prentice-Hall, Inc., 1997.
36. Wolf, W., *Modern VLSI Design: A Systems Approach.* Prentice-Hall, Inc., 1994.
37. Kacprzak, T. and Albicki, A., "Analysis of Metastable Operation in RS CMOS Flip-Flops," *IEEE Journal of Solid-State Circuits,* vol. SC-22, pp. 57-64, Feb. 1987.
38. Jackson, T. A. and Albicki, A., "Analysis of Metastable Operation in D latches," *IEEE Transactions on Circuits and Systems — I: Fundamental Theory and Applications*, vol. CAS I-36, pp. 1392-1404, Nov. 1989.
39. Friedman, E. G., "Latching Characteristics of a CMOS Bistable Register," *IEEE Transactions on Circuits and Systems—I: Fundamental Theory and Applications*, vol. CAS I-40, pp. 902-908, Dec. 1993.
40. Unger, S. H., "Double-Edge-Triggered Flip-Flops," *IEEE Transactions on Computers*, vol. C-30, pp. 447-451, June 1981.
41. Lu, S.-L., "A Novel CMOS Implementation of Double-Edge-Triggered D-flip-flops," *IEEE Journal of Solid State Circuits,* vol. SC-25, pp. 1008-1010, Aug. 1990.
42. Afghani, M. and Yuan, J., "Double-Edge-Triggered D-Flip-Flops for High-Speed CMOS Circuits," *IEEE Journal of Solid State Circuits*, vol. SC-26, pp. 1168-1170, Aug. 1991.
43. Hossain, R., Wronski, L., and Albicki, A., "Double Edge Triggered Devices: Speed and Power Constraints," *Proceedings of the 1996 IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 1491-1494, 1993.
44. Blair, G. M., "Low-Power Double-Edge Triggered Flip-Flop," *Electronics Letters,* vol. 33, pp. 845-847, May 1997.

45. Lin, I., Ludwig, J. A., and Eng, K., "Analyzing Cycle Stealing on Synchronous Circuits with Level-Sensitive Latches," *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp. 393-398, June 1992.
46. Lee, J. fuw, Tang, D. T., and Wong, C. K., "A Timing Analysis Algorithm for Circuits with Level-Sensitive Latches," *IEEE Transactions on Computer-Aided Design*, vol. CAD-15, pp. 535-543, May 1996.
47. Szymanski, T. G., "Computing Optimal Clock Schedules," *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp. 399-404, June 1992.
48. Dagenais, M. R. and Rumin, N. C., "On the Calculation of Optimal Clocking Parameters in Synchronous Circuits with Level-Sensitive Latches," *IEEE Transactions on Computer-Aided Design*, vol. CAD-8, pp. 268-278, Mar. 1989.
49. Sakallah, K. A., Mudge, T. N., and Olukotun, O. A., "*check*T_c and *min*T_c: Timing Verification and Optimal Clocking of Synchronous Digital Circuits," *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 552-555, Nov. 1990.
50. Sakallah, K. A., Mudge, T. N., and Olukotun, O. A., "Analysis and Design of Latch-Controlled Synchronous Digital Circuits," *IEEE Transactions on Computer-Aided Design*, vol. CAD-11, pp. 322-333, Mar. 1992.
51. Kourtev, I. S. and Friedman, E. G., "Topological Synthesis of Clock Trees with Non-Zero Clock Skew," *Proceedings of the 1997 ACM/IEEE International Workshop on Timing Issues in the Specification and Design of Digital Systems*, pp. 158-163, Dec. 1997.
52. Kourtev, I. S. and Friedman, E. G., "Topological Synthesis of Clock Trees for VLSI-Based DSP Systems," *Proceedings of the IEEE Workshop on Signal Processing Systems,* pp. 151-162, Nov. 1997.
53. Kourtev, I. S., and Friedman, E. G., "Integrated Circuit Signal Delay," *Encyclopedia of Electrical and Electronics Engineering*. Wiley Publishing Company, vol. 10, pp. 378-392, 1999.
54. Neves, J. L. and Friedman, E. G., "Synthesizing Distributed Clock Trees for High Performance ASICs," *Proceedings of the IEEE ASIC Conference,* pp. 126-129, Sept. 1994.
55. Neves, J. L. and Friedman, E. G., "Buffered Clock Tree Synthesis with Optimal Clock Skew Scheduling for Reduced Sensitivity to Process Parameter Variations," *Proceedings of the ACM/SIGDA International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pp. 131-141, Nov. 1995.
56. Deokar, R. R. and Sapatnekar, S. S., "A Fresh Look at Retiming via Clock Skew Optimization," *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, pp. 310-315, June 1995.

Appendix

Glossary of Terms

The following notations are used in this section.

1. Clock Signal Parameters
   - T_{CP}: The clock period of a circuit
   - Δ_L: The tolerance of the leading edge of any clock signal
   - Δ_T: The tolerance of the trailing edge of any clock signal
   - Δ_L^L: The tolerance of the leading edge of a clock signal driving a latch
   - Δ_T^L: The tolerance of the trailing edge of a clock signal driving a latch
   - Δ_L^F: The tolerance of the leading edge of a clock signal driving a flip-flop
   - Δ_T^F: The tolerance of the trailing edge of a clock signal driving a flip-flop
   - C_{Wm}^L: The minimum width of the clock signal in a circuit with latches
   - C_{Wm}^F: The minimum width of the clock signal in a circuit with flip-flops
2. Latch Parameters
   - D_{CQ}^L: The clock-to-output delay of a latch
   - D_{CQ}^{Li}: The clock-to-output delay of the latch R_i
   - D_{CQm}^L: The minimum clock-to-output delay of a latch
   - D_{CQm}^{Li}: The minimum clock-to-output delay of the latch R_i
   - D_{CQM}^L: The maximum clock-to-output delay of a latch
   - D_{CQM}^{Li}: The maximum clock-to-output delay of the latch R_i
   - D_{DQ}^L: The data-to-output delay of a latch
   - D_{DQ}^{Li}: The data-to-output delay of the latch R_i
   - D_{DQm}^L: The minimum data-to-output delay of a latch
   - D_{DQm}^{Li}: The minimum data-to-output delay of the latch R_i
   - D_{DQM}^L: The maximum data-to-output delay of a latch
   - D_{DQM}^{Li}: The maximum data-to-output delay of the latch R_i
   - δ_S^L: The setup time of a latch
   - δ_S^{Li}: The setup time of the latch R_i
   - δ_H^L: The hold time of a latch
   - δ_H^{Li}: The hold time of the latch R_i
   - t_{AM}^L: The latest arrival time of the data signal at the data input of a latch
   - t_{AM}^{Li}: The latest arrival time of the data signal at the data input of the latch R_i
   - t_{Am}^L: The earliest arrival time of the data signal at the data input of a latch
   - t_{Am}^{Li}: The earliest arrival time of the data signal at the data input of the latch R_i
   - t_{QM}^L: The latest arrival time of the data signal at the data output of a latch

$t_{QM}^{L\mathrm{i}}$: The latest arrival time of the data signal at the data output of the latch R_i

t_{Qm}^{L}: The earliest arrival time of the data signal at the data output of a latch

$t_{Qm}^{L\mathrm{i}}$: The earliest arrival time of the data signal at the data output of the latch R_i

3. Flip-flop Parameters

D_{CQ}^{F}: The clock-to-output delay of a latch

$D_{CQ}^{F\mathrm{i}}$: The clock-to-output delay of the latch R_i

D_{CQm}^{F}: The minimum clock-to-output delay of a flip-flop

$D_{CQm}^{F\mathrm{i}}$: The minimum clock-to-output delay of the flip-flop R_i

D_{CQM}^{F}: The maximum clock-to-output delay of a flip-flop

$D_{CQM}^{F\mathrm{i}}$: The maximum clock-to-output delay of the flip-flop R_i

δ_{S}^{F}: The setup time of a flip-flop

$\delta_{S}^{F\mathrm{i}}$: The setup time of the flip-flop R_i

δ_{H}^{F}: The hold time of a flip-flop

$\delta_{H}^{F\mathrm{i}}$: The hold time of the flip-flop R_i

t_{AM}^{F}: The latest arrival time of the data signal at the data input of a flip-flop

$t_{AM}^{F\mathrm{i}}$: The latest arrival time of the data signal at the data input of the flip-flop R_i

t_{Am}^{F}: The earliest arival time of the data signal at the data input of a flip-flop

$t_{Am}^{F\mathrm{i}}$: The earliest arrival time of the data signal at the data input of the flip-flop R_i

t_{QM}^{F}: The latest arrival time of the data signal at the data output of a flip-flop

$t_{QM}^{F\mathrm{i}}$: The latest arival time of the data signal at the data output of the flip-flop R_i

t_{Qm}^{F}: The earliest arrival time of the data signal at the data output of a flip-flop

$t_{Qm}^{F\mathrm{i}}$: The earliest arrival time of the data signal at the data output of the flip-flop R_i

4. Local Data Path Parameters

$\mathsf{R}_\mathsf{i} \rightsquigarrow \mathsf{R}_\mathsf{f}$: A local data path from register R_i to register R_f exists

$\mathsf{R}_\mathsf{i} \not\rightsquigarrow \mathsf{R}_\mathsf{f}$: A local data path from register R_i to register R_f does not exist

Hwang, J. "ROM/PROM/EPROM"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

48

ROM/PROM/EPROM

Jen-Sheng Hwang
National Science Council

48.1 Introduction

Read-only memory (ROM) is the densest form of semiconductor memory, which is used for the applications such as video game software, laser printer fonts, dictionary data in word processors, and sound-source data in electronic musical instruments.

The ROM market segment grew well through the first half of the 1990s, closely coinciding with a jump in personal computer (PC) sales and other consumer-oriented electronic systems, as shown in Fig. 48.1.[1] Because a very large ROM application base (video games) moved toward compact disc ROM-based systems (CD-ROM), the ROM market segment declined. However, greater functionality memory products have become relatively cost-competitive with ROM. It is believed that the ROM market will continue to grow moderately through the year 2003.

48.2 ROM

Read-only memories (ROMs) consist of an array of core cells whose contents or state is preprogrammed by using the presence or absence of a single transistor as the storage mechanism during the fabrication process. The contents of the memory are therefore maintained indefinitely regardless of the previous history of the device and/or the previous state of the power supply.

Core Cells

A binary core cell stores binary information through the presence or absenc of a single transistor at the intersection of the wordline and bitline. ROM core cells can be connected in two possible ways: a parallel NOR array of cells or a series NAND array of cells each requiring one transistor per storage cell. In this case, either connecting or disconnecting the drain connection from the bitline programs the ROM cell. The NOR array is larger as there is potentially one drain contact per transistor (or per cell) made to each bitline. Potentially, the NOR array is faster as there are no serially connected transistors as in the NAND array approach. However, the NAND array is much more compact as no contacts are required within the array itself. However, the serially connected pull-down transistors that comprise the bitline are potentially very slow.[2]

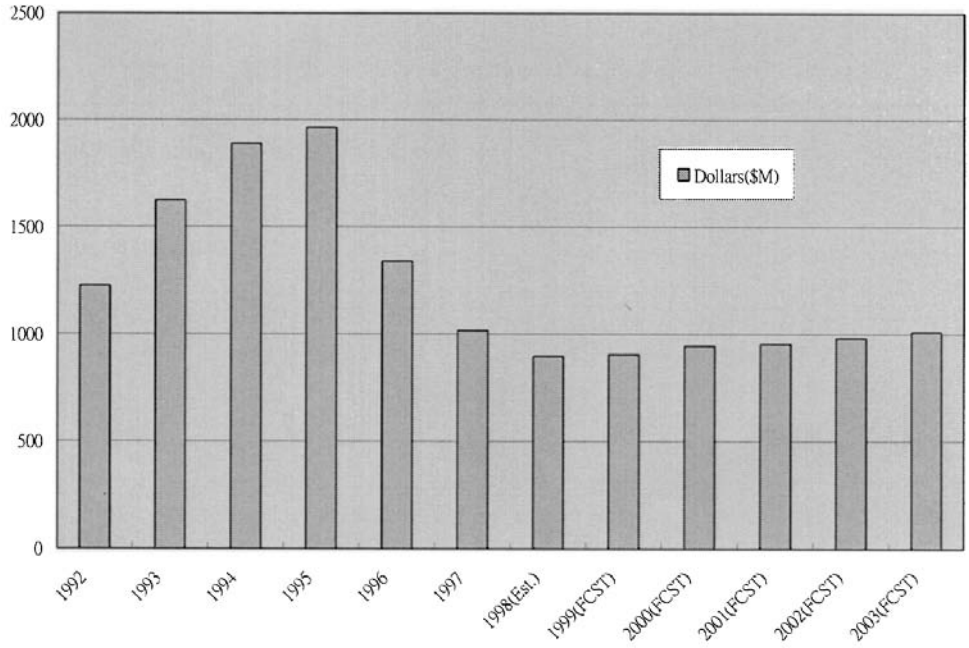


FIGURE 48.1 The ROM market growth and forecast.

Encoding multiple-valued data in the memory array involves a one-to-one mapping of logic value to transistor characteristics at each memory location and can be implemented in two ways:

(i) adjust the width-to-length (W/L) ratios of the transistors in the core cells of the memory array; or
(ii) adjust the threshold voltage of the transistors in the core cells of the memory array.[3]

The first technique works on the principle that W/L ratio of a transistor determines the amount of current that can flow through the device (i.e., the transconductance). This current can be measured to determine the size of the device at the selected location and hence the logic value stored at this location. In order to store 2 bits per cell, one would use one of four discrete transistor sizes. Intel Corp. used this technique in the early 1980s to implement high-density look-up tables in its i8087 math co-processor. Motorola Inc. also introduced a four-state ROM cell with an unusual transistor geometry that had variable W/L devices. The conceptual electrical schematic of the memory cell along with the surrounding peripheral circuitry is shown in Fig. 48.2.[2]

Peripheral Circuitry

The four states in a two-bit per cell ROM are four distinct current levels. There are two primary techniques to determine which of the four possible current levels an addressed cell generates. One technique compares the current generated by a selected memory cell against three reference cells using three separate sense amplifiers. The reference cells are transistors with W/L ratios that fall in between the four possible standard transistor sizes found in the memory array as illustrated in Fig. 48.3.[2]

The approach is essentially a two-bit flash analog-to-digital (A/D) converter. An alternate method for reading a two-bit per cell device is to compute the time it takes for a linearly rising voltage to match the output voltage of the cell. This time interval then can be mapped to the equivalent two-bit binary code corresponding the memory contents.

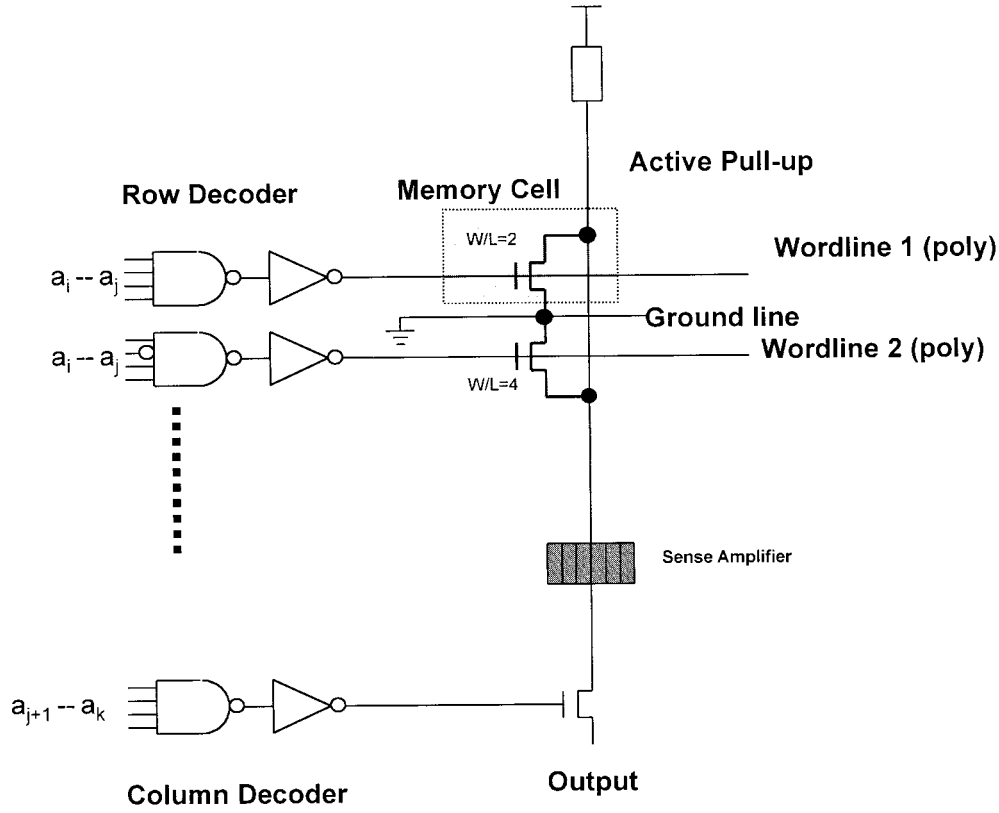


FIGURE 48.2 Geometry-variable multiple-valued NOR ROM.

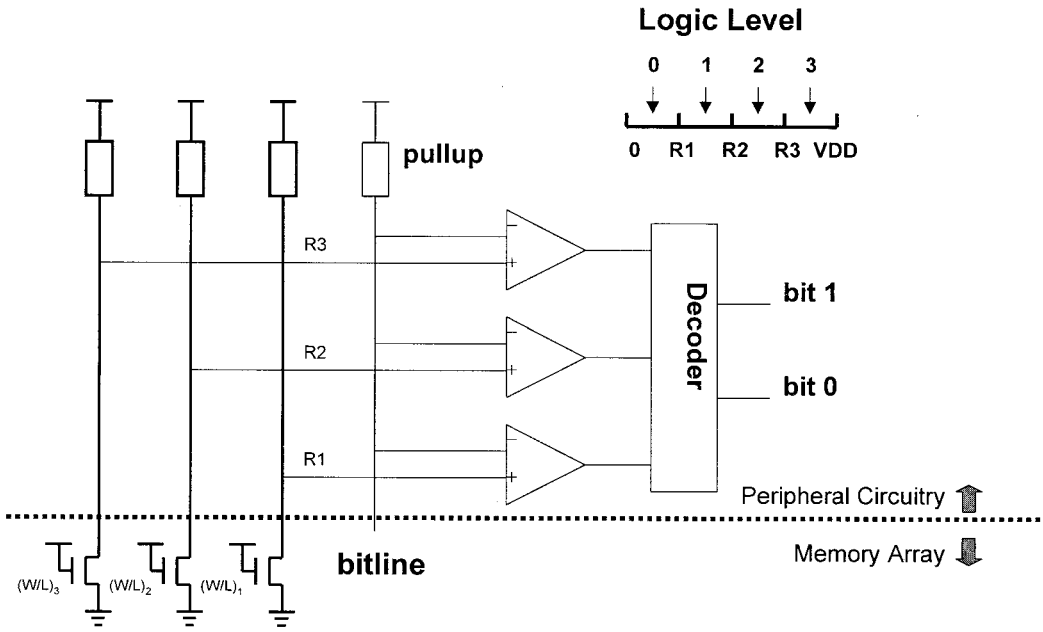


FIGURE 48.3 ROM sense amplifier.

Architecture

Constructing large ROMs with fast access times requires the memory array to be divided into smaller memory banks. This gives rise to the concept of divided wordlines and divided bitlines that reduces the capacitance of these structures allowing for faster signal dynamics. Typically, memory blocks would be no larger than 256 rows by 256 columns. In order to quantitatively compare the area advantage of the multiple-valued approach, one can calculate the area per bit of a two-bit per cell ROM divided by the area per bit of a one-bit per cell ROM. Ideally, one would expect this ratio to be 0.5. In the case of a practical two-bit per cell ROM,[4] the ratio is 0.6 since the cell is larger than a regular ROM cell in order to accommodate any one of the four possible size transistors. ROM density in the Mb capacity range is in general very comparable to that of DRAM density despite the differences in fabrication technology.[2]

In user-programmable or field-programmable ROMs, the customer can program the contents of the memory array by blowing selected fuses (i.e., physically altering them) on the silicon substrate. This allows for a "one-time" customization after the ICs have been fabricated. The quest for a memory that is nonvolatile and electrically alterable has led to the development of EPROMs, EEPROMs, and flash memories.[2]

48.3 PROM

Since process technology has shifted to QLM or PLM to achieve better device performance, it is important to develop a ROM technology that offers short TAT, high density, high speed, and low power. There are many types of ROM each with merits and demerits:[5]

- The diffusion programming ROM has excellent density but has a very long process cycle time.
- The conventional VIA-2 contact programming ROM has better cycle time, but it has poor density.
- An architecture VIA-2 contact programming ROM for QLM and PLM processes has simple processing with high density which obtains excellent results targeting 2.5 V and 2.0 V supply voltage.

Read Only Memory Module Architecture

The details of the ROM module configuration are shown in Fig. 48.4. This ROM has a single access mode (16-bit data read from half of ROM array) and a dual access mode (32-bit data read from both

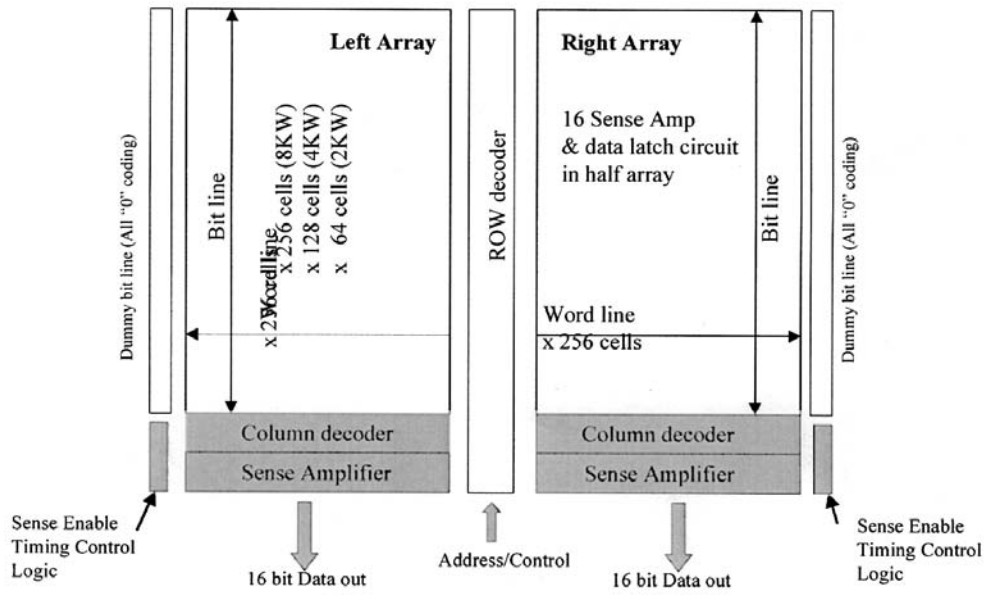


FIGURE 48.4 ROM module array configuration.

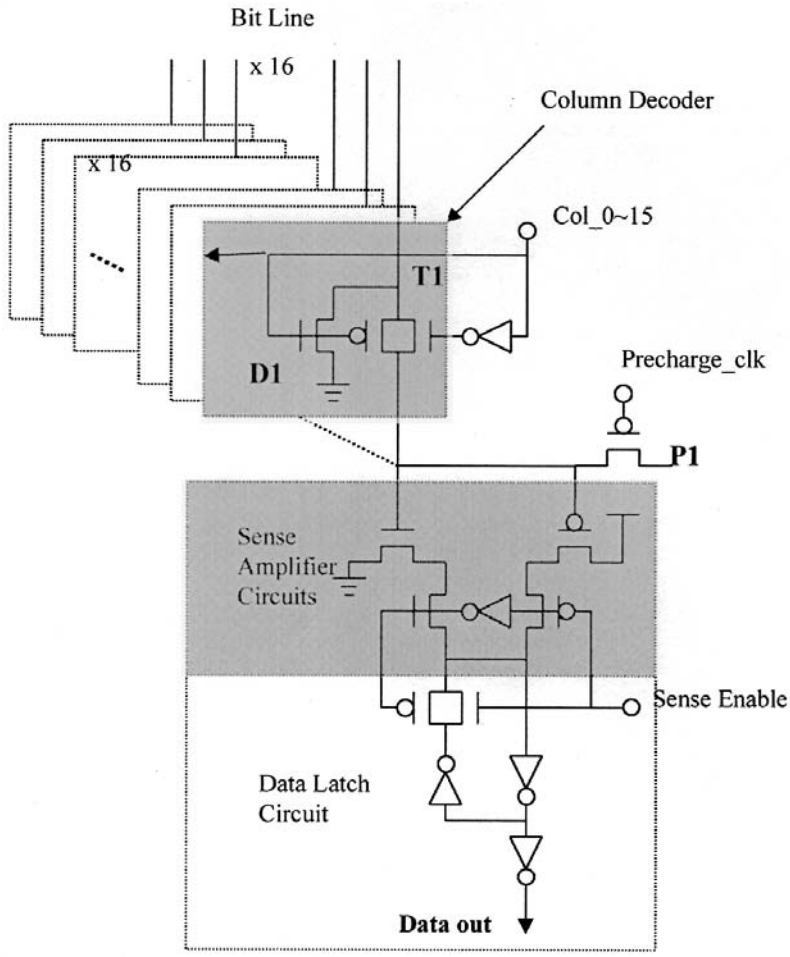


FIGURE 48.5 Detail of low power selective bit line precharge and sense amplifier circuits.

ROM array) with external address and control signals. One block in the array contains 16-bit lines and is connected to a sense amplifier circuit as shown in Fig. 48.5. In the decoder, only one bit line in 16 bits is selected and precharged by P1 and T1.[5]

16 bits in half array at a single access mode or 32 bits in a dual access mode are dynamically precharged to VDD level. Dl is a pull down transistor to keep unselected bit lines at ground level. The speed of the ROM will be limited by bit line discharge time in the worst case ROM coding. When connection exists on all of bit lines vertically, total parasitic capacitance Cbs on the bit line by N-diffusions and Cbg will be a maximum. Tills situation is shown in Fig. 48.6a. In the 8KW ROM, 256 bit cells are in the vertical direction, resulting in 256 times of cell bit line capacitance. In this case, discharge time from VDD to GND level is about 6 – 8ns at VDD = 1.66 V and depends on ROM programming type such as diffusion or VIA-2. Short circuit currents in the sense amplifier circuits arc avoided by using a delayed enable signal (Sense Enable). There are dummy bit lines on both sides of the array as indicated in Fig 48.4. This line contains "0" s on all 256 cells and has the longest discharge time. It is used to generate timing for a delayed enable signal that activates the sense amplifier circuits. These circuits were used for all types of ROM to provide a fair comparison of the performance of each type of ROM.[5]

Conventional Diffusion Programming ROM

Diffusion programmed ROM is shown in Fig. 48.6. This ROM has the highest density because bit line contact to discharge transistor can be shared by two-bit cells (as shown in Fig. 48.6). Cell-A in Fig. 48.6a is coding "0" adding diffusion which constructs transistor, but Cell-B is coding "1" which does not have diffusion and resulted in field oxide without transistor as shown in Fig. 48.6c. This ROM requires very long fabrication cycle time since process steps for the diffusion programming are required.[5]

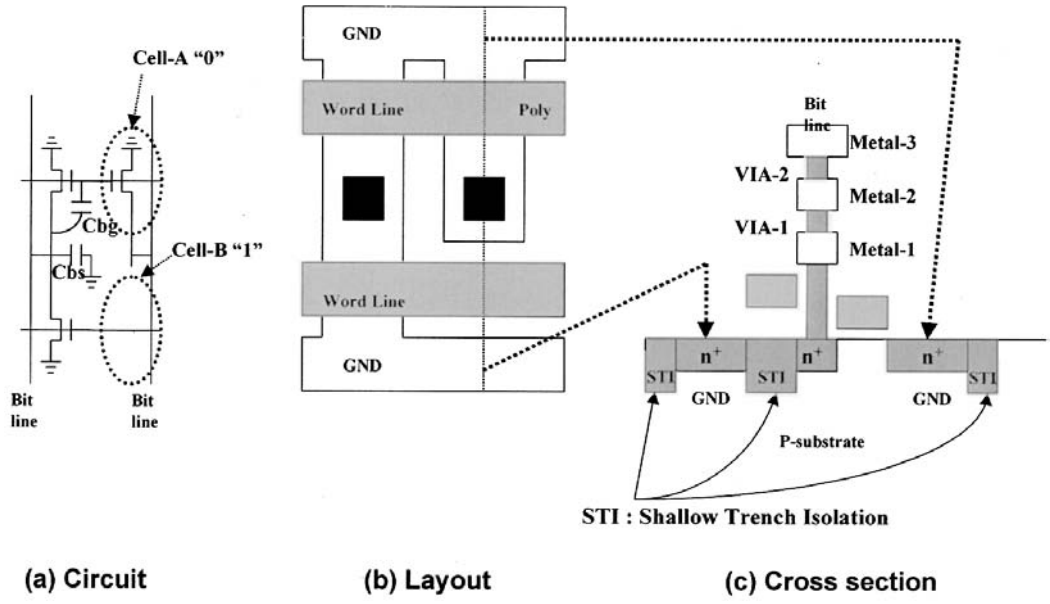


FIGURE 48.6 Diffusion programming ROM.

Conventional VIA-2 Contact Programming ROM

In order to obtain better fabrication cycle time, conventional VIA-2 contact programming ROM was used as shown in Fig. 48.7, Cell-C in Fig. 48.7a is coding "1" Cell-D is coding "1". There are determined by VIA-2 code existence on bit cells. The VIA-2 is final stage of process and base process can be completed just before VIA-2 etching and remaining process steps are quite few. So, VIA-2 ROM fabrication cycle time is about 1/5 of the diffusion ROM. The demerit of VIA-2 contact and other type of contact programming ROM was poor density. Because diffusion area and contact must be separated in each ROM bit cell as shown in Fig. 48.7c, this results in reduced density, speed, and increased power. Metal-4 and VIA-3 at QLM process were used for word line strap in the ROM since RC delay time on these nobles is critical for 100MIPS DSP.[5]

New VIA-2 Contact Programming ROM

The new architecture VIA-2 programming ROM is shown in Fig. 48.8. A complex matrix constructs each 8-bit block with GND on each side. Cell-E in Fig. 48.8a is coding "0". Bit4 and N4 are connected by VIA-2. Cell-F is coding "1" since Bit5 and N5 are disconnected. Coding other bit lines (Bit 0, 1, 2, 3,5, 6, and 7) follow the same procedure. This is one of the coding examples to discuss worst case operating speed. In the layout shown in Fig. 48.8b, the word line transistor is used not only in the active mode but also to isolate each bit line in the inactive mode. When the word line goes high, all transistors are turned on.

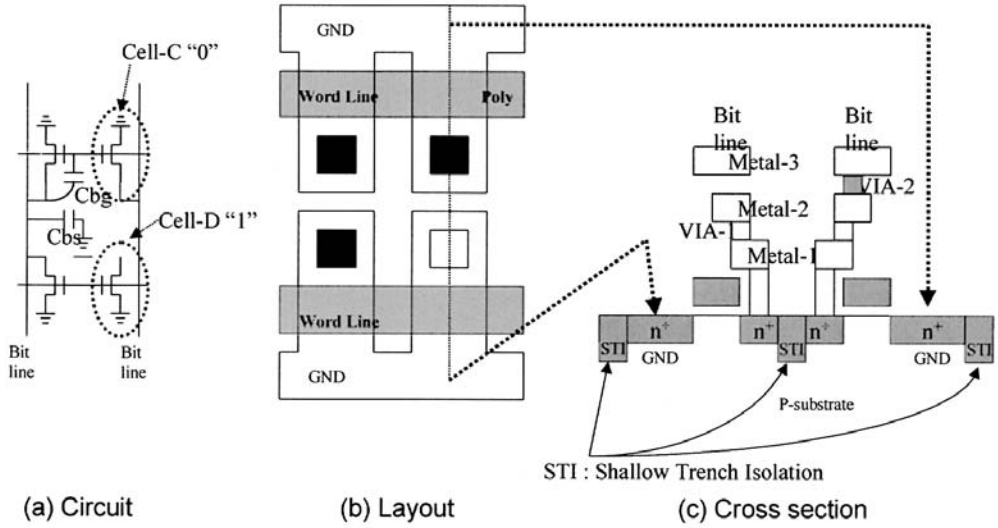


FIGURE 48.7 Conventional VIA-2 programming ROM.

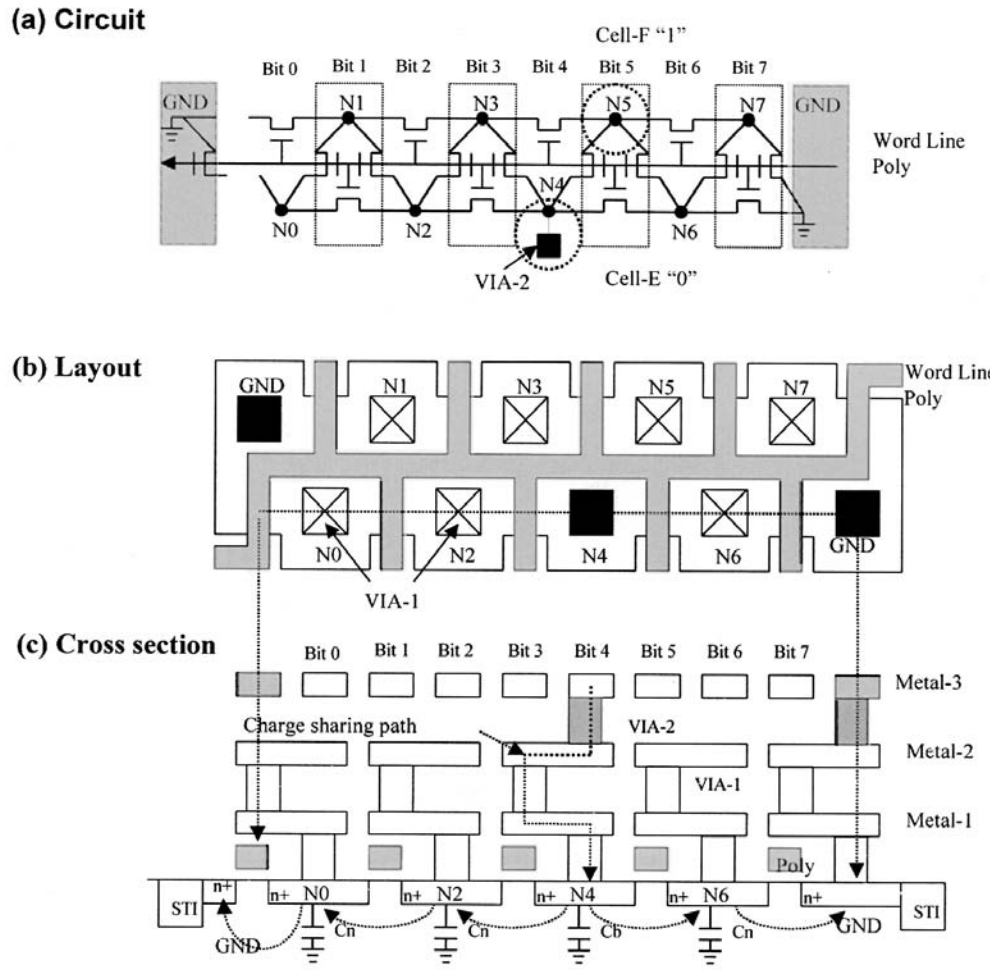


FIGURE 48.8 New VIA-2 programming ROM.

All nodes (N0 - N7) are horizontally connected with respect to GND. If VIA-2 code exists on all or some of nodes (N0 - N7) in the horizontal direction, the discharge time of bit lines is very short since this ROM uses a selective bit fine precharge method.[5]

Figure 48.9 shows timing chart of each key signal and when Bit4 is accessed, for example, only this line will be precharged during precharge phase. However, all other bit lines are pulled down to GND by Dl transistors as shown in Fig. 48.4. When VIA-2 code exists like N4 and Bit4, this line will be discharged. But if it does not exist, this line will stay at VDD level dynamically as described during word line active phase, which is shown in Fig. 48.9. After this operation, valid data appears on data out node of data latch circuits.[5]

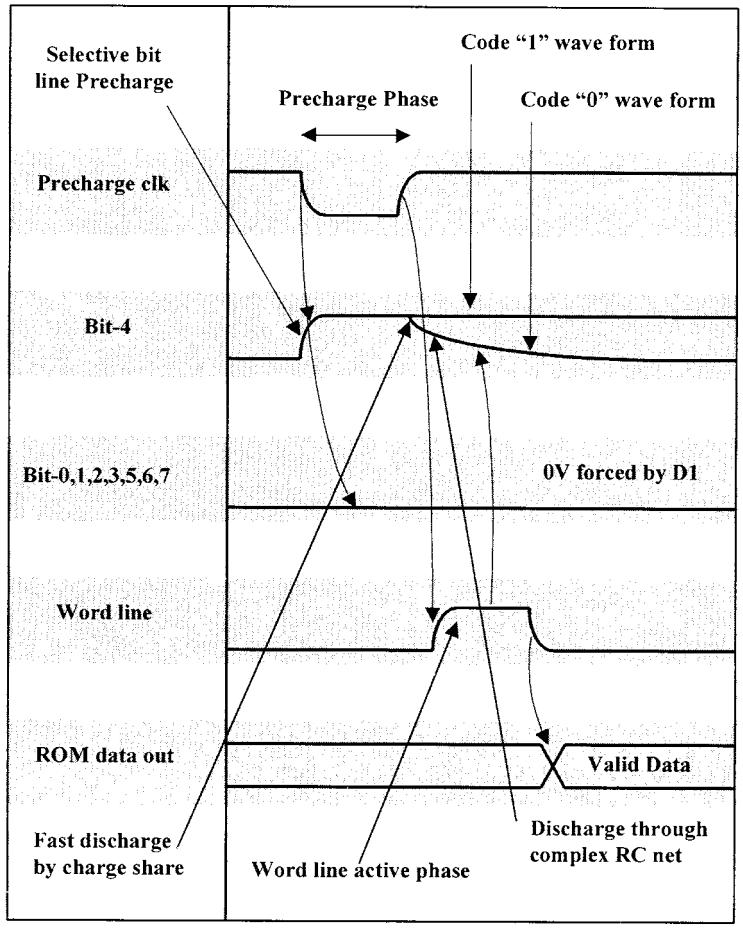


FIGURE 48.9 Timing chart of new VIA-2 programming ROM.

In order to evaluate worst case speed, no VIA-2 coding on horizontal bit cell was used since transistor series resistance at active mode will be maximum with respect to GND. However, in this situation, charge sharing effects and lower transistor resistance during the word line active mode allow fast discharge of bit lines despite the increased parasitic capacitance on bit line to 1.9 times. This is because all other nodes (N0-N7) will stay at GND dynamically. The capacitance ratio between bit line (Cb) and all nodes except N4 (Cn) was about 20:1. Fast voltage drop could be obtained by charge sharing at the initial stage of bit line discharging. About five voltage drop could be obtained on 8KW configuration through the charge-sharing path shown in Fig. 48.8c. With this phenomenon, the full level discharging was mainly determined by complex transistor RC network connected to GND as shown in Fig. 48.8a. This new ROM has much

wider transistor width than conventional ROMs and much smaller speed degradation due to process deviations, because conventional ROMs typically use the minimum allowable transistor size to achieve higher density and are more sensitive due to process variations.[5]

Comparison of ROM Performance

The performance comparison of each type of ROM are listed in Table 48.1. 8KW ROM module area ratio was indicated using same array configuration, and peripheral circuits with layout optimization to achieve fair comparison. The conventional VIA-2 ROM was 20% bigger than diffusion ROM, but new VIA-2 ROM was only 4% bigger. TAT ratio (days for processing) was reduced to 0.2 due to final stage of process steps. SPICE simulations were performed to evaluate each ROM performance considering low voltage applications. The DSP targets 2.5 V and 2.0 V supply voltage as chip specification with low voltage comer at 2.3 V and 1.8 V, respectively. However, a lower voltage was used in SPICE simulations for speed evaluation to account for the expected 7.5 supply voltage reduction due to the IR drop from the external supply voltage on the DSP chip. Based on this assumption, VDD = 2.13 V and VDD = 1.66 V were used for speed evaluation. The speed of new VIA-2 ROM was optimized at 1.66V to get over 100 MHz and demonstrated 106 MHz operation atVDD = 1.66 V, 125dc, (based on typical process models). Additionally, 149 MHz at VDD = 2.13 V, 125dc was demonstrated with the typical model and 123 MHz using the slow model. This is a relatively small deviation induced by changes in process parameters such as width reduction of the transistors. By using the fast model, operation at 294 MHz was demonstrated without any timing problems. This means the new ROM has very high productivity with even three sigma of process deviation and wide range of voltages and temperatures.[5]

TABLE 48.1 Comparison of ROM Performance

Comparison Item	Diffusion ROM	Conventional VIA-2 ROM	New VIA-2 ROM
8KW (Area ratio)	1.0	1.2	1.04
TAT (Day ratio)	1.0	0.2	0.2
Speed @ 2,13 V, 125dc. Weak.	83 MHz	86 MHz	123 MHz
Speed @ 2.13 V, 125dc. Typical.	166 MHz	98M Hz	149 MHz
Speed @ 2.81 V, -40dc. Strong.	277 MHz	179 MHz	294 MHz
Speed @ 1.66 V. 125dc, Typical.	103 MHz	75 MHz	106 MHz
Power@2.81 V,-40dc. Strong. 100 MHz. (16-bit single access)	15.6 mW	19.3 mW	2 UrnW
Power@2.81 V@40dc. Strong. 100 MHz. (32-bit dual access)	29.6 mW	37.1 mW	401 mW

Performance was measured with worst coding (all coding "1").

References

1. Karls, J., *Status 1999: A Report On The Integrated Circuit Industry*, Integrated Circuit Engineering Corporation, 1999.
2. Gulak, P. G., A review of multiple-valued memory technology, IEEE International Symposium on Multi-valued Logic, 1998
3. Rich, D.A., A Survey of Multi valued memories, *IEEE Trans. On Comput.*, vol. C-35, no. 2, pp. 99-106, Feb. 1986.
4. Prince, B., *Semiconductor Memories*, 2nd ed., John Wiley & Sons Ltd., New York, 1991.
5. Takahashi, H., Muramatsu, S., and Itoigawa, M., A new contact programming ROM architecture for digital signal processor, Symposium on VLSI Circuits, 1998.

Tseng, Y. "SRAM"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

49 SRAM

Yuh-Kuang Tseng
Industrial Research and Technology Institute

49.1 Read/Write Operation

Figure 49.1 shows a simplified readout circuit for an SRAM. The circuit has static bit-line loads composed of pull-up PMOS devices M1 and M2. The bit-lines are pulled up to VDD by bit-line load transistors M1 and M2. During the read cycle, one word-line is selected. The bit line BL is discharged to a level determined by the bit-line load transistor M1, the accessed transistor N1, and the driver transistor N2 as shown in Fig. 49.1(b). At this time, all selected memory cells consume a dc column current flowing through the bit-line load transistors, accessed transistors, and driver transistors. This current flow increases the operating power and decreases the access speed of the memory.

Figure 49.2 shows a simplified circuit diagram for SRAM write operation. During the write cycle, the input data and its complement are placed on the bit-lines. Then the word-line is activated. This will force the memory cell to flip into the state represented on the bit-lines, whereas the new data is stored in the memory cell. The write operation can be described as follows. Consider a high voltage level and a low voltage level are stored in both node 1 and node 2, respectively. If the data is to be written into the cell, then node 1 becomes low and node 2 becomes high. During this write cycle, a dc current will flow from VDD through bit-line load transistor M1 and write circuits to ground. This extra dc current flow in write cycle increases the power consumption and degrades the write speed performance. Moreover, in the tail portion of write cycle, if data 0 has been written into node 1 as shown in Fig. 49.2, the turn-on word-line transistor N1 and driver transistor N2 form a discharge circuit path to discharge the bit-line voltage. Thus, the write recovery time is increased. In high-speed SRAM, write recovery time is an important component of the write cycle time. It is defined as the time necessary to recover from the write cycle to the read state after the WE signal is disabled.[1] During the write recovery period, the selected cell is in the quasi-read condition,[2] which consumes dc current as in the case of read cycle.

Based on the above discussion, the dc current problems that occur in the read and write cycles should be overcome to reduce power dissipation and improve speed performance. Some solutions for the dc current problems of conventional SRAM will be described. During the active mode (read cycle or write cycle), the word-line is activated, and all selected columns consume a dc current. Thus, the word-line activation duration should be shortened to reduce the power consumption and improve speed performance during the active mode. This is possible by using the Address Transition Detection (ATD) technique[3] to generate the pulsed word-line signal with enough time to achieve the read and write operation, as shown in Fig. 49.3.

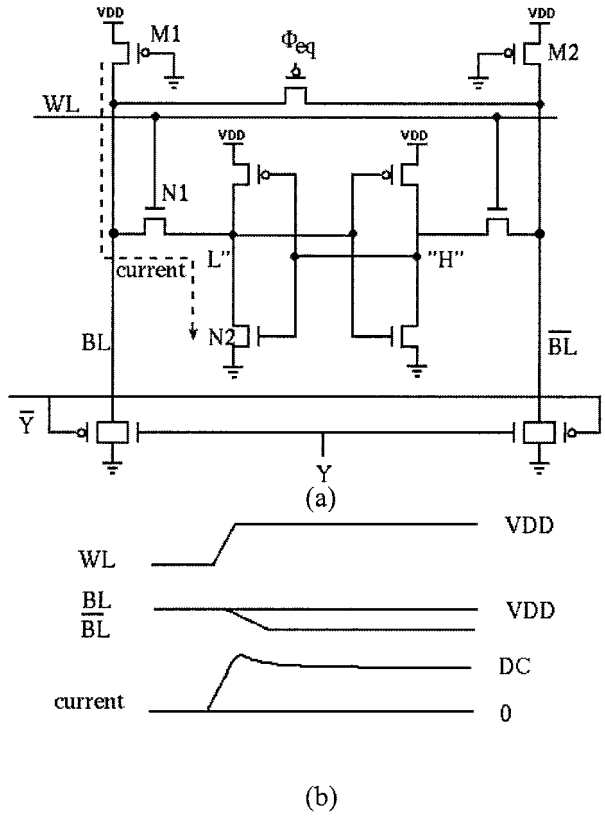


FIGURE 49.1 (a) Simplified readout circuit for an SRAM, (b) signal waveform.

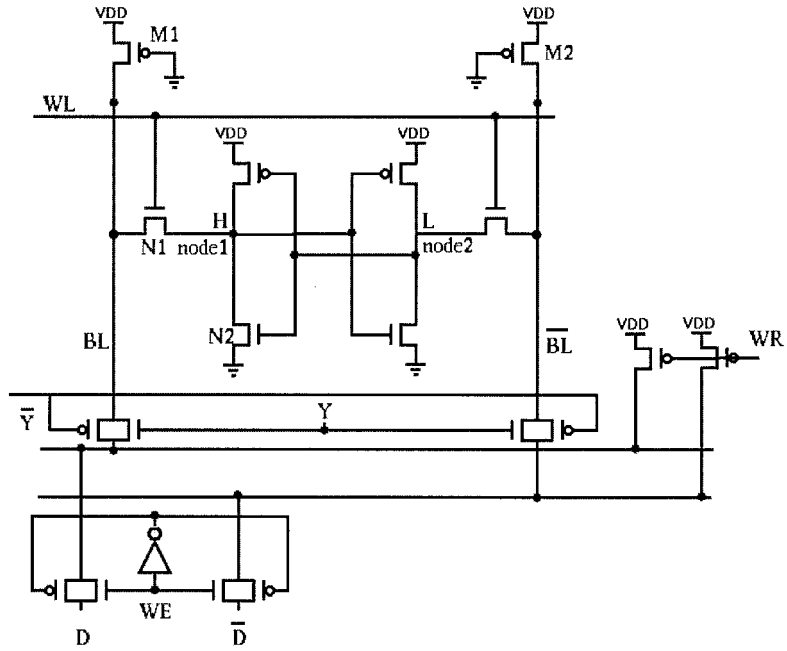


FIGURE 49.2 Simplified circuit diagram for SRAM write operation

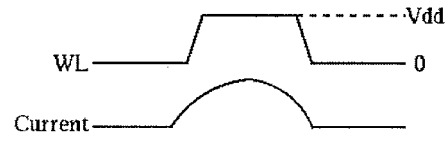


FIGURE 49.3 Word-line signal and current reduction by pulsing the word line

However, the memory cells asserted by the pulsed word-line signal still consume dc current from VDD through bit-line load transistors, accessed transistors, and driver transistors or write circuits to the ground during the word-line activation period. A dynamic bit-line loads circuit technique[2,4-6] can be used to eliminate the dc power consumption during operation period.

Figure 49.4 shows a simplified circuit configuration and time diagram for read and write operation. In the read cycle, the bit-line load transistors are turned off because the Φ_{LD} signal is in the high state. The bit-line load consists of only the stray capacitance. Therefore, the selected memory cell can rapidly drive the bit-line load, resulting in a fast access time. Moreover, the dc column current consumed by the other activated memory cells can be eliminated. Similarly, the dc current consumption in the write cycle can be eliminated.

A memory cell's readout current I_{cell} depends on the channel conductance of the transfer gates in a memory cell. As the supply voltage is scaled down, the speed performance of SRAM is decreased, significantly, due to small cell's readout current. To increase the channel conductance, widening the channel width and/or boosting word-line voltage are used. For low-voltage operation, boosting the word-line voltage is effective in shortening the delay time, in contrast to widening the channel width. However, this causes an increased power dissipation and a large transition time due to enhanced bit-line swing. To solve these problems, a step-down boosted-word-line scheme that shortens the readout time with little power dissipation penalty was reported by Morimura and Shibata in 1998.[7]

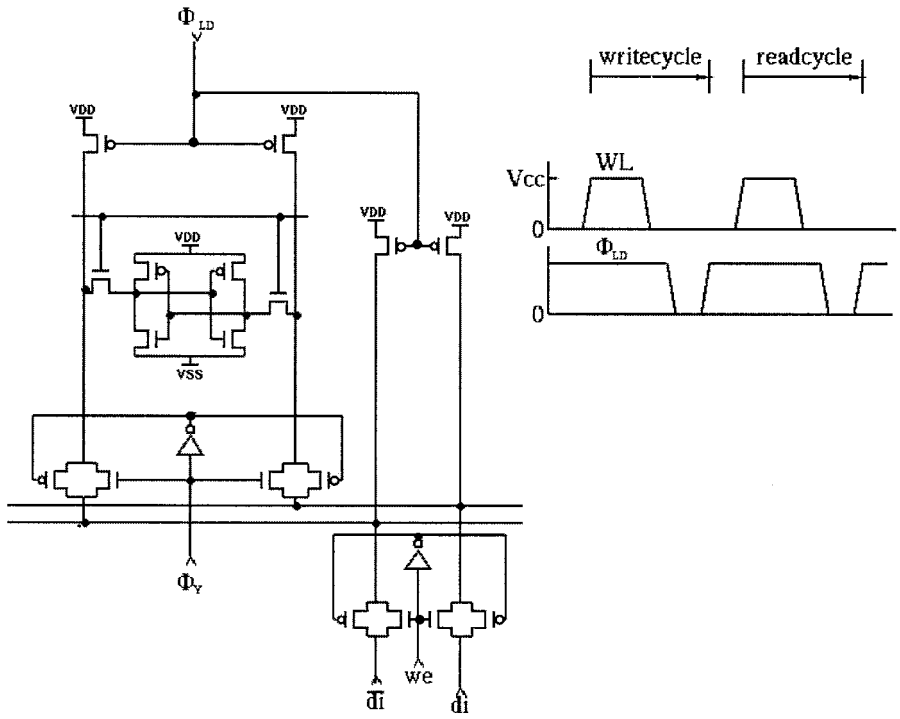


FIGURE 49.4 Simplified circuit configuration and time diagram for read and write operation.

The concept of this scheme is shown in Fig. 49.5(b), in contrast to the conventional full-bootsted-word-line scheme in Fig. 49.5(a). The step-down boosted-word-line scheme also boosts the selected word-line, but the boosted period is restricted only at the beginning of memory cell access. This enables the sensing operation to start early, by fast bit-line transition. During the sensing period of bit-line signals, the word-line potential is stepped down to the supply voltage to suppress the power dissipation; the reduced bit-line signals are sufficient to read out data by current sensing, and the reduced bit-line swing is effective in shortening the bit-line transition time in the next read cycle (Fig. 49.5(c)). As a result, fast readout is accomplished with little dissipation penalty (Fig. 49.5(d)).

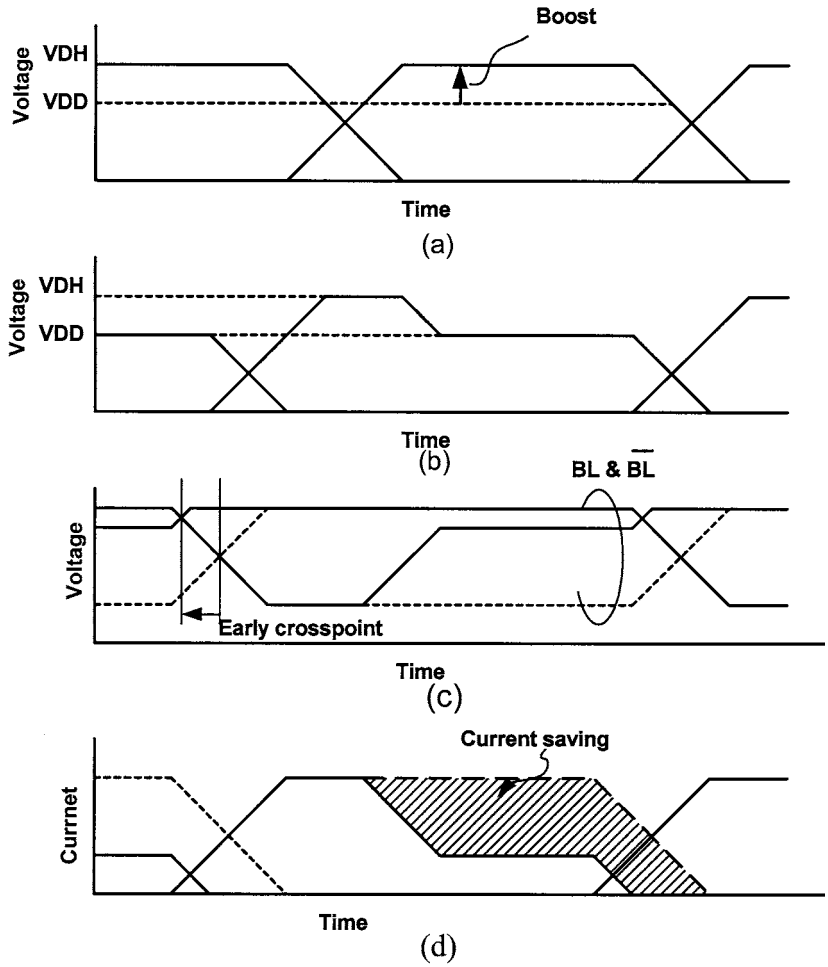


FIGURE 49.5 Step-down boosted-word-line scheme: (a) conventional boosted word-line, (b) step-down boosted word-line, (c) bit-line transition, and (d) current consumption of a selected memory cell. (From Ref. 7.)

The step-down boosted-word-line scheme is also used in data writing. In the writing cycle, the proposed scheme is just as effective in reducing the memory-cell current because the memory cells unselected by column-address signals consume the same power as in the read cycle. The boosted word-line voltage shortens the time for writing data because it increases the channel conductance of the access transistor in the selected memory cells. The writing recovery operation starts after the word-line voltage is stepped down. Reducing the memory cell's current accelerates the recovery operation of lower bit-lines. So, a shorter recovery time than that of the conventional full-boosted-word-line scheme is obtained.

Other circuit techniques for dc column current reduction, such as divided word-line (DWL)[8] and hierarchical word decoding (HWD)[9] structures will be described in the following sections.

49.2 Address Transition Detection (ATD) Circuit for Synchronous Internal Operation[1,10]

The address transition detection (ATD) circuit plays an important role in achieving internal synchronization of operation in SRAM. ATD pulses can be used to generate the different time signals for pulsing word-lines, sensing amplifier, and bit-line equalization. The ATD pulse activating $\phi_{(ai)}$ is generated with XOR circuits by detecting "L" to "H" or "H" to "L" transitions of any input address signal a_i, as shown in Fig. 49.6. All the ATD pulses generated from all the address input transitions are summed up to one pulse, ϕ_{ATD} as shown in Fig. 49.6. The pulse width of ϕ_{ATD}, is controlled by the delay element τ. The pulse width is usually stretched out with a delay circuit and used to reduce or speed up signal propagation in the SRAM.

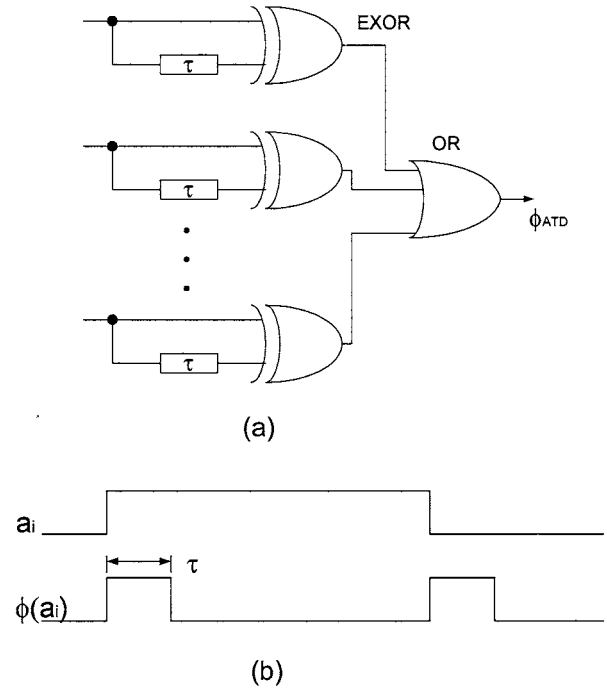


FIGURE 49.6 (a) Summation circuit of all ATD pulses generated from all address transitions (b) ATD pulse waveform. (From Ref. 10.)

49.3 Decoder and Word-Line Decoding Circuit[10-13]

Two kinds of decoders are used in SRAM: the row decoder and the column decoder. Row decoders are needed to select one row of word-lines out of a set of rows in the array. A fast decoder can be implemented by using AND/NAND and OR/NOR gates. Figure 49.7 shows the schematic diagrams of static and dynamic AND gate decoders. The static NAND-type structure is chosen due to its low power consumption, that is, only the decoded row transitions. The dynamic structure is chosen due to its speed and power improvement over conventional static NAND gates.

From a low-voltage operation standpoint, a dynamic NOR-base decoding would provide lower delay times through the decoder due to the limited amount of stacking of devices. Figure 49.8 shows circuit

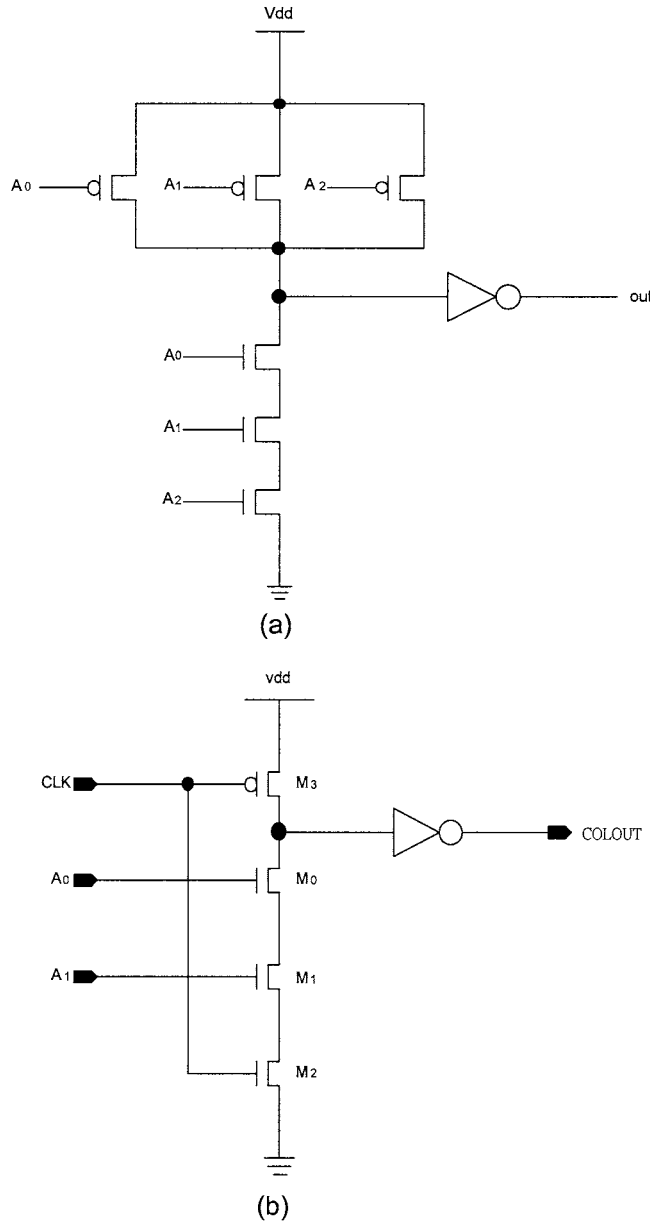


FIGURE 49.7 Circuit diagrams of a three-input AND gate: (a) static CMOS, (b) dynamic CMOS.

diagrams of dynamic NOR gates. The dynamic CMOS gate as shown in Fig. 49.8(a) consists of input-NMOSs whose drain nodes are precharged to a high level by a PMOS when a clock signal Φ is at a low level, and conditionally discharged by the input-NMOSs when a clock signal Φ is at a high level. The delay time of the dynamic NOR/OR gate does not increase when the number of input signals increases. This is because only one PMOS and two NMOSs are connected in series, even if the number of input signals is large. However, the output of the OR signal is slower than that of the NOR signal because the OR signal is generated from the inverter driven by the NOR signal.

Figure 49.8 (b) shows the source-coupled-logic (SCL)[11] NOR/OR circuit. When a clock signal Φ is at a low level, the drain nodes of the NMOS (N1, N2) are precharged to a high level in the circuit. If at

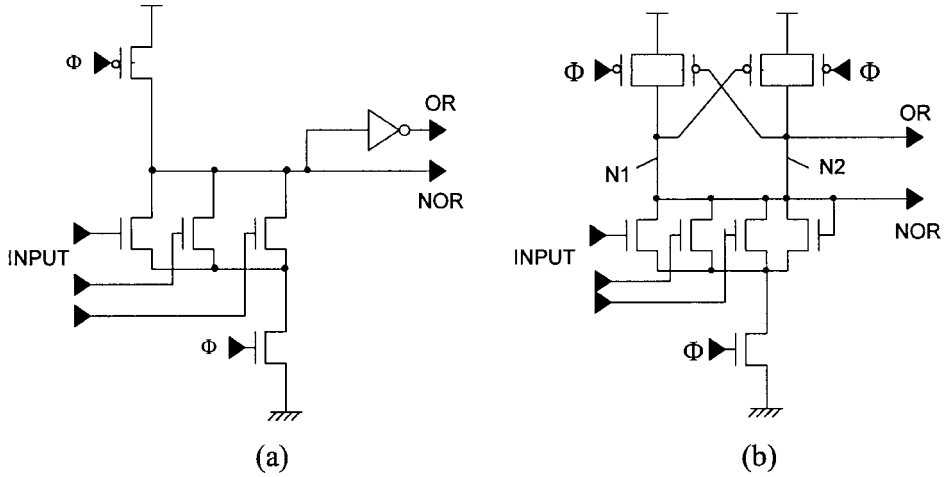


FIGURE 49.8 Circuit diagrams of three-input NOR/OR gates: (a) dynamic CMOS, (b) SCL> (From Ref. 11)

least one of input signals of the circuit is at a high level and the clock Φ then turns to a high level, node N1 is discharged to a low level and node N2 remains at a high level. On the other hand, if all the input signals are at a low level and Φ then turns to a high level, node N2 is discharged and node N1 remains at a high level. The SCL circuit can produce an OR signal and a NOR signal simultaneously. Thus, the SCL circuit is suitable for predecoders that have a large number of input signals and for address buffers that need to produce OR and NOR signals simultaneously.

Column decoders select the desired bit pairs out of the sets of bit pairs in the selected row. A typical dynamic AND gate decoder as shown in Fig. 49.7(b) can be used for column decoding because the AND structure meets the delay requirements (column decode is not in the worst-case delay path) and does so at a much lower power consumption.

A highly integrated SRAM adopts a multi-divided memory cell array structure to achieve high-speed word decoding and reduce column power dissipation. For this purpose, many high-speed word-decoding circuit architectures have been proposed, such as divided word-line (DWL)[8] and hierarchical word decoding (HWD)[9] structures. The multi-stage decoder circuit technique is adopted in both word-decoding circuit structures to achieve high-speed and low-power operation. The multi-stage decoder circuit has advantages over the one-stage decoder in reducing the number of transistors and fanin. Also, it reduces the loading on the address input buffers. Figure 49.9 shows the decoder structure for a typical partitioned memory array with divided word-line (DWL). The cell array is divided into N_B blocks. If the SRAM has N_C columns, each block contains N_C/N_B columns. The divided word-line in each block is activated by the global word-line and the vertical block select line. Consequently, only the memory cells connected to one divided word-line within a selected block are accessed in a cycle. Hence, the column current is reduced because only the selected columns switch. Moreover, the word-line selection delay, which is the sum of the global word-line delay and the divided word-line delay, is reduced. This is because the total capacitance of the global word-line is smaller than that of a conventional word-line. The delay time of each divided word-line is small due to the short length. In the block decoder, an additional signal Φ, which is generated from an ATD pulse generator, can be adopted to enable the decoder and ensure the pulse activated word-line.

However, in high-density SRAM, with a capacity of more than 4 Mb, the number of blocks in the DWL structure will have to increase. Therefore, the capacitance of the global word-line will increase and that causes the delay and power to increase. To solve this problem, the hierarchical word decoding (HWD)[9] circuit structure, as shown in Fig. 49.10, was proposed. The word-line is divided into multi-levels. The number of levels is determined by the total capacitance of the word select line to efficiently distribute it.

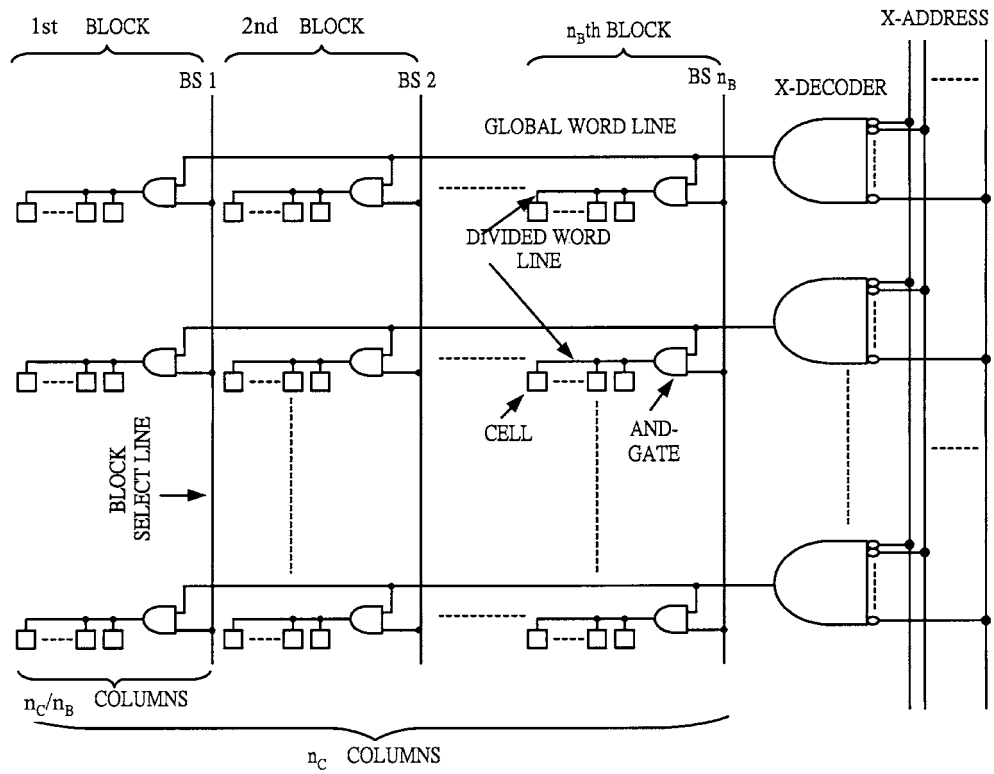


FIGURE 49.9 Divided word-line (DWL) structure. (From Ref. 8.)

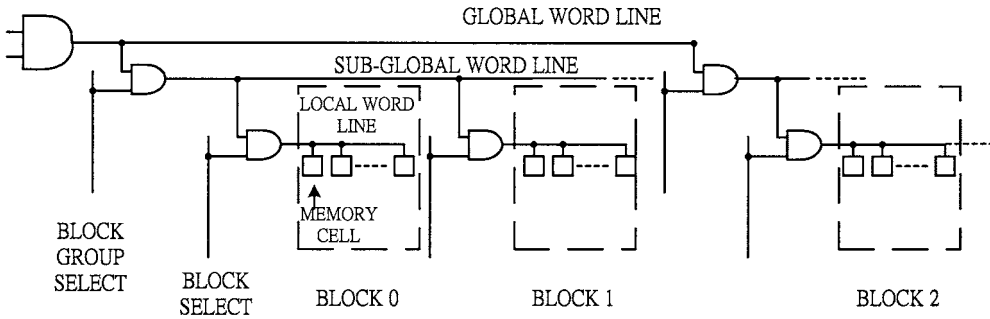


FIGURE 49.10 Hierarchical word decoding structure.

Hence, the delay and power are reduced. Figure 49.11 shows the delay time and the total capacitance of the word decoding path comparison for the optimized DWL and HWD structures of 256-Kb, 1-Mb, and 4-Mb SRAMs.

49.4 Sense Amplifier[10]

During the read cycle, the bit-lines are initially precharged by bit-line load transistors. When the selected word-line is activated, one of the two bit-lines is pulled low by driver transistor, while the other stays high. The bit-line pull-down speed is very slow due to the small cell size and large bit-line load capacitance. Differential sense amplifiers are used for speed purposes because they can detect and amplify a very small

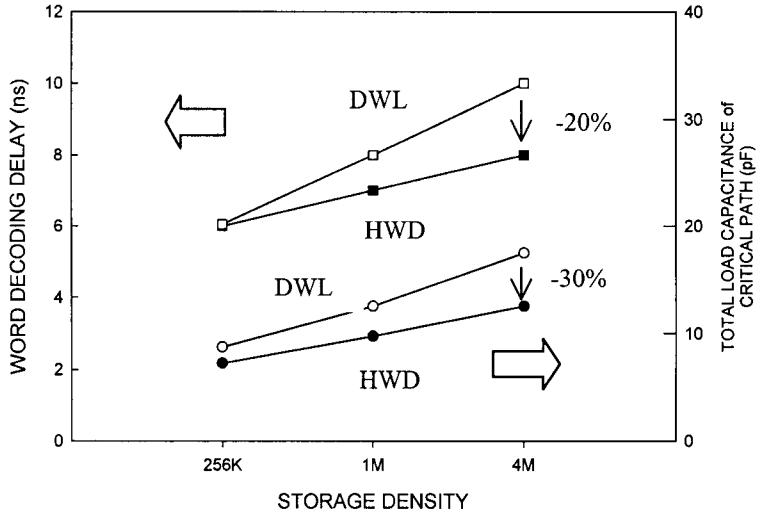


FIGURE 49.11 Comparison of DWL and HWD. (From Ref. 9. With permission.)

level difference between two bit-lines. Thus, a fast sense amplifier is an important factor in realizing fast access time.

Figure 49.12 shows a switching scheme of well-known current-mirror sense amplifiers.[14] Two amplifiers are serially connected to obtain a full supply voltage swing output because one stage of the amplifier

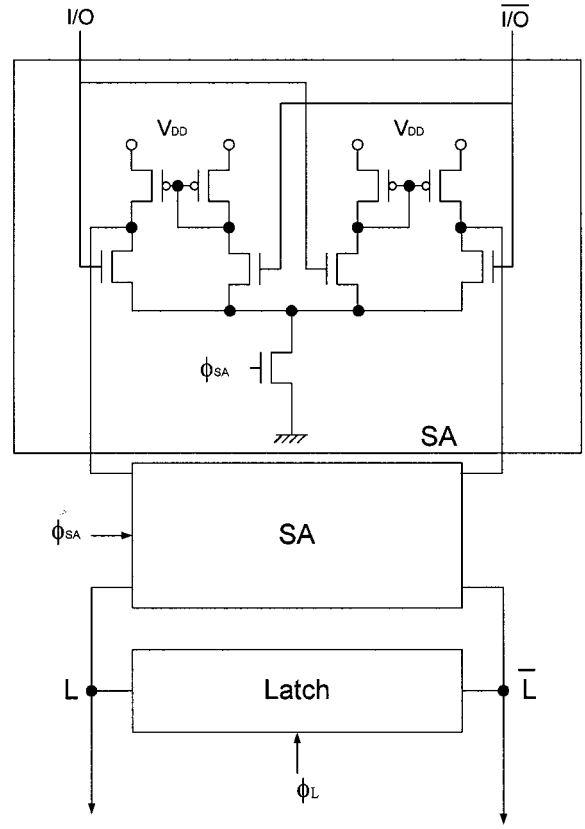


FIGURE 49.12 Two-stage current-mirror sense amplifier. (From Refs. 10 and 14. With permission.)

does not provide enough gain for a full swing. The signal Φ_{SA} is generated with an ATD pulse. It is asserted for a period of time, enough to amplify the small difference on data lines; then it is deactivated and the amplified output is latched. Hence, the switch reduces the power consumption, especially at relatively low frequencies.

A latch-type sense amplifier such as a PMOS cross-coupled amplifier,[15] as shown in Fig. 49.13, greatly reduces the dc current after amplification and latching, because the amplifier provides a nearly full supply voltage swing with positive feedback of outputs to PMOSFETs. As a result, the current in the PMOS cross-coupled sense amplifier is less than one fifth of that in a current-mirror amplifier. Moreover, this positive feedback effect gives much faster sensing speed than the conventional amplifier. To obtain correct and fast operation, the equalization element EQL is connected between the output terminals and are turned on with pulse signals Φ_S and its complement during the transition period of the input signals.

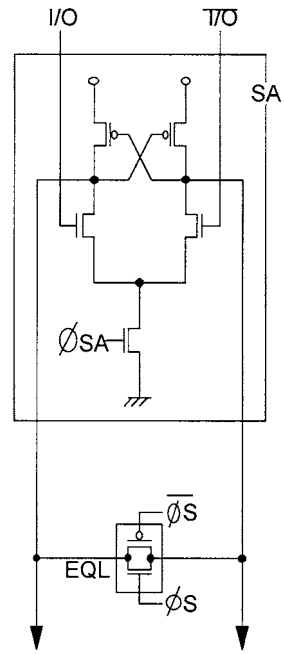


FIGURE 49.13 PMOS cross-coupled amplifier. (From Ref. 15. With permission.)

However, the latch-type sense amplifier has a large dependence on the input voltage swing, especially at low current operation conditions. An NMOS source-controlled latched sense amplifier[16] as shown in Fig. 49.14 is able to quickly amplify an input voltage swing as small as 10 mV. The sense amplifier consists of two PMOS loads, two NMOS drivers, and two feedback inverters. The sense amplifier control (SAC) signal is driven by the CS input buffer, and Φ_S is a sense-amplifier equalizing pulse generated by the ATD pulse. The gate terminal of the NMOS driver is connected to the local data bus (LD1 and LD2), and the source terminal of the NMOS driver is controlled by the feedback inverter connected to the opposite output node of sense amplifier. Thus, the NMOS driver connected to the high-going output node turns off immediately. Therefore, the charge-up time of that node can be reduced because no current is wasted in the NMOS driver.

A bidirectional sense amplifier, called a bidirectional read/write shared sense amplifier (BSA),[17] is shown in Fig. 49.15. The BSA plays three roles. It functions as a sense amplifier for read operations, and it serves as a write circuit and a data input buffer for write operations. It consists of an 8-to-1 column selector and bit-line precharger, a CMOS dynamic sense amplifier, an SR flip-flop, and an I/O circuit.

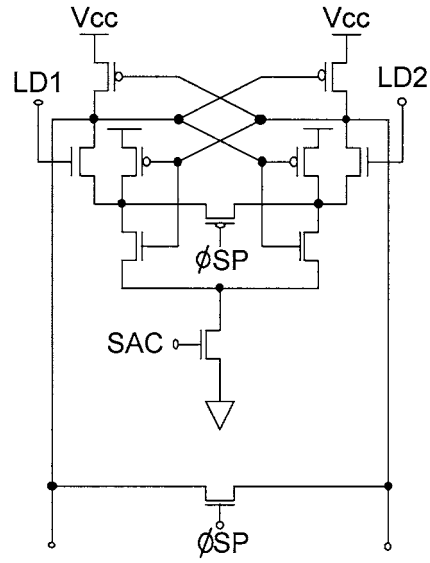


FIGURE 49.14 NMOS source-controlled latched sense amplifier. (From Ref. 16. With permission.)

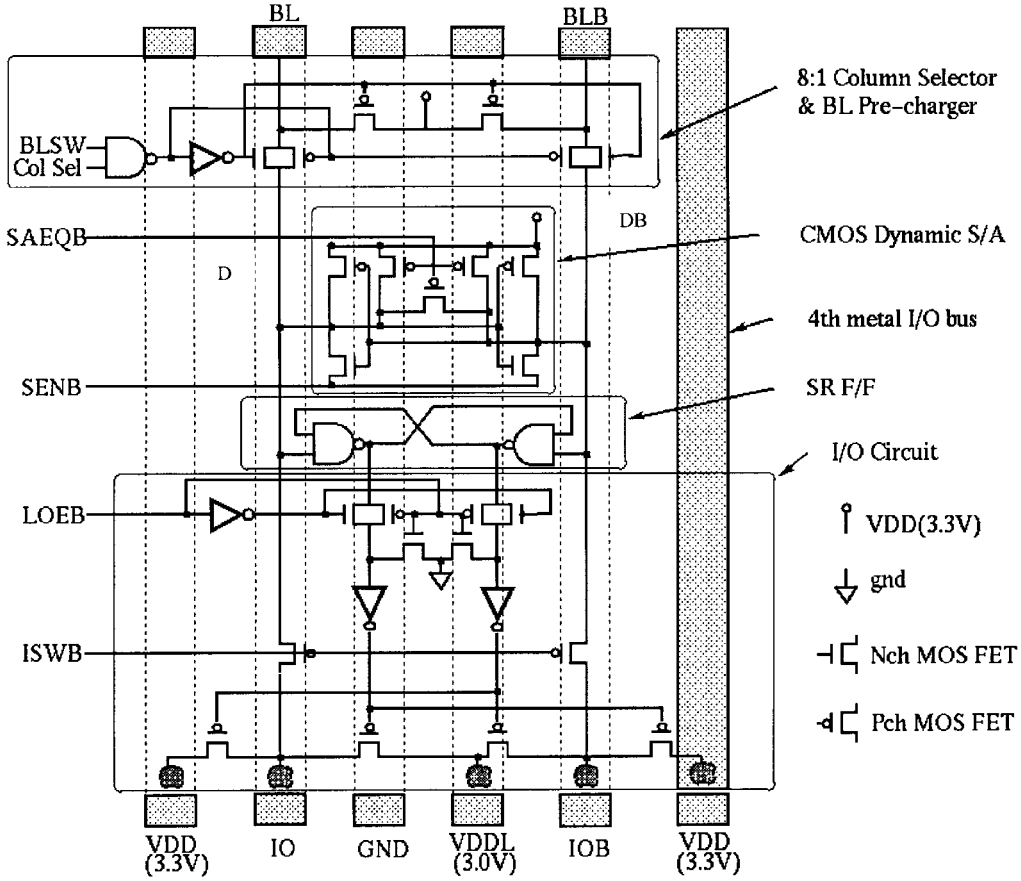


FIGURE 49.15 Schematic diagram of BSA. (From Ref. 17. With permission.)

Eight bit-line pairs are connected to a CMOS dynamic sense amplifier through CMOS transfer gates. The BLSW signal is used to select a column and to precharge bit-lines. When the BLSW signal is high, one of eight bit-line pairs is connected to the sense amplifier. When the BLSW signal is low, all bit-line pairs are precharged to VDD level. The SAEQB signal controls the sense amplifier equalization. When the SAEQB signal is low, sense nodes D and DB are equalized and precharged to the VDD level. The SENB signal activates the CMOS dynamic sense amplifier. The SR flip-flop holds the result. The output circuit consists of four p-channel transistors. If the result is high, I/O is connected to VDD (3.3 V) and IOB is connected to VDD (3 V) through p-channel devices. VDDL is a 3-V power supply provided externally. The I/O pair is connected to the sense amplifier through p-channel transfer gates controlled by ISWB. During write operations, ISWB falls to connect the I/O pair to the sense amplifier.

Figure 49.16 shows operational waveforms of the BSA. At the beginning of the read operations, after some intrinsic delay from the rising edge of the SACLK, data from the selected cell is read onto the bit-line pair. At the same time, the BLSW and the SAEQB rise. One of the eight CMOS transfer gates is turned on, the bit-line pair is connected to sense nodes D and DB, and precharging of the CMOS sense amplifier and bit-line pair is terminated. After the signal on the bit-line pair signal is sufficiently developed, the BLSW falls to disconnect the bit-line pair from the sense nodes D and DB. At the same time, the SENB falls to activate the sense amplifier. After the differential output data is latched onto the SR flip-flop, the SAEQB falls to start the equalization of the bit-line pair and the CMOS sense amplifier.

At the beginning of the write operations, after some delay from the rising edge of SACLK, the ISWB signal falls, and the differential I/O pair is directly connected to the sense amplifier through p-channel

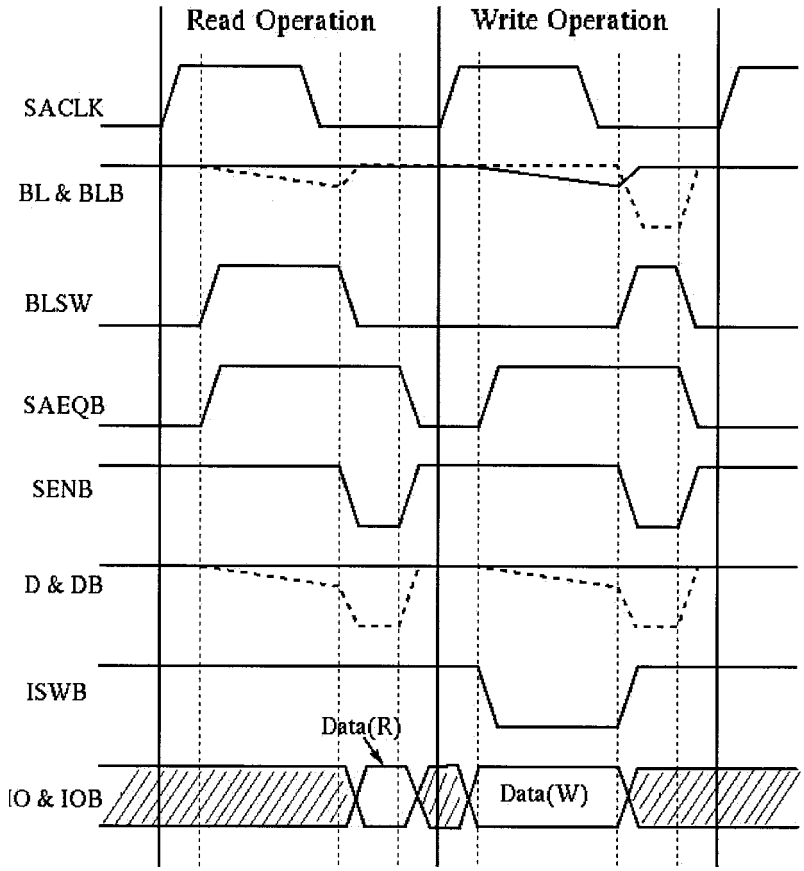


FIGURE 49.16 Operational waveforms of the BSA. (From Ref. 17. With permission.)

transfer gates. After the signals D and DB are sufficiently developed, ISWB turns off the p-channel transfer gates to disconnect the sense amplifier from the I/O pair. At the same time, the SENB falls to sense the data, and BLSW rise to connect the sense amplifier to the bit-line pair. After the data is written into the selected memory cell, SAEQB and BLSW fall to start equalization of the bit-line pair and the CMOS sense amplifier.

Conventional sense amplifiers operate incorrectly when threshold voltage deviation is larger than bit-line swing, a current-sensing sense amplifier proposed by Izumikawa et al. in 1997 can continue to operate normally.[18] Figure 49.17 illustrates the sense amplifier operations. Bit-lines are always charged up to VDD through load PMOSFETs. When memory-cells are selected with a word-line, the voltage difference in a bit-line pair appears (Fig. 49.17(a)). During this period, all column-select PMOSFETs are off, and no dc current flows in the sense amplifier. The sense amplifier differential outputs, referred to as ReadData, are equalized at ground level through pull-down NMOSFETs M7 and M8.

After a 40-mV difference appears in a bit-line pair, power switch M9 of the sense amplifier and one column-select pair of PMOSFETs are set to on (Fig. 49.17(b)). The difference in bit-line voltages causes

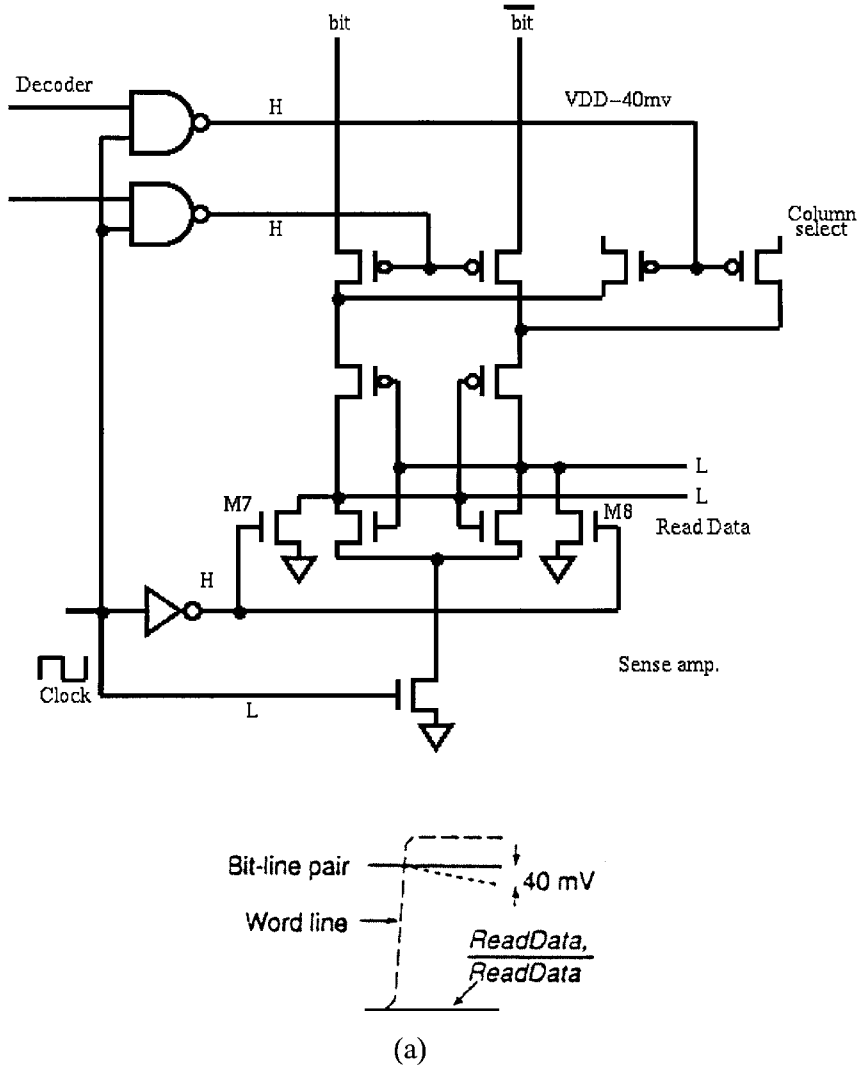


FIGURE 49.17(a) Sense amplifier operation: (a) before sensing. (From Ref. 18. With permission.)

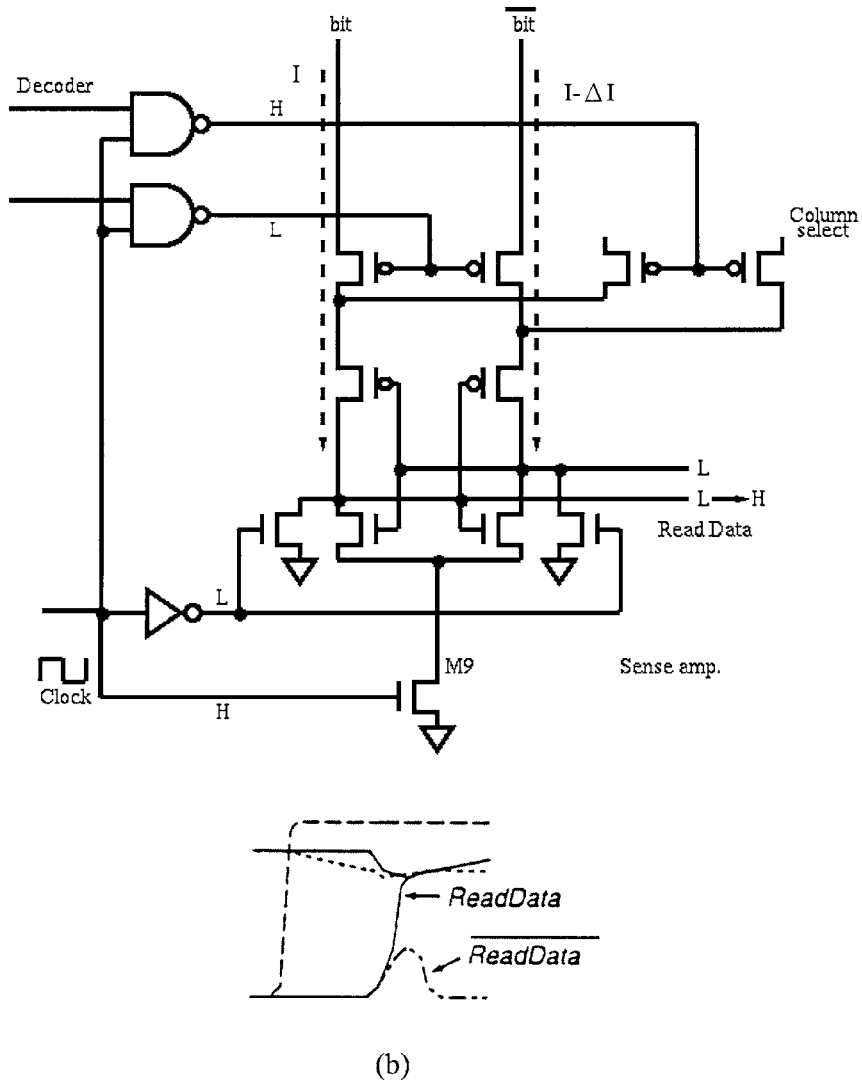


FIGURE 49.17(b) Sense amplifier operation: (b) sensing. (From Ref. 18. With permission.)

a current difference between the differential pair PMOS in the sense amplifier, which appears as an output voltage difference. This voltage difference is amplified, and the read operation is accomplished. The current is automatically cut off because of the CMOS inverter. Consequently, the small bit-line swing is sensed without dc current consumption.

49.5 Output Circuit[4]

The key issue for designing the high-speed SRAM with byte-wide organization is noise reduction. There are two kinds of noise: VDD noise and GND noise. In the high-speed SRAM with byte-wide organization, when the output transistors drive a large load capacitance, the noise is generated and multiplied by 8 because eight outputs may change simultaneously. It is a fundamentally serious problem for the data zero output. That is to say, when the output NMOS transistor drives the large load capacitance, the GND potential of the chip goes up because of the peak current and the parasitic inductance of the GND line. Therefore, the address buffer and the ATD circuit are influenced by the GND bounce, and unnecessary signals are generated.

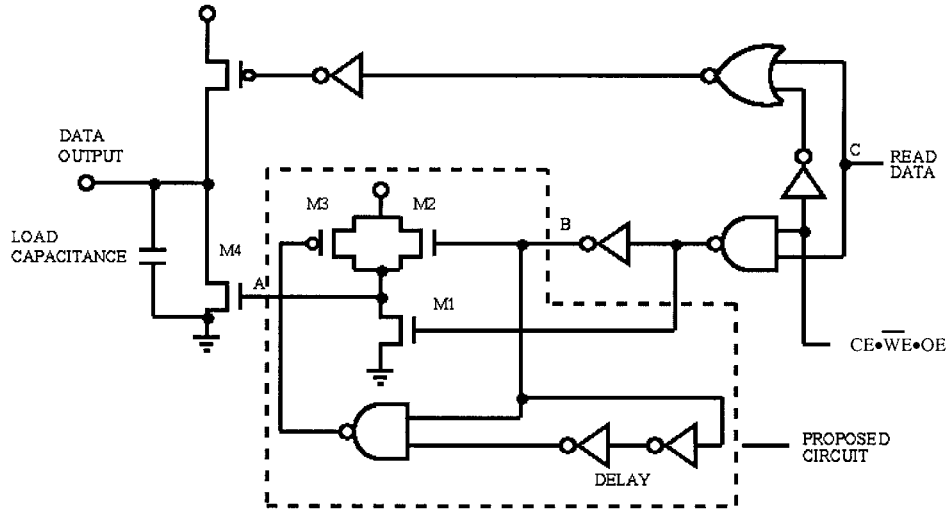


FIGURE 49.18 Noise-reduction output circuit. (From Ref. 4. With permission.)

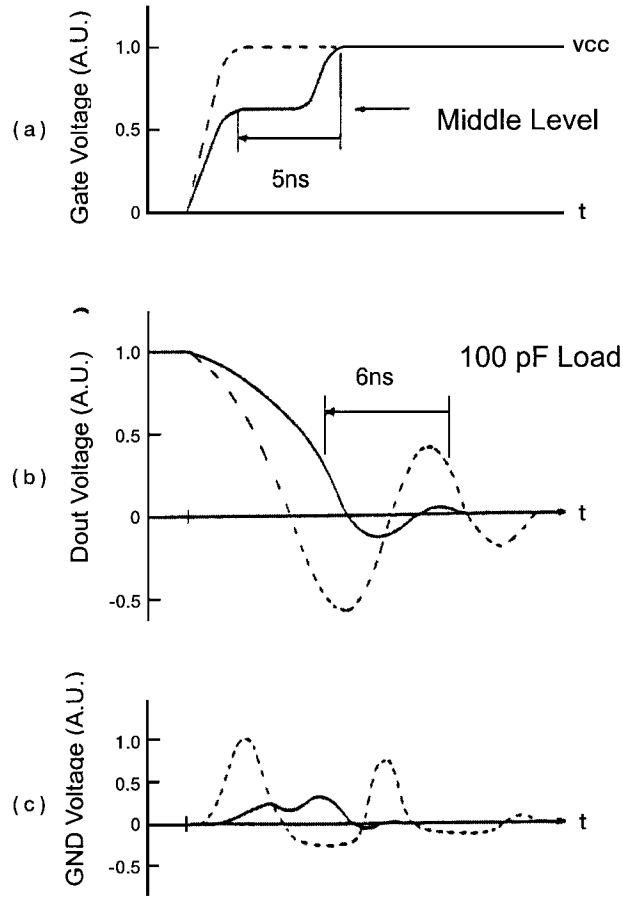


FIGURE 49.19 Waveforms of noise-reduction output circuit (solid line) and conventional output circuit: (a) gate bias, (b) data output, and (c) GND bounce. (From Ref. 4. With permission.)

Figure 49.18 shows a noise-reduction output circuit. The waveforms of the noise-reduction output circuit and conventional output circuit are shown in Fig. 49.19. In the conventional circuit, nodes A and B are connected directly as shown in Fig. 49.18. Its operation and characteristics are shown by the dotted lines in Fig. 49.18. Due to the high-speed driving of transistor M4, the GND potential goes up, and the valid data are delayed by the output ringing. A new noise-reduction output circuit consists of one PMOS transistor, two NMOS transistors, one NAND gate, and the delay part (its characteristics are shown by the solid lines in Fig. 49.19). The operation of this circuit is explained as follows. The control signals CE and OE are at high level and signal WE is at low level in the read operation. When the data zero output of logical high level is transferred to node C, transistor M1 is cut off, and M2 raises node A to the middle level. Therefore, the peak current that flows into the GND line through transistor M4 is reduced to less than one half that of the conventional circuit because M4 is driven by the middle level. After a 5-ns delay from the beginning of the middle level, transistor M3 raises node A to the VDD level. As a result, the conductance of M4 becomes maximum, but the peak current is small because of the low output voltage. Therefore, the increase of GND potential is small, and the output ringing does not appear.

References

1. Bellaouar, A. and Elmasry, M. I., *Low-Power Digital VLSI Design Circuit and Systems*, Kluwer Academic Publishers, 1995.
2. Ishibashi, K. et al., "A 1-V TFT-Load SRAM Using a Two-Step Word-Voltage Method," *IEEE J. Solid-State Circuits*, vol. 27, no. 11, pp. 1519-1524, Nov. 1992.
3. Chen, C.-W. et al., "A Fast 32KX8 CMOS Static RAM with Address Transition Detection," *IEEE J. Solid-State Circuits*, vol. SC-22, no. 4, pp. 533-537, Aug. 1987.
4. Miyaji, F. et al., "A 25-ns 4-Mbit CMOS SRAM with Dynamic Bit-Line Loads," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp.1213-1217, Oct. 1989.
5. Matsumiya, M. et al., "A 15-ns 16-Mb CMOS SRAM with Interdigitated Bit-Line Architecture," *IEEE J. Solid-State Circuits*, vol. 27, no. 11, pp. 1497-1502, Nov. 1992.
6. Mizuno, H. and Nagano, T., "Driving Source-Line Cell Architecture for Sub-1V High-Speed Low-Power Applications," *IEEE J. Solid-State Circuits*, no. 4, pp. 552-557, Apr. 1996.
7. Morimura, H. and Shibata, N., "A Step-Down Boosted-Wordline Scheme for 1-V Battery-Operated Fast SRAM's," *IEEE J. Solid-State Circuits*, no. 8, pp. 1220-1227, Aug. 1998.
8. Yoshimito, M. et al., "A Divided Word-Line Structure in the Static RAM and Its Application to a 64 K Full CMOS RAM," *IEEE J. Solid-State Circuits*, vol. SC-18, no. 5, pp. 479-485, Oct. 1983.
9. Hirose, T. et al., "A 20-ns 4-Mb CMOS SRAM with Hierarchical Word Decoding Architecture," *IEEE J. Solid-State Circuits*, vol. 25, no. 5, pp. 1068-1074, Oct. 1990.
10. Itoh, K., Sasaki, K., and Nakagome, Y., "Trends in Low-Power RAM Circuit Technologies," *Proceedings of the IEEE*, pp. 524-543, Apr. 1995.
11. Nambu, H. et al., "A 1.8-ns Access, 550-MHz, 4.5-Mb CMOS SRAM," *IEEE J. Solid-State Circuits*, vol. 33, no. 11, pp. 1650-1657, Nov. 1998.
12. Cararella, J. S., "A Low Voltage SRAM For Embedded Applications," *IEEE J. Solid-State Circuits*, vol. 32, no. 3, pp. 428-432, Mar. 1997.
13. Prince, B., *Semiconductor Memories: A Handbook of Design, Manufacture, and Application,* 2nd edition, John Wiley & Sons, 1991.
14. Minato, O. et al., "A 20-ns 64 K CMOS RAM," in *ISSCC Dig. Tech. Papers*, pp. 222-223, Feb. 1984.
15. Sasaki, K., et al., "A 9-ns 1-Mbit CMOS SRAM," *IEEE J. Solid-State Circuits*, vol. 24, no. 5, pp. 1219-1224, Oct. 1989.
16. Seki, T. et al., "A 6-ns 1-Mb CMOS SRAM with Latched Sense Amplifier," *IEEE J. Solid-State Circuits*, vol. 28, no. 4, pp. 478-482, Apr. 1993.

17. Kushiyama, N. et al., "An Experimental 295 MHz CMOS 4K X 256 SRAM Using Bidirectional Read/Write Shared Sense Amps and Self-Timed Pulse Word-Line Drivers," *IEEE J. Solid-State Circuits*, vol. 30, no. 11, pp. 1286-1290, Nov. 1995.
18. Izumikawa, M. et al., "A 0.25-μm CMOS 0.9-V 100M-Hz DSP Core," *IEEE J. Solid-State Circuits*, vol. 32, no. 1, pp. 52-60, Jan. 1997.

Wu, C. "Embedded Memory"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

50

Embedded Memory

Chung-Yu Wu
National Chiao Tung University

50.1 Introduction

As CMOS technology progresses rapidly toward the deep submicron regime, the integration level, performance, and fabrication cost increase tremendously. Thus, low-integration low-performance small circuits or systems chips designed using deep submicron CMOS technology are not cost-effective. Only high-performance system chips that integrate CPU (central processing unit), DSP (digital signal processing) processors or multimedia processors, memories, logic circuits, analog circuits, etc. can afford the deep submicron technology. Such system chips are called system-on-a-chip (SOC) or system-on-silicon (SOS).[1,2] A typical example of SOC chips is shown in Fig. 50.1.

Embedded memory has become a key component of SOC and more practical than ever for at least two reasons:[3]

1. Deep submicron CMOS technology affords a reasonable tradeoff for large memory integration in other circuits. It can afford ULSI (ultra large-scale integration) chips with over 10^9 elements on a single chip. This scale of integration is large enough to build an SOC system. This size of circuitry inevitably contains different kinds of circuits and technologies. Data processing and storage are the most primitive and basic components of digital circuits, so that the memory implementation on logic chip has the highest priority. Currently in quarter-micron CMOS technology, chips with up to 128 Mbits of DRAM and 500 Kgates of logic circuit, or 64 Mbits of DRAM and 1 Mgates of logic circuit, are feasible.
2. Memory bandwidth is now one of the most serious bottlenecks to system performance. The memory bandwidth is one of the performance determinants of current von Neuman-type MPU (microprocessing unit) systems. The speed gap between MPUs and memory devices has been increased in the past decade. As shown in Fig. 50.1, the MPU speed has improved by a factor of 4 to 20 in the past decade. On the other hand, in spite of exponential progress in storage capacity, minimum access times for each quadrupled storage capacity have improved only by a factor of two, as shown in Fig. 50.2. This is partly due to the I/O speed limitation and to the fact that major

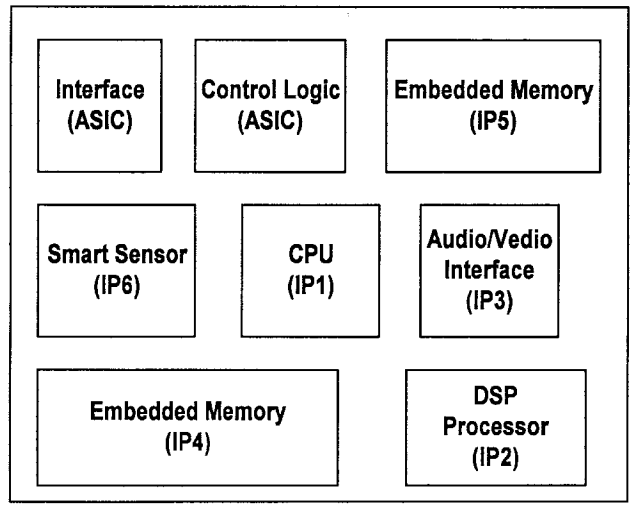


FIGURE 50.1 An example of system-on-a-chip (SOC).

efforts in semiconductor memory development have focused on density and bit cost improvements. This speed gap creates a strong demand for memory integration with MPU on the same chip. In fact, many MPUs with cycle times better than 60 ns have on-chip memories. The new trend in MPUs, (i.e., RISC architecture) is another driving force for embedded memory, especially for cache applications.[4] RISC architecture is strongly dependent on memory bandwidth, so that high-performance, non-ECL-based RISC MPUs with more than 25 to 50 MHz operation must be equipped with embedded cache on the chip.

50.2 Merits and Challenges

The main characteristics of embedded memories can be summarized as follows.[5]

On-chip Memory Interface

Advantages include:

1. Replacing off-chip drivers with smaller on-chip drivers can reduce power consumption significantly, as large board wire capacitive loads are avoided. For instance, consider a system which needs a 4-Gbyte/s bandwidth and a bus width of 256 bits. A memory system built with discrete SDRAMs (16-bit interface at 100 MHz) would require about 10 times the power of an embedded DRAM with an internal 256-bit interface.
2. Embedded memories can achieve much higher fill frequencies,[6] which is defined as the bandwidth (in Mbit/s) divided by the memory size in Mbit (i.e., the fill frequency is the number of times per second a given memory can be completely filled with new data), than discrete memories. This is because the on-chip interface can be up to 512 bits wide, whereas discrete memories are limited to 16 to 64 bits. Continuing the above example, it is possible to make a 4-Mbit embedded DRAM with a 256-bit interface. In contrast, it would take 16 discrete 4-Mbit chips (256 K×16) to achieve the same width, so the granularity of such a discrete system is 64 Mbits. But the application may only call for, say, 8 Mbits of memory.
3. As interface wire lengths can be optimized for application in embedded memories, lower propagation times and thus higher speeds are possible. In addition, noise immunity is enhanced.

Challenges and disadvantages include:

1. Although the power consumption per system decreases, the power consumption per chip may increase. Therefore, junction temperature may increase and memory retention time may decrease. However, it should be noted that memories are usually low-power devices.
2. Some sort of minimal external interface is still needed in order to test the embedded memory. The hybrid chip is neither a memory nor a logic chip. Should it be tested on a memory or logic tester, or on both?

System Integration

Advantages include:

1. Higher system integration saves board space, packages, and pins, and yields better form factors.
2. Pad-limited design may be transformed into non-pad-limited by choosing an embedded solution.
3. Better speed scalability, along with CMOS technology scaling.

Challenges and disadvantages include:

1. More expensive packages may be needed. Also, memories and logic circuits require different power supplies. Currently, the DRAM power supply (2.5 V) is less than the logic power supply (3.3 V); but this situation will reverse in the future due to the back-biasing problem in DRAMs.
2. The embedded memory process adds another technology for which libraries must be developed and characterized, macros must be ported, and design flows must be tuned.
3. Memory transistors are optimized for low leakage currents, yielding low transistor performance, whereas logic transistors are optimized for high saturation currents, yielding high leakage currents. If a compromise is not acceptable, expensive extra manufacturing steps must be added.
4. Memory processes have fewer layers of metal than do logic circuit processes. Layers can be added at the expense of fabrication cost.
5. Memory fabs are optimized for large-volume production of identical products, for high-capacity utilization and for high yield. Logic fabs, while sharing these goals, are slanted toward lower batch sizes and faster turnaround time.

Memory Size

The advantage is that:

1. Memory size can be customized and memory architecture can be optimized for dedicated applications.

Challenges and disadvantages include:

1. On the other hand, the system designer must know the exact memory requirement at the time of design. Later extensions are not possible, as there is no external memory interface. From the customer's point of view, the memory component goes from a commodity to a highly specialized part that may command premium pricing. As memory fabrication processes are quite different, second-sourcing problems abound.

50.3 Technology Integration and Applications[3,5]

The memory technologies for embedded memories have a wide variation—from ROM to RAM—as listed in Table 50.1.[3] In choosing these technologies, one of the most important figure of merits is the compatibility to logic process.

TABLE 50.1 Embedded Memory Technologies and Applications

Embedded Memory Technology	Compatibility to Logic Process	Applications
ROM	Diffusion, Vt, Contact programming High compatibility to logic process	Microcode, program storage PAL, ROM-based logic
E/E^{2}prom	High-voltage device, tunneling insulator required	Program, parameter storage, sequencer, learning machine
SRAM	6-Tr/4-Tr single/double poly load cells. Wide range of compatibility	High-speed buffers, cache memory
DRAM	Gate capacitor /4-T /planar /stacked / trench cells. Wide range of compatibility	High-density, high bit rate storage

Source: From Ref. 3.

1. Embedded ROM: ROM technology has the highest compatibility to logic process. However, its application is rather limited. PLA, or ROM-based logic design, is a well-used but rather special case of embedded ROM category. Other applications are limited to storage for microcode or well-debugged control code. A large size ROM for tables or dictionary applications may be implemented in generic ROM chips with lower bit cost.
2. Embedded EPROM/E^{2}ROM: EPROM/E^{2}PROM technology includes high-voltage devices and/or thin tunneling insulators, which require two to three additional mask steps and processing steps to logic process. Due to its unique functionality, PROM-embedded MPUs[7] are well used. To minimize process overhead, single poly E^{2}PROM cell has been developed.[8] Counterparts to this approach are piggy-back packaged EPROM/MPUs or battery-backed SRAM/MPUs. However, considering process technology innovation, on-chip PROM implementation is winning the game.
3. Embedded SRAM is one of the most frequently used memory embedded in logic chips. Major applications are high-speed on-chip buffers such as TLB, cache, register file, etc. Table 50.2 gives a comparison of some approaches for SRAM integration. A six-transistor cell approach may be the most highly compatible process, unless any special structures used in standard 6-Tr SRAMs are employed. The bit density is not very high. Polysilicon resistor load 4-Tr cells provide higher bit density with the cost of process complexity associated with additional polysilicon-layer resistors. The process complexity and storage density may be compromised to some extent using a single layer of polysilicon. In the case of a polysilicon resistor load SRAM, which may have relaxed specifications with respect to data holding current, the requirement for substrate structure to achieve good soft error immunity is more relaxed as compared to low stand-by generic SRAMs. Therefore, the TFT (thin-film transistor) load cell may not be required for several generations due to its complexity.

TABLE 50.2 Embedded SRAM Options

SRAM Cell Type	Features
CMOS 6-Tr cell	No extra process steps to logic Lower bit density (Cell size, Acell=2.0 a.u.) Wide operational margin Low data-load current
NMOS 4-Tr Polysilicon Load Cell	
-Single Poly:	1 additional step to logic process Higher density (Acell=1.25 a.u.)
-Double Poly:	3 addititional steps to logic process Higher density (Acell=1 a.u.)

Source: From Ref. 3.

4. Embedded DRAM (eDRAM) is not as widely used as SRAMs. Its high density features, however, are very attractive. Several different embedded DRAM approaches are listed in Table 50.3. A trench or stacked cell used in commodity DRAMs has the highest density, but the complexity is also high.

TABLE 50.3 Embedded DRAM Technology Options

Technology	Features
Standard DRAM Trench/Stacked Cell	High density (cell size A_{cell} = 1 a.u.)
	Large process overhead,>45% additional to logic
Planar C-plate poly-Si Cell	High density ($A_{cell>}$ = 1.3 a.u.)
	Process overhead >35% additional to logic
Gate capacitor +	Relatively high density (A_{cell} = 2.5 a.u.)
1-Tr Cell	No additional process to logic
4-Tr Cell	High speed, short cycle time
	Density is equivalent to 2-poly SRAM cell
	(equiv. to SRAM excpt refresh. A_{cell} = 5 a.u.)

Source: From Ref. 3.

The cost is seldom attractive when compared to a multi-chip approach using standard DRAM, which is the ultimate in achieving low bit cost. This type of cell is well suited for ASM (application specific memory), which will be described in the next section. A planar cell with multiple (double) polysilicon structures is also suitable for memory-rich applications.[9] A gate capacitor storage cell approach can be fully compatible to logic process providing relatively high density.[10] The four-Tr cell (4-Tr SRAM cell minus resistive load) provides the same speed and density as SRAM, but full compatibility to logic process and requires refresh operation.[11]

50.4 Design Methodology and Design Space[3,5]

Design Methodology

The design style of embedded memory should be selected according to applications. This choice is critically important for the best performance and cost balancing. Figure 50.2 shows the various design styles to implement embedded memories.

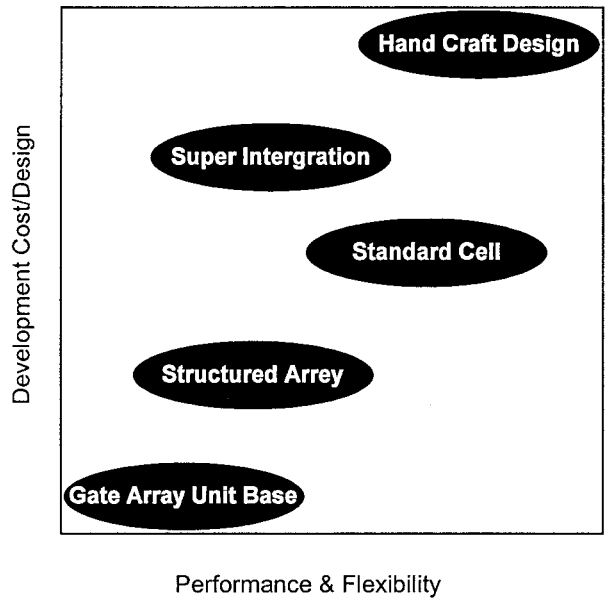


FIGURE 50.2 Various design styles for embedded memories. (From Ref. 3.)

The most primitive semi-custom design style is based on unit the memory cell. It provides high flexibility in memory architecture and short design TAT (turn around time). However, the memory density is the lowest among various approaches.

The structured array is a kind of gate array that has a dedicated memory array region in the master chip that is configurable to several variations of memory organizations by metal layer customization. Therefore, it provides relatively high density and short TAT. Configurability and fixed maximum memory area are the limitations to this approach.

The standard cell design has high flexibility to the extent that the cell library has a variety of embedded memory designs. But in many cases, new system design requires new memory architectures. The memory performance and density is high, but the mask-to-chip TAT tends to be long.

Super integration is an approach that integrates existing chip design, including I/O pads, so the design TAT is short and proven designs can be used. However, availability of memory architecture is limited and the mask-to-chip TAT is long.

Hand-craft design (does not necessarily mean the literal use of human hands, but heavy interactive design) provides the most flexibility, high performance, and high density; but design TAT is the longest. Thus, design cost is the highest so that the applications are limited to high-volume and/or high-end systems. Standard memories, well-defined ASMs, such as video memories,[12] integrated cache memories,[13] and high-performance MPU-embedded memories, are good examples.

An eDRAM (embedded DRAM) designer faces a design space that contains a number of dimensions not found in standard ASICs, some of which we will subsequently review. The designer has to choose from a wide variety of memory cell technologies which differ in the number of transistors and in performance.

Also, both DRAM technology and logic technology can serve as a starting point for embedding DRAM. Choosing a DRAM technology as the base technology will result in high memory densities but suboptimal logic performance. On the other hand, starting with logic technology will result in poor memory densities, but fast logic circuits. To some extent, one can therefore trade logic speed against logic area. Finally, it is also possible to develop a process that gives the best of both worlds—most likely at higher expense. Furthermore, the designer can trade logic area for memory area in a way heretofore impossible.

Large memories can be organized in very different ways. Free parameters include the number of memory banks, which allow the opening of different pages at the same time, the length of a single page, the word width, and the interface organization. Since eDRAM allows one to integrate SRAMs and DRAMs, the decision between on/off-chip DRAM-and SRAM/DRAM-partitioning must be made.

In particular, the following problems must be solved at the system level:

- Optimizing the memory allocation
- Optimizing the mapping of the data into memory such that the sustainable memory bandwidth approaches the peak bandwidth
- Optimizing the access scheme to minimize the latency for the memory clients and thus minimize the necessary FIFO depth

The goals are to some extent independent of whether or not the memory is embedded. However, the number of free parameters available to the system designer is much larger in an embedded solution, and the possibility of approaching the optimal solution is thus correspondingly greater. On the other hand, the complexity is also increased. It is therefore incumbent upon eDRAM suppliers to make the tradeoffs transparent and to quantize the design space into a set of understandable if slightly suboptimal solutions.

50.5 Testing and Yield[3,5]

Although embedded memory occupies a minor portion of the total chip area, the device density in the embedded memory area is generally overwhelming. Failure distribution is naturally localized at memory areas. In other words, embedded memory is a determinant of total chip yield to the extent that the memory portion has higher device density weighted by its silicon area.

For a large memory-embedded VLSI, memory redundancy is helpful to enhance the chip yield. Therefore, the embedded-memory testing, combined with the redundancy scheme, is an important issue. The implementation of means for direct measurement of embedded memory on wafer as well as in assembled samples is necessary.

In addition to off-chip measurement, on-chip measurement circuitry is essential for accurate AC evaluation and debugging. Testing DRAMs is very different from testing logic. In the following, the main points of notice are discussed.

- The fault models of DRAMs explicitly tested for are much richer. They include bit-line and word-line failures, crosstalk, retention time failures, etc.
- The test patterns and test equipment are highly specialized and complex. As DRAM test programs include a lot of waiting, DRAM test times are quite high, and test costs are a significant fraction of total cost.
- As DRAMs include redundancy, the order of testing is: (1) pre-fuse testing, (2) fuse blowing, (3) post-fuse testing. There are thus two wafer-level tests.

The implication on eDRAMs is that a high degree of parallelism is required in order to reduce test costs. This necessitates on-chip manipulation and compression of test data in order to reduce the off-chip interface width. For instance, Siemens Corp. offers a synthesizable test controller supporting algorithmic test pattern generation (ATPG) and expected-value comparison [partial built-in self test (BIST)].

Another important aspect of eDRAM testing is the target quality and reliability. If eDRAM is used for graphics applications, occasional "soft" problems, such as too short retention time of a few cells, are much more acceptable than if eDRAM is used for program data. The test concept should take this cost-reduction potential into account, ideally in conjunction with the redundancy concept.

A final aspect is that a number of business models are common in eDRAM, from foundry business to ASIC-type business. The test concept should thus support testing the memory, either from a logic tester or a memory tester, so that the customer can do memory testing on his logic tester if required.

50.6 Design Examples

Three examples of embedded memory designs are described. The first one is a flexible embedded DRAM design from Siemens Corp.[5] The second one is the embedded memories in MPEG environment from Toshiba Corp.[14] The last one is the embedded memory design for a 64-bit superscaler RISC microprocessor from Toshiba Corp. and Silicon Graphics, Inc.[15]

A Flexible Embedded DRAM Design[5]

There is an increasing gap between processor and DRAM speed: processor performance increases by 60% per year in contrast to only a 10% improvement in the DRAM core. Deep cache structures are used to alleviate this problem, albeit at the cost of increased latency, which limits the performance of many applications. Merging a microprocessor with DRAM can reduce the latency by a factor of 5 to 10, increase the bandwidth by a factor of 50 to 100, and improve the energy efficiency by a factor of 2 to 4.[16]

Developing memory is a time-consuming task and cannot be compared with a high-level based logic design methodology which allows fast design cycles. Thus, a flexible memory concept is a prerequisite for a successful application of eDRAM. Its purpose is to allow fast construction of application-specific memory blocks that are customized in terms of bandwidth, word width, memory size, and the number of memory banks, while guaranteeing first-time-right designs accompanied by all views, test programs, etc.

A powerful eDRAM approach that permits fast and safe development of embedded memory modules is described. The concept, developed by Siemens Corp. for its customers, uses a 0.24-μm technology based on its 64/256 Mbit SDRAM process.[5] Key features of the approach include:

- Two building-block sizes, 256 Kbit and 1 Mbit; memory modules with these granularities can be constructed
- Large memory modules, from 8 to 16 Mbit upwards, achieving an area efficiency of about 1 Mbit/mm^2
- Embedded memory sizes up to at least 128 Mbits
- Interface widths ranging from 16 to 512 bits per module
- Flexibility in the number of banks as well as the page length
- Different redundancy levels, in order to optimize the yield of the memory module to the specific chip
- Cycle times better than 7 ns, corresponding to clock frequencies better than 143 MHz.
- A maximum bandwidth per module of about 9 Gbyte/s
- A small, synthesizable BIST controller for the memory (see next section)
- Test programs, generated in a modular fashion

Siemens Corp. has made eDRAM since 1989 and has a number of possible applications of its eDRAM approach in the pipeline, including TV scan-rate converters, TV picture-in-picture chips, modems, speech-processing chips, hard-disk drive controllers, graphics controllers, and networking switches. These applications cover the full range of memory sizes (from a few Mbits to 128 Mbits), interface widths (from 32 to 512 bits), and clock frequencies (from 50 to 150 MHz), which demonstrates the versatility of the concept.

Embedded Memories in MPEG Environment[14]

Recently, multimedia LSIs, including MPEG decoders, have been drawing attention. The key requirements in realizing multimedia LSIs are their low-power and low-cost features. This example presents embedded memory-related techniques to achieve these requirements, which can be considered as a review of the state-of-the-art embedded memory macro techniques applicable to other logic LSIs.

Figure 50.3 shows embedded memory macros associated with the MPEG2 decoder. Most of the functional blocks use their own dedicated memory blocks and, consequently, memory macros are rather

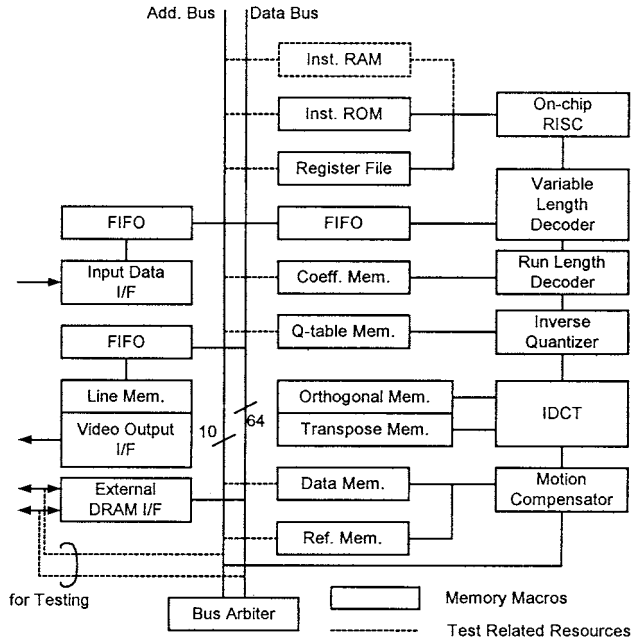


FIGURE 50.3 Block diagram of MPEG2 decoder LSI. (From Ref. 14.)

small and distributed on a chip. Memory blocks are also connected to a central address/data bus for implementing direct test mode.

An input buffer for the IDCT is shown in Fig. 50.4. Eight 16-bit data from D0 to D7 come from the inverse quantization block sequentially. The stored data should then be read out as 4-bit chunks orthogonal to the input sequence. The 4-bit data is used to address a ROM in the IDCT to realize a distributed arithmetic algorithm.

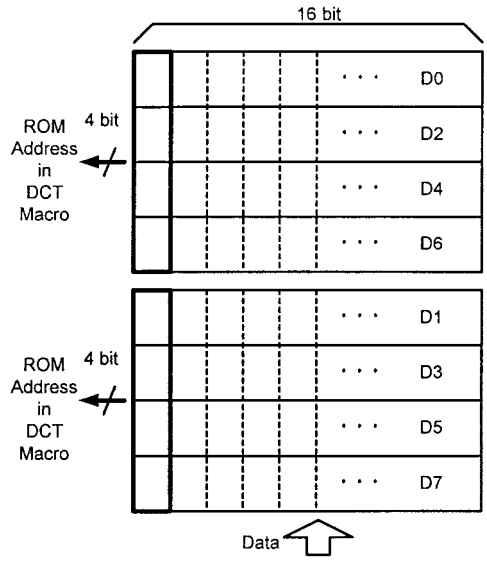


FIGURE 50.4 Input buffer structure for IDCT. (From Ref. 14.)

The circuit diagram of an orthogonal memory whose circuit diagram is shown in Fig. 50.5. It realizes the above-mentioned functionality with 50% of the area and the power that would be needed if the IDCT input buffer were built with flip-flops. In the orthogonal memory, word-lines and bit-lines run both vertically and horizontally to achieve the functionality. The macro size of the orthogonal memory is 420 μm × 760 μm, with a memory cell size of 10.8 μm × 32.0 μm.

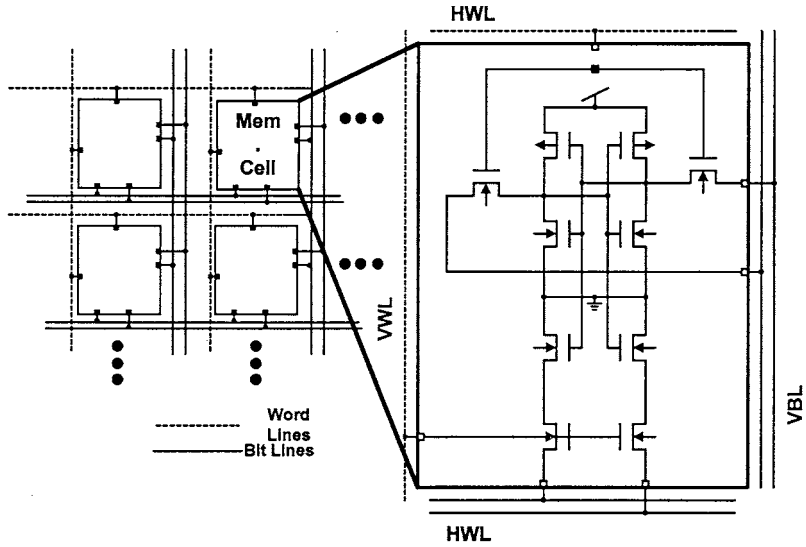


FIGURE 50.5 Circuit diagram of orthogonal memory.(From Ref. 14.)

FIFOs and other dual-port memories are designed using a single-port RAM operated twice in one clock cycle to reduce area, as shown in Fig. 50.6. A dual-port memory cell is twice as large as a single-port memory cell.

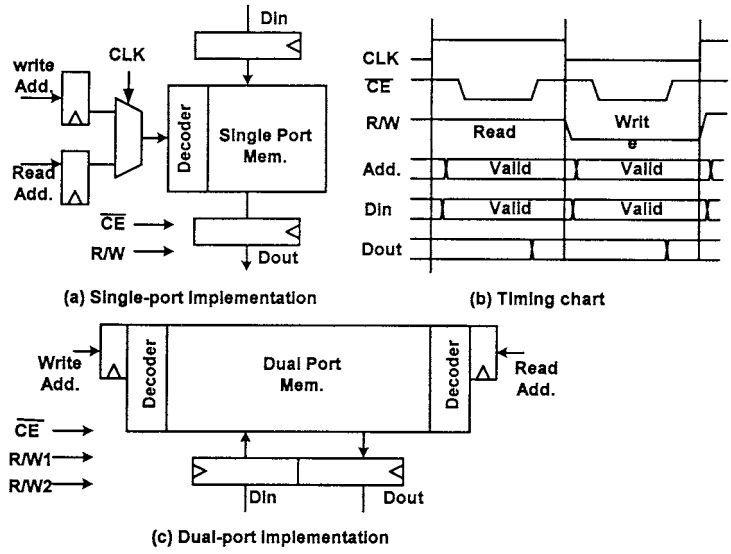


FIGURE 50.6 Realizing dual-port memory with a single-port memory (FIFO case). (From Ref. 14.)

All memory blocks are synchronous self-timed macros and contain address pipeline latches. Otherwise, the timing design needs more time, since the lengths of the interconnections between latches and a decoder vary from bit to bit. Memory power management is carried out using a Memory Macro Enable signal when a memory macro is not accessed, which reduces the total memory power to 60%.

Flip-flop (F/F) is one of the memory elements in logic LSIs. Since digital video LSIs tend to employ several thousand F/Fs on a chip, the design of the F/F is crucial for small area and low power. The optimized F/F with hold capability is shown in Fig. 50.7. Due to the optimized smaller transistor sizes, especially for clock input transistors, and a minimized layout accomodating a multiplexer and a D-F/F in one cell, 40% smaller power and area are realized compared with a normal ASIC F/F.

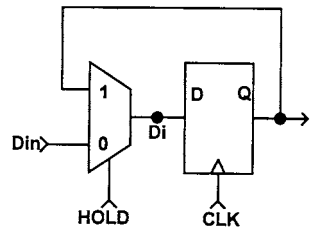


FIGURE 50.7 Optimized flip-flop. (From Ref. 14.)

Establishing full testability of on-chip memories without much overhead is another important issue. Table 50.4 compares three on-chip memory test strategies: a BIST (Built-In Self Test), a scan test, and a direct test. The direct test mode, where all memories can be directly accessed from outside in a test mode, is implemented because of its inherent small area. In a test mode, DRAM interface pads are turned into test pins and can access to each memory block through internal buses, as shown in Figs. 50.3 and 50.8.

TABLE 50.4 Comparison of Various Memory Test Strategies

Items	Direct	Scan	BIST
Area	○	Δ	X
Test time	○	X	○
Pattern control	○	○	X
Bus capacitance	Δ	○	○
At-speed test	○	X	○

○: Good Δ: Fair X: Poor
Source: Ref. 14.

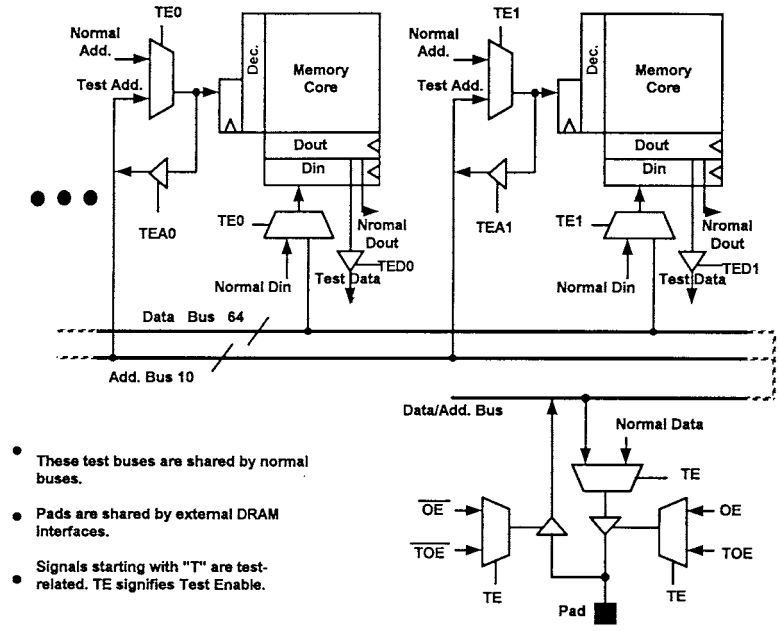


FIGURE 50.8 Direct test architecture for embedded memories. (From Ref. 14.)

The present MPEG2 decoder contains a RISC whose firmware is stored in an on-chip ROM. In order to make the debugging easy and extensive, an instruction RAM is put outside the pads in parallel to the instruction ROM and activated by an Al-masterslice in an initial debugging stage as shown in Fig. 50.9. For a sample chip mounted in a plastic package, the instruction RAM is cut out by a scribe line. This scheme enables extensive debugging and early sampling at the same time for firmware-ROM embedded LSIs.

Embedded Memory Design for a 64-bit Superscaler RISC Microprocessor[15]

High-performance embedded memory is a key component in VLSI systems because of the high-speed and wide bus width capability eliminating inter-chip communication. In addition, multi-ported buffer memories are often demanded on a chip. Furthermore, a dedicated memory architecture that meets the special constraint of the system can neatly reduce the system critical path.

On the other hand, there are several issues in embedded RAM implementation. The specialty or variety of the memories could increase design cost and chip cost. Reading very wide data causes large power dissipation. Test time of the chip could be increased because of the large memory. Therefore, design efficiency, careful power bus design, and careful design for testability are necessary.

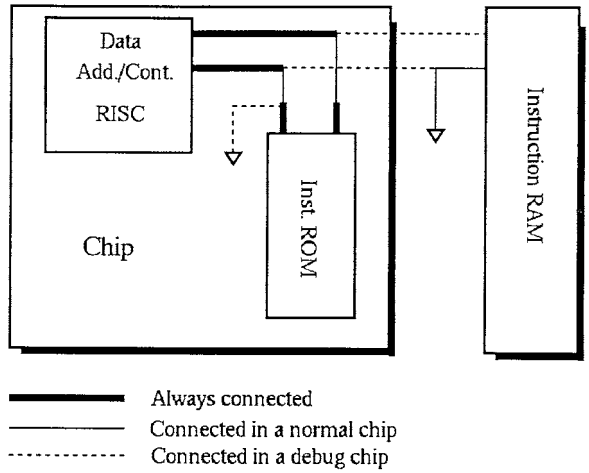


FIGURE 50.9 Instruction RAM masterslice for code debugging. (From Ref. 14.)

TFP is a high-speed and highly concurrent 64-bit superscaler RISC microprocessor, which can issue up to four instructions per cycle.[17,18] Very wide bandwidth of on-chip caches is vital in this architecture. The design of the embedded RAMs, especially on caches and TLB, is reported.

The TFP integer unit (IU) chip implements two integer ALU pipelines and two load/store pipelines. The block diagram is shown in Fig. 50.10. A five-stage pipeline is shown in Fig. 50.11. In the TFP IU chip, RAM blocks occupy a dominant part of the real estate. The die size is 17.3 mm × 17.3 mm. In addition to other caches, TLB, and register file, the chip also includes two buffer queues: SAQ (store address queue) and FPQ (floating point queue). Seventy-one percent of all overall 2.6 million transistors are used for memory cells. Transistor counts of each block are listed in Table 50.5.

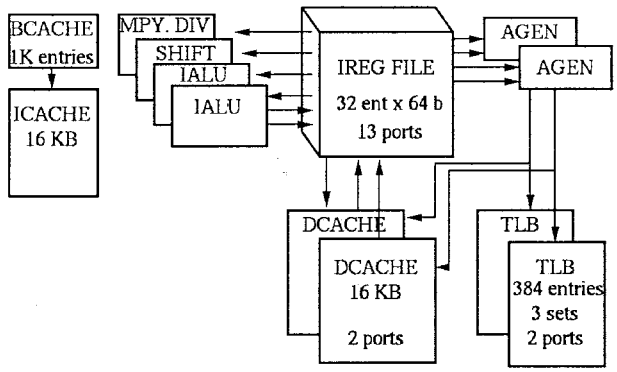


FIGURE 50.10 Block diagram of TFP IU. (From Ref. 15.)

The first generation of TFP chip was fabricated using Toshiba's high-speed 0.8 μm CMOS technology: double poly-Si, triple metal, and triple well. A deep n-well was used in PLL and cache cell arrays in order to decouple these circuits from the noisy substrate or power line of the CMOS logic part. The chip operates up to 75 MHz at 3.1 V and 70°C, and the peak performance reaches 300 MIPS.

Features of each embedded memory are summarized in Table 50.6. Instruction, branch, and data caches are direct mapped because of the faster access time. High-resistive poly-Si load cells are used for these caches since the packing density is crucial for the performance.

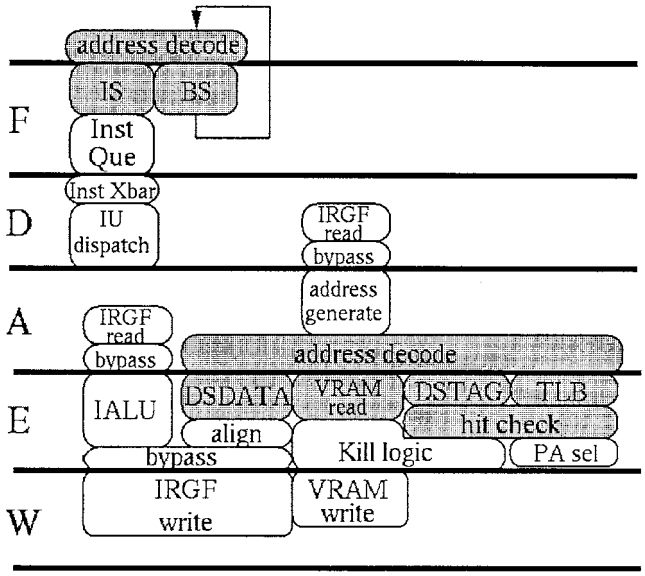


FIGURE 50.11 TFP IU pipelining. (From Ref. 15.)

TABLE 50.5 Transistor Counts

Block	Transistor Count	Ratio (%)
Cache, TLB memory cell	1,761,040	67.02%
RegFile, FPQ, SAQ memory cells	106,624	4.06%
Custom block without memory cell	209,218	19.38%
Random blocks	250,621	9.54%
Total	2,627,503	100.00%

Source: Ref. 15.

Instruction cache (ICACHE) is 16 KB of virtual address memory. It provides four instructions (128 bit wide) per cycle. Branch cache (BCACHE) contains branch target address with one flag bit to indicate a predicted branch. BCACHE contains 1-K entries and is virtually indexed in parallel with ICACHE.

Data cache (DCACHE) is 16 KB, dual ported, and supports two independent memory instructions (two loads, or one load and one store) per cycle. Total memory bandwidth of ICACHE and DCACHE reaches 2.4 GB/s at 75 MHz. Floating point load/store data bypass DCACHE and go directly to bigger external global cache.[17,19] DCACHE is virtually indexed and physically tagged.

TLB is dual ported, three-set-associative memory containing 384 entries. A unique address comparison scheme is employed here, which will be described in the following section. It supports several different page sizes, ranging from 4 KB to 16 MB. TLB is indexed by low-order 7 bits of virtual page number (VPN). The index is hashed by exclusive-OR with a low-order ASID (address space identifier) so that many processes can co-exist in TLB at one time.

Since several different RAMs are used in TFP chip, the design efficiency is important. Consistent circuit schemes are used for each of the caches and TLB RAMs. Layout is started from the block that has the tightest area restriction, and the created layout modules are exported to other blocks with small modification.

The basic block diagram of cache blocks is shown in Fig. 50.12, and timing diagram is shown in Fig. 50.13. Unlike a register file or other smaller queue buffers, these blocks employ dual-railed bit-lines. To achieve 75-MHz operation in the worst-case condition, it should operate at 110 MHz under typical conditions. In this targeted 9-ns cycle time, address generation is done about 3 ns before the end of the

TABLE 50.6 Summary of Embedded RAM Features

Block	Feature	Cell Size
Instruction cache	16 KB, direct mapped	Hi-R cell
(ICACHE)	32 B line size Vitually addressed 4 instructions per cycle	6.75 μm × 9 μm
Branch Cache (BCACHE)	1 K entries, direct mapped	Hi-R cell 6.75 μm × 9 μm
Data cache	2-ported, 16 KB, direct mapped 32 B line size Virtually indexed and physically tagged Write through	Hi-R cell 12.6 μm × 9.45 μm
Valid RAM (VRAM)	One valid bit for 32 b word 4-ported (2 read, 2 write) 34.3μm × 18.9μm	CMOS cell
TLB	3 sets, 384 entries 2-ported Index is hashed by ASID Supported page size: 4K,8K,16K,64K,1M,4,16M	CMOS cell 21.2 μm ×13.7 μm
Register file	64 b × 32 entries 13-ported (9 read, 4 write)	CMOS cell 59.5 μm × 42.8 μm
Floating point queue (FPQ)	Dispatches 4 floating-point instructions per cycle 3-ported (2 read, 1 write) 16 entries	16.1 μm × 40.7 μm
Store address queue (SAQ)	Content addressable 3-ported (1 read, 1 write, 1 compare) 32 entries, 2 banked	CMOS cell 35.1 μm × 17.1 μm

Source: Ref. 15.

cycle, as shown in Fig. 50.11. To take advantage of this big address set-up time, address is received by transparent latch: TLAT_N (transparent while clock is low) instead of flip-flop. Thus, decode is started as soon as address generation is done and is finished before the end of the cycle. Another transparent latch—TLAT_P (transparent while clock is high)—is placed after the sense amplifier and it holds read data while the clock is low.

Word-line (WL) is enabled while clock is high. Since the decode is already finished, WL can be driven to "high" as fast as possible. The sense amplifier is enabled (SAE) with a certain delay after the word-line. The paired current-mirror sense amplifier is chosen since it provides good performance without overly strict SAE timing. Bit-line is precharged and equalized while the clock is low. The clock-to-data delay of DCACHE, which is the biggest array, is 3.7 ns under typical conditions: clock-to-WL is 0.9 ns and WL-to-data is 2.8 ns. Since on-chip PLL provides 50% duty clock, timing pulses such as SAE or WE (write enable) are created from system clock by delaying the positive edge and negative edge appropriately.

As both word-line and sense amplifier are enabled in just half the time of one cycle, the current dissipation is reduced by half. However, the power dissipation and current spike are still an issue because the read/write data width is extremely large. Robust power bus matrix is applied in the cache and TLB blocks so that the dc voltage drop at the worst place is limited to 60 mV inside the block.

From a minimum cycle time viewpoint, write is more critical than read because write needs bigger bit-line swing, and the bit-line must be precharged before the next read. To speed up precharge time,

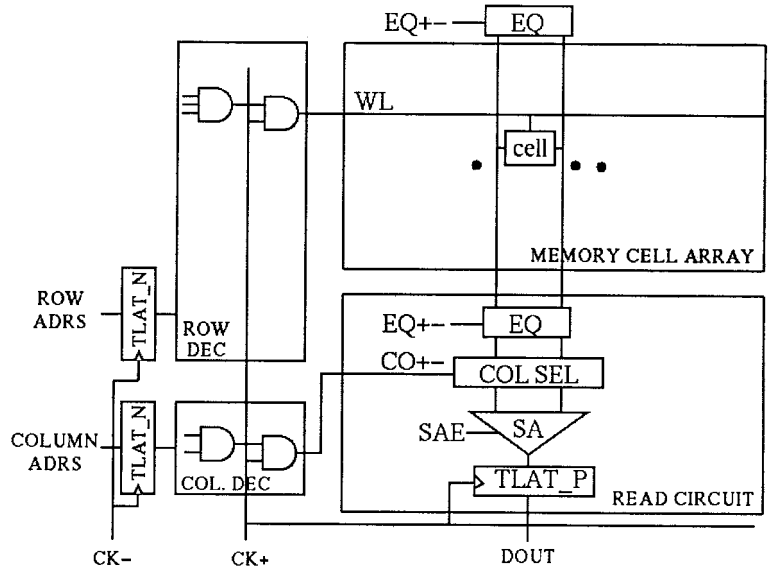


FIGURE 50.12 Basic RAM block diagram. (From Ref. 15)

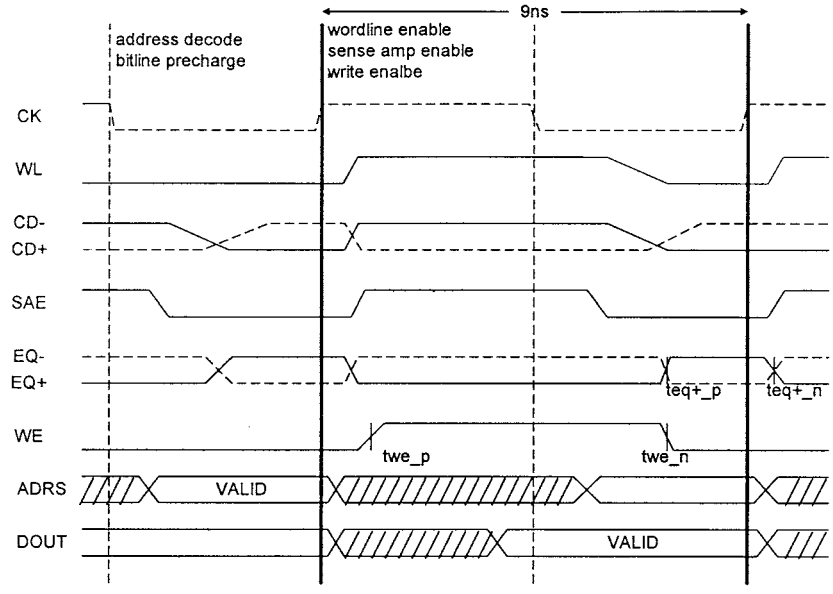


FIGURE 50.13 RAM timing diagram. (From Ref. 15)

precharge circuitry is placed on both the top and bottom of the bit-line. In addition, the write circuitry dedicated to cache-refill is placed on the top side of DCACHE and ICACHE to minimize the wire delay of the write data from input pad. Write data bypass selector is implemented so that the write data is available as read data in the same cycle with no timing penalty.

Virtual to physical address translation and following cache hit check are almost always one of the critical paths in a microprocessor. This is because the cache tag comparison has to wait for the VTLB (RAM that contains virtual address tag) search operation and the following physical address selection from PTLB (RAM that contains physical address).[20] A timing example of the conventional scheme is

shown in Fig. 50.14. In TFP, the DCACHE tag is directly compared with all the three sets of PTLB data in parallel—which are merely candidates of physical address at this stage—without waiting for the VTLB hit results. The block diagram and timing are shown in Figs. 50.15 and 50.16. By the time this hit check of the cache tag is done, VTLB hit results are just ready and they select the PTLB hit result immediately. The "ePmatch" signal in Fig. 50.16 is the overall cache hit result. Although three times more comparators are needed, this scheme saves about 2.8 ns as compared to the conventional one.

In TLB, sense amplifiers of each port are separately placed on the top and bottom of the array to mitigate the tight layout pitch of the circuit. A large amount of wire creates problems around VTLB,

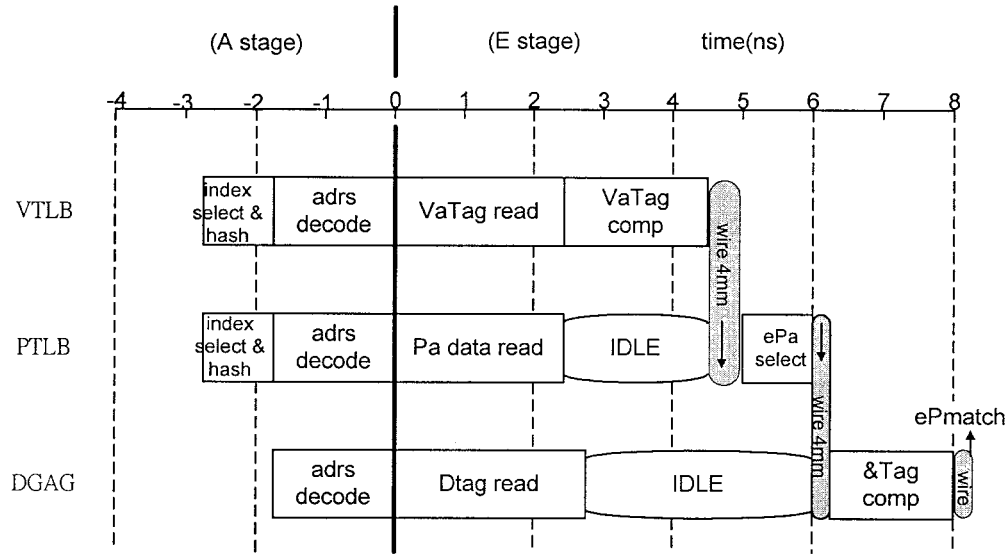


FIGURE 50.14 Conventional physical cache hit check. (From Ref. 15.)

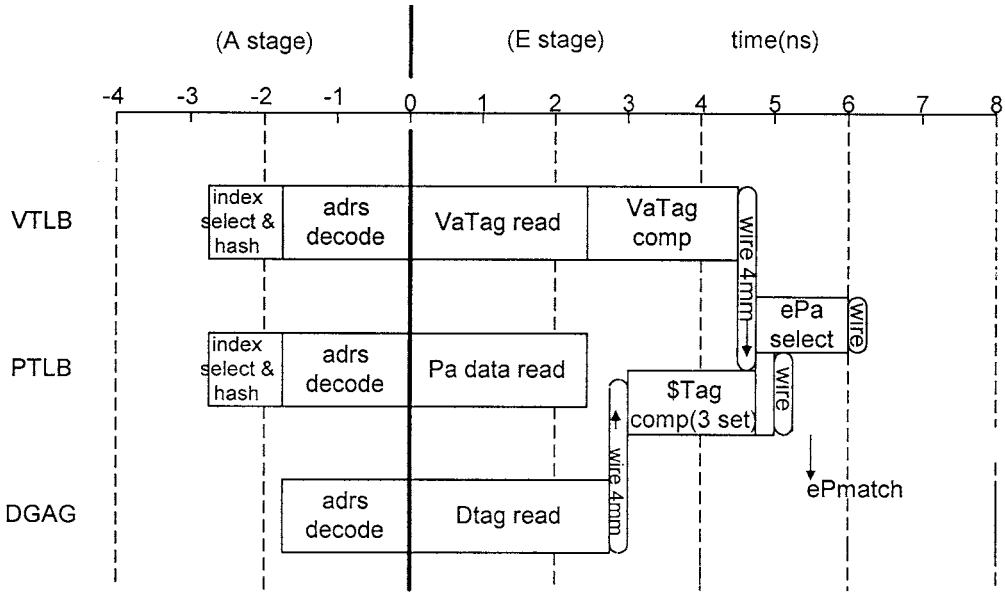


FIGURE 50.15 TFP physical cache hit check. (From Ref. 15.)

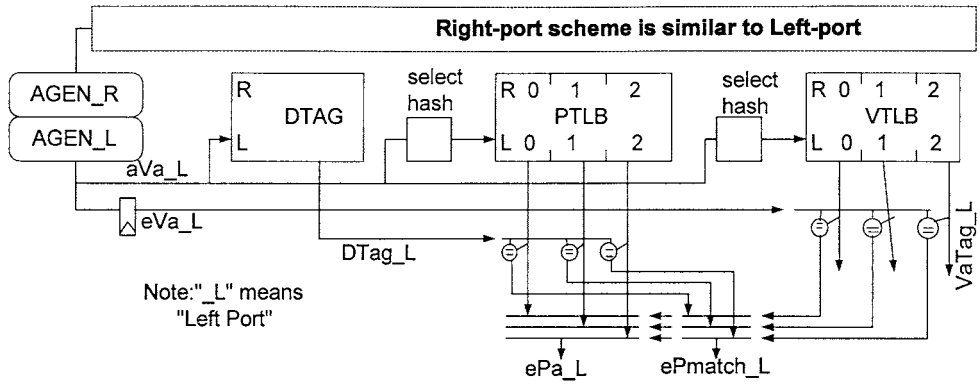


FIGURE 50.16 Block diagram of TLB and DTAG. (From Ref. 15.)

PTLB, and DTAG (DCACHE tag RAM) from both layout and critical path viewpoints. This was solved by piling them to build a data path (APATH: Address Data Path) by making the most of the metal-3 vertical interconnection. Although this metal-3 signal line runs over TLB arrays in parallel with the metal-1 bit-line, the TLB access time is not degraded since horizontal metal-2 word-line shields the bit-line from the coupling noise. The data fields of three sets are scrambled to make the data path design tidy; 39-bit (in VTLB) and 28-bit (in PTLB) comparators of each set consist of optimized AND-tree. Wired-OR type comparators are rejected because a longer wired-OR node in this array configuration would have a speed penalty.

As TFP supports different page sizes, VPN and PFN (page frame number) fields change, depending on the page size. The index and comparison field of TLB are thus made selectable by control signals.

32-bit DCACHE data are qualified by one valid bit. A valid bit needs the read-modify-write operation based on the cache hit results. However, this is not realized in one cycle access because of tight timing. Therefore, two write ports are added to valid bit and write access is moved to the next cycle: the W-stage. The write data bypass selector is essential here to avoid data hazard.

To minimize the hardware overhead of the VRAM (valid bit RAM) row decoder, two schemes are applied. First, row decoders of read ports are shared with DCACHE by pitch-matching one VRAM cell height with two DCACHE cells. Second, write word-line drivers are made of shift registers that have read word-lines as inputs. The schematic is shown in Fig. 50.17.

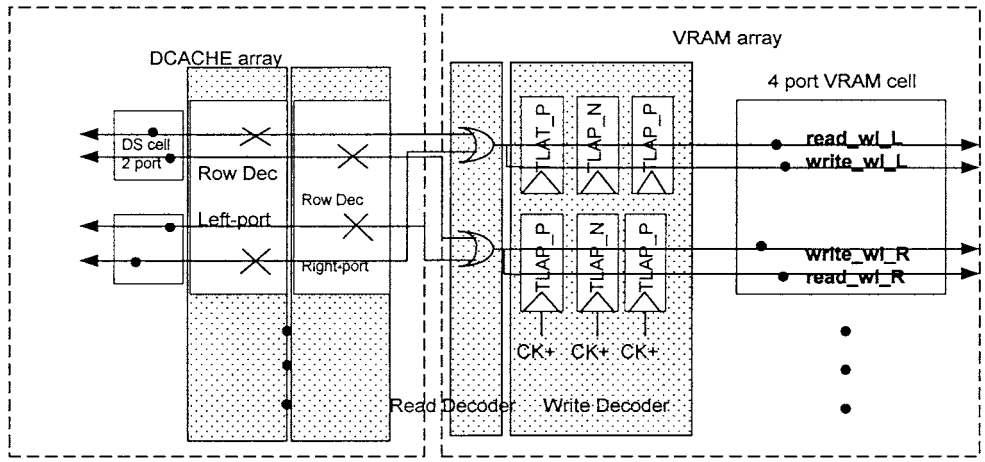


FIGURE 50.17 VRAM row decoder. (From Ref. 15.)

Although the best way to verify the whole chip layout is to do DRC (design rule check) and LVS (layout versus schematic) check that includes all sections and the chip, it was not possible in TFP since the transistor count is too large for CAD tools to handle. Thus, it was necessary to exclude a large part of the memory cells from the verification flow. To avoid possible mistakes around the boundary of the memory cell array, a few rows and columns were sometimes retained on each of the four sides of a cell array. In the case when this breaks signal continuity, text is added on the top level of the layout to make a virtual connection, as shown in Fig. 50.18. These works are basically handled by CAD software plus small programming without editing the layout by hand.

Direct testing of large on-chip memory is highly preferable in VLSI because of faster test time and complete test coverage. TFP IU defines cache direct test in JTAG test mode, in which cache address, data, write enable, and select signals are directly controlled from the outside. Thus, very straightforward evaluation is possible. Utilizing 64-bit, general-purpose bus that runs across the chip, the additional hardware for the data transfer is minimized.

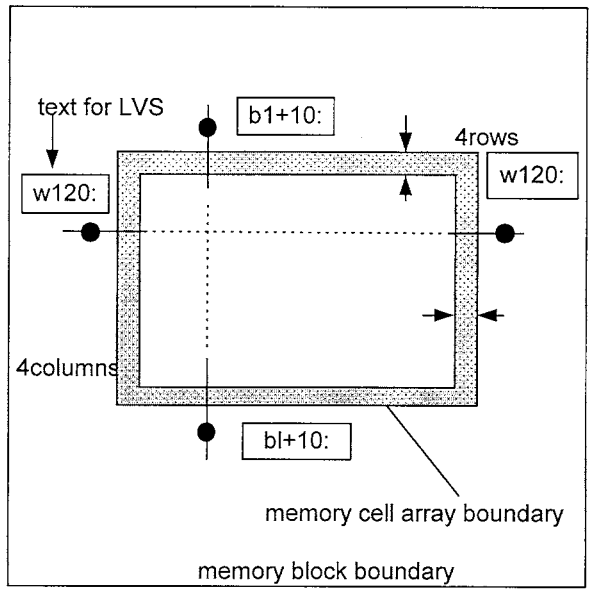


FIGURE 50.18 RAM layout verification. (From Ref. 15.)

Since defect density is a function of device density and device area, large on-chip memory can be a determinant of total chip yield. Raising embedded memory yield can directly lead to the rise of the chip yield. Failure symptoms of the caches have been analyzed by making a fail-bit-map, and this has been fed back to the fabrication process.

References

1. Borel, J., Technologies for Multimedia Systems on a Chip. In *1997 International Solid State Circuits Conference, Digest of Technical Papers*, 40, 18-21, Feb. 1997.
2. De Man, H., Education for the Deep Submicron Age: Business as Usual? In *Proceedings of the 34th Design Automation Conference*, p. 307-312, June 1997.
3. Iizuka, T., Embedded Memory: A Key to High Performance System VLSIs. *Proceedings of 1990 Symposium on VLSI Circuits*, p. 1-4, June 1990.

4. Horowitz, M., Hennessy, J., Chow, P., Gulak, P., Acken, J., Agrawal, A., Chu, C., McFarling, S., Przybylski, S., Richardson, S., Salz, A., Simoni, R., Stark, D., Steenkiste, P., Tjiang, S., and Wing, M., A 32b Microprocessor with On-chip 2K-Byte Instruction Cache. *ISSCC Dig. of Tech. Papers*, p. 30-31, Feb. 1987.
5. Wehn, N. and Hein, S., Embedded DRAM architectural trade-offs. *Proceedings of Design, Automation and Test in Europe*, p. 704-708, 1998.
6. Przybylski, S. A., New DRAM Technologies: A Comprehensive Analysis of the New Architectures. Report, 1996.
7. Wada, Y., Maruyama, T., Chida, M., Takeda, S., Shinada, K., Sekiguchi, K., Suzuki, Y., Kanzaki, K., Wada, M., and Yoshikawa, M., A 1.7-Volt Operating CMOS 64 KBit E2PROM. *Symp. on VLSI Circ., Kyoto, Dig. of Tech. Papers*, p. 41-42, May 1989.
8. Matsukawa, M., Morita, S., Shinada, K., Miyamoto, J., Tsujimoto, J., Iizuka, T., and Nozawa, H., A High Density Single Poly Si Structure EEPROM with LB (Lowered Barrier Height) Oxide for VLSI's. *Symp. on VLSI Technology, Dig. of Tech. Papers*, p. 100-101, 1985.
9. Sawada, K., Sakurai, T., Nogami, K., Iizuka, T., Uchino, Y., Tanaka, Y., Kobayashi, T., Kawagai, K., Ban, E., Shiotari, Y., Itabashi, Y., and Kohyama, S., A 72K CMOS Channelless Gate Array with Embedded 1Mbit Dynamic RAM. *IEEE CICC, Proc.* 20.3.1, May 1988.
10. Archer, D., Deverell, D., Fox, F., Gronowski, P., Jain, A., Leary, M., Olesin, A., Persels, S., Rubinfeld, P., Schmacher, D., Supnik, B., and Thrush, T., A 32b CMOS Microprocessor with On-Chip Instruction and Data Caching and Memory Management. *ISSCC Digest of Technical Papers*, p. 32-33; Feb. 1987.
11. Beyers, J. W., Dohse, L. J., Fucetola, J. P., Kochis, R. L., Lob, C. G., Taylor, G. L., and Zeller, E. R., A 32b VLSI CPU Chip. *ISSCC Digest of Technical Papers*, p. 104-105, Feb. 1981.
12. Ishimoto, S., Nagami, A., Watanabe, H., Kiyono, J., Hirakawa, N., Okuyama, Y., Hosokawa, F., and Tokushige, K., 256K Dual Port Memory. *ISSCC Digest of Technical Papers*, p. 38-39, Feb. 1985.
13. Sakurai, T., Nogami, K., Sawada, K., Shirotori, T., Takayanagi, T., Iizuka, T., Maeda, T., Matsunaga, J., Fuji, H., Maeguchi, K., Kobayashi, K., Ando, T., Hayakashi, Y., and Sato, K., A Circuit Design of 32Kbyte Integrated Cache Memory. *1988 Symp. on VLSI Circuits*, p. 45-46, Aug. 1988.
14. Otomo, G., Hara, H., Oto, T., Seta, K., Kitagaki, K., Ishiwata, S., Michinaka, S., Shimazawa, T., Matsui, M., Demura, T., Koyama, M., Watanabe, Y., Sano, F., Chiba, A., Matsuda, K., and Sakurai, T., Special Memory and Embedded Memory Macros in MPEG Environment. *Proceedings of IEEE 1995 Custom Integrated Circuits Conference*, p. 139-142, 1995.
15. Takayanagi, T., Sawada, K., Sakurai, T., Parameswar, Y., Tanaka, S., Ikumi, N., Nagamatsu, M., Kondo, Y., Minagawa, K., Brennan, J., Hsu, P., Rodman, P., Bratt, J., Scanlon, J., Tang, M., Joshi, C., and Nofal, M., Embedded Memory Design for a Four Issue Superscaler RISC Microprocessor. *Proceedings of IEEE 1994 Custom Integrated Circuits Conference*, p. 585-590, 1994.
16. Patterson, D. et al. Intelligent RAM (IRAM): Chips that Remember and Compute. In *1997 International Solid State Circuits Conference, Digest of Technical Papers*, 40, 224-225, February 1997.
17. Hsu, P., Silicon Graphics TFP Micro-Supercomputer Chip Set. *Hot Chips V Symposium Record*, p. 8.3.1-8.3.9, Aug. 1993.
18. Ikumi, N. et al., A 300 MIPS, 300 MFLOPS Four-Issue CMOS Superscaler Microprocessor. *ISSCC 94 Digest of Technical Papers*, Feb. 1994.
19. Unekawa, Y. et al., A 110 MHz/1Mbit Synchronous TagRAM. *1993 Symposium on VLSI Circuits Digest of Technical Papers*, p. 15-16, May 1993.
20. Takayanagi, T. et al., 2.6 Gbyte/sec Cache/TLB Macro for High-Performance RISC Processor. *Proceedings of CICC'91*, p. 10.21.1-10.2.4, May 1991.

Shen R.S., et al."Flash Memories"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

51

Flash Memories

Rick Shih-Jye Shen
Frank Ruei-Ling Lin
Amy Hsiu-Fen Chou
Evans Ching-Song Yang
Charles Ching-Hsiang Hsu
National Tsing-Hua University

51.1 Introduction

In past decades, owing to process simplicity, stacked-gate memory devices have become the mainstream in the non-volatile memory market. This chapter is divided into seven sections to review the evolution of stacked-gate memory, device operation, device structures, memory array architectures, and flash memory system. In Section 51.2, a short historical review of stacked-gate memory device and the current flash device are described. Following this, the current–voltage characteristics, charge injection/ejection mechanisms, and the write/erase configurations are mentioned in detail. Based on the descriptions of device operation, some modifications in the memory device structure to improve performance are addressed in Section 51.4. Following the introductions of single memory device cells, descriptions of the memory array architectures are employed in Section 51.6 to facilitate the understanding of device operation. In Section 51.7, a table lists the history of flash memory development over the past decade. Finally, Section 51.8 is dedicated to the issues related to implementation of a flash memory system.

51.2 Review of Stacked-Gate Non-Volatile Memory

The concept of a memory device with a floating gate was first proposed by Kahng and Sze in 1967.[1] The suggested device structure was started from a basic MOS structure. As shown in Fig. 51.1, the insulator in the conventional MOS structure was replaced with a thin oxide layer (I1), an isolated metal layer (M1), and a thick oxide layer (I2). These stacked oxide and metal layers led to the so-called MIMIS structure. In this

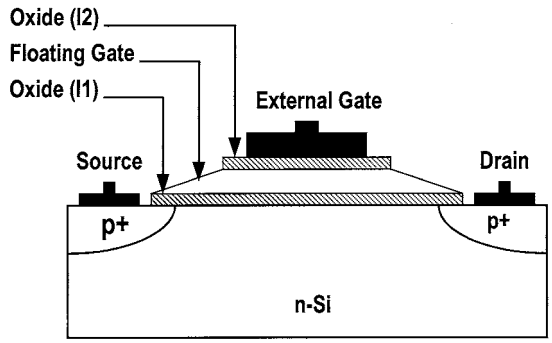


FIGURE 51.1 Schematic cross-section of MIMIS structure.

device structure, the first insulator layer I1 had to be thin enough to allow electrons injected into the floating gate M1. Besides, the second insulator layer I2 is required to be thick enough to avoid the loss of stored charge during charge injection operation. During electron injection operation, a high electric field (~10 MV/cm) enables the electron tunneling through I1 directly, and the injected electrons are captured in the floating gate and thus change the I–V characteristics. On the other hand, a negative voltage is applied at the external gate to remove the stored electrons during the discharge operation by the same direct tunneling mechanism. Owing to the very thin oxide layer I1, the defects in the oxide and the back tunneling phenomena lead to a poor charge retention capability. However, this MIMIS structure demonstrated, for the first time, the possibility of implementation of non-volatile memory device based on the MOS structure.

After MIMIS was invented, several improvements were proposed to enhance the performance of MIMIS. One was the utilization of dielectric material with a large amount of electron-trapping centers as a replacement of the floating metal gate.[2,3] The injected electrons would be trapped in the bulk and also at the interface traps in the dielectric material, such as silicon nitride (Si_3N_4), Al_2O_3, Ta_2O_5. The device structure with these insulating layers as electron storage node was referred as a *charge trapping device*. Another solution to improve the oxide quality and charge retention capability was the increase of the thickness of the tunnel dielectric I1. This device structure based on the MIMIS structure but with a thicker insulating layer was also referred as *floating gate device*.

In the initial development period, the charge trapping devices had several advantages compared with floating gate devices. They allowed high density, good write/erase endurance capability, and fast programming/erase time. However, the main obstacle for the wide application in charge trapping devices was the poorer charge retention capability than in floating gate devices. On the other hand, the floating gate devices showed a major drawback of not being electrically erasable. Therefore, the erase operation had to be preceded by the time-consuming UV-irradiation process. However, the floating gate devices had been applied successfully because of the following advantages and improvements. First, the floating gate devices were compatible with the standard double polysilicon NMOS process and then became compatible with CMOS process after minor modification. Second, an excellent charge retention capability was obtained because of the thicker gate oxide. Besides, the thicker oxide leads to a relieved gate disturbance issue. Furthermore, the development of electrical erase operation technique during the 1980s made the write/erase operation easier and more efficient. Based on these reasons, most commercial non-volatile memory companies focused their research efforts on the floating gate devices. Therefore, floating gate devices have become the mainstream product in the non-volatile market.

A high operation voltage is unavoidable when the thickness of oxide I1 increases in MIMIS structure. Thus, another way to achieve electron injection was necessary to make the injection operation more efficient. In 1971, the introduction of a memory element with avalanche injection scheme was demonstrated.[4] This first operating floating gate device — named Floating gate Avalanche injection MOS (FAMOS), as shown in Fig. 51.2 — was a p-channel MOSFET in which no electrical contact was made

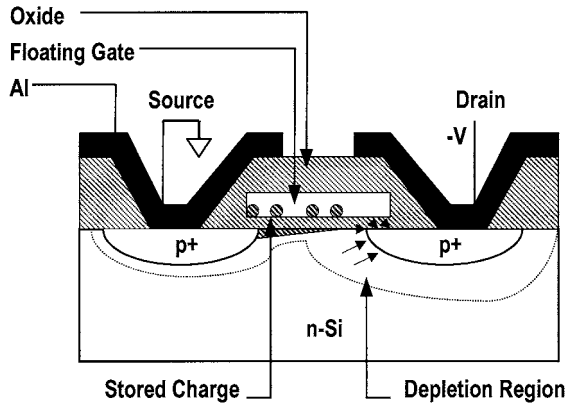


FIGURE 51.2 Schematic cross-section of FAMOS structure.

to the silicon gate. The injection operation of the FAMOS memory structure is initiated by avalanche phenomena in the drain region underneath the gate. The electron-hole pair generation is caused by applying a high reversed bias at the drain/substrate junction. Some of generated electrons drift toward the floating gate by the positive oxide field which is induced by the capacitive coupling between floating gate and drain. However, the inefficient injection process was the major drawback in this device structure.

In order to improve the injection efficiency, the Stacked-gate Avalanche injection MOS (SAMOS) with an external gate was proposed, as shown in Fig. 51.3. Owing to the additional gate bias, the programming speed was improved by an increased drift velocity of electrons in the oxide and the field induced energy barrier lowering at the Si–SiO_2, interface. Besides, by employing this control gate, the electrical erase operation became possible by building up a high electric field across the inter-polysilicon dielectric.

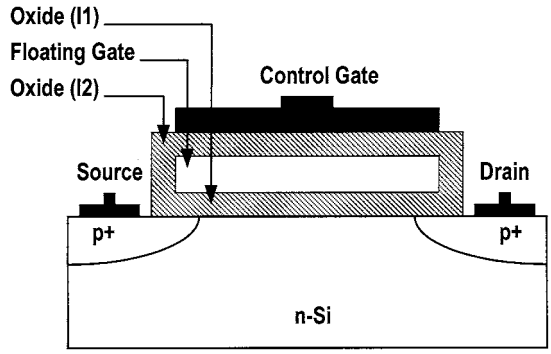


FIGURE 51.3 Schematic cross-section of p-channel SAMOS structure.

All the stacked-gate devices mentioned above are p-channel devices, which utilize avalanche injection scheme. However, if a smaller access time is required for the read operation, n-channel devices are necessary because of higher channel carrier mobility. Since the avalanche injection in an n-channel device is based on the hole injection, other injection mechanisms are required for n-channel stacked-gate memory cells. There are two major injection schemes for the n-channel memory cell. One is the channel hot electron injection (CHEI) and the other one is high electric field (Fowler-Nordheim, FN) tunneling mechanism. These two operation schemes lead to different device structures. The memory devices using

the CHEI scheme allow a thicker gate oxide, whereas the memory devices using FN tunneling scheme require thinner oxide. In 1980, researches at Intel Corp. proposed the FLOTOX (FLOating gate Tunnel OXide) device, as shown in Fig. 51.4, in which the electrons are injected into and ejected from the floating gate through a high-quality thin oxide region outside the channel region.[5] The FLOTOX cell must be isolated by a select transistor to avoid the over-erase issue and therefore it consists of two transistors. Although this limits the density of such memory in comparison with EPROM and the Flash cell, it enables the byte-by-byte erase and reprogramming operation without having to erase the entire chip or sector. Based on this, the FLOTOX cell is suitable for the applications in which low density, high reliability, and non-volatile memory are required.

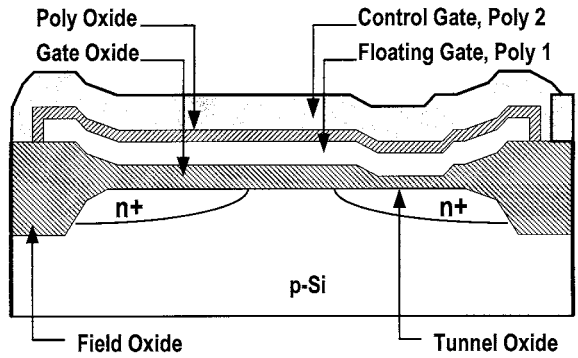


FIGURE 51.4 Schematic cross-section of FLOTOX structure.

Another modification of operation from EEPROM is the erase of the whole memory chip instead of erasing a byte. By using an electrical erase signal, all cells in the memory chip, which is called a Flash device, are erased simultaneously. The first Flash memory cell was proposed and realized in a three-layer polysilicon technology by Toshiba Corp.[6] The first polysilicon is used as the erase gate, the second polysilicon as the floating gate, and the third polysilicon as the control gate, as shown in Fig. 51.5(c). In this device, programming operation is performed by channel hot electron injection and erase operation is carried out by extracting the stored electron from the floating gate to erase gate for all the bits at the same time.

51.3 Basic Flash Memory Device Structures

n-Channel Flash Cell

Based on the concept proposed by researchers at Toshiba Corp., the developments in Flash memory have burgeoned since the end of 1980s. There are three categories of device structures based on the n-channel MOS structure. Besides the triple polysilicon Flash cell, the most popular Flash cell structures are the ETOX cell and the split-gate cell.

In 1985, Mukherjee et. al.[7,9] proposed a source-erase Flash cell called the ETOX (EPROM with Tunnel OXide). This cell structure is the same as that of the UV-EPROM, as shown in Fig. 51.6, but with a thin tunnel oxide layer. The cell is programmed by CHEI and erased by applying a high voltage at the source terminal.

A split-gate memory cell was proposed by Samachisa et. al. in 1987.[8] This split-gate Flash cell with a drain-erase type has two polysilicon layers, as shown in Fig. 51.7. The cell can be regarded as two transistors in series. One is a floating gate memory, which is similar to an EPROM cell; the other, which is used as a select transistor, is an enhancement transistor controlled by the control gate.

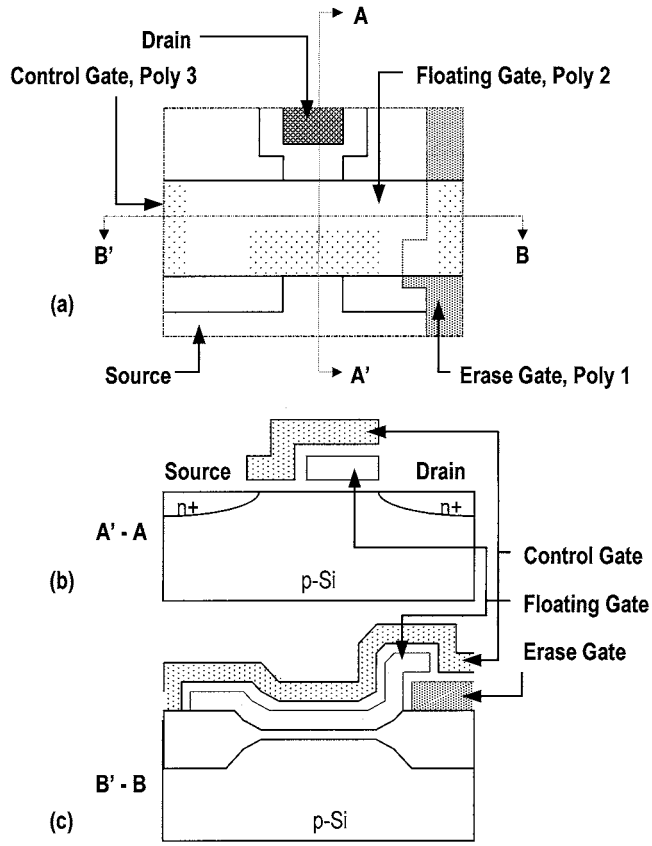


FIGURE 51.5 Tripe-gate Flash memory structure proposed by Toshiba: (a) layout of the cell; (b) cross-section along the channel length, and (c) cross-section along the channel width.

p-Channel Flash Cell

The p-channel Flash memory cell was first proposed by Hsu et. al. in 1992.[9] Recently, several studies have been done on this device structure.[10–13] This Flash cell structure is similar to the ETOX cell but with p-channel. The erase mechanism is still by FN tunneling. As to the electron injection, there are two injection schemes that can be employed: CHEI and BBHE (Band-to-Band tunneling induced Hot Electron injection).[11] The p-channel Flash cell features high electron injection efficiency, scalability, immunity to the hot hole injection and reduced oxide field during programming. Based on these advantages, the p-channel Flash memory cell seems to reveal a high potential for future low-power Flash applications.

51.4 Device Operations

Device Characteristics

Capacitive Coupling Effects and Coupling Ratios

The I–V characteristics of stacked gate can be derived from the MOSFET characteristics accompanying with the capacitive-coupling factors. For a stacked-gate device, the device structure can be depicted as an equivalent capacitive circuit, as shown in Fig. 51.8. Owing to being isolated from other terminals, the potential of floating gate, V_{FG}, can be expressed as not only the total contributions from four terminals of the device, but also from the contribution of the stored charge in the floating gate:

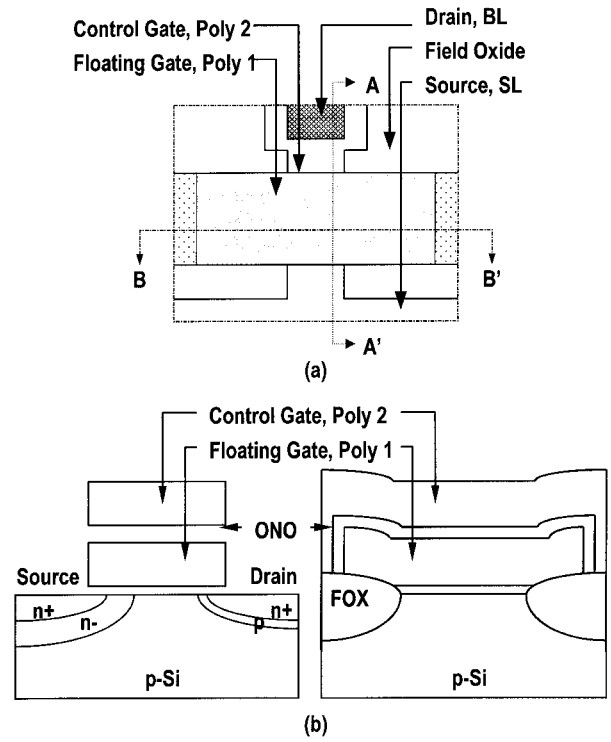


FIGURE 51.6 Schematic cross-section of ETOX-type Flash memory cell: (a) the top view of the cell, and (b) the cross-section along the channel length and channel width.

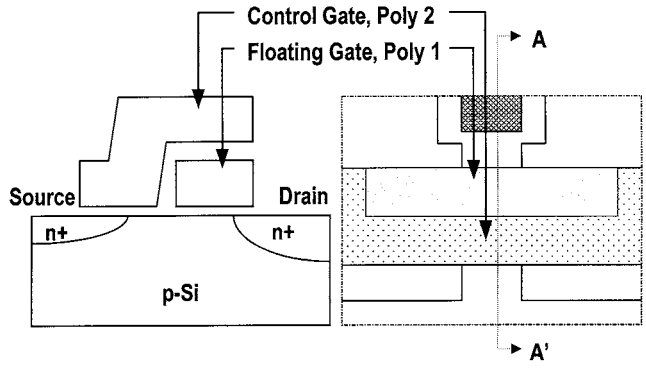


FIGURE 51.7 Schematic cross-section of split-gate Flash memory cell.

$$V_{FG} = \frac{C_{FG}}{C_{TOTAL}} V_G + \frac{C_B}{C_{TOTAL}} V_{WELL} + \frac{C_D}{C_{TOTAL}} V_D + \frac{C_S}{C_{TOTAL}} V_S - \frac{Q}{C_{TOTAL}} \tag{51.1}$$

$$C_{TOTAL} = C_{FG} + C_B + C_D + C_S \tag{51.2}$$

and

$$\alpha_{FG} = \frac{C_{FG}}{C_{TOTAL}}, \alpha_B = \frac{C_B}{C_{TOTAL}}, \alpha_D = \frac{C_D}{C_{TOTAL}}, \alpha_S = \frac{C_S}{C_{TOTAL}} \tag{51.3}$$

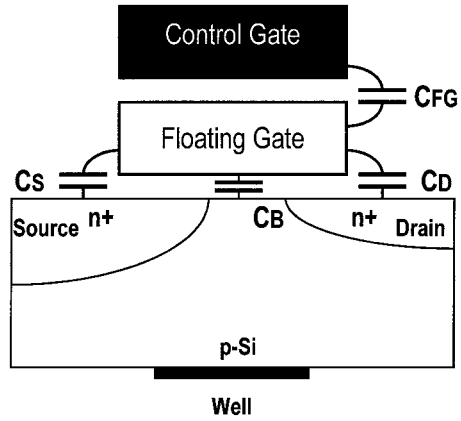


FIGURE 51.8 Schematic cross-section of stacked-gate device and its equivalent capacitive model.

where C_{FG}, C_B, C_D, and C_S are the capacitances between floating gate and control gate, well terminal, drain terminal, and source terminal, respectively. Q is the charge stored on the floating gate, and α_{FG}, α_B, α_D, α_S are the gate, well, drain, and source coupling ratios, respectively.

Current–Voltage Characteristics

The current–voltage relationship in a stacked-gate device has been studied and modeled in detail.[14,15] By employing Eq. 51.1 for general I–V characteristics in MOSFETs, a simplified I-V relationship in stacked gate devices can be obtained:

$$\begin{aligned} V_{FG} &= \frac{C_{FG}}{C_{TOTAL}} V_G + \frac{C_D}{C_{TOTAL}} V_D - \frac{Q}{C_{TOTAL}} \\ &= \alpha_{FG}\left(V_G + \frac{C_D}{C_{FG}} V_D - \frac{Q}{C_{FG}} \right) \\ \text{for } V_S &= V_{WELL} = 0\,V \end{aligned} \tag{51.4}$$

In the linear region,

$$\begin{aligned} I_D &= \frac{\mu n \cdot C_{ox} \cdot W}{L}\left(V_{FG} - V_{TH} - \frac{V_D}{2} \right) \cdot V_D \\ &= \frac{\alpha_{FG} \cdot \mu n \cdot C_{ox} \cdot W}{L}\left[V_G + \left(\frac{C_D}{C_{FG}} - \frac{1}{2} \right) V_D - \frac{Q}{C_{FG}} - \frac{V_{TH}}{\alpha_{FG}} \right] V_D \end{aligned} \tag{51.5}$$

And also in saturation region,

$$\begin{aligned} I_D &= \frac{\mu n \cdot C_{ox} \cdot W}{2L} (V_{FG} - V_{TH})^2 \\ &= \frac{\alpha_{FG}{}^2 \cdot \mu n \cdot C_{ox} \cdot W}{2L}\left(V_G + \frac{C_D}{C_{FG}} V_D - \frac{Q}{C_{FG}} - \frac{V_{TH}}{\alpha_{FG}} \right)^2 \end{aligned} \tag{51.6}$$

From Eqs. 51.5 and 51.6, it is clearly demonstrated that the stacked-gate device suffers from drain bias coupling during operation. An increase of drain current can be observed, both in output characteristics and transfer characteristics. Fig. 51.9 shows the subthreshold characteristics of both the n-channel and p-channel Flash devices. An obvious increase of the subthreshold current can be observed while the drain

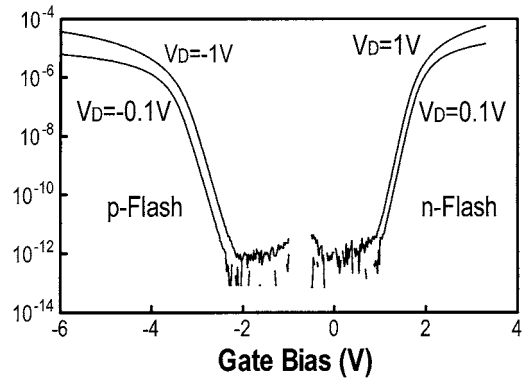


FIGURE 51.9 The subthreshold characteristics of n- and p-channel Flash memory cells.

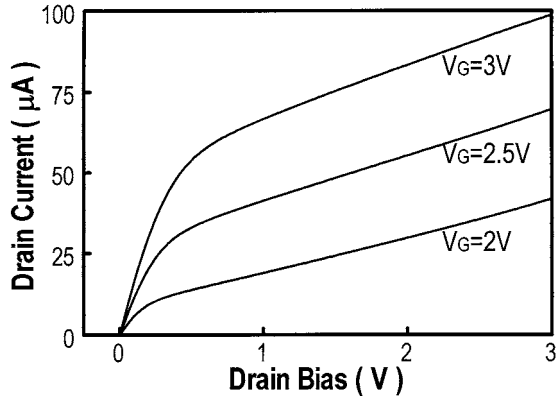


FIGURE 51.10 The output characteristics of stacked-gate memory cells.

bias increases. In addition, the increased drain current characteristics in the saturation region are shown in Fig. 51.10.

Threshold Voltage of Flash Memory Devices

Threshold voltage is defined as the minimum voltage needed to turn on the device. For a stacked-gate device, the threshold voltage measured from the control gate is an indicator of charge storage condition. From Eq. 51.4, we can obtain

$$V_{FGTH} = \alpha_{FG}\left(V_{GTH} + \frac{C_D}{C_{FG}}V_D - \frac{Q}{C_{FG}}\right) \tag{51.7}$$

According to this equation, there exists a linear relationship between threshold voltage measured from floating gate and control gate, drain bias, and stored charge amount. The threshold voltage measured from the floating gate is only determined by the process procedures and device structures. Therefore, the change of the threshold voltage measured from control gate linearly depends on the change of the stored charge amount under a fixed drain bias in a specific stacked-gate device. Thus, this can be expressed as

$$\Delta V_{GTH} = \frac{\Delta Q}{C_{FG}} \tag{51.8}$$

Based on this relationship, the amount of charge storage in stacked-gate memory cell can be monitored by the measured threshold voltage. As shown in Fig. 51.11, the transfer characteristic shifts toward a higher gate bias region, while the increasing amount of electrons are stored in the floating gate for both n- and p-channel Flash memory cells. Thus, device conduction during read operation determines the stored information of the stacked-gate devices. At a specific gate bias condition for reading, as shown in Fig. 51.11, the memory with/without stored charge would lead to different amounts of drain current. The stored electron in the floating gate leads no current flow through the channel at the "READ" bias in the n-channel Flash cell, whereas the channel would conduct at the read operation for the p-channel cell with the electron stored in the floating gate. The sense amplifier in the peripheral circuit can detect the drain current and provide the stored information for external applications.

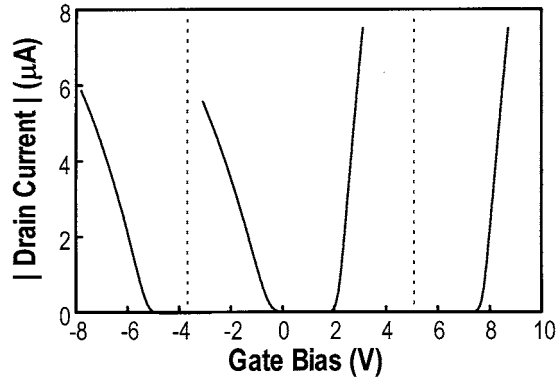


FIGURE 51.11 The transfer characteristics of n- and p-channel Flash memory cells.

Carrier Transport Schemes

Transport of charge through the oxide layer is the basic mechanism that permits operation of stacked-gate memory devices. It makes possible charging and discharging of the floating gate. In order to achieve the write/erase operations, the charge must move across the potential barrier built by the insulating layers between floating gate and other terminals of the memory device. There are different charge transport mechanisms and they can be categorized by the charge energy:[16]

1. Charges with sufficiently high energy can surmount the Si–SiO_2 potential barrier, including:
   a. Hot electrons initiated from substrate avalanche
   b. Hot electrons in a junction (initiated from p-n junction avalanche)
   c. Thermally excited electrons (thermionic emissions and Schottky effect)
   d. "Lucky" electrons at the drain side (Auger scattering)
2. Charges with lower energy can cross the barrier by quantum mechanical tunneling effects:
   a. Trap-assisted tunneling through sites located within the barrier
   b. Direct tunneling when the tunneling distance is equal to the thickness of the oxide
   c. Fowler-Nordheim (FN) tunneling

Hot carrier injection and FN tunneling injection are the common charge injection mechanisms in Flash memory cells. In this section, these charge injection mechanisms will be described in more detail.

Channel Hot Electron Injection (CHEI)

Figure 51.12 shows the schematic diagram of the CHEI for n- and p-channel MOSFET. When applying a high voltage at the drain terminal of an on-state device, electrons moving from the source terminal to the drain side are accelerated by the high lateral channel electric field near the drain terminal. Figure

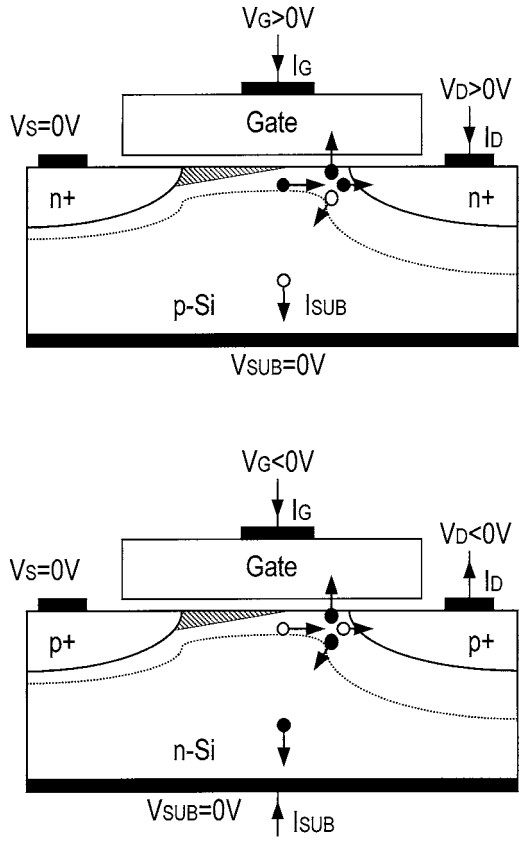


FIGURE 51.12 Schematic illustration of the channel hot carrier effect in (a) n-channel MOSFET, and (b) p-channel MOSFET.

51.13 shows the plots of simulated electric field along the channel region. Notice that the electric field increases abruptly in the pinch-off region when the location approaches the drain terminal. Under the oxide field, which is favorable for attracting electrons, part of the heated electrons gain enough energy to surmount the Si–SiO_2 potential barrier and inject into the gate terminal.

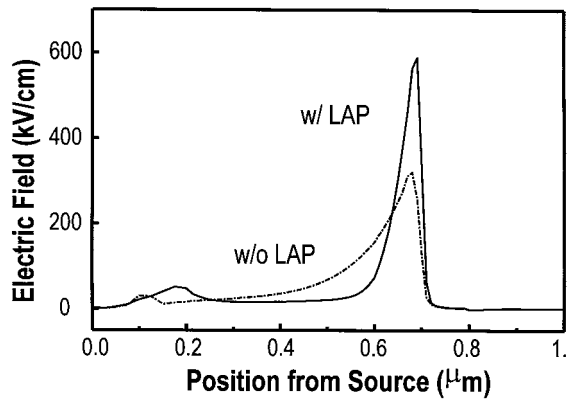


FIGURE 51.13 Simulated electric field along the channel in the n-channel MOSFET.

Figure 51.14 shows the qualitative plot of gate current characteristic for n-channel MOSFETs. For the gate bias in the region "I", a quite small gate current can be characterized. In this subthreshold region, the carrier injection mainly originates from the avalanche injection, which will be discussed in the next section. In region II, the channel conducts and the channel current increases as the gate bias increases and thus the gate current induced by CHEI increases. As the gate bias increases further, the gate current peaks at a high gate bias. Following the peak value of the gate current, the decreasing gate current is mainly caused by the decrease of the lateral electric field, as illustrated in region III.

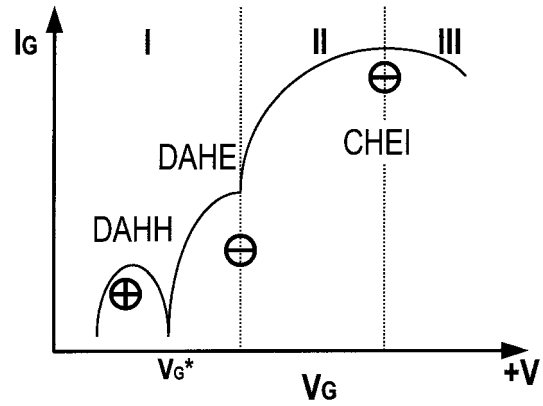


FIGURE 51.14 Schematic gate current behavior in n-channel MOSFET.

On the other hand, the measured gate current characteristic in p-channel MOSFETs is shown in Fig. 51.15. Owing to the large potential barrier and short mean free path, the hot hole generated and accelerated in the channel cannot gain enough energy to surmount the oxide barrier. Thus, electron current initiated by channel hot electrons is still the dominant component of gate current in the p-channel MOSFET.[17,18] Besides, the gate current peaks at a lower gate bias in a p-channel MOSFET and has a larger peak value than that in an n-channel MOSFET. In larger gate bias regions, the gate current is dominated by hole injection, which may be caused by the oxide field favoring the injection of the conducting holes into the gate terminal.[19]

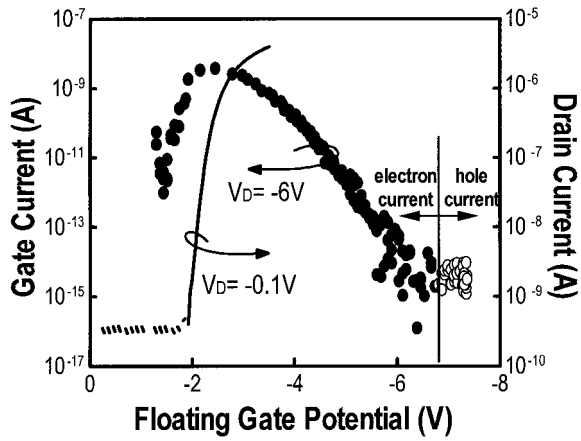


FIGURE 51.15 The gate current behavior of p-channel MOSFET measured from the threshold voltage shift of the stacked-gate structure.

In the 1980s, there were several approaches to describe the channel hot electron injection into the gate terminal. Takeda, et al.[20] modeled the gate current in n-channel MOSFETs as thermionic emission from the heated electron gas over the Si–SiO_2 potential barrier. This thermionic gate current model, referred as the "effective electron temperature model," assumes that the heated electrons become an electron gas with a Maxwellian distribution with an effective temperature $T_e(x)$. The temperature $T_e(x)$ depends on the electric field and the location in the channel. The gate current is given by

$$J_G = q \cdot n_S \cdot \left(\frac{kT_e}{2\pi m^*}\right)^{1/2} \cdot \exp\left(-\frac{\Phi_B}{k \cdot T_e}\right) \cdot \exp\left(-\frac{d}{l}\right) \tag{51.9}$$

where n_s is the surface electron density, k is the Boltzmann constant, m^* is the effective electron mass, Φ_B is the Si–SiO_2 potential barrier, d is the distance of the electron from the interface at $T_e(x)$, and the λ is the mean free path. The last term in Eq. 51.9 accounts for the probability of energy loss due to the collision while the electron moves toward the Si–SiO_2 interface.

Another gate current model, the lucky electron model, is based on the assumption that an electron is injected into oxide by obtaining enough energy from the lateral channel electric field without suffering any collision. The lucky electron approach for hot electron injection was originated by Shockley[21] and Verway et. al.,[22] who applied it in the study of substrate hot electron injection in MOSFETs and subsequently refined and verified by Ning et. al.[23] Hu modified the substrate lucky electron injection model and applied it to CHEI in MOSFETs.[24] In this model, there are three probabilities to describe the physical mechanism responsible for CHEI gate current.[25] They are (1) the probability of a hot electron to gain enough kinetic energy and normal momentum, (2) the probability of not suffering any inelastic collision during transport to the Si–SiO_2 interface, and (3) the probability of not suffering collision in oxide image-potential well. Thus, the gate current originated from CHEI is given by

$$I_G = \int_0^L I_D \frac{(P_1 \cdot P_2 \cdot P_3)}{\lambda r} dx \tag{51.10}$$

where I_D is the channel current, L is the channel length, and λ_r is the redirection scattering mean free path. P_1 is the probability that an electron can gain the energy equals the energy barrier under the channel electric field E without suffering optical phonon scattering and can be expressed as

$$P_1 = \exp\left(-\frac{\Phi_B}{E\lambda}\right) \tag{51.11}$$

where λ is the mean free path for optical phonon scattering. P_2 is the probability of not suffering any inelastic collision during transport to the Si–SiO_2 interface and can be expressed as

$$P_2 = \frac{\int_{y=0}^{\infty} n(y) \cdot \exp\left(-\frac{y}{\lambda}\right) dy}{\int_{y=0}^{\infty} n(y)\, dy} \tag{51.12}$$

The last probability factor is the scattering in the oxide image-potential well. P_3 can be expressed as[26]:

$$P_3 = \exp\left(-\frac{y_o}{\lambda_{ox}}\right) \tag{51.13}$$

Ong et al. modified the lucky electron model to analyze the hot electron injection effects in p-channel MOSFETs.[27,28] Based on Eq. 51.10 and substituting substrate current (I_{SUB}) for drain current (I_D), the gate current in p-channel MOSFETs can be expressed as:

$$I_G = \int_{y=0}^{y=L} I_{SUB} \frac{(P_1 \cdot P_2 \cdot P_3)}{\lambda r} dy \tag{51.14}$$

After describing the channel hot electron injection mechanisms, the charge injection characteristics based on the CHEI scheme are discussed. First, the output characteristics (I_D–V_D) of a memory cell are taken into account. The output characteristic of a stacked-gate device can be regarded as an injection indicator to examine the effects of channel hot electron injection under different device operation conditions and device structures. The output characteristics of the n-channel Flash memory under a high gate bias are shown in Fig. 51.16(a). The drain current rolls off at a lower drain bias as the channel length of the device decreases. This indicates obviously that the channel length reduction results in the increase of the lateral channel electric field and therefore the enhancement of hot electron injection. As the electron injection initiates, the stored electrons retard the conduction of the channel and the device is gradually turned off owing to the continuous electron injection. On the contrary, the output characteristics in the p-channel Flash memory, as shown in Fig. 51.16(b), reveal a quite different I–V behavior after electron injection. Owing to the reduction of threshold voltage after electron injection, the enhancement of further channel conduction can be observed as the drain bias increases.

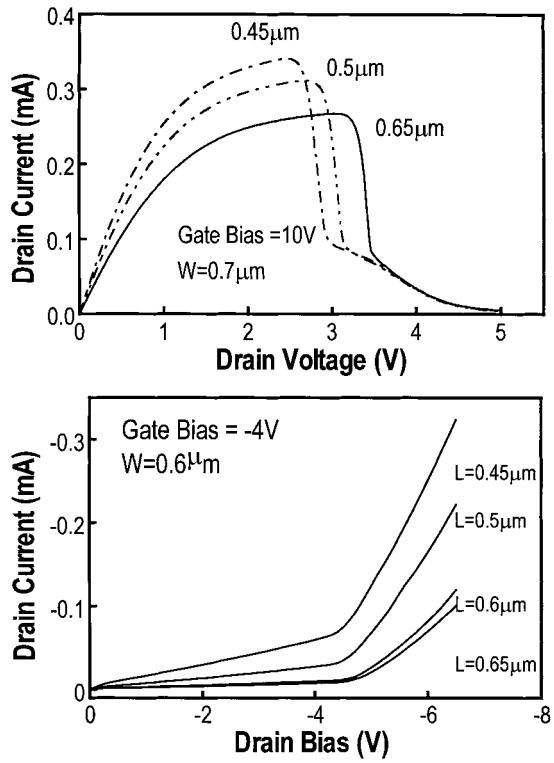


FIGURE 51.16 (a) The output characteristics of the n-channel Flash memory at high gate bias, and (b) the output characteristics of the p-channel Flash memory at high gate bias.

Second, the programming characteristics of the n- and p-channel Flash memory are demonstrated. Figure 51.17(a) shows the gate bias effects on the CHEI programming characteristics in an n-channel Flash memory cell. The threshold voltage increases as the electron injection process prolongs and then saturates at different values for different gate biases. On the other hand, Fig. 51.17(b) shows the CHEI programming characteristics in a p-channel Flash memory cell. Compared with the n-channel cell, the programming characteristic in the p-channel Flash cell reveals a large dependence on the gate bias condition. This is mainly caused by the CHEI that distributes within a narrower gate bias condition. The gate current in the p-MOSFET peaks at lower gate bias and decreases steeply when the gate bias becomes more negative. Therefore, the injected electrons during programming accompanied by the control gate bias lead to a more negative floating gate potential and the programming behavior is quite different at different gate bias conditions.

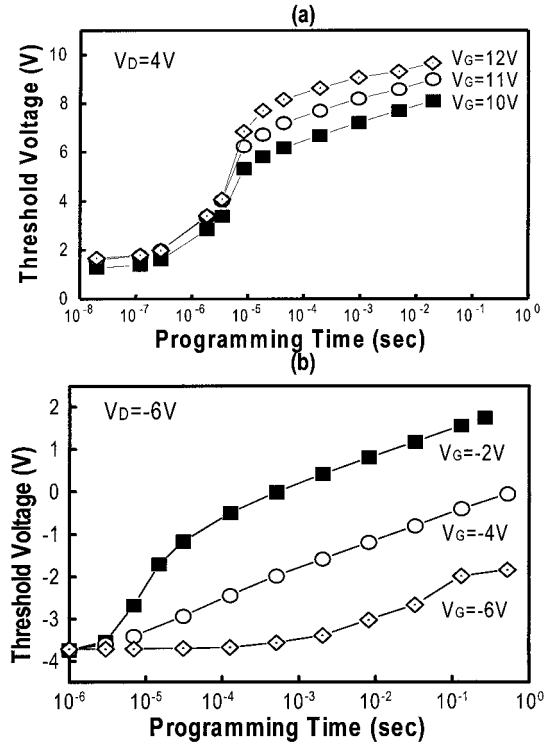


FIGURE 51.17 (a) The programming characteristics of the n-channel Flash memory using channel hot electron injection scheme; (b) the programming characteristics of the p-channel Flash memory using channel hot electron injection.

Drain Avalanche Hot Carrier (DAHC) Injection

As shown in the region I of Fig. 51.14, the characteristic of the gate current is still a function of the gate voltage in n-channel MOSFETs. When V_G is smaller than V_G^*, drain avalanche hot hole (DAHH) is the dominant carrier injected into the gate. On the other hand, when V_G is larger than V_G^*, drain avalanche hot electron (DAHE) is the dominant carrier injected into the gate terminal. V_G^* is the point at which the amounts of the injected hot hole and injected hot electron are in balance. At this gate bias condition, the gate current is not observed.

Conceptually, the existence of hot hole injection seems questionable because of the high barrier (3.8 eV) for hole injection at the Si–SiO_2 interface. However, hot hole gate currents have been experimentally identified and modeled.[29,32] Hofmann et. al.[30] employed the effective electron temperature model[20] and

the concept of oxide scattering effects[25] based on the two-dimensional distribution of electric field, charge carrier, and current density calculated by computer simulator. The hot hole injection and hot electron injection initiated by the avalanche generation were manifested qualitatively. Saks et al.[32] proposed a modified floating gate technique to characterize these extremely small gate currents. It showed that a small positive gate current exists for gate bias near the threshold voltage. They also suggested that the hole current increases with increasing drain bias and decreasing effective channel length, which is analogous to the dependencies for channel hot electron injection. Comparison of hot hole and hot electron gate current as a function of the effective channel length also suggested that the lateral electric field near the drain plays an important role in the hole injection.

In the stacked-gate devices, in the DAHH region, holes are injected into the floating gate, which increases the floating gate voltage gradually, and finally the floating gate voltage reaches the point V_G^*. On the contrary, in the DAHE region, electrons are injected into the floating gate, which decreases the floating gate, and the floating gate voltage also reaches the point V_G^*. Thus, the threshold voltage of the stacked-gate device would distribute at a specific value after the DAHC injection operation. As shown in Fig. 51.18, the threshold voltage of the flash cell after a period of DAHC operation time can converge to a specific value. For the cell with a threshold voltage larger than the converged value, the floating gate voltage is more negative than V_G^*, the hole injection occurs and makes the threshold voltage decrease. On the other hand, for the cell with a threshold voltage smaller than the converged value, it reveals a more positive potential in the floating gate, the electron injection occurs and increases the threshold voltage. In the Flash application, the DAHC injection is usually applied to the convergent operation.[33] Owing to the process-induced device variations, the electron ejection operation usually causes a wide threshold distribution. Additionally, a trapped hole in the oxide enhances the FN tunneling current and generates the erratic erased cell.[34] By employing the DAHC operation, a tighter threshold voltage distribution can be obtained.[35]

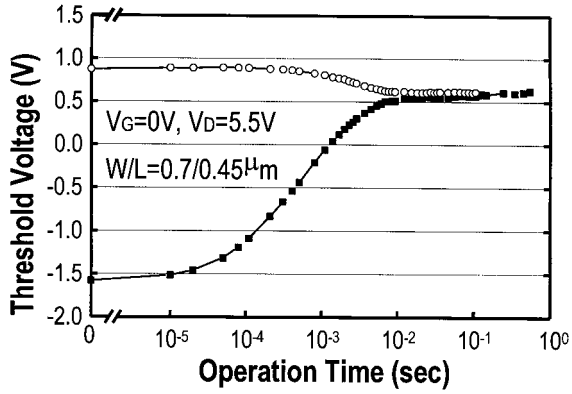


FIGURE 51.18 The convergetn characteristics of the n-channel Flash memory cell with DAHC operation.

Band-to-Band Tunneling Induced Hot Carrier Injection (BBHC)

Carrier injection initiated by band-to-band tunneling accompanied by lateral junction electric field is also an important charge transport mechanism in Flash memory. As shown in Fig. 51.19, the BBHC operation conditions for n- and p-channel lead to different charge injection behaviors. For n-channel MOSFETs, the negative gate bias and positive drain bias lead to the possible hole injection toward the gate terminal. For p-channel MOSFETs, the operation conditions lead to the possible electron injection toward the gate terminal. The initiation of the BBHC injection can be divided into two procedures. One is the band-to-band tunneling, and the other is the acceleration due to lateral electric field and injection due to favorable oxide field.

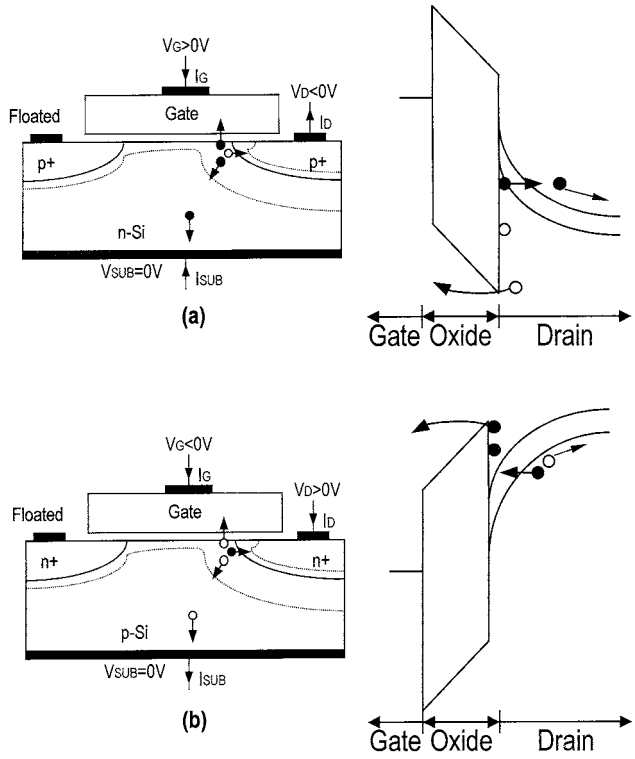


FIGURE 51.19 The schematic illustration for BBHC injection for: (a) n-channel MOSFET, and (b) p-channel MOSFET.

The band-to-band tunneling phenomenon is usually referred as gate-induced drain leakage current.[36] When a high drain voltage is applied with a grounded gate terminal, a deep depletion region is formed underneath the gate-to-drain overlap region. Electron-hole pairs are generated by the tunneling of valence band electrons into the conduction band and then collected by the drain and substrate terminals, separately. Since the minority carriers (hole in n-MOSFET and electron in p-MOSFET) generated by band-to-band tunneling in the drain region flow to the substrate due to the lateral electric field, the deep depletion region is always present and the band-to-band tunneling process proceeds without forming an inversion layer. The band-to-band tunneling characteristic can be estimated by the calculation of electric field distribution and the tunneling probability.[37,38] Based on the depletion approximation and the assumption of uniform impurity distribution, the electric field $E(x)$ in the depletion region is given by

$$E(x) = \frac{Q \cdot N_o}{\varepsilon_{si}} \sqrt{\frac{2 \cdot \varepsilon_{si} \cdot V_{bend}}{q \cdot N_o}} \left(1 - x \sqrt{\frac{q \cdot N_o}{2 \cdot \varepsilon_{si} \cdot V_{bend}}}\right) \tag{51.15}$$

where V_{bend} is the value of the band bending, N_o is the impurity density, and x is the coordinate normal to the Si–SiO_2 interface. The continuity equation at the Si–SiO_2 interface can be expressed as

$$\varepsilon_{si} \cdot E(x = 0) = \varepsilon_{ox} \cdot E_{ox} = \varepsilon_{ox} \frac{V_D - V_{bend}}{T_{ox}} \tag{51.16}$$

The tunneling characteristics are usually approximated by the relationship derived from the reverse biased p-n junction tunnel diode:[39]

$$J = B_1 \cdot E^2 \exp\left(-\frac{B_2}{E}\right) \tag{51.17}$$

where B_1 and B_2 are physical constants. Most of the generated minority carriers are drained away from the substrate terminal. However, owing to the sufficient lateral electric field across the depletion region, these hot carriers may encounter Auger scattering and generate another electron-hole pair.[40] When the drain bias is higher than Si–SiO_2 barrier, the top barrier position seen by the cold generated minority carriers is lower at the depletion edge in the channel. Thus, the injection probability of the minority carrier becomes much higher. The probability of the generated minority carrier injection is given by.[41]

$$\begin{aligned} P_{inject} &= \int \exp\left(-\frac{d(V)}{\lambda}\right) dW(V) \\ &\approx \left(\frac{2V_D}{\Phi_B} - 1\right) \cdot \exp\left(-\frac{\Phi_B}{q \cdot E_m \cdot \lambda}\right) \end{aligned} \tag{51.18}$$

Thus, the injected current accompanied with Eq. 51.17 and oxide scattering factor P expressed in Eq. 51.13 can be given by

$$J_{inject} = P \cdot P_{inject} \cdot J \tag{51.19}$$

In the n-channel MOSFET, the BBHC injection process leads to a significant amount of hot hole injection.[42,43] This situation is mostly encountered in the electron ejection operation of a Flash memory device with "edge" Fowler-Nordheim tunneling. The hole injection into the gate terminal would result in not only the deviation of the memory state, but also severe long-term device instability issues. However, on the contrary, the BBHC injection process leads to the electron injection in the p-channel MOSFET and has been employed in the programming scheme for p-channel Flash memory cell.[10,11] Figure 51.20(a) shows the BBHE characteristics of the p-channel MOSFET. The drain and gate currents monotonically increase with respect to the gate bias because of the increase of the band-to-band tunneling efficiency and the more favorable oxide field for electron injection. Owing to operating in the off state, the electron injection efficiency of the BBHE scheme is much larger than that in the CHEI operation. The BBHE injection reveals a rather high injection efficiency (I_G/I_D) up to 10^{-2}, which provides a quite efficient programming operation for the p-channel Flash cell.[10] Figure 51.20(b) shows the programming characteristics based on the BBHE injection mechanism. The programming time is greatly shortened as the control gate voltage increases. As compared with the CHEI scheme shown in Fig. 51.17(b), the BBHE approach indeed reveals a faster programming speed.

Fowler-Nordheim (FN) Tunneling

The FN tunneling formula proposed by Fowler and Nordheim in 1928 can be described as

$$J_{tunnel} = Co \cdot E^2 \cdot \exp\left(-\frac{4\sqrt{2m^* \cdot \Phi_B^{\,3}}}{3 \cdot q \cdot \hbar \cdot E}\right) \tag{51.20}$$

where J_{tunnel} and E are the tunneling current density and electric field across the oxide layer, respectively. Besides, C_o is a material-dependent constant and m^* is the carrier effective mass. The tunneling theory is developed using the semi-classical independent electron model. For a carrier with energy qU_o, the general expression for the transmission coefficient T_c through on energy barrier depends on the barrier shape $U(x)$, as shown in Fig. 51.21. The value of T_c is derived using the WKB (Wentzel-Kramers-Brillouin) approximation.[44,46]

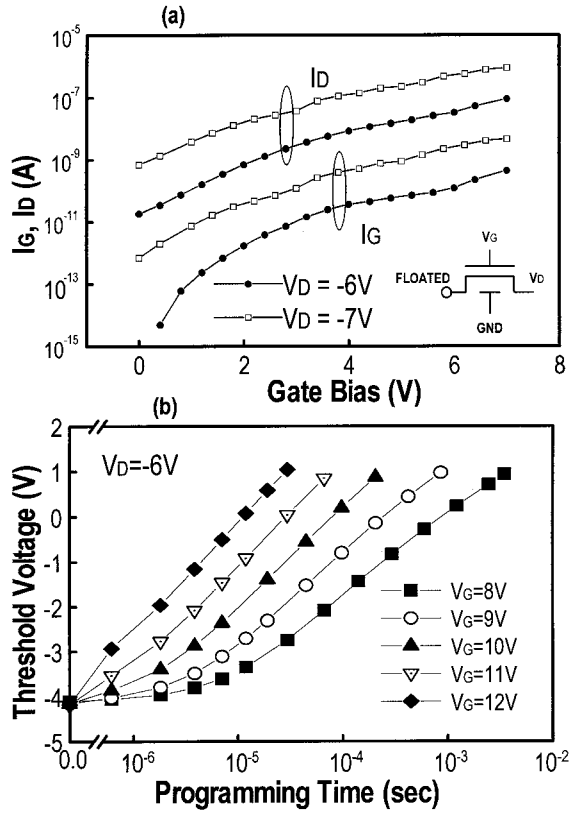


FIGURE 51.20 (a) The BBHE behavior in p-channel MOSFET with different bias conditions; and (b) the programming characteristics in p-channel Flash memory cell with BBHE injection scheme.

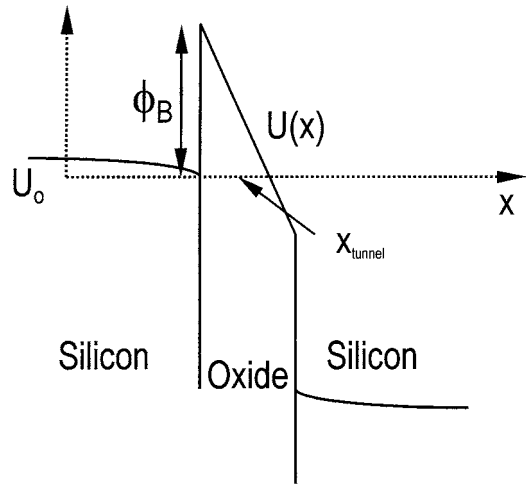


FIGURE 51.21 Schematic diagram of the potential barrier in the polysilicon-oxide-silicon system under applied high voltage.

(51.21)

$$\ln Tc = -\sqrt{\frac{8 \cdot m^* \cdot q}{\hbar}} \cdot \int_0^{X_{tunnel}} \sqrt{U(x) - U_o}\, dx$$

The tunneling current is obtained by integrating the product of the density of states $N_c(W)$ and the transmission coefficient from lowest occupied energy W_G to infinity,

$$J_{tunnel} = \int_{WG}^{\infty} Nc(W)\, Tc(W)\, dW \tag{51.22}$$

This expression is valid for any barrier shape. Under a strong oxide field E, the effective barrier is triangular and the coefficient can be obtained by integrating,

$$U(x) = \phi_B - E \cdot x \tag{51.23}$$

$$\ln Tc = \frac{-4\sqrt{2 \cdot m^* \cdot \Phi_B^{\,3}}}{3 \cdot \hbar \cdot q \cdot |E|} \tag{51.24}$$

where Φ_B is the barrier height, $\Phi_B = q\phi_B$.

Solving Eqs. 51.22 and 51.24 with the assumption that only electrons at the Fermi level contribute to the current yields the Fowler-Nordheim formula for the tunneling current density J_{tunnel} at high electric field:

$$J_{tunnel} = \frac{q^3 \cdot E^2}{16 \cdot \pi^2 \cdot \hbar \cdot \Phi_B} \cdot \exp\left(-\frac{4\sqrt{2 \cdot m^* \cdot \Phi_B^{\,3}}}{3 \cdot \hbar \cdot q \cdot E}\right) \tag{51.25}$$

This equation can also be expressed as

$$J_{tunnel} = \alpha \cdot E^2 \exp\left(-\frac{\beta}{E}\right) \tag{51.26}$$

where α and β are Fowler-Nordheim constants. The value of α is in the range of 4.7×10^{-5} to 6.32×10^{-7} A/V^2 and β is in the range of 2.2×10^8 to 3.2×10^8 V/cm.[47]

The barrier height and tunneling distance determine the tunneling efficiency. Generally, the barrier height at the Si–SiO_2 interface is about 3.1 eV, which is material dependent. This parameter is determined by the electron affinity and work function of the gate material. On the other hand, the tunneling distance depends on the oxide thickness and the voltage drop across the oxide. As indicated in Eq. 51.26, the tunneling current is exponentially proportional to the oxide field. Thus, a small variation in the oxide thickness or voltage drop would lead to a significant tunneling current change. Figure 51.22 shows the Fowler-Nordheim plot which can manifest the Fowler-Nordheim constants α and β. The Si–SiO_2 barrier height can be determined based on this F-N plot by quantum-mechanical (QM) modeling.[48]

Comparisons of Electron Injection Operations

As mentioned in the above section, there are several operation schemes that can be employed for electron injection, whereas only FN tunneling can be employed for ejecting electrons out of the floating gate. Owing to the specific features of the electron injection mechanism, the utilization of an electron injection scheme thereby determines the device structure design, process technology, and circuit design. The main features of CHEI and FN tunneling for n-channel Flash memory cell and also CHEI and BBHE injection for p-channel Flash memory cell are compared in Tables 51.1 and 51.2.

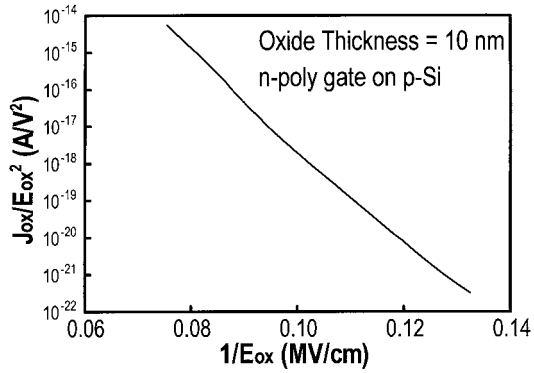


FIGURE 51.22 Fowler-Nordheim plot of the thin oxide.

TABLE 51.1 Comparisons of Fowler-Nordheim Tunneling and Channel Hot Electron Injection as Programming Scheme for Stacked-Gate Devices

FN Tunneling Injection Scheme	CHEI Scheme
Low power consumption	High power consumption
• Single external power supply	• Complicated circuitry technique
High oxide field	Low oxide field
• Thinner oxide thickness required	• Oxide can be thicker
• Higher trap generation rate	• Higher oxide integrity
• More severe read disturbance issue	• Low read disturbance issue
• Highly technological problem	
Slower programming speed	Faster programming speed

TABLE 51.2 Comparisons of Band-to-Band Tunneling Induced Hot Electron Injection and Channel Hot Electron Injection as Programming Scheme for Stacked-Gate Devices

	BBHE Injection Scheme	CHEI Scheme
Power consumption	Lower	Higher
Injection efficiency	Higher	Lower
Programming speed	Faster	Slower
Electron injection window	Wider	Narrower
Oxide field	Higher	Lower

List of Operation Modes

The employment of different electron transport mechanisms to achieve the programming and erase operations can lead to different device operation modes. Typically, in commercial applications, there are three different operation modes for n-channel Flash cells and two different operation modes for p-channel Flash cells. In the n-channel cell, as shown in Fig. 51.23, the write/erase operation modes include: (1) programming operation with CHEI and erase operation with FN tunneling ejection at source or drain side,[6–8,49–61] as shown in Fig. 51.23(a), usually referred as NOR-type operation mode; (2) programming operation with FN tunneling ejection at drain side and erase operation with FN tunneling injection through channel region,[62–70] as shown in Fig. 51.23(b), usually referred as AND-type operation mode; and (3) programming and erase operations with FN tunneling injection/ejection through channel region,[71–78] usually referred as NAND-type operation mode. As to the p-channel cell, as shown in Fig. 51.24, the write/erase operation modes include: (1) programming operation with CHEI at drain side and erase operation with FN tunneling ejection through channel region,[9] as shown in Fig. 51.24(a); (2) programming operation with BBHE at drain side and erase operation with FN tunneling injection through channel region,[10,11] as shown in Fig. 51.24(b).

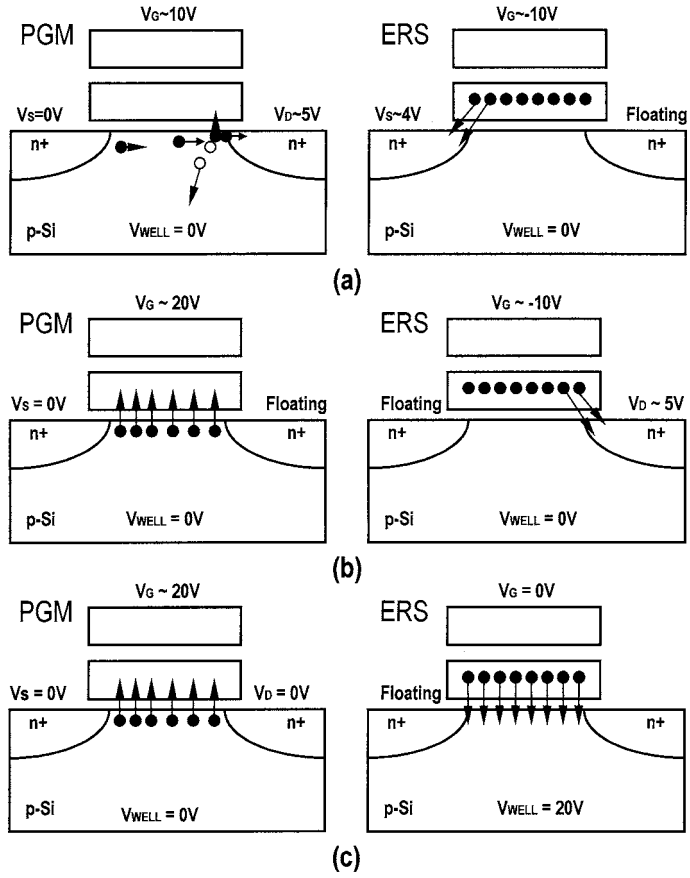


FIGURE 51.23 Different n-channel Flash write/erase operations: (a) programmming operation with CHEI at drain side and erase operation with FN tunneling ejection at source side; (b) programming operation with FN tunneling ejection at drain side and erase operation with tunneling injection through channel region; and (c) programming and erase operations with FN tunneling injection/ejection through channel region.

These operation modes not only lead to different device structures but also different memory array architectures. The main purpose of utilizing various device structures for different operation modes is based on the consideration of the operation efficiency, reliability requirements, and fabrication procedures. In addition, the operation modes and device structures determine, and also are determined by, the memory array architectures. In the following sections, the general improvements of the Flash device structures and the array architectures for specific operation modes are described.

51.5 Variations of Device Structure

CHEI Enhancement

As mentioned above, alternative operation modes are proposed to achieve pervasive purposes and various features, which are approached either by CHEI or FN tunneling injection. Furthermore, it is indicated that the over 90% of the Flash memory product ever shipped is the CHEI-based Flash memory device.[79] With the major manufacturers' competition, many innovations and efforts are dedicated to improve the performance and reliability of CHEI schemes.[50,53,56,57,61,80–83] As described in Eq. 51.11, an increase in the

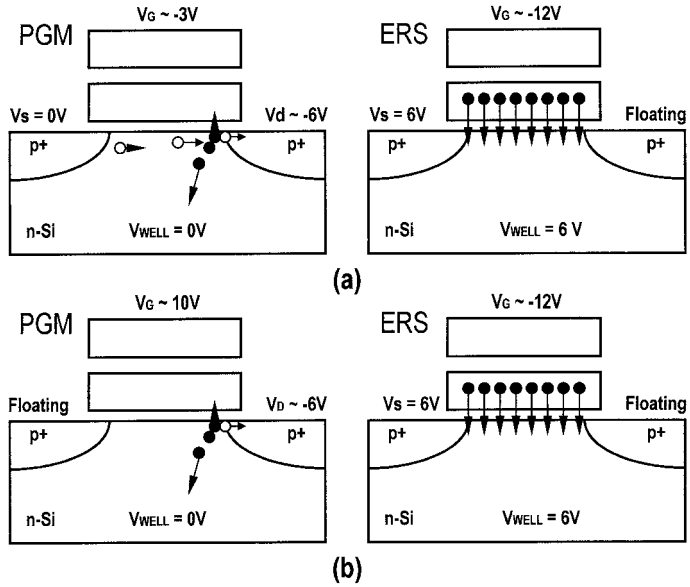


FIGURE 51.24 Different p-channel Flash write/erase operations: (a) programming operation with CHEI at drain side and erase operation with FN tunneling ejection through channel region; and (b) programming operation with BBHE at drain side and erase operation with FN tunneling injection through channel region.

electric field can enhance the probability of the electrons gaining enough energy. Therefore, the major approach to improve the channel hot electron injection efficiency is to enhance the electric field near the drain side. One of the structure modifications is utilizing the large-angle implanted p-pocket (LAP) around the drain to improve the programming speed.[56,57,60,83] LAP has also been used to enhance the punch-through immunity for scaling down capability.[50,53] As demonstrated in Fig. 51.13, the device with LAP has a twofold maximum electric field of that in the device without LAP structure. According to our previous report,[83] additionally, the LAP cell with proper process design can satisfy the cell performance requirements such as read current and punch-through resistance and also reliable long-term charge retention. Besides, the utilization of the p-pocket implantation can achieve the low-voltage operation and feasible scaling down capability simultaneously.

FN Tunneling Enhancement

From the standpoint of power consumption, the programming/erase operation based on the FN tunneling mechanism is unavoidable because of the low current during operation. As the dimension of Flash memory continues scaling down, in order to lower the operation voltage, a thinner tunnel oxide is needed. However, it is difficult to scale down the oxide thickness further due to reliability concerns. There are two ways to overcome this issue. One method is to raise the tunneling efficiency by employing a layer of electron injector on top of the tunnel oxide. Another method is to improve the gate coupling ratio of the memory cell without changing the properties of the insulator between the floating gate and well.

The electron injectors on the top of the tunnel oxide enhance the electric field locally and thus the tunneling efficiency is improved. Therefore, the onset of tunneling behavior takes place at a lower operation voltage. There are two materials used as electron injectors: polyoxide layer[84] and silicon-rich oxide (SRO) layer.[85] The surface roughness of the polyoxide is the main feature for electron injectors. However, owing to the properties of the polyoxide, the electron trapping during write/erase operation limits the application for Flash memory cells. On the other hand, the oxide layer containing excess silicon exhibits lower charge trapping and larger charge-to-breakdown characteristics. These silicon components

in the SRO layer form tiny silicon islands. The high tunneling efficiency is caused by the electric field enhancement of these silicon islands. Lin et al.[47] reported that the Flash cell with SRO layer can achieve the write/erase capability up to 10^6 cycles. However, the charge retentivity of the Flash memory cell with electron injector layers would be poorer than the conventional memory cell because the charge loss is also aggravated by the enhancement of the SRO layer. Thus, the stacked-gate device with SRO layer was also proposed as a volatile memory cell which can feature a longer refresh time than that in the conventional DRAM cell.[86]

Improvement of Gate Coupling Ratio

Another way to reduce the operation voltage is to increase the gate coupling ratio of the memory cell. From the description in the Section 51.4, the floating gate potential can be increased with an increased gate coupling ratio, through an enlarged inter-polysilicon capacitance. For the sake of obtaining a large interpoly capacitance, it is indispensable to reduce the interpoly dielectric thickness or increase the interpoly capacitor area. However, the reduced interpoly dielectric thickness would lead to charge loss during long-term operation. Therefore, a proper structure modification without increasing the effective cell size is necessary to increase the interpoly capacitance. It was proposed to put an extended floating gate layer over the bit-line region by employing two steps of polysilicon layer deposition.[68,87] Such device structure with memory array modifications would achieve a smaller effective cell size and a high coupling ratio (up to 0.8). Shirai et al.[88] proposed the process modification the increase to effective area on the top surface of the floating gate layer. This modified process, which forms a hemispherical-grained (HSG) polysilicon layer, can achieve a high capacitive coupling ratio (up to 0.8). However, the charge retentivity would be a major concern in considering the material as the electric injector.

51.6 Flash Memory Array Structures

NOR Type Array

In general, most of the Flash memory array, as shown in Fig. 51.25(a), is the NOR-type array.[49–61] In this array structure, two neighboring memory cells share a bit-line contact and a common source line. Therefore, a half the drain contact size and half the source line width is occupied in the unit memory cell. Since the memory cell is connected to the bit-line directly, the NOR-type array features random access and lower series resistance characteristics. The NOR-type array can be operated in a larger read current and thus a faster read operation speed. However, the drawback of the NOR-type array is the large cell area per unit cell. In order to maintain the advantages in NOR-type array and also reduce the cell size, there were several efforts to improve the array architectures. The major improvement in the NOR-type array is the elimination of bit-line contacts — the employment of buried bit-line configuration.[52] This concept evolves from the contactless EPROM proposed by Texas Instruments Inc. in 1986.[89] By using this contactless bit-line concept, the memory cell has a 34% size reduction.

AND Type Families

Another modification of the NOR-type array accompanied by a different operation mode is the AND-type array. In the NOR-type array, the CHEI is used as the electron injection scheme. However, owing to the considerations of power consumption and series resistance contributed by the buried bit-line/source, both the programming and erase operations utilize FN tunneling to eliminate the above concerns. Some improvements and modifications based on the NOR-type array have been proposed, including DIvided-bitline NOR (DINOR) proposed by Mitsubishi Corp.,[65,68] Contactless NOR (AND) proposed by Hitachi Corp.,[64,66] Asymmetrical Contactless Transistor (ACT) cell by Sharp Corp.,[69] and Dual String NOR (DuSNOR) by Samsung Corp.[70] and Macronix, Inc.[67] The DINOR architecture employs the main bit-line and sub-bit-line configuration to reduce the disturbance issue during FN programming. The AND and DuSNOR structures

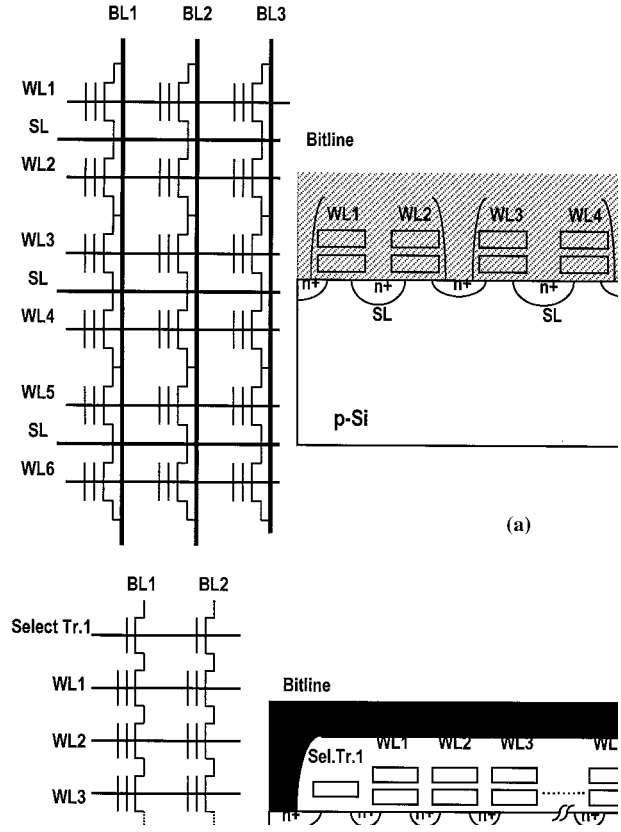


FIGURE 51.25 (a) Schematic top view and cross-section of the NOR-type Flash memory array; and (b) schematic top view and cross-section of the NAND-type Flash memory array.

consist of strings of memory cells with n^+ buried source and bit-lines. String-select and ground-select transistors are attached to the bit and source line, respectively. In DuSNOR structure, a smaller cell size can be realized because every two adjacent cell strings share a source line. Although a smaller cell size can be obtained utilizing the buried bit-line and source line, the resistance of the buried diffusion line would degrade the read performance. The read operation consideration will be the dominant factor in determining the size of a memory string in the AND and DuSNOR structures.

NAND Type Array

In order to realize a smaller Flash memory cell, the NAND structure was proposed in 1987.[90] As shown in Fig. 51.25(b), the memory cells are arranged in series. It was reported that the cell size of the NAND structure is only 44% of that in the NOR-type array under the same design rules. The operation mechanisms of a single memory cell in the NAND architecture is the same as NOR and AND architectures. However, the programming and read operations are more complex. Besides, the read operation speed is lower than that in the NOR-type structure because a number of memory cells are connected in series.

Originally, the NAND structure was operated with CHEI programming an FN tunneling through the channel region.[90] Later on, edge FN ejection at drain side was employed.[62,63] However, owing to reliability concerns, operations utilizing the bi-polarity write/erase scheme were then proposed to reduce the oxide damage.[71–78] Owing to the memory cells in the NAND structure being operated by FN write and erase, in order to improve the FN operation efficiency and reduce the operation voltage, the booster plate technology on the NAND structure was proposed by Samsung Corp.[77]

51.7 Evolution of Flash Memory Technology

In this section, as in Table 51.3, the development of device structures, process technology, and array architectures for Flash memory are listed by date. The burgeoning development in Flash memory devices reveals a prospective future.

TABLE 51.3 The Development of the Flash Memory

Year	Technology	Affiliation	Ref.
1984	Flash memory (2 μm, 64 μm^2)	Toshiba (Japan)	6
1985	Source-side erase type Flash (1.5 μm, 25 μm^2, 512 Kb)	EXCL (USA)	7
1986	Source-side injection (SI-EPROM)	UC Berkley (USA)	49
1987	Drain-erase type Flash, split gate device (128 Kb)	Seeq, UC Berkley (USA)	8
1987	NAND structure EEPROM (1 μm, 6.43 μm^2, 512 Kb)	Toshiba (Japan)	90
1987	Source-side erase Flash (0.8 μm, 9.3 μm^2)	Hitachi (Japan)	50
1988	ETOX-type Flash (1.5 μm, 36 μm^2, 256 Kb)	Intel (USA)	91
1988	NAND EEPROM (1 μm, 9.3 μm^2, 4 Mb)	Toshiba (Japan)	62
1988	NAND EEPROM (1 μm, 12.9 μm^2, 4 Mb)	Toshiba (Japan)	63
1988	Poly-poly erase Flash (1.2 μm, 18 μm^2)	WSI (USA)	92
1988	Contactless Flash (1.5 μm, 40.5 μm^2)	TI (USA)	93
1989	Negative gate erase	AMD (USA)	94
1989	ETOX-type Flash (1 μm, 15.2 μm^2, 1 Mb)	Intel (USA)	95
1989	Sidewall Flash (1 μm, 14 μm^2)	Toshiba (Japan)	51
1989	Punch-through-erase	Toshiba (Japan)	96
1990	Well-erase, bi-polarity W/E operation	Toshiba (Japan)	71, 72
1990	NAND, new self-aligned patterning (0.6 μm, 2.3 μm^2)	Toshiba (Japan)	97
1990	Contactless Flash, ACEE (0.8 μm, 8.6 μm^2, 4 Mb)	TI (USA)	98
1990	FACE cell (0.8 μm, 4.48 μm^2)	Intel (USA)	52
1990	Negative gate erase (0.6 μm, 3.6 μm^2, 16 Mb)	Mitsubishi (Japan)	54
1990	Tunnel diode-based contactless Flash	TI (USA)	99
1990	p-Pocket EPROM cell (0.6 μm, 16 Mb)	Toshiba (Japan)	53
1991	SAS process	Intel (USA)	100
1991	PB-FACE cell (0.8 μm, 4.16 μm^2)	Intel (USA)	101
1991	Burst-pulse erase (0.6 μm, 3.6 μm^2)	NEC (Japan)	56
1991	SSW-DSA cell (0.4 μm, 1.5 μm^2, 64 Mb)	NEC (Japan)	57
1991	Sector erase (0.6 μm, 3.42 μm^2, 16 Mb)	Hitachi (Japan)	64
1991	Self-convergence erase	Toshiba (Japan)	33, 35
1991	Virtual ground, auxiliary gate (0.5 μm, 2.59 μm^2)	Sharp (Japan)	59
1992	AND cell (0.4 μm, 1.28 μm^2, 64 Mb)	Hitachi (Japan)	66
1992	DINOR array (0.5 μm, 2.88 μm^2, 16 Mb)	Mitsubishi (Japan)	65
1992	2-Step erase method	NEC (Japan)	102
1992	Buried source side injection	TI (USA)	60
1992	p-Channel Flash Cell with SRO layer	IBM (USA)	9
1993	HiCR cell (0.4 μm, 1.5 μm^2, 64 Mb)	NEC (Japan)	87
1993	3-D sidewall Flash	Philip, Stanford (USA)	103
1993	Asymmetrical offset S/D DINOR (0.5 μm, 1.0 μm^2)	Mitsubishi (Japan)	68
1993	NAND EEPROM (0.4 μm, 1.13 μm^2, 64 Mb)	Toshiba (Japan)	74
1994	Self-convergent method	Motorola (USA)	104
1994	Substrate hot electron (SHE) erase	Mitsubishi (Japan)	105
1994	Dual-bit Split-Gate (DSG) cell (multi-level cell)	Hyundai (Korea)	106
1994	SA-STI NAND EEPROM (0.35 μm, 0.67 μm^2, 256 Mb)	Toshiba (Japan)	75
1994	SST cell	SST (USA)	124
1994	AND cell (0.25 μm, 0.4 μm^2, 256 Mb)	Hitachi (Japan)	107
1995	Multi-level NAND EEPROM	Toshiba (Japan)	108
1995	Convergence erase scheme	UT, AMD (USA)	109
1995	DuSNOR array (0.5 μm, 1.6 μm^2)	Samsung (Korea)	70
1995	CISEI programming scheme	AT&T, Lucent (USA)	110
1995	SAHF cell (0.3 μm, 0.54 μm^2, 256 Mb)	NEC (Japan)	88
1995	P-Flash with BBHE scheme (0.4 μm)	Mitsubishi (Japan)	10

continued

TABLE 51.3 (continued) The Development of the Flash Memory

1995	ACT cell (0.3 μm, 0.39 μm^2)	Sharp (Japan)	69
1995	Multi-level with self-convergence scheme	National (USA)	111
1995	Multi-level SWATT NAND cell (0.35 μm, 0.67 μm^2)	Toshiba (Japan)	112
1995	SCIHE injection scheme	AMD (USA)	113
1995	Alternating word-line voltage pulse	NKK (Japan)	114
1996	Self-limiting programming p-Flash	Mitsubishi (Japan)	11
1996	High speed NAND (HS-NAND) (2 μm^2, 16 Mb)	Samsung (Korea)	76
1996	Booster plate NAND (0.5 μm, 32 Mb)	Samsung (Korea)	77
1996	Shared bitline NAND (256 Mb)	Samsung (Korea)	115
1997	Φ-Cell	SGS-Thomson (France)	116
1997	NAND with STI (256 Mb)	Toshiba (Japan)	117
1997	Shallow groove isolation (SGI)	Hitachi (Japan)	118
1997	Word-line self-boosting NAND	Samsung (Korea)	119
1997	SPIN cell	Motorola (USA)	120
1997	Booster line technology for NAND	Samsung (Korea)	121
1997	AMG array	WSI (USA)	122
1997	High k interpoly dielectric	Lucent (USA)	123
1997	Self-convergent operation for p-Flash	NTHU (ROC)	12

51.8 Flash Memory System

Applications and Configurations

Flash memory is a single-transistor memory with floating gate for storing charges. Since 1985, the mass production of Flash memory has shared the market of non-volatile memory. The advantages of high density and electrical erasable operation make Flash memory an indispensable memory in the applications of programmable systems, such as network hubs, modems, PC BIOS, microprocessor-based systems, etc. Recently, image cameras and voice recorders have adopted Flash memory as the storage media. These applications require battery operation, which cannot afford large power consumption. Flash memory, a true non-volatile memory, is very suitable for these portable applications because stand-by power is not necessary.

In the interest of portable systems, the specification requirements of Flash memory include some special features that other memories (e.g., DRAM, SRAM) do not have. For example, multiple internal voltages with single external power supply, power-down during stand-by, direct execution, simultaneous erase of multiple blocks, simultaneous re-program/erase of different blocks, precise regulation of internal voltage, embedded program/erase algorithms to control threshold voltage. Since 1995, an emerging need of Flash memory is to increase the density by doubling the number of bits per cell. The charge stored in the floating gate is controlled precisely to provide multi-level threshold voltage. The information stored in each cell can be 00, 01, 10, or 11. Using multi-level storage can decrease the cost per bit tremendously. The multi-level Flash memories have two additional requirements: (1) fast sensing of multi-level information, and (2) high-speed multi-level programming. Since the memory cell characteristics would be degraded after cycling, which leads to fluctuation of programmed states, fast sensing and fast programming are challenged by the variation of threshold voltage in each level.

Another development is analog storage of Flash memory, which is feasible for image storage and voice record. The threshold voltage can be varied continuously between the maximum and minimum values to meet the analog requirements. Analog storage is suitable for recording the information that can tolerate distortion between the storing information and the restored information (e.g., image and speech data).

Before exploring the system design of Flash memory, the major differences between Flash memory and other digital memory, such as SRAM and DRAM, should be clarified. First, multiple sets of voltages are required in Flash memory for programming, erase, and read operations. The high-voltage related circuit is a unique feature that differs from other memories (e.g., DRAM, SRAM). Second, the charac-

teristics of Flash memory cell are degrading because of stress by programming and erasing. The controlling of an accurate threshold voltage by an internal finite state machine is the special function that Flash memory must have. In addition to the mentioned features, address decoding, sense amplifier, and I/O driver are all required in Flash memory. The system of Flash memory, as a result, can be regarded as a simplified mixed-signal product that employs digital and analog design concepts.

Figure 51.26 shows the block diagram of Flash memory. The word-line driver, bit-line driver, and source-line driver control the memory array. The word-line driver is high-voltage circuitry, which includes a logic X-decoder and level shifter. The interface between the bit-line driver and the memory array is the Y-gating. Along the bit-line direction, the sense amplifier and data input/output buffer are in charge of reading and temporary storage of data. The high-voltage parts include charge-pumping and voltage regulation circuitry. The generated high voltage is used to proceed programming and erasing operations. Behind the X-decoder, the address buffer catches the address. Finally, a finite state machine, which executes the operation code, dictates the operations of the system. The heart of the finite state machine is the clocking circuit, which also feeds the clock to a two-phase generator for charge-pumping circuits. In the following sections, the functions of each block will be discussed in detail.

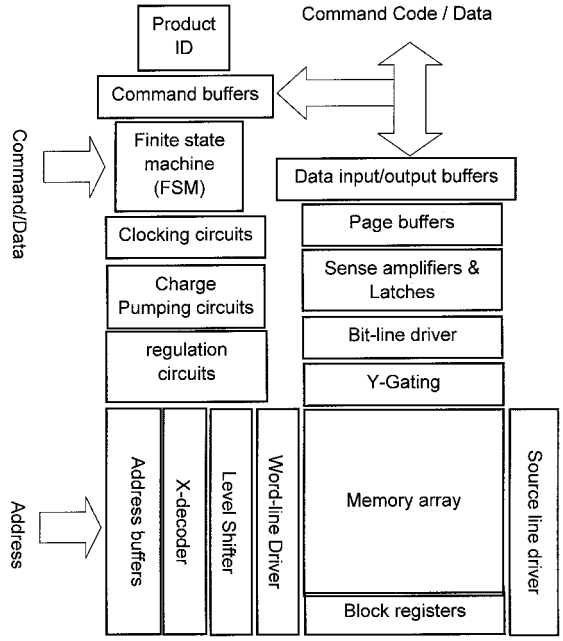


FIGURE 51.26 The block diagram of Flash memory system.

Finite State Machine

A finite state machine (FSM) is a control unit that processes commands and operation algorithms. Figure 51.27(a) demonstrates an example of an FSM. Figure 51.27(b) shows the details of an FSM. The command logic unit is an AND-OR-based logic unit that generates next state codes, while the state register latches the current state. The current state is related to the previous state and input state. State transitions follow the designated state diagram or state table that describe the functionality to translate state codes into controlling signals that are required by other circuits in the memory. The tendency to develop Flash memories goes in the direction of simultaneous program, erase, and read in different blocks. The global FSM takes charge of command distribution, address transition detection (ATD), and data input/output. The address command and data are queued when the selected FSM is busy. The local FSM deals with

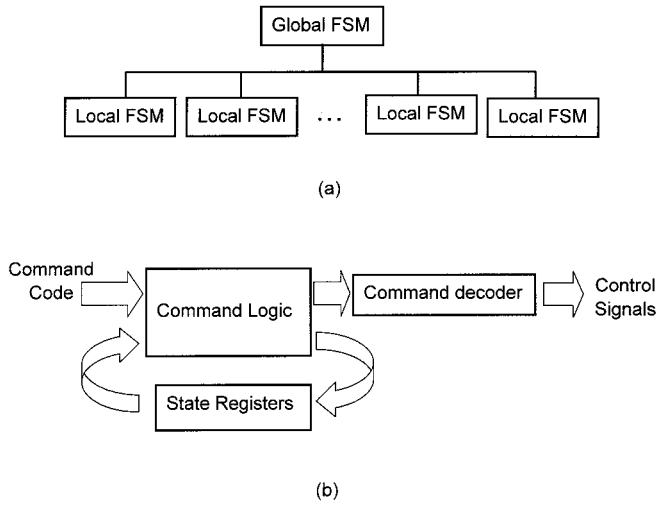


FIGURE 51.27 (a) The hierarchical architecture of a finite state machine; and (b) the block diagram of a finite state machine.

operations, including read, program, and erase, within the local block. The local FSM is activated and completes an operation independently when a command is issued. The global FSM manages the tasks distributing among local FSMs according to the address. The hierarchical local and global FSM can provide parallel processing; for instance, one block is being programmed while the other block is being erased. This feature of simultaneous read/write reduces the system overhead and speeds up the Flash memory. One example of the algorithm used in the FSM is shown in Fig. 51.28. The global FSM loads operating code (OP code) first, then the address transition detection (ATD) enables latch of the address when a different but valid address is observed. The status of the selected block is checked if the command can be executed right away, whereas the command, address, and/or data input are stored in the queues. The queue will be read when the local FSM is ready for excuting the next command. The operation code and address are decoded. Sense amplifiers are activated if a read command is issued. Charge-pumping circuits are back to work if a write command is issued. After all preparations are made, the process routine begins, which will be explained later. Following the completion of the process routine, the FSM checks its queues. If there is any command queued for delayed operation, the local FSM reads the queued data and continues the described procedures. Since these operations are invisible to the external systems, the system overhead is reduced.

The process routine is shown in Fig. 51.29. The read procedure waits for the completion signal of the sense amplifier, and then the valid data is sent immediately. The programming and erasing operations require a verification procedure to ascertain completion of the operation. The iteration of program-verification and erase-verification proceeds to fine-tune the threshold voltage. However, if the verification time exceeds the predetermined value, the block will be identified as a failure block. Further operation to this block is inhibited. Since the FSM controls the operations of the whole chip, a good design of the FSM can improve the operational speed.

Level Shifter

The level shifter is an interface between low-voltage and high-voltage circuits. Flash memory requires high voltage on the word-line and bit-line during programming and erasing operations. The high voltage appearing in a short time is regarded as a pulse. Figure 51.30 shows an example of a level shifter. The input signal is a pulse in V_{cc}/ground level, which controls the duration of a high-voltage pulse. The supply

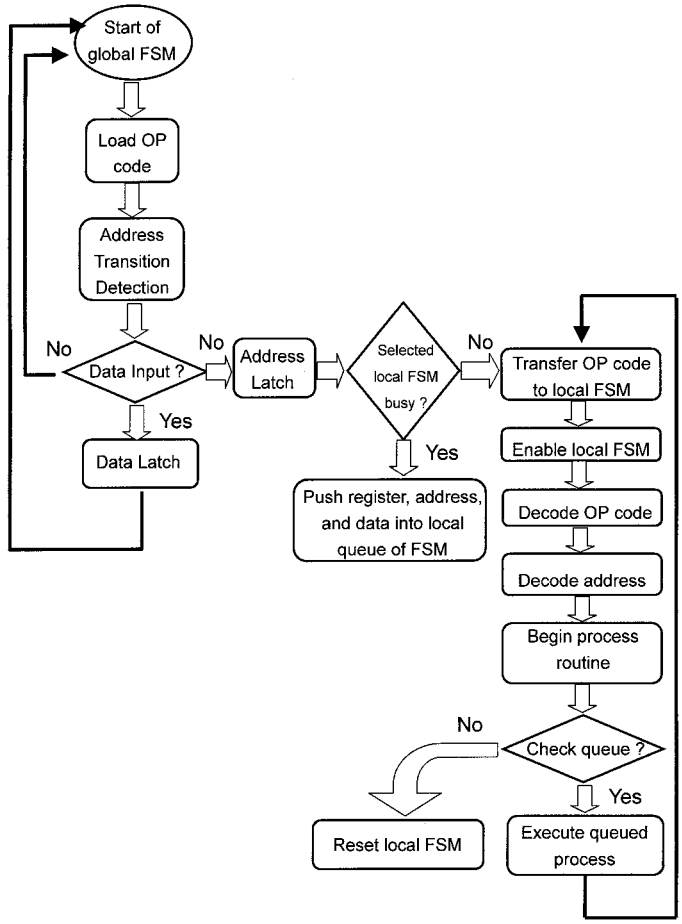


FIGURE 51.28 The algorithims of a finite state machine for simultaneous read-write feature.

of the level shifter determines the output voltage level of the high-voltage pulse. The level shifter is a positive feedback circuit, which turns stable at the ground level and supply voltage level (high voltage is generated from charge-pumping circuits). The operation of the level shifter can be realized as follows. The low-voltage input can only turn off the NMOS transistor but cannot turn off the PMOS parts. On the other hand, high voltage can only turn off the PMOS transistor. Therefore, generation of two mutually inverted signals can turn off the individual loading path and provide no leakage current during stand-by. The challenges of the design are the transition power consumption and the possibility of latch-up. The delay of the feedback loop will result in large leakage current flowing from the high-voltage supply to ground. The leakage current is similar to the transition current of conventional CMOS circuits, but larger due to the delay of the feedback loop. As the large leakage current occurs due to generated substrate current by hot carriers, the level shifter is susceptible to latch-up. The design of the level shifter should focus on speeding up the feedback loop and employing a latch-up-free apparatus. More sophisticated level shifters should be designed to provide tradeoff between switching power and the switching speed.

The level shifter is used in the word-line driver and the bit-line driver if the bit-line requires a voltage larger than the external power supply. The driver is expected to be small because the word-line pitch is nearly minimum feature size. Thus, the major challenges are to simplify the level shifter and to provide a high-performance switch.

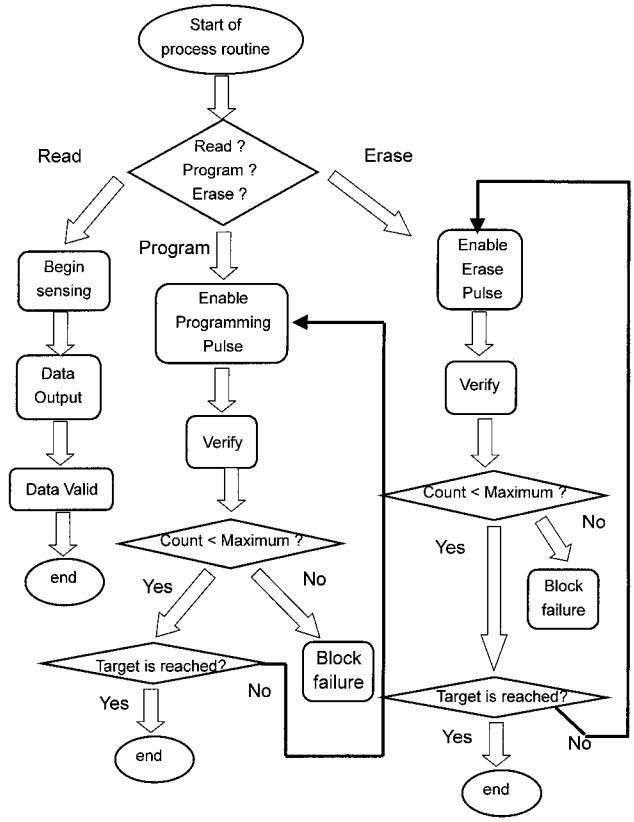


FIGURE 51.29 The algorithm of the process routine in Fig. 51.28.

Charge-Pumping Circuit

The charge-pumping circuit is a high-voltage generator that supplies high voltage for programming and erasing operations. This kind of circuit is well-known in power equipment, such as power supplies, high-voltage switches, etc. A conventional voltage generator requires a power transformer, which transforms input power to output power without loss. In other words, low voltage and large current is transformed to high voltage and low current. The transformer uses the inductance and magnetic flux to generate high voltage very efficiently. However, in the VLSI arena, it is difficult to produce inductors and the charge-pumping method is used instead. Figure 51.31 shows an example of a charge-pumping circuit that consists of multiple-stage pumping units. Each unit is composed of a one-way switch and a capacitor. The one-way switch is a high-voltage switch that does not allow charge to flow back to the input. The capacitor stores the transferred charge and gradually produces high voltage. No two consecutive stages operate at the same time. In other words, when one stage is transferring the charge, the next stage and the previous stage should serve as an isolation switch, which eliminates charge loss. Therefore, a two-phase clocking signal is required to proceed with the charge-pumping operation, producing no voltage drop between the input and output of the switch and large current drivability of the output. In addition, the voltage level must be higher than the previous stage. Therefore, the two-phase clocking signal must be level-shifted to individual high voltages to turn on and off the one-way switch in each pumping unit. A smaller charge-pumping or a more sophisticated level-shift circuit can be employed as self-boosted parts. The generated high voltage, in most cases, is higher than the required voltage. A regulation circuit, which can generate stable voltage and is immune to the fluctuation of external supply voltage and the operating temperature, is used to regulate the voltage and will be described later.

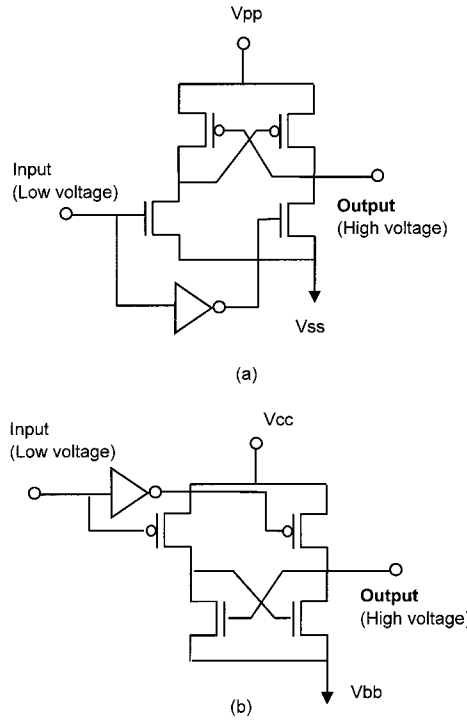


FIGURE 51.30 Level shifter: (a) positive polarity pulse, and (b) negative polarity pulse.

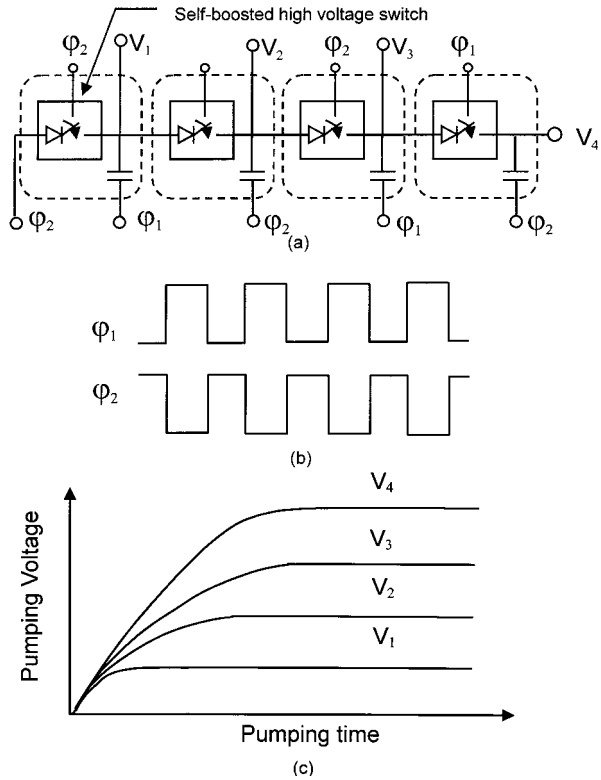


FIGURE 51.31 (a) Charge-pumping circuit; (b) two-phase clock; and (c) pumping voltage.

Sense Amplifier

The sense amplifier is an analog circuit that amplifies small voltage differences. Many circuits can be employed — from the simplest two-transistor cross-coupled latches to the complicated cascaded current-mirrors sense amplifiers. Here, a symbolic diagram is used to represent the sense amplifier in the following discussion. The focus of the sensing circuit is on multi-level sensing, which is currently the engineering issue in Flash memory. Figures 51.32(a) and (b) show the schemes of parallel sensing and consecutive sensing, respectively. These two schemes are based on analog-to-digital conversion (ADC). Information stored in the Flash memory can be read simultaneously with multiple comparators working at the same time. The outputs of the comparators are encoded into N digits for 2^N levels. Figure 51.32(b) shows the consecutive sensing scheme. The sensing time will be N times longer than the parallel sensing for 2^N levels. The sensing algorithm is a conventional binary search that compares the middle values in the consecutive range of interest. Only one sense amplifier is required for a cell. In the example, the additional sense amplifier is used for speeding up the sensing process. The second-stage sense amplifier can be pre-charged and prepared while the first-stage sense amplifier is amplifying the signal. Thus, the sensing time overhead is reduced.

When a multi-level scheme is used, the threshold voltage should be as tight as possible for each level. The depletion of unselected cells is strictly inhibited because the leakage current from unselected cells will destroy

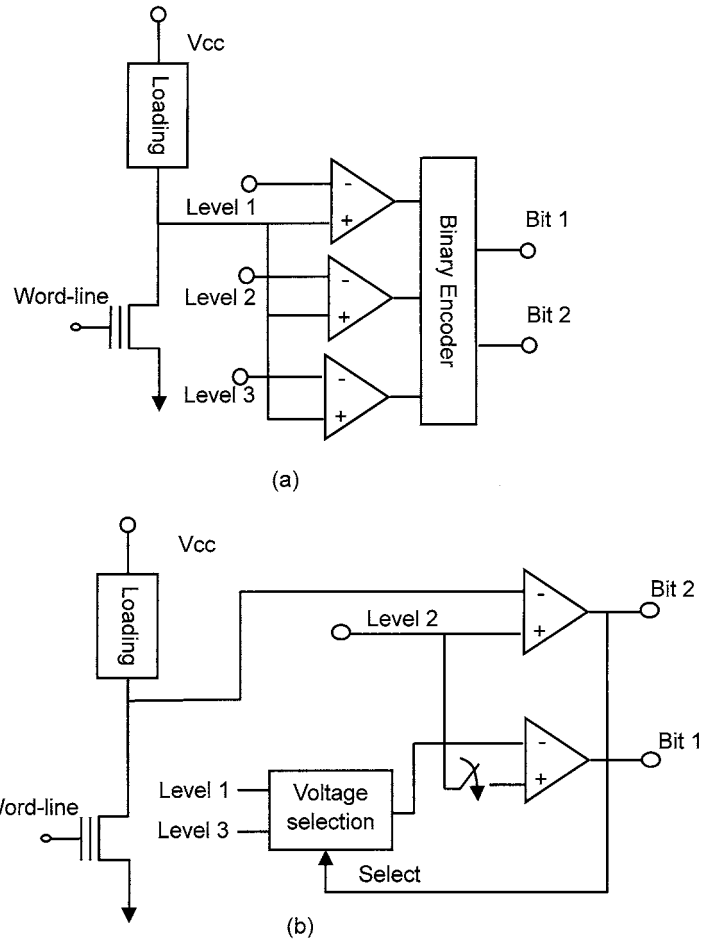


FIGURE 51.32 (a) Parallel sensing scheme, and (b) consecutive sensing scheme.

the true signal, which leads to error during sensing. Another challenge in multi-level sensing is the generation of reference voltages. Since the reference voltages are generated from the power supply, the leakage along the voltage divider path is unavoidable. Besides, the generated voltages are susceptible to the temperature variation and process-related resistance variation. If the variation of reference voltages cannot be minimized to a certain value, the ambiguous decision would be made for multi-level sensing due to unavoidable threshold spread for each level. Therefore, to provide high-sensitivity sense amplifier and to generate precise and robust reference voltages are the major developing goals for more than four-level Flash memory.

Voltage Regulator

A voltage regulator is an accurate voltage generator that is immune to temperature variation, process-related variation, and parasitic component effects. The concept of voltage regulation arises from the temperature-compensated device and the negative feedback circuits. Semiconductor carrier concentration and mobility are all dependent on the ambient temperature. Some devices have positive temperature coefficients, while others have negative coefficients. We can use both kinds of devices to produce a composite device for complete compensation. Figure 51.33 shows two back-to-back connected diodes that can be insensitive to the temperature over the temperature range of interest, if the doping concentration is properly designed. The forward-bias diode is negatively sensitive to temperature: the higher the temperature, the lower the cut-in voltage. On the other hand, the reverse-bias diode shows a reverse characteristic in the breakdown voltage. When connecting the two diodes and optimizing the diode characteristics, the regulated voltage can be insensitive to temperature. Nevertheless, the generated voltage is usually not what we want. A feedback loop, as shown in Fig. 51.34, is needed to generate precise

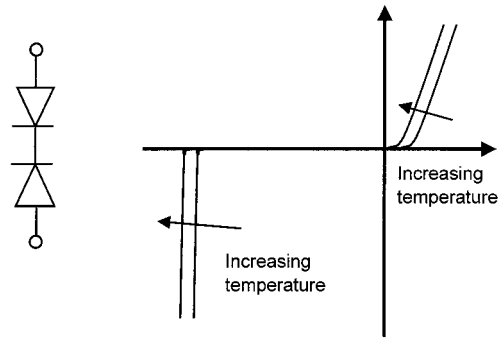


FIGURE 51.33 (a) Back-to-back connected temperature-compensated dual diodes; and (b) the characteristics of a diode as a function of temperature.

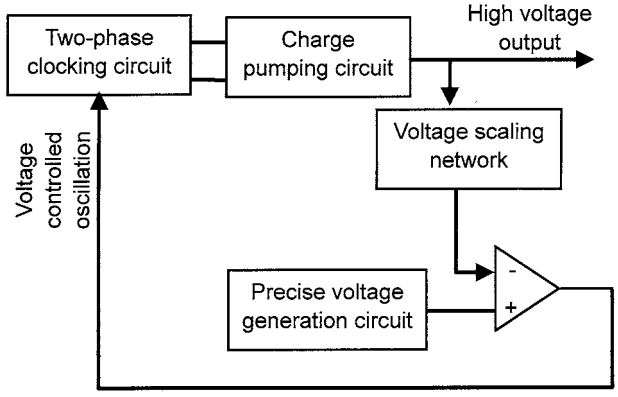


FIGURE 51.34 Voltage regulation block diagram.

programming and erasing voltage. The charge-pumping output voltage and drivability are functions of the two-phase clocking frequency. The pumping voltage can be scaled to be compared with the precise voltage generator to provide a feedback signal for the clocking circuit whose frequency can be varied. With the feedback loop, the generated voltage can be insensitive to temperature. Whatever the desired output voltage is, the structure can be applied in general to produce temperature-insensitive voltage.

Y-Gating

Y-gating is the decoding path of bit-lines. The bit-line pitch is as small as the minimum feature size. One register and one sense amplifier per bit-line is difficult to achieve. Y-gating serves as a switch that makes multiple bit-lines share one latch and one sense amplifier. Two approaches — indirect decoding and direct decoding — used as the Y-gating are shown in Figs. 51.35(a) and (b), respectively. Regarding the indirect decoding, if 2^N bit-lines are decoded using one-to-two decoding unit, the cascaded stages are required with N decoding control lines. However, when the direct decoding schemes is used, 2^N bit-lines require 2^N decoding lines to establish a one-to-2N decoding network, and the pre-decoder is required to generate the decoding signal. The area penalty of indirect decoding is reduced but the voltage drop along

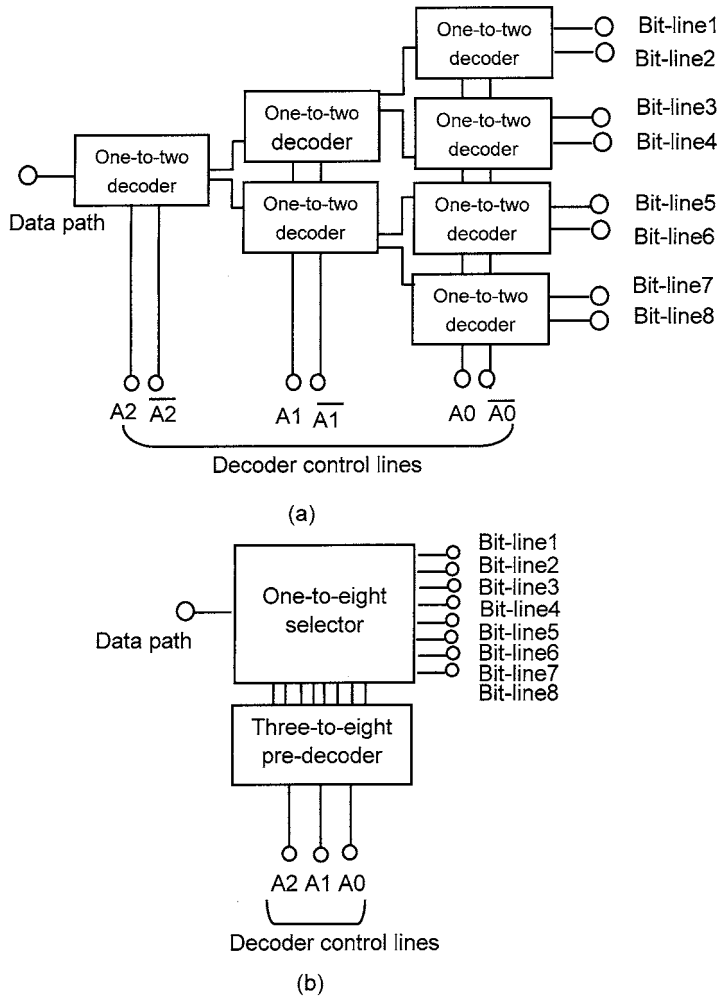


FIGURE 51.35 (a) Indirect decoding, and (b) direct decoding.

the decoding path is of concern. To avoid the voltage drop, a boosted decoding line should be used to overcome the threshold voltage of the passing transistor. Another approach to eliminate voltage drop is the employment of a CMOS transfer gate. However, the area penalty arises again due to well-to-well isolation. Since Flash memory is very sensitive to the drain voltage, the boosted decoding control lines, together with the indirect decoding scheme, are suggested.

Page Buffer

A page buffer is static memory (SRAM-like memory) that serves as a temporary storage of input data. The page buffer also serves as temporary storage of read data. With the page buffer, Flash memory can increase its throughput or bandwidth during programming and read, because external devices can talk to the page buffer in a very short time without waiting for the slow programming of Flash memory. After the input data is transferred to the page buffer, the Flash memory begins programming and external devices can do other tasks. The page size should be carefully designed according to the applications. The larger the page size, the more data can be transferred into Flash memory without having to wait for the completion of programming. However, the area penalty limits the page size. There exists a proper design of page buffer for the application of interest.

Block Register

The block register stores the information about the individual block. The information includes failure of the block, write inhibit, read inhibit, executable operation, etc., according to the applications of interest. Some blocks, especially the boot block, are write-inhibited after first programming. This prevents virus injection in some applications, such as PC BIOS. The block registers are also Flash memory cells for storing block information, which will not disappear after power-off. When the local FSM is working on a certain block, the first thing is to check the status of the block by reading the register. If the block is identified as a failure block, no further operation can be made in this block.

Summary

Flash memory is a system with mixed analog and digital systems. The analog circuits include voltage-generation circuits, analog-to-digital converter circuits, sense amplifier circuits, and level-shifter circuits. These circuits require excellent functionality but small area consumption. The complicated analog designs in the pure-analog circuit do not meet the requirements of Flash memory, which requires large array efficiency, large memory density, and large storage volume. Therefore, the design of these analog circuits tends toward reduced design and qualified function. On the other hand, the digital parts of Flash memory are not as complicated as those digital circuits used in pure digital signal process circuits. Therefore, the mixed analog and digital Flash memory system can be implemented in a simplified way. Furthermore, Flash memory is a memory cell-based system. All the functions of the circuits are designed according to the characteristics of the memory cell. Once the cell structure of a memory differs, it will result in a completely different system design.

References

1. Kahng, D. and Sze, S. M., A floating gate and its application to memory devices, *Bell Syst. Tech. J.*, vol. 46, p. 1283, 1967.
2. Frohman-Bentchlowsky, D., An integrated metal-nitride-oxide-silicon (MNOS) memory, *IEDM Tech. Dig.*, 1968.
3. Pao, H. C and O'Connel, M., *Appl. Phys. Lett.* no. 12, p. 260, 1968.
4. Frohman-Bentchlowsky, D., A fully decoded 2048-bit electrically programmable FAMOS read only memory, *IEEE J. Solid-State Circuits*, vol. SC-6, no. 5, p. 301, 1971.

5. Johnson, W., Perlegos, G., Renninger, A., Kuhn, G., and Ranganath, T., A 16k bit electrically erasable non-volatile memory, *Tech. Dig. IEEE ISSCC*, p. 152, 1980.
6. Masuoka, F., Asano, M., Iwahashi, H., Komuro, T., and Tanaka, S., A new Flash EEPROM cell using triple polysilicon technology, *IEDM Tech. Dig.*, p. 464, 1984.
7. Mukherjee, S., Chang, T., Pang, R., Knecht, M., and Hu, D., A single transistor EEPROM cell and its implementation in a 512K CMOS EEPROM, *IEDM Tech. Dig.*, p. 616, 1985.
8. Samachisa, G., Su, C.-S., Kao, Y.-S., Smarandoiu, G., Wang, C. Y.-M., Wong, T., and Hu, C., A 128K Flash EEPROM using double-polysilicon technology, *IEEE J. Solid-State Circuits*, vol. SC-22, no. 5, p. 676, 1987.
9. Hsu, C. C.-H., Acovic, A., Dori, L., Wu, B., Lii, T., Quinlan, D., DiMaria, D., Taur, Y., Wordeman, M., and Ning, T., A high speed, low power p-channel Flash EEPROM using silicon rich oxide as tunneling dielectric, *Ext. Abstract of 1992 SSDM*, p. 140, 1992.
10. Ohnakado, T., Mitsunaga, K., Nunoshita, M., Onoda, H., Sakakibara, K., Tsuji, N., Ajika, N., Hatanaka, M., and Miyoshi, H., Novel electron injection method using band-to-band tunneling induced hot electron (BBHE) for Flash memory with p-channel cell, *IEDM Tech. Dig.*, p. 279, 1995.
11. Ohnakado, T., Takada, H., Hayashi, K., Sugahara, K., Satoh, S., and Abe, H., Novel self-limiting program scheme utilizing n-channel select transistors in p-channel DINOR Flash memory, *IEDM Tech. Dig.*, 1996.
12. Shen, S.-J., Yang, C.-S., Wang, Y.-S., and Hsu, C. C.-H., Novel self-convergent programming scheme for multi-level p-channel Flash memory, *IEDM Tech. Dig.*, p. 287, 1997.
13. Chung, S. S., Kuo, S. N., Yih, C. M., and Chao, T. S., Performance and reliability evaluations of p-channel Flash memories with different programming schemes, *IEDM Tech. Dig.*, 1997.
14. Wang, S. T., On the I-V characteristics of floating gate MOS transistors, *IEEE Trans. Electron Devices*, vol. ED-26, no. 9, p. 1292, 1979.
15. Liong, L. C. and Liu, P.-C., A theoretical model for the current-voltage characteristics of a floating gate EEPROM cell, *IEEE Trans. Electron Devices*, vol. ED-40, no. 1, p. 146, 1993.
16. Manthey, J. T., Degradation of Thin Silicon Dioxide Films and EEPROM Cells, Ph.D. dissertation, 1990.
17. Ng, K. K. and Taylor, G. W., Effects of hot-carrier trapping in n and p channel MOSFETs, *IEEE Trans. Electron Devices*, vol. ED-30, p. 871, 1983.
18. Selmi, L., Sangiorgi, E., Bez, R., and Ricco, B., Measurement of the hot hole injection probability from Si into SiO_2 in p-MOSFETs, *IEDM Tech. Dig.*, p. 333, 1993.
19. Tang, Y., Kim, D. M., Lee, Y.-H., and Sabi, B., Unified characterization of two-region gate bias stress in submicronmeter p-channel MOSFET's, *IEEE Electron Device Lett.*, vol. EDL-11, no. 5, p. 203, 1990.
20. Takeda, E., Kume, H., Toyabe, T., and Asai, S., Submicrometer MOSFET structure for minimizing hot carrier generation, *IEEE Trans. Electron Devices*, vol. ED-29, p. 611, 1982.
21. Shockley, W., Problems related to p-n junction in silicon, *Solid- State Electron.*, vol. 2, p. 35, 1961.
22. Verwey, J. F., Kramer, R. P., and de Maagt B. J., Mean free path of hot electrons at the surface of boron-doped silicon, *J. Appl. Phys.*, vol. 46, p. 2612, 1975.
23. Ning, T. H., Osburn, C. M., and Yu, H. N., Emission probability of hot electrons from silicon into silicon dioxide, *J. Appl. Phys.*, vol. 48, p. 286, 1977.
24. Hu, C., Lucky-electron model of hot-electron emission, *IEDM Tech. Dig.*, p. 22, 1979.
25. Tam, S., Ko, P.-K., and Hu, C., Lucky-electron model of channel hot electron injection in MOSFET's, *IEEE Trans. Electron Devices*, vol. ED-31, p. 1116, 1984.
26. Berglung, C. N. and Powell, R. J., Photoinjection into SiO2. Electron scattering in the image force potential well, *J. Appl. Phys.*, vol. 42, p. 573, 1971.
27. Ong, T.-C., Ko, P. K., and Hu, C., Modeling of substrate current in p-MOSFET's, *IEEE Electron Device Lett.*, vol. EDL-8, no. 9, p. 413, 1987.
28. Ong, T.-C., Seki, K., Ko, P. K., and Hu, C., P-MOSFET gate current and device degradation, *Proc. IEEE/IRPS*, p. 178, 1989.

29. Takeda, E., Suzuki, N., and Hagiwara, T., Device performance degradation due to hot carrier injection at energies below the Si-SiO_2 energy barrier, *IEDM Tech. Dig.*, p. 396, 1983.
30. Hofmann, K. R., Werner, C., Weber, W., and Dorda, G., Hot-electron and hole emission effects in short n-channel MOSFET's, *IEEE Trans. Electron Devices*, vol. ED-32, no. 3, p. 691, 1985.
31. Nissan-Cohen, Y., A novel floating-gate method for measurement of ultra-low hole and electron gate currents in MOS transistors, *IEEE Electron Device Lett.*, vol. EDL-7, no. 10, p. 561, 1986.
32. Sak, N. S., Hereans, P. L., Hove, L. V. D., Maes, H. E., DeKeersmaecker, R. F., and Declerck, G. J., Observation of hot-hole injection in NMOS transistors using a modified floating gate technique, *IEEE Trans. Electron Devices*, vol. ED-33, no. 10, p. 1529, 1986.
33. Yamada, S., Suzuki, T., Obi, E., Oshikiri, M., Naruke, K., and Wada, M., A self-convergence erasing scheme for a simple stacked gate Flash EEPROM, *IEDM Tech. Dig.*, p. 307, 1991.
34. Ong, T. C., Fazio, A., Mielke, N., Pan, S., Righos, N., Atwood, G., and Lai, S., Erratic erase in ETOX Flash memory array, *Proc. Symp. on VLSI Technology*, p. 83, 1993.
35. Yamada, S., Yamane, T., Amemiya, K., and Naruke, K., A self-convergence erase for NOR Flash EEPROM using avalanche hot carrier injection, *IEEE Trans. Electron Devices*, vol. ED-43, no. 11, p. 1937, 1996.
36. Chen, J., Chan, T. Y., Chen, I. C., Ko, P. K., and Hu, C., Subbreakdown drain leakage current in MOSFET, *IEEE Electron Device Lett.*, vol. EDL.-8, no. 11, p. 515, 1987.
37. Chan, T. Y., Chen, J., Ko, P. K., and Hu, C., The impact of gate-induced drain leakage on MOSFET scaling, *IEDM Tech. Dig.*, p. 718, 1987.
38. Shrota, R., Endoh, T., Momodomi, M., Nakayama, R., Inoue, S., Kirisawa, R., and Masuoka, F., An accurate model of sub-breakdown due to band-to-band tunneling and its application, *IEDM Tech. Dig.*, p. 26, 1988.
39. Chang, C. and Lien, J., Corner-field induced drain leakage in thin oxide MOSFET's, *IEDM Tech. Dig.*, p. 714, 1987.
40. Chen, I.-C., Coleman, D. J., and Teng, C. W., Gate current injection initiated by electron band-to-band tunneling in MOS devices, *IEEE Electron Device Lett.*, vol. EDL-10, no. 7, p. 297, 1989.
41. Yoshikawa, K., Mori, S., Sakagami, E., Ohshima, Y., Kaneko, Y., and Arai, N., Lucky-hole injection induced by band-to-band tunneling leakage in stacked gate transistor, *IEDM Tech. Dig.*, p. 577, 1990.
42. Haddad, S., Chang, C., Swanminathan, B., and Lien, J., Degradation due to hole trapping in Flash memory cells, *IEEE Electron Device Lett.*, vol. EDL-10, no. 3, p. 117, 1989.
43. Igura, Y., Matsuoka, H., and Takeda, E., New device degradation due to Cold carrier created by band-to-band tunneling, *IEEE Electron Device Lett.*, vol. 10, no. 5, p. 227, 1989.
44. Lenzlinger, M. and Snow, E. H., Fowler-Nordheim tunneling into thermally grown SiO_2, *J. Appl. Phys.*, vol. 40, no. 1, p. 278, 1969.
45. Weinberg, Z. A., On tunneling in MOS structure, *J. Appl. Phys.*, vol. 53, p. 5052, 1982.
46. Ricco, B. and Fischetti, M. V., Temperature dependence of the currents in silicon dioxide in the high field tunneling regime, *J. Appl. Phys.*, vol. 55, p. 4322, 1984.
47. Lin, C. J., Enhanced Tunneling Model and Characteristics of Silicon Rich Oxide Flash Memory, Ph.D. dissertation, 1996.
48. Olivo, P., Sune, J., and Ricco, B., Determination of the Si-SiO_2 barrier height from the Fowler-Nordheim plot, *IEEE Electron Device Lett.*, vol. EDL-12, no. 11, p. 620, 1991.
49. Wu, A. T., Chan, T. Y., Ko, P. K., and Hu, C., A source-side injection erasable programmable read-only-memory (SI-EPROM) device, *IEEE Electron Device Lett.*, vol. EDL-7, no. 9, p. 540, 1986.
50. Kume, H., Yamamoto, H., Adachi, T., Hagiwara, T., Komori, K., Nishimoto, T., Koike, A., Meguro, S., Hayashida, T., and Tsukada, T., A Flash-erase EEPROM cell with an asymmetric source and drain structure, *IEDM Tech. Dig.*, p. 560, 1987.
51. Naruke, K., Yamada, S., Obi, E., Taguchi, S., and Wada, M., A new Flash-erase EEPROM cell with a side-wall select-gate on its source side, *IEDM Tech. Dig.*, p. 603, 1989.

52. Woo, B. J., Ong, T. C., Fazio, A., Park, C., Atwood, D., Holler, M., Tam, S., and Lai, S., A novel memory cell using Flash array contact-less EPROM (FACE) technology, *IEDM Tech. Dig.*, p. 91, 1990.
53. Ohshima, Y., Mori, S., Kaneko, Y., Sakagami, E., Arai, N., Hosokawa, N., and Yoshikawa, K., Process and device technologies for 16M bit EPROM's with large-tilt-angle implanted p-pocket cell, *IEDM Tech. Dig.*, p. 95, 1990.
54. Ajika, N., Obi, M., Arima, H., Matsukawa, T., and Tsubouchi, N., A 5 volt only 16M bit Flash EEPROM cell with a simple stacked gate structure, *IEDM Tech. Dig.*, p. 115, 1990.
55. Manos, P. and Hart, C., A self-aligned EPROM structure with superior data retention, *IEEE Electron Device Lett.*, vol. EDL-11, no. 7, p. 309, 1990.
56. Kodama, N., Saitoh, K., Shirai, H., Okazawa, T., and Hokari, Y., A 5V only 16M bit Flash EEPROM cell using highly reliable write/erase technologies, *Proc. Symp. on VLSI Technology*, p. 75, 1991.
57. Kodama, N., Oyama, K., Shirai, H., Saitoh, K., Okazawa, T., and Hokari, Y., A symmetrical side wall (SSW)-DSA cell for a 64-M bit Flash memory, *IEDM Tech. Dig.*, p. 303, 1991.
58. Liu, D. K. Y., Kaya, C., Wong, M., Paterson, J., and Shah, P., Optimization of a source-side-injection FAMOS cell for Flash EPROM application, *IEDM Tech. Dig.*, p. 315, 1991.
59. Yamauchi, Y., Tanaka, K., Shibayama, H., and Miyake, R., A 5V-only virtual ground Flash cell with an auxiliary gate for high density and high speed application, *IEDM Tech. Dig.*, p. 319, 1991.
60. Kaya, C., Liu, D. K. Y., Paterson, J., and Shah, P., Buried source-side injection (BSSI) for Flash EPROM programming, *IEEE Electron Device Lett.*, vol. EDL-13, no. 9, p. 465, 1992.
61. Yoshikawa, K., Sakagami, E., Mori, S., Arai, N., Narita, K., Yamaguchi, Y., Ohshima, Y., and Naruke, K., A 3.3V operation nonvolatile memory cell technology, *Proc. Symp. on VLSI Technology*, p. 40, 1992.
62. Shirota, R., Itoh, Y., Nakayama, R., Momodomi, M., Inoue, S., Kirisawa, R., et al., A new NAND cell for ultra high density 5V-only EEPROM's, *Proc. Symp. on VLSI Technology*, p. 33, 1988.
63. Momodomi, M., Kirisawa, R., Nakayama, R., Aritome, S., Endoh, T., Itoh, T., et al., New device technologies for 5V- only 4Mb EEPROM with NAND structure cell, *IEDM Tech. Dig.*, p. 412, 1988.
64. Kume, H., Tanaka, T., Adachi, T., Miyamoto, N., Saeki, S., Ohji, Y., et al., A 3.42 μm^2 Flash memory cell technology conformable to a sector erase, *Proc. Symp. on VLSI Technology*, p. 77, 1991.
65. Onoda, H., Kunori, Y., Kobayashi, S., Ohi, M., Fukumoto, A., Ajika, N., and Miyoshi, H., A novel cell structure suitable for a 3 volt operation, sector erase Flash memory, *IEDM Tech. Dig.*, p. 599, 1992.
66. Kume, H., Kato, M., Adachi, T., Tanaka, T., Sasaki, T., and Okazaki, T., A 1.28 μm^2 contactless memory cell technology for a 3V-only 64M bit EEPROM, *IEDM Tech. Dig.*, p. 991, 1992.
67. Method for manufacturing a contact-less floating gate transistor, U.S. Patent 5453391, 1993.
68. Ohi, M., Fukumoto, A., Kunori, Y., Onoda, H., Ajika, N., Hatanaka, M., and Miyoshi, H., An asymmetrical offset source/drain structure for virtual ground array Flash memory with DINOR operation, *Proc. Symp. on VLSI Technology*, p. 57, 1993.
69. Yamauchi, Y., Yoshimi, M., Sato, S., Tabuchi, H., Takenaka, N., and Sakiyam, K., A new cell structure for sub-quarter micron high density Flash memory, *IEDM Tech. Dig.*, p. 267, 1995.
70. Kim, K. S., Kim, J. Y., Yoo, J. W., Choi, Y. B., Kim, M. K., Nam, B. Y., et al, A novel dual string NOR (DuSNOR) memory cell technology scalable to the 256M bit and 1G bit Flash memory, *IEDM Tech. Dig.*, p. 263, 1995.
71. Kirisawa, R., Aritome, S., Nakayama, R., Endoh, T., Shirota, R., and Masuoka, F., A NAND structures cell with a new programming technology for highly reliable 5V-only Flash EEPROM, *Proc. Symp. on VLSI Technology*, p. 129, 1990.
72. Aritome, S., Kirisawa, R., Endoh, T., Nakayama, R., Shirota, R., Sakui, K., Ohuchi, K., and Masuoka, F., Extended data retention characteristics after more than 10^4 write and erase cycles in EEPROM's, *Proc. IEEE/IRPS*, p. 259, 1990.
73. Endoh, T., Iizuka, H., Aritome, S., Shirota, R., and Masuoka, F., New write/erase operation technology for Flash EEPROM cells to improve the read disturb characteristics, *IEDM Tech. Dig.*, p. 603, 1992.

74. Aritome, S., Hatakeyama, K., Endoh, T., Yamaguchi, T., Shuto, S., Iizuka, H., et al., A 1.13 μm^2 memory cell technology for reliable 3.3V 64M NAND EEPROM's, *Ext. Abstract of 1993 SSDM*, p. 446, 1993.
75. Aritome, S., Satoh, S., Maruyama, T., Watanabe, H., Shuto, S., Hermink, G. J., Shirota, R., Watanabe, S., and Masuoka, F., A 0.67 μm^2 self-aligned shallow trench isolation cell (SA-STI cell) for 3V-only 256M bit NAND EEPROM's, *IEDM Tech. Dig.*, p. 61, 1994.
76. Kim, D. J., Choi, J. D., Kim, J. Oh, H. K., and Ahn, S. T., and Kwon, O.H., Process integration for the high speed NAND Flash memory cell, *Proc. Symp. on VLSI Technology*, p. 236, 1996.
77. Choi, J. D., Kim, D. J., Jang, D. S., Kim, J., Kim, H. S., Shin, W. C., Ahn, S. T., and Kwon, O. H., A novel booster plate technology in high density NAND Flash memories for voltage scaling down and zero program disturbance, *Proc. Symp. on VLSI Technology*, p. 238, 1996.
78. Entoh, T., Shimizu, K., Iizuka, H., and Masuoka, F., A new write/erase method to improve the read disturb characteristics based on the decay phenomena of the stress induced leakage current for Flash memories, *IEEE Trans. Electron Device*, vol. ED-45, no. 1, p. 98, 1998.
79. Lai, S. K., NVRAM technology, NOR Flash design and multi-level Flash, I*EDM NVRAM Technology and Application Short Course*, 1995.
80. Yamada, S., Hiura, Y., Yamane, T., Amemiya, K., Ohshima, Y., and Yoshikawa, K., Degradation mechanism of Flash EEPROM programming after programming/erase cycles, *IEDM Tech. Dig.*, p. 23, 1993.
81. Cappelletti, P., Bez, R., Cantarelli, D., and Fratin, L., Failure mechanisms of Flash cell in program/erase cycling, *IEDM Tech. Dig.*, p. 291, 1994.
82. Liu, Y. C., Guo, J.-C., Chang, K. L., Huang, C. I., Wang, W. T., Chang, A., and Shone, F., Bitline stress effects on Flash EPROM cells after program/erase cycling, *IEEE Nonvolatile Semiconductor Memory Workshop*, 1997.
83. Shen, S.-J., Chen, H.-M., Lin, C.-J., Chen, H.-H., Hong, G., and Hsu, C. C.-H., Performance and reliability trade-off of large-tilted-angle implant p-pocket (LAP) on stacked-gate memory devices, *Japan. J. Appl. Phys.*, vol. 36, part 1, no. 7A, p. 4289, 1997.
84. DiMaria, D. J., Dong, D. W., Pesavento, F. L., Lam, C., and Brorson, B. D., Enhanced conduction and minimized charge trapping in electrically alterable read-only memories using off-stoichiometric silicon dioxide films, *J. Appl. Phys.*, vol. 55, p. 300, 1984.
85. Lin, C.-J., Hsu, C. C.-H., Chen, H.-H., Hong, G., and Lu, L. S., Enhanced tunneling characteristics of PECVD silicon-rich-oxide (SRO) for the application in low voltage Flash EEPROM, *IEEE Trans. Electron Device*, vol. ED-43, no. 11, p. 2021, 1996.
86. Shen, S.-J., Lin C.-J., and Hsu, C. C.-H, Ultra fast write speed, long refresh time, low FN power operated volatile memory cell with stacked nanocrystalline Si film, *IEDM Tech. Dig.*, p. 515, 1996.
87. Hisamune, Y. S., Kanamori, K., Kubota, T., Suzuki, Y., Tsukiji, M., Hasegawa, E., et al., A high capacitive-coupling ratio (HiCR) cell for 3V-only 64 M bit and future Flash memories, *IEDM Tech. Dig.*, p. 19, 1993.
88. Shirai, H., Kubota, T., Honma, I., Watanabe, H., Ono, H., and Okazawa, T., A 0.54 μm^2 self-aligned, HSG floating gate cell (SAHF cell) for 256M bit Flash memories, *IEDM Tech. Dig.*, p. 653, 1995.
89. Esquivel, J., Mitchel, A., Paterson, J., Riemenschnieder, B., Tieglaar, H., et al., High density contactless, self aligned EPROM cell array technology, *IEDM Tech. Dig.*, p. 592, 1986.
90. Masuoka, F., Momodomi, M., Iwata, Y., and Shirota, R., New ultra high density EPROM and Flash EEPROM with NAND structure cell, *IEDM Tech. Dig.*, p. 552, 1987.
91. Kynett, V. N., Baker, A., Fandrich, M. L., Hoekstra, G. P., Jungroth, O., Hreifels, J. A., et al., An in-system re-programmable 32K × 8 CMOS Flash memory, *IEEE J. Solid Stat.*, vol. SC-23, no. 5, p. 1157, 1988.
92. Kazerounian, R., Ali, S., Ma, Y., and Eitan, B., A 5 volt high density poly-poly erase Flash EPROM cell, *IEDM Tech. Dig.*, p. 436, 1988.
93. Gill, M., Cleavelin, R., Lin, S., D'Arrigo, I., Santin, G., Shah, P., et al., A 5-volt contactless 256K bit Flash EEPROM technology, *IEDM Tech. Dig.*, p. 428, 1988.

94. Flash EEPROM array with negative gate voltage erase operation, U.S. Patent 5077691, filed:1989.
95. Kynett, V. N., Fandrich, M. L., Anderson, J., Dix, P., Jungroth, O., Hreifels, J. A., et al., A 90ns one-million erase/program cycle 1Mbit Flash memory, *IEEE J. Solid-State Circuits*., vol. SC-24, no. 5, p. 1259, 1989.
96. Endoh, T., Shirota, R., Tanaka, Y., Nakayama, R., Kirisawa, R., Aritome, S., and Masuoka, F., New design technology for EEPROM memory cells with 10 million write/erase cycling endurance, *IEDM Tech. Dig*., p. 599, 1989.
97. Shirota, R., Nakayama, R., Kirisawa, R., Momodomi, M., Sakui, K., Itoh, Y., et al., A 2.3 μm^2 memory cell structure for 16M bit NAND EEPROM's, *IEDM Tech. Dig*., p. 103, 1990.
98. Riemenschneider, B., Esquivel, A. L., Paterson, J., Gill, M., Lin, S., Schreck, J., et al., A process technology for a 5-volt only 4M bit Flash EEPROM with an 8.6 μm^2 cell, *Proc. Symp. on VLSI Technology*, p. 125, 1990.
99. Gill, M., Cleavelin, R., Lin, S., Middendorf, M., Nguyen, A., Wong, J., et al., A novel sub-lithographic tunnel diode based 5V-only Flash memory, *IEDM Tech.Dig*., p. 119, 1990.
100. Self-aligned source process and apparatus, U.S. Patent 5103274, filed:1991.
101. Woo, B. J., Ong, T. C., and Lai, S., A poly-buffered FACE technology for high density Flash memories, *Proc. Symp. on VLSI Technology*, p. 73, 1991.
102. Oyama, K., Shirai, H., Kodama, N., Kanamori, K., Saitoh, K., et al., A novel erasing technology for 3.3V Flash memory with 64 Mb capacity and beyond, *IEDM Tech. Dig*., p. 607, 1992.
103. Pein, H. and Plummer, J. D., A 3-D side-wall Flash EPROM cell and memory array, *IEEE Electron Device Lett*., vol. EDL-14, no. 8, p. 415, 1993.
104. Dhum, D. P., Swift, C. T., Higman, J. M., Taylor, W. J., Chang, K. T., Chang, K. M., and Yeargain, J. R., A novel band-to-band tunneling induced convergence mechanism for low current, high density Flash EEPROM applications, *IEDM Tech. Dig*., p. 41, 1994.
105. Tsuji, N., Ajika, N., Yuzuriha, K., Kunori, Y., Hatanaka, M., and Miyoshi, H., New erase scheme for DINOR Flash memory enhancing erase/write cycling endurance characteristics, *IEDM Tech. Dig*., p. 53, 1994.
106. Ma. Y., Pang, C. S., Chang, K. T., Tsao, S. C., Frayer, J. E., Kim, T., Jo, K., Kim, J., Choi, I., and Park, H., A dual-bit split-gate EEPROM (DSG) cell in contactless array for single Vcc high density Flash memories, *IEDM Tech. Dig*., p. 57, 1994.
107. Kato, M., Adachi, T., Tanaka, T., Sato, A., Kobayashi, T., Sudo, Y., et al., A 0.4 μm self-aligned contactless memory cell technology suitable for 256M bit Flash memory, *IEDM Tech. Dig*., p. 921, 1994.
108. Hemink, G. J., Tanaka, T., Endoh, T., Aritome, S., and Shirota, R., Fast and accurate programming method for multi-level NAND EEPROM's, *Proc. Symp. on VLSI Technology*, p. 129, 1995.
109. Hu, C.-Y., Kencke, D. L., Banerjee, S. K., Richart, R., Bandyopadhyay, B., Moore, B., Ibok, E., and Garg, S., A convergence scheme for over-erased Flash EEPROM's using substrate-bias-enhanced hot electron injection, *IEEE Electron Device Lett*., vol. EDL-16, no. 11, p. 500, 1995.
110. Bude, J. D., Frommer, A., Pinto, M. R., and Weber, G. R., EEPROM/Flash sub 3.0V drain-source bias hot carrier writing, *IEDM Tech. Dig*., p. 989, 1995.
111. Chi, M. H and Bergemont, A., Multi-level Flash/EPROM memories: new self-convergent programming methods for low-voltage applications, *IEDM Tech. Dig*., p. 271, 1995.
112. Aritome, S., Takeuchi, Y., Sato, S., Watanabe, H., Shimizu, K., Hemink, G., and Shirota, R., A novel side-wall transistor cell (SWATT cell) for multi-level NAND EEPROMs, *IEDM Tech. Dig*., p. 275, 1995.
113. Hu, C.-Y., Kencke, D. L., Banerjee, S. K., Richart, R., Bandyopadhyay, B., Moore, B., Ibok, E., and Garg, S., Substrate-current-induced hot electron (SCIHE) injection: a new convergence scheme for Flash memory, *IEDM Tech. Dig*., p. 283, 1995.
114. Gotou, H., New operation mode for stacked gate Flash memory cell, *IEEE Electron Device Lett*., vol. EDL-16, no. 3, p. 121, 1995.

115. Shin, W. C., Choi, J. D., Kim, D. J., Kim, J., Kim, H. S., Mang, K. M., et al., A new shared bit line NAND cell technology for the 256Mb Flash memory with 12V programming, *IEDM Tech. Dig.*, p. 173, 1996.
116. Papadas, C., Guillaumot, B., and Cialdella, B., A novel pseudo-floating-gate Flash EEPROM device (-cell), *IEEE Electron Device Lett.*, vol. EDL-18, no. 7, p. 319, 1997.
117. Shimizu, K., Narita, K., Watanabe, H., Kamiya, E., Takeuchi, Y., Yaegashi, T., Aritome, S., and Watanabe, T., A novel high-density $5F^2$ NAND STI cell technology suitable for 256Mbit and 1Gbit Flash memories, *IEDM Tech. Dig.*, p. 271, 1997.
118. Kobayashi, T., Matsuzaki, N., Sato, A., Katayama, A., Kurata, H., Miura, A., Mine, T., Goto, Y., et al. A 0.24 μm^2 cell process with 0.18 μm width isolation and 3-D interpoly dielectric films for 1Gb Flash memories, *IEDM Tech. Dig.*, p. 275, 1997.
119. Choi, J. D., Lee, D. G., Kim, D. J., Cho, S. S., Kim, H. S., Shin, C. H., and Ahn, S. T., A triple polysilicon stacked Flash memory cell with wordline self-boosting programming, *IEDM Tech. Dig.*, p. 283, 1997.
120. Chen, W.-M., Swift, C., Roberts, D., Forbes, K., Higman, J., Maiti, B., Paulson, W., and Chang, K.-T., A novel flash memory device with split gate source side injection and ONO charge storage stack (SPIN), *Proc. Symp. on VLSI Technology*, p. 63, 1997.
121. Kim, H. S., Choi, J. D., Kim, J., Shin, W. C., Kim, D. J., Mang, K. M., and Ahn, S. T., Fast parallel programming of multi-level NAND Flash memory cells using the booster-line technology, *Proc. Symp. on VLSI Technology*, p. 65, 1997.
122. Roy, A., Kazerounian, R., Irani, R., Prabhakar, V., Nguyen, S., Slezak, Y., et al., A new Flash architecture with a 5.8l2 scalable AMG Flash cell, *Proc. Symp. on VLSI Technology*, p. 67, 1997.
123. Lee, W.-H., Clemens, J. T., Keller, R. C., and Manchanda, L., A novel high K interpoly dielectric (IPD) Al_2O_3 for low voltage/high speed Flash memories: erasing in msec at 3.3V, *Proc. Symp. on VLSI Technology*, p. 117, 1997.
124. Kianian, S., et al., A novel 3-volt-only, small sector erase, high density Flash EEPROM, *Proc. Symp. on VLSI Tech.*, p. 71, 1994.

Cheng, K. "Dynamic Random Access Memory"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

52

Dynamic Random Access Memory

Kuo-Hsing Cheng
Tamkang University

52.1 Introduction

The first dynamic RAM (DRAM) was proposed in 1970 with a capacity of 1 Kb. Since then, DRAMs have been the major driving force behind VLSI technology development. The density and performance of DRAMs have increased at a very fast pace. In fact, the densities of DRAMs have quadrupled about every three years.

The first experimental Gb DRAM was proposed in 1995[1,2] and remains commercially available in 2000. However, multi-level storage DRAM techniques are used to improve the chip density and to reduce the defect-sensitive area on a DRAM chip.[3,4] The developments in VLSI technology have produced DRAMs that realize a cheaper cost per bit compared with other types of memories.

52.2 Basic DRAM Architecture

The basic block diagram of a standard DRAM architecture is shown in Fig. 52.1. Unlike SRAM, the addresses on the standard DRAM memory are multiplexed into two groups to reduce the address input pin counts and to improve the cost-effectiveness of packaging. Although the number of address input pin counts can be reduced by half using the multiplexed address scheme on the standard DRAM memory, the timing control of the standard DRAM memory becomes more complex and the operation speed is reduced. For high-speed DRAM applications, separate address input pins can be used to reduce the timing control complexity and to improve the operation speed.

In general, the address transition detector (ATD) circuit is not needed in a DRAM memory. DRAM controller provides Row Address Strobe ($\overline{\text{RAS}}$) and Column Address Strobe ($\overline{\text{CAS}}$) to latch in the row addresses and the column addresses. As shown in Fig. 52.1, the pins of a standard DRAM are:

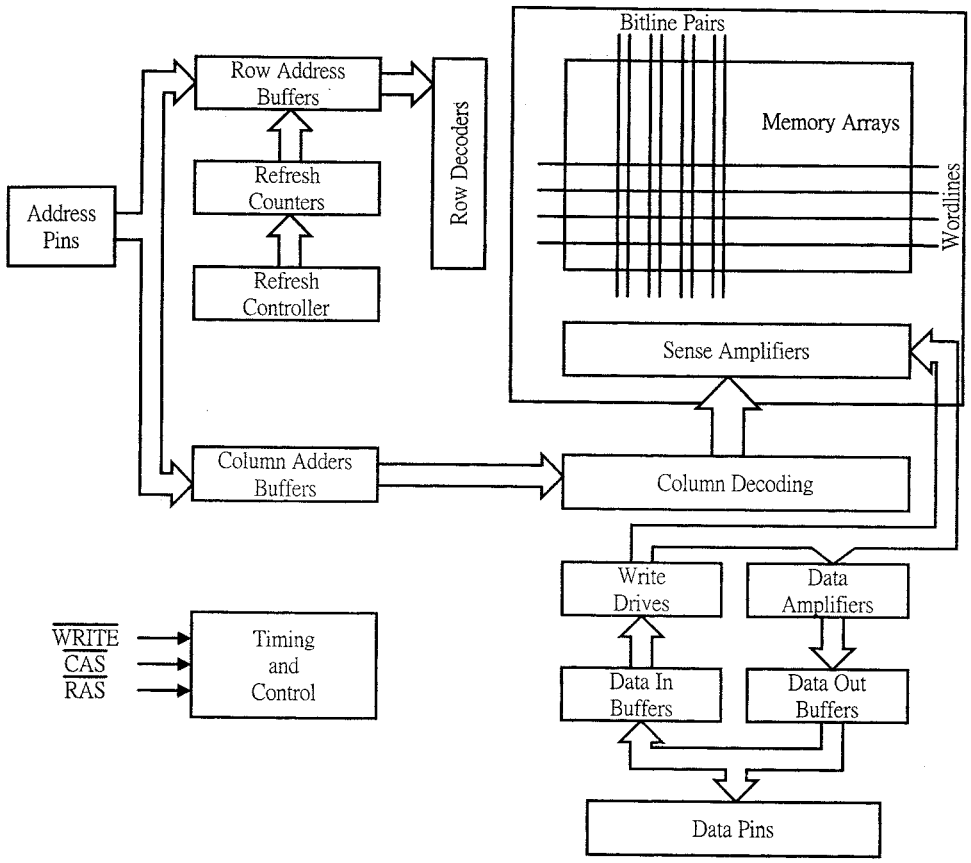


FIGURE 52.1 Basic block diagram of a standard DRAM architecture.

- Address: which are multiplexed in time into two groups, the row addresses and the column addresses
- Address control signals: the Row Address Strobe $\overline{\text{RAS}}$ and the Column Address Strobe $\overline{\text{CAS}}$
- Write enable signal: $\overline{\text{WRITE}}$
- Input/output data pins
- Power-supply pins

An example of address-multiplexed DRAM timing during basic READ mode is shown in Fig. 52.2. The row-falling edge of the address strobe ($\overline{\text{RAS}}$) samples the address and starts the READ operation mode. The row addresses are supplied into the address pins and then comes the row address strobe ($\overline{\text{RAS}}$) signal. Column addresses are not required until the row addresses are sent in and latched. The column addresses are applied into address pins and then latched in by the column address strobe ($\overline{\text{CAS}}$) signal. The access time t_{RAS} is the minimum time for the $\overline{\text{RAS}}$ signal to be low and t_{RC} is the minimum READ cycle time. Notice that the multiplexed address arrangement penalizes the access time of the standard DRAM memory.

The CMOS DRAMs have several rapid access modes in addition to the basic modes. Figure 52.3 shows an example of the rapid access modes. The timing waveform shown in Fig. 52.3 for DRAM operation is the page mode operation. In this mode, the row addresses are applied to the address pins and then clocked by the row address strobe $\overline{\text{RAS}}$ signal, and the column addresses are latched into the DRAM chip on the falling edge of $\overline{\text{CAS}}$ signal as in a basic READ mode. Along a selected row, the individual column

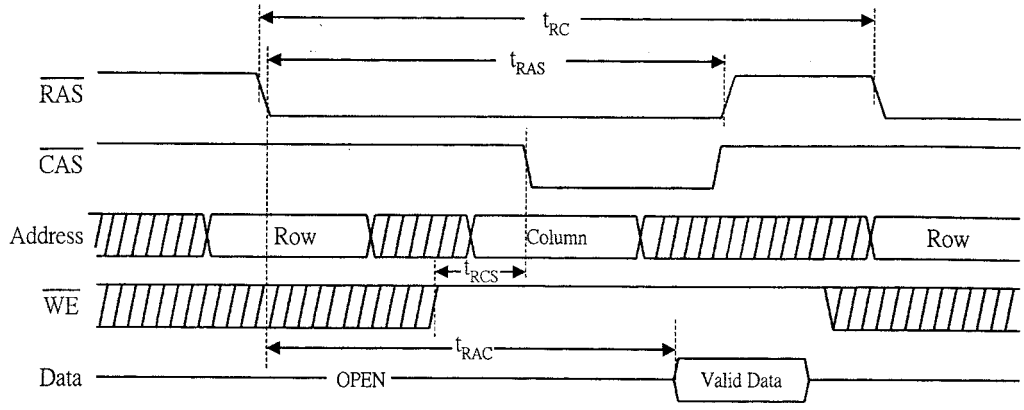


FIGURE 52.2 Read timing diagram for 4M × 1 DRAM.

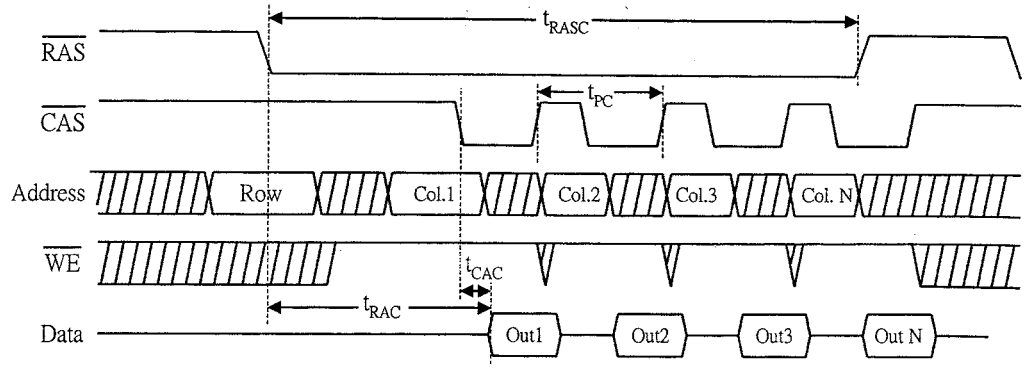


FIGURE 52.3 Fast page mode read timing diagram.

bit can be rapidly accessed, and readout is randomly controlled by the column address and the column address strobe $\overline{CAS}$. By using the page mode, the access time per bit is reduced.

52.3 DRAM Memory Cell

In early CMOS DRAM storage cell design, three-transistor and four-transistor cells were used in 1-Kb and 4-Kb generations. Later, a particular one-transistor cell, as shown in Fig. 52.4(a), became the industry standard.[5,6] The one-transistor (1T) cell achieves smaller cell size and low cost. The cell consists of an n-channel MOSFET and a storage capacitor *Cs*. The charge is stored in the capacitor *Cs* and the n-channel MOSFET functions as the access transistor. The gate of the n-channel MOSFET is connected to the word-line *WL* and its source/drain is connected to the bit-line. The bit-line has a capacity C_{BL}, including the parasitic load of the connected circuits.

The DRAM cell stores one bit of information as the charge on the cell storage capacitor *Cs*. Typical values for the storage capacitor *Cs* are 30 to 50 fF. When the cell stores "1", the capacitor is charged to $V_{DD} - V_t$. When the stores "0", the capacitor is discharged to 0 V.

During the READ operation, the voltage of the selected word-line is high; the access n-channel MOSFET is turned on, thus connecting the storage capacitor *Cs* to the bit-line capacitance C_{BL} as shown in Fig. 52.4(b). The bit-line capacitance C_{BL}, including the parasitic load of the connected circuits, is

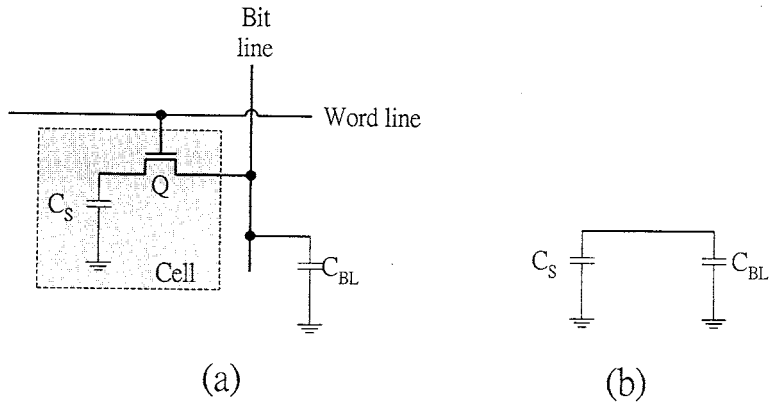


FIGURE 52.4 (a) The one-transistor DRAM cell; and (b) during the READ operation, the voltage of the selected word-line is high, thus connecting the storage capacitor *Cs* to the bit-line capacitance C_{BL}.

about 30 times larger than the storage capacitor *Cs*. Before the selection of the DRAM cell, the bit-line is precharged to a fixed voltage, typically $V_{DD}/2$.[7] By using the charge conservation principle, during the READ operation, the bit-line voltage changes by

$$V_s = \Delta V_{BL} = \frac{C_S}{C_{BL} + C_S}\left(V_{cs} - \frac{V_{DD}}{2}\right) \tag{52.1}$$

Here, V_{cs} is the storage voltage on the DRAM cell capacitor *Cs*. A ratio $R = C_{BL}/Cs$ is important for the read sensing operation. If the cell stores "1" with a voltage $V_{cs} = V_{DD} - V_t$, we have the small bit-line sense signal

$$\Delta V(1) = \frac{1}{1+R}\left(\frac{V_{DD}}{2} - V_t\right) \tag{52.2}$$

If the cell stores "0" with a voltage $V_{cs} = 0$, we have the small bit-line sense signal

$$\Delta V(0) = \frac{1}{1+R}\left(\frac{V_{DD}}{2}\right) \tag{52.3}$$

Since ratio $R = C_{BL}/Cs$ is large, these readout bit-line sense signals $\Delta V(1)$ and $\Delta V(0)$ are very small. Typical values for the sense signal are about 100 mv.

For low-voltage operation, the supply voltage V_{DD} is reduced. Thus, a lower *R* ratio is required to maintain the sense signals to have enough margin against noise. The main approach is to use a large cell storage capacitor *Cs*. As shown in Fig. 52.5, a conventional *Cs* was implemented by a simple planar-type capacitor. The charge storage in the cell takes place on both the poly-1 gate oxide and the depletion capacitances. The planar DRAM cells have been used in the 1-T DRAMs from the 16 kb to the 1 Mb. The limits of the planar DRAM cell for retaining sufficient capacitance were reached in the mid-1980s in the 1-Mb DRAM. With the increased density higher than 1 Mb, smaller horizontal geometry on the surface of the wafer can be achieved by making increased use of the vertical dimension.[8] One approach is to use a trench capacitor, as shown in Fig. 52.6(a).[9] It is folded vertically into the surface of the silicon in the form of a trench. Another approach for reducing horizontal capacitor size is to stack the capacitor *Cs* over the n-channel MOSFET access transistor, as shown in Fig. 52.6(b).

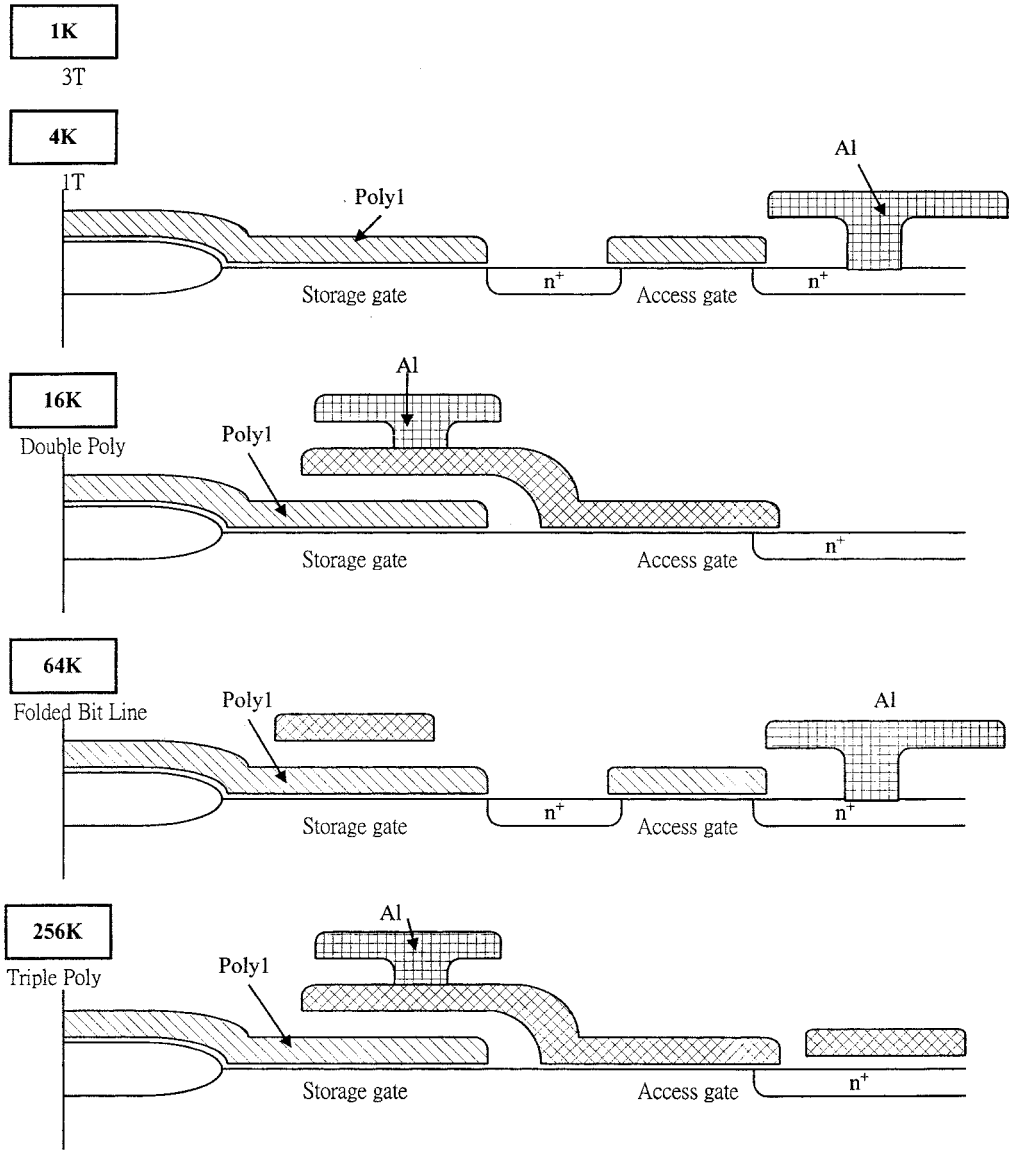


FIGURE 52.5 Structural innovations of planar DRAM cells.

52.4 Read/Write Circuit

As shown in the previous section, the readout process is destructive because the resulting voltage of the cell capacitor *Cs* will no longer be $(V_{DD} - V_t)$ or 0 V. Thus, the same data must be amplified and written to the cell in every readout process.

Next to the storage cells, a sense amplifier with positive feedback structure, as shown in Fig. 52.7, is the most important component in a memory chip to amplify the small readout signal in the readout process. The input and output nodes of the differential positive feedback sense amplifier are connected to the bit-lines *BL* and $\overline{BL}$. The small readout signal appearing between *BL* and $\overline{BL}$ is detected by the differential sense amplifier and amplified to a full-voltage swing at *BL* and $\overline{BL}$. For example, if the DRAM

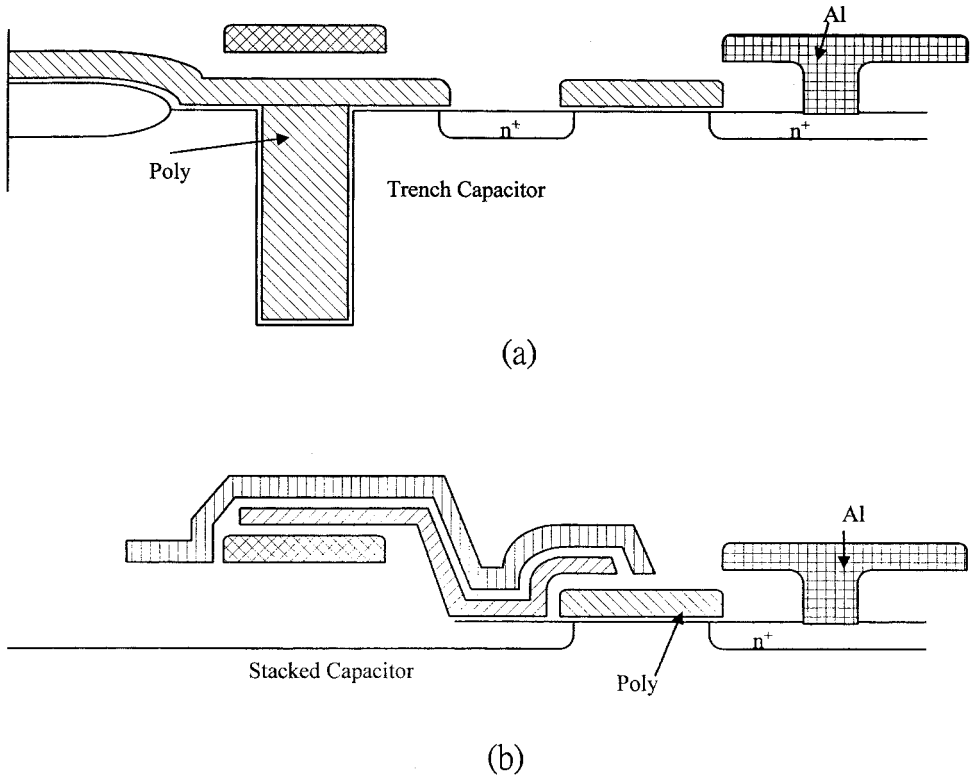


FIGURE 52.6 Schematic cross-section of DRAM cells: (a) trench capacitor cell, and (b) stacked capacitor cell.

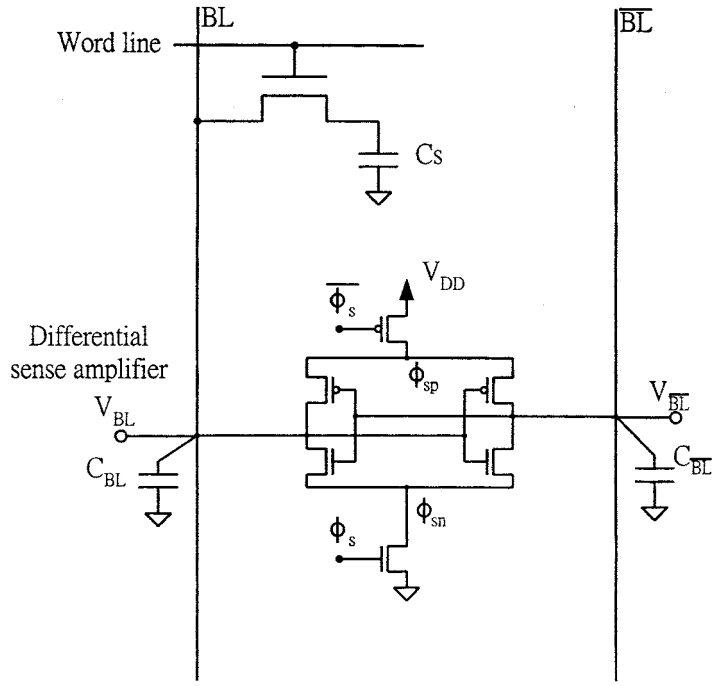


FIGURE 52.7 A differential sense amplifier connected to the bit-line.

memory cell in *BL* has a stored data "1", then a small positive voltage $\Delta V(1)$ will be generated and added to the bit-line *BL* voltage after the readout process. The voltage in the bit-line *BL* will be $\Delta V(1) + V_{DD}/2$. In the same time, the bit-line $\overline{BL}$ will keep its previous precharged voltage level, which is precharged to $V_{DD}/2$. Thus, the small positive voltage $\Delta V(1)$ appears between *BL* and $\overline{BL}$, with V_{BL} higher than $V_{\overline{BL}}$, immediately after the readout process. It is amplified by the differential sense amplifier. The waveforms of V_B before and after activating the sense amplifier are shown in Fig. 52.8. After the sensing and restoring operations, the voltage V_{BL} rises to V_{DD}, and the voltage $V_{\overline{BL}}$ falls to 0 V. The output at *BL* is then sent to the DRAM output pin.

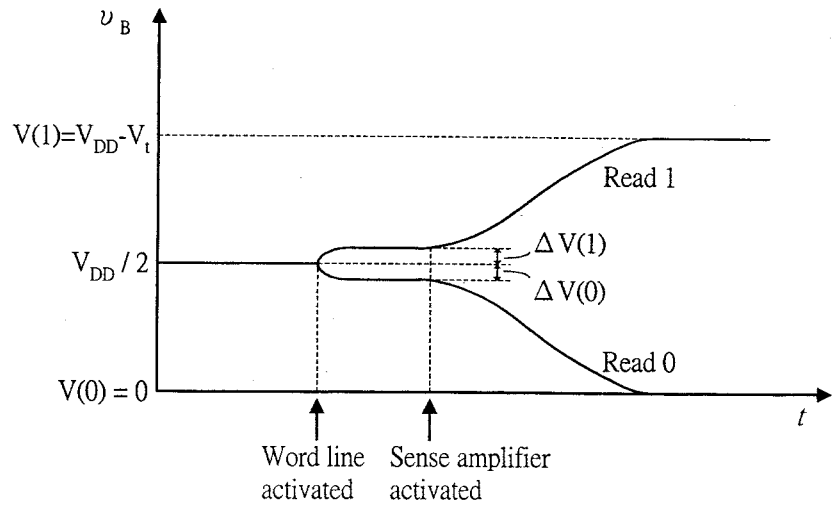


FIGURE 52.8 Timing waveform of V_B.

The various circuits for read, write precharge, and equalization function are shown in Fig. 52.9. The sequence of the read operation is performed as follows.

1. Initially, both the bit-lines *BL* and $\overline{BL}$ are precharged to $V_{DD}/2$ and equalized before the data readout process. The precharge and equalizer circuits are activated by rising the control signal Φ_p. This will cause the bit-lines *BL* and $\overline{BL}$ to be at equal voltage. The control signal Φ_p goes low after the precharge and equalization.
2. The signal *WL* is selected by the row decoder. It goes up to connect the storage cell to the bit-lines *BL* and $\overline{BL}$. A small voltage difference then appears between the bit-lines. The voltage level of the word-line signal *WL* can be greater than V_{DD} to overcome the threshold voltage drop of the n-channel MOSFET transistor. Thus, the stored voltage level of data "1" at the memory cell can be raised to V_{DD}.
3. Once a small voltage difference is generated between the bit-lines *BL* and $\overline{BL}$ by the storage cell, the differential sense amplifier is turned on by pulsing the sense control signal Φ_s high and the sense control signal Φ_s low. Then, the small voltage difference is amplified by the differential sense amplifier. The voltage levels in *BL* and $\overline{BL}$ will quickly move to V_{DD} or 0 V by the regenerative action of the positive feedback operation in the differential sense amplifier.
4. After the readout sensing and restoring operations, the voltage levels of the bit-lines have a full voltage swing. Then the differential voltage levels at the bit-lines are read out to the differential output lines *O* and $\overline{O}$, through a read circuit. A main sense amplifier is used to read and to amplify the output-lines. After these processes, the output data is selected and transferred to the output buffer.

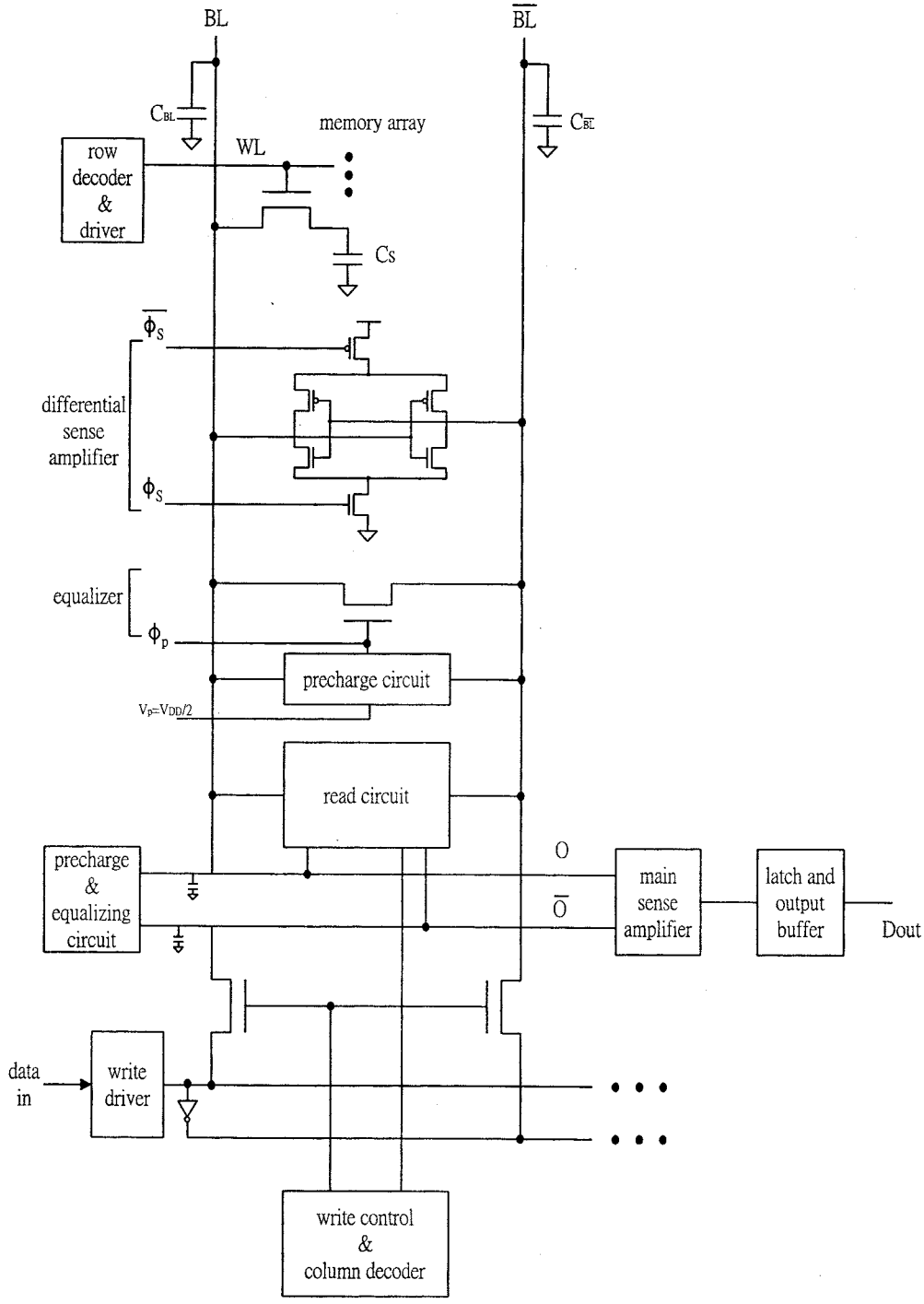


FIGURE 52.9 (a) Schematic circuit diagram of DRAM.

In the write mode, the write control signal $\overline{WRITE}$ is activated. Selected bit-lines *BL* and $\overline{BL}$ are connected to a pair of input data controlled by the write control and write driver. The write circuit drives the voltage levels at the bit-lines to V_{DD} or 0 V, and the data are transferred to the DRAM cell when access transistor is turned on.

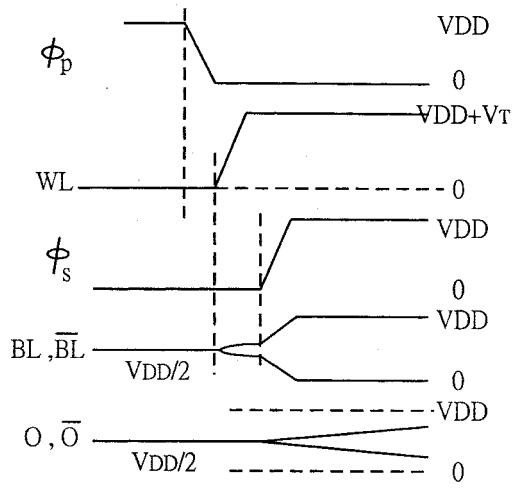


FIGURE 52.9(b) READ operation waveforms.

52.5 Synchronous (Clocked) DRAMs

The application of multimedia is a very hot topic nowadays, and the multimedia systems require high speed and large memory capacity to improve the quality of data processing. Under this trend, high density, high bandwidth, and fast access time are the key requirements of future DRAMs.

The synchronous DRAM (SDRAM) has the characteristic of fast access speed, and is widely used for memory application in multimedia systems. The first SDRAM appeared in the 16-Mb generation, and the current state-of-the-art product is a Gb SDRAM with GB/s bandwidth.[10–14]

Conventionally, the internal signals in asynchronous (non-clocked) DRAMs are generated by "address transition detection" (ATD) techniques. The ATD clock can be used to activate the address decoder and driver, the sense amplifier, and the peripheral circuit of DRAMs. Therefore, the asynchronous DRAMs require no external system clocks and have a simple interface. However, during the asynchronous DRAM access cycle, the process unit must wait for the data from the asynchronous DRAM, as shown in Fig. 52.10. Therefore, the speed of the asynchronous DRAM is slow.

On the other hand, the synchronous interface (clocked) DRAMs making it under the control of the edge of the system clock. The input addresses of a synchronous DRAM are latched into the DRAM, and the output data is available after a given number of clock cycles — during which the processor unit is

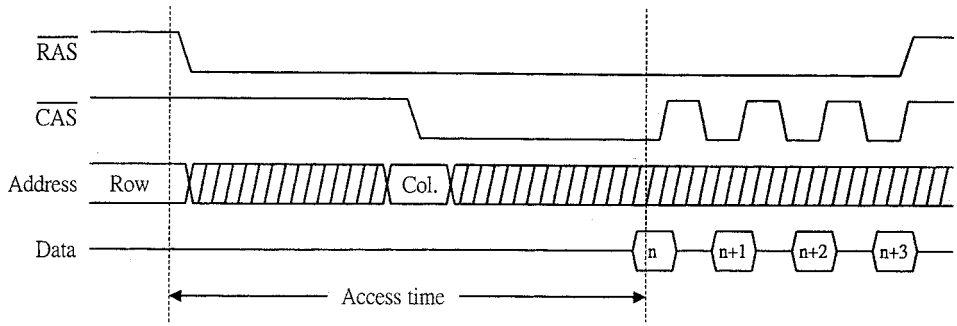


FIGURE 52.10 Read cycle timing diagram for asynchronous DRAM.

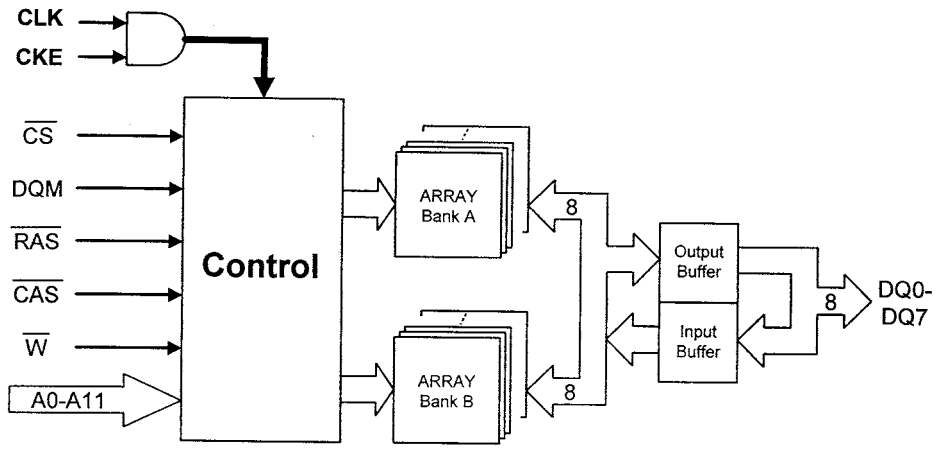


FIGURE 52.11 Read cycle timing diagram for synchronous DRAM.

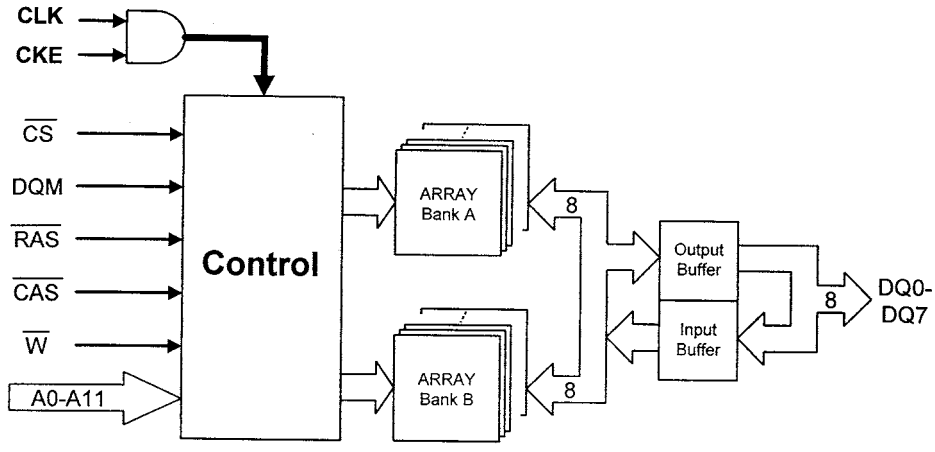


FIGURE 52.12 Block diagrams of a synchronous DRAM.

free and does not wait for the data from the SDRAM, as shown in Fig. 52.11. The block diagram of an SDRAM is shown in Fig. 52.12. With the synchronous interface scheme, the effective operation speed of a given system is improved.

52.6 Prefetch and Pipelined Architecture in SDRAMs

The system clock activates the SDRAM architecture. In order to speed up the average access time, it is possible to use the system clock to store the next address in the input latch or to be sequentially clocked out for each address access output from the output buffer, as shown in Fig. 52.13.[15]

During the read cycle of the prefetch SDRAM, more than one data word is fetched from the memory array and sent to the output buffer. Using the system clock to control the prefetch register and buffer, multiple words of data can be sequentially clocked out for each address access. As shown in Fig. 52.13, the SDRAM has a 6-clock-cycle RAS latency to prefetch 4-bit data.

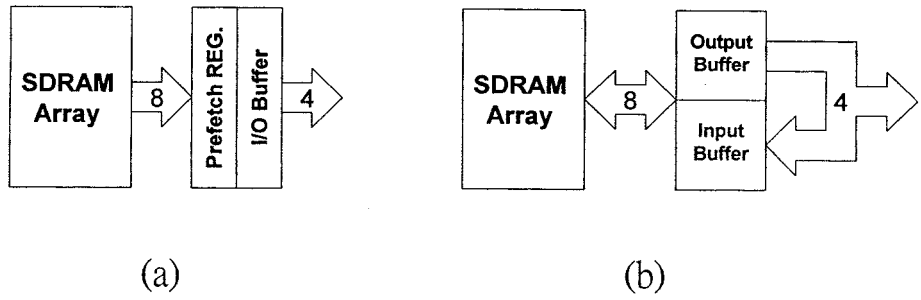


FIGURE 52.13 Block diagrams of two types of synchronous DRAM output: (a) prefetch (b) pipelined.

52.7 Gb SDRAM Bank Architecture

To consider the Gb SDRAM realization, the chip layout and bank/data bus architecture is important for data access. Figure 52.14 shows the conventional bank/data bus architecture of 1-Gb SDRAM.[16] It contains 64 DQ pins, 32 × 32-Mb SDRAM blocks, and four banks; and they all prefetch 4 bits. During the read cycle, the eight 32-Mb DRAM blocks of one bank are accessed simultaneously. The 256-bit data is accessed to the 64 DQ pins and 4 bits are prefetched. In an activated 32-Mb array block, 32-bit data is accessed and associated with eight specific DQ pins. Therefore, it requires a data I/O bus switching circuit between the 32-Mb SDRAM bank and the eight DQ pins. It makes the data I/O bus more complex, and the access time is slower.

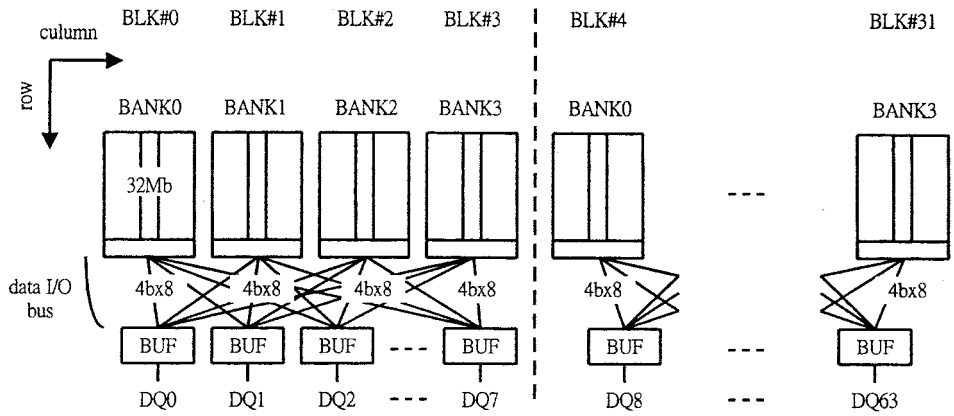


FIGURE 52.14 1-Gb SDRAM bank/data bus architecture.

In order to simplify the bus structure, the distributed bank (D-bank) architecture is proposed as shown in Fig. 52.15. The 1-Gb SDRAM is implemented by 32 × 32-Mb distributed banks. A 32-Mb distributed bank contains two 16-Mb memory arrays as shown in Fig. 52.16. The divided word-line technique is used to activate the segment along the column direction. Using this scheme, each of the eight 2-Mb segments is selectively activated; sense amplifiers of one of the eight segments are activated; and all the 16-K sense amplifiers are activated simultaneously. As compared with the conventional architecture, the distributed bank architecture has a much simplified data I/O bus structure.

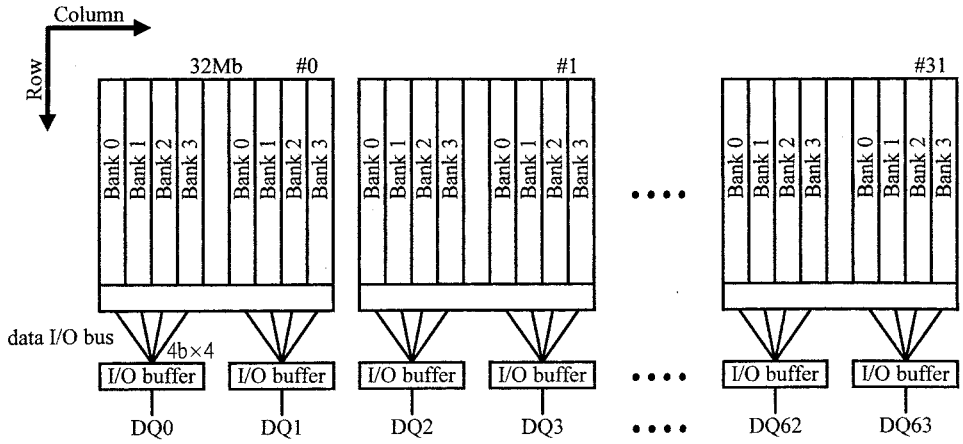


FIGURE 52.15 1-Gb SDRAM D-bank architecture.

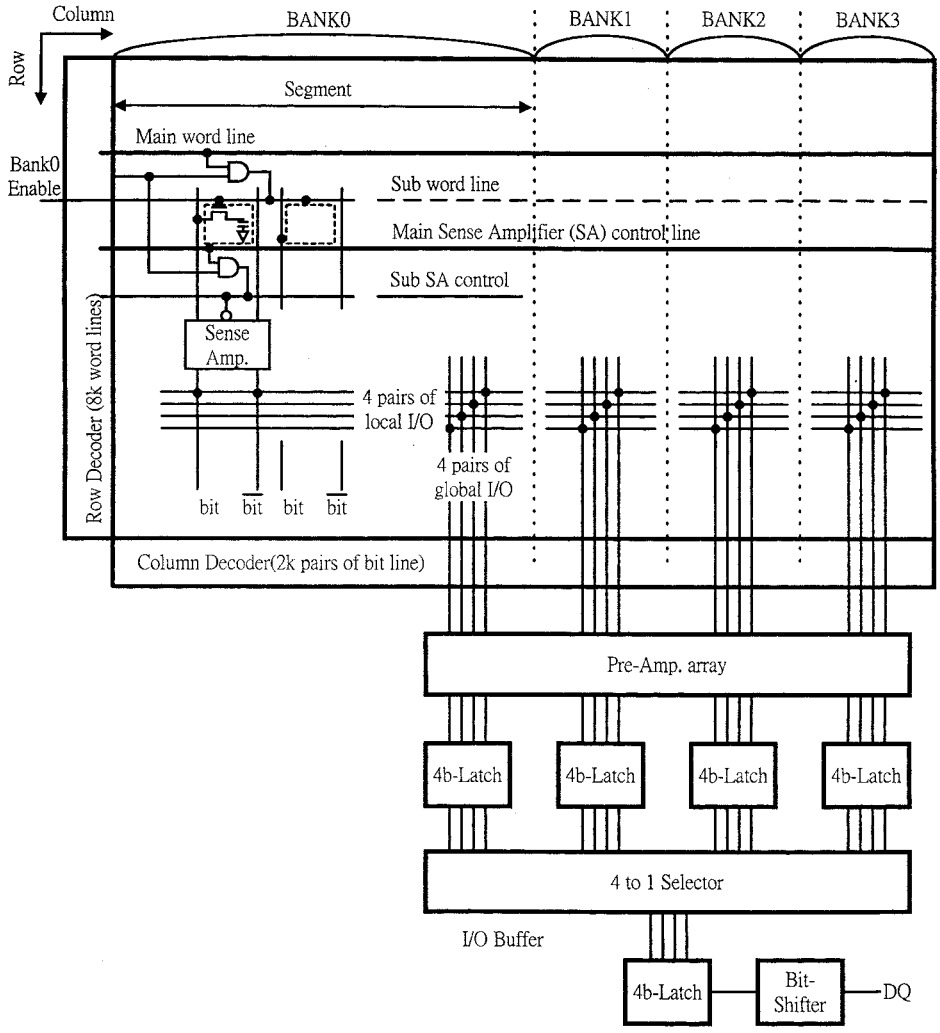


FIGURE 52.16 16-Mb memory array for architecture D-bank.

52.8 Multi-level DRAM

In modern application-specific IC (ASIC) memory designs, there are some important items — memory capacity, fabrication yield, and access speed — that need to be considered. The memory capacity required for ASIC application has been increasing very rapidly, and the bit-cost reduction is one of the most important issues for file application DRAMs. In order to achieve high yield, it is important to reduce the defect-sensitive area on a chip.

The multi-level storage DRAM technique is one of the circuit technologies that can reduce the effective cell size. It can store multiple voltage levels in a single DRAM cell. For example, in a four-level system, each DRAM cell corresponds to 2-bit data of "11", "10", "01", and "00". Thus, the multi-level storage technique can improve the chip density and reduce the defect-sensitive area on a DRAM chip, and it is one of the solutions to the "density and yield" problem.

52.9 Concept of 2-bit DRAM Cell

The 2-bit DRAM is an important architecture in the multi-level DRAM. Let us discuss an example of a multi-level technique used for a 4-Gb DRAM by NEC.[17] Table 52.1 lists both the 2-bit/4-level storage concept and the conventional 1-bit/2-level storage concept. In the conventional 1-bit/2-level DRAM cell, the storage voltage levels are *Vcc* or *GND,* corresponding to logic values "1" or "0". The signal charge is one half the maximum storage charge. In the 2-bit/4-level DRAM cell, the storage voltage levels are *Vcc*, two-thirds *Vcc*, one-third *Vcc*, and *GND,* corresponding to logic values "11", "10", "01", and "10", respectively. Three reference voltage levels are used to detect these four storage levels. Reference levels are positioned at the midlevel between the four storage levels. Thus, the signal charge between the storage and reference levels is one sixth of the maximum storage charge.

TABLE 52.1 Four-Level Storage

	Four-Level Storage			
	Data	Storage Voltage Level	Reference Level	Signal Level
	11	Vcc		1/6 Vcc
			5/6 Vcc	
4-Level	10	2/3 Vcc		
(2-bit)			3/6 Vcc	
Storage	01	1/3 Vcc		
			1/6 Vcc	
	00	GND		
2-Level	1	Vcc	1/2 Vcc	1/2 Vcc
Storage	0	GND		

Sense and Timing Scheme

The circuit diagram of the 2-bit/4-level storage technique is shown in Fig. 52.17. A pair of bit-lines is separated into two sections by transfer switches in order to have a capacitance ratio of two between Sections A and B.

Two sense amplifiers and two cross-coupled capacitors *Cc* are connected to each section. During the stand-by cycle, the transfer signal TG is high and the transfer switch is turned ON. The bit-lines are precharged to the half-*Vcc* level. As shown in Fig. 52.17(b), at time T1, the circuit is operated in the active cycle, and a word-line is selected and the charge stored in the cell *Cs* is transferred to the bit-lines. At time T2, the transfer switches are turned OFF and the bit-lines are isolated. At time T3, the sense amplifier in Section A is activated and the bit-lines in Section A are driven to *Vcc* and *GND,* depending on the stored data. The amplified data in Section A is the most significant bit (MSB) of the stored data because the reference level is half-*Vcc*.

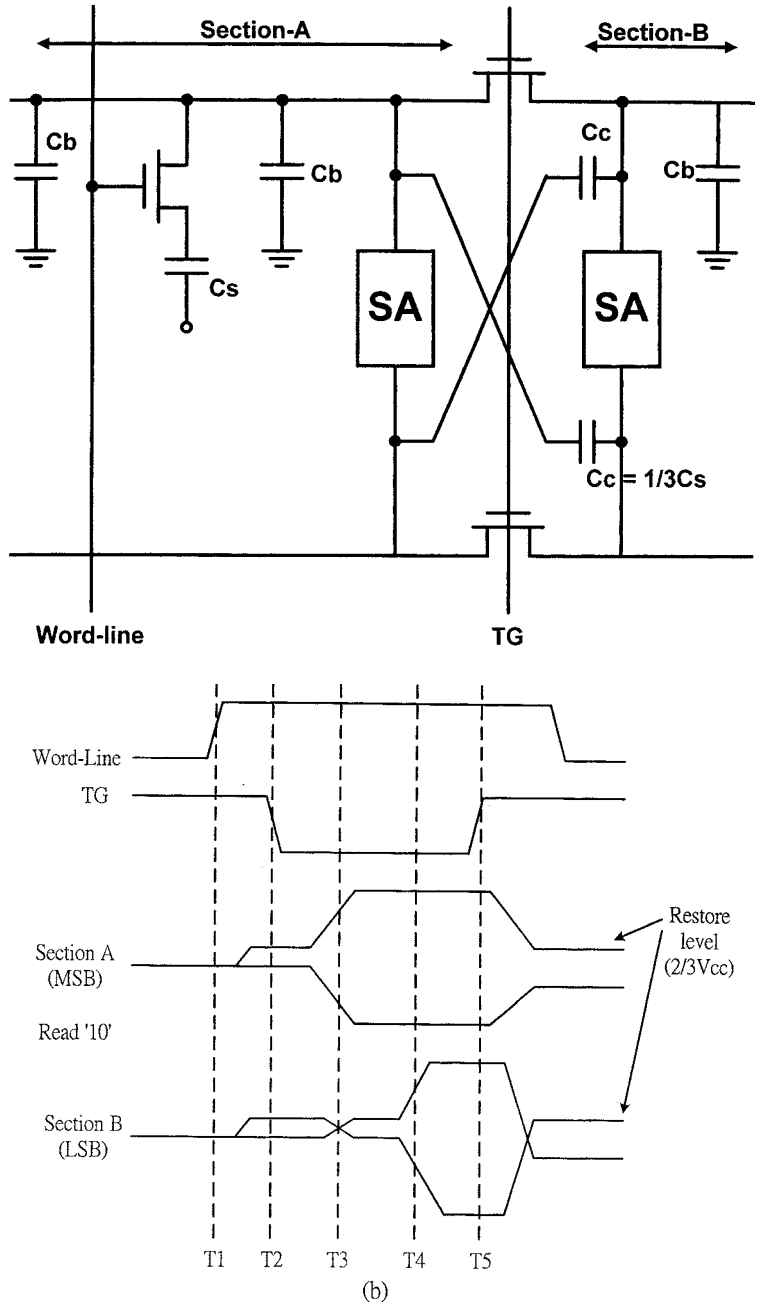


FIGURE 52.17 Principle of sense and restore: (a) circuit diagram, and (b) timing diagram.

At the same time interval, the MSB is transferred to the bit-lines in Section B through a cross-coupled capacitor *Cc*. It can change the bit-line level in Section B for subsequent least significant bit (LSB) sensing. At time T4, the sense amplifier in section B is activated and the LSB is sensed. At time T5, the transfer switch is turned ON, the charge on each bit-line is shared, and the read-out data is restored to the memory cell.

Charge-Sharing Restore Scheme

Table 52.2 lists the restored level generated by the charge-sharing restore scheme. The MSB is latched in Section A, and the LSB is latched in Section B. The capacitance ratio between Sections A and B is 2. The charge of the MSB and the charge of the LSB are combined on the bit-line, and the restore level $V_{restore}$ is generated.

TABLE 52.2 Charge-Sharing Restore Scheme

Charge–Sharing Restore Scheme				
		MSB		
Restore Level		1	0	$V_{restore} = Vcc\frac{2Cb \bullet MSB + Cb \bullet LSB}{3Cb}$
LS	1	Vcc	1/3 Vcc	
B	0	2/3 Vcc	0 (GND)	

Charge-Coupling Sensing

Figure 52.18 shows the charge in bit-line levels due to coupling capacitor *Cc*. The MSB is sensed using the reference level of half-*Vcc*, as mentioned earlier. The MSB generates the reference level for LSB sensing. When V_s is defined as the absolute signal level of data "11" and "00", the absolute signal level of data "10" and "01" is one-third of V_s. Here, V_s is directly proportional to the ratio between storage capacitor *Cs* and bit-line capacitance.

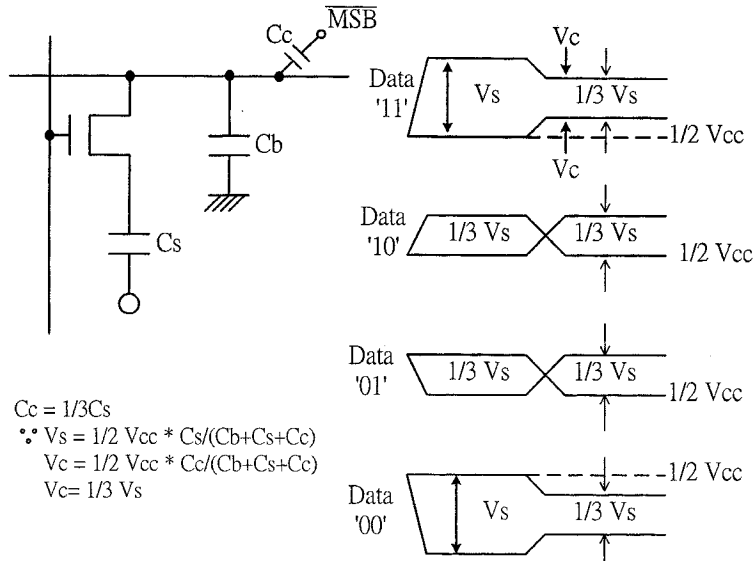


FIGURE 52.18 Charge-coupling sensing.

In the case of sensing data "11", the initial signal level is V_s. After MSB sensing, the bit-line level in Section B is changed for LSB sensing by the MSB through coupling capacitor *Cc*. The reference bit-line in Section B is raised by *Vc*, and the other bit-line is reduced by *Vc*. For LSB sensing, *Vc* is one-third of V_s due to the coupling capacitor *Cc*.

Using the two-step sensing scheme, the 2-bit data in a DRAM cell can be implemented.

References

1. Sekiguchi., T. et al., "An Experimental 220MHz 1Gb DRAM," *ISSCC Dig. Tech. Papers*, pp. 252-253, Feb. 1995.
2. Sugibayashi, T., et al., "A 1Gb DRAM for File Applications," *ISSCC Dig. Tech. Papers*, pp. 254-255, Feb. 1995.
3. Murotani, T. et al., "A 4-Level Storage 4Gb DRAM," *ISSCC Dig. Tech. Papers*, pp. 74-75, Feb. 1997.
4. Furuyama, T. et al., "An Experimental 2-bit/Cell Storage DRAM for Macrocell or Memory-on-Logic Application," *IEEE J. Solid-State Circuits*, vol. 24, no. 2, pp. 388-393, April 1989.
5. Ahlquist, C.N. et al., "A 16k 384-bit Dynamic RAM," *IEEE J Solid-State Circuits,* vol. SC- 11, no. 3, Oct. 1976.
6. El-Mansy, Y. et al., "Design Parameters of the Hi-C SRAM cell," *IEEE J. Solid-State Circuits*, vol. SC-17, no. 5, Oct. 1982.
7. Lu, N. C. C., "Half-V_{DD} Bit-Line Sensing Scheme in CMOS DRAM's," *IEEE J. Solid-State Circuits*, vol. SC-19, no. 4, Aug. 1984.
8. Lu, N. C. C., "Advanced Cell Structures for Dynamic RAMs," *IEEE Circuits and Devices Magazine*, pp. 27-36, Jan. 1989.
9. Mashiko, K. et al., "A 4-Mbit DRAM with Folded-Bit-Line Adaptive Sidewall-Isolated Capacitor (FASIC) Cell," *IEEE J. Solid-State Circuits,* vol. SC-22, no. 5, Oct. 1987.
10. Prince, B., et al., "Synchronous Dynamic RAM," *IEEE Spectrum*, p. 44, Oct. 1992.
11. Yoo, J.-H. et al., "A 32-Bank 1Gb DRAM with 1GB/s Bandwidth," *ISSCC Dig. Tech. Papers*, pp. 378-379, Feb. 1996.
12. Nitta, Y. et al., "A 1.6GB/s Data-Rate 1Gb Synchronous DRAM with Hierarchical Square-Shaped Memory Block and Distributed Bank Architecture," *ISSCC Dig. Tech. Papers*, pp. 376-377, Feb. 1996
13. Yoo, J.-H. et al., "A 32-Bank 1 Gb Self-Strobing Synchronous DRAM with 1 Gbyte/s Bandwidth," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, pp. 1635-1644, Nov. 1996.
14. Saeki, T. et al., "A 2.5-ns Clock Access, 250-MHz, 256-Mb SDRAM with Synchronous Mirror Delay," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, pp. 1656-1668, Nov. 1996.
15. Choi, Y. et al., "16Mb synchronous DRAM with 125Mbyte/s data rate," *IEEE J. Solid-State Circuits*, vol. 29, no. 4, April 1994.
16. Sakashita, N. et al., "A 1.6GB/s Data-Rate 1-Gb Synchronous DRAM with Hierarchical Square Memory Block and Distributed Bank Architecture," *IEEE J. Solid-State Circuits*, vol. 31, no. 11, pp. 1645-1655, Nov. 1996.
17. Okuda, T. et al., "A Four-Level Storage 4-Gb DRAM," *IEEE J. Solid-State Circuits*, vol. 32, no. 11, pp. 1743-1747, Nov. 1997.
18. Prince, B., *Semiconductor Memories*, 2nd edition, John Wiley & Sons, 1993.
19. Prince, B., *High Performance Memories New Architecture DRAMs and SRAMs Evolution and Function*, 1st edition, Betty Prince, 1996.
20. *Toshiba Applications Specific DRAM Databook*, D-20, 1994.

Margala, M. "Low-Power Memory Circuits"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

53

Low-Power Memory Circuits

Martin Margala
University of Alberta

53.1 Introduction

In recent years, rapid development in VLSI fabrication has led to decreased device geometries and increased transistor densities of integrated circuits, and circuits with high complexities and very high frequencies have started to emerge. Such circuits consume an excessive amount of power and generate an increased amount of heat. Circuits with excessive power dissipation are more susceptible to run time failures and present serious reliability problems. Increased temperature from high-power processors tends to exacerbate several silicon failure mechanisms. Every 10°C increase in operating temperature approximately doubles a component's failure rate. Increasingly expensive packaging and cooling strategies are required as chip power increases.[1,2] Due to these concerns, circuit designers are realizing the importance of limiting power consumption and improving energy efficiency at all levels of design. The second driving force behind the low-power design phenomenon is a growing class of personal computing devices, such as portable desktops, digital pens, audio- and video-based multimedia products, and wireless communications and imaging systems, such as personal digital assistants, personal communicators, and smart cards. These devices and systems demand high-speed, high-throughput computations, complex functionalities, and often real-time processing capabilities.[3,4] The performance of these devices is limited by the size, weight, and lifetime of batteries. *Serious reliability problems, increased design costs, and battery-operated applications have prompted the IC design community to look more aggressively for new approaches and methodologies that produce more power-efficient designs, which means significant reductions in power consumption for the same level of performance.*

Memory circuits form an integral part of every system design as dynamic RAMs, static RAMs, ferroelectric RAMs, ROMs, or Flash memories significantly contribute to system-level power consumption. Two examples of recently presented reduced-power processors show that 43% and 50.3%, respectively, of the total system power consumption is attributed to memory circuits.[5,6] Therefore, reducing the power dissipation in memories can significantly improve the system power-efficiency, performance, reliability, and overall costs.

In this chapter, all sources of power consumption in different types of memories will be identified; several low-power techniques will be presented; and the latest developments in low-power memories will be analyzed.

53.2 Read-Only Memory (ROM)

ROMs are widely used in a variety of applications (permanent code storage for microprocessors or data look-up tables in multimedia processors) for fixed long-term data storage. The high area density and new submicron technologies with multiple metal layers increase the popularity of ROMs for a low-voltage, low-power environment. In the following section, sources of power dissipation in ROMs and applicable efficient low-power techniques are examined.

Sources of Power Dissipation

A basic block diagram of a ROM architecture is presented in Fig. 53.1.[7,8] It consists of an address decoder, a memory controller, a column multiplexer/driver, and a cell array. Table 53.1 lists an example of a power dissipation in a 2 K × 18 ROM designed in 0.6-μm CMOS technology at 3.3 V and clocked at 10 MHz.[8] The cell array dissipates 89% of the total ROM power, and 11% is dissipated in the decoder, control logic, and the drivers. The majority of the power consumed in the cell array is due to the precharging of large capacitive bit-lines. During the read and write cycles, more than 18 bit-lines are switched per access because the word-line selects more bit-lines than necessary. The example in Fig. 53.2 shows a 12-1 multiplexer and a bit-line with five transistors connected to it. This topology consumes excessive amounts of power because 4 more bit lines will switch instead of just one. The power dissipated in the decoder, control logic, and drivers is due to the switching activity during the read and precharge cycles and generating control signals for the entire memory.

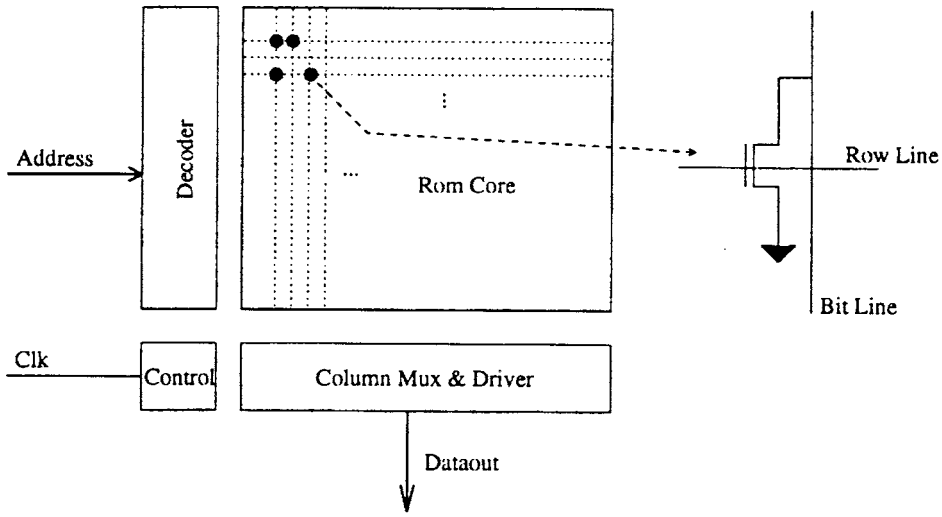


FIGURE 53.1 Basic ROM architecture. (© 1997, IEEE. With permission.)

Low-Power ROMs

In order to significantly reduce the power consumption in ROMs, every part of the architecture has to be targeted and multiple techniques have to be applied. Angel and Swartzlander[8] have identified several architectural improvements in the cell array that minimize energy waste and improve efficiency. These techniques include:

TABLE 53.1 Power Dissipation ROM 2 K × 18

Block **	Power (mW)	Percentage (%)
Decoder	0.06	2.1
ROM core	2.24	89
Control	0.18	7.2
Drivers	0.05	1.7

(Source: © 1997, IEEE. With permission.)

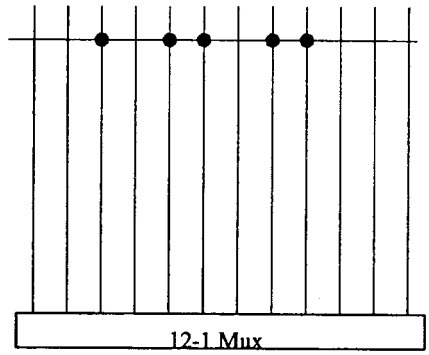


FIGURE 53.2 ROM bit-lines. (© 1997, IEEE. With permission.)

- Hierarchical word line
- Selective precharging
- Minimization of non-zero terms
- Inverted ROM core(s)
- Row(s) inversion
- Sign magnitude encoding
- Sign magnitude and inverted block
- Difference encoding
- Smaller cell arrays

All of these methods result in a reduction of the capacitance and/or switching activity of bit- and row-lines. A *hierarchical word-line* approach divides memory in separate blocks and runs the block word-line in one layer and a global word-line in another layer. As a result, only the bit cells of the desired block are accessed. A *selective precharging* method addresses the problem of activating multiple bit-lines, although only a single memory location is being accessed. By using this method, only those bit-lines that are being accessed are precharged. The hardware overhead for implementing this function is minimum. A *minimization of non-zero terms* reduces the total capacitance of bit- and row-lines because zero-terms do not switch bit-lines. This also reduces the number of transistors in the memory core. An *inverted ROM* applies to a memory with a large number of 1s. In this case, the entire ROM array could be inverted and the final data will be inverted back in the output driver circuitry. Consequently, the number of transistors and the capacitance of bit- and row-lines are reduced. An *inverted row* method also minimizes non-zero terms, but on a row-by-row basis. This type of encoding requires an extra bit (MSB) that indicates whether or not a particular row is encoded. A *sign and magnitude* encoding is used to store negative numbers. This method also minimizes the number of 1s in the memory. However, a two's complement conversion is required when data are retrieved from the memory. A *sign and magnitude and an inverted block* is a combination of the two techniques described previously. A *difference encoding* can be used to reduce the size of the cell array. In applications where

a ROM is accessed sequentially and the data read from one address does not change significantly from the following address, the memory core can store the difference between these two entries instead of the entire value. The disadvantage is a need for an additional adder circuit to calculate the original value. In applications where different bit sizes of data are needed, *smaller memory arrays* are useful to implement. If stored in a single memory array, its bit size is determined by the largest number. However, most of the bit positions in smaller numbers are occupied by non-zero values that would increase the bit-line and row-line capacitance. Therefore, by grouping the data to smaller memory arrays according to their size, significant savings in power can be achieved.

On the circuit level, powerful techniques that minimize the power dissipation can be applied. The most common technique is reducing the power supply voltage to approximately $V_{dd} \approx 2V_t$ in a correlation with the architectural-based scaling. In this region of operation, the CMOS circuits achieve the maximum power efficiency.[9,10] This results in large power savings because the power supply is a quadratic term in a well-known dynamic power equation. In addition, the static power and short-circuit power are also reduced. It is important that all the transistors in the decoder, control logic, and driver block be sized properly for low-power, low-voltage operation. Rabaey and Pedram[9] have shown that the ideal low-power sizing is when $C_d = C_L/2$, where C_d is the total parasitic capacitance from driving transistors and C_L is the total load capacitance of a particular circuit node. By applying this method to every circuit node, a maximum power efficiency can be achieved. Third, different logic styles should be explored for the implementation of the decoder, control logic, and drivers. Some alternative logic styles are superior to standard CMOS for low-power, low-voltage operation.[11,12] Fourth, by reducing the voltage swing of the bit-lines, significant reduction in switching power can be obtained. One way of implementing this technique is to use NMOS precharge transistors. The bit-lines are then precharged to $V_{dd} - V_t$. A fifth method can be applied in cases when the same location is accessed repeatedly.[8] In this case, a circuit called a *voltage keeper* can be used to store past history and avoid transitions in the data bus and adder (if sign and magnitude is implemented). The sixth method involves limiting short-circuit dissipation during address decoding and in the control logic and drivers. This can be achieved by careful design of individual logic circuits.

53.3 Flash Memory

In recent years, flash memories have become one of the fastest growing segments of semiconductor memories.[13,14] Flash memories are used in a broad range of applications, such as modems, networking equipment, PC BIOS, disk drives, digital cameras, and various new microcontrollers for leading-edge embedded applications. They are primarily used for permanent mass data storage. With the rapidly emerging area of portable computing and mobile telecommunications, the demand for low-power, low-voltage flash memories increases. Under such conditions, flash memories must employ low-power tunneling mechanisms for both write and erase operations, thinner tunneling dielectrics, and on-chip voltage pumps.

Low-Power Circuit Techniques for Flash Memories

In order to prolong the battery life in mobile devices, significant reductions of power consumption in all electronic components have to be achieved. One of the fundamental and most effective methods is a reduction in power supply voltage. This method has also been observed in Flash memories. Designs with a 3.3-V power supply, as opposed to the traditional 5-V power supply, have been reported.[15–20] In addition, multi-level architectures that lower the cost per bit, increase memory density, and improve energy efficiency per bit, have emerged.[17,20] Kawahara et al.[22] and Otsuka and Horowitz[23] have identified major bottlenecks when designing Flash memories for low-power, low-voltage operation and proposed suitable technologies and techniques for deep sub-micron, sub-2V power supply Flash memory design. Due to its construction, a Flash memory requires high voltage levels for program and erase operations, often exceeding 10 V (V_{pp}). The core circuitry that operates at these voltage levels cannot be as aggressively

scaled as the peripheral circuitry that operates with standard V_{dd}. Peripheral devices are designed to improve the power and performance of the chip, whereas core devices are designed to improve the read performance. Parameters such as the channel length, the oxide thickness, the threshold voltage, and the breakdown voltage must be adjusted to withstand high voltages. Technologies that allow two different transistor environments on the same substrate must be used. An example of transistor parameters in a multi-transistor process is given in Table 53.2.

TABLE 53.2 Transistor Parameters

	V_{DD} transistor		V_{PP} transistor	
	nmos	pmos	nmos	pmos
Channel length	0.6 μm	1.2 μm		
Oxide thickness	10 nm		22.3 nm	
Threshold voltage	0.4 V		0.79 V	0.97 V

Source: © 1997, IEEE. With permission.

Technologies reaching deep sub-micron levels — 0.25 μm and lower — can experience three major problems (summarized in Fig. 53.3): (1) layout of the peripheral circuits due to a scaled Flash memory cell; (2) an accurate voltage generation for the memory cells to provide the required threshold voltage and narrow deviation; and (3) deviations in dielectric film characteristics caused by large numbers of memory cells. Kawahara et al.[22] have proposed several circuit enhancements that address these problems. They proposed a sensing circuit with a relaxed layout pitch, bit-line clamped sensing multiplex, and intermittent burst data transfer for a three times feature-size pitch. They also proposed a low-power dynamic bandgap generator with voltage boosted by using triple-well bipolar transistors and voltage-doubler charge pumping, for accurate generation of 10 to 20 V that operate at V_{dd} under 2.5 V. They demonstrated these improvements on a 128-Mb experimental chip fabricated using 0.25-μm technology.

On the circuit level, three problems have been identified by Otsuka and Horowitz:[23] (1) interface between peripheral and core circuitry; (2) sense circuitry and operation margin; and (3) internal high voltage generation.

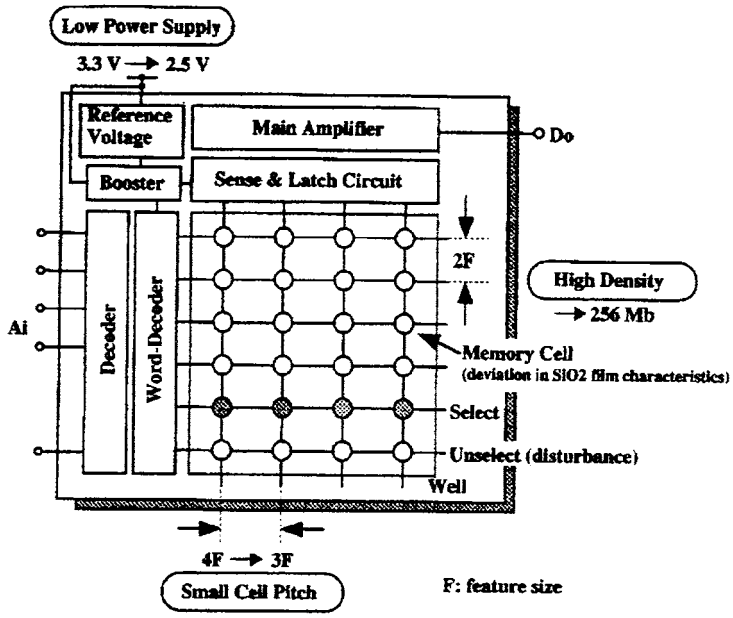


FIGURE 53.3 Quarter-micron flash memory. (© 1996, IEEE. With permission.)

During program and erase modes, the core circuits are driven with higher voltage than the peripheral circuits. This voltage is higher than V_{dd} in order to achieve good read performance. Therefore, a level-shifter circuit is necessary to interface between the peripheral and core circuitry. However, when a standard power supply (V_{dd}) is scaled to 1.5 V and lower, the threshold voltage of V_{pp} transistors will become comparable to one half of V_{dd} or less, which results in significant delay and poor operation margin of the level shifter and, consequently, degrades the read performance. A level shifter is necessary for the row decoder, column selection, and source selection circuit. Since the inputs to the level shifters switch while V_{pp} is at the read V_{pp} level, the performance of the level shifter needs to be optimized only for a read operation. In addition to a standard erase scheme, Flash memories utilizing a negative-gate erase or program scheme have been reported.[15,19] These schemes utilize a single voltage supply that results in lower power consumption. The level shifters in these Flash memories have to shift a signal from V_{dd} to V_{pp} and from Gnd to V_{bb}. Conventional level shifters suffer from delay degradation and increased power consumption when driven with low power supply voltage. There are several reasons attributed to these effects. First, at low V_{dd} (1.5 V), the threshold voltage of V_{pp} transistors is close to half the power supply voltage, which results in an insufficient gate swing to drive the pull-down transistors as shown in Fig. 53.4. This also reduces the operation margin of these shifters for the threshold voltage fluctuation of the V_{pp} transistor. Second, a rapid increase in power consumption at V_{dd} under 1.5 V is due to dc current leakage through V_{pp} to Gnd during the transient switching. At 1.5 V, 28% of the total power consumption of V_{pp} is due to dc current leakage. Two signal shifting schemes have been proposed: one for a standard flash memory and another for a negative-gate erase or program Flash memories. The first proposed design is shown in Fig. 53.5. This high-level shifter uses a bootstrapping switch to overcome the degradation due to a low input gate swing and improves the current driving capability of both pull-down drivers. It also improves the switching delay and the power consumption at 1.5 V because the

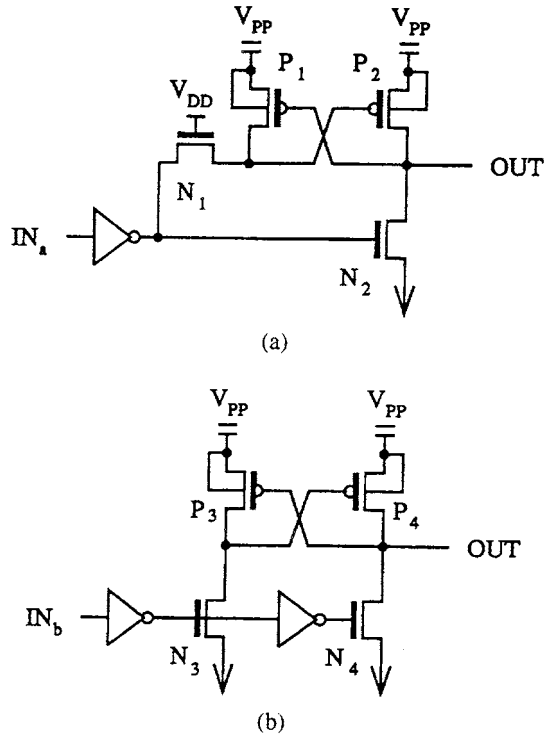


FIGURE 53.4 Conventional high-level shifter circuits with (a) feedback pMOS, (b) cross-coupled pMOS. (© 1997, IEEE. With permission.)

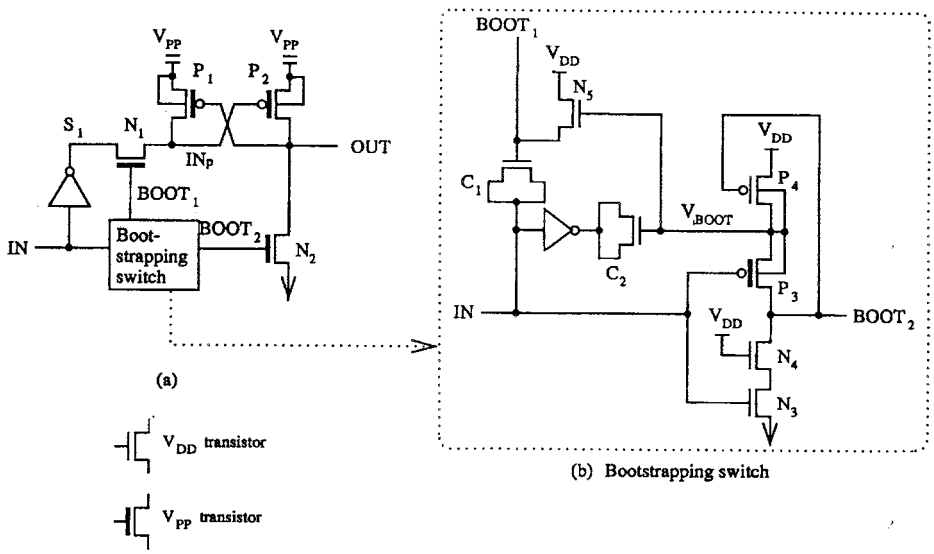


FIGURE 53.5 A high-level shifter circuit with bootstrapping switch. (© 1997, IEEE. With permission.)

bootstrapping reduces the dc current leakage during the transient switching. Consequently, the bootstrapping technique increases the operation margin. The layout overhead from the bootstrapping circuit, capacitors, and an isolated n-well is negligible compared to the total chip area because it is used only as the interface between the peripheral circuitry and the core circuitry. Figure 53.6 shows the operation of the proposed high-level shifter, and Fig. 53.7 illustrates the switching delay and the power consumption

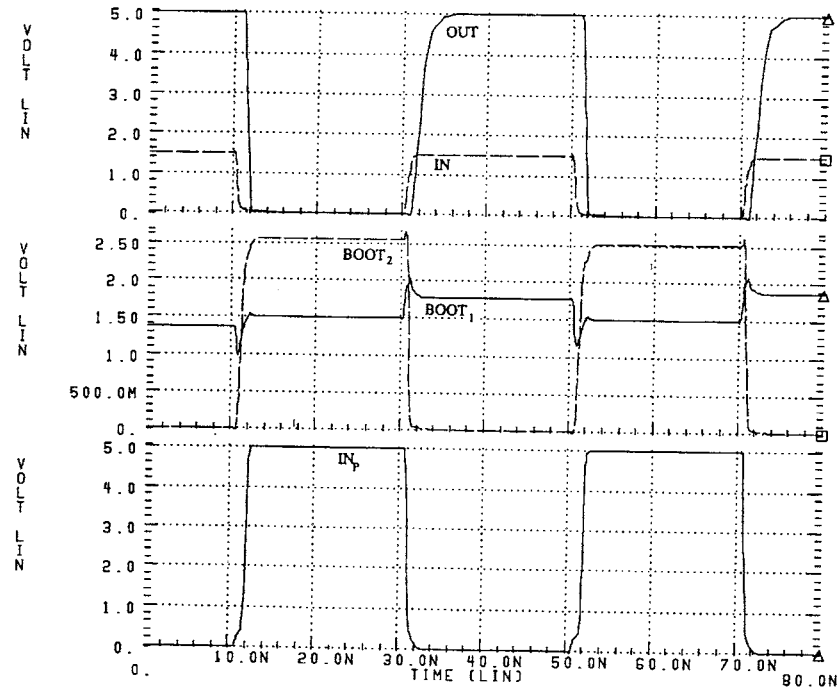


FIGURE 53.6 Operation of the proposed high-level shifter circuit. (© 1997, IEEE. With permission.)

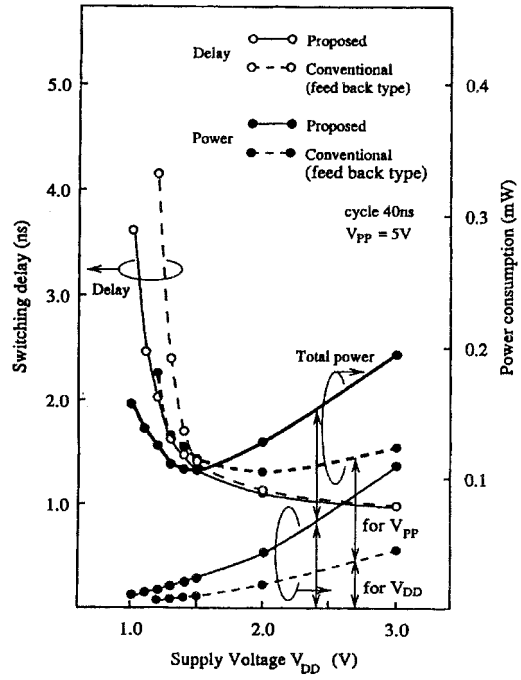


FIGURE 53.7 Comparison between proposed and conventional high-level shifters. (© 1997, IEEE. With permission.)

versus the power supply voltage of the conventional design and the proposed design. The second proposed design, shown in Fig. 53.8, is a high/low-level shifter that also utilizes a bootstrapping mechanism to improve the switching speed, reduce dc current leakage, and improve operation margin. The operation of the proposed shifter is illustrated in Fig. 53.9. At 1.5 V, the power consumption decreases by 40% compared to a conventional two-stage high/low-level shifter, as shown in Fig. 53.10. The proposed level shifter does not require an isolated n-well and therefore the circuit is suitable for a tight-pitch design and a conventional well layout.

In addition to the more efficient level-shift scheme, Otsuka and Horowitz[23] also addressed the problem of sensing under very low power supply voltages (1.5 V) and proposed a new self-bias bit-line sensing method that reduces the delay's dependence on bit-line capacitance and achieves a 19-ns reduction of the sense delay at low voltages. This enhances the power efficiency of the chip.

On a system level, Tanzawa, et al.[25] proposed an on-chip error correcting circuit (ECC) with only 2% layout overhead. By moving the ECC from off-chip to on-chip, 522-Byte temporary buffers that are required for conventional ECC and occupy a large part of ECC area, have been eliminated. As a result, the area of ECC circuit has been reduced by a factor of 25. The on-chip ECC has been optimized, which resulted in an improved power-efficiency by a factor of two.

53.4 Ferroelectric Memory (FeRAM)

Ferroelectric memory combines the advantages of a non-volatile Flash memory and the density and speed of a DRAM memory. Advances in low-voltage, low-power design toward mobile computing applications have been seen in the literature.[28,29] Hirano et al.[28] reported a new 1-transistor/1-capacitor nonvolatile ferroelectric memory architecture that operates at 2 V with 100-ns access time. They achieved these results using two new improvements: a bit-line-driven read scheme and a non-relaxation reference cell. In previous ferroelectric architectures, either a cell-plate-driven or non-cell-plate driven read scheme, as shown in Figs. 53.11(a) and (b), was used.[30,31] Although the first architecture could operate at low

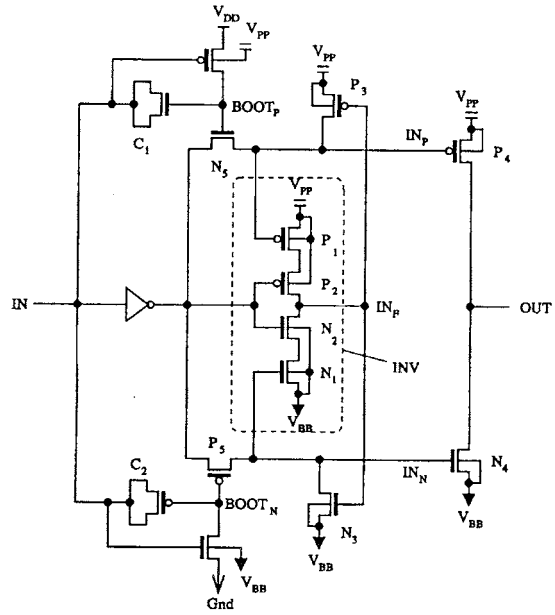


FIGURE 53.8 Proposed high/low-level shifter circuit. (© 1997, IEEE. With permission.)

FIGURE 53.9 Operation of the proposed high/low-level shifter circuit. (© 1997, IEEE. With permission.)

supply voltages, the large capacitance of the cell plate, which connects to many ferroelectric capacitors and a large parasitic capacitor, would degrade the performance of the read operation due to large transient time necessary to drive the cell plate. The second architecture suffers from two problems. The first problem is the risk of losing the data stored in the memory due to the leakage current of a capacitor. The storage

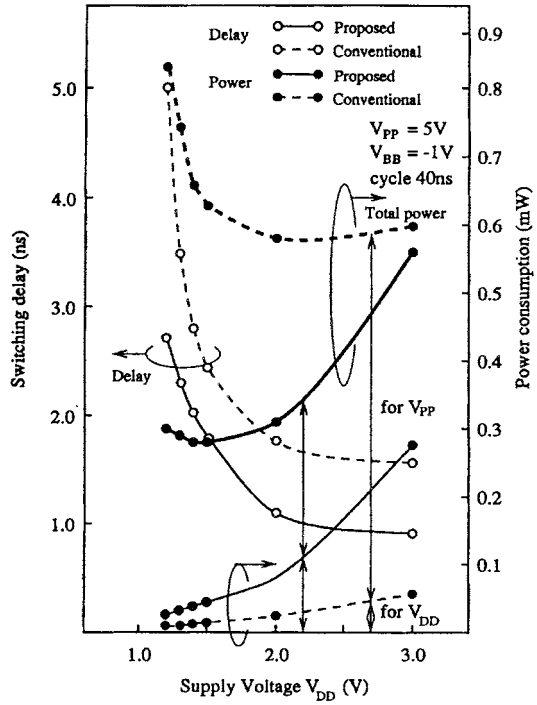


FIGURE 53.10 Comparison between proposed and conventional high/low-level shifters. (© 1997, IEEE. With permission.)

node of a memory cell is floating and the parasitic p-n junction between the storage node and the substrate leaks the current. Consequently, the storage node reaches the V_{ss} level and another node of the capacitor is kept at 1/2 V_{dd}, which causes the data destruction. Therefore, this scheme requires a refresh operation of memory cell data. The second problem arises from a low-voltage operation. Due to a voltage across the memory cell capacitor being at 1/2 V_{dd} under this scheme, the supply voltage must be twice as high as the coercive voltage of ferroelectric capacitors, which prevents the low-voltage operation. To overcome these problems, Hirano et al.[28] have developed a new bit-line-driven read scheme which is shown in Figs. 53.12 and 53.13. The bit-line-driven circuit precharges the bit-lines to supply V_{dd} voltage. The cell plate line is fixed at ground voltage in the read operation. An important characteristic of this configuration is that the bit-lines are driven, while the cell plate is not driven. Also, the precharged voltage level of the bit-lines is higher than that of the cell plate. Figure 53.14 shows the limitations of previous schemes and the new scheme. During the read operation, the first previously presented scheme[30] requires a long delay time to drive the cell plate line. However, the proposed scheme exhibits faster transient response because the bit-line capacitance is less than 1/100 of the cell plate-line capacitance. The second previously presented scheme[31] requires a data refresh operation in order to secure data retention. The read scheme proposed by Hirano et al.[28] does not require any refresh operation since the cell plate voltage is at 0 V during the stand-by mode.

The reference voltage generated by a reference cell is a critical aspect of a low-voltage operation of ferroelectric memory. The reference cell is constructed with one transistor and one ferroelectric capacitor. While a voltage is applied to the memory cell to read the data, the bit-line voltage reading from the reference cell is set to about the midpoint of "H" and "L" which are read from the main-memory-cell data. The state of the reference cell is set to "Ref" as shown at the left side of Fig. 53.15. However, a ferroelectric capacitor suffers from the relaxation effect, which decreases the polarization as shown at the right side of Fig. 53.15. As a result, each state of the main memory cells and the reference cell is shifted, and the read operation of "H" data is marginal and prohibits the scaling of power supply voltage.

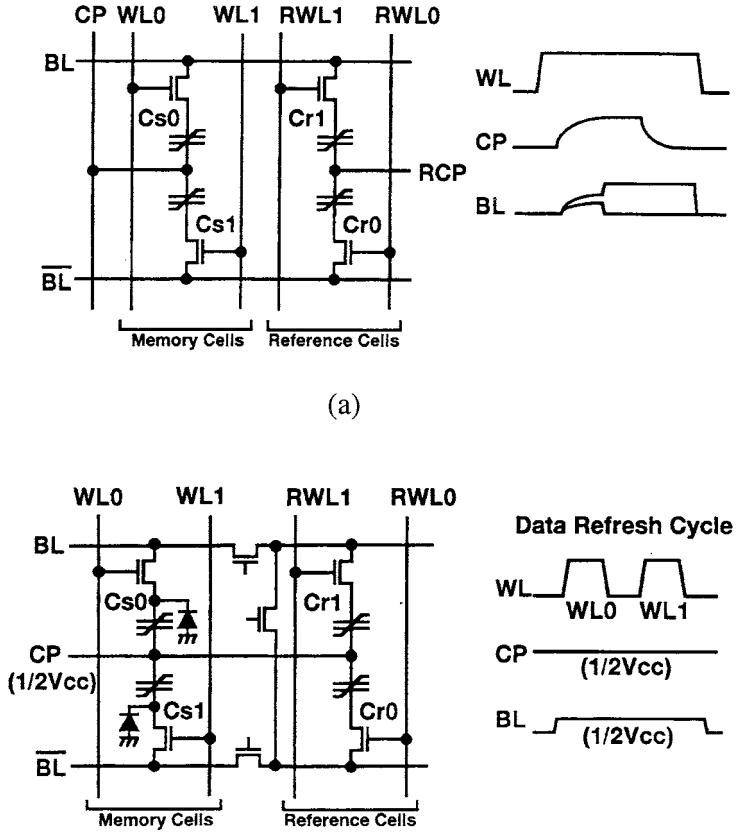


FIGURE 53.11 (a) Cell-plate-driven read scheme, and (b) non-cell-plate-driven read scheme. (© 1997, IEEE. With permission.)

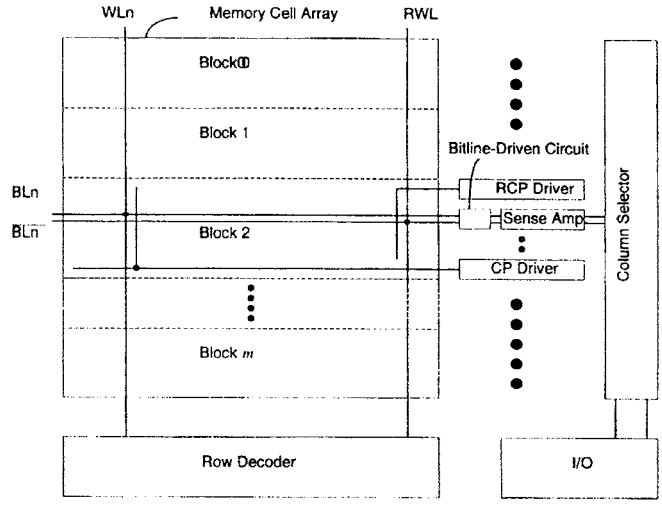


FIGURE 53.12 Memory cell array architecture. (© 1997, IEEE. With permission.)

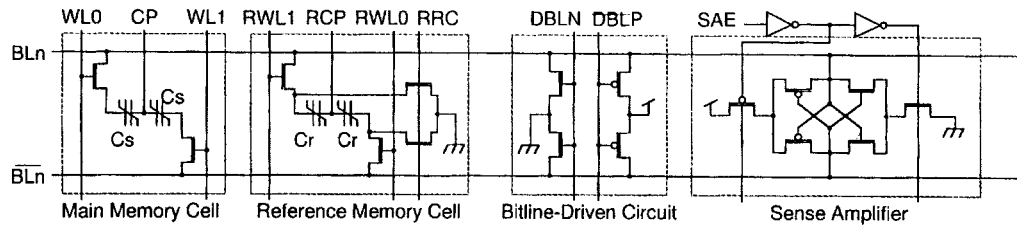


FIGURE 53.13 Memory cell and peripheral circuit with bit-line-driven read scheme. (© 1997, IEEE. With permission.)

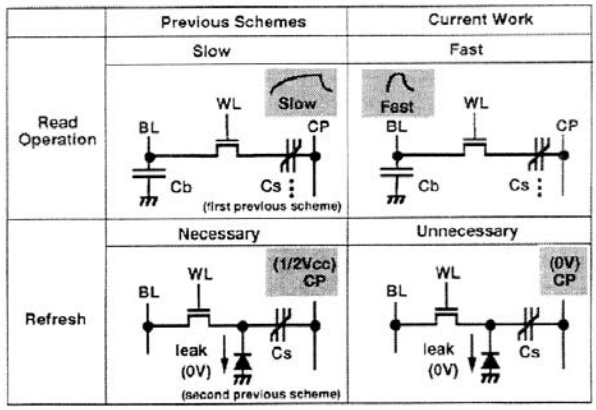


FIGURE 53.14 Limitations of previous schemes and proposed solutions. (© 1997, IEEE. With permission.)

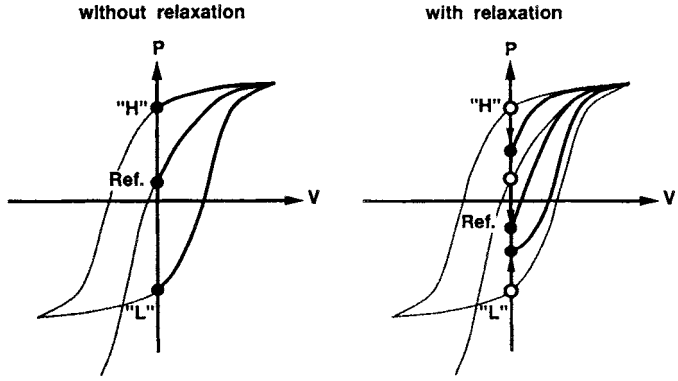


FIGURE 53.15 Reference cell proposed by Sumi et al. in Ref. 30. (© 1997, IEEE. With permission.)

Hirano et al.[28] have developed a reference cell that does not suffer from a relaxation effect, moves always along the curve from the "Ref" point, and therefore enlarges the read operation margin for "H" data. This proposed scheme enables a low-voltage operation down to 1.4 V.

Fujisawa et al.[29] addressed the problem of achieving high speed and low power operation in ferroelectric memories. Previous designs suffered from excessive power dissipation due to the need of a refresh cycle[30,31] because of the leakage current from a capacitor storage node to the substrate where the cell plates are fixed to 1/2 V_{dd}. Figure 53.16 shows a comparison of the power dissipation between ferroelectric memories

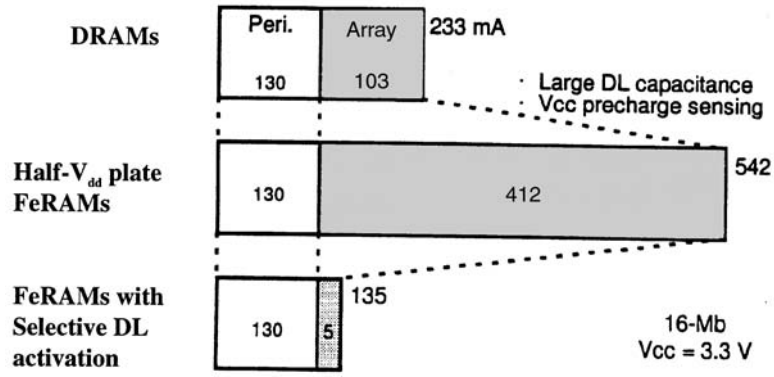


FIGURE 53.16 Comparison of the power dissipation between FeRAMs and DRAMs. (© 1997, IEEE. With permission.)

(FeRAMs) and DRAMs. It can be observed that the power consumption of peripheral circuits is identical, but the power consumption of memory array sharply increases in the 1/2 V_{dd} plate FeRAMs. These problems can be summarized as follows:

- The memory cell capacitance is large and therefore the capacitance of the data-line needs to be set larger in order to increase the signal voltage of non-volatile data.
- The non-volatile data cannot be read by the 1/2 V_{dd} subdata-line precharge technique because the cell plate is set to 1/2 V_{dd}. Therefore, the data-line is precharged to V_{dd} or Gnd.

When the memory cell density rises, the number of activated data-lines increases. This increases power dissipation of the array. A selective subdata-line activation technique as shown in Fig. 53.17, which was proposed by Hamamoto et al., overcomes this problem. However, its access time is slower compared to all-subdataline activation because the selective subdataline activation requires a preparation time. Therefore, neither of these two techniques can simultaneously achieve low-power and high-speed operation.

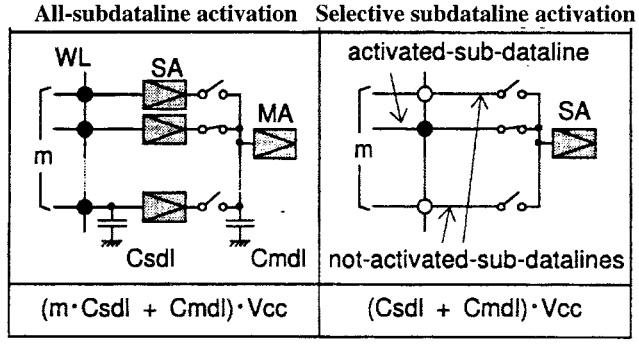


FIGURE 53.17 Low power dissipation techniques. (© 1997, IEEE. With permission.)

Fujisawa et al.[29] demonstrated a low-power high-speed FeRAM operation using an improved charge-share modified (CSM) precharge-level architecture. The new CSM architecture solves the problems of slow access speed and high power dissipation. This architecture incorporates two features that reduce the sensing period, as shown in Fig. 53.18. The first feature is the charge-sharing between the parasitic capacitance of the main data-line (MDL) and the subdata-line (SDL). During the stand-by mode, all SDLs and MDLs are precharged to 1/2 V_{dd} and V_{dd}, respectively. During the read operation, the

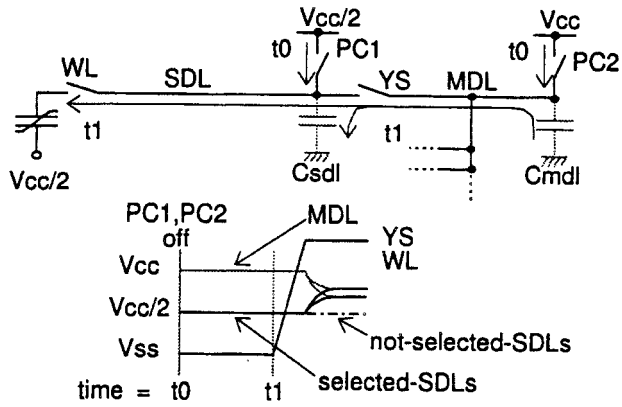


FIGURE 53.18 Principle of the CSM architecture. (© 1997, IEEE. With permission.)

precharge circuits are all cut off from the data lines (time t_0). After the y-selection signal (YS) is activated (time t_1), the charge in the parasitic capacitance of the MDL (C_{mdl}) is transferred to the selected parasitic capacitance of the SDL (C_{sdl}) and the selected SDL potential is raised by charge-sharing. As a result, the voltage is applied only to a memory cell intersecting selected word-line (WL) and YS. The second feature is a simultaneous activation of WL and YS without causing a loss of the readout voltage. During the write operation, only data of the selected memory cell is written, whereas all the other memory cells keep their non-volatile data.

Consequently, the power dissipation does not increase during this operation. The writing period is equal to the sensing period because WL and YS can also be activated simultaneously in the write cycle.

53.5 Static Random-Access Memory (SRAM)

SRAMs have experienced a very rapid development of low-power, low-voltage memory design during recent years due to an increased demand for notebooks, laptops, hand-held communication devices, and IC memory cards. Table 53.3 summarizes some of the latest experimental SRAMs for very low-voltage and low-power operation.

TABLE 53.3 Low-Power SRAMs Performance Comparison

Memory Size (Ref.)	Power Supply	CMOS Technology	Access Time	Power Dissipation
4 Kb (40)	0.9 V	0.6 μm	39 ns	18 μW @ 1 MHz
4 Kb (40)	1.6 V	0.6 μm	12 ns	64 μW @ 1 MHz
32 Kb (44)	1 V	0.35 μm	17 ns	5 μW @ 50 MHz
32 Kb (48)	1 V	0.35 μm	11.8 ns	3 μW @ 10 MHz
32 Kb (49)	1 V	0.25 μm	7.3 ns	0.9 μW @ 100 MHz
32 Kb (42)	1 V	0.25 μm	—	0.9 μW @ 100 MHz
32 Kb (55)	1 V	0.25 μm	7 ns	3.9 μW @ 100 MHz
256 Kb (53)	1.4 V	0.4 μm	60 ns	3.6 μW @ 5 MHz
1 Mb (50)	1 V	0.5 μm	74 ns	1 μW @ 10 MHz
1 Mb (52)	0.8 V	0.35 μm	10 ns	5 μW @ 100 MHz
4.5 Mb (51)	1.8 V	0.25 μm	1.8 ns	2.8 W @ 550 MHz
7.5 Mb (47)	3.3 V	0.6 μm	6 ns	8.42 μW @ 50 MHz
7.5 Mb (58)	3.3 V	0.8 μm	18 ns	4.8 μW @ 20 MHz

In this section, active and passive sources of power dissipation in SRAMs will be discussed and common low-power techniques will be analyzed.

Low-Power SRAMs

Sources of SRAM Power

There are different sources of active and stand-by (data retention) power present in SRAMs. The active power is the sum of the power consumed by the following components:

- Decoders
- Memory array
- Sense amplifiers
- Periphery (I/O circuitry, write circuitry, etc.) circuits

The total active power of an SRAM with $m \times n$ array of cells can be summarized by the expression[9,33,34]:

$$P_{active} = (mi_{active} + m(n-1)i_{leak} + (n+m)fC_{DE}V_{INT} + mi_{DC}\Delta tf + C_{PT}V_{INT}f + I_{DCP})V_{dd} \quad (53.1)$$

where i_{active} is the effective current of selected cells, i_{leak} is the effective data retention current of the unselected memory cells, C_{DE} is the output node capacitance of each decoder, V_{INT} is the internal power supply voltage, i_{DC} is the dc current consumed during the read operation, Δt is the activation time of the dc current consuming parts (i.e., sense amplifiers), f is the operating frequency, C_{PT} is the total capacitance of the CMOS logic and the driving circuits in the periphery, and I_{DCP} is the total static (dc) or quasi-static current of the periphery. Major sources of I_{DCP} are column circuitry and differential amplifiers on the I/O lines.

The stand-by power of an SRAM has a major source represented by $i_{leak}mn$ because the static current from other sources is negligibly small (sense amplifiers are disabled during this mode). Therefore, the total stand-by power can be expressed as:

$$P_{standby} = mni_{leak} \times V_{dd} \quad (53.2)$$

Techniques for Low-Power Operation

In order to significantly reduce the power consumption in SRAMs, all contributors to the total power must be targeted. The most efficient techniques used in recent memories are:

- Capacitance reduction of word-lines and the number of cells connected to them, data-lines, I/O lines, and decoders
- DC current reduction using new pulse operation techniques for word-lines, periphery, circuits, and sense amplifiers
- AC current reduction using new decoding techniques (i.e., multi-stage static CMOS decoding)
- Operating voltage reduction
- Leakage current reduction (in active and stand-by mode) utilizing multiple threshold voltage (MT-CMOS) or variable threshold voltage technologies (VT-CMOS)

Capacitance Reduction

The largest capacitive elements in a memory are word-lines, bit-lines, and data-lines, each with a number of cells connected to them. Therefore, reducing the size of these lines can have a significant impact on power consumption reduction. A common technique often used in large memories is called Divided Word Line (DWL), which adopts a two-stage hierarchical row decoder structure as shown in Fig. 53.19.[34] The number of sub-word-lines connected to one main word-line in the data-line direction is generally four, substituting the area of a main row decoder with the area of a local row decoder. DWL features two-step decoding for selecting one word-line, greatly reducing the capacitance of the address lines to a row decoder and the word-line RC delay.

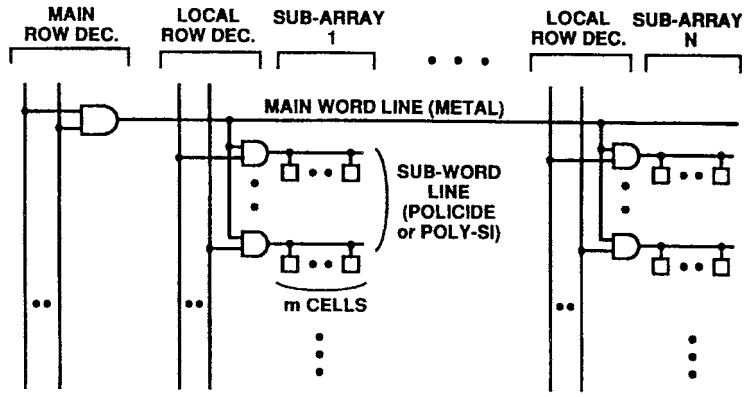


FIGURE 53.19 Divided word-line structure (DWL). (© 1995, IEEE. With permission.)

A single bit-line cross-point cell activation (SCPA) architecture reduces the power further by improving the DWL technique.[36] The architecture enables the smallest column current possible without increasing the block division of the cell array, thus reducing the decoder area and the memory core area. The cell architecture is shown in Fig. 53.20. The Y-address controls the access transistors and the X-address. Since only one memory cell at the cross-point of X- and Y- is activated, a column current is drawn only by the accessed cell. As a result, the column current is minimized. In addition, SCPA allows the number of blocks to be reduced because the column current is independent of the number of block divisions in the SCPA. The disadvantage of this configuration is that during the write "high" cycle, both X- and Y-lines have to be boosted using a word-line boost circuit.

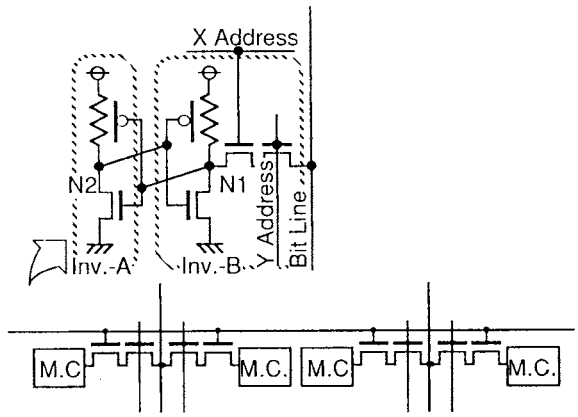


FIGURE 53.20 Memory cell used for SCPA architecture. (© 1994, IEEE. With permission.)

Caravella proposed a similar subdivision technique to DWL, which he demonstrated on 64 × 64 bit cell array.[39,40] If C_j is a parasitic capacitance associated with a single bit cell load on a bit-line (junction and metal) and if C_{ch} is a parasitic capacitance associated with a single bit cell on the word-line (gate, fringe, and metal), then the total bit-line capacitance is $64 \times C_j$ and the total word capacitance is $64 \times C_{ch}$. If the array is divided into four isolated sub-arrays of 32×32 bit cells, the total bit-line and word-line capacitances would be halved, as shown in Fig. 53.21. The total capacitance per read/write that would need to be discharged or charged is given by $1024 \times C_j + 32 \times C_{ch}$ for the sub-array architecture as opposed to $4096 \times C_j + 64 \times C_{ch}$ for the 64×64 array. This technique carries a penalty due to additional decode and control logic and routing.

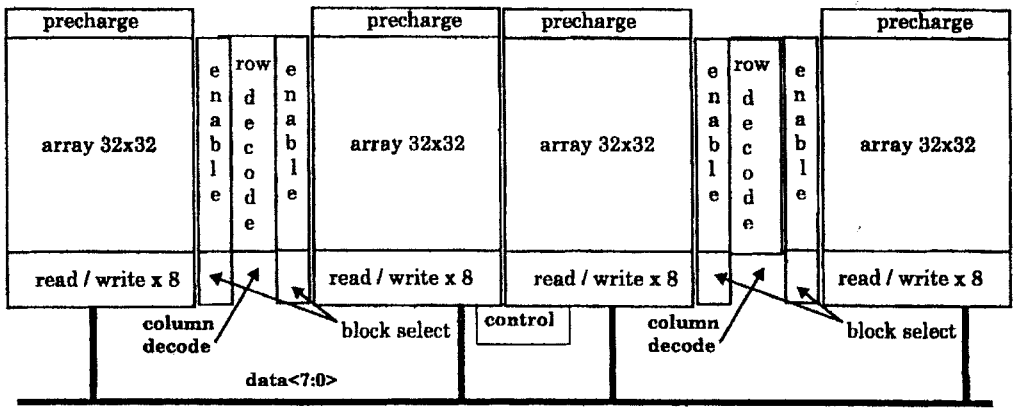


FIGURE 53.21 Memory architecture. (© 1997, IEEE. With permission.)

Pulse Operation Techniques

Pulsing the word-lines, equalization, and sense lines can shorten the active duty cycle and thus reduce the power dissipation. In order to generate different pulse signals, an on-chip address transition detection (ATD) pulse generator is used.[34] This circuit, shown in Fig. 53.22, is a key element for the active power reduction in memories.

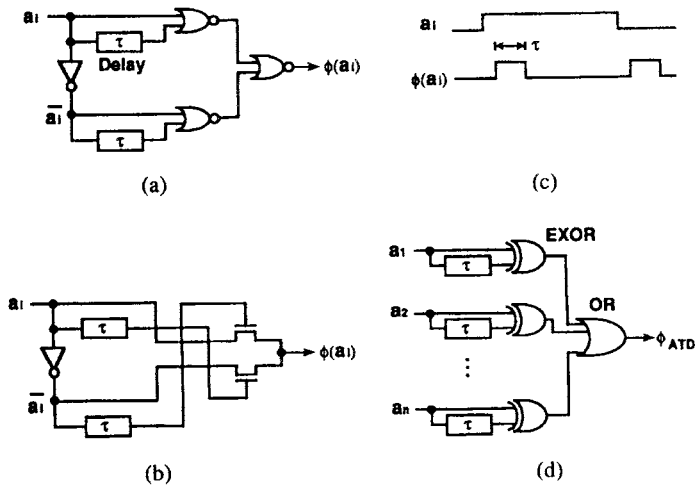


FIGURE 53.22 Address transition detection circuits: (a) and (b) ATD pulse generators; (c) ATD pulse wave-forms; and (d) a summation circuit of all ATD pulses generated from all address transitions. (© 1995, IEEE. With permission.)

An ATD generator consists of delay circuits (i.e., inverter chains) and an XOR circuit. The ATD circuit generates a $\phi(a_i)$ pulse every time it detects an "L"-to-"H" or "H"-to-"L" transition on the input address signal a_i. Then, all ATD-generated pulses from all address transitions are summed through an OR gate to a single pulse ϕ_{ATD}. This final pulse is usually stretched out with a delay circuit to generate different pulses needed in the SRAM and used to reduce power or speed up a signal propagation.

Pulsed operation techniques are also used to reduce power consumption by reducing the signal swing on high-capacitance predecode lines, write-bus-lines, and bit-lines without sacrificing the performance.[37,42,49] These techniques target the power that is consumed during write and decode operations.

Most of the power savings comes from operating the bit-lines from $V_{dd}/2$ rather than V_{dd}. This approach is based on the new half-swing pulse-mode gate family. Figure 53.23 shows a half-swing pulse-mode AND gate. The principle of the operation is in a merger of a voltage-level converter with a logical AND. A positive half-swing (transitions from a rest state $V_{dd}/2$ to V_{dd} and back to $V_{dd}/2$) and a negative half-swing (transitions from a rest state $V_{dd}/2$ to Gnd and back to $V_{dd}/2$) combined with the receiver-gate logic style result in a full gate overdrive with negligible effects of the low-swing inputs on the performance of the receiver. This structure is combined with a self-resetting circuitry and a PMOS leaker to improve the noise margin and the speed of the output reset transition, as shown in Figure 53.24.

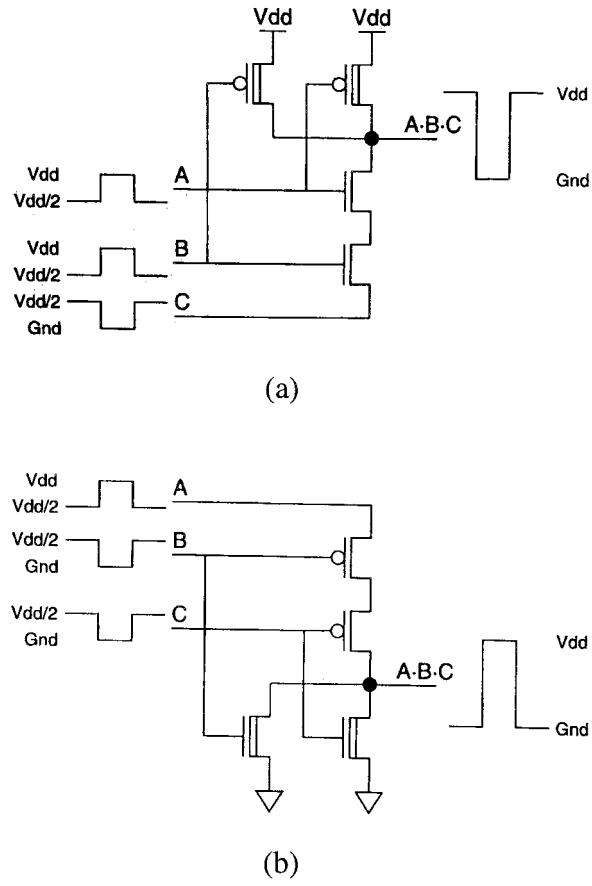


FIGURE 53.23 Half-swing pulse-mode AND gate: (a) NMOS-style, and (b) PMOS-style (© 1998, IEEE. With permission.)

Both negative and positive half-swing pulses can reduce the power consumption further by using a charge recycling. The charge used to produce the assert transition of a positive pulse can also be used to produce the reset transition of a negative pulse. If the capacitances of positive and negative pulses match, then no current would be drawn from the $V_{dd}/2$ power supply ($V_{dd}/2$ voltage is generated by an on-chip voltage converter). Combining the half-swing pulse-mode logic with the charge recycling techniques, 75% of the power on high-capacitance lines can be saved.[49]

AC Current Reduction

One of the circuit techniques that reduces AC current in memories is multi-stage decoding. It is common that fast static CMOS decoders are based on OR/NOR and AND/NAND architectures. Figure 53.25 shows one example of a row decoder for a three-bit address. The input buffers drive the interconnect capacitance

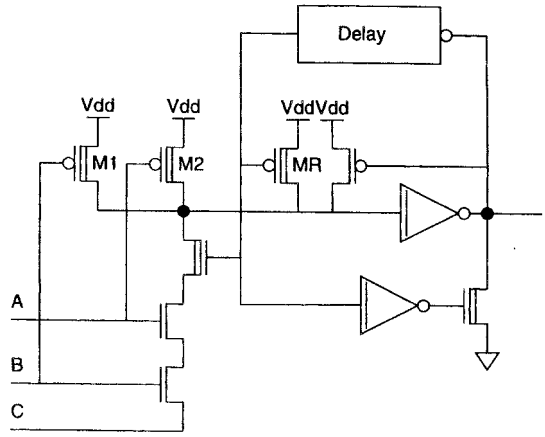


FIGURE 53.24 Self-resetting half-swing pulse-mode gate with a PMOS leaker. (© 1998, IEEE. With permission.)

of the address line and also the input capacitance of the NAND gates. By using a two-stage decode architecture, the number of transistors, fanin and the loading on the address input buffers are reduced, as shown in Fig. 53.26. As a result, both speed and power are optimized. The signal ϕ_x, generated by the ATD pulse generator, enables the decoder and secures pulse-activated word-line

Operating Voltage Reduction and Low-Power Sensing Techniques

Operating voltage reduction is the most powerful method for power conservation. Power supply voltage reductions down to 1 V[35,42,44,46,48–50,55] and below[40,52,53] have been reported. This aggressively scaled environment requires news skills in new fast-speed and low-power sensing schemes. .A charge-transfer sense

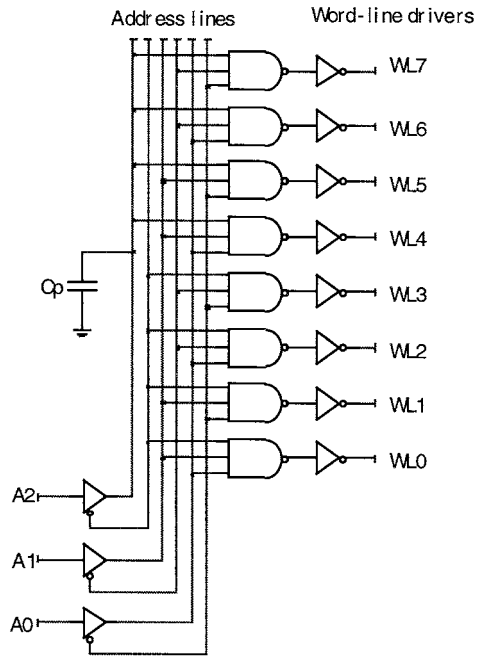


FIGURE 53.25 A row decoder for a three-bit address.

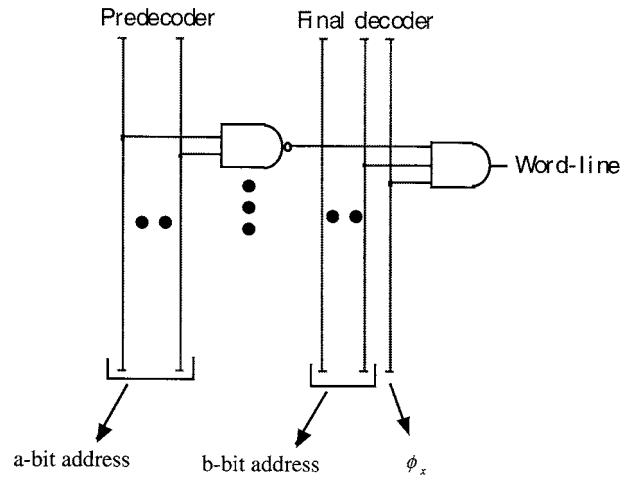


FIGURE 53.26 A two-stage decoder architecture.

amplifying scheme combined with a dual-V_t CMOS circuit achieves a fast sensing speed and a very low power dissipation at 1 V power supply.[44,55] At this voltage level, the "roll-off" on threshold voltage versus gate length, the shortest gate length causes the V_{th} mismatch between the pair of MOSFETs in the differential sense amplifier. Figure 53.27 shows the schematic of a charge-transfer sense amplifier. The charge-transfer (CT) transistors perform the sensing and act as a cross-couple latch. For the read operation, the supply voltage of the sense amplifiers changes from 1 V to 1.5 V by p-MOSFETs. The threshold voltage mismatch between two CTs is completely compensated because CTs themselves form a latch. Consequently, the bit-line precharge time, before the word-line pulse, can be omitted due to improved sensitivity. The cycle time is shortened because all clock timing signals in read operation are completed within the width of the word-line pulse.

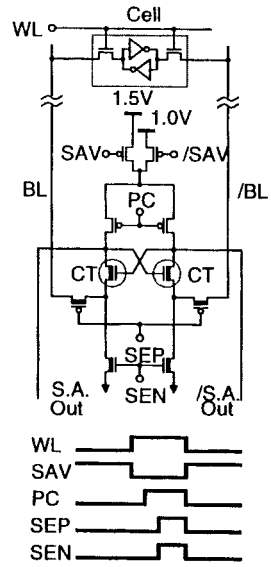


FIGURE 53.27 Charge-transfer sense amplifier. (© 1998 IEEE. With permission.)

Another method is the step-down, boosted-word-line scheme combined with current-sensing amplification. Boosting a selected word-line voltage shortens the bit-line delay before the stored data are sensed. The power consumption is reduced during the word-line selection using a stepping down technique of selected world-line potential.[46] However, this causes an increased power dissipation and a large transition time due to enhanced bit-line swing. The operation of this scheme is shown in Figure 53.28. After the selected word-line is boosted, it is restricted to only a short period at the beginning of the memory-cell access. This enables an early sensing operation. When the bit-lines are sensed, the word-line potential is reduced to the supply voltage level to suppress the power dissipation. Reduced signals on the bit-lines are sufficient to complete the read cycle with the current sensing. A fast read operation is obtained with little power penalty. The step-down boosting method is also used for write operation. The circuit diagram of this method is shown in Fig. 53.29. Word drivers are connected to the boosted-pulse generator via switches S_1 and S_2. These switches separate the parasitic capacitance C_B from the boosted line, thus reducing its capacitance. NMOS transistors are more suitable for implementing these switches because they do not require a level-shift circuit. Transistor Q1 is used for the stepping-down function. During the boost, the gate electrode is set to V_{dd}. If the word-line charge exceeds $V_{dd} + |V_{tp}|$, then Q1 ($|V_{tp}|$ is a threshold voltage of Q1) turns on and the word-line is clamped. After the stepping-down process, ϕ_{SEL} switches low and Q1 guarantees V_{dd} voltage on the word-line.

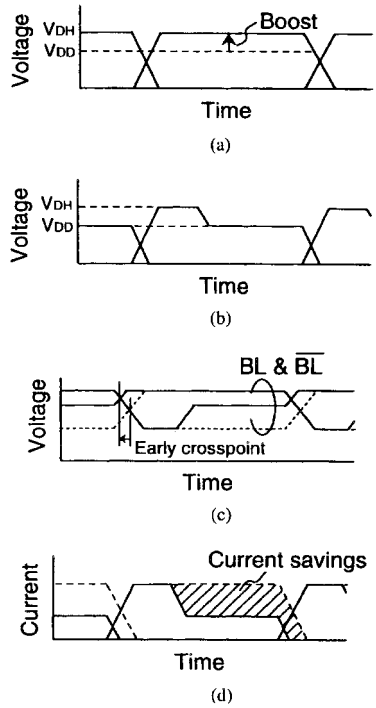


FIGURE 53.28 Step-down boosted word-line scheme: (a) conventional, (b) step-down boosted word-line, (c) bit-line transition, and (d) current consumption of a selected memory cell. (© 1998 IEEE. With permission.)

An efficient method for reducing the AC power of bit-lines and data-lines is to use the current-mode read and write operations based on new current-based circuit techniques.[47,56,57] Wang et al. proposed a new SRAM cell that supports current-mode operations with very small voltage swings on bit-lines and datalines. A fully current-mode technique consumes only 30% of the power consumed by a previous current-read-only design. Very small voltage swings on bit-lines and data-lines lead to a significant reduction of ac power. The new memory cell has seven transistors, as shown in Fig. 53.30. The additional

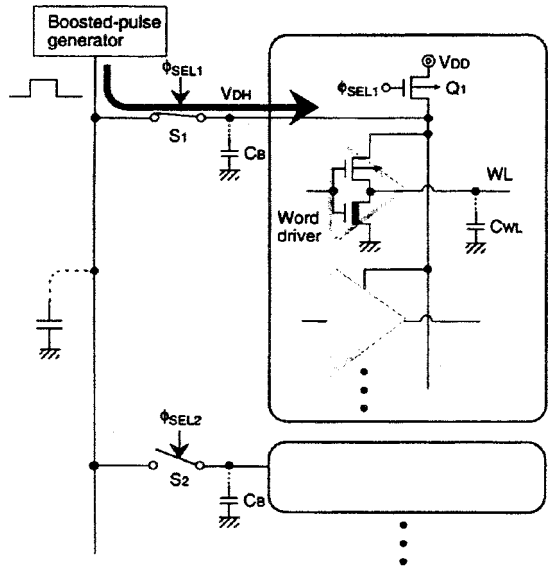


FIGURE 53.29 Circuit schematic of step-down boosted word-line method. (© 1998 IEEE. With permission.)

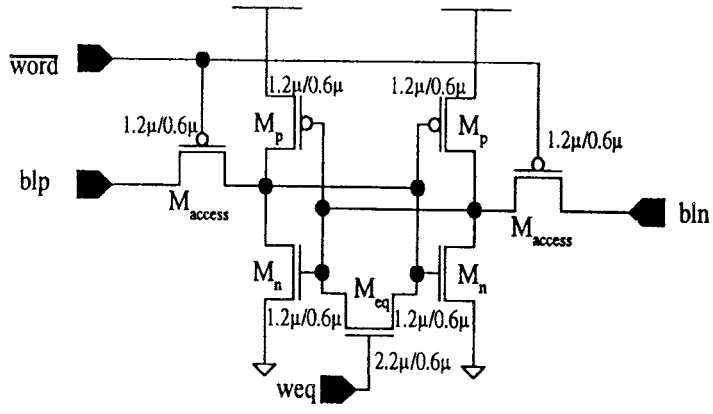


FIGURE 53.30 New 7-transistor SRAM memory cell. (© 1998, IEEE. With permission.)

transistor M_{eq} clears the content of the memory cell prior to the write operation. It performs the cell equalization. This transistor is turned off during the read operation so it does not disrupt the normal operation. An n-type current conveyor is inserted between the data input cell and the memory cell in order to perform a current-mode write operation, which is a complementary way to read. The equalization transistor is sized to be as large as possible to improve fast equalization speed, but not to increase the cell size. After suitable sizing, the new 7-transistor cell is 4.3% smaller than its 6-transistor counterpart, as illustrated in Fig. 53.31.

Another new current-mode sense amplifier for 1.5-V power supply was proposed by Wang and Lee.[57] The new circuit overcomes the problems of a conventional sense amplifier with pattern dependency by implementing a modified current conveyor. A pattern-dependency problem limits the scaling of the operating voltage. Also, the circuit does not consume any DC power because it is constructed as a complementary device. As a result, the power consumption is reduced by 61 to 94% compared with a

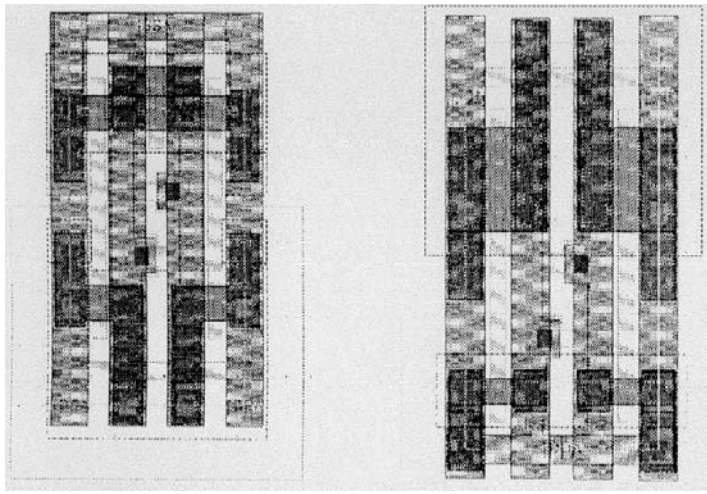


FIGURE 53.31 SRAM cell layout: (a) 6T cell, and (b) new 7T cell. (© 1998, IEEE. With permission.)

conventional design. The circuit structure of the modified current conveyor is similar to a conventional current conveyor design. However, an extra PMOS transistor Mp7, as seen in Fig. 53.32, is used. The transistor is controlled by RX signal (a complement of CS). After every read cycle, transistor Mp7 is turned on and equalizes nodes RXP and RXN, which eliminates any residual differential voltage between these two nodes (limitation in conventional designs).

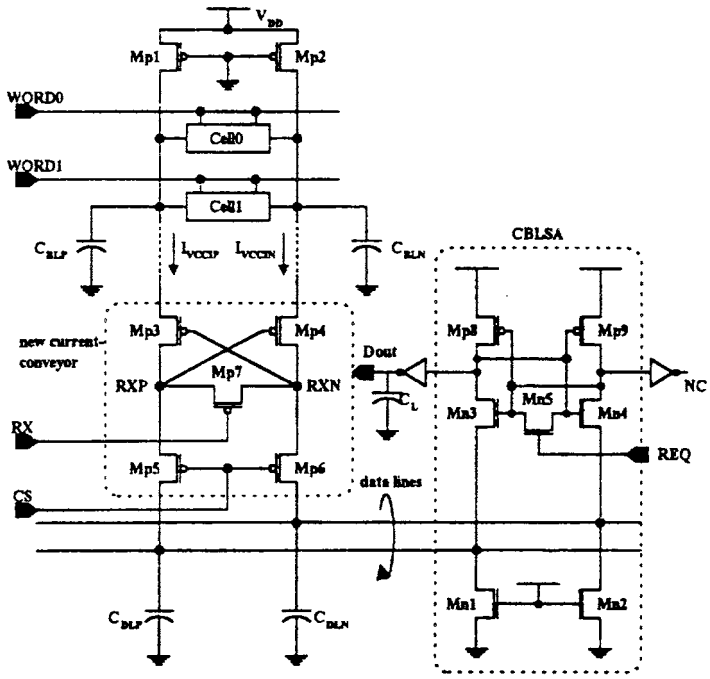


FIGURE 53.32 SRAM read circuitry with the new current-mode sense amplifier. (© 1998, IEEE. With permission.)

Leakage Current Reduction

In order to effectively reduce the dynamic power consumption, the threshold voltage is reduced along with the operating voltage. However, low threshold voltages increase the leakage current during both active and stand-by modes. The fundamental method for a leakage current reduction is a dual-V_{th} or a variable-V_{th} circuit technique. An example of one such technique is shown in Fig. 53.33.[44,55] Here, high V_{th} MOS transistors are utilized to reduce the leakage current during stand-by mode. As the supply voltage for the word decoder (g) is lowered to 1 V, all transistors forming the decoder are low V_{th} to retain high performance. The leakage currents during the stand-by mode are substantially reduced by a cut-off switch (SWP, SWN). SWN consists of a high V_{th} transistor, and SWP consists of a low V_{th} transistor. Both switches are controlled by a 1.5-V signal. Hence, the SWN gains considerable conductivity. SWP can be quickly cut off because of the reverse-biasing. The operating voltage of the local decoder (w) is boosted to 1.5 V. The high operating voltage gives sufficient drivability even to high V_{th} transistors.

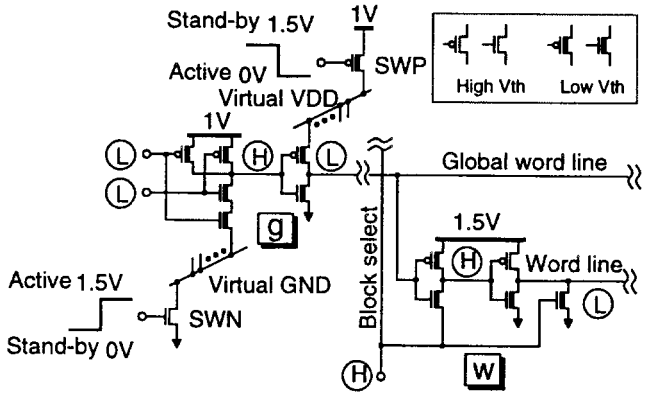


FIGURE 53.33 Dual V_{th} CMOS circuit scheme. (© 1998, IEEE. With permission.)

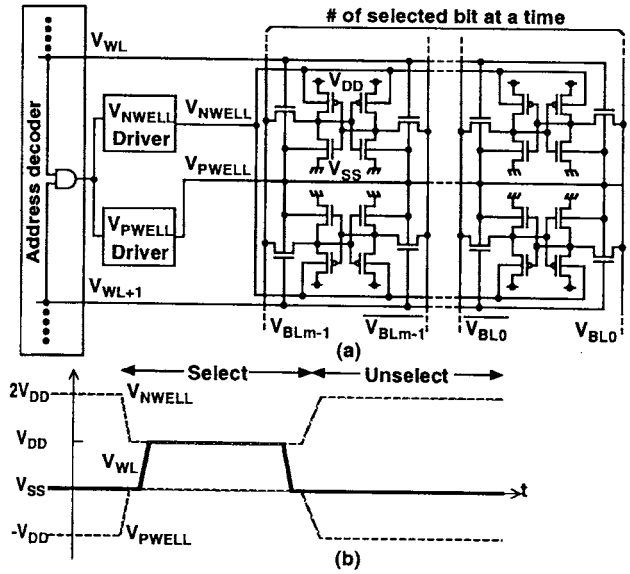


FIGURE 53.34 Dynamic leakage cut-off scheme: (a) circuit schematic and (b) its operation. (© 1998, IEEE. With permission.)

This technique belongs to schemes that use dynamic boosting of the power supply voltage and word-lines. However, in these schemes, the gate voltage of MOSFETs is often raised to more than 1.4 V, although the operating voltage is 0.8 V. This creates reliability problems.

Kawaguchi et al.[54] introduced a new technique — a dynamic leakage cut-off (DLC) scheme. Operation waveforms are shown in Fig. 53.34. A dynamic change of n-well and p-well bias voltages to V_{dd} and V_{ss}, respectively, for selected memory cells is the key feature of this architecture. At the same time, the non-selected memory cells are biased with ~$2V_{dd}$ for V_{NWELL}, and ~–V_{dd} for V_{PWELL}. After this, the V_{th} of the selected cells becomes low, which aids in high drive. Thus, a fast operation is executed. On the other hand, the V_{th} of the unselected memory cells is high enough to achieve low subthreshold current consumption. This technique is similar to the Variable Threshold CMOS (VT CMOS) technique; however, the difference is in the synchronization signal of the well bias. While in VT CMOS, the well bias is synchronized with a stand-by signal, DLC technique is synchronized with the word-line signal.

Nii et al.[48] improved the MT-CMOS technique further and proposed the Auto-Backgate Controlled (ABC) MT-CMOS method. The ABC MT-CMOS reduces significantly the leakage current during the "sleep" mode. The circuit diagram of this method is shown in Fig. 53.35. Transistors Q1-Q4 are high-threshold devices that act as switches to cut off the leakage current. The internal circuitry is designed with low-V_t devices. During the active mode, signal *SL* is pulled low and $\overline{SL}$ is pulled high. Q1, Q2, and Q3 turn on, Q4 turns off, and virtual power supply VVDD and the substrate bias BP become 1 V. During the sleep mode, signal *SL* is pulled high, $\overline{SL}$ is pulled low, and Q1, Q2, and Q3 turn off, whereas Q4 turns on and BP becomes 3.3 V. The leakage current that flows from Vdd2 to ground through D1, and D2 determines voltages Vd1, Vd2, and Vm. Vd1 is a bias between the source and the substrate of the PMOS transistors, Vd2 is a bias of the NMOS transistors, and Vm is a voltage between the virtual power line VVDD and the virtual ground VGND. The leakage current is reduced to 20 pA/cell.

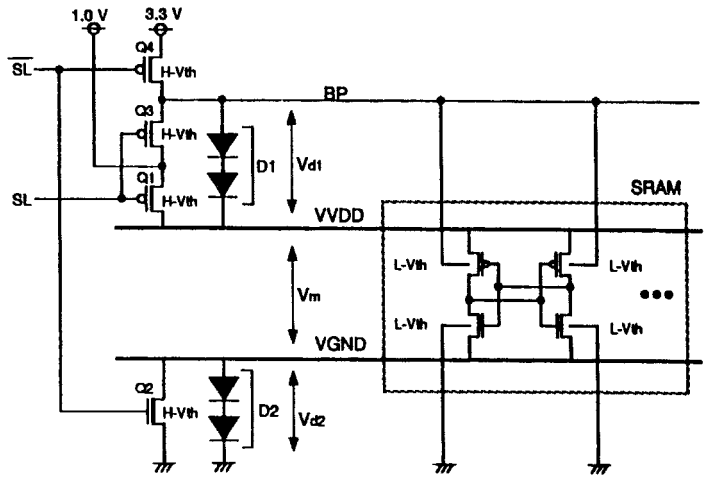


FIGURE 53.35 A schematic diagram of ABC-MT-CMOS circuit. (© 1998, IEEE. With permission.)

53.6 Dynamic Random-Access Memory (DRAM)

Similar to all previous types of memories, DRAM has undergone a remarkable development toward higher access speed, higher density, and reduced power.[34,61–64] As for reducing power, a variety of techniques targeting various sources of power in DRAMs have been reported. In this section, sources of power consumption will be discussed and then several methods for the reduction of active and data retention power in DRAMs will be described.

Low-Power DRAM Circuits

Sources of DRAM Power

The total power dissipated in a DRAM has two components: the active power and the data retention power. Major contributors to the active power are: decoders (row and column), memory array, sense amplifier, DC current dissipation of other circuits (a refresh circuitry, a substrate back-bias generator, a boosted level generator, a voltage reference circuit, a half-Vdd generator and a voltage down converter), and remaining periphery circuits (main sense amplifier, I/O buffers, write circuitry, etc). The total active power can be described as:

$$P_{active} = [(mC_D\Delta V_D + C_{PT}V_{INT})f + I_{DCP}]V_{dd} \quad (53.3)$$

where C_D is the data-line capacitance, ΔV_D is the data-line voltage swing (0.5 V_{dd}), m is the number of cells connected to the activated data-line, C_{PT} is the capacitance of the periphery circuits, V_{INT} is the internal supply voltage, and I_{DCP} is the static current.

The total data retention power is given as:

$$P_{retention} = = [(mC_D\Delta V_D + C_{PT}V_{INT})(n/t_{REF}) + I_{DCP}]V_{dd} \quad (53.4)$$

where n is the number of words that require refresh and $1/t_{REF}$ is the frequency of the refresh operation (current).

Techniques for Low-Power Operation

To reduce power consumption during both modes of DRAM operation, many circuit techniques can be applied, including:

- Capacitance reduction, especially of data-lines, word-lines, and shared I/O, using partial activation of multi-divided data-lines and partial activation of multi-divided word-lines
- Lowering of external and internal voltages
- DC power reduction of peripheral circuits during the active mode by using static CMOS decoders, pulse techniques, and ATD circuit, similar to SRAMs
- Refresh power reduction (in addition to capacitance reduction and operating voltages reduction, which are also applicable to the refresh mode, decreasing the frequency of refresh cycle or decreasing the number of words n that require refresh affects the total refresh power).
- AC and DC power reduction of circuits such as a voltage down converter (VDC), a half-voltage generator (HVG), a boosted voltage generator (BVG), and a back-bias generator (BBG)

Capacitance Reduction

Charging and discharging large data- and word-lines contribute to large amounts of dissipated power in a DRAM.[34,64] Therefore, minimizing the capacitance of these lines can accomplish significant gains in power savings. There are two fundamental methods used to reduce capacitance in DRAMs: partial activation of multi-divided data-line and partial activation of multi-divided word-line. The concept of both techniques is shown in Figs. 53.36 and 53.37.

The foundation of partial activation of multi-divided data-line (Fig. 53.36) is in reducing the number of memory cells connected to an active data-line, thus reducing its capacitance C_D. The data-lines are divided into small sections with shared I/O circuitry and a sense amplifier. By sharing these resources, further reduction of C_D is achieved. The partial activation is performed by activating only one sense amplifier along the data-line. The principle of the partial activation of multi-divided word-line (see Fig. 53.37) is very similar to that of SRAMs. A single word-line is divided into several ones by the subword-line drivers (SWL). Every SWL has to be selected by the main word-line (MWL) and the row select line signal (RX). Thus, only a partial word-line will be activated.

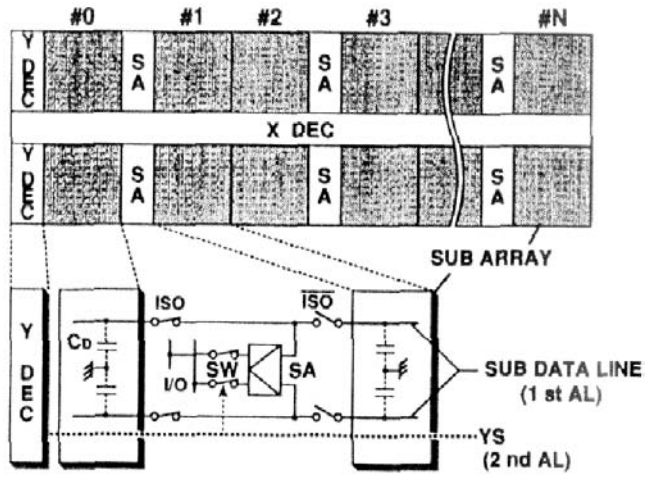


FIGURE 53.36 Multi-divided data-line architecture. (© 1995, IEEE. With permission.)

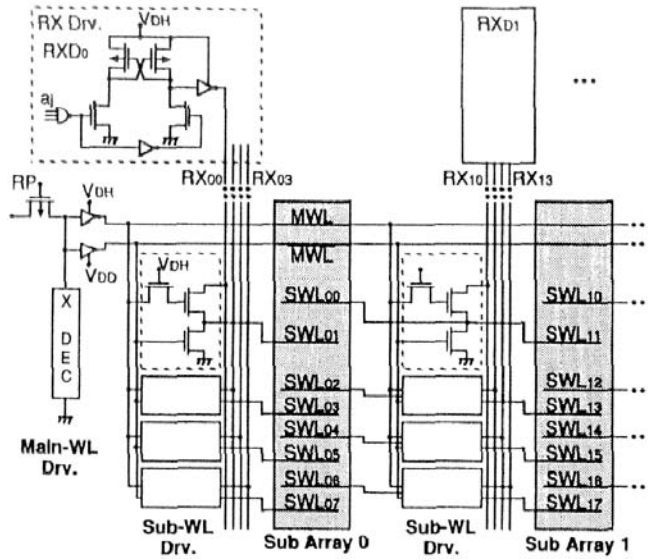


FIGURE 53.37 Hierarchical word-line architecture. (© 1995, IEEE. With permission.)

A similar method, called a hierarchical decoding scheme with dynamic CMOS series logic predecoder, has been proposed for synchronous DRAMs (SDRAMs).[65,66] This method targets the power losses in the peripheral region of the memory. This power is consumed due to the large capacitive loading of the data-lines, the address-lines, and the predecoder lines. The scheme is shown in Fig. 53.38. The hierarchical decoder uses predecoded signal lines where the redundancy circuits are connected directly from the global lines. This results in a reduced capacitive loading and a 50% reduction in the number of bus lines (column and row decoders). This circuit technique can be combined with a design of a small-swing single-address driver with a dynamic predecoder.[65,66] This scheme allows a reduction of 23 address lines. The schematic diagram of this circuit is shown in Fig. 53.39. Also, the scheme achieves a small swing in address lines with a short pulse-driven pull-up transistor with a level holder of half-V_{INT} power. The pull-up for the reduced swing bus line is achieved with a short pulse and its width brings the bus signal close to the small swing voltage (V_{INTL}).

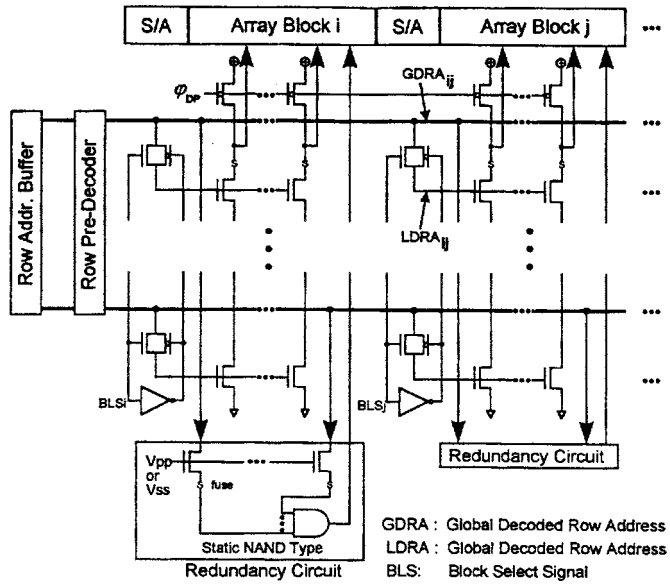


FIGURE 53.38 A decoding scheme with the hierarchical predecoded row signal and global signals shared with redundancy. (© 1998, IEEE. With permission.)

DC Current Reduction

During the active mode, most of the DC power in DRAMs and SDRAMs is consumed by the periphery circuits and I/O lines. The decoding and pulsed operation techniques based on an ATD circuit and similar to those for SRAMs can be applied. In order to minimize power consumption of I/O lines in SDRAMs, two circuit techniques have been proposed.[68] As for the first technique, the extended small-swing read operation ($\Delta V_{I/O} = \pm 200$ mV), the small-swing data paths (local I/O and global I/O) are extended up to the output buffer stages through main I/O (MIO) lines (see Fig. 53.39). Shared current sense amplifiers (I/O sense amplifiers) also reduce power consumption. In the second technique, the single I/O line driving

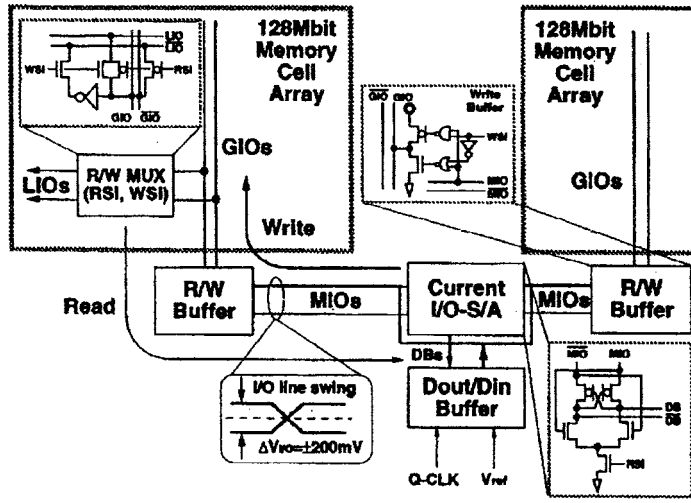


FIGURE 53.39 Block diagram of I/O datapath. (© 1996, IEEE. With permission.)

write operation halves the operating current of long global I/O lines and main I/O lines. By combining these two methods, as much as 30% of total peripheral power can be saved.

Another power-saving method for low-power SDRAMs is based on a new cell-operating concept.[69] When the operating voltage of the memory array is scaled to 1.8 V for 1-Gb SDRAMs, the performance significantly degrades due to the following factors. First, the sensing speed decreases due to the noticeable threshold voltage of source-floated transistors. Second, a triple-pumping circuit may be required to increase the power of boosted word-lines (relatively high V_{pp}). The concept of the proposed method is that the bit-lines are precharged to ground level (V_{ss}). The word-line reset voltage is –0.5 V (as compared with 1/2 V_{dd} in conventional schemes) so that a cell leakage current can be prevented while lowering the threshold voltage of pass transistors. This eliminates word-line boosting because the triple-boosting circuit is no longer required.

Operating Voltages Reduction

Lowering external and internal operating voltages is considered as an important technique for achieving significant savings of power. In both active and stand-by modes, voltages from different sources, such as V_{dd}, V_{INT}, or ΔV_D, as described in Eqs. 53.3 and 53.4, largely contribute to a total power consumption. Over the last decade, a trend in the reduction of the external power supply voltage V_{dd} for DRAMs has been observed, sliding from 12 V down to 3.3, 2.5, and 1.2 V.[66,67,69,76,79] An experimental circuit with V_{dd} as low as 1 V has been recently reported.[77] The lack of a universal standard external operating power supply voltage has resulted in DRAMs with an on-chip voltage-down converter (VDC) that uses widely accepted power supply voltages V_{dd}, such as 5 V or lately 3.3 V, and lowers the operating voltage for the memory core, thus gaining power savings.[33,34,73] VDC is one of the most important DRAM circuits in achieving DRAM operation at battery voltage levels. In power-limited applications, VDC must have a stand-by current less than 1 μA over a wide range of operating temperatures, process, and power supply voltage variations. Also, its output impedance has to be low. There are additional on-chip voltage generators: half-V_{dd} generator (HVG) for precharging bit-lines; back-bias generator (BBG) for subthreshold current and junction capacitance reduction, improving device isolation and latch-up immunity and circuit protection against voltage undershoots of input signals; and boosted voltage generator (BVG) for driving the word-lines.[33,34]

The HVG circuit has been used since 1-Mb DRAM generation. It is an efficient technique to reduce the voltage swing on bit-lines from a full V_{dd} swing to $1/2V_{dd}$ swing. During the sensing, one bit-line switches from $1/2V_{dd}$ to V_{dd} and the second bit-line from $1/2V_{dd}$ to ground. As a result, the peak switching current is reduced and the noise level is suppressed. Recently, a new technique that eliminates $1/2V_{dd}$ bit-line switching was proposed.[70] This new method, called "non-precharged bit-line sensing" (NPBS), provides the following three features (as seen in Fig. 53.40): (1) the precharge operation time is reduced by 78% because the bit-lines are not substantially precharged; (2) the sensing speed increases because the bit-lines that have not been precharged remain at low or high levels, increasing the V_{GS} and V_{DS}

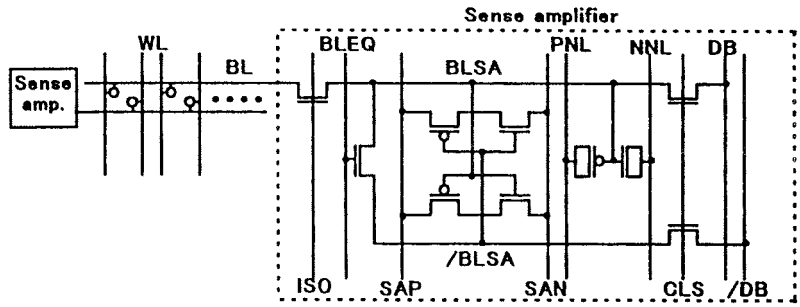


FIGURE 53.40 NPBS circuit and its operation. (© 1998, IEEE. With permission.)

voltages for the sense amplifier transistor; (3) the power dissipation is reduced when the same data occur on the bit-line. The power is reduced by about 43%.

In order to maintain or improve the speed and reliability of DRAM operations, the threshold voltage V_t has to follow the same scaling pattern as the main power supply voltage. This scenario, however, results in a rapid increase of leakage currents in the entire memory during both active and stand-by modes. Therefore, an internal back-bias generator (BBG) circuit, also known as the charge-pump, is needed to improve low-voltage, low-power operation by reducing the subthreshold currents. Figure 53.41 shows the schematic of a pumping circuit that avoids the V_t losses.[71] When the clock (clk) is at logic low, the node voltage of the node A reaches $|V_{tp}| - V_{dd}$. The PMOS transistor p1 clamps the voltage of the node B to the ground level. The VBB voltage settles at $|V_{tp}| - V_{dd} - V_{tn}$. When clk changes to logic high, the node A changes to V_{tp} and the node B is capacitively coupled to $-V_{dd}$. As a result, V_{BB} voltage changes to $-V_{dd}$. This circuit requires triple-well technology to eliminate minority carrier injection of the n1 transistor. To limit the power consumption of this circuit during DRAM's stand-by mode, the frequency of the clk signal can be reduced. This is possible to implement with BBG's own ring oscillator controlled by BBG's enable signal.

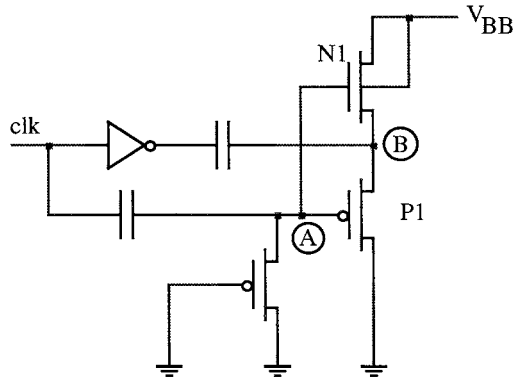


FIGURE 53.41 Low-voltage pumping circuit.

A boosted voltage circuit (BVG) is used in DRAMs to generate a power supply signal higher than V_{dd} for driving the word-lines. This word-line voltage is higher than V_{dd} by at least the threshold voltage. The boosted level cannot be directly applied to drive the load. An isolation transistor is necessary to separate the switching boosted voltage from the load. One such arrangement is shown in Fig. 53.42.[72] This particular circuit generates an output of $2V_{dd}$. Voltage scaling has no effect on its performance and, therefore, it is suitable for V_{dd} reduction down to sub-1V levels.

Leakage Current Reduction and Data-Retention Power

The key limitation in achieving a battery (1 V) or solar cell (0.5 V) operation will be the subthreshold power consumption that will dominate both active and stand-by DRAM modes. In this subsection, circuit techniques that drastically reduce leakage and data-retention power will be described.

Several methods that address the exponentially increasing threshold voltage in rapidly scaled technologies have been proposed. One such method, a well-driving scheme, uses a dynamic Vt by driving the well (see Fig. 53.43).[64,74] Thus, the threshold voltage is higher during the stand-by mode than in the active mode. The advantage of this method is a fast operation in the active mode and a leakage current suppression in the stand-by mode.

To reduce the subthreshold currents in various DRAM voltage generators, a self-off-time detector circuit could be used.[75] It automatically evaluates the optimal off-time interval and controls the dynamic ON/OFF switching ratio of power-dissipation circuits such as level detectors. This method is directly applicable to any on-chip voltage generator or self-refresh circuit. The block diagram of this architecture is shown in Fig. 53.44.

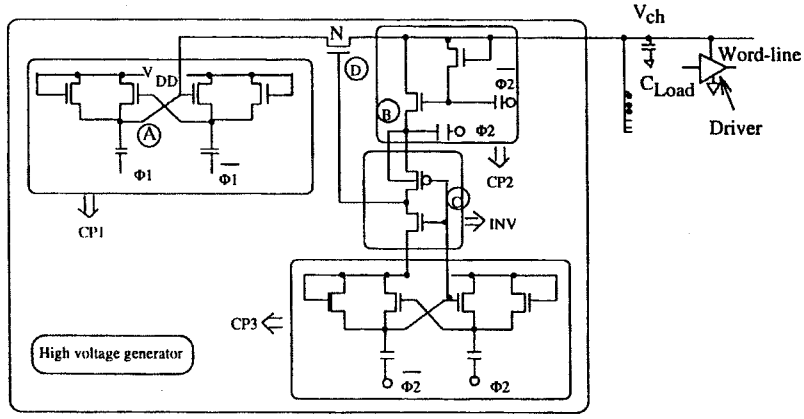


FIGURE 53.42 Boosted voltage generator. (© 1991, IEEE. With permission.)

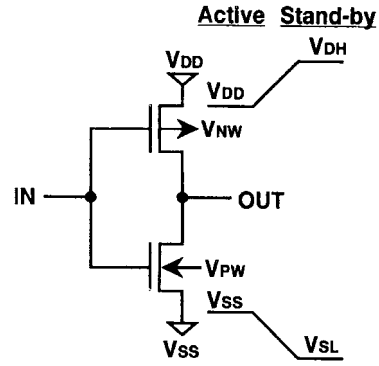


FIGURE 53.43 Low-voltage well-driving scheme. (© 1995, IEEE. With permission.)

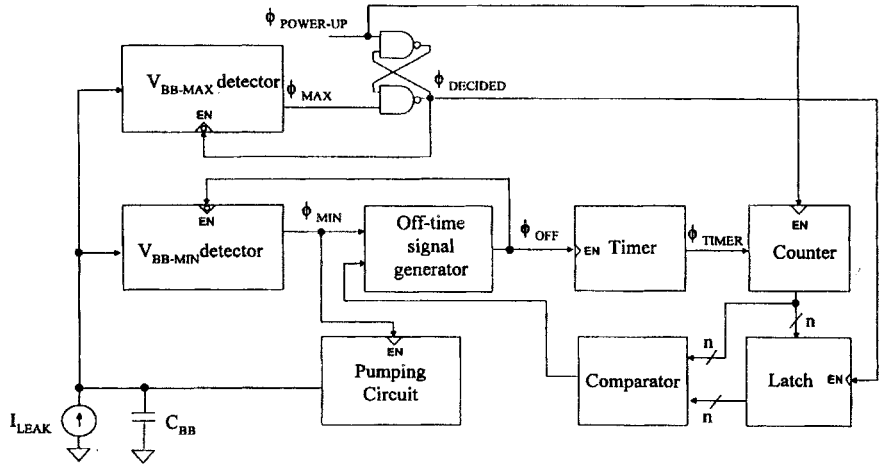


FIGURE 53.44 Block diagram of BBG circuit using the self-off-time detector. (© 1997, IEEE. With permission.)

A charge-transfer presensing scheme (CTPS) with $1/2V_{cc}$ bit-line precharge and a nonreset block control scheme (NRBC) reduces the data-retention current by 75%.[76] The principle of the CTPS technique is shown in Fig. 53.45. The sense amplifier SA and the bit-line BL are separated by the transfer-gate TG. The bit-line is precharged to $1/2V_{ccA}$ (power supply voltage for the array) and the sense amplifier node is precharged to a voltage higher than V_{ccA}. When TG is at a low level, the word-line WL is activated and the data from the memory cell MC is transferred to the bit-line BL. A small voltage change appears on the bit-line pair. Then, the TG voltage is set to the voltage for the charge-transfer condition, and the charge of SA node is transferred to the bit-line. The transfer is complete when the bit-line voltage reaches $V_{TG} - V_{tn}$. After that, a large variation of the readout voltage appears on the SA pair.

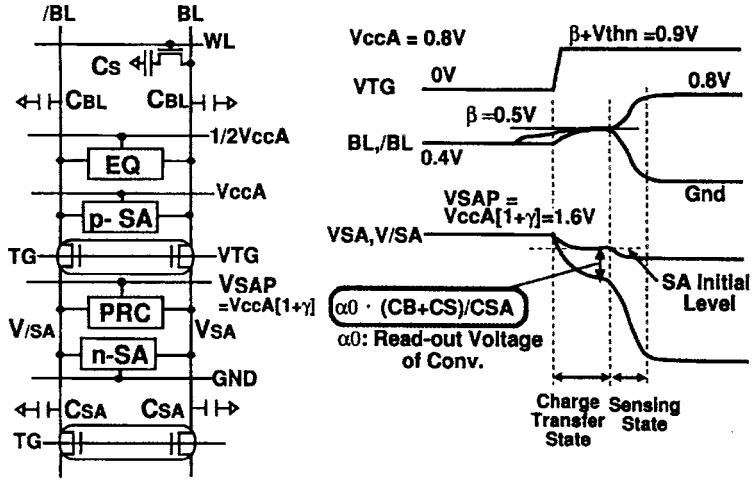


FIGURE 53.45 Concept of CTPS and its circuit organization; BL = $1/2V_{cc}$, V_{ccA} = 0.8 V. (© 1997, IEEE. With permission.)

The CTSP technique reduces the active array current and prolongs the data-retention time. The data-retention power can be reduced further by the nonreset row block control scheme (NRBC), which is used to reduce the charge/discharge number of row block control circuits to 1/128 of the conventional method. The NRBC architecture is shown in Fig. 53.46. NRBC is a divided word-line structure where one subword-line (SWL) in the selected row block is activated if one main word-line (MWL) and one

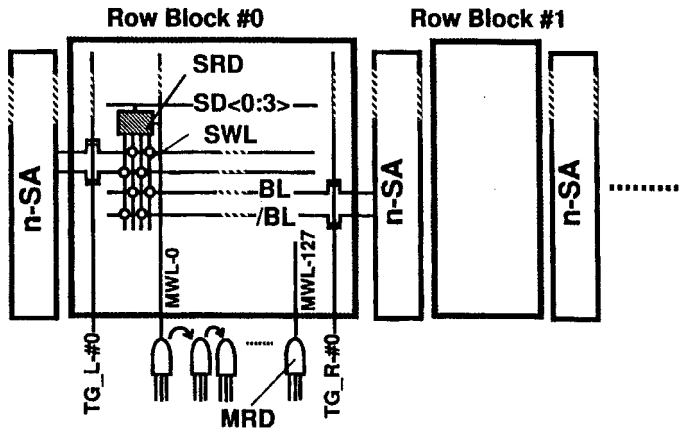


FIGURE 53.46 Basic circuits of the row block control in NRBC. (© 1997 IEEE. With permission.)

of four subdecode signals (SD0~3) are activated in this row block. Also, the transfer-gates TG_L and TG_R are activated at both sides of this row block. After the data-retention mode is set, SD and TG signals do not swing fully at every cycle but only every 128 cycles for activating the same row block. As a result, the row control current is reduced by 70% compared with the conventional scheme.

Another effective method for leakage current reduction is subthreshold leakage current suppression system (SCSS), shown in Fig. 53.47.[78] The method features high drivability (I_{ds}) and low-V_t transistors. The principle of this method is reducing the active mode leakage current with a body bias control and reducing the stand-by mode current by body bias and switched-source impedance. PMOS transistors use the boosted word-line voltage as a body bias, whereas NMOS transistors use memory cell substrate voltage as a body bias. In addition to leakage suppression techniques, extending the refresh time can also significantly reduce power consumption during the stand-by mode, as shown in Eq. 53.4.[67,80,81] The refresh time is determined from the time needed for the stored charge in the memory cell to keep enough margin against leakage at high temperature. In order to achieve long refresh characteristics for a low voltage operation, a negative word-line method can be applied.[67] Figure 53.48 shows the concept of this method.

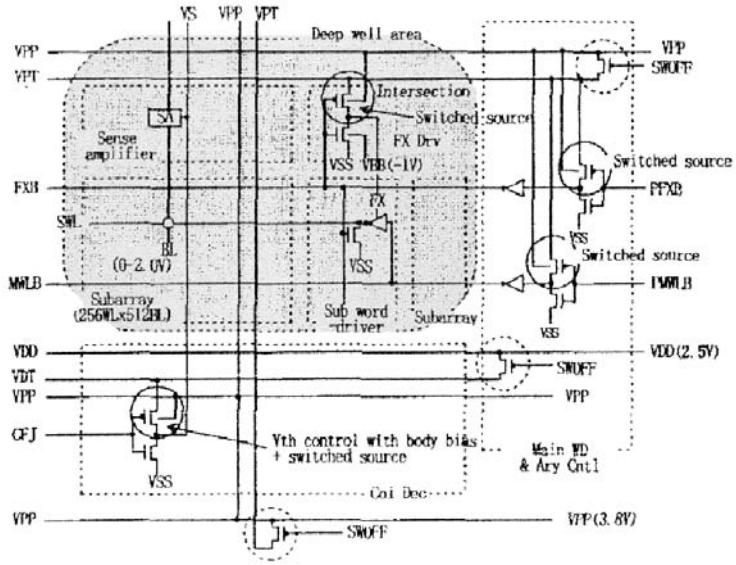


FIGURE 53.47 Subthreshold leakage current suppression system. (© 1998, IEEE. With permission.)

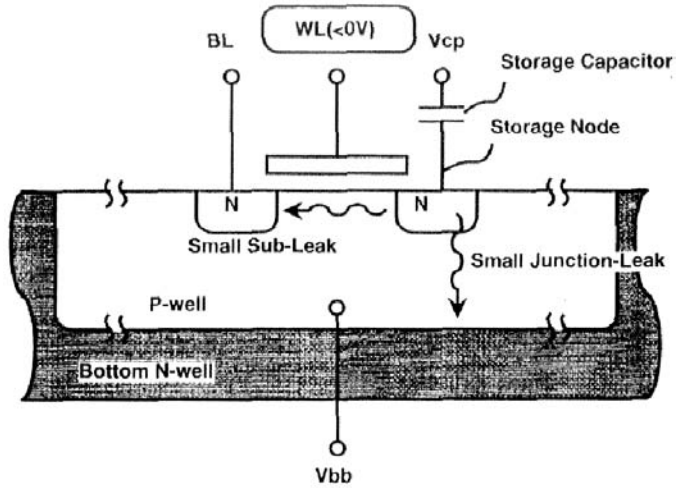


FIGURE 53.48 Principle of the negative voltage word-line technique. (© 1997, IEEE. With permission.)

A negative gate-source voltage V_{gs} is applied, which decreases the subthreshold current of the MC transistor and provides a noise-free dynamic refresh. It also enables the shallow back-bias voltage V_{bb} that reduces the electrical field between the storage node and the p-well region under the memory cell and results in a small junction leakage current. This achieves longer static refresh time. Figure 53.49 shows an example of the negative voltage word-line driver. Dual-period self-refresh (DPS-refresh) scheme is a method that can extend the refresh time by four to six times.[80] The principle of the DPS-refresh scheme is shown in Fig. 53.50 and the corresponding timing diagram in Fig. 53.51. The key concept is to use two different internal self-refresh periods. All word-lines are separated into two groups according to retention test data that are stored in a PROM mode register implemented in the chip periphery. The short period t_1 corresponds to a conventional self-refresh period determined by the minimum retention time in a chip. The long period t_2 is set to the optimum refresh value. If all memory cells connected to a specific word-line have a retention time longer than t_2, they are called long-period word-line cells (LPWL) and are refreshed in the long period of t_2. Otherwise, they are called short-period word-line cells (SPWL) and the word-line is refreshed in the short period t_1. The DPS-refresh operation is then achieved by periodically skipping refresh cycles for LPWLs. The operation is composed of T_1 periods repeated $(n - 1)$, times followed by a T_2. For a refresh cycle during T_1 period, the $\overline{inhibit\_k}$, where k is from 0 to 3, goes low if the word-line selected in the array block k is an LPWL and disables all AND-gated MSi signals. As a result, the refresh operation is not executed. However, during the T_2-

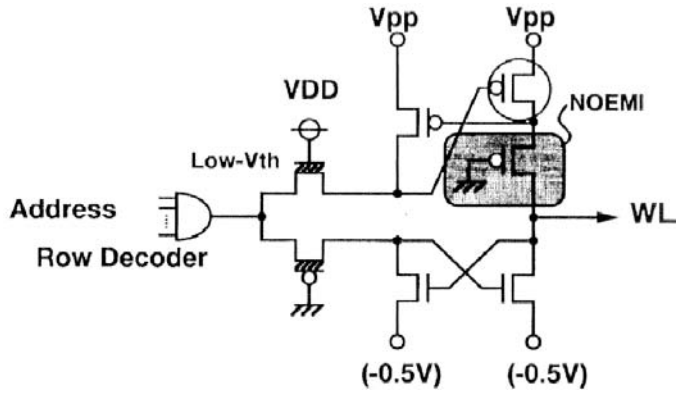


FIGURE 53.49 Negative voltage word-line driver. (© 1997, IEEE. With permission.)

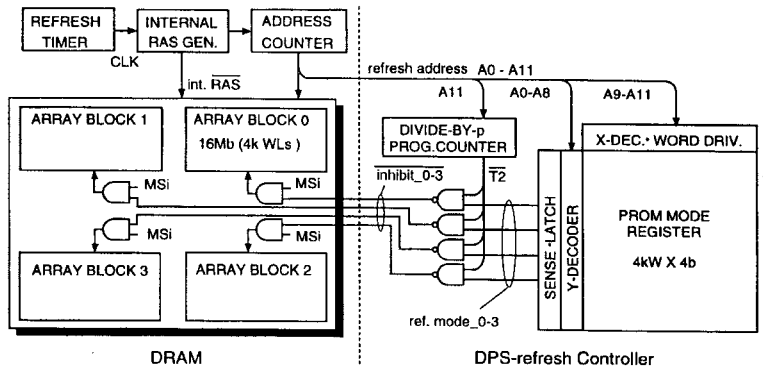


FIGURE 53.50 A schematic diagram of mode-register controlled DPS-refresh method. (© 1998, IEEE. With permission.)

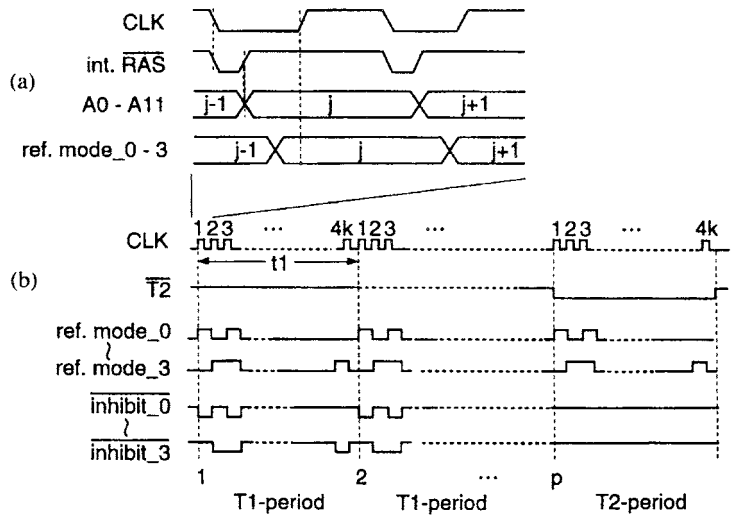


FIGURE 53.51 Timing diagram: (a) PROM read operation, and (b)DPS-refresh operation. (© 1998, IEEE. With permission.)

period, $\overline{inhibit\_k}$ signals are driven high by $\overline{T2}$ clock signal. This signal is generated by the most significant bit refresh address A11 divided by p period using the programmable divide-by-p counter. The period of A11 is equal to the short refresh period t_1. Consequently, LPWLs are refreshed every "$p \times t_1$" periods. The advantage of the DPS-refresh operation is that word-lines which have the same refresh address but are located in different array blocks are individually controlled by $\overline{inhibit\_k}$ signals, which aids in prolonging the refresh time. Using this method, one half of the self-refresh current is saved compared with the conventional self-refresh technique.

53.7 Conclusion

In this chapter, the latest developments in low-power circuit techniques and methods for ROMs, Flash memories, FeRAMs, SRAMs, and DRAMs were described. All major sources of power dissipation in these memories were analyzed. Key techniques for drastic reduction of power consumption were identified. These are: capacitance reduction, very low operating voltages, DC and AC current reduction, and suppression of leakage currents. Many of the reviewed techniques are applicable to other applications such as ASICs, DSPs, etc. Battery and solar-cell operation requires an operating voltage environment in sub-1V area. These conditions demand new design approaches and more sophisticated concepts to retain high device reliability. Experimental circuits operating at these voltage levels slowly start to emerge in all types of memories. However, there is no universal solution for any of these designs, and many challenges still await memory designers.

References

1. Pivin, D., "Pick the Right Package for Your Next ASIC Design," *EDN*, vol. 39, no. 3, pp. 91–108, Feb. 3, 1994.
2. Small, C., "Shrinking Devices Put the Squeeze on System Packaging," *EDN*, vol. 39, no. 4, pp. 41–46, Feb. 17, 1994.
3. Manners, D., "Portables Prompt Low-Power Chips," *Electronics Weekly*, no. 1574, p. 22, Nov. 13, 1991.

4. Mayer, J., "Designers Heed the Portable Mandate," *EDN*, vol. 37, no. 20, pp. 65–68, Nov. 5, 1992.
5. Stephany, R. et al., "A 200MHz 32b 0.5W CMOS RISC Microprocessor," in *ISSCC Digest of Technical Papers*, pp. 15.5-1 to 15.5-2, Feb. 1998.
6. Igura, H. et al., "An 800MOPS 100mW 1.5V Parallel DSP for Mobile Multimedia Processing," in *ISSCC Digest of Technical Papers*, pp. 18.3-1 to 18.3-2, Feb., 1998.
7. Sharma, A. K., *Semiconductor Memories — Technology, Testing and Reliability*, IEEE Press, 1997.
8. de Angel, E. and Swartzlander, E. E. Jr., "Survey of Low Power Techniques for ROMs," in *Proceedings of ISLPED'97*, pp. 7–11, Aug., 1997.
9. Rabaey, J. and Pedram, M., Editors, *Low-Power Methodologies*, Kluwer Academic Publishers, 1996.
10. Margala, M. and Durdle, N. G., "Noncomplementary BiCMOS Logic and CMOS Logic Styles for Low-Voltage Low-Power Operation — A Comparative Study," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 10, pp. 1580–1585, Oct. 1998.
11. Margala, M. and Durdle, N. G., "1.2 V Full-Swing BiNMOS Logic Gate," *Microelectronics Journal*, vol. 29, no. 7, pp. 421–429, Jul. 1998.
12. Margala, M. and Durdle, N. G., "Low-Power 4-2 Compressor Circuits," *International Journal of Electronics*, vol. 85, no. 2, pp. 165–176, Aug. 1998.
13. Grossman, S., "Future Trends in Flash Memories," in *Proceedings of MTDT'96*, pp. 2–3, Aug. 1996.
14. Verma, R., "Flash Memory Quality and Reliability Issues," in *Proceedings of MTDT'96*, pp. 32–36, Aug. 1996.
15. Ohkawa, M. et al., "A 98 mm^2 Die Size 3.3-V 64-Mb Flash Memory with FN-NOR Type Four-Level Cell," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1584–1589, Nov. 1996.
16. Kim, J.-K. et al., "A 120-mm^2 64-Mb NAND Flash Memory Achieving 180 ns/Byte Effective Program Speed," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 5, pp. 670–679, May 1997.
17. Jung, T.-S. et al., "A 117-mm^2 3.3-V Only 128-Mb Multilevel NAND Flash Memory for Mass Storage Applications," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1575–1583, Nov. 1996.
18. Hiraki, M. et al., "A 3.3V 90 MHz Flash Memory Module Embedded in a 32b RISC Microcontroller," in *Advanced Program of ISSCC'99*, p. 17, Nov. 1998.
19. Atsumi, S. et al. ,"A 3.3 V-only 16 Mb Flash Memory with row-decoding scheme," in *ISSCC Digest of Technical Papers*, pp. 42–43, Feb. 1996.
20. Takeuchi, K. et al., "A Multipage Cell Architecture for High-Speed Programming Multilevel NAND Flash Memories," *IEEE Journal Solid-State Circuits*, vol. 33, no. 8, pp. 1228–1238, Aug. 1998.
21. Takeuchi, K. et al., "A Negative Vth Cell Architecture for Highly Scalable, Excellently Noise Immune and Highly Reliable NAND Flash Memories," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 234–235, Jun. 1998.
22. Kawahara, T. et al., "Bit-Line Clamped Sensing Multiplex and Accurate High Voltage Generator for Quarter-Micron Flash Memories," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1590–1600, Nov. 1996.
23. Otsuka, N. and Horowitz, M., "Circuit Techniques for 1.5-V Power Supply Flash Memory," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1217–1230, Aug. 1997.
24. Mihara, M. et al., "A 29 mm2 1.8V-Only 16 Mb DINOR Flash Memory with Gate-Protected Poly-Diode Charge Pump," in *Advanced Program of ISSCC'99*, p. 17, Nov. 1998.
25. Tanzawa, T. et al., "A Compact On-Chip ECC for Low Cost Flash Memories," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 5, pp. 662–669, May 1997.
26. Nozoe, A. et al., "A 256Mb Multilevel Flash Memory with 2MB/s Program Rate for Mass Storage Application," in *Advanced Program of ISSCC'99*, p. 17, Nov. 1998.
27. Imamiya, K. et al., "A 130 mm^2 256Mb NAND Flash with Shallow Trench Isolation Technology," in *Advanced Program of ISSCC'99*, p. 17, Nov. 1998.
28. Hirano, H. et al., "2-V/100ns 1T/1C Nonvolatile Ferroelectric Memory Architecture with Bitline-Driven Read Scheme and Nonrelaxation Reference Cell," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 5, pp. 649–654, May 1997.

29. Fujisawa, H. et al., "The Charge-Share Modified (CSM) Precharge-Level Architecture for High-Speed and Low-Power Ferroelectric Memory," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 5, pp. 655–661, May 1997.
30. Sumi, T. et al., "A 256Kb nonvolatile ferroelectric memory at 3 V and 100 ns," in *ISSCC Digest of Technical Papers*, pp. 268–269, Feb. 1994.
31. Koike, H. et al., "A 60-ns 1-Mb Nonvolatile Ferroelectric Memory with a Nondriven Cell Plate Line Write/Read Scheme," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1625–1634, Nov. 1996.
32. Womack, R. et al., "A 16-kb ferroelectric nonvolatile memory with a bit parallel architecture," in *ISSCC Digest of Technical Papers*, pp. 242–243, Feb. 1989.
33. Bellaouar, A. and Elmasry, M. I., *Low-Power Digital VLSI Design, Circuits and Systems*, Kluwer Academic Publishers, 1996.
34. Itoh, K. et al., "Trends in Low-Power RAM Circuit Technologies," *Proceedings of the IEEE*, pp. 524–543, Apr. 1995.
35. Morimura, H. and Shibata, N., "A 1-V 1-Mb SRAM for Portable Equipment," in *Proceedings of ISLPED'96*, pp. 61–66, Aug. 1996.
36. Ukita, M. et al., "A Single Bitline Cross-Point Cell Activation (SCPA) Architecture for Ultra Low Power SRAMs," in *ISSCC Digest of Technical Papers*, pp. 252–253, Feb. 1994.
37. Amrutur, B. S. and Horowitz, M. A., "Techniques to Reduce Power in Fast Wide Memories," in *Proceedings of SLPE'94*, pp. 92–93, 1994.
38. Toyoshima, H. et al., "A 6-ns, 1.5-V, 4-Mb BiCMOS SRAM," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1610–1617, Nov. 1996.
39. Caravella, J. S., "A 0.9 V, 4 K SRAM for Embedded Applications," in *Proceedings of CICC*, pp. 119–122, May 1996.
40. Caravella, J. S., "A Low Voltage SRAM for Embedded Applications," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 3, pp. 428–432, Mar. 1997.
41. Haraguchi, Y. et al., "A Hierarchical Sensing Scheme (HSS) of High-Density and Low-Voltage Operation SRAMs," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 79–80, Jun. 1997.
42. Mori, T. et al., "A 1V 0.9 mW at 100 MHz 2k×16b SRAM utilizing a Half-Swing Pulsed- Decoder and Write-Bus Architecture in 0.25 μm Dual-Vt CMOS," in *ISSCC Digest of Technical Papers*, pp. 22.4-1–22.4-2, Feb. 1998.
43. Kuang, J. B. et al., "SRAM Bitline Circuits on PD SOI: Advantages and Concerns," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 837–843, Jun. 1997.
44. Kawashima, S. et al., "A Charge-Transfer Amplifier and an Encoded-Bus Architecture for Low-Power SRAM's," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 5, pp. 793–799, May 1998.
45. Amrutur, B. S. and Horowitz, M. A., "A Replica Technique for Wordline and Sense Control in Low-Power SRAM's," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 8, pp. 1208–1219, Aug. 1998.
46. Morimura, H. and Shibata, N., "A Step-Down Boosted-Wordline Scheme for 1-V Battery Operated Fast SRAM's," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 8, pp. 1220-1227, Aug. 1998.
47. Wang, J.-S. et al., "Low-Power Embedded SRAM Macros with Current-Mode Read/Write Operations," in *Proceedings of ISLPED*, pp. 282-287, Aug. 1998.
48. Nii, K. et al., "A Low Power SRAM using Auto-Backgate-Controlled MT-CMOS," in *Proceedings of ISLPED*, pp. 293-298, Aug. 1998.
49. Mai, K. W. et al., "Low-Power SRAM Design Using Half-Swing Pulse-Mode Techniques," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1659-1671, Nov. 1998.
50. Sato, H. et al., "A 5-MHz, 3.6mW, 1.4-V SRAM with Nonboosted, Vertical Bipolar Bit- Line Contact Memory Cell," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1672- 1681, Nov. 1998.
51. Nambu, H. et al., "A 1.8-ns Access, 550-MHz, 4.5-Mb CMOS SRAM," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1650-1658, Nov. 1998.
52. Yamauchi, H. et al., "A 0.8V/100MHz/sub-5mW-Operated Mega-bit SRAM Cell Architecture with Charge-Recycle Offset-Source Driving (OSD) Scheme," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 126–127, Jun. 1996.

53. Itoh, K. et al., "A Deep Sub-V, Single Power-Supply SRAM Cell with Multi-Vt Boosted Storage Node and Dynamic Load," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 132–133, Jun. 1996.
54. Kawaguchi, H. et al., "Dynamic Leakage Cut-off Scheme for Low-Voltage SRAM's," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 140–141, Jun. 1998.
55. Fukushi, I. et al., "A Low-Power SRAM Using Improved Charge Transfer Sense Amplifiers and a Dual-Vth CMOS Circuit Scheme," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 142–143, Jun. 1998.
56. Khellah, M. and Elmasry, M. I., "Circuit Techniques for High-Speed and Low-Power Multi-Port SRAMS," in *Proceedings of ASIC*, pp. 157–161, Sep. 1998.
57. Wang, J.-S. and Lee, H.Y., "A New Current-Mode Sense Amplifier for Low-Voltage Low- Power SRAM Design," in *Proceedings of ASIC*, pp. 163–167, Sep. 1998.
58. Shultz, K. J. et al., "Low-Supply-Noise Low-Power Embedded Modular SRAM," *IEE Proceedings-Circuits, Devices and Systems*, vol. 143, no. 2, pp. 73–82, Apr. 1996.
59. van der Wagt, P. et al., "RTD/HFET Low Standby Power SRAM Gain Cell," *Texas Instruments Research Web-site*, 4 pages, 1997.
60. Greason, J. et al., "A 4.5 Megabit, 560MHz, 4.5GByte/s High Bandwidth SRAM," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 15–16, Jun. 1997.
61. Aoki, M. and Itoh, K., "Low-Voltage and Low-Power ULSI Circuit Techniques," *IEICE Transactions on Electronics*, vol. E77-C, no. 8, pp. 1351–1360, Aug. 1994.
62. Suzuki, T. et al., "High-Speed Circuit Techniques for Battery-Operated 16 MBit CMOS DRAM," *IEICE Transactions on Electronics*, vol. E77-C, no. 8, pp. 1334–1342, Aug. 1994.
63. Lee, K. et al., "Low-Voltage, High-Speed Circuit Designs for Gigabit DRAM's," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 5, pp. 642–648, May 1997.
64. Itoh, K. et al., "Limitations and Challenges of Multigigabit DRAM Chip Design," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 5, pp. 624–634, May 1997.
65. Lee, K.-C. et al., "A 1GBit SDRAM with an Independent Sub-Array Controlled Scheme and a Hierarchical Decoding Scheme," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 103–104, Jun. 1997.
66. Lee, K. et al., "A 1GBit SDRAM with an Independent Sub-Array Controlled Scheme and a Hierarchical Decoding Scheme," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 5, pp. 779–786, May 1998.
67. Tsuruda, T. et al., "High-Speed/High-Bandwidth Design Methodologies for On-Chip DRAM Core Multimedia System LSI's," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 3, pp. 477–482, Mar. 1997.
68. Joo, J.-H. et al., "A 32-Bank 1 Gb Self-Strobing Synchronous DRAM with 1 GByte/s Bandwidth," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1635–11644, Nov. 1996.
69. Eto, S. et al., "A 1-Gb SDRAM with Ground-Level Precharged Bit Line and Nonboosted 2.1-V Word Line," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1697–1702, Nov. 1998.
70. Kato, Y. et al., "Non-Precharged Bit-Line Sensing Scheme for High-Speed Low-Power DRAMs," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 16–17, Jun. 1998.
71. Tsikikawa, Y. et al., "An Efficient Back-Bias Generator with Hybrid Pumping Circuit for 1.5V DRAMs," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 85–86, May 1993.
72. Nakagome, Y. et al., "An Experimental 1.5-V 64-Mb DRAM," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 4, pp. 465–471, Apr. 1991.
73. Tanaka, H. et al., "A Precise On-Chip Voltage Generator for a Giga-Scale DRAM with a Negative Word-Line Scheme," in *Digest of Technical Papers of Symposium on VLSI Circuits*, pp. 94–95, Jun. 1998.
74. Seta, K. et al., "50% Active Power Saving Without Speed Degradation Using Standby Power Reduction (SPA) Circuit," in *ISSCC Digest of Technical Papers*, pp. 318–319, Feb. 1995.
75. Song, H. J., "A Self-Off-Time Detector for Reducing Standby Current of DRAM," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 10, pp. 1535–1542, Oct. 1997.

76. Tsukude, M. et al., "A 1.2- to 3.3-V Wide Voltage-Range/Low-Power DRAM with a Charge-Transfer Presensing Scheme," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1721–1727, Nov. 1997.
77. Shimomura, K. et al., "A 1-V 46-ns 16-Mb SOI-DRAM with Body Control Technique," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1712–1720, Nov. 1997.
78. Hasegawa, M. et al., "A 256 Mb SDRAM with Subthreshold Leakage Current Suppression," in *ISSCC Digest of Technical Papers*, pp. 5.5-1 to 5.5-2, Feb. 1998.
79. Okudi, T. and Murotani, T., "A Four-Level Storage 4-Gb DRAM," *IEEE Journal of Solid- State Circuits*, vol. 32, no. 11, pp. 1743–1747, Nov. 1997.
80. Idei, Y. et al., "Dual-Period Self-Refresh Scheme for Low-Power DRAM's with On-Chip PROM Mode Register," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 2, pp. 253–259, Feb. 1998.
81. Tanizaki, T. et al., "Practical Low Power Design Architecture for 256 Mb DRAM," in *Proceedings of ESSCIRC'97*, pp. 188–191, Sep. 1997.
82. Hamanoto, T. et al., "400-MHz Random Column Operating SDRAM Techniques with Self-Skew Compensation," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 5, pp. 770–778, May 1998.

Song, B. "Nyquist-Rate ADC and DAC"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

54

Nyquist-Rate ADC and DAC

Bang-Sup Song
University of California at San Diego

54.1 Introduction

The rapidly growing electronics field has witnessed the digital revolution that started with the digital telephone switching system in the early 1970s. The trend continued with digital audio in the 1980s and with digital video in the 1990s. The digital technique is expected to prevail in the coming multimedia era and to influence even future wireless PCS/PCN systems. All electrical signals in the real world are analog in nature, and their waveforms are continuous in time. Since most signal processing is done numerically in discrete time, devices that convert an analog waveform into a stream of discrete digital numbers, or vice versa, have become technical necessities in implementing high-performance digital processing systems. The former is called an analog-to-digital converter (ADC or A/D converter), and the latter is called a digital-to-analog converter (DAC or D/A converter).

Typical systems in this digital era can be grouped and explained as in Fig. 54.1. The processed data are stored and recovered later using magnetic or optical media such as tape, magnetic disc, or optical disc. The system can also transmit or receive data through communication channels such as telephone

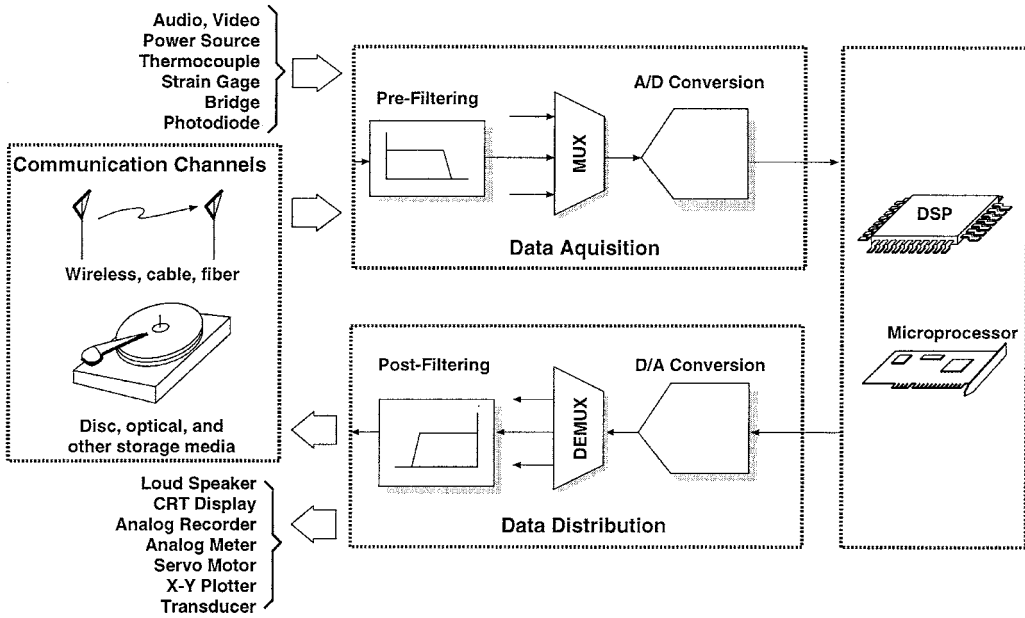


FIGURE 54.1 Information processing systems.

switch, cable, optical fiber, and wireless RF media. Through the Internet computer networks, even compressed digital video images are now made accessible from anywhere and at any time.

Resolution

Resolution is a term used to describe a minimum voltage or current that an ADC/DAC can resolve. The fundamental limit is a quantization noise due to the finite number of bits used in the ADC/DAC. In an N-bit ADC, the minimum incremental input voltage of $V_{ref}/2^N$ can be resolved with a full-scale input range of V_{ref}. That is, limited 2^N digital codes are available to represent the continuous analog input. Similarly, in an N-bit DAC, 2^N input digital codes can generate distinct output levels separated by $V_{ref}/2^N$ with a full-scale output range of V_{ref}. The *signal-to-noise ratio* (SNR) is defined as the power ratio of the maximum signal to the in-band uncorrelated noise. The spectrum of the quantization noise is evenly distributed within the *Nyquist bandwidth* (half the sampling frequency). This inband rms noise decreases by 6 dB when the oversampling ratio is doubled. This implies that, when oversampled, the SNR within the signal band can be made higher. The SNR of an ideal N-bit ADC/DAC is approximated as

$$\text{SNR} = 1.5 \times 2^{2N} \approx 6.02\text{N} + 1.76 \left(\text{dB}\right) \tag{54.1}$$

The resolution is usually characterized by the SNR, but the SNR accounts only for the uncorrelated noise. The real noise performance is better represented by the *signal-to-noise and distortion ratio* (SNDR, SINAD, or TSNR), which is the ratio of the signal power to the total inband noise including harmonic distortion. Also, a slightly different term is often used in place of the SNR. The useful signal range or *dynamic range* (DR) is defined as the power ratio of the maximum signal to the minimum signal. The minimum signal is defined as the smallest signal for which the SNDR is 0 dB, while the maximum signal is the full-scale signal. Therefore, the SNR of the non-ideal ADC/DAC can be lower than the ideal DR because the noise floor can be higher with a large signal present. In practice, performance is not only

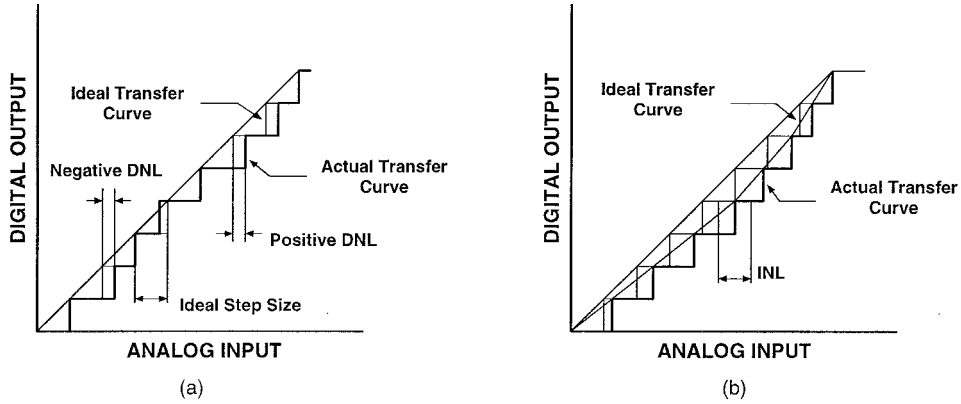


FIGURE 54.2 Definition of ADC nonlinearities: (a) DNL and (b) INL.

limited by the quantization noise but also by non-ideal factors such as noises from circuit components, power supply coupling, noisy substrate, timing jitter, settling, and nonlinearity, etc. An alternative definition of the resolution is the *effective number of bits* (ENOB), which is defined by

$$\text{ENOB} = \frac{\text{SNDR} - 1.76}{6.02} \left(\text{bits}\right) \tag{54.2}$$

Usually, the ENOB is defined for the signal at half the sampling frequency.

Linearity

The input/output ranges of an ideal N-bit ADC/DAC are equally divided into 2^N small units, and one least significant bit (LSB) in the digital code corresponds to the analog incremental voltage of $V_{ref}/2^N$. Static ADC/DAC performance is characterized by *differential nonlinearity* (DNL) and *integral nonlinearity* (INL). The DNL is a measure of deviation of the actual ADC/DAC step from the ideal step for one LSB, and the INL is a measure of deviation of the ADC/DAC output from the ideal straight line drawn between two end points of the transfer characteristic. Both DNL and INL are measured in the unit of an LSB. In practice, the largest positive and negative numbers are usually quoted to specify the static performance. The examples of these DNL and INL definitions for ADC are explained in Fig. 54.2.

However, several different definitions of INL may result, depending on how two end points are defined. In some architectures, the two end points are not exactly 0 and V_{ref}. The non-ideal reference point causes an offset error, while the non-ideal full-scale range gives rise to a gain error. In most applications, these offset and gain errors resulting from the non-ideal end points do not matter, and the integral linearity can be better defined in a relative measure using a straight-line linearity concept rather than the end-point linearity. The straight line can be defined as two end points of the actual transfer function, or as a theoretical straight line adjusted for best fit. The former definition is sometimes called end-point linearity, while the latter is called best-straight-line linearity.

Unlike ADC, the output of a DAC is a sampled-and-held step waveform held constant during a word clock period. Any deviation from the ideal step waveform causes an error in the DAC output. High-speed DACs which usually have a current output are either terminated with a 50 to 75-Ω low-impedance load or buffered by a wideband transresistance amplifier. The linearity of a DAC is often limited dynamically by the non-ideal settling of the output node. Anything other than ideal exponential settling results in linearity errors.

Monotonicity

In both the ADC and the DAC, the output should increase over its full range as the input increases. That is, the negative DNL should be smaller than one LSB for any ADC/DAC to be monotonic. Monotonicity is critical in most applications, in particular digital control or video applications. The source of non-monotonicity is an inaccuracy in binary weighting of a DAC. For example, the most significant bit (MSB) has a weight of half the full range. If the MSB weight is not accurate, the full range is divided into two non-ideal half ranges, and a major error occurs at the midpoint of the full scale. The similar non-monotonicity can take place at the quarter and one-eighth points. In DACs, monotonicity is inherently guaranteed if a DAC uses thermometer decoding. However, it is impractical to implement high-resolution DACs using thermometer codes since the number of elements grows exponentially as the number of bits increases. Therefore, to guarantee monotonicity in practical applications, DACs have been implemented using either a segmented DAC or an integrator-type DAC. Oversampling interpolative DACs also achieve monotonicity using a pulse-density modulated bitstream filtered by a lossy integrator or by a low-pass filter. Similarly, ADCs using slope-type, subranging, or oversampling architectures are monotonic.

Clock Jitter

Jitter is loosely defined as a timing error in analog-to-digital and digital-to-analog conversions. The clock jitter greatly affects the noise performance of both ADCs and DACs. For example, in ADCs, the right signal sampled at the wrong time is the same as the wrong signal sampled at the right time. Similarly, DACs need precise timing to correctly reproduce an analog output signal. If an analog waveform is not generated with the identical timing with which it is sampled, distortion will result because the output changes at the wrong time. This in turn introduces either spurious components related to the jitter frequency or a higher noise floor unless the jitter is periodic. If the jitter has a Gaussian distribution with an rms jitter of Δt, the worst-case SNR resulting from this random clock jitter is

$$\mathrm{SNR} = -20 \times \log \frac{2\pi B \Delta t}{M^{1/2}} \tag{54.3}$$

where B is the signal bandwidth and M is the oversampling ratio. The oversampling ratio M is defined as

$$M = \frac{f_s}{2B} \tag{54.4}$$

where f_s is the sampling clock frequency. The timing jitter error is more critical in reproducing high-frequency components. In other words, for an N-bit ADC/DAC, an upper limit for the tolerable clock jitter is

$$\Delta t \le \frac{1}{2\pi B 2^N} \left(\frac{2M}{3} \right)^{1/2} \tag{54.5}$$

This implies that the error power induced in the baseband by clock jitter should be no larger than the quantization noise. For example, a Nyquist-rate 16-b ADC/DAC with a 22-kHz bandwidth should have a clock jitter of less than 90 ps.

Nyquist-Rate vs. Oversampling

In recent years, high-resolution ADCs and DACs at the low end of the spectrum such as for digital audio, voice, and instrumentation are dominantly implemented using oversampling techniques. Although

Nyquist-rate techniques can achieve comparable resolution, such techniques are in general sensitive to non-ideal factors such as process, component matching, and even environmental changes. The inherent advantage of oversampling provides a unique solution in the digital VLSI environment. The oversampling technique achieves high resolution by trading speed for accuracy. The oversampling lessens the effect of quantization noise and clock jitter. However, the quantization or regeneration of a signal above MHz using oversampling techniques is costly even if possible. Therefore, typical applications for high-sampling rates require sampling at a Nyquist rate.

54.2 ADC Design Arts

The conversion speed of the ADC is limited by the time needed to complete all comparator decisions. Flash ADCs make all the decisions at once, while successive-approximation ADCs make one-bit decisions at a time. Although it is fast, the complexity of the flash ADC grows exponentially. On the other hand, the successive-approximation ADC is simple but slow since the bit decisions are made in sequence. Between these two extremes, there exist many architectures resolving a finite number of bits at a time, such as pipeline and multi-step ADCs. They balance complexity and speed. Figure 54.3 shows recent high-speed ADC applications in the resolution-versus-speed plot. ADC architecture depends on system requirements. For example, with IF (intermediate frequency) filters, wireless receivers need only 5 to 6 b ADC at a few MHz sampling rate. However, without IF filters, the dynamic range of 12 to 14 b is required for the IF sampling depending on IF as shown in Fig. 54.3.

State of the Art

Some architectures are preferred to others for certain applications. Three architectures stand out for three important areas of applications. For example, the oversampling converter is exclusively used to achieve high resolution above the 12-b level at low frequencies. The difficulty in achieving better than 12-b matching in conventional techniques gives a fair advantage to the oversampling technique. For medium

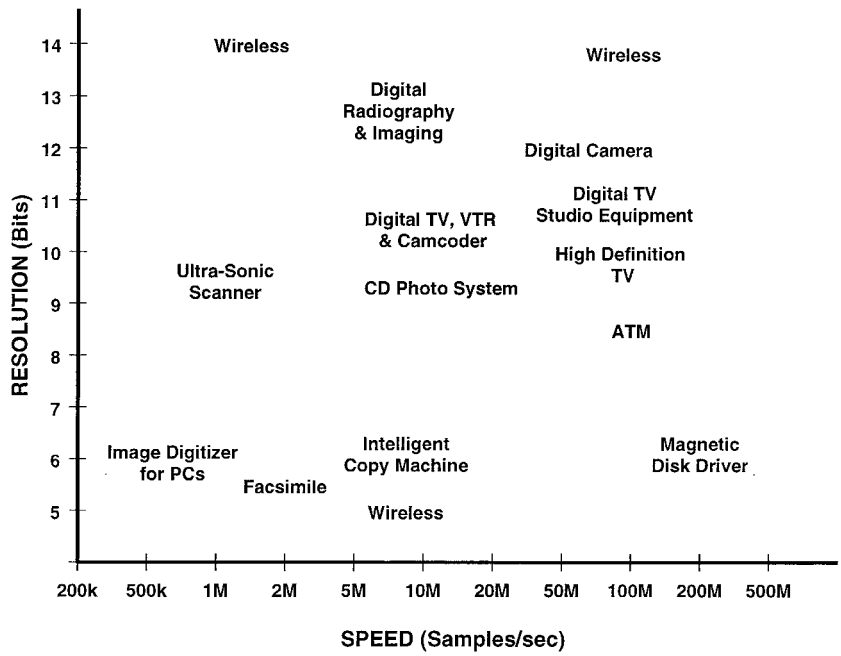


FIGURE 54.3 Recent high-speed ADC applications.

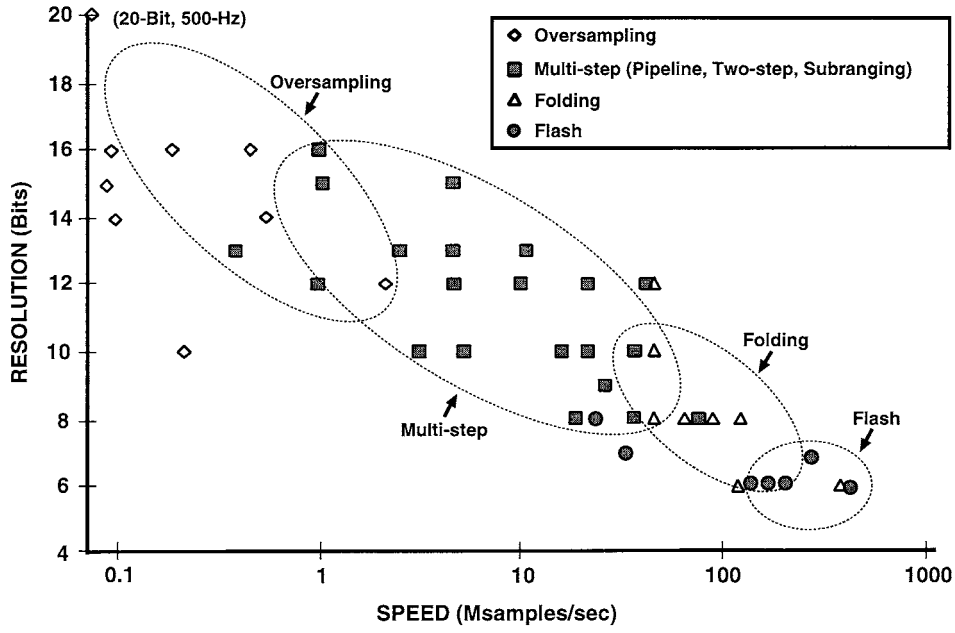


FIGURE 54.4 Performance of recently published ADCs: resolution versus speed.

speed with high resolution, pipeline or multi-step ADCs are promising. At extremely high frequencies, only flash and folding ADCs survive, but with low resolution. Figure 54.4 is a resolution-versus-speed plot showing this trend. As both semiconductor process and design technologies advance, the performance envelope will be pushed further. The demand for higher resolution at higher sampling rates is a main driver of this trend.

Technical Challenge in Digital Wireless

In digital wireless systems, a need to quantize and to create a block of spectrum with low intermodulation has become the single most challenging problem. Implementing IF filters digitally has already become a necessity in wireless cell sites and base stations. Even in hand-held units, placing data conversion blocks closer to the RF (radio frequency) has many advantages. A substantial improvement in system cost and complexity of the RF circuitry can be realized by implementing high selectivity function digitally, and the digital IF can increase immunity to adjacent and alternate channel interferences. Furthermore, the RF transceiver architecture can be made independent of the system and can be adapted to different standards using software. Low-spurious, low-power data converters are key components in this new software radio environment.

The fundamental limit in quantizing IF spectrum is the crosstalk and overload, and the system performance heavily depends on the SFDR (spurious-free dynamic range) of the sampling ADC. To meet this growing demand, low-spurious data conversion blocks are being actively developed in ever-increasing numbers. For a 14b-level ideal dynamic range while sampling at 50 MHz, it is necessary to control the sampling jitter down below 0.32 ps. Considering that the current state-of-the-art commercial bipolar chip exhibits the jitter range of 0.7 ps, the jitter on the order of a fraction of a picosecond is considered to be at the limit of CMOS capability. However, unlike nonlinearity that causes interchannel mixing, the random jitter in IF sampling increases only the random noise floor. As a result, the random jitter is not considered fundamental in this application.

This challenging new application for digital IF processing will lead to the implementation of data converters with very wide SFDR of more than 90 dB. Considering the current state of the art in CMOS

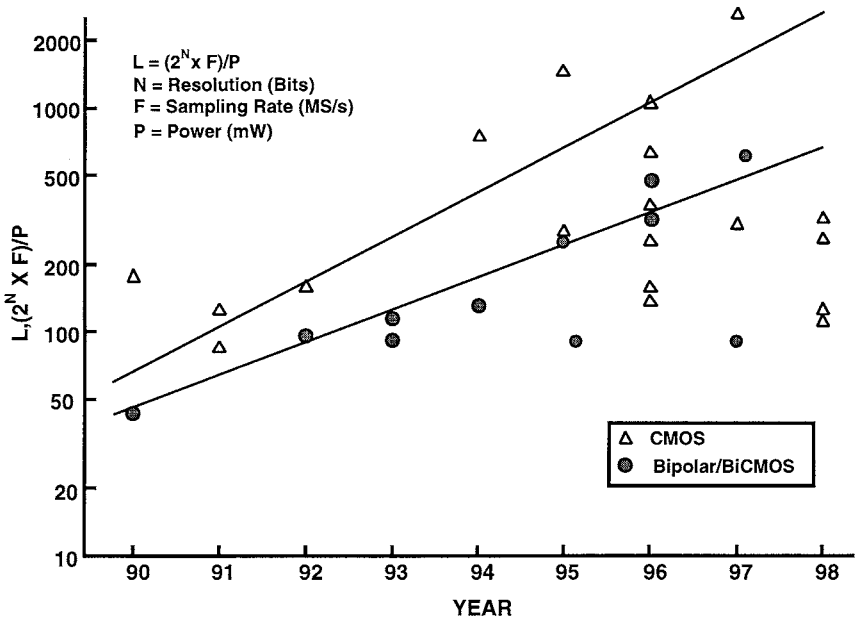


FIGURE 54.5 Figure 54.of merit (L) versus year.

ADCs, most architectures known to date are unlikely to achieve a high sampling rate of over 50 MHz, even with 0.2 to 0.3 μm technologies. Although a higher sampling rate of 65 MHz is reported using bipolar and BiCMOS (bipolar and CMOS), it has been implemented at a 12-b level. However, two high-speed candidate architectures, pipeline (or multi-step) and folding, are potential candidate architectures to challenge these limits with new system approaches.

ADC Figure of Merit

The ADC performance is often represented by a figure of merit L which is defined as $L = 2^N \times f_s/P$, where N is the number of bits, f_s is the sampling rate in Msamples/s, and P is the power consumption in mW. The higher the L is, the more bits are obtained at higher speed with lower power. The plot of L versus year shown in Fig. 54.5 shows the low-power and high-speed trend both for leading integrated CMOS and bipolar/BiCMOS ADCs published in the last decade.

54.3 ADC Architectures

In general, the main criteria of choosing ADC architectures are resolution and speed, but auxiliary requirements such as power, chip area, supply voltage, latency, operating environment, or technology often limit the choices. The current trend is toward low-cost integration without using expensive discrete technologies such as thin film and laser trimming. Therefore, a growing number of ADCs are being implemented using mainstream VLSI technologies such as CMOS or BiCMOS.

Slope-Type ADC

Traditionally, slope-type ADCs have been used for multimeters or digital panel meters mainly because of their simplicity and inherent high linearity. There can be many variations, but dual- or triple-slope techniques are commonly used because the single-slope method is sensitive to the switching error. The resolution of this type of ADC depends on the accurate control of charge on the capacitor. The dual-slope technique in Fig. 54.6(a) starts with the initialization of the integrating capacitor by opening the

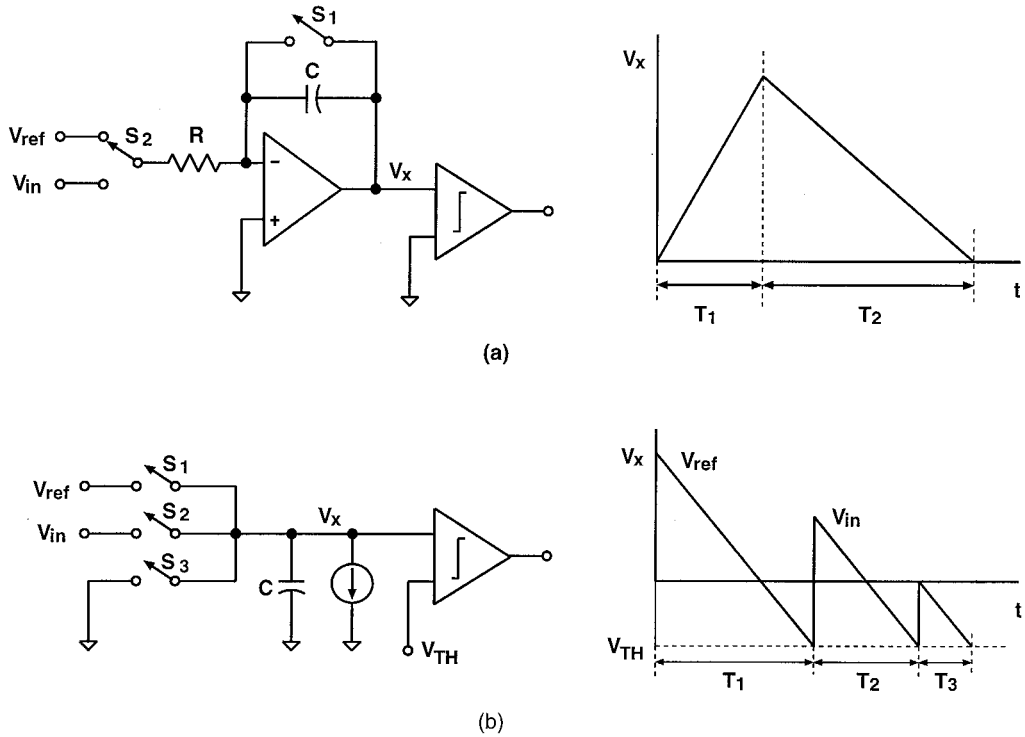


FIGURE 54.6 (a) Dual-slope and (b) triple-slope ADC techniques.

switch S_1 with the input switch S_2 connected to V_{ref}. If V_{ref} is negative, V_x will increase linearly with a slope of V_{ref}/RC. After a time T_1, the switch S_2 is switched to V_{in}. Then, V_x will decrease with a new slope of $-V_{in}/RC$. The comparator detects the zero-crossing time T_2. From T_1 and T_2, the digital ratio of V_{in}/V_{ref} can be obtained as T_1/T_2. The triple-slope technique shown in Fig. 54.6(b) needs no op-amp to reduce the offset effect. Unlike the dual-slope method comparing two slopes, it measures three times T_1, T_2, and T_3 by charging the capacitor with V_{ref}, V_{in}, and ground with three switches S_1, S_2, and S_3, respectively. The comparator threshold can be set to negative V_{TH}. From three time measurements, the ratio of V_{in}/V_{ref} can be computed as $(T_2 - T_3)/(T_1 - T_3)$.

Successive-Approximation ADC

The simplest concept of A/D conversion is comparing analog input voltage with an output of a DAC. The comparator output is fed back through the DAC as explained in Fig. 54.7. The successive-approximation

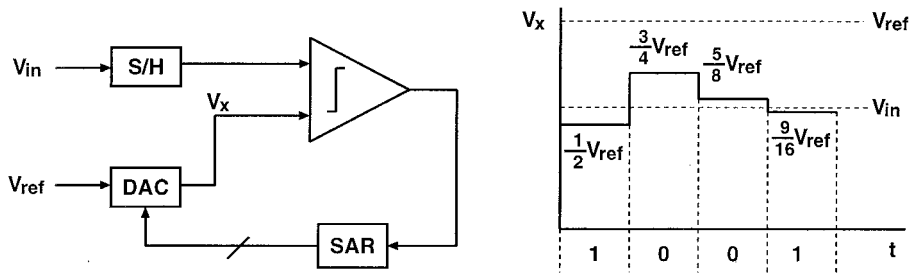


FIGURE 54.7 Successive-approximation ADC technique.

register (SAR) performs the most straightforward binary comparison. The sampled input is compared with the DAC output by progressively dividing the range by two as explained in the 4-b example. The conversion starts by sampling input, and the first MSB decision is made by comparing the sample-and-hold (S/H) output with $V_{ref}/2$ by setting the MSB of the DAC to 1. If the input is higher, the MSB stays as 1. Otherwise, it is reset to 0. In the second bit decision, the input is compared with $3V_{ref}/4$ in this example by setting the second bit to 1. Note that the previous decision set the MSB to 1. If the input is lower, as in the example shown, the second bit is set to 0, and the third bit decision is done by comparing the input with $5V_{ref}/8$. This comparison continues until all the bits are decided. Therefore, the N-bit successive-approximation ADC requires N+1 clock cycles to complete one sample conversion.

The performance of the successive-approximation ADC is limited by the DAC resolution and the comparator accuracy. The commonly used DACs for this architecture are a resistor-string DAC and a capacitor-array DAC. Although binary-weighted capacitors have a 10b-level matching in MOS,[1] diffused resistors have poor matching and high voltage coefficient. If differential resistor-string DACs are used, performance can be improved to the capacitor-array DAC level.[2] In general, the capacitor DAC exhibits poor DNL while the resistor-string DAC exhibits poor INL.

Flash ADC

The most straightforward way of making an ADC is to compare the input with all the divided levels of the reference simultaneously. Such a converter is called a flash ADC, and the conversion occurs in one step. The flash ADC is the fastest among all ADCs. The flash ADC concept is explained in Fig. 54.8, where divided reference voltages are compared to the input. The binary encoder is needed because the output of the comparator bank is thermometer-coded. The resolution is limited both by the accuracy of the

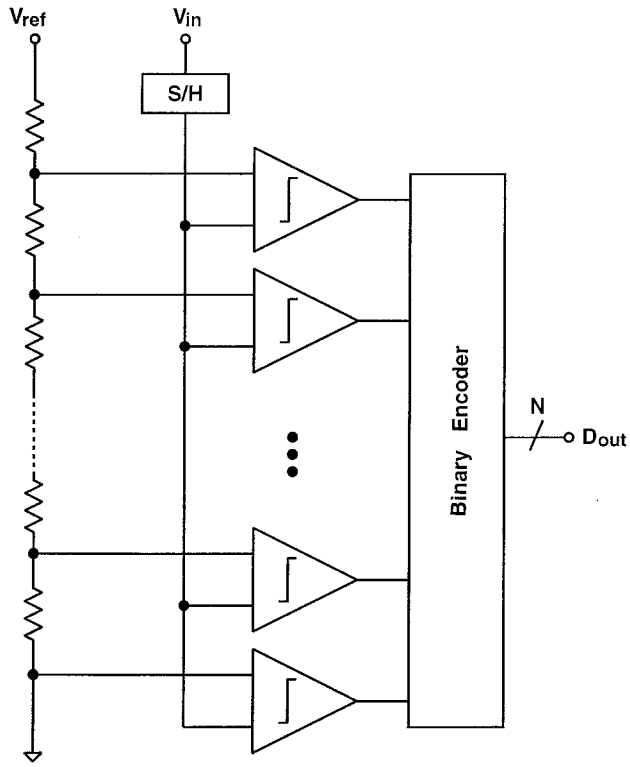


FIGURE 54.8 Flash ADC technique.

divided reference voltages and by the comparator resolution. The metastability of the comparator produces a sparkle noise when the gain of the comparator is low. The reference division can be done using capacitor dividers[3,4] or transistor sizing[5] for small-scale flash ADCs. However, only resistor-string DACs can provide references as the number of bits grows.

In practical implementations, the limit is the exponential growth in the number of comparators and resistors. For example, an N-bit flash needs $2^N - 1$ comparators and 2^N resistors. Furthermore, for the Nyquist-rate sampling, the input needs a S/H to freeze the input for comparison. As the number of bits grows, the comparator bank presents a significant loading to the input S/H, diminishing the speed advantage of this architecture. Also, the control of the reference divider accuracy and the comparator resolution degrades, and the power consumption becomes prohibitively high. As a result, flash converters with more than 10-b resolution are rare. Flash ADCs are commonly used as coarse quantizers in the pipeline or multi-step ADCs. The folding/interpolation ADC, which is conceptually a derivative of the flash ADC, reduces the number of comparators by folding the input range.[6]

For high resolution, the flash ADC needs a low-offset comparator with high gain, and the comparator is often implemented in a multi-stage configuration with offset cancelation. The front-end of the multi-stage comparator is called a preamplifier. A technique called *interpolation* saves the number of preamplifiers by interpolating the adjacent preamplifier outputs as shown in Fig. 54.9(a), where two preamplifier outputs V_a and V_b are used to generate three more outputs V_1, V_2, and V_3 using a resistor divider. The interpolation can improve the DNL within the interpolated range, but the overall DNL and INL are not improved. Interpolating any arbitrary number of levels is possible by making more resistor taps. The interpolation is usually done using resistors, but the interpolation in the current domain is also possible. However, interpolating with independent current sources does not improve the DNL.

Another technique called *averaging*, as explained in Fig. 54.9(b) is often used to average out the offsets of the neighboring preamplifiers as well as to enhance the accuracy of the reference divider.[7] The idea is to couple the outputs of the preamplifier transconductance (G_m) stage so that the significant errors can be spread over the adjacent preamplifier outputs as explained. For example, if the coupling resistor value is infinite, there exists no averaging. As the coupling resistor value decreases, one preamplifier output becomes the weighted sum of the outputs of its neighboring preamplifiers. Therefore, the overall DNL

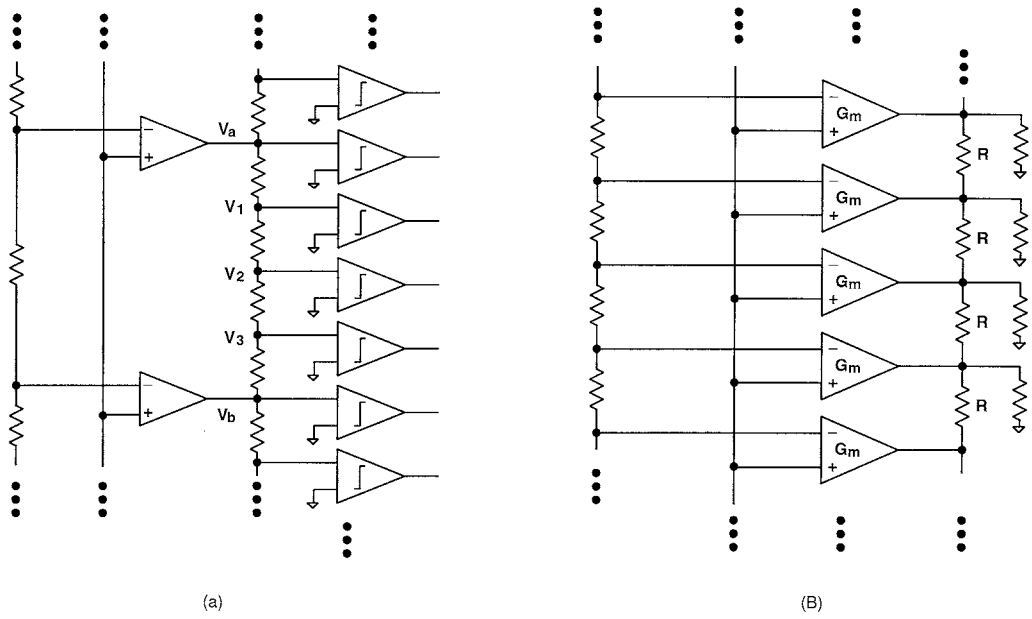


FIGURE 54.9 (a) Interpolation and (b) averaging techniques.

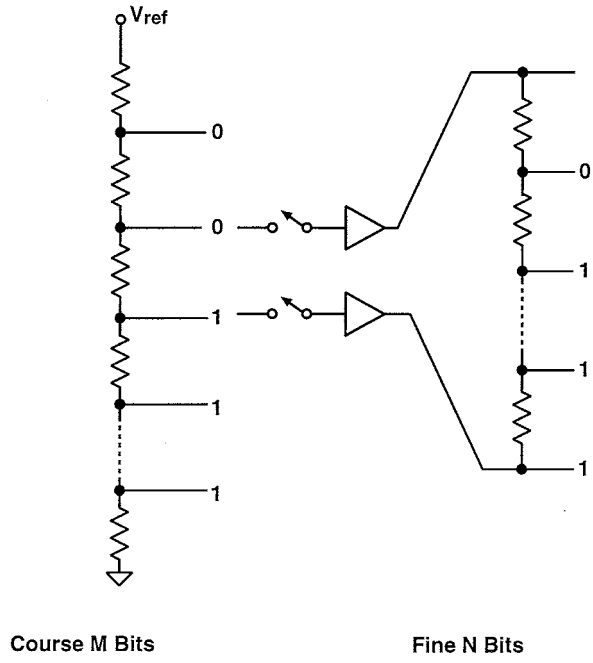


FIGURE 54.10 Coarse and fine reference ladders for two-step subranging ADC.

and INL can improve significantly.[8] However, for the case in which errors to average have the same polarity, the averaging is not that effective. In practice, both the interpolation and the averaging concepts are often combined.

Subranging ADC

Although the interpolation and averaging techniques simplify the flash ADC, the number of comparators stays the same. Instead of making all the bit decisions at once, resolving a few bits at a time makes the system simpler and more manageable. It also enables us to use a digital error correction concept. The simplest subranging ADC concept is explained in Fig. 54.10 for the two-step conversion case. It is a straightforward subranging since one subrange out of 2^M subranges is chosen in the coarse M-bit decision. Once one subrange is selected, the N-bit fine decision can be done using a fine reference ladder interpolating the selected subrange.

Note that the subrange after the coarse decision is $V_{ref}/2^M$ and the fine comparators should have a resolution of M+N bits. Unless the digital error correction with redundancy is used, the coarse comparators should also have a resolution of M+N bits.

Multi-Step ADC

The tactic of making a few bit decisions at a time as shown in the subranging case can be generalized. A slight modification of the subranging architecture shown in Fig. 54.11(a) to include a residue amplifier with a gain of 2^M results in Fig. 54.11(b). The residue is defined as the difference between the input and the nearest DAC output lower than the input. The difference between the two concepts is subtle, but including one residue amplifier drastically changes the system requirements. The obvious advantage of using the residue amplifier is that the fine comparators do not need to be accurate because the residue from the coarse decision is amplified by 2^M. That is, the subrange after the coarse decision is no longer $V_{ref}/2^M$. The disadvantage is the accuracy and settling of the high-gain residue amplifier.

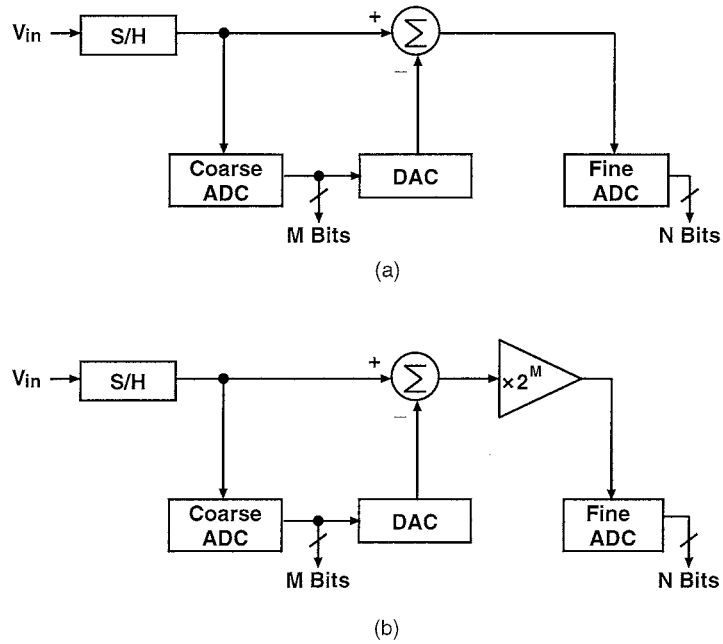


FIGURE 54.11 Variations of the subranging concepts: (a) without and (b) with residue amplifier.

Whether the residue is amplified or not, the subranging block consists of a coarse ADC, a DAC, a residue subtractor, and an amplifier. In theory, this block can be repeated as shown in Fig. 54.12. How many times it is repeated determines the number of steps. So, in general terms, the n-step ADC has n–1 subranging blocks. To complete a conversion in one cycle, usually poly-phase subdivided clocks are needed. Due to the difficulty in clocking, the number of steps for the multi-step architecture is usually limited to two, which does not incur a speed penalty and needs the standard two-phase clocking.

There are many variations in the multi-step architecture. If no ploy-phase clocking is used, it is called a *ripple ADC*. Also in the two-step ADC, if one ADC is repeatedly used both for the coarse and fine decisions, it is called a *recycling ADC*.[9] In this ADC example, the capacitor-array multiplying DAC (MDAC) also performs the S/H function in addition to the residue amplification. This MDAC, with

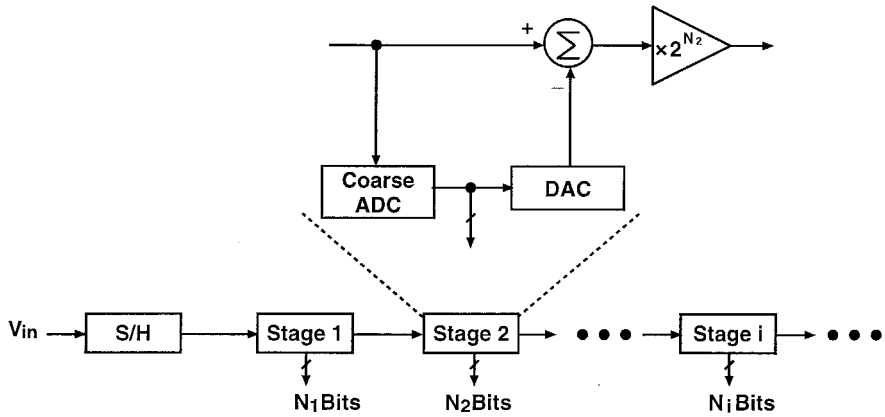


FIGURE 54.12 Multi-step ADC architecture.

either a binary-ratioed or thermometer-coded capacitor array, is a general form of the residue amplifier. The same capacitor array has been used with a comparator to implement a charge-redistribution successive-approximation ADC.[1] This MDAC is suited for MOS technologies, but other forms of the residue amplification are possible using resistor-strings or current DACs.

Pipeline ADC

The complexity of the two-step ADC, although manageable and simpler than the flash ADC, still grows exponentially as the number of bits to resolve increases. Specifically, for high resolution above 12 b, the complexity reaches about the maximum, and a need to pipeline subranging blocks arises. The pipeline ADC architecture shown in Fig. 54.13 is the same as the subranging or multi-step ADC architecture shown in Fig. 54.12 except for the interstage S/H. Since the S/Hs are clocked by alternating clock phases, each stage needs to perform the decision and the residue amplification in each clock phase. Pipelining the residue greatly simplifies the ADC architecture. The complexity grows only linearly with the number of bits to resolve. Due to its simplicity, the pipeline ADCs have been gaining popularity in the digital VLSI environment.

In the pipeline ADC, each stage resolves a few bits quickly and transfers the residue to the following stage so that the residue can be resolved further in the subsequent stages. Therefore, the accuracy of the interstage residue amplifier limits the overall performance. The following four non-ideal error sources can affect the performance of the multi-step or pipeline ADCs: ADC resolution, DAC resolution, gain error of the residue amplifier, and inaccurate settling of the residue amplifier. The offset of the residue amplifier does not affect the linearity, but it appears as a system offset. Among these four error sources, the first three are static, but the residue amplifier settling is dynamic. If the residue amplifier is assumed to settle within one clock phase, three static error sources are limiting the linearity performance.

Figure 54.14 explains the residue from the 2-b stage in the systems shown in Figs. 54.12 and 54.13. In the ideal case, as the input is swept from 0 to the full range V_{ref}, the residue change from 0 to V_{ref} repeats each time V_{ref} is subtracted at the ideal locations of the 2-b ADC thresholds, which are $V_{ref}/4$ apart. In this case, the 2-b stage does not introduce any nonlinearity error. However, in the other cases with ADC, DAC, and gain errors, the residue ranges do not match with the ideal full-scale V_{ref}. If the residue range is smaller than the full range, missing codes are generated; and if the residue goes out of bounds, excessive codes are generated at the ADC thresholds. Unlike the DAC and gain errors, the ADC error appears as a shift of the residue either by the amounts of V_{ref} or $-V_{ref}$ as long as the DAC subtracts the ideal V_{ref} and the residue amplifier gain is ideal. This suggests that the error can be corrected digitally by adding or subtracting the full range.

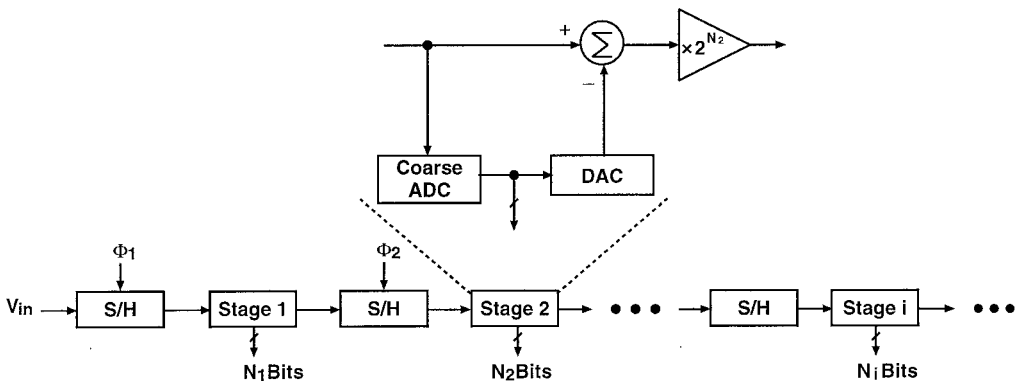


FIGURE 54.13 Pipeline ADC architecture.

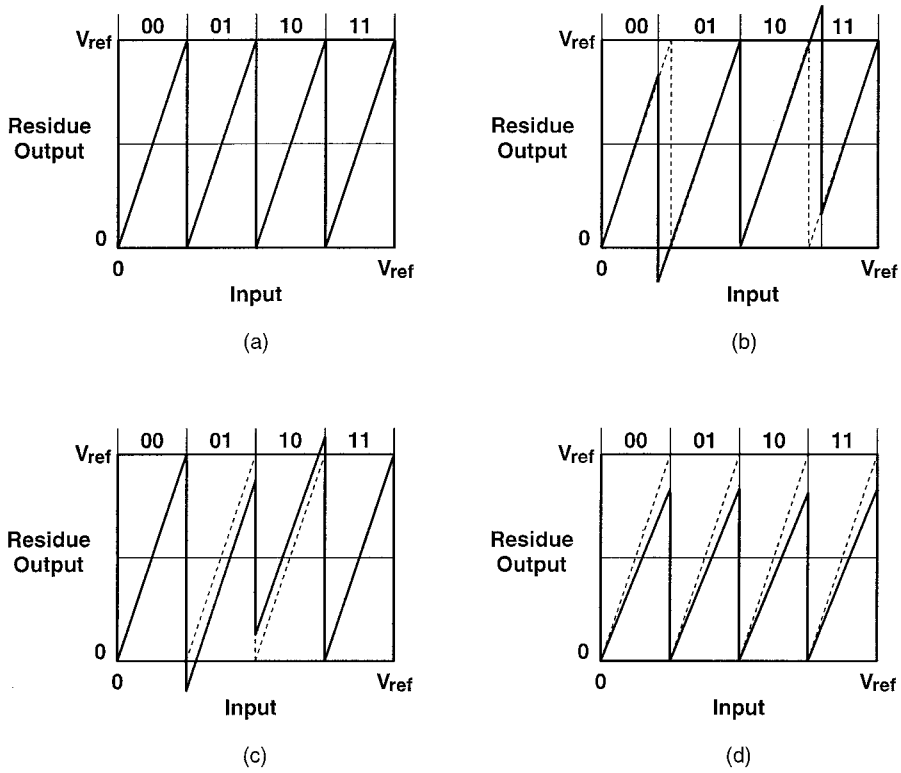


FIGURE 54.14 2b residue versus input: (a) ideal, and with (b) ADC, (c) DAC, and (d) gain errors.

Digital Error Correction

Any multi-step or pipeline ADC system can be made insensitive to the ADC error if the ADC error is digitally corrected. The residue normally going out of the full range can still be digitized by the following stage if the residue amplifier gain is reduced. That is, if the residue amplifier gain is set to 2^{N-1} instead of 2^N, the residue plots are as shown in Fig. 54.15. If the residue is bounded with the full range of 0 to V_{ref}, the inner range from $V_{ref}/4$ to $3V_{ref}/4$ is the normal conversion range, and two redundant outer ranges are used to cover the residue error resulting from the inaccurate coarse conversion. Now the problem is that this redundancy requires extra resolution to cover the overrange. The half ranges on both sides are used to cover the residue error in this 2-b case. That is, one full bit of extra resolution is necessary for

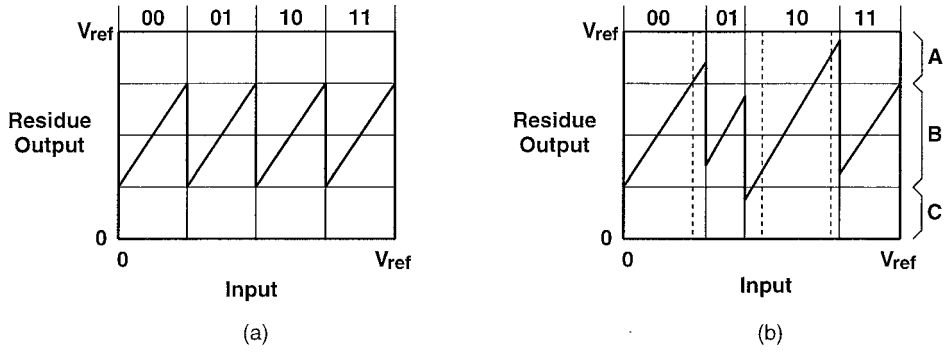


FIGURE 54.15 Over-ranged 2-b residue versus input: (a) ideal and with (b) ADC errors.

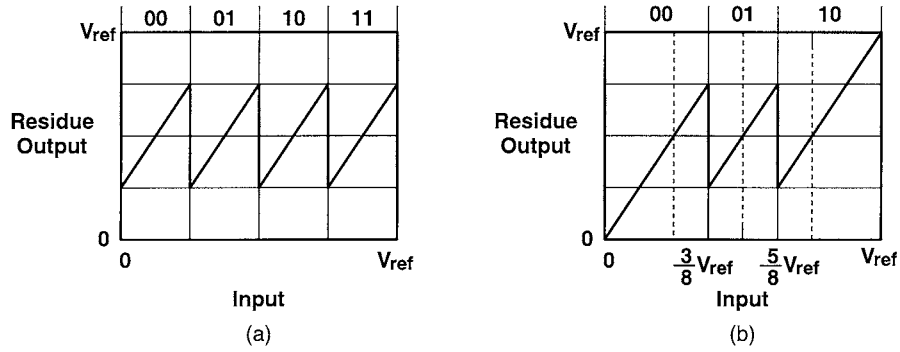


FIGURE 54.16 Half-bit shifted 2-b residue versus input: (a) with and (b) without ADC threshold-level shift.

redundancy in the 2-b case. However, the amount of redundancy depends on the ADC error range to be corrected in the multi-bit cases.[10] In general, it is a tradeoff between comparator resolution and redundancy.

The range marked as B is the normal range, and A and C indicate the correction ranges in Fig. 54.15(b). The digital correction works by subtracting 1 from the previous decision if the residue exceeds $3V_{ref}/4$, as in the case marked as A. On the other hand, if the residue goes below $V_{ref}/4$, as in the case marked as C, 1 is added from the previous decision. Although the digital subtraction is simple, biasing the ADC threshold levels by half the ideal interval has an advantage. Figure 54.16 compares the residue plots for two cases with and without the ADC threshold shift by $V_{ref}/8$. This is to fully utilize the ADC conversion range from 0 to V_{ref}. The shift of $V_{ref}/8$ makes the residue start from 0 and end at V_{ref} in Fig. 54.16(b), contrary to the previous case where the residue starts from $V_{ref}/4$ and ends at $3V_{ref}/4$. This results in saving one comparator. The former case needs 2^N–1 comparators, while the latter case needs 2^N–2. The only minor issue is that the latter exhibits a half LSB systematic offset due to this shift.

This half-bit-level shift makes the ADC error occur only with the same polarity. As a result, only the addition is necessary for digital correction in the case of Fig. 54.16(b). This is explained in the 4-b ADC made of three stages using one-bit correction per stage in Fig. 54.17. The vertical axis marks the signal and residue levels as well as ADC decision levels. The dotted and shaded areas follow the residue paths when the ADC error occurs, but the end results are the same after digital correction. This half interval shift is valid for stages resolving any number of bits. Overall, the digital error correction enables fast data conversion using inaccurate comparators. However, the DAC resolution and the residue gain error still remain as the fundamental limits in multi-step and pipeline ADCs. The currently known ways to overcome these limits are either trimming or self-calibration.

One-Bit Pipeline ADC

The degenerate case of the pipeline ADC is when only one bit is resolved per stage as shown in Fig. 54.18. Each stage multiplies its input V_{in} by two and subtracts the reference voltage V_{ref} to generate the residue voltage. If the sign of $2V_{in} - V_{ref}$ is positive, the bit is 1 and the residue goes to the next stage. Otherwise, the bit is 0 and V_{ref} is added back to the residue before it goes to the next stage. However, in reality, it is more desirable if the reference restoring time is saved. In the non-restoring algorithm, the previous bit decision affects the polarity of the reference voltage to be used in the current bit decision. If the previous bit is 1, the residue voltage is $2V_{in} - V_{ref}$, as in the restoring algorithm. But if the previous bit is 0, the residue voltage is $2V_{in} + V_{ref}$.

The switched-capacitor implementation of the basic functional block performing $2V_{in} \pm V_{ref}$ is explained using two identical capacitors and one op-amp in Fig. 54.19 [11]. During the sampling phase, the bottom plates of two capacitors are switched to the input, and the top plate is connected either to the op-amp output or to the op-amp input common-mode voltage. During the amplification phase, one

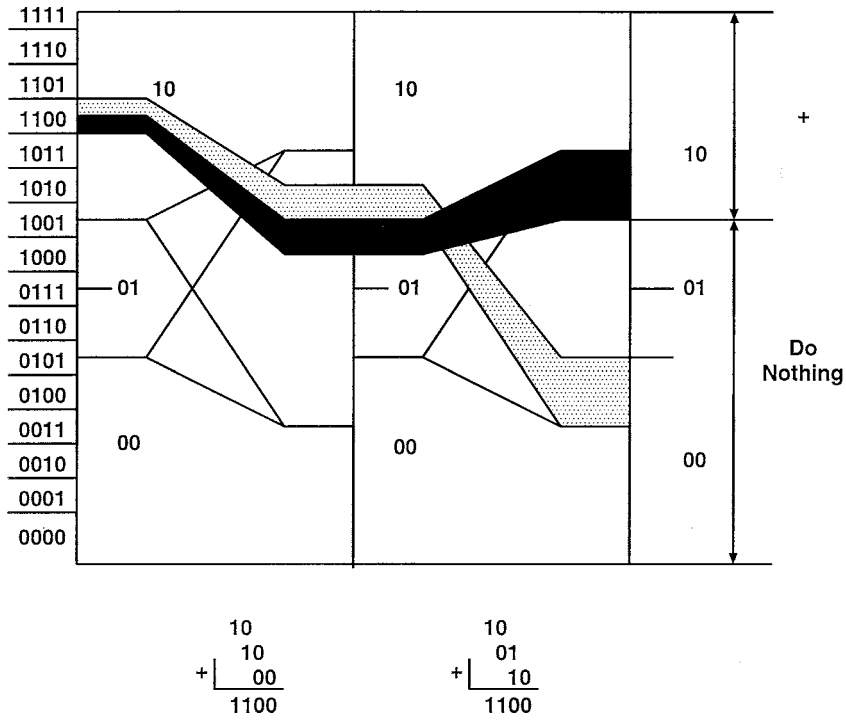


FIGURE 54.17 Example of digital error correction for three-stage 4b ADC (1100).

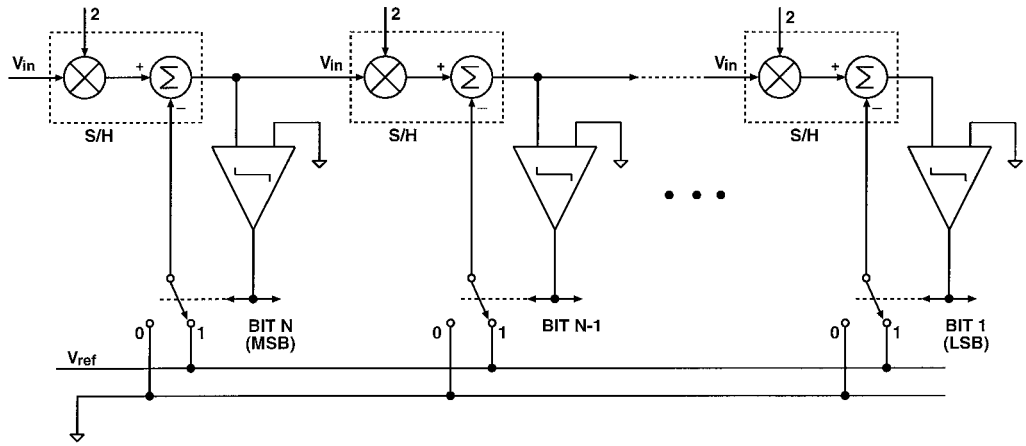


FIGURE 54.18 One-bit per stage pipeline ADC architecture.

of the capacitors is connected to the output of the op-amp for feedback, but the other is connected to $\pm V_{ref}$. Then, the output of the op-amp will be $2V_{in} - V_{ref}$ and $2V_{in} + V_{ref}$, respectively, after the op-amp settles.

However, this simple one-bit pipeline ADC is of no use if the comparator resolution is limited. If any redundancy is used for digital correction, at least two bits should be resolved. A close look at Fig. 54.16(b) gives a clue to using the simple functional block shown in Fig. 54.19 for the 2-b residue amplification. The case explained in Fig. 54.16(b) is sometimes called 1.5-b rather than 2-b because it needs only three

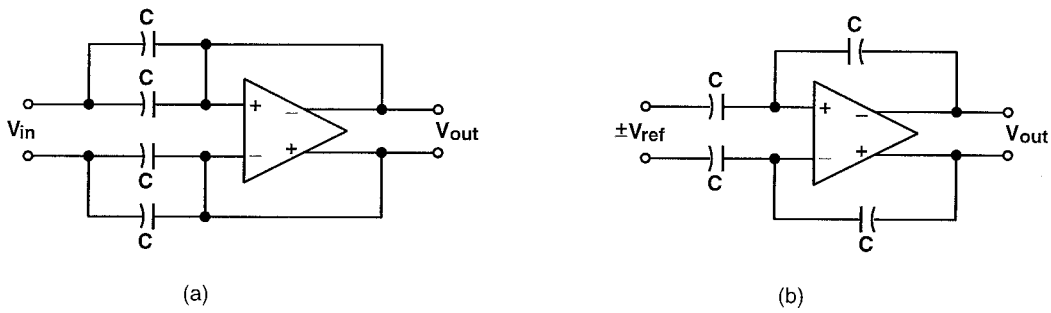


FIGURE 54.19 The simplest two-level MDAC: (a) sampling phase and (b) amplification phase.

DAC levels rather than four. The functional block in Fig. 54.19 can have a two-level DAC subtracting $\pm V_{ref}$. However, in differential architecture, by shorting the input, one midpoint can be interpolated. Using the tri-level DAC, the 1.5-b pipeline ADC can be implemented with the following algorithm.[12] If the input V_{in} is lower than $-V_{ref}/4$, the residue output is $2V_{in} + V_{ref}$. If the input is higher than $V_{ref}/4$, the residue output is $2V_{in} - V_{ref}$. If the input is in the middle, the output is $2V_{in}$.

Algorithmic, Cyclic, or Recursive ADC

The interstage S/H used in the multi-step architecture provides a flexibility of the pipeline architecture. In the pipeline structure, the same hardware repeats as shown in Fig. 54.13. That is, the throughput rate of the pipeline is fast while the overall latency is limited by the number of stages. Instead of repeating the hardware, using the same stage repeatedly greatly saves hardware, as shown in Fig. 54.20. That is, the throughput rate of the pipeline is directly traded for hardware simplicity. Such a converter is called an *algorithmic*, *cyclic*, or *recursive ADC*. The functional blocks used for the algorithmic ADC are identical to the ones used in the pipeline ADC.

Time-Interleaved Parallel ADC

The algorithmic ADC just described sacrifices the throughput rate for small hardware. However, the *time-interleaved parallel ADC* takes quite the opposite direction. It duplicates more hardware in parallel for higher throughput rates. The system is shown in Fig. 54.21, where the throughput rate increases by the number of parallel paths strobed. Although it significantly improves the throughput rate and many refinements have been reported, it suffers from many problems.[13] Due to the multiplexing, even static nonlinearity mismatch between paths appears as a fixed pattern noise. Also, it is difficult to generate clocks with exact delays, and inaccurate clocking increases the noise floor.

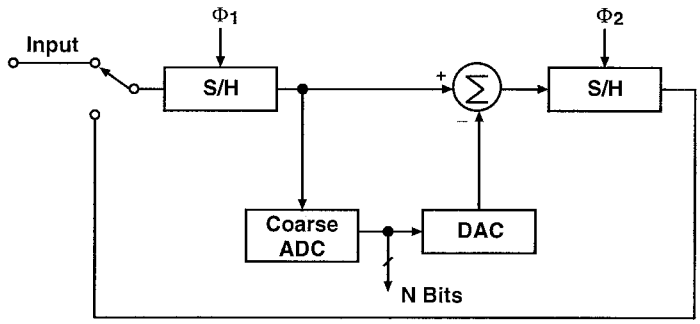


FIGURE 54.20 Algorithmic, cyclic, or recursive ADC architecture.

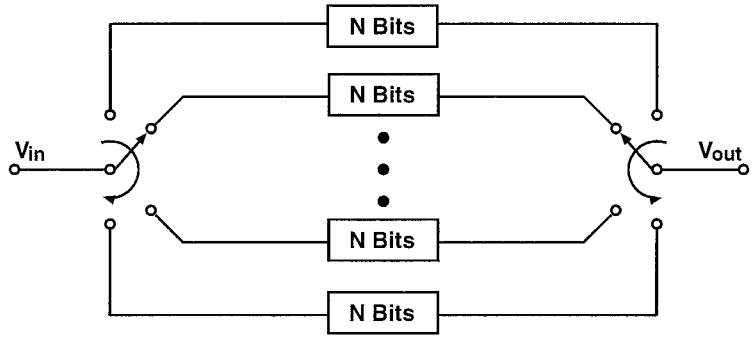


FIGURE 54.21 Time-interleaved parallel ADC architecture.

Folding ADC

The folding ADC is similar to the flash ADC except for using fewer comparators. This reduction in the number of comparators is achieved by replacing the comparator preamplifiers with folding amplifiers. In its original arrangements,[14] the folding ADC digitizes the folded signal with a flash ADC. The folded signal is equivalent in concept to the residue of the subranging, multi-step, or pipeline ADC, but the difference is that the generation of the folding signal is done solely in the analog domain. Since the digitized code from the folding amplifier output repeats over the whole input range, a coarse coding is required, as in all subranging-type ADCs.

Consider a system configured as a 4-b folding ADC as shown in Fig. 54.22. Four folding amplifiers can be placed in parallel to produce four folded signals. Comparators check the outputs of the folding amplifiers for zero crossing. If the input is swept, the outputs of the fine comparators show a repeating pattern, and eight different codes can be obtained by the four comparators. Because there are two identical fine code patterns, one comparator is needed to distinguish them. However, if this coarse comparator is misaligned with the fine quantizer, missing codes will result. A digital correction similar to that for the multi-step or pipeline ADC can be employed to correct the coarse quantizer error. For this system example, one-bit redundancy is used by adding two extra comparators in the coarse quantizer. The shaded region in the figure is where errors occur.

Having several folded signals instead of one has many advantages in designing high-speed ADCs. The folding amplifier requires neither linear output nor accurate settling. This is because in the folding ADC, the zero-crossings of the folded signals matter, but not their absolute values. Therefore, the offset of the folding amplifiers becomes the most important design issue. The resolution of the folding ADC can be further improved using the interpolation concept. When the adjacent folded signals are interpolated by I times, the number of zero-crossing points are also increased by I times. So, the resolution of the final ADC is improved by $\log_2 I$ bits. The higher bound for the degree of interpolation is set by the comparator resolution, the gain of the folding amplifiers, the linearity of folded signals, and the interpolation accuracy. Since the folding process increases the internal signal bandwidth by the number of foldings, the folding ADC performance is limited by the folding amplifier bandwidth. To increase the number of foldings while maintaining the reasonable comparator resolution, the folding amplifier's gain should be high. Since the higher gain limits the amplifier bandwidth, it is necessary to cascade the folding stages.[6,8]

54.4 ADC Design Considerations

In general, multi-step ADCs are made of cascaded low-resolution ADCs. Each low-resolution ADC stage provides a residue voltage for the subsequent stage, and the accuracy of the residue voltage limits the resolution of the converter. One of the residue amplifiers commonly used in CMOS is a switched-capacitor MDAC, whose connections during two clock phases are illustrated in Fig. 54.23 for an N-bit case. An

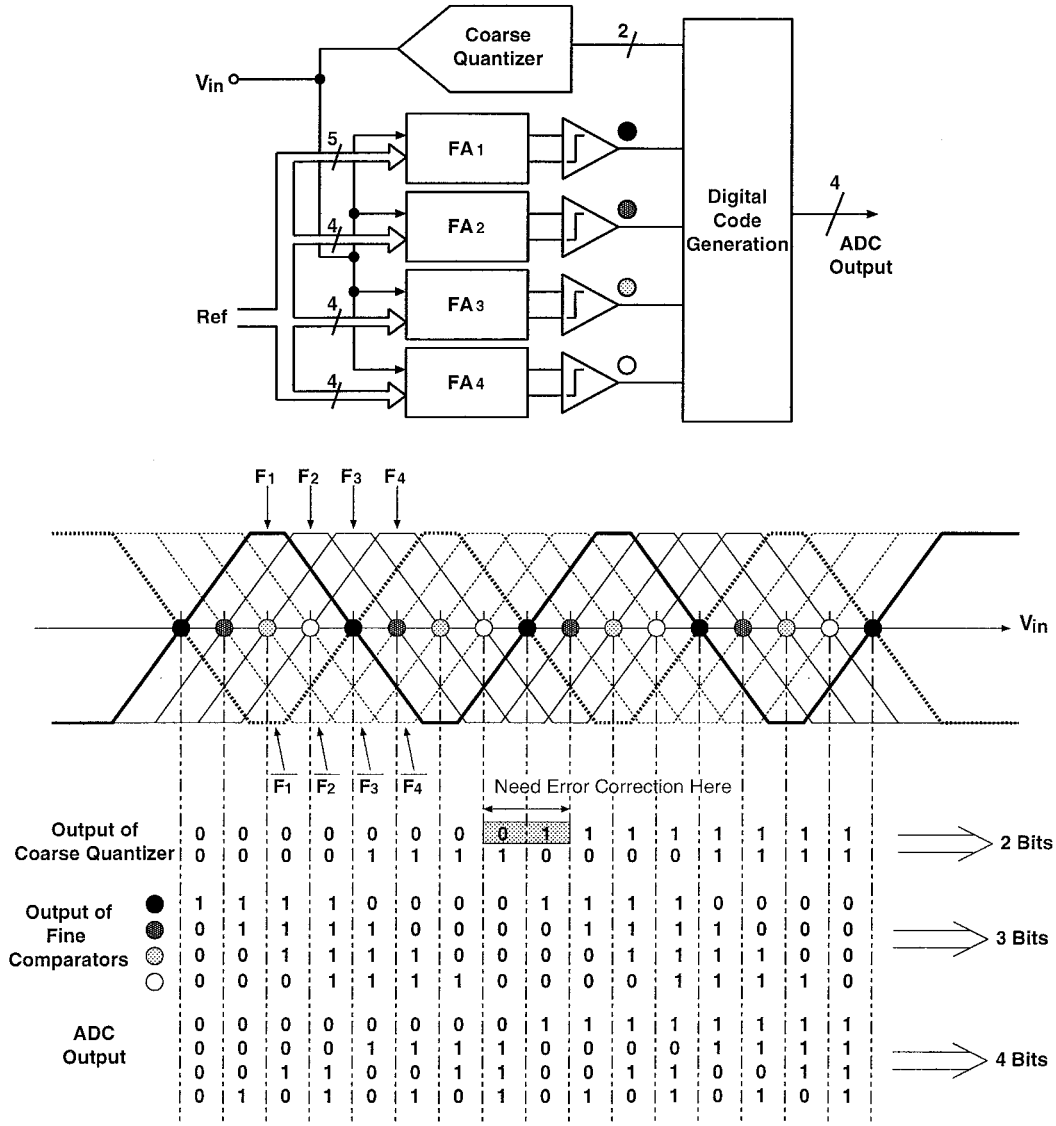


FIGURE 54.22 A 4-b folding ADC example with digital correction.

extra capacitor C is usually added to double the feedback capacitor size so that the residue voltage may remain within the full-scale range for digital correction.

Sampling Error Considerations

Since the ADC works on a sampled signal, the accuracy in sampling fundamentally limits the system performance. It is well known that the noise power to be sampled on a capacitor along with the signal is KT/C. It is inversely proportional to the sampling capacitor size. The sampled rms voltage noise is 64 μV with 1 pF, but decreases to 20 μV with 10 pF. For accurate sampling, sampling capacitors should be large, but sampling on large capacitors takes time. The speed of the ADC is fundamentally limited by the sampling KT/C noise.

In sampling, there exists another important error source. Direct sampling on a capacitor suffers from switch feedthrough error due to the charge injection when switches are opened. A common way to reduce

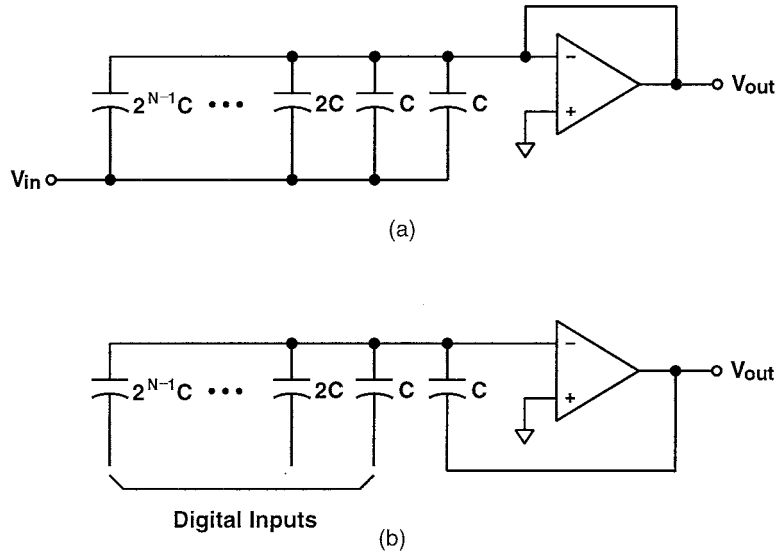


FIGURE 54.23 General N-bit residue amplifier: (a) sampling phase and (b) amplification phase.

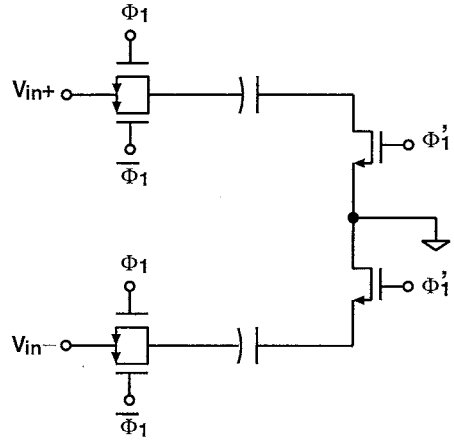


FIGURE 54.24 Open-loop bottom-plate differential sampling on capacitors.

this charge feedthrough error is to turn off the switches connected to the sampling capacitor top plate slightly earlier than the switches connected to the bottom plate. This is explained in Fig. 54.24. Usually, the top plate is connected to the op-amp summing or comparator input node. The top plate switch is switched with one MOS transistor with a clock phase marked as Φ_1' which makes a falling transition earlier than other clocks. The bottom plate is switched with a CMOS switch (both NMOS and PMOS) with clocks marked as Φ_1 and $\overline{\Phi_1}$. These clocks make falling transitions after the prime clock does. The net effect is that the feedthrough voltage stays constant because the top plate samples the same voltage repeatedly. The differential sampling using two capacitors symmetrically is known to provide the most accurate sampling known to date.

Unless limited by speed, the sampling error as well as the low-end spectrum of the sampled noise can be eliminated using a correlated double sampling (CDS) technique. The system has been used to remove the flicker noise or slowly-varying offset such as in charge-coupled device (CCD). The CDS needs two sampling clocks. The idea is to subtract the previously sampled sampling error from the new sample

TABLE 54.1 Three Dominant ADC Architectures

	Interpolated Flash	Multi-step	Pipeline
Matching	Least	Medium	Most critical
Feedthrough	Most critical	Medium	Least
Bandwidth	Least	Most critical	Medium
Settling	Least	Medium	Most critical
Gain	Least	Medium	Most critical
Speed	Fast	Slow	Medium
Complexity	Complex	Medium	Simple
Problems	Clock jitter	Low loop gain	Matching
	Time skew		Wide bandwidth
	Sampling error		High gain

after one clock delay. The result is to null the sampling error spectrum at every multiple of the sampling frequency f_s. The CDS is effective only for the low-frequency spectrum.

Techniques for High-Resolution and High-Speed ADCs

Considering typical requirements, three representative ADC architectures are compared in Table 54.1. To date, all techniques known to improve ADC resolution are as follows: trimming, dynamic matching, ratio-independent technique, capacitor-error averaging, walking reference, and self-calibration. However, the trimming is irreversible and expensive. It is only possible at the factory or with precision equipments. The dynamic matching technique is effective, but it generates high-frequency noise. The ratio-independent techniques either require many clock cycles or are limited to improve differential linearity. The latter case is good for monotonicity, but it also requires accurate comparison. The capacitor-error averaging technique requires three clock cycles, and the walking reference is sensitive to clock sampling error. The self-calibration technique requires extra hardware for calibration and digital storage, but its compatibility with existing proven architectures may provide potential solutions both for high resolution and for high speed.

The ADC self-calibration concepts originated from the direct code-mapping concept using memory. The calibration is to predistort the digital input to the DAC so that the DAC output can match the ideal level from the calibration equipment. Due to the precision equipment needed, this method has limited use. The first self-calibration concept applied to the binary-ratioed successive-approximation ADC is to internally measure capacitor DAC ratio errors using a resistor-string calibration DAC as shown in Fig. 54.25.[15] Later, an improved concept of the digital-domain calibration was developed for the multi-step or pipeline ADCs.[16]

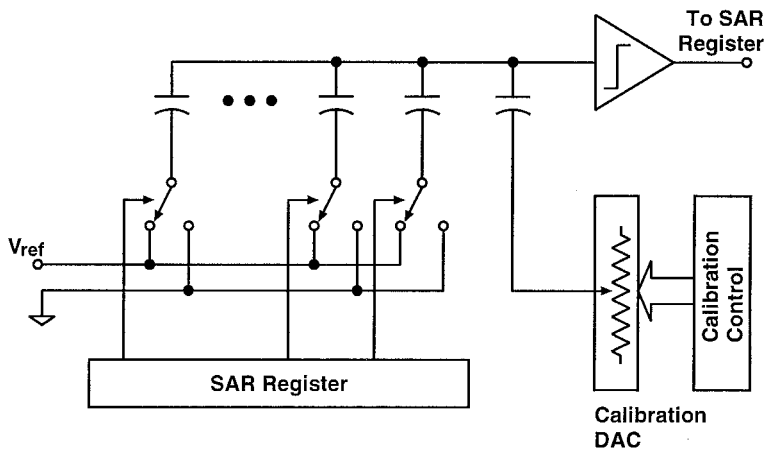


FIGURE 54.25 Self-calibrated successive-approximation ADC.

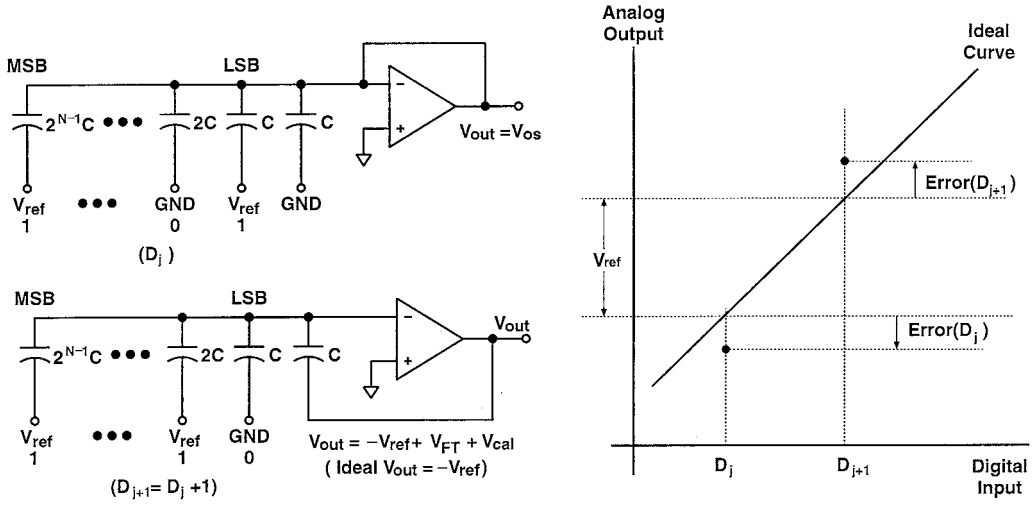


FIGURE 54.26 Segment-error measuring technique for digital self-calibration.

The general concept of the digital-domain calibration is to measure code or bit errors, to store them in the memory, and to subtract them during the normal operation. The concept is explained in Fig. 54.26 using a generalized N-bit MDAC with a capacitor array. If the DAC code increases by 1, the MDAC output should increase by V_{ref} or $V_{ref}/2$ with digital correction. Any deviation from this ideal step is defined as a segment error. Code errors are obtained by accumulating segment errors. This segment-error measurement needs two cycles. The first cycle is to measure the switch feedthrough error, and the next cycle is to measure the segment error. The segment error is simply measured as shown in Fig. 54.26 using the LSB-side of the ADC by increasing the digital code by 1. In the case of N = 1, the segment error becomes a bit error. If binary bit errors are measured and stored, code errors should be calculated for subtraction during the normal operation. How to store DAC errors is a tradeoff issue. Examples of the digital calibration are well documented for the cases of segment-error[17] and bit-error[18] measurements, respectively.

54.5 DAC Design Arts

There are many different circuit techniques used to implement DACs, but the popular ones widely used today are of the parallel type in which all bits change simultaneously upon the application of an input code word. Serial DACs, on the other hand, produce an analog output only after receiving all digital input data in a sequential form. When DACs are used as stand-alone devices, their output transient behaviors limited by glitch, slew rate, word clock jitter, settling, etc. are of paramount importance, but used as subblocks of ADCs, DACs need only to settle within a given time interval. An output S/H, usually called a *deglitcher*, is often used for better transient performance. Three of the most popular architectures of DACs are resistor string, ratioed current sources, and a capacitor array. The current-ratioed DAC finds most applications as a stand-alone DAC, while the resistor-string and capacitor-array DACs are mainly used as ADC subblocks.

For speeds over 100 MHz, most state-of-the-art DACs employ current sources switched directly to output resistors.[19-23] Furthermore, owing to the high bit counts (12 to 16 b), segmented architectures are employed, with the current sources broken into two or three segments. The CMOS design has the advantages of lower power, smaller area, and lower manufacturing costs. In all cases, it is of interest to note that the dynamic performance of the DACs degrades rapidly as input frequencies increase, and true dynamic performance is not attained except at low frequencies. Since a major application of wide-bandwidth, high-resolution DACs is in communications, poor dynamic performance is undesirable,

owing to the noise leakage from frequency multiplexed channels into other channels. The goal of better dynamic performance continues to be a target of ongoing research and development.

54.6 DAC Architectures

An N-bit DAC provides a discrete analog output level, either voltage or current, for every level of 2^N digital words that is applied to the input. Therefore, an ideal voltage DAC generates 2^N discrete analog output voltages for digital inputs varying from 000…00 to 111…11. In the unipolar case, the reference point is 0 when the digital input is 000…00; but in bipolar or differential DACs, the reference point is the midpoint of the full scale when the digital input is 100…00. Although purely current-output DACs are possible, voltage-output DACs are common in most applications.

Resistor-String DAC

The simplest voltage divider is a resistor string. Reference levels can be generated by connecting 2^N identical resistors in series between V_{ref} and ground. Switches to connect the divided reference voltages to the output can be either 1-out-of-2^N decoder or binary tree decoder as shown in Fig. 54.27 for the 3-b example. Since it requires a good switch, the stand-alone resistor-string DAC is easier to implement using CMOS. However, the lack of switches does not limit the application of the resistor string as a voltage reference divider subblock for ADCs in other process technologies.

Resistor strings are widely used as an integral part of the flash ADC as a reference divider. All resistor-string DACs are inherently monotonic and exhibit good differential linearity. However, they suffer from poor integral linearity and also have the drawback that the output resistance depends on the digital input

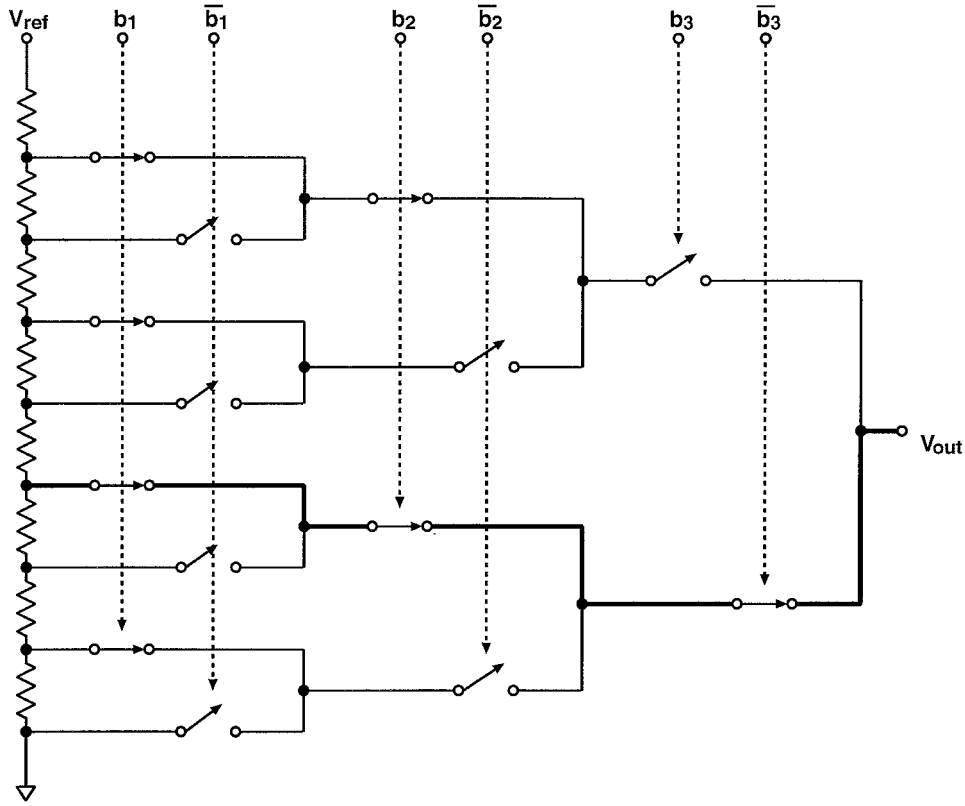


FIGURE 54.27 Resistor-string DAC.

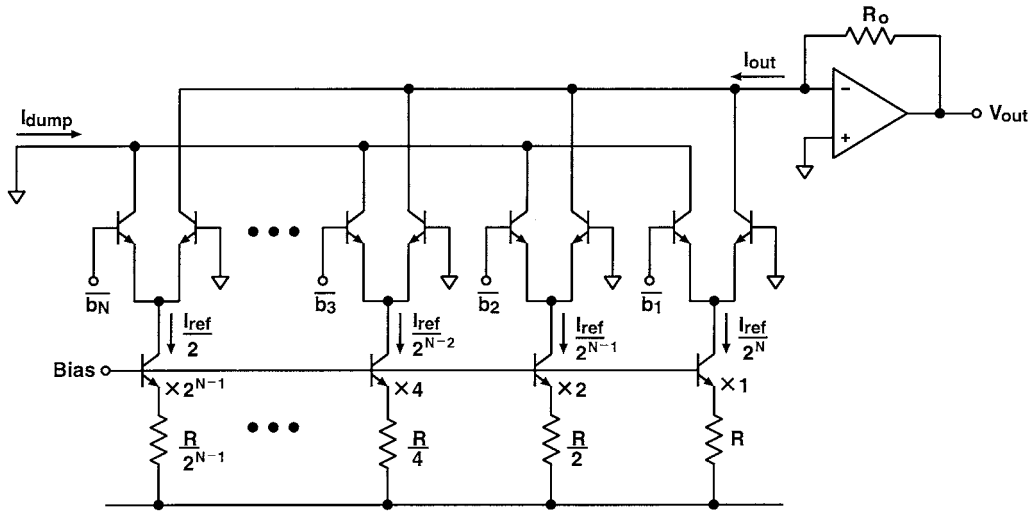


FIGURE 54.28 Current-ratioed DAC.

code. This causes a code-dependent settling time when charging the capacitive load. This non-uniform settling time problem can be alleviated by adding low-resistance parallel resistors or by compensating the MOS switch overdrive voltages.

Current-Ratioed DAC

The most popular stand-alone DACs in use today are *current-ratioed DACs*. There are two types: one is a weighted-current DAC and the other is an R-2R DAC. The weighted-current DAC shown in Fig. 54.28 is made of an array of switched binary-weighted current sources and the current summing network. In bipolar technology, the binary weighting is achieved by ratioed transistors and emitter resistors with binary related values of R, R/2, R/4, etc., while in MOS technology, only ratioed transistors are used. DACs relying on active device matching can achieve an 8b-level performance with a 0.2 to 0.5% matching accuracy using a 10- to 20-μm device feature size, while degeneration with thin-film resistors gives a 10b-level performance. The current sources are switched on or off by means of switching diodes or emitter-coupled differential pairs (source-coupled pairs in CMOS). The output current summing is done by a wideband transresistance amplifier; but in high-speed DACs, the output current directly drives a resistor load for maximum speed. The weighted-current design has the advantage of simplicity and high speed, but it is difficult to implement a high-resolution DAC because a wide range of emitter resistors and transistor sizes are used, and very large resistors cause problems with both temperature stability and speed.

R-2R Ladder DAC

This large resistor ratio problem is alleviated by using a resistor divider known as an *R-2R ladder*, as shown in Fig. 54.29. The R-2R network consists of series resistors of value R and shunt resistors of value 2R. The top of each shunt resistor of value 2R has a single-pole double-throw electronic switch that connects the resistor either to ground or to the current summing node. The operation of the R-2R ladder network is based on the binary division of current as it flows down the ladder. At any junction of series resistor of value R, the resistance looking to the right side is 2R. Therefore, the input resistance at any junction is R, and the current splits into two equal parts at the junction since it sees equal resistances in both directions. As a result, binary-weighted currents flow into shunt resistors in the ladder. The digitally controlled switches direct the currents to either ground or to the summing node. The advantage of the

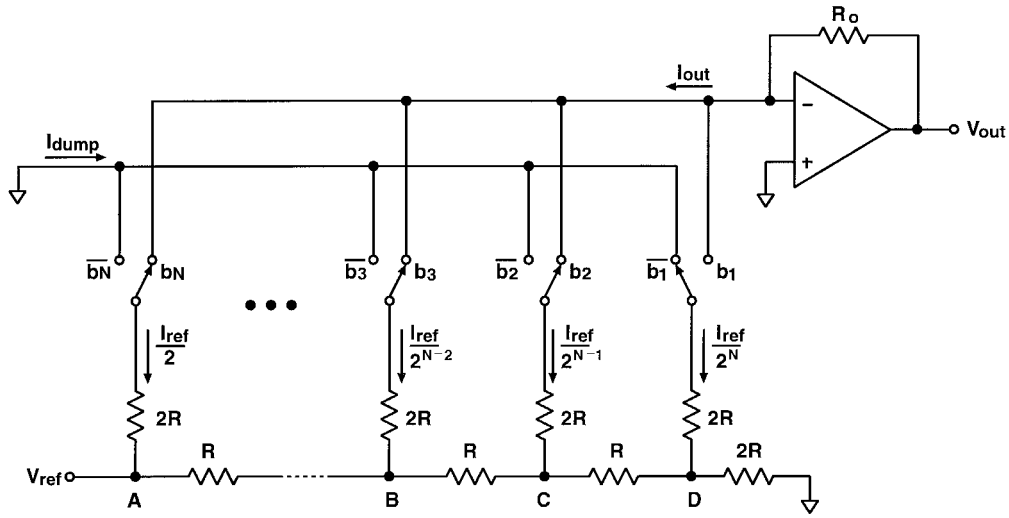


FIGURE 54.29 R-2R DAC.

R-2R ladder method is that only two values of resistors are used, greatly simplifying the task of matching or trimming and temperature tracking. In addition, for high-speed applications, relatively low resistor values can be used. Excellent results can be obtained using laser-trimmed thin-film resistor networks. Since the output of the R-2R DAC is the product of the reference voltage and the digital input word, the R-2R ladder DAC is often called an MDAC.

Capacitor-Array DAC

Capacitors made of double-poly or poly-diffusion in MOS technology are considered one of the most accurate passive components comparable to thin-film resistors in the bipolar process, both in the matching accuracy and voltage and temperature coefficients.[1] The only disadvantage in the capacitor-array DAC implementation is the use of a dynamic charge redistribution principle. A switched-capacitor counterpart of the resistor-string DAC is a parallel capacitor array of 2^N unit capacitors with a common top plate. The capacitor-array DAC is not appropriate for stand-alone applications without a feedback amplifier virtually grounding the top plate and an output S/H or deglitcher. The operation of the capacitor-array DAC shown in Fig. 54.30(a) is based on the thermometer-coded DAC principle and has the distinct advantage of monotonicity. However, due to the complexity of handling the thermometer-coded capacitor array, a binary-weighted capacitor array is often used, as shown in Fig. 54.30(b) by grouping unit capacitors in binary ratio values. One important application of the capacitor-array DAC is as a reference DAC for ADCs. As in the case of the R-2R MDAC, the capacitor-array DAC can be used as an MDAC to amplify residue voltages for multi-step or pipeline ADCs.

Thermometer-Coded Segmented DAC

Applying a two-step conversion concept, a DAC can be made in two levels using coarse and fine DACs. The fine DAC divides one coarse MSB segment into fine LSBs. If one fixed MSB segment is subdivided to generate LSBs, matching among MSB segments creates a non-monotonicity problem. However, if the next MSB segment is subdivided instead of the fixed segment, the segmented DAC can maintain monotonicity regardless of the MSB matching. This is called the next-segment approach. The most widely used segmented DAC is a current-ratioed DAC, whose MSB DAC is made of identical elements for the next-segment approach, except that the LSB DAC is a current divider as shown in Fig. 54.31. To implement a segmented DAC using two resistor-string DACs, voltage buffers are needed to drive the LSB DAC

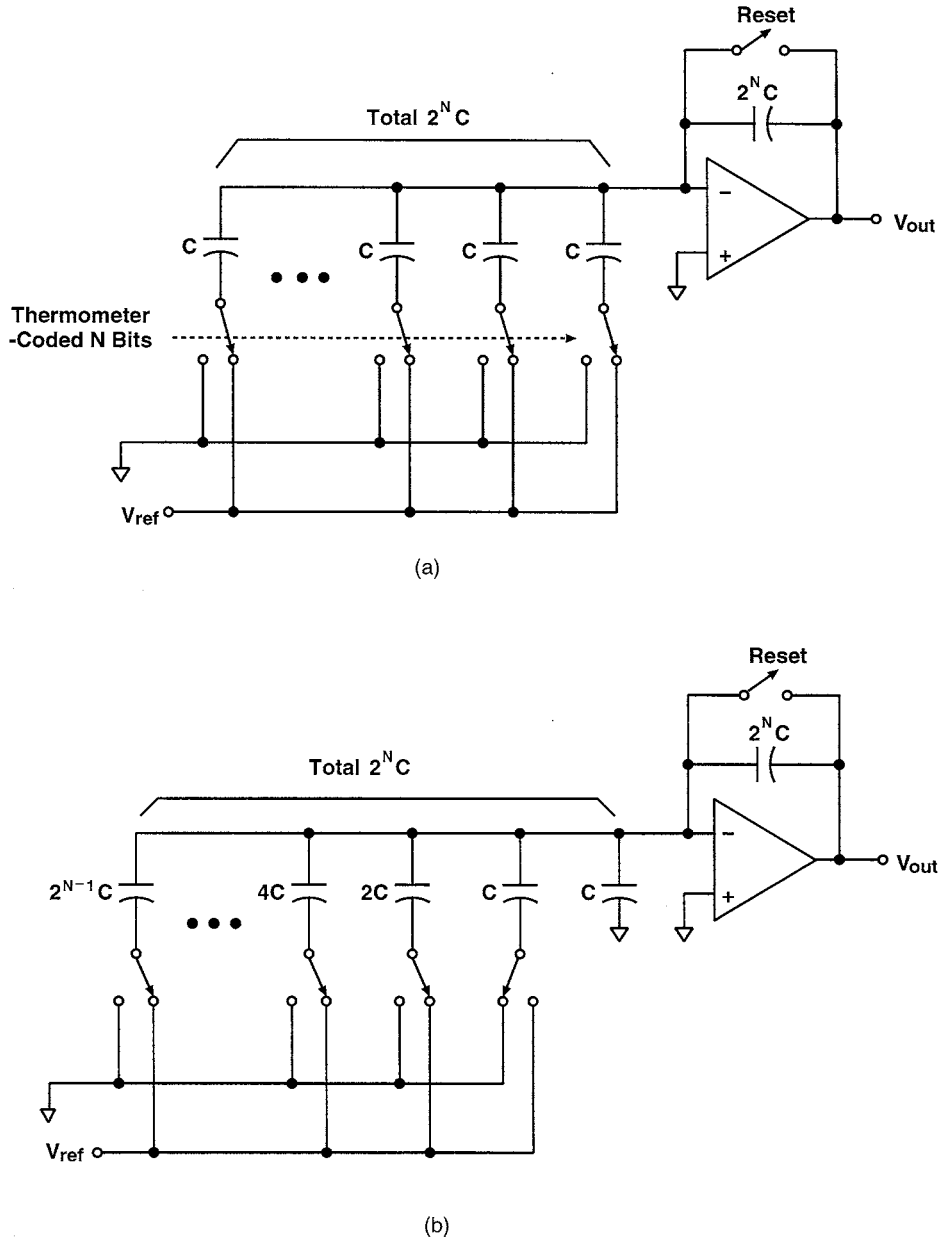


FIGURE 54.30 Capacitor-array DACs: (a) thermometer-coded and (b) binary-weighted.

without loading the MSB DAC. Although the resistor-string MSB DAC is monotonic, overall monotonicity is not guaranteed due to the offsets of the voltage buffers. The use of a capacitor-array LSB DAC eliminates the need for voltage buffers.

Integrator-Type DAC

As mentioned, monotonicity is guaranteed only in a thermometer-coded DAC. The thermometer coding of a DAC output can be implemented either by repeating identical DAC elements many times or by using

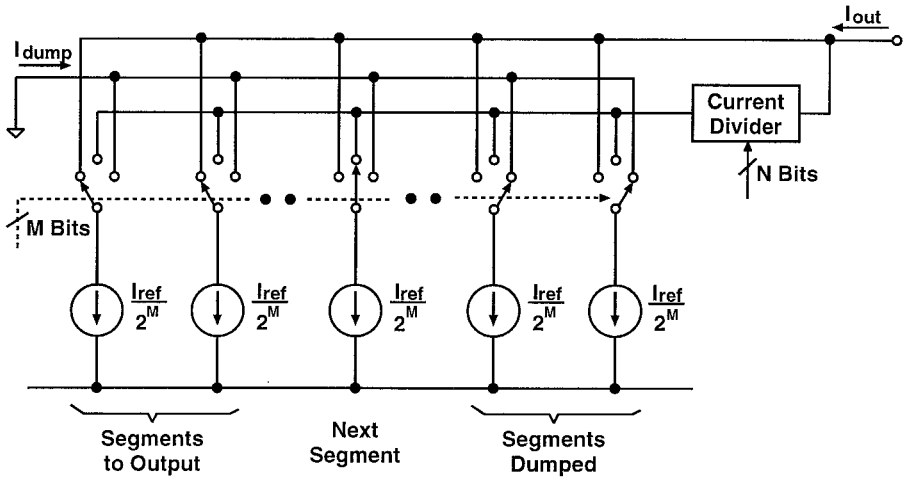


FIGURE 54.31 Thermometer-coded segmented DAC.

the same element over and over. The former requires more hardware, but the latter requires more time. In the continuous-time integrator-type DAC, the integrator output is a linear ramp and the time to stop integration can be controlled digitally. Therefore, monotonicity can be maintained. Similarly, the discrete-time integrator can integrate a constant amount of charge repeatedly and the number of integrations can be controlled digitally. The integration approach can give high accuracy, but its disadvantage is that its slow speed limits its applications.

54.7 DAC Design Considerations

Figure 54.32 illustrates two step responses of a DAC when it settles with a time constant τ and when it slews with a slew rate S. The transient errors given by the shaded areas are h/τ and $h^2/2S$, respectively. This implies that a single time-constant settling of the former case only generates a linear error in the output, which does not affect the DAC linearity, but the slew-limited settling generates a nonlinear error. Even in the single-time constant case, the code-dependent settling time constant can introduce a nonlinearity error because the settling error is a function of the time constant t. This is true for a resistor-string DAC, which exhibits a code-dependent settling time because the output resistance of the DAC depends on the digital input.

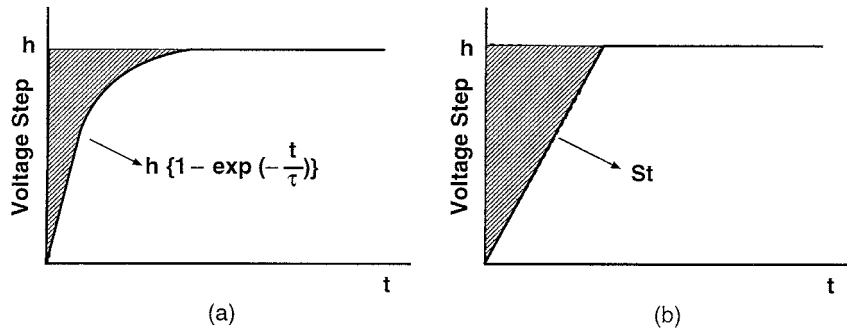


FIGURE 54.32 DAC settling cases: (a) exponential and (b) slew-limited case.

Effect of Limited Slew Rate

The slew-rate limit is a significant source of nonlinearity since the error is proportional to the square of the signal, as shown in Fig. 54.32(b). The height and width of the error term change with the input. The worst-case harmonic distortion (HD) when generating a sinusoidal signal with a magnitude V_o with a limited slew rate of S is[24]:

$$HD_k = 8\frac{\sin^2\frac{\omega T_c}{2}}{\pi k\left(k^2-4\right)} \times \frac{V_o}{ST_c}, \quad k = 1, 3, 5, 7\cdots \tag{54.6}$$

where T_c is the clock period. For a given distortion level, the minimum slew rate is given. Any exponential system with a bandwidth of ω_o gives rise to signals with the maximum slew rate of $2\omega_o V_o$. Therefore, by making $2\omega_o V_o > S$, the DAC system will exhibit no distortion due to the limited slew rate.

Glitch

Glitches are caused by small time differences between some current sources turning off and others turning on. Take, for example, the major code transition at half-scale from 011…11 to 100…00. Here, the MSB current source turns on while all other current sources turn off. The small difference in switching times results in a narrow half-scale glitch, as shown in Fig. 54.33. Such a glitch, for example, can produce distorted characters in CRT display applications. To alleviate both glitch and slew-rate problems related to transients, a DAC is followed by a deglitcher. The deglitcher stays in the hold mode while the DAC changes its output value. After the switching transients have settled, the deglitcher is changed to the sampling mode. By making the hold time suitably long, the output of the deglitcher can be made independent of the DAC transient response. However, the slew rate of the deglitcher is on the same order as that of the DAC, and the transient distortion will still be present — now as an artifact of the deglitcher.

Techniques for High-Resolution DACs

The following methods are often used to improve the linearity of DACs: Laser trimming, off-chip adjustment, common-centroid layout technique, dynamic element matching technique, voltage or current sampling, and electronic calibration techniques. The trend is toward more sophisticated and intelligent electronic solutions that overcome and compensate for some of the limitations of conventional trimming techniques. *Electronic calibration* is a general term to describe various circuit techniques, which

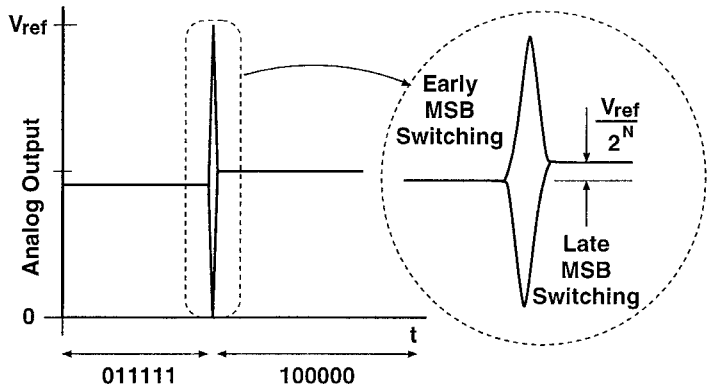


FIGURE 54.33 DAC output glitch.

usually predistort the DAC transfer characteristic so that the DAC linearity can be improved. The self-calibration is to incorporate all the calibration mechanisms and hardware on the DAC as a built-in function so that users can recalibrate whenever necessary.

The application of *dynamic element matching* to the binary-weighted current DAC is a straightforward switching of two complementary currents.[25] Its application to the binary voltage divider using two identical resistors or capacitors requires exchanging resistors or capacitors. This can be easily achieved by reversing the polarity of the reference voltage for the divide-by-two case. However, in the general case of N-element matching, the current division is inherently simpler than the voltage division. In general, to match the N independent elements, a switching network with N inputs and N outputs is required. The function of the switching network is to connect any input out of N inputs to one output with an average duty cycle of 1/N. The simplest one is a barrel shifter rotating the input-output connections in a predetermined manner. This barrel shifter generates a low-frequency modulated error when N gets larger because the same pattern repeats every N clocks. A more sophisticated randomizer with the same average duty cycle can distribute the mismatch error over the wider frequency range.

The *voltage or current sampling concept* is an electronic alternative to direct mechanical trimming. The voltage sampler is usually called a S/H, while the current sampler is called a current copier. The voltage is usually sampled on the input capacitor of a buffer amplifier, and the current is usually sampled on the input capacitor of a transconductance amplifier such as MOS transistor gate. Therefore, both voltage and current sampling techniques are ultimately limited by their sampling accuracy.

The idea behind the voltage or current sampling DAC is to use one voltage or current element repeatedly. One example of the voltage sampling DAC is a discrete-time integrating DAC. The integrator integrates a constant charge repeatedly, and its output is sampled. This is equivalent to generating equally spaced reference voltages by stacking identical unit voltages.[26] The fundamental problem associated with this sampling voltage DAC approach is the accumulation of the sampling error and noise in generating larger voltages. Similarly, the current sampling DAC can sample a constant current on current sources made of MOS transistors.[27] Since one reference current is copied on other identical current samplers, the matching accuracy can be maintained as long as the sampling errors are kept constant. Since it is not practical to make a high-resolution DAC using voltage or current sampling alone, this approach is limited to generating MSB DACs for the segmented DAC or for the subranging ADCs.

Self-calibration is based on an assumption that the segmented DAC linearity is limited by the MSB DAC so that only errors of MSBs can be measured, stored in memory, and recalled during normal operation. There are two different ways of measuring the MSB errors. In one method, individual-bit non-linearities, usually appearing as component mismatch errors, are measured digitally,[15,18] and a total error, which is called a code error, is computed from individual-bit errors depending on the output code during normal conversion. On the other hand, the other method measures and stores digital code errors directly and eliminates the digital code-error computation during normal operation.[16,17] The former requires less digital memory, while the latter requires fewer digital computations.

References

1. J. McCreary and P. Gray, All-MOS charge redistribution analog-to-digital conversion techniques-part I, *IEEE J. Solid-State Circuits*, vol. SC-10, pp. 371-379, Dec. 1975.
2. S. Ramet, A 13-bit 160kHz differential analog to digital converter, *ISSCC Dig. Tech. Papers*, pp. 20-21, Feb. 1989.
3. C. Mangelsdorf, H. Malik, S. Lee, S. Hisano, and M. Martin, A two-residue architecture for multistage ADCs, *ISSCC Dig. Tech. Papers*, pp. 64-65, Feb. 1993.
4. W. Song, H. Choi, S. Kwak, and B. Song, A 10-b 20-Msamples/s low power CMOS ADC, *IEEE J. Solid-State Circuits*, vol. 30, pp. 514-521, May 1995.
5. T. Cho and P. Gray, A 10-bit, 20-Msamples/s, 35-mW pipeline A/D converter, *IEEE J. Solid-State Circuits*, vol. 30, pp. 166-172, Mar. 1995.

6. P. Vorenkamp and R. Roovers, A 12-bits, 60MSPS cascaded folding & interpolating ADC, *IEEE J. Solid-State Circuits*, vol. 32, pp. 1876-1886, Dec. 1997.
7. K. Kattmann and J. Barrow, A technique for reducing differential nonlinearity errors in flash A/D converters, *ISSCC Dig. Tech. Papers*, pp. 170-171, Feb. 1991.
8. K. Bult and A. Buchwald, An embedded 240-mW 10-bit 50MS/s CMOS ADC in 1-mm^2, *IEEE J. Solid-State Circuits*, vol. 32, pp. 1887-1895, Dec. 1997.
9. B. Song, S. Lee, and M. Tompsett, A 10b 15-MHz CMOS recycling two-step A/D converter, *IEEE J. Solid-State Circuits*, vol. SC-25, pp. 1328-1338, Dec. 1990.
10. S. Lewis and P. Gray, A pipelined 5-Msamples/s 9-bit analog-to-digital converter, *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 954-961, Dec. 1987.
11. B. Song, M. Tompsett, and K. Lakshmikumar, A 12-bit 1-Msample/s capacitor error-averaging pipelined A/D converter, *IEEE J. Solid-State Circuits*, vol. SC-23, pp. 1324-1333, Dec. 1988.
12. S. Lewis, S. Fetterman, G. Gross Jr., R. Ramachandran, and T. Viswanathan, A 10-b 20-Msample/s analog-to-digital converter, *IEEE J. Solid-State Circuits*, vol. SC-27, pp. 351-358, Mar. 1992.
13. C. Conroy, D. Cline, and P. Gray, An 8-b 85-MS/s parallel pipeline A/D converter in 1-μm CMOS, *IEEE J. Solid-State Circuits*, vol. SC-28, pp. 447-454, Apr. 1993.
14. R. Plassche and R. Grift, A high speed 7 bit A/D converter, *IEEE J. Solid-State Circuits*, vol. SC-14, pp. 938 – 943, Dec. 1979.
15. H. Lee, D. Hodges, and P. Gray, A self-calibrating 15-bit CMOS A/D converter, *IEEE J. Solid-State Circuits*, vol. SC-19, pp. 813-819, Dec. 1984.
16. S. Lee and B. Song, Digital-domain calibration of multistep analog-to-digital converters, *IEEE J. Solid-State Circuits*, vol. SC-27, pp. 1679-1688, Dec.1992.
17. S. Kwak, B. Song, and K. Bacrania, A 15-b, 5-Msamples/s low spurious CMOS ADC, *IEEE J. Solid-State Circuits*, vol. 32, pp. 1866-1875, Dec. 1997.
18. A. Karanicolas, H. Lee, and K. Bacrania, A 15-b 1-Msample/s digitally self calibrated pipeline ADC, *IEEE J. Solid-State Circuits*, vol. 28, pp. 1207-1215, Dec. 1993.
19. D. Mercer, A 16-b D/A converter with increased spurious free dynamic range, *IEEE J. Solid-State Circuits*, vol. 29, pp. 1180-1185, Oct. 1994.
20. B. Tesch and J. Garcia, A low glitch 14-b 100-MHz D/A converter, *IEEE J. Solid-State Circuits*, vol. 32, pp. 1465-1469, Sept. 1997.
21. D. Mercer and L. Singer, 12-b 125 MSPS CMOS D/A designed for special performance, *Proc. IEEE Int. Symp. Low Power Electronics and Design*, pp. 243-246, Aug. 1996.
22. C. Lin and K. Bult, A 10b 250MSample/s CMOS DAC in 1mm^2, *ISSCC Dig. Tech. Papers*, pp. 214-215, Feb. 1998.
23. A. Marques, J. Bastos, A. Bosch, J. Vandenbusche, M. Steyaert, and W. Sansen, A 12b accuracy 300M sample/s update rate CMOS DAC, *ISSCC Dig. Tech. Papers*, pp. 216-217, Feb. 1998.
24. D. Freeman, Slewing distortion in digital-to-analog conversion, *J. Audio Eng. Soc.*, vol. 25, pp. 178-183, Apr. 1977.
25. R. Plassche, Dynamic element matching for high accuracy monolithic D/A converters, *IEEE J. Solid-State Circuits*, vol. SC-11, pp. 795-800, Dec. 1976.
26. D. Kerth, N. Sooch, and E. Swanson, A 12-bit 1-MHz two-step flash ADC, *IEEE J. Solid-State Circuits,* vol. SC-24, pp. 250-255, Apr. 1989
27. D. Groeneveld, H. Schouwenaars, H. Termeer, and C. Bastiaansen, A self-calibration technique for monolithic high-resolution D/A converters, *IEEE J. Solid-State Circuits*, vol. SC-24, pp. 1517-1522, Dec. 1989.

Fattaruso, J.W., Williams III, L.A.
"Oversampled Analog-to-Digital and Digital-to-Analog Converters"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

55

Oversampled Analog-to-Digital and Digital-to-Analog Converters

John W. Fattaruso
Louis A. Williams III
Texas Instruments, Incorporated

55.1 Introduction

In the absence of some form of calibration or trimming, the precision of the Nyquist rate converters described Chapter 54 is strictly dependent on the precision of the VLSI components that comprise the converter circuits. Oversampled data converters are a means of exchanging the speed and data processing capability of modern sub-micron integrated circuits for precision that would otherwise not be readily attainable.[1,2] The precision of an oversampled data converter can exceed the precision of its circuit components by several orders of magnitude.

In this chapter, the basic operation and design techniques of the most widely used class of oversampled data converters — *sigma-delta*[1] *modulators* — are described. In Section 55.2, the basic theory of sigma-delta

[1]The reader will find functionally identical modulator blocks in the literature named either "sigma-delta" modulators or "delta-sigma" modulators, with the choice of terminology largely up to personal preference. We use the former term here.

modulators is presented, using both time-domain and frequency-domain approaches. The issue of non-harmonic tones is also discussed. In Section 55.3, more complex sigma-delta architectures are described, including higher-order architectures, cascaded architectures, and bandpass architectures. Filtering techniques unique to sigma-delta modulators are presented in Section 55.4. In Section 55.5, the basic circuit building blocks for sigma-delta modulators are described; and in Section 55.6, circuit design issues specific to sigma-delta-based data converters are discussed.

55.2 Basic Theory of Operation

Oversampled data conversion techniques have their roots in the design of signal coders for communication systems.[3-6] Oversampling techniques differ from Nyquist techniques in that their comprehension and design procedures draw equally from time-domain and frequency-domain representations of signals, whereas Nyquist techniques are readily understood in just the time domain.

In general, the function of data conversion by oversampling is typically performed by a serial connection of a modulator and various filter blocks. In analog-to-digital (A/D) conversion, shown in Fig. 55.1, the analog input signal $x_i(t)$ is first bandlimited by an anti-alias filter, then sampled at a rate f_S. This sampling rate is M times faster than a comparable Nyquist rate converter with the same signal bandwidth; the value of M is the oversampling ratio. The sampled signal $x[n]$ is coded by a modulator block that quantizes the data into a finite number of discrete levels. The resulting coded signal $y[n]$ is down-sampled, or decimated, by a factor of M to produce an output that is comparable to a Nyquist rate converter. Digital-to-analog oversampled data conversion is basically the reverse of analog-to-digital conversion. As shown in Fig. 55.2, the Nyquist-rate digital samples are oversampled by an interpolation filter, coded by a modulator, and then reconstructed in the analog domain by an analog filter.

In both analog-to-digital and digital-to-analog data conversion, the block with the most unique signal processing properties is the modulator. The remainder of this section and the subsequent sections focus on the properties and architectures for oversampled data modulators.

Time-Domain Representation

The simplest modulator that would perform the requisite conversion to discrete output levels is the quantization function $Q(x)$ shown in Fig. 55.3. This quantization can be thought of as merely the sum

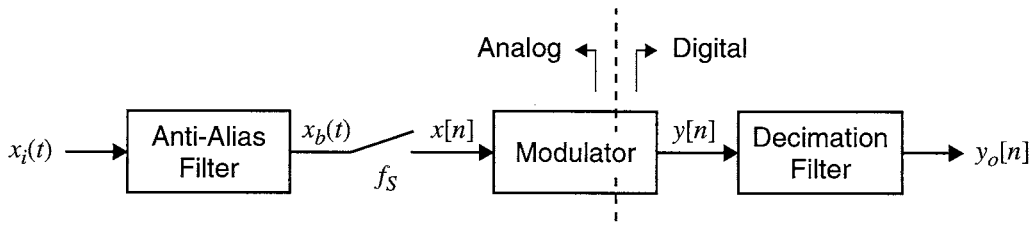


FIGURE 55.1 Oversampled A/D conversion.

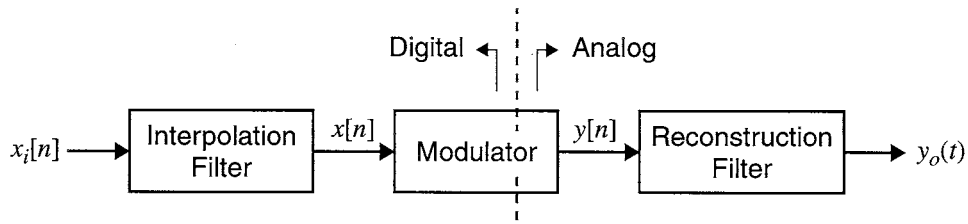


FIGURE 55.2 Oversampled D/A conversion.

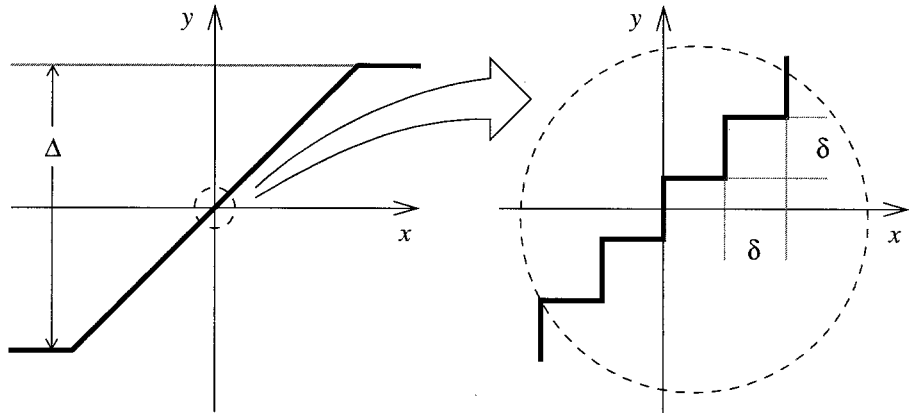


FIGURE 55.3 Quantizer transfer function.

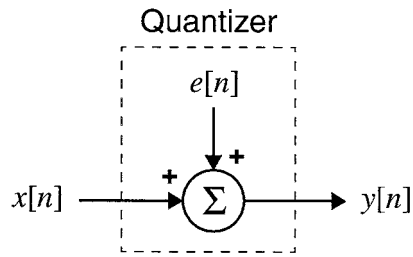


FIGURE 55.4 Quantization error.

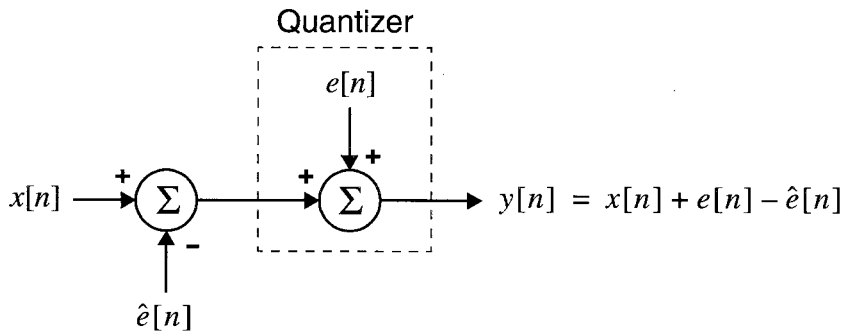


FIGURE 55.5 Error correction feedback.

of the original signal $x[n]$ with a sampled error signal $e[n]$, as illustrated in Fig. 55.4. In Nyquist rate converters, the error is reduced by using a large number of small steps in the quantizer characteristic. In oversampled data converters, specifically sigma-delta modulators, the error is corrected by a feedback network. This correction is made by estimating the error in advance and subtracting it from the input, as shown in Fig. 55.5, where is the error estimate. If this estimate were perfect, $\hat{e}[n]$ would equal $e[n]$ and the output $y[n]$ would equal the input $x[n]$. However, since the error is not known until it is made, $e[n]$ is not known when $\hat{e}[n]$ is needed. Therefore, some means must be found to estimate the error. In the case of sigma-delta converters, the error can be estimated by exploiting some knowledge of the frequency domain behavior of the input signal. Specifically, it is assumed that the signal is changing very slowly from sample to sample, or equivalently, its bandwidth is much less than the sampling rate.

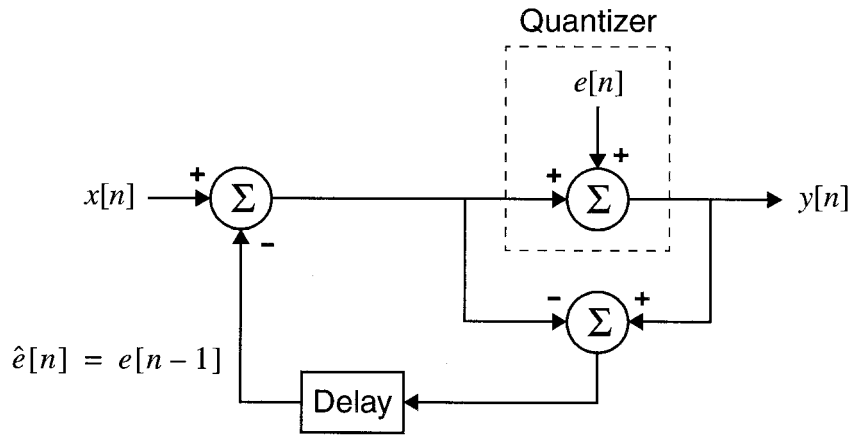


FIGURE 55.6 First-order error estimation.

For exceedingly slow signals, a first-order estimate of the error to be committed in quantization can be formed. The first-order estimate of the current error $e[n]$ is simply the previous error $e[n-1]$. This error may be found simply by a subtraction across the quantization block as shown in Fig. 55.6, and the output $y[n]$ is

$$y[n] = x[n] + e[n] - e[n-1] \tag{55.1}$$

The essential property of this structure is that if an error is committed that is not large enough to be corrected by a displacement to another quantizer level on the next sample, then the history of successive errors accumulate in the feedback loop until they eventually push the quantizer into another level. In this manner, the output of the quantizer will, over time, correct the errors committed in previous samples, increasing the precision of the information being generated as a time sequence of samples.

As will be shown in Section 55.5, the most convenient and accurate sampled-data circuit building block in practice is an integrator. With a few straightforward steps, the system of Fig. 55.6 can be transformed into that of Fig. 55.7, where the delay element is now immersed in an integrator feedback loop. The output of this transformed modulator is

$$y[n] = x[n-1] + e[n] - e[n-1] \tag{55.2}$$

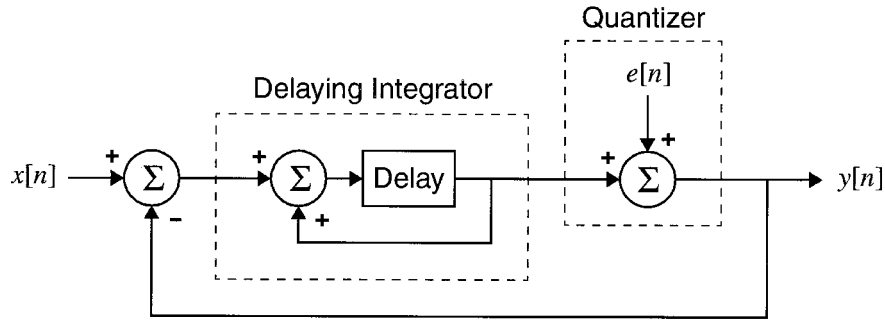


FIGURE 55.7 First-order equivalent modulator.

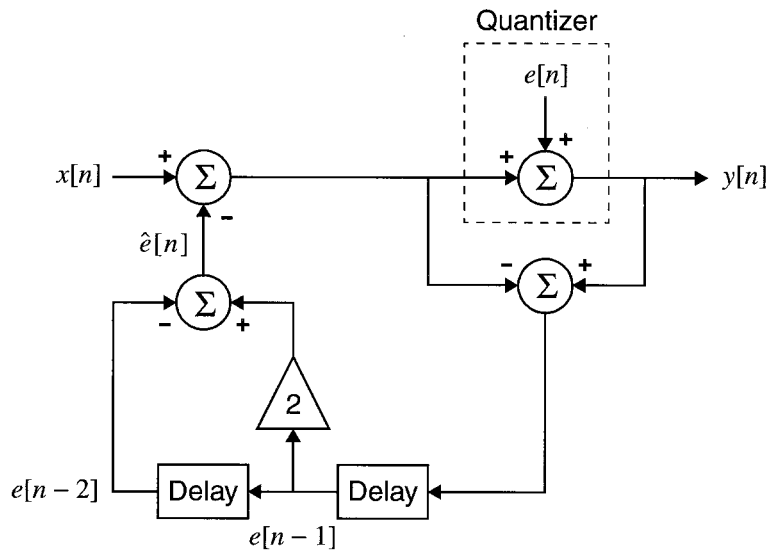


FIGURE 55.8 Second-order error estimation.

Comparing Eqs. 55.1 and 55.2, it is evident that this transformation does require the addition of a single clock delay block in the input path $x[n]$, but this extra clock cycle of latency has no effect on the precision or frequency response of the modulator. The structure in Fig. 55.7 is generally known as a first-order sigma-delta modulator.[7-9]

An increase in precision can be obtained by using more accurate estimates of the expected quantizer error.[6] A second-order estimate of $e[n]$ may be formed by assuming that the error $e[n]$ varies linearly with time. In this case, an estimate of the current error $e[n]$ may be computed by changing the previous error $e[n-1]$ by an amount equal to the change between $e[n-2]$ and $e[n-1]$. The second order error estimate is thus

$$\hat{e}[n] = e[n-1] + \left(e[n-1] - e[n-2]\right) = 2e[n-1] - e[n-2] \tag{55.3}$$

and is illustrated in Fig. 55.8. The output of the second-order estimation modulator is

$$y[n] = x[n] + e[n] - 2e[n-1] + e[n-2] \tag{55.4}$$

It can be shown, after a number of steps, that the modulator in Fig. 55.8 can be transformed into a modulator in which the feedback loop delays are again immersed in practical integrator blocks. This second-order sigma-delta modulator[10-12] is shown in Fig. 55.9; the output of this transformed modulator is

$$y[n] = x[n-2] + e[n] - 2e[n-1] + e[n-2] \tag{55.5}$$

which is entirely equivalent to that given by Eq. 55.4, except for the addition of two inconsequential delays of the input signal $x[n]$.

A further increase in precision can be obtained using even higher-order estimates of the quantizer error, such as quadratic or cubic. These high-order error estimate modulators can also be transformed

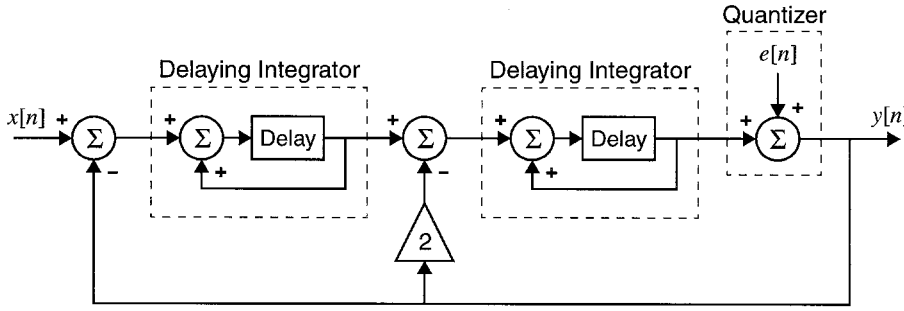


FIGURE 55.9 Second-order equivalent modulator.

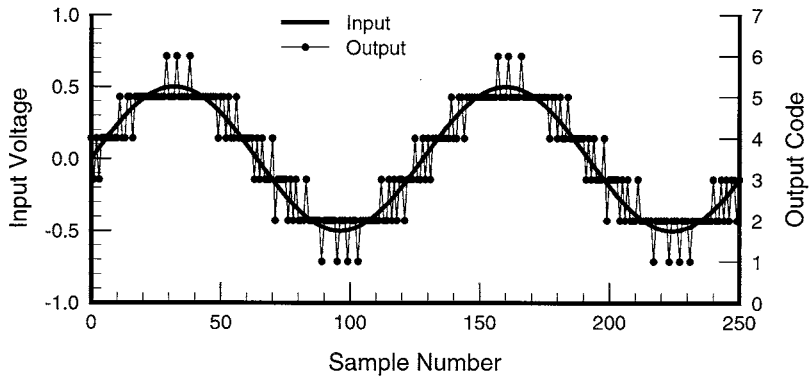


FIGURE 55.10 First-order sample output.

into a series of delaying integrators in a feedback loop. Unfortunately, as discussed in Section 55.3, practical difficulties emerge for orders greater than two, and alternative architectures are generally needed.

Computer simulation of modulator systems is straightforward, and Fig. 55.10 shows the simulated output of the first-order modulator of Fig. 55.7 when fed with a simple sinusoidal input. The resolution of the quantizer in the modulator loop was assumed to be eight levels. (The modulator output is drawn with continuous lines to emphasize the oscillatory nature of the modulator output, but the quantities plotted have meaning only at each sample time.) The coarsely quantized output code generally follows the input, but with occasional transitions that track intermediate values over local sample regions.

A second-order modulator with an eight-level quantizer exhibits the simulated behavior shown in Fig. 55.11. Note that the oscillations by which the loop attempts to minimize quantization error appear "busier" than in the first-order case of Fig. 55.10. It will be shown in the frequency domain that, for a given signal bandwidth, the more vibrant output code oscillations in Fig. 55.11 actually represent the input signal with higher precision than the first-order case in Fig. 55.10.

A special case that is of practical significance is the second-order modulator with a two-level quantizer, that is, simply a comparator closing the feedback loop. A simulation of such a modulator is shown in Fig. 55.12. Although the quantized representation at the output appears crude, the continuous nature of the input level is expressed in the density of output codes. When the input level is high (around sample numbers 32 and 160), there is a greater preponderance of '1' output codes; and at low swings of the input (around sample numbers 96 and 224), the '0' output code dominates.

The examples in Figs. 55.10 to 55.12 demonstrate that information generated from the modulator expresses, in a high-speed coded form, the coarsely quantized input signal and the deviation between the signal and the quantization levels. Although the time domain coded modulator output looks somewhat unintelligible, the output characteristics are clearer in the frequency domain.

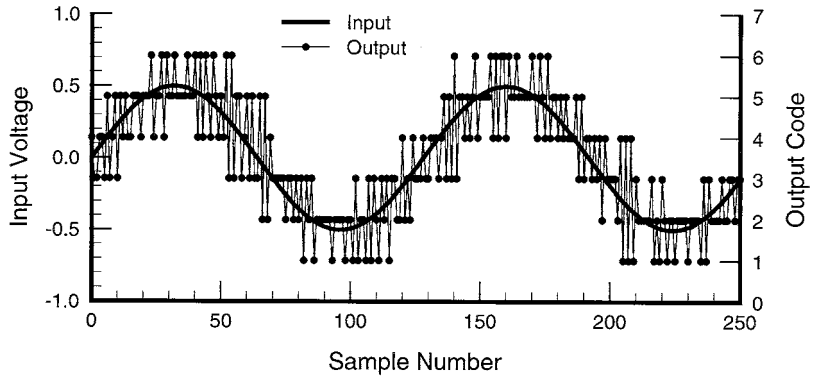


FIGURE 55.11 Second-order sample output.

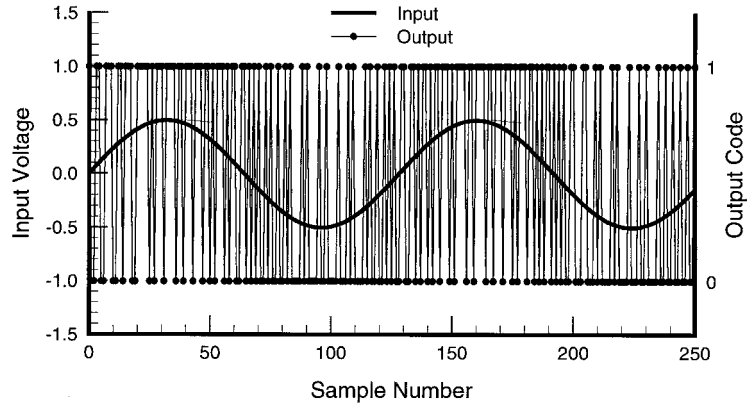


FIGURE 55.12 Second-order one-bit modulator sample output.

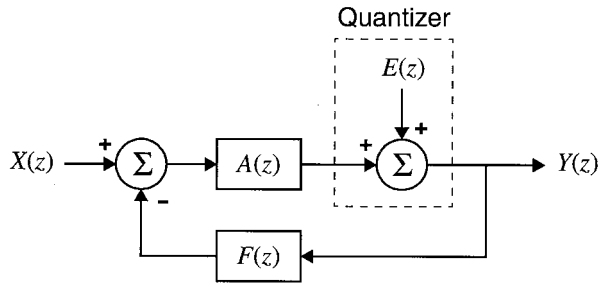


FIGURE 55.13 Generalized sigma-delta modulator.

Frequency-Domain Representation

The modulators in Figs. 55.7 and 55.9 can be generalized as the sampled-data system shown in Fig. 55.13, where the time domain signals $x[n]$, $y[n]$, and $e[n]$ are written as their frequency-domain equivalents $X(z)$, $Y(z)$, and $E(z)$. The modulator output in Fig. 55.13 can be written in terms of the input $X(z)$ and the quantizer error $E(z)$ as

$$Y(z) = H_x(z)X(z) + H_e(z)E(z) \tag{55.6}$$

where the input transfer function, $H_x(z)$, is

$$H_x(z) = \frac{A(z)}{1 + A(z)F(z)} \tag{55.7}$$

and the error transfer function, $H_e(z)$, is

$$H_e(z) = \frac{1}{1 + A(z)F(z)}. \tag{55.8}$$

Strictly speaking, the error $E(z)$ is directly dependent on the input $X(z)$. Nonetheless, if the input to the quantizer is sufficiently busy, that is, the input to the quantizer crosses through several quantization levels, the quantizer error approaches having the behavior of a random value that is uniformly distributed between $\pm\delta/2$ and is uncorrelated with the input, where δ is the quantization level separation illustrated in Fig. 55.3. In the frequency domain, the error noise power spectrum is uniform with a total error power of $\delta^2/12$.[13,14] The error power between the frequencies f_L and f_H is

$$S_{ee} \approx \frac{\delta^2}{6 f_S} \int_{f_L}^{f_H} \left| H_e\left(e^{j\pi f/f_s}\right) \right|^2 df \tag{55.9}$$

where f_S is the sampling rate. The error power between f_L and f_H can be reduced independent of the quantizer error level separation δ by having a small error transfer function $H_e(z)$ in that frequency band. Sigma-delta modulators have this property.

A sigma-delta modulator is a system such as that in Fig. 55.13 in which the error transfer function $H_e(z)$ is small and the input transfer function $H_x(z)$ is about unity for some band of frequencies. That is,

$$\left| H_e\left(e^{j2\pi f/f_s}\right) \right| \ll 1; \quad f_L \le f \le f_H \tag{55.10}$$

$$\left| H_x\left(e^{j2\pi f/f_s}\right) \right| \approx 1; \quad f_L \le f \le f_H \tag{55.11}$$

The requirements in Eqs. 55.10 and 55.11 are equivalent to requiring that the loop gain be large and the feedback gain be unity, that is

$$\left| A\left(e^{j2\pi f/f_s}\right) \right| \gg 1; \quad f_L \le f \le f_H \tag{55.12}$$

$$\left| F\left(e^{j2\pi f/f_s}\right) \right| \approx 1; \quad f_L \le f \le f_H \tag{55.13}$$

There are many system designs that have the sigma-delta properties of high loop gain and unity feedback gain. The previous examples in Figs. 55.7 and 55.9 are part of an important class of modulator architectures called *noise-differencing modulators* that are particularly well suited to VLSI implementation. The forward path in a noise-differencing sigma-delta modulator consists of a series of delaying integrators. The *order* of the modulator is defined as the number of integrators. The forward gain of an L-th order modulator is

$$A(z)=\left(\frac{z^{-1}}{1-z^{-1}}\right)^{L} \tag{55.14}$$

The feedback gain in a noise-differencing modulator is designed such that the modulator open-loop gain is

$$A(z)F(z)=\frac{1}{\left(1-z^{-1}\right)^{L}}-1 \tag{55.15}$$

From Eqs. 55.14 and 55.15, it follows that the output for an L-th order noise-differencing sigma-delta modulator is

$$Y(z)=z^{-L}X(z)+\left(1-z^{-1}\right)^{L}E(z) \tag{55.16}$$

The simulated frequency response for a second-order noise-differencing sigma-delta modulator with a sinusoidal input is shown in Fig. 55.14. The large spike in the center is the original input signal. It is clear from the plot that the noise energy is lowest at low frequencies. Noise-differencing modulators are designed to reduce the quantization noise in the baseband, that is, $f_L = 0$ and $f_H \ll f_S$. The oversampling ratio, M, is

$$M=\frac{f_S}{2f_H} \tag{55.17}$$

(In a Nyquist rate converter, $M = 1$.) The baseband noise power for a noise-differencing modulator is

$$S_{ee}\approx\frac{\delta^2}{6f_S}\int_0^{f_H}\left|\left(1-e^{-j2\pi f/f_S}\right)^{L}\right|^2 df\approx\frac{\delta^2}{12}\frac{\pi^{2L}}{2L+1}\frac{1}{M^{2L+1}} \tag{55.18}$$

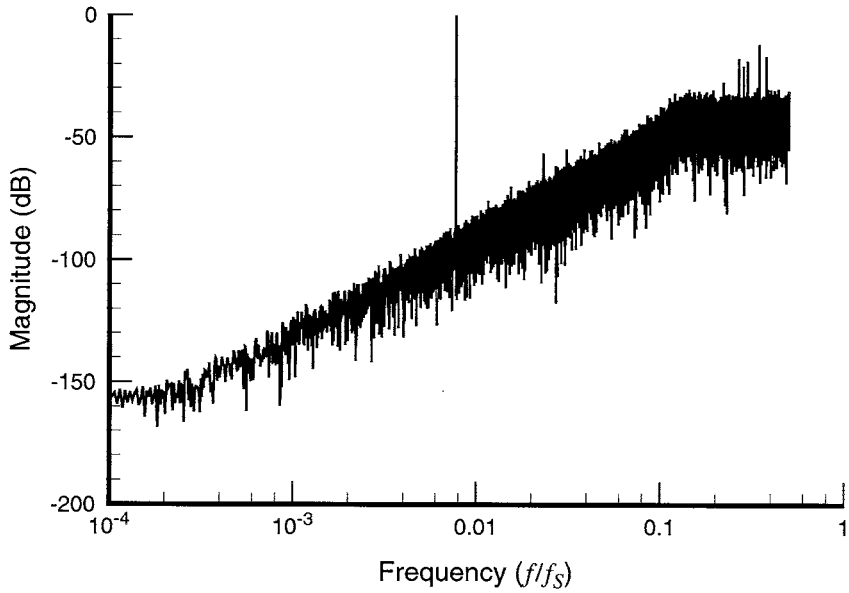


FIGURE 55.14 Second-order simulated frequency response.

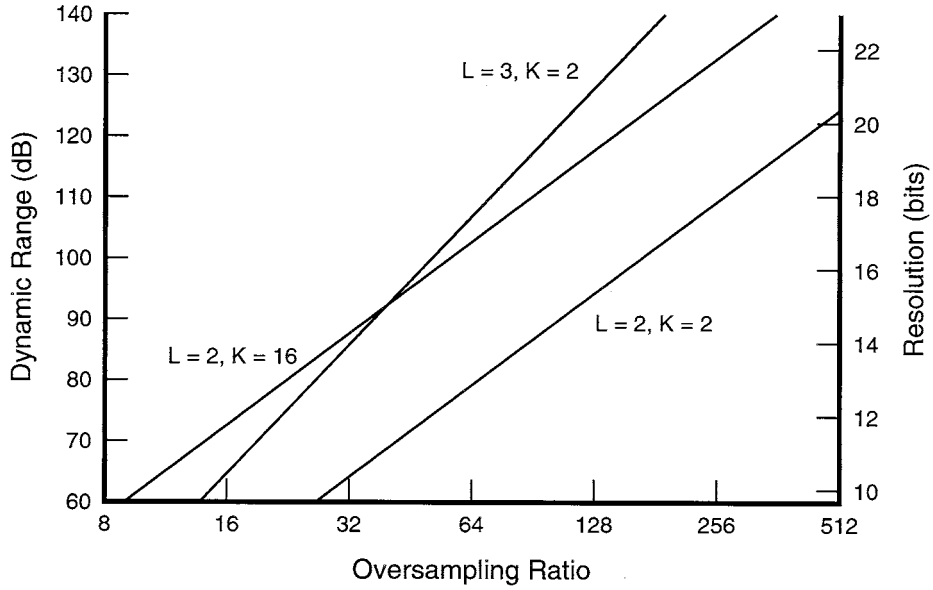


FIGURE 55.15 Calculated dynamic range vs. oversampling ratio.

One important measure of a sigma-delta modulator is its dynamic range, defined here as the ratio of the maximum sinusoidal input power to the noise power. With the quantizer output limited, as shown in Fig. 55.3, to a range of Δ, the maximum sinusoidal signal power is $\Delta^2/8$. The quantizer range Δ is related to the quantizer level separation δ by the number of quantization levels K, where

$$\delta = \frac{\Delta}{K-1} \tag{55.19}$$

The dynamic range of a noise-differencing sigma-delta modulator is then

$$DR = \frac{3}{2}\frac{2L+1}{\pi^{2L}} M^{2L+1}\left(K-1\right)^2 \tag{55.20}$$

Because the dynamic range is such a strong function of the oversampling ratio, the number of bits required to achieve a given dynamic range is substantially less in a sigma-delta modulator than in a Nyquist-rate converter. To illustrate this, the dynamic range, as given by Eq. 55.20, is shown in Fig. 55.15 as a function of the oversampling ratio, M, for three combinations of modulator order, L, and number of quantization levels, K. The equivalent resolution in bits that would be required of a Nyquist-rate converter to achieve the same dynamic range is shown in the right-hand axis of this figure. It can be inferred from Eq. 55.20 that a large dynamic range can be obtained even with only two quantization levels. This is important when circuit imperfections in actual sigma-delta data converter implementations are considered.

Sigma-Delta Modulators in Data Converters

The generalized modulator shown in Fig. 55.13 must be subtly modified when applied to the A/D and D/A converters in Figs. 55.1 and 55.2. In an A/D converter, the quantizer is actually a coarse K-level A/D converter (ADC), having an analog input and a digital output. (Since K is generally small, the K-level ADC is usually just a small flash converter, as described in Chapter 54.) The quantized digital code is fed

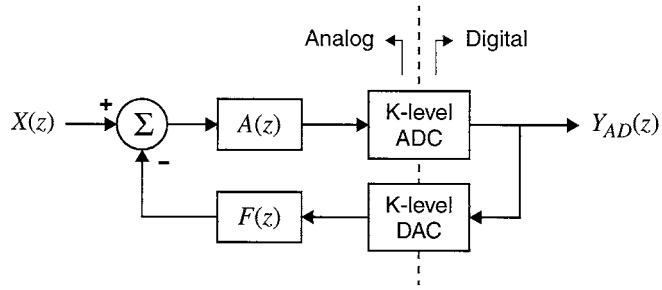


FIGURE 55.16 Sigma-delta modulator for A/D conversion.

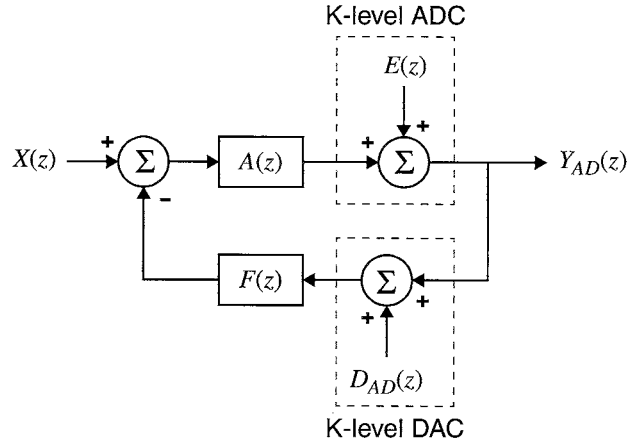


FIGURE 55.17 *K*-level DAC error in sigma-delta A/D converter.

back into the analog $F(z)$ through a K-level D/A converter (DAC), as shown in Fig. 55.16. Imperfections in this K-level DAC will introduce an additional error term $D_{AD}(z)$ as shown in Fig. 55.17. With the addition of this DAC error term, the modulator output is

$$Y_{AD}(z) = H_x(z)\left[X(z) - F(z)D_{AD}(z)\right] + H_e(z)E(z) \tag{55.21}$$

Since the feedback transfer function, $F(z)$, is unity in the band of interest (see Eq. 55.13), the DAC error is indistinguishable from the input. If there are more than two quantization levels, any mismatch between the level separations in the DAC will manifest itself as distortion because the DAC input is signal dependent. On the other hand, if there are only two quantization levels, there is only one level separation, and errors in the DAC levels will not cause distortion. (At worst, DAC errors in a two-level modulator will introduce a dc offset and a gain error.) Thus, with two-level sigma-delta modulators, it is possible to achieve low distortion and low-noise performance without precise component matching.

Unfortunately, most, if not all, of the statistical conditions that led to Eq. 55.9 and the subsequent equations are violated when a two-level single-threshold quantizer is used in a sigma-delta modulator.[15] Furthermore, the effective gain of the quantizer, which in Fig. 55.3 is implied to be unity, is undefined for a single-threshold quantizer. Nonetheless, empirical evidence has indicated that Eq. 55.20 is still a reasonable approximation for two-level noise-differencing sigma-delta modulators, and is useful as a design guideline for the amount of oversampling needed to achieve a specific dynamic range for a given modulator order.[16]

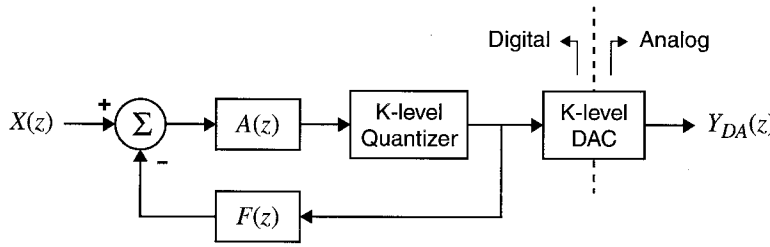


FIGURE 55.18 Sigma-delta modulator for D/A conversion.

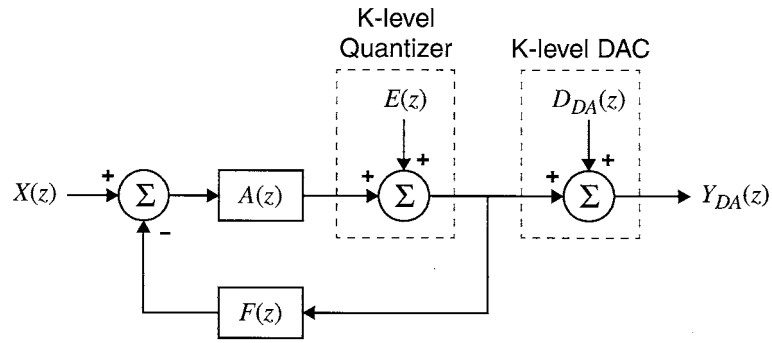


FIGURE 55.19 *K*-level DAC error in sigma-delta D/A converter.

As in sigma-delta ADCs, there is also a DAC error term in sigma-delta based DACs. In a sigma-delta DAC, the modulator loop is implemented digitally, and the output of that loop is applied to a coarse *K*-level DAC that provides the analog input for the reconstruction filter, as shown in Fig. 55.18. Imperfections in the *K*-level DAC will introduce an error term $D_{DA}(z)$ as shown in Fig. 55.19. With the addition of this error term, the modulator output is

$$Y_{DA}(z) = H_x(z)X(z) + D_{DA}(z) + H_e(z)E(z) \tag{55.22}$$

Since the input transfer function $H_x(z)$ is unity in the band of interest (see Eq. 55.11), the DAC error is indistinguishable from the input, just as in the A/D case. Once again, two-level quantization can be used to avoid DAC-introduced distortion.

Tones

One problem with the simplified noise model of sigma-delta modulators that led to Eq. 55.20 is the failure to predict non-harmonic tones. This is especially true for two-level modulators. Repetitive patterns in the coded modulator output that cause discrete spectral peaks at frequencies not harmonically related to any input frequency can occur in sigma-delta modulators.[10,16-18] These tones can manifest themselves as odd "chirps" or "pops," and they exist even in ideal sigma-delta modulators; they are not caused by circuit imperfections.[2]

The origin of sigma-delta tones is illustrated in the following example. Consider a first-order sigma-delta modulator, such as that in Fig. 55.7, with a dc input of 0.0005. Let the quantizer have two output levels, +0.5 and –0.5. The output of such a modulator will be a sequence of +0.5's and –0.5's such that the time average of the outputs is 0.0005. To achieve this average, the output of the first-order modulator will be a stream of alternating +0.5's and –0.5's, with an extra +0.5 every 1000 clock cycles. This is

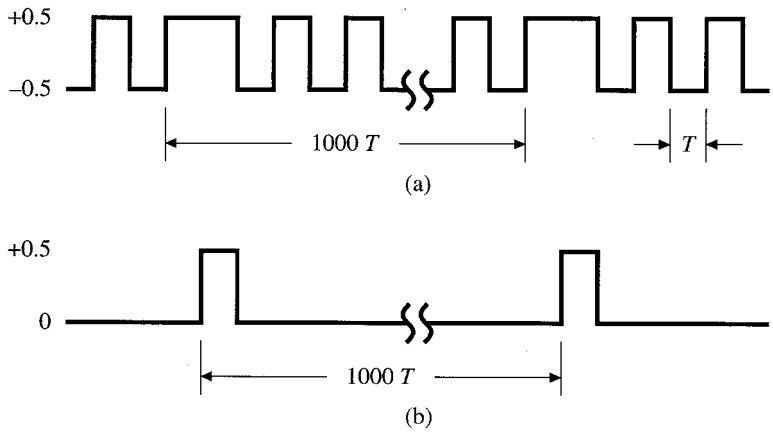


FIGURE 55.20 (a) Output sequence with average of 0.0005; and (b) running average of (a).

illustrated in Fig. 55.20(a), where T is the clock period ($T = 1/f_S$). The two-cycle running average of this output is shown in Fig. 55.20(b). For the most part, this running average is zero, except that at every 1000 clock cycles, there is a one clock cycle pulse. This repetitive pulse produces a tone in the output spectrum at a frequency of

$$f_P = \frac{f_S}{1000} = \frac{M}{500} f_H \tag{55.23}$$

If the oversampling ratio M is less than 500, this tone will appear in the baseband spectrum.

In sigma-delta modulators with more active input signals, the output sequence is typically more complex than that illustrated in Fig. 55.20. Nonetheless, the concept underlying tone behavior is that repeating patterns in the quantizer output cause non-uniformity in the quantizer error spectrum, which in the worst case is a discrete spectral peak. A measured tone for a second-order modulator with a dc input is shown in Fig. 55.21.[19]

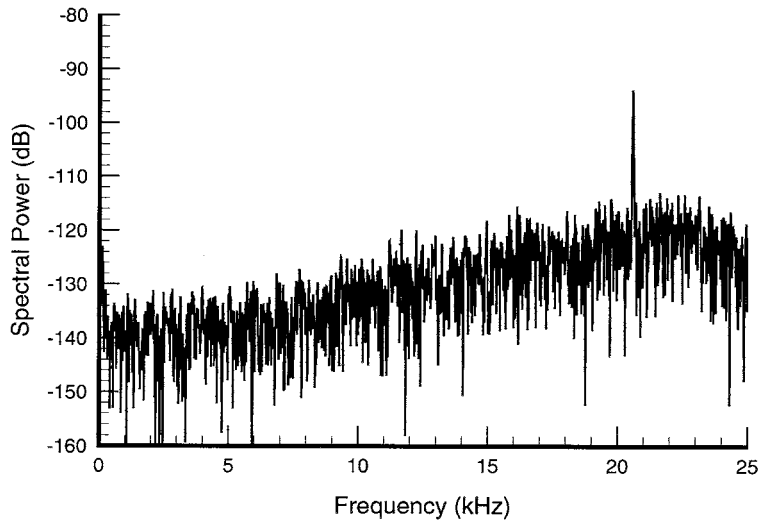


FIGURE 55.21 Measured tone in second-order modulator with dc input.

Several means of mitigating sigma-delta tones have been used. The first rule is to avoid using first-order sigma-delta modulators. Aside from having inferior noise-shaping properties compared to other modulator architectures, first-order modulators have terrible tone properties.[15,16] The situation improves dramatically with second- and higher-order modulators. In fact, the presence of tones may only be a perceptual or marketing concern, as the total tone power is usually less than the broadband quantization noise power.[20]

When tone magnitudes must be reduced, several techniques have proven effective. These include dither, cascaded architectures, and multi-level quantization. Of these three, dither is the only technique whose sole benefit is the reduction of tones. The simplest type of dither is to add a moderate amplitude out-of-band signal, such as a square wave, to the input.[9,21,22] This dither signal is attenuated by the same filter that attenuates the quantization noise. The purpose of this dither is to keep the sigma-delta modulator out of modes that produce patterns, and for some types of tones this technique is effective. A more rigorously effective technique is to add a large amplitude pseudo-random noise signal at the quantizer input.[23] This noise is spectrally shaped just like the quantization noise, and is the most effective dither scheme for eliminating tones. Its drawbacks are the expense in silicon area of the random noise generator and the 2- to 3-dB reduction in dynamic range caused by the dither noise.

The other two tone mitigation techniques, cascaded architectures and multi-level quantization, are simply more complex sigma-delta architectures that happen to have improved tone properties over simple noise-differencing two-level sigma-delta modulators. These techniques are covered in Sections 55.3 and 55.4, respectively.

55.3 Alternative Sigma-Delta Architectures

Equation 55.20 appears to indicate that the order of the modulator, L, can be any value, and that increasing L would be beneficial. However, one further problem with two-level sigma-delta modulators is that two-level noise-differencing modulators of order greater than two can exhibit unstable behavior.[10] For this reason, only first- and second-order modulators were discussed in the Section 55.2. Nonetheless, there have been acceptably stable practical alternative architectures that achieve quantization noise shaping that is superior to a second-order modulator. Two such architectures, high-order and cascaded modulators, are discussed in this section.

Another assumption in the previous section was that the noise-shaped region in a sigma-delta modulator is centered around dc. This is not necessarily the case; sigma-delta modulators with noise-shaped regions at frequencies other than near dc are called *bandpass modulators* and are discussed at the end of this section.

High-Order Modulators

A high-order modulator is a modulator such as that depicted in Fig. 55.13 in which there are more than two zeros in the noise transfer function. As stated earlier, if two-level quantization is employed, a simple noise-differencing series of integrators cannot be used, as such architectures produce unstable oscillations with large inputs that do not recover when the input is removed. To overcome this problem, high-order modulators use forward and feedback transfer functions that are more complex than the noise-differencing functions in Eqs. 55.14 and 55.15.[24-26]

The general rule of thumb in the design of high-order modulators is that the modulator can be made stable if

$$\lim_{z\to\infty} H_e(z) = 1 \tag{55.24}$$

$$\left|H_e(z)\right| \le A, \text{ for } |z| = 1 \tag{55.25}$$

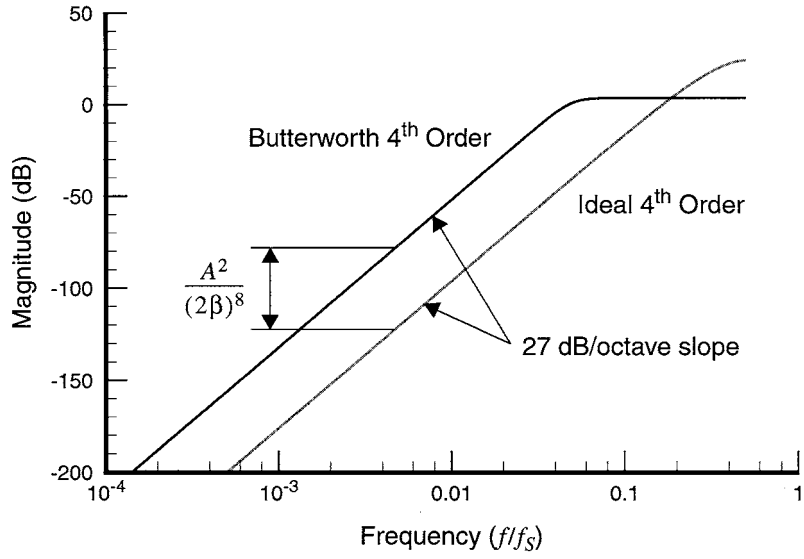


FIGURE 55.22 Fourth-order error spectrum.

and the integrator outputs are clipped and/or scaled to prevent self-sustaining instability.[26,27] The maximum error gain A is about 1.5, but the value used represents a tradeoff between noise attenuation and modulator stability. These rules cover a broad class of filter types and modulator architectures, and the type of filter used generally follows the traditions of the previous designers in an organization.

As an example, consider a fourth-order modulator with a highpass Butterworth error transfer function having a maximum gain, A, of 1.5, and a cutoff frequency set such that Eq. 55.24 is satisfied. The error spectrum of the Butterworth filter is shown in Fig. 55.22, along with the error transfer function of an ideal fourth-order difference. While the Butterworth filter holds the maximum gain to 1.5 (3.5 dB), and while both filters have a fourth-order noise shaping slope in the baseband (27 dB/octave), the error power in the baseband is 44 dB higher with the Butterworth filter than with the ideal noise-differencing filter. This error penalty is typical of high-order designs; there is usually a direct tradeoff between stability and noise reduction.

Consider the more general case of an L-th order highpass Butterworth error transfer function. The error transfer function of such a filter around the unit circle is

$$\left|H_e\left(e^{j\omega}\right)\right|^2 = \frac{A^2\left(\frac{1}{\beta}\tan\frac{\omega}{2}\right)^{2L}}{1+\left(\frac{1}{\beta}\tan\frac{\omega}{2}\right)^{2L}} \tag{55.26}$$

The filter coefficients $H_e(z)$ for needed to satisfy Eq. 55.26 can be computed using standard digital filter design techniques.[28] For a given filter order, L, and gain, A, the parameter β must be chosen to satisfy Eq. 55.24. (The condition in Eq. 55.25 is always satisfied when Eq. 55.26 is true.) These solutions can be computed numerically, and it is found empirically that

$$\beta = A^e\beta_N \tag{55.27}$$

TABLE 55.1 High-Order Butterworth Gain Factors and Dynamic Range (DR) Loss

				Zero Placement	
L	β_N	β	Loss in DR (dB)	DR Improvement (dB)	Net Loss in DR (dB)
3	0.052134	0.1570	33.7	8.0	25.8
4	0.051709	0.1557	44.1	12.8	31.2
5	0.033866	0.1020	72.6	17.9	54.7
6	0.034903	0.1051	84.8	23.2	61.6
7	0.025390	0.0764	117.7	28.6	89.1

Note: Except for β_N, all values are calculated for $A = 1.5$.

where the values for β_N are tabulated in Table 55.1. The loss in dynamic range relative to an ideal noise-differencing modulator, given by, $A^2/(2\beta)^{2L}$, is also tabulated. In spite of this loss, high-order modulators can still achieve better noise performance than second-order modulators. However, because of the compromise in dynamic range required to stabilize high-order modulators, third-order modulators are generally not worth the effort. More common are fourth-and fifth-order modulators.

The noise penalty required to stabilize high-order modulators can be mitigated to some extent by alternate zero placement.[25] Classic noise-differencing modulators place all of the zeros of the error transfer function at dc ($z = 1$). This causes most of the noise power to be concentrated at the highest baseband frequencies. If, instead, the zeros are distributed throughout the baseband, the total noise in the baseband can be reduced, as illustrated in Fig. 55.23. The amount by which zero placement can improve the noise transfer function is summarized in Table 55.1. Also tabulated is the net loss in dynamic range of a high-order Butterworth modulator that uses zero placement relative to an ideal noise-differencing modulator that has zeros at dc.

Cascaded Modulators

Cascaded, or multi-stage, architectures are an alternative means of achieving higher-order noise shaping without the stability problems of the high-order modulators described in the previous section.[29,30] In a cascaded modulator, two or more stable first- or second-order modulators are connected in series, with

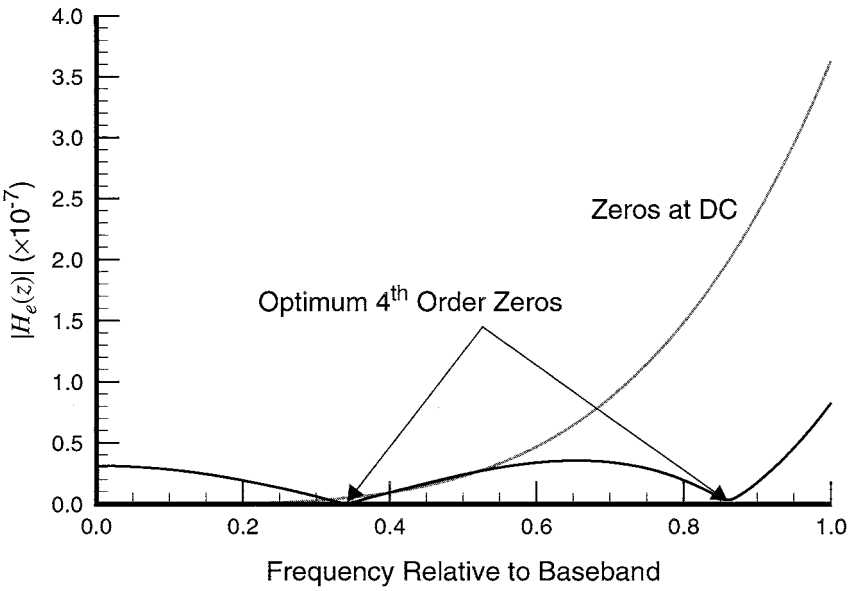


FIGURE 55.23 Fourth-order distributed zeros.

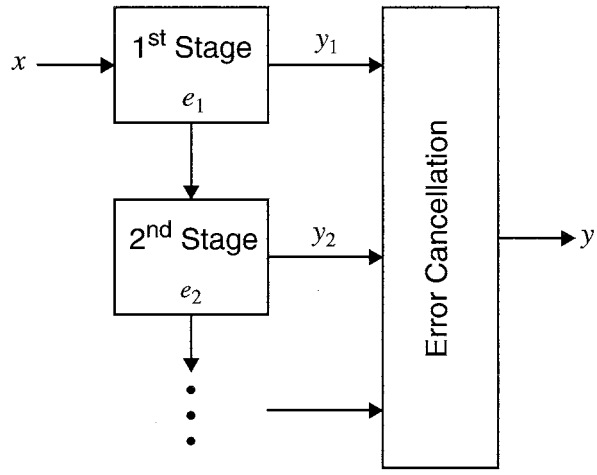


FIGURE 55.24 Cascaded sigma-delta modulators.

the input of each stage being the error from the previous stage, as illustrated in Fig. 55.24. Referring to this illustration, the first stage of the cascade has two outputs, y_1 and e_1. The output y_1 is an estimate of the input x. The error in this estimate is e_1. The second stage has as its input the error from the first stage, e_1, and its outputs are y_2 and e_2. The second stage output y_2 is an estimate of the first stage error e_1. By subtracting this estimate of the first stage error from the output of the first stage, y_1, only the second stage error remains. Thus, the error cancellation network uses the output of one stage to cancel the error in the previous stage.

For example, in a cascaded architecture comprising a second-order noise-differencing modulator followed by a first-order noise-differencing modulator, the transforms of the output of the two stages, as given by Eq. 55.16, are

$$Y_1(z) = z^{-2}X(z) + \left(1 - z^{-1}\right)^2 E_1(z) \tag{55.28}$$

$$Y_2(z) = z^{-1}E_1(z) + \left(1 - z^{-1}\right)E_2(z) \tag{55.29}$$

If the error cancellation network combines the two outputs such that

$$Y(z) = z^{-1}Y_1(z) - \left(1 - z^{-1}\right)^2 Y_2(z) \tag{55.30}$$

then the error in the first stage will be cancelled, and the output will be

$$Y(z) = z^{-3}X(z) - \left(1 - z^{-1}\right)^3 E_2(z) \tag{55.31}$$

The final output of this cascaded modulator is third-order noise shaped. As a general rule, the noise shaping of a cascaded architecture is comparable to a single-stage modulator whose order is the sum of all the orders in the cascade.

The extent to which the errors in a cascaded modulator can be cancelled depends on the matching between the stages. The earliest multi-stage modulators were cascades of three first-order stages, often called the MASH architecture.[29] The disadvantage of this structure is that in order to achieve third-order

performance, the error in the first stage, which is only first-order shaped, must be cancelled. Cancelling this relatively large error places a stringent requirement on inter-stage matching. An alternative architecture that has much more relaxed matching requirements is the cascade of a second-order modulator followed by a first-order modulator. This architecture, like the MASH, ideally achieves third-order noise shaping. Its advantage is that the matching can be 100 times worse than a MASH and still achieve better noise shaping performance.[31]

An additional benefit of cascaded modulators is improved tone performance. It has been shown both analytically and experimentally that the error spectra of the second and subsequent stages in a cascade are not plagued by the spectral tones that can exist in single-stage modulators.[19,32] To the extent that the first-stage error is cancelled, any tones in the first-stage error spectrum are attenuated, and the final output of the cascaded modulator is nearly tone-free.

Bandpass Modulators

The aforementioned sigma-delta architectures, called herein *baseband modulators*, all have zeros at or near dc; that is, at frequencies much less than the modulator sampling rate. It is also possible to group these zeros at some other point in the sampling spectrum; such architectures are called *bandpass modulators*. Bandpass architectures are useful in systems that need to quantize a narrow band signal that is centered at some frequency other than dc. A common example of such a signal is the intermediate frequency (IF) signal in a communications receiver.

The simplest method for designing a bandpass modulator is by applying a transformation to an existing baseband modulator architecture. The most common transformation is to replace occurrences of z with $-z^2$.[2] Such an architecture has zeros at $f_S/4$ and is stable if the baseband modulator is stable.[33] A comparison of the error transfer function of baseband and bandpass modulators is shown in Fig. 55.25. Note that a bandpass modulator generated through this transformation has twice the order of its equivalent baseband counterpart. For example, a fourth-order bandpass modulator is comparable to a second-order baseband modulator.

The noise shaping properties of a bandpass modulator generated through the $-z^2$ transformation are equivalent to the baseband modulator that was transformed. Thus, the approximation in Eq. 55.20 can be used where L is the order of the baseband modulator that was transformed and M is the effective oversampling ratio, which in a bandpass modulator is the sampling rate divided by the signal bandwidth.

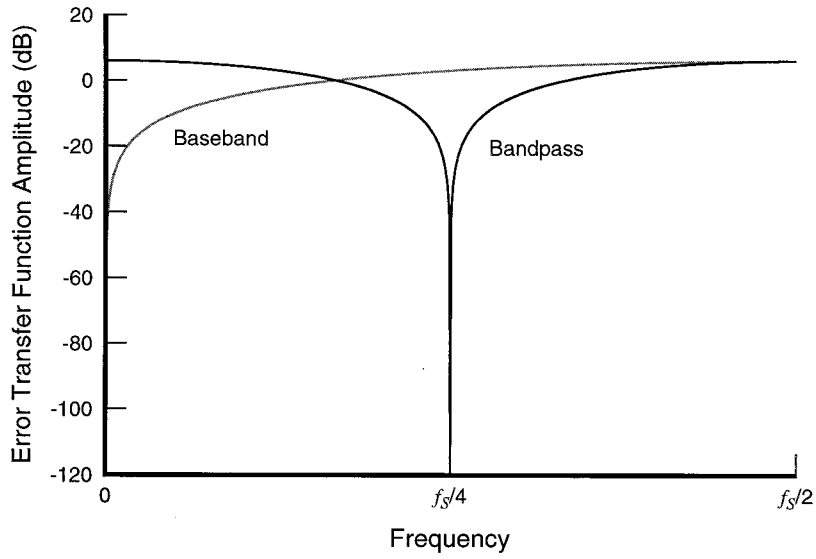


FIGURE 55.25 Bandpass noise transfer function.

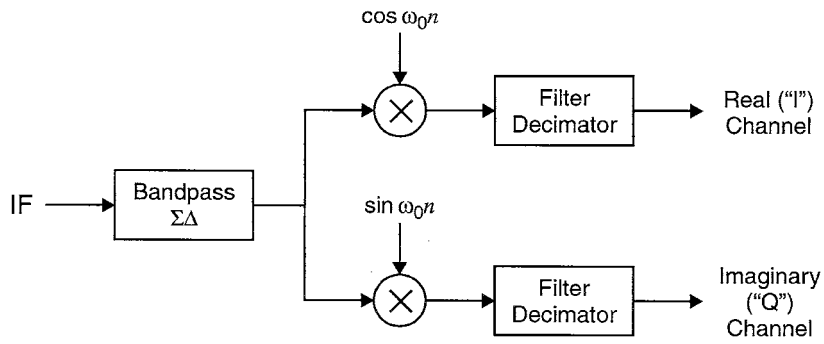


FIGURE 55.26 IQ demodulation with a bandpass modulator.

There are advantages and disadvantages to bandpass modulators when compared with traditional down-conversion and baseband modulation. One advantage of the bandpass modulator is its insensitivity to $1/f$ noise. Since the signal of interest is far from dc, $1/f$ noise is often insignificant. Another advantage of bandpass modulation applies specifically to bandpass modulators having zeros at $f_S/4$ that are used in quadrature I and Q demodulation systems. If the narrowband IF signal is to be demodulated by a cosine and sine waveform, as shown in Fig. 55.26, the demodulation operation becomes simple multiplication by 1, –1, or 0 when the demodulation frequency is $f_S/4$.[34] Furthermore, because a single modulator is used, the bandpass modulator is free of the I/Q path mismatch problems that can exist in baseband demodulation approaches.

Two disadvantages of bandpass modulators involve the sampling operation. Sampling in a bandpass modulator has linearity requirements that are comparable to a Nyquist-rate converter sampling at the same IF frequency; this is much more severe than the linearity requirements of the sampling operation in a baseband converter with the same signal bandwidth. Also, because of the higher signal frequencies, the sampling in bandpass modulators is much more sensitive to clock jitter. To date, the state of the art in bandpass modulators has about 20 dB less in dynamic range than comparable baseband modulators.[2] While the remainder of this chapter focuses once again on baseband modulators, many of the techniques are applicable to bandpass modulators as well.

55.4 Filtering for Sigma-Delta Modulators

In Sections 55.2 and 55.3 of this chapter, the discussion focused on the operation of the sigma-delta modulator core. While this core is the most unique aspect of sigma-delta data conversion, there are also filtering blocks that constitute an important part of sigma-delta A/D and D/A converters. In this section, the non-modulator components in baseband sigma-delta converters, namely the analog and digital filters, are described. First, the requirements of the analog anti-alias and reconstruction filters are described. Second, typical architectures for the decimation and interpolation filters are discussed. While much of the design of these filters use standard techniques covered elsewhere in this volume, there are aspects of these filters that are specific to sigma-delta modulator applications.

Anti-alias and Reconstruction Filters

The purpose of the anti-alias filter, shown in Fig. 55.1 at the input of the sigma-delta A/D converter, is, as the name would indicate, to prevent aliasing. The sampling operation maps, or aliases, all frequencies into the range bounded by $\pm f_S/2$.[28] Specifically, all signals within a baseband bandwidth of multiples of the sampling rate are mapped into the baseband. This is generally undesirable, so the anti-alias filter is designed to attenuate this aliasing to some tolerable level. One advantage of sigma-delta converters over Nyquist-rate converters is that this anti-aliasing filter has a relatively wide transition region. As illustrated in Fig. 55.27, the passband region for this filter is the signal bandwidth f_B, while the stopband region for

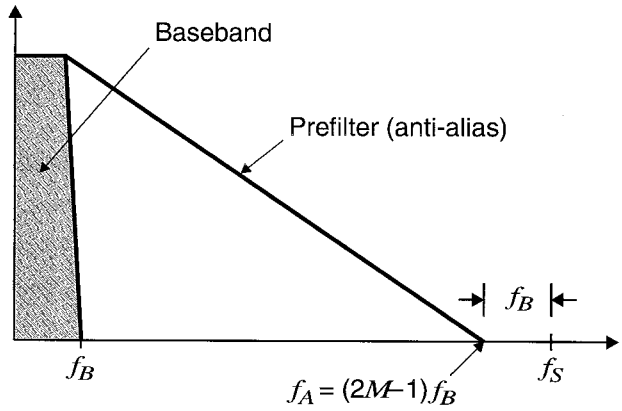


FIGURE 55.27 Anti-alias filter for sigma-delta A/D converters.

this filter is only within f_B of the sampling rate. Thus, the transition region is $2(M-1)\,f_B$, and since $M \gg 1$, the transition region is relatively wide. A wide transition region generally means a simple filter design. The precise nature of the anti-alias filter is application dependent, and can be designed using any number of standard analog filter techniques.[35]

The reconstruction filter, shown in Fig. 55.2 at the output of the sigma-delta D/A converter, is also an analog filter. Its primary purpose is to remove unwanted out-of-band quantization noise. The extent to which this noise must be removed varies widely from system to system. If the analog output is to be applied to an element that is naturally bandlimited, such as a speaker, then very little attenuation may be necessary. On the other hand, if the output is applied to additional analog circuitry, care must be taken lest the high-frequency noise distort and map itself into the baseband. Circuit techniques for this filter are addressed further in Section 55.5.

Decimation and Interpolation Filters

In general, the filter characteristics of the decimation filter, shown in Fig. 55.1 at the output of the sigma-delta A/D converter, are much sharper than those of the anti-alias filter; that is, the transition region is narrower. The saving grace is that the filter is implemented digitally, and modern sub-micron processes have made complex digital filters economically feasible. Nonetheless, care must be taken or the filter architecture will become more computationally complex than is necessary.

The basic purpose of the decimation filter is to attenuate quantization noise and unwanted signals outside the baseband so that the output of the decimation filter can be down-sampled, or decimated, without significant aliasing. Normally, the most efficient means of accomplishing this is to apply a multi-rate filter architecture, such as that illustrated in Fig. 55.28.[36,37] The comb filter is a relatively crude, but easy to implement, filter that has zeros equally spaced throughout the sampled spectrum. The frequency response of an N-th order comb filter, $H_C(z)$, is

$$H_C(z) = \left(\frac{1}{R} \frac{1 - z^{-R}}{1 - z^{-1}} \right)^N \tag{55.32}$$

where R is the impulse response length of the comb filter. If R is set equal to the decimation ratio of the comb filter (the comb filter input rate divided its output rate), then the filter zeros will occur at every point that would alias to dc.[38,39] If the filter order N is one more than the modulator order, then the comb filter will be adequate to attenuate the out-of-band quantization noise to the point where it does not adversely increase the baseband noise after decimation.[40]

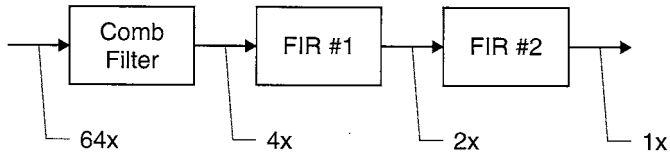


FIGURE 55.28 Typical decimation filter architecture.

Following the comb filter is typically a series of one or more FIR filters. Since the sample rates of these FIR filters are much slower than the oversampled clock rate, each filter output can be computed over many clock cycles. Also, since the output of each filter is decimated, only the samples that will be output need to be calculated. These properties can be exploited to devise computationally efficient structures for decimation filtering.[41]

In the example in Fig. 55.28, the first FIR filter is decimating from 4× to 2× oversampling. Since the output of this filter is still oversampled, the transition region is relatively wide and the attenuation at midband need not be very high. Thus, an economical half-band filter (a filter in which every other coefficient is zero) can be used.[37]

The final FIR filter is by far the most complex. It usually has to have a very sharp transition region, and for strict anti-alias performance, it cannot be a halfband filter. In high-performance sigma-delta modulators, this filter is often in the range of 50 to 200 taps in length. Standard digital filter design techniques can be used to select that tap weights for this filter.[28] Since it is going to be a complex filter anyway, it can also be used to compensate for any frequency droop in the previous filter stages.

The interpolation filter, shown in Fig. 55.2 at the input of the sigma-delta D/A converter, up-samples the input digital words to the oversampling rate. In many ways, this filter is the inverse of a decimation filter, typically comprising a complex up-sampling FIR filter, optionally followed by one or more simple FIR filters, followed by an up-sampling comb filter. The up-sampling operation, without this filter, would produce images of the baseband spectrum at multiples of the baseband frequency. The purpose of the interpolation is to attenuate these images to a tolerable level. What constitutes tolerable is very much a system-dependent criterion. Design techniques for the interpolation filter parallel those of the decimation filter discussed above.

55.5 Circuit Building Blocks

For analog-to-digital conversion, the modulator is implemented primarily in the analog domain as shown in Fig. 55.16. In digital-to-analog conversion, the modulator output if filtered by an analog reconstruction filter as depicted in Fig. 55.2. The basic analog circuit building blocks for these data converters are described in this section. These building blocks include switched-capacitor integrators, the amplifiers that are imbedded in the integrators, comparators, and circuits for sigma-delta based D/A conversion. At the end of this section, the techniques for continuous-time sigma-delta modulation are briefly discussed.

Switched-Capacitor Integrators

Switched-capacitor integration stages are commonly used to perform the signal processing functions of integration and summation required for realization of the discrete time transfer functions $A(z)$ and $F(z)$ in Fig. 55.16. The circuit techniques outlined herein are drawn from a rich literature of switched-capacitor filters[42-45] that is detailed elsewhere in this volume.

Figure 55.29 is a typical integrator stage for the case of single bit feedback,[11,20] and is designed to perform the discrete time computation

$$V_{OUT}(z) = K\frac{z^{-1}}{1-z^{-1}}\left(V_{IN}(z) - V_{DAC}(z)\right) \tag{55.33}$$

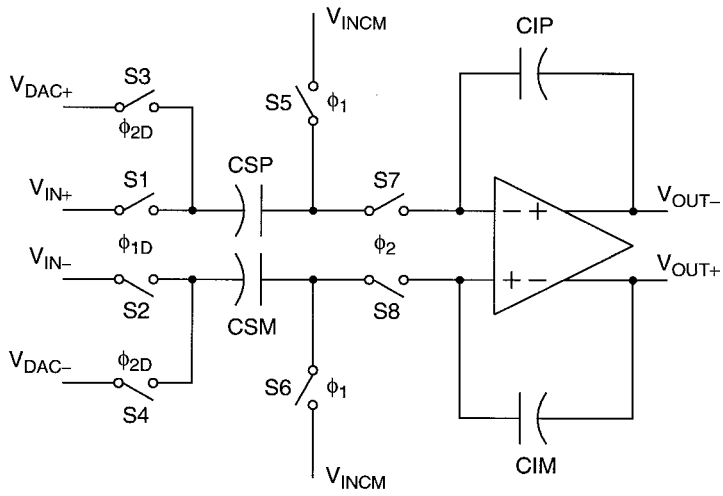


FIGURE 55.29 Typical integrator stage.

independent of the parasitic capacitances associated with the capacitive devices shown. The curved line in the capacitor symbol is the device terminal with which the preponderance of the parasitic capacitance is associated. For example, this will be the bottom plate of a stacked planar capacitance structure, where the parasitic capacitance is that between the bottom plate and the IC substrate. The circuit's precision stems from the conservation of charge at the two input nodes of the operational amplifier, and the cyclic return of the potential at those nodes to constant voltages. More details may be found in the chapter on switched capacitor filters (Chapter 59).

Fully differential circuits will be shown here, as these are almost universally preferred over single-ended circuits in monolithic implementations owing to their greatly improved power supply rejection, MOS switch feedthrough rejection, and suppression of even-order non-linearities. The switches shown in Fig. 55.29 are generally full CMOS switches, as detailed in Fig. 55.30. However, integrators with very low power supply voltages may necessitate the use of only one polarity of switch device, possibly with a switch gate voltage-boosting arrangement.[46] Sampling capacitors CSP and CSM are designed with the same capacitance C_S, and the effect of slight fabrication mismatches between the two will be mitigated by the common-mode rejection of the amplifier. Similarly, integration capacitors CIP and CIM are designed to be identical with capacitance C_I.

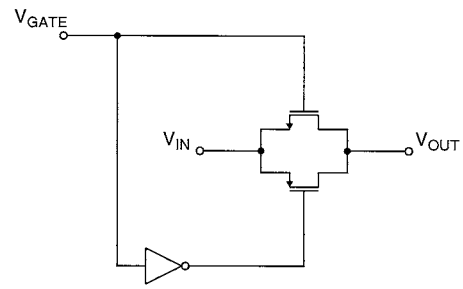


FIGURE 55.30 Full CMOS switch.

The discrete-time signal to be integrated is applied between the input terminals V_{IN+} and V_{IN-}, and the output is taken between V_{OUT+} and V_{OUT-}. V_{INCM} is the common mode input voltage required by the amplifier. The single-bit DAC feedback voltage is applied between V_{DAC+} and V_{DAC-}. The stage must be clocked by two non-overlapping signals, ϕ_1 and ϕ_2. During the ϕ_1 phase, the differential input voltage is sampled on the bottom plates of CSP and CSM, while their top plates are held at the amplifier common-mode input level. During this phase, the amplifier summing nodes are isolated from the capacitor network, and the amplifier output will remain constant at its previously integrated value. During the ϕ_2 phase, the bottom plates of the sampling capacitors CSP and CSM experience a differential potential shift of $(V_{DAC} - V_{IN})$, while the top plates are routed into the amplifier summing nodes. By forcing its differential input voltage to a small level, the amplifier will effect a transfer of a charge of $C_S(V_{IN} - V_{DAC})$ to the integration capacitors, and therefore the differential output voltage will shift to a new value by an

increment of $(C_S/C_I)(V_{IN} - V_{DAC})$. Since this output voltage shift will accumulate from cycle to cycle, the discrete-time transfer function will be that of Eq. 55.33, with

$$K = \frac{C_S}{C_I} \tag{55.34}$$

Over several cycles of initial operation, the amplifier input terminals will be driven to the common-mode level that is precharged onto the top plates of the sampling capacitors.

In order to suppress any signal-dependent clock feedthrough from the switches, it is helpful to slightly delay the clock phases that switch variable signal voltages with respect to the phases that switch current into constant potentials. The channel charge in each turned-on switch device can potentially dissipate onto the sampling capacitors when the switches are turned off, producing an error in the sampled charge. This channel charge is dependent on the difference between the switch gate-to-source voltage and its threshold voltage; and as the source voltage varies with signal voltage, the clock feedthrough charge will vary with the signal. By turning the switches that see constant potentials at the end of each cycle off first, and thus floating the sampling capacitor, the only clock feedthrough is a charge that is to the first order independent of signal level, and results only in a common-mode shift that is suppressed by the amplifier. This acts to reduce the non-linearity of the integrator and the harmonic distortion generated by the modulator.

The timing for the delayed and undelayed clocks is illustrated in Fig. 55.31, where the clock phases ϕ_{1D} and ϕ_{2D} represent phases that are slightly delayed versions of ϕ_1 and ϕ_2, respectively. The delayed clocks drive the switches that are subject to full-signal voltage swings, the analog and reference voltage inputs, as shown in Fig. 55.29. The undelayed clocks drive the switches associated with the amplifier summing node and common-mode input bias voltage, which will always be driven to the same potential by the end of each clock cycle. A typical clock generator circuit to produce these phase relationships is shown in Fig. 55.32. The delay time Δt is generated by the propagation delay through two CMOS inverters.

Other, more complex integration circuits are used in some sigma-delta implementations, for example, to suppress errors due to limited amplifier gain[47,48] or to effectively double the sampling rate of the integrators.[49,50] For the modulator structures discussed in Section 55.3 that are more elaborate than a second-order loop, more complex switched-capacitor filtering is required. These may still, however, be designed with the same basic integrator architecture as in Fig. 55.29, but with extra sampling capacitors feeding the amplifier summing node to implement additional signal paths.[26,33,51] Consult Chapter 59 in this volume on switched-capacitor filtering for more information.

Operational Amplifiers

Embedded in the switched-capacitor integrator shown in Fig. 55.29 is an operational amplifier. There are three major types of operational amplifiers typically used in switched-capacitor integrators[52]: the

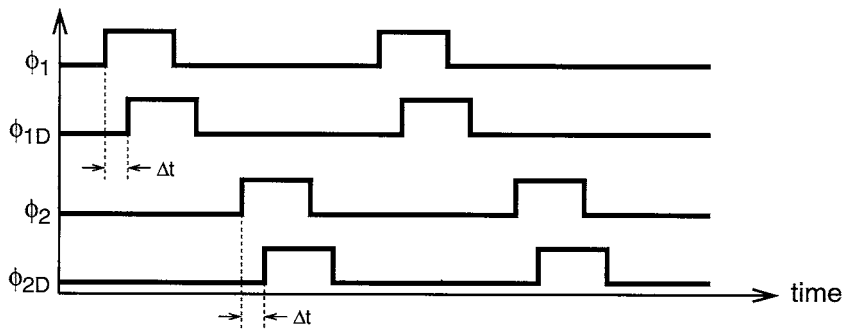


FIGURE 55.31 Delayed clock timing.

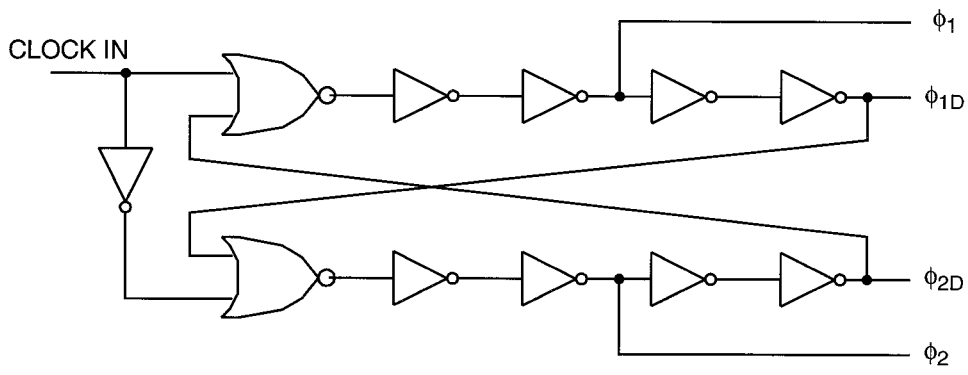


FIGURE 55.32 Non-overlapping clock generator with delayed clocks.

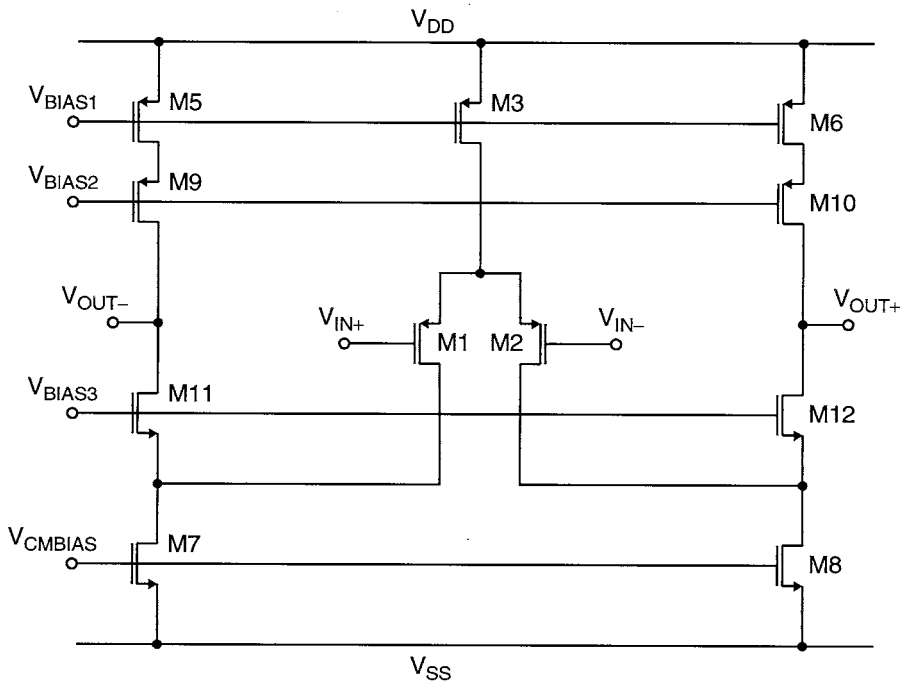


FIGURE 55.33 Folded cascode amplifier.

folded cascode amplifier,[42] shown in Fig. 55.33; the two-stage amplifier,[43] shown in Fig. 55.34; and the class AB amplifier,[45] shown in Fig. 55.35.

When the available supply voltage is high enough to permit stacking of cascode devices to develop high gain, a *folded cascode amplifier* is commonly used. A typical topology is shown in Fig. 55.33. The input devices are PMOS, since most IC processes feature PMOS devices that exhibit lower $1/f$ noise than their NMOS counterparts.[53] The input differential pair M1 and M2 is biased with the drain current of M3. FETs M5-M8 function as current sources, and M9-M12 form cascode devices that boost the output impedance. The amplifier is compensated for stability in the integrator feedback loop by the dominant pole that is formed at its output node with the high output impedance and the load capacitance. In an integrator stage, the amplifier will be loaded with the load capacitance of the following stage sampling capacitance as well as its own integration capacitance. The non-dominant pole at the drains of M1 and M2 limit the unity-gain frequency, which can be quite high.

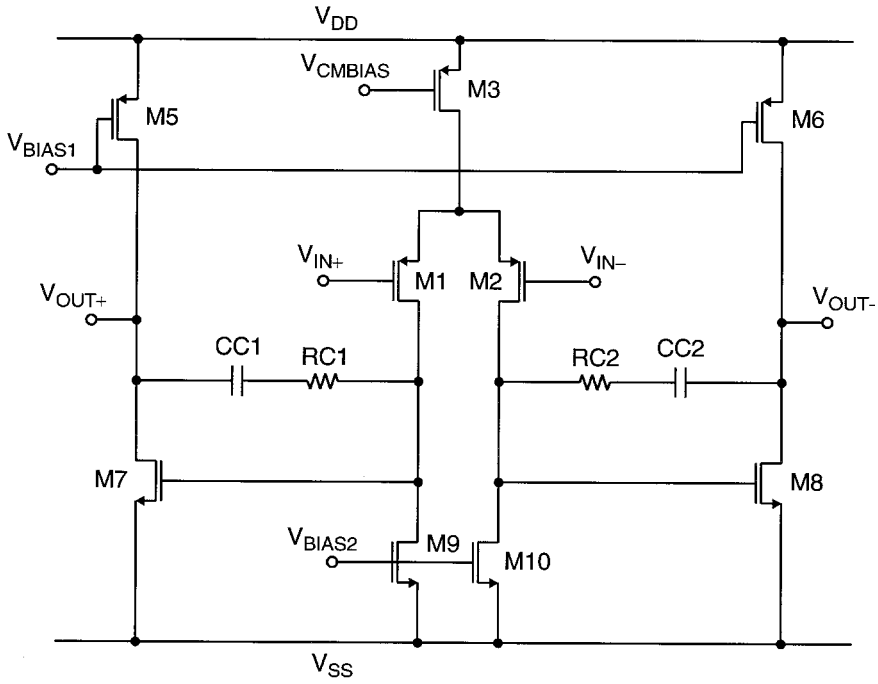


FIGURE 55.34 Two-stage amplifier.

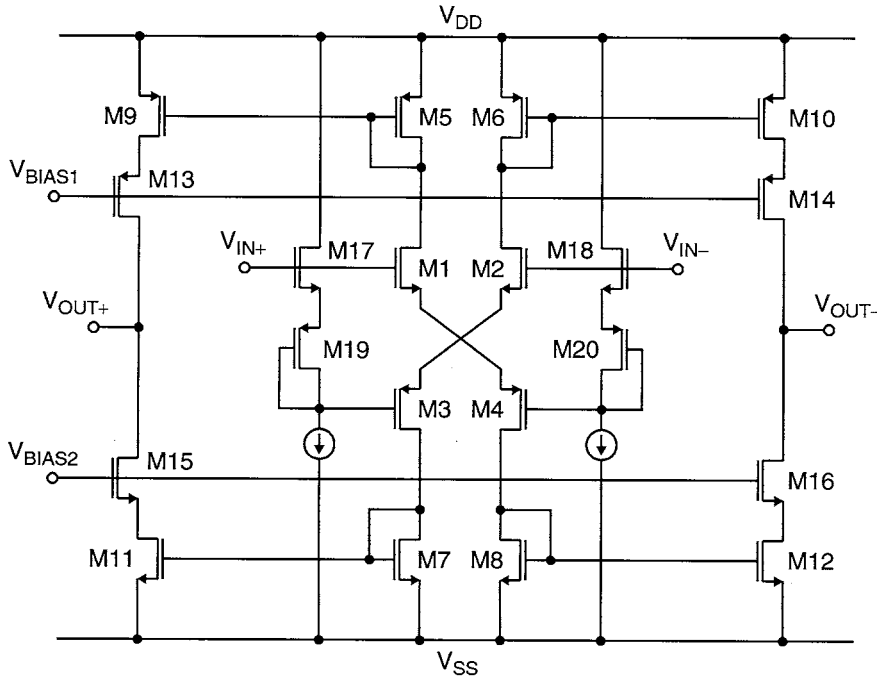


FIGURE 55.35 Class AB amplifier.

When the power supply voltage is limited, and cascode devices cannot be stacked and still preserve adequate signal swing, a *two-stage amplifier* is a common alternative to the folded cascode amplifier. As shown in Fig. 55.34, the input differential pair of M1 and M2 now feed the active load current sources

of M9 and M10 to form the first stage. The second stage comprises common-source amplifiers M7 and M8, loaded with current sources M5 and M6. Due to the presence of two poles from the two stages of roughly comparable frequencies, compensation is generally achieved with a pole-splitting RC local feedback network as shown.[52] Often, the resistors RC1 and RC2 are actually implemented as NMOS devices biased into their ohmic region by tying their gates to V_{DD}. In this arrangement, the effective resistance of RC1 and RC2 will approximately track any drift in mobility of M7 and M8 over temperature and processing variations, preserving the compensated phase margin. For a given process, the bandwidth of a two-stage amplifier is less than what can be achieved than by a folded cascode design; but because the two-stage has no stacked cascode devices, the signal swing is higher.

In the case of modulators with higher clock speeds, both folded cascode and two-stage amplifiers may have unacceptably long settling times; in these amplifiers, the maximum slewing current that can be applied to charge or discharge the load capacitance is limited by fixed current sources. This slewing limitation can be overcome by a *class AB amplifier* topology that can supply a variable amount of output current and is capable of providing a large pulse of current early in the settling cycle when the differential input error voltage is high. A typical class AB amplifier topology is shown in Fig. 55.35. The input differential pair from the folded cascode and two-stage designs is replaced by M1 through M4, and their drain currents are mirrored to the output current sources M9–M12 by diode-connected devices M5–M8. Cascode devices M13–M16 enhance the output impedance and gain. As with the folded cascode design, frequency compensation is accomplished by a dominant pole at the output node. The input voltage is fed directly to the NMOS input devices and to the PMOS input devices through the level-shifting source follower and diode combination M17–M20. This establishes the quiescent bias current through the input network M1–M4, and therefore through the output devices as well.

In each of the three amplifier topologies discussed above, there is either one or a set of two matched current sources driving both differential outputs. These current sources are controlled by a gate bias line labeled V_{CMBIAS}. The current output of these devices will determine the common-mode output voltage of the amplifier independent, to the first order, of the amplified differential signal. The appropriate potential for V_{CMBIAS} is determined by a feedback loop that is only operable in the common mode and is separate from the differential feedback instrumental in the charge integration process.

Since a discrete time modulator is, by its nature, clocked periodically, a natural choice for the implementation of this common-mode feedback loop is the switched-capacitor network of Fig. 55.36.[44,45] Capacitors CCM1 and CCM2 act as a voltage divider for transient voltages that derives the average, or common-mode, voltage of the amplifier output terminals. This applies corrective negative feedback transients to the V_{CMBIAS} node to stabilize the feedback loop during each clock period while the amplifier is differentially settling.

A dc bias is then maintained on CCM1 and CCM2 by the switched-capacitor network on the left side of the figure. This will slowly transfer the charge necessary to establish and maintain a dc level shift that

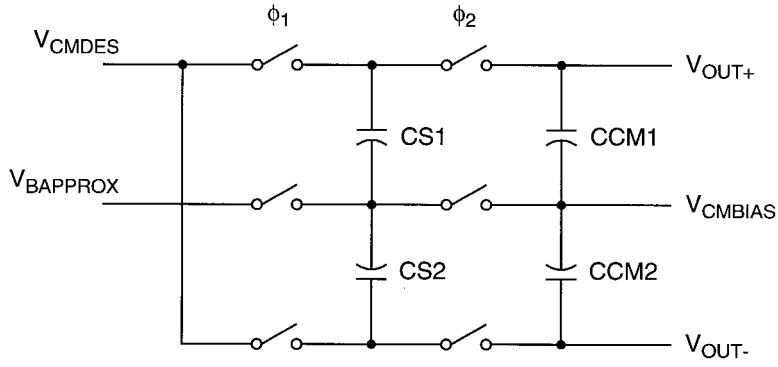


FIGURE 55.36 Switched-capacitor common-mode feedback.

makes up the difference between the common-mode level desired at the amplifier output terminals (V_{CMDES}) and the approximate gate bias required by the common mode current devices ($V_{BAPPROX}$). The former is usually set at mid-supply by a voltage divider, and the latter can be derived from a matched diode-connected device. Since the clocking of this switching network is done synchronously to the amplifier integrator clocking, no charge injection will occur during the sensitive settling process of the amplifier. In order to minimize the charge injection at the clock transitions, capacitors CS1 and CS2 are usually made very small, and therefore dozens of clock cycles may be required for the common-mode bias to settle and the modulator to become operable.

Comparators

The noise shaping mechanism of the modulator feedback loop allows the loop behavior to be tolerant of large errors in circuit behavior at locations closer to the output end of the network. Modulators are generously tolerant of large offset errors in the comparators used in the A/D converter forming the feedback path. For this reason, almost all modulators use simple regenerative latches as comparators. No preamp is generally needed, as the small error from clock kickback can easily be tolerated. Simulations show that offset errors that are even as large as 10% of the reference level will not degrade modulator performance significantly.

The circuit of Fig. 55.37 is typical.[54] This is essentially a latch composed of two cross-connected CMOS inverters, M1–M4. Switch devices M5–M8 will disconnect this network when the clock input is low, and throw the network into a regenerative mode with the rising edge of the clock. The state in which the regeneration will settle may be steered by the relative strengths of the bias current output by devices M9 and M10, which in turn depend on the differential input voltage.

Complete Modulator

Figure 55.38 illustrates a complete second-order, single-bit feedback modulator assembled from the components discussed above.[11] The discrete time integrator gain factors that are derived in Sections 55.2 and 55.3 are realized by appropriate ratios between the integration and sampling capacitors in each stage.

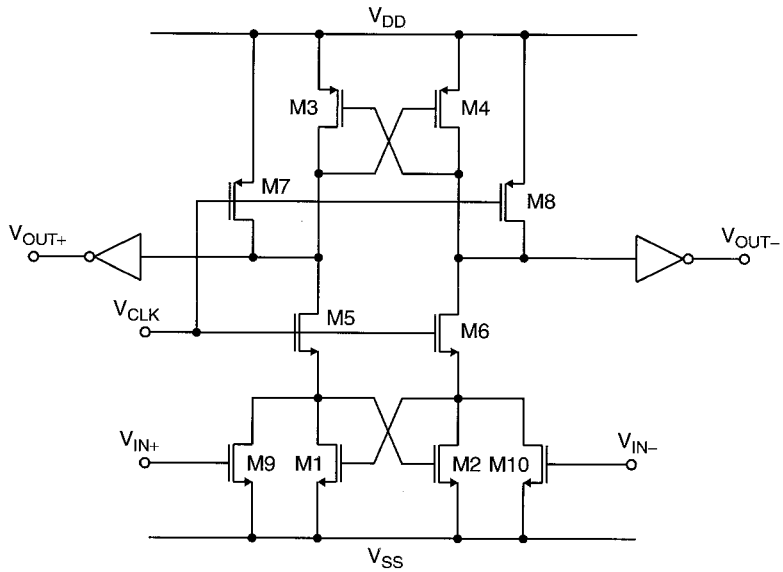


FIGURE 55.37 Typical modulator comparator.

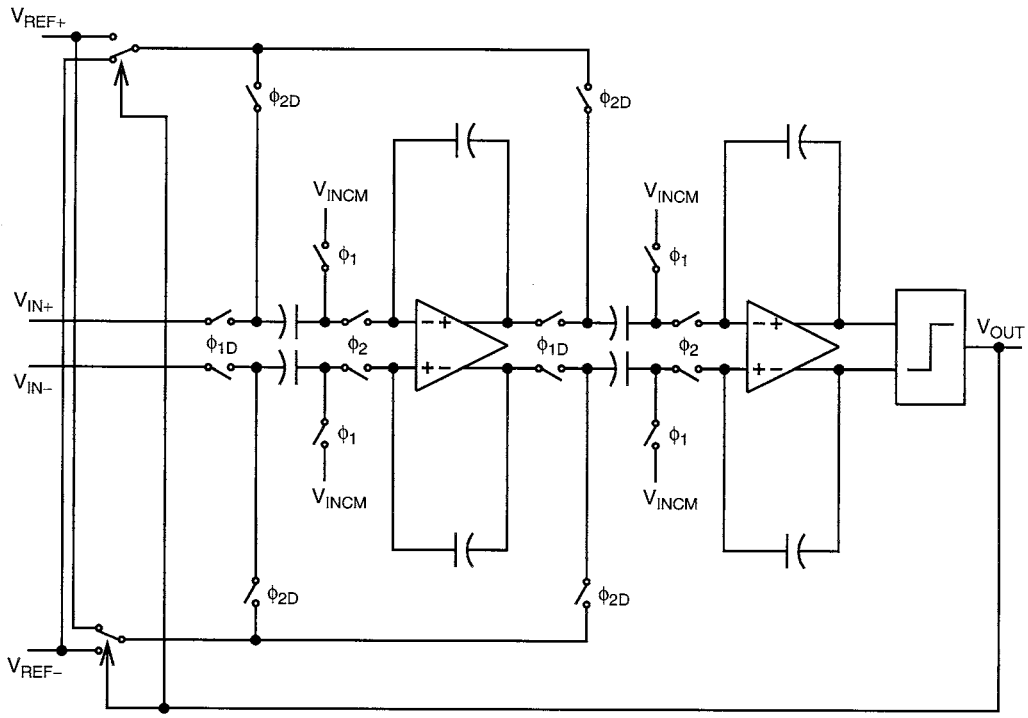


FIGURE 55.38 Complete second-order sigma-delta modulator.

Since the single-bit feedback DAC is only responsible for generating two output levels, it may be implemented by simply switching an applied differential reference voltage V_{REF+} to V_{REF-} in a direct or reversed sense to the sampling capacitor bottom plates during the amplifier integration phase, ϕ_2.

D/A Circuits

For the DAC system shown in Fig. 55.2, the oversampled bit stream is generated by straightforward digital implementations of the modulator signal flow graphs discussed in Section 55.2. The remaining analog components are the low-resolution DAC block and the reconstruction filter. Integrated sigma-delta D/A implementations often employ two-level quantization, and the DAC block may either be designed as charge-based[55] or current-based.[56] Multi-level DAC approaches are also used, but for harmonic content less than about –60 dB below the reference, some form of dynamic element matching must be added, as discussed in Section 55.6.

The charge-based approach for sigma-delta D/A conversion is illustrated in Fig. 55.39, which is similar to the switched-capacitor integrator of Fig. 55.29, but without an analog signal input. As in Fig. 55.38, V_{DAC} may be either polarity of V_{REF} according to the bit value to be converted. Figure 55.40 shows a typical topology for current-based converters. In both cases, the leaky integration function around the amplifier contributes to the first pole of the reconstruction filtering. An efficient combination of the current-based approach and a digital delay line realizing an FIR reconstruction filter is also possible.[57]

Additional reconstruction filtering beyond that provided by in the DAC may also be necessary. This is accomplished using the appropriate analog sampled-data filtering techniques described in Chapter 59 of this section.

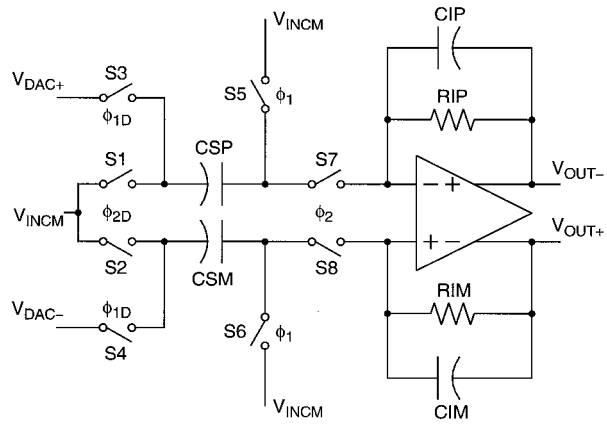


FIGURE 55.39 Charge-based DAC.

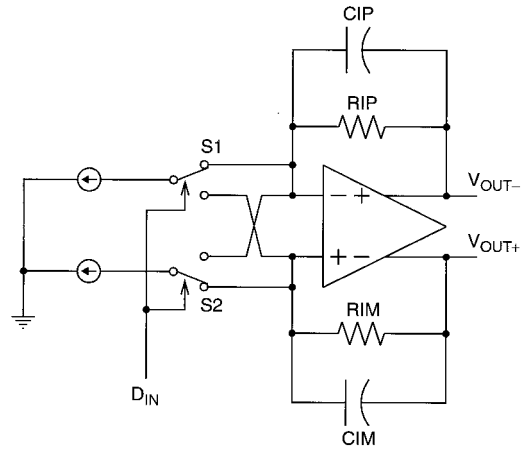


FIGURE 55.40 Current-based DAC.

Continuous-Time Modulators

In general, the amplifiers contained in the switched-capacitor integrators in a sampled-data sigma-delta data converter dissipate the majority of the analog circuit power. Since the integrator sections must settle accurately within each clock period at the oversampled rate, the amplifiers must often be designed with a unity-gain frequency much higher than the oversampled rate; typical unity-gain frequencies are in the hundreds of MHz.

In applications in which dissipating the lowest possible power is important, sigma-delta modulators may also be implemented using continuous-time integrators. In these continuous-time modulators, the analog signal is not sampled until the quantizer at the back of the modulator loop.[58] Owing to the typical means employed for the DAC feedback, continuous-time modulators tend to be more sensitive to sampling clock jitter, but the influences of any aliasing distortion and non-linearity at the sampler take place late in the loop where noise shaping is steepest, and as a consequence the anti-aliasing filter of Fig. 55.1 may often be omitted.[59] The power advantage comes from the relaxed speed requirement of the integrator stages, which now need only have unity-gain frequencies on the order of the oversampled clock frequency.

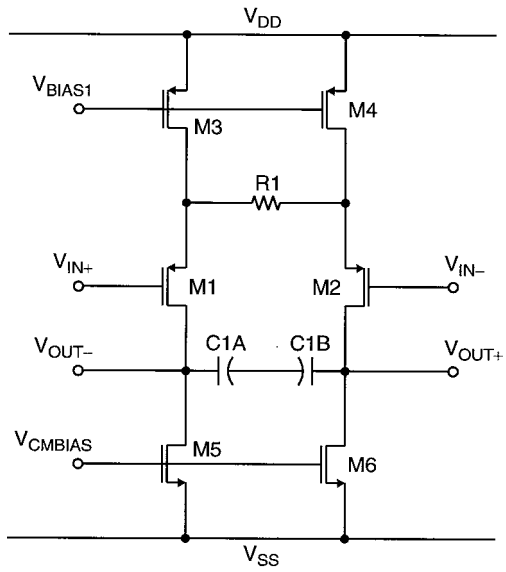


FIGURE 55.41 Gm-C integrator.

Instead of switched-capacitor discrete-time integrators, the continuous-time modulators generally use active Gm-C integrators. Circuits like that shown in Fig. 55.41 are typical.[59] The input differential pair M1 and M2 is degenerated by source resistance R1 to improve linearity. The output analog voltage is developed across capacitor C1, which may be split as shown to place the bottom plate parasitic capacitance at a common-mode node. As the integrator is now unclocked, continuous-time common-mode feedback must be used, as discussed in the literature for continuous time filtering.[60]

55.6 Practical Design Issues

As with any design involving analog components, there are a number of circuit limitations and tradeoffs in sigma-delta data converter design. The design considerations discussed in this section include *kT/C* noise, integrator scaling, amplifier gain, and sampling non-linearity. Also discussed in this section are the techniques of integrator reset and multi-level feedback.

kT/C Noise

In switched-capacitor-based modulators, one fundamental non-ideality associated with using a MOS device to sample a voltage on a capacitor is the presence of a random variation of the sampled voltage after the MOS switch opens.[61-63] This random component has a Gaussian distribution with a variance of *kT/C*, where *k* is Boltzman's constant, *C* is the capacitance, and *T* is the absolute temperature. The variation stems from thermal noise in the resistance of the MOS channel as it is opening. The noise voltage has a mean power of $4kTRB$, where *R* is the channel resistance and *B* is the bandwidth. It is low-pass filtered by its characteristic resistance and the sampling capacitor to an equivalent noise bandwidth of $1/RC$. The total integrated variance will thus be *kT/C*, independent of the resistance of the switch.

If, in the process of developing the integrated signal, a sampling operation on *n* capacitors is used, then since we assume Gaussian noise distribution, the variance of the eventual integrated value will be *nkT/C*. For the case of a fully differential integrator, where a differential signal is sampled onto two sampling capacitors and then transferred to two integration capacitors, *n* is 4. This effect, along with the input referred noise of the amplifier, will limit the achievable noise floor of the modulator. The first stage sampling capacitors must be sized so as to limit this noise contribution to an acceptable level. From this

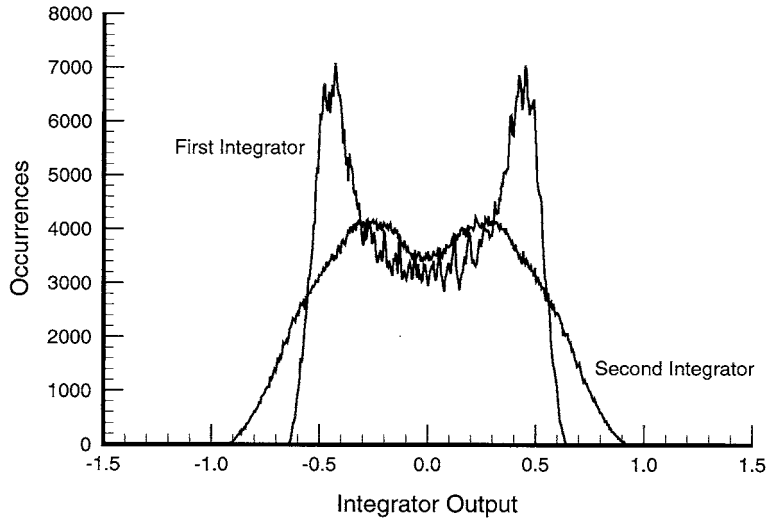


FIGURE 55.42 Integrator output distribution for an eight-level modulator.

starting point, and the capacitive ratios required for realizing the various integrator gains, the remaining capacitor sizes can be determined. The modulator will be much less sensitive to kT/C noise generated in integrators past the first, and the capacitors in theses integrators can be made considerably smaller.

Integrator Gain Scaling

The integration stages in Section 55.2 were discussed as ideal elements, capable of developing any real output voltage. In practice, the output voltage of real integrators is limited to at most the supply voltage of the embedded amplifier. To ensure that this limitation does not adversely affect the modulator performance, a survey of the likely limit of integrator output voltages must be made for a given value of the DAC reference voltage. The modulator may be simulated over a large number of samples with a representative sinusoidal input, and a histogram of all encountered output voltages tabulated. These histograms may be expected to scale linearly with the reference voltage level. In general, this statistical survey will show that a modulator designed to realize the integrator gain constants in the ideal topologies of Sections 55.2 and 55.3 will have different ranges of expected output voltages from each of its integrators. For example, Figs. 55.42 and 55.43 show the simulated output voltages at the two integrators in a second-order modulator with eight-level and two-level feedback, respectively. Since the largest value possible of reference level will generally mean the best available signal-to-noise ratio for a given circuit power consumption, the integrator gain constants may be adjusted from their straightforward values so that the overall modulator transfer function remains the same, but the output voltages are scaled so that no integrator limits the signal swing markedly before the other.[11] Figures 55.44 and 55.45 illustrate the properly scaled second-order modulator examples.

Amplifier Gain

Another mechanism by which the actual characteristic of the integrator circuits fall short of the ideal is the limitation of finite amplifier gain. A study of many simulations of modulators with various amplifier gains[11] has shown that a modulator needs amplifiers with gains about numerically equal to the decimation ratio of the filter that follows it in order to avoid significant noise shaping errors. At least this is the result with perfectly linear amplifiers, and in practice, amplifier gains often need to be at least 10 times this high to avoid distortion in the integrator characteristic due to the non-linearity of the amplifier gain characteristic.

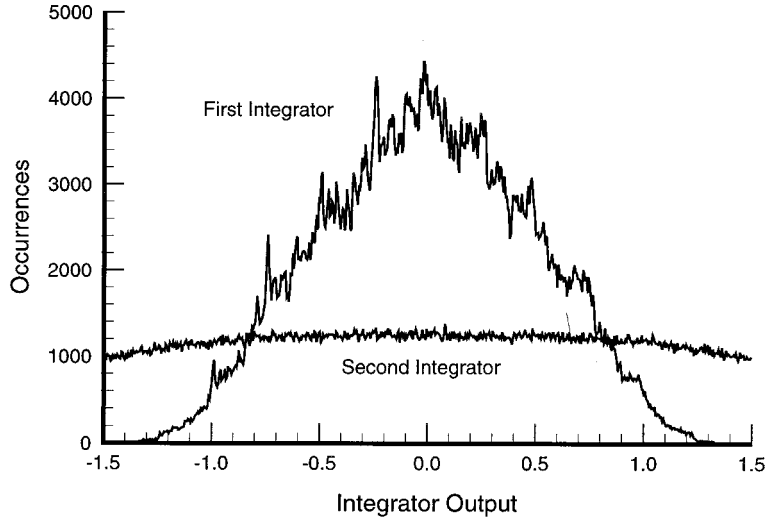


FIGURE 55.43 Integrator output distribution for a two-level modulator.

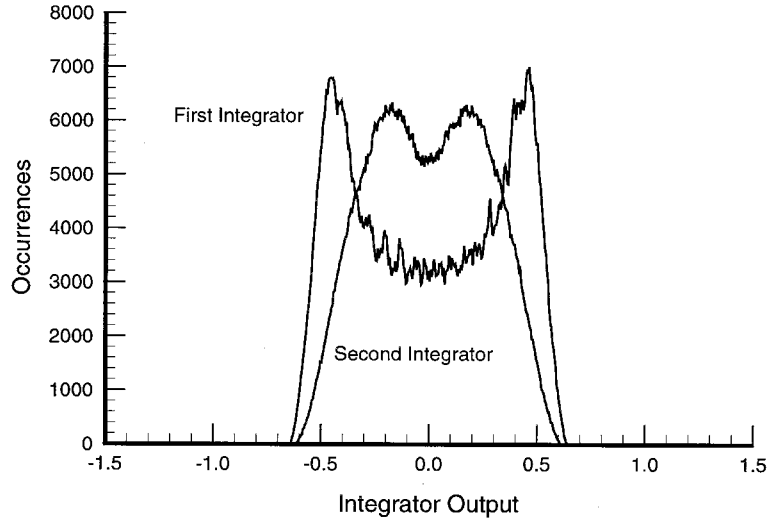


FIGURE 55.44 Integrator output distribution for a scaled eight-level modulator.

One approach used when the simple circuits of Section 55.5 (Operational Amplifiers) do not develop enough gain in a given process is the regulated cascode gain enhancement.[64,65,68] Figure 55.46 illustrates a typical circuit topology. This subcircuit may be substituted for the output common-source amplifier stages in the amplifiers of Figs. 55.33 to 55.35 if the power supply voltage can accommodate its somewhat increased requirement for headroom.

Sampling-Nonlinearity and Reference Corruption

The sigma-delta modulator is remarkably tolerant of most circuit non-idealities past the input sampling network. However, the linearity of the sampling process at the very first input sampling capacitor will be the upper bound for the linearity for the entire modulator. Care must be exercised to ensure the

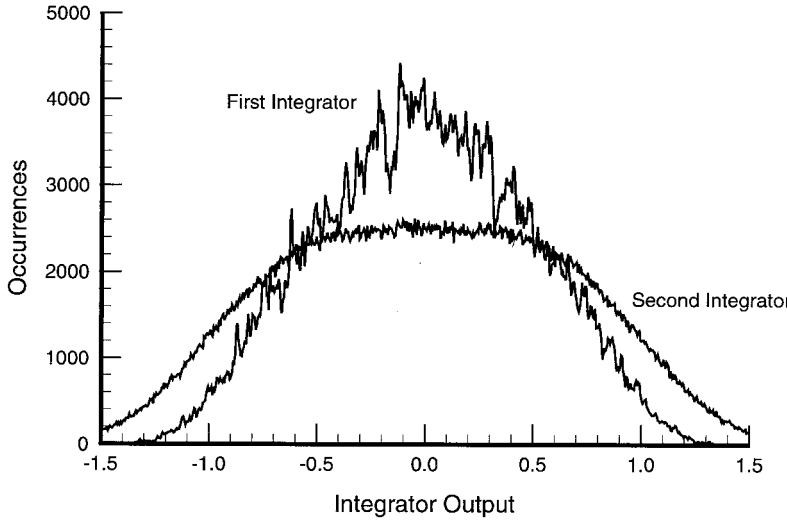


FIGURE 55.45 Integrator output distribution for a scaled two-level modulator.

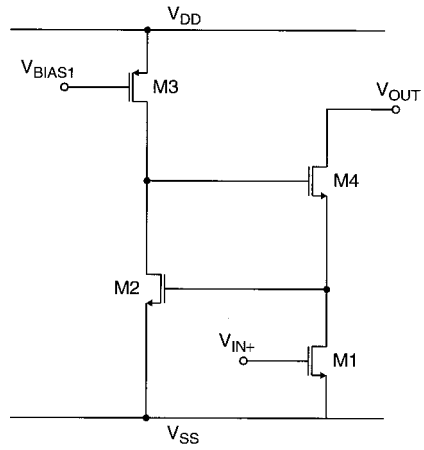


FIGURE 55.46 Regulated cascode topology.

switches are sufficiently large so that the sampled voltage will be completely settled through their non-linear resistance, but not so large that any residual signal dependent clock feedthrough is significant.

Another susceptibility of modulators is to non-linear corruption of the reference voltage. If the digital bit stream output, through a parasitic feedback path either on- or off-chip, can affect the reference voltage sampled during clock phase ϕ_2 in Fig. 55.29, then there will be a term in the output signal dependent on the square of the input voltage. This will distort the noise shaping properties of the modulator and generate second harmonic distortion, even with fully differential circuitry. This is illustrated in the spectrum in Fig. 55.47, which is the output of a modulator having the same conditions as Fig. 55.14, except that a parasitic feedback path is assumed that would change the reference voltage by 1% for the "1" output bits on the previous cycle, relative to its value with "0" output bits. As can be seen by comparison with Fig. 55.14, that the ability of the modulator to shift quantization noise out of the baseband has been greatly compromised, and a prominent second harmonic has been generated. Care must be taken in chip and printed circuit board application design so that the reference voltage remains isolated from the signals carrying the output bit stream.

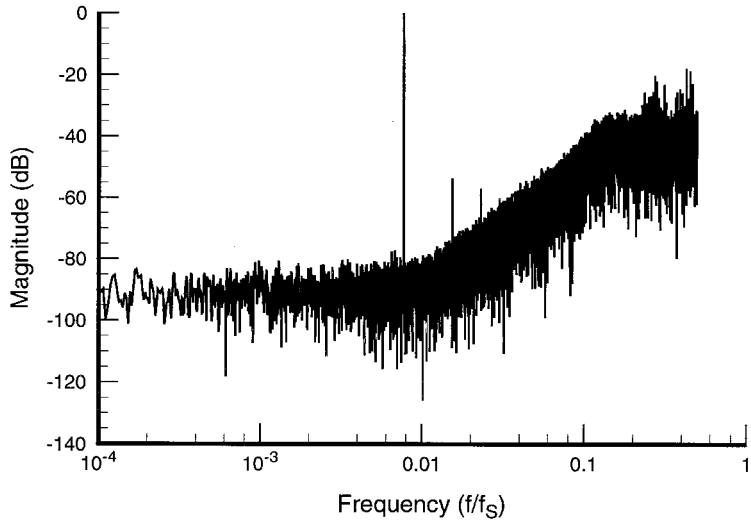


FIGURE 55.47 Output spectrum with reference corruption.

Fully differential circuitry is almost universally employed in integrated VLSI modulators to reduce sampling non-linearity and reference contamination. Even-order non-linearities and common-mode switch feedthrough are cancelled with fully differential circuits, and power supply rejection is greatly improved, leading to more isolated reference potentials. For high-precision modulators, the integrator topology is often changed from that of Fig. 55.29 to Fig. 55.48.[26,51] The input signal and the DAC output voltage are sampled independently during phase ϕ_1, and then both are discharged together into the summing node during ϕ_2. At the expense of additional area for capacitors and higher kT/C noise, this

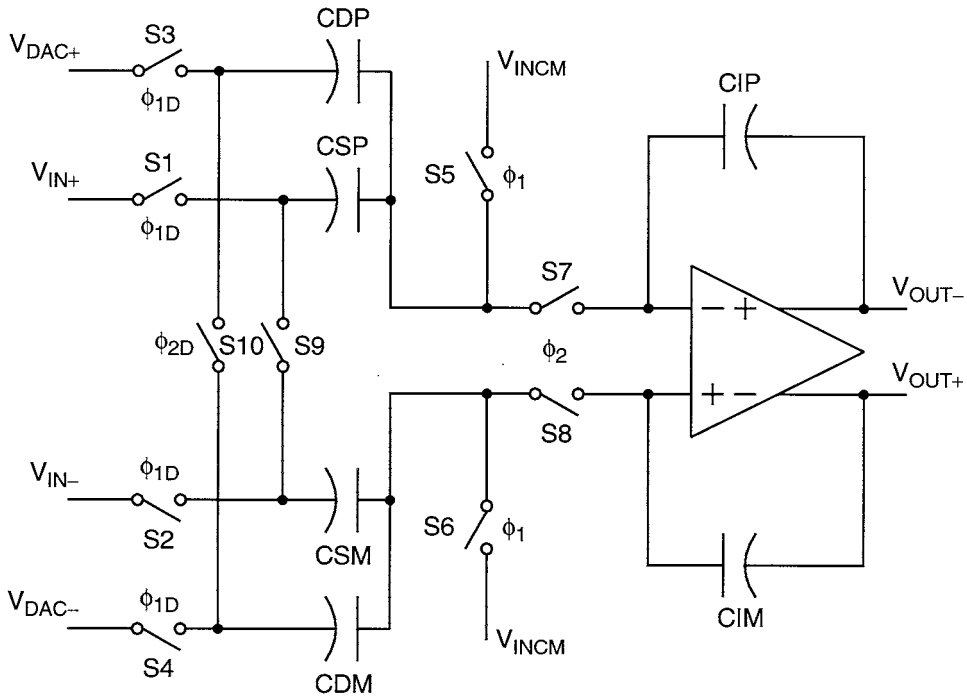


FIGURE 55.48 Integrator with separate DAC feedback capacitor.

arrangement ensures that the same charge is drawn from the reference supply onto the DAC sampling capacitors CDP and CDM and then discharged into the summing node each cycle. Thus, a potentially undesirable mechanism for reference supply loading that is dependent on the output bit history is eliminated.[66]

High-Order Integrator Reset

Although careful design of the loop filter for higher-order modulators, as discussed above in Section 55.3 (High-Order Modulators), will yield a generally stable design, their stability cannot be mathematically guaranteed, as in the case of second-order loops. To protect against the highly undesirable state of low frequency limit cycle oscillations due to an occasional, but improbable, input overload condition, some form of forced integrator reset is sometimes used.[26,51] Generally, these count the consecutive '1' or '0' bits out of the modulator, and close a resetting switch to discharge integration capacitors for a short time if the modulator generates a longer consecutive sequence than normal operation allows. This will naturally interrupt the operation of the modulator, but will only be triggered in the case of pathological input patterns for which linear operation would not necessarily be expected.

Another approach to a stability safety mechanism for higher-order loops is to arrange the scaling of the integrator gains so that they clip against their maximum output voltage swings in a prescribed sequence as the input level rises. The sequence is designed to gradually lower the effective order of the modulator,[59] and return operation to a stable mechanism.

Multi-level Feedback

Expanding the second-order modulator to more than two-level feedback can be accomplished by the circuit in Fig. 55.49. For K-level feedback, $K-1$ latch comparators are arranged in a flash structure as

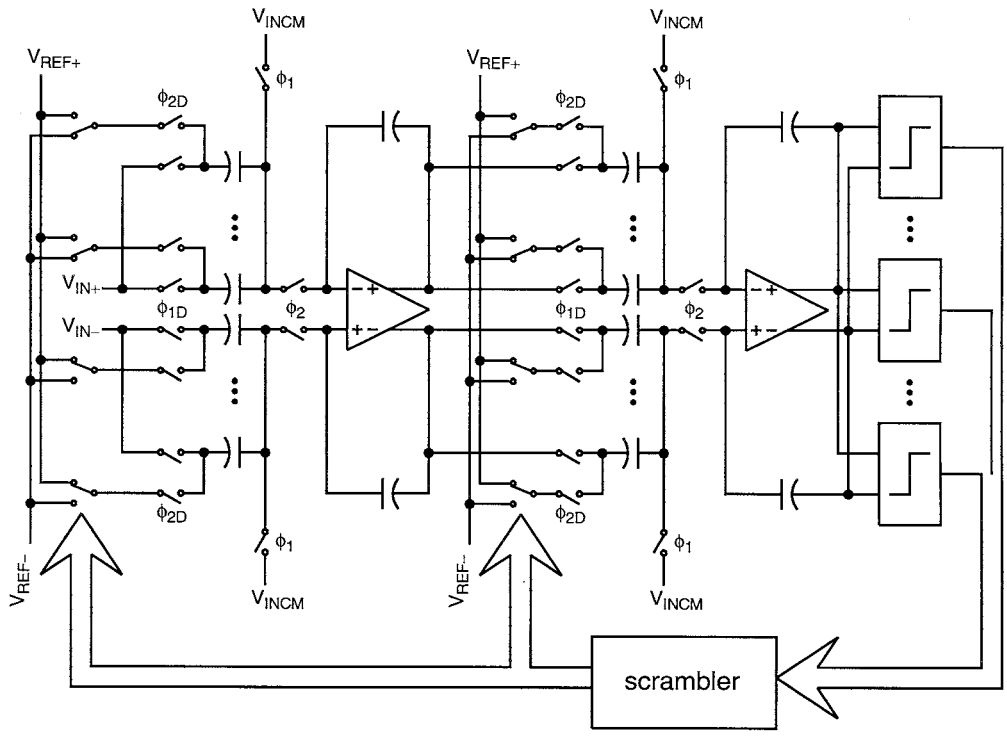


FIGURE 55.49 Multi-bit second-order modulator.

shown on the right. There must be a different offset voltage designed into each of the comparators so that they detect when each quantization level is crossed by the second integrator output. This can be implemented by a resistor string[54,67] or an input capacitive sampling network.[68] The output of the $K-1$ comparators is then a thermometer code representation of the integrator output. This may be translated into binary for the modulator output, but the raw thermometer code is the most convenient to use as a set of feedback signals. They each will drive a switch that will select either V_{REF+} or V_{REF-} to be used as the bottom plate potential for the integrator sampling capacitors. If all sampling capacitors are of equal value, the net charge being integrated will have a term that varies linearly with the quantization level. Each comparator output drives two switches, so there are $2(K-1)$ switches and capacitors in the sampling array.

In any practical integrated structure, even if careful common-centroid layout techniques are used, the precision with which the various capacitors forming the sampling array will be matched is typically limited to 0.1 or 0.2%. As discussed in Section 55.2, this will limit the harmonic distortion that is inherent in the modulator to about –60 dB or higher. However, by varying the assignment of which sampling switch is driven by which comparator output dynamically as the modulator is clocked, much of this distortion may be traded off for white or frequency-shaped noise at the modulator output. This technique is referred to as *dynamic element matching*.

One simple way of implementing dynamic element matching is indicated in Fig. 55.49 with the block labeled "scrambler." This block typically comprises an array of switches that provide a large number of permutations in the way the comparator output lines can be mapped onto sampling switch lines. A multiple-stage butterfly network is one relatively simple approach.[69] The mapping permutation is then changed every modulator clock cycle. Assuming each comparator output will, over a time period less than the final baseband period, end up mapped to all the sampling capacitors, all of the capacitor mismatches will be averaged out. The energy that would be found in input signal harmonics without scrambling will be spread out in some fashion into a higher modulator noise output.

There have been various algorithms published in the literature on how best to control the sequence of mapping perturbations. A simple random sequence will render the mismatch into white noise, increasing the baseband output noise floor.[68,69] More complex sequences relying on a knowledge of the history of quantizer levels are capable of coloring the spread noise so that much of it appears outside the baseband and is suppressed by the following decimation filtering.[70-73]

55.7 Summary

In this chapter, a brief overview of sigma-delta data converters has been presented. Sigma-delta data conversion is a technique that effectively trades speed for resolution. High-linearity data conversion can be accomplished in modern IC processes without expensive device trimming or calibration. For a far more detailed treatment of this topic, refer to Norsworthy, Schreier, and Temes.[2] For a compilation of some of the seminal papers that helped establish sigma-delta modulation as a mainstream technique, refer to Candy and Temes.[1]

References

1. J. Candy and G. Temes, *Oversampling Delta-Sigma Data Converters*, IEEE Press, 1992.
2. S. Norsworthy, R. Schreier, and G. Temes, *Delta-Sigma Data Converters: Theory, Design, and Simulation*, IEEE Press, 1996.
3. C. Cutler, Transmission systems employing quantization, U.S. Patent 2,927,962, Mar. 8, 1960.
4. H. Spang III and P. Schultheiss, Reduction of quantizing noise by use of feedback, *IRE Trans. on Communication Systems*, pp. 373–380, Dec. 1962.
5. H. Inose and Y. Yasuda, A unity bit coding method by negative feedback, *Proc. IEEE*, vol. 51, pp. 1524–1535, Nov. 1963.

6. S. Tewksbury and R. Hallock, Oversampled, linear predictive and noise-shaping coders of order N>1, *IEEE Trans. on Circuits and Systems*, vol. CAS-25, pp. 436–447, July 1978.
7. J. Candy, A use of limit cycle oscillations to obtain robust analog-to-digital converters, *IEEE Trans. on Communications*, vol. COM-22, pp. 298–305, Mar. 1974.
8. H. Fiedler and B. Hoefflinger, A CMOS pulse density modulator for high-resolution A/D converters, *IEEE J. of Solid-State Circuits*, vol. SC-19, pp. 995–996, Dec. 1984.
9. B. Leung, R. Neff, P. Gray, and R. Brodersen, Area-efficient multichannel oversampled PCM voice-band coder, *IEEE J. of Solid-State Circuits*, vol. SC-23, pp. 1351–1357, Dec. 1988.
10. J. Candy, A use of double integration in sigma delta modulation, *IEEE Trans. on Communications*, vol. COM-33, pp. 249–258, Mar. 1985.
11. B. Boser and B. Wooley, The design of sigma-delta modulation analog-to-digital converters, *IEEE J. of Solid-State Circuits*, vol. 23, pp. 1298–1308, Dec. 1988.
12. V. Friedman, D. Brinthaupt, D. Chen, T. Deppa, J. Elward, E. Fields, J. Scott, and T. Viswanathan, A dual-channel voice-band PCM codec using ΣΔ modulation technique, *IEEE J. of Solid-State Circuits*, vol. 24, pp. 274–280, Apr. 1989.
13. W. Bennett, Spectra of quantized signals, *Bell System Tech. Journal*, vol. 27, pp. 446–472, 1948.
14. B. Widrow, Statistical analysis of amplitude quantized sampled-data systems, *Trans. AIEE*, vol. 79, pp. 555–568, Jan. 1961.
15. R. Gray, Oversampled sigma-delta modulation, *IEEE Trans. on Communications*, vol. COM-35, pp. 481–489, May 1987.
16. J. Candy and O. Benjamin, The structure of quantization noise from sigma-delta modulation, *IEEE Trans. on Communications*, vol. COM-29, pp. 1316–1323, Sept. 1981.
17. B. Boser and B. Wooley, Quantization error spectrum of sigma-delta modulators, *1988 IEEE Intnl. Symp. on Circuits and Systems,* pp. 2331–2334, 1988.
18. R. Gray, Quantization noise spectra, *IEEE Trans. Information Theory,* vol. 36, pp. 1220–1244, Nov. 1990.
19. L. Williams and B. Wooley, A third-order sigma-delta modulator with extended dynamic range, *IEEE J. of Solid-State Circuits,* vol. 29, Mar. 1994.
20. B. Brandt, D. Wingard, and B. Wooley, Second-order sigma-delta modulation for digital-audio signal acquisition, *IEEE J. of Solid-State Circuits,* vol. 26, pp. 618–627, Apr. 1991.
21. J. Everard, A single-channel PCM codec, *IEEE J. of Solid-State Circuits,* vol. SC-14, pp. 25–37, Feb. 1979.
22. M. Hauser, P. Hurst, and R. Brodersen, MOS ADC-filter combination that does not require precision analog components, *ISCC Dig. Tech. Papers,* pp. 80–82, Feb. 1985.
23. S. Norsworthy, Effective dithering of sigma-delta modulators, *Proc. of the 1992 IEEE Intnl. Symp. on Circuits and Systems,* pp. 1304–1307, May 1992.
24. D. Welland, B. Del Signore, E. Swanson, T. Tanaka, K. Hamashita, S. Hara, and K. Takasuka, A stereo 16-bit detla-sigma A/D converter for digital audio, *J. Audio Engineering Society,* vol. 37, pp. 476–486, June 1989.
25. K. Chao, S. Nadeem, W. Lee, and C. Sodini, A higher order topology for interpolative modulators for oversampling A/D converters, *IEEE Trans. on Circuits and Systems,* vol. 37, pp. 309–318, Mar. 1990.
26. R. Adams, P. Ferguson, A. Ganesan, S. Vincelette, A. Volpe, and R. Libert, Theory and practical implementation of a fifth-order sigma-delta A/D converter, *J. Audio Eng. Soc.,* vol. 39, pp. 515–528, July/Aug. 1991.
27. R. Schreier, An empirical study of high-order single-bit delta-sigma modulators, *IEEE Trans. on Circuits and Systems. II. Analog and Digital Signal Processing,* vol. 40, Aug. 1993.
28. A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ, 1989.
29. Y. Matsuya, K. Uchimura, A. Iwata, T. Kobayashi, M. Ishikawa, and T. Yoshitome, A 16-bit over-sampling A-to-D conversion technology using triple-integration noise shaping, *IEEE J. of Solid-State Circuits,* vol. SC-22, pp. 921–929, Dec. 1987.

30. L. Longo and M. Copeland, A 13 bit ISDN-band oversampling ADC using two-stage third order noise shaping, *IEEE 1988 Custom Integrated Circuits Conference,* pp. 21.2.1–4, 1988.
31. L. Williams and B. Wooley, Third-order cascaded sigma-delta modulators, *IEEE Trans. on Circuits and Systems,* vol. 38, pp. 489–498, May 1991.
32. P.-W. Wong and R. Gray, Two stage sigma-delta modulation, *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. 38, pp. 1937–1952, Nov. 1990.
33. L. Longo and B.-R. Horng, A 15b 30kHz bandpass sigma-delta modulators, *1993 IEEE Intnl. Solid-State Circuits Conf.,* pp. 226–227, Feb. 1993.
34. R. Schreier and W. M. Snelgrove, Decimation for bandpass sigma-delta analog-to-digital conversion, *1990 IEEE Intnl. Symp. on circuits and Systems,* vol. 3, pp. 1801–1804, May 1990.
35. R. Gregorian and G. Temes, *Analog MOS Integrated Circuits for Signal Processing,* John Wiley & Sons, 1986.
36. D. Goodman and M. Carey, Nine digital filters for decimation and interpolation, *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. ASSP-25, pp. 126–126, Apr. 1977.
37. R. Crochiee and L. Rabiner, *Multirate Digital Signal Processing,* Prentice-Hall, Englewood Cliffs, NJ, 1983.
38. E. Hogenauer, An economical class of digital filters for decimation and interpolation, *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. ASSP-29, pp. 155–162, Apr. 1981.
39. S. Chu and C. Burrus, Multirate filters designs using comb filters, *IEEE Trans. on circuits and Systems,* vol. CAS-31, pp. 913–924, Nov. 1984.
40. J. Candy, Decimation for sigma delta modulation, *IEEE Trans. on Communications,* vol. COM-34, pp. 72–76, Jan. 1986.
41. B. Brandt and B. Wooley, A low-power, area-efficient digital filter for decimation and interpolation, *IEEE J. of Solid-State Circuits,* vol. 29, pp. 679–687, June 1994.
42. T. Choi, et al., High-frequency CMOS switched-capacitor filters for communications applications, *IEEE J. of Solid-State Circuits,* vol. 15, pp. 929–938, Dec. 1980.
43. W. C. Black, et al., A high-performance low power CMOS channel filter, *IEEE J. of Solid-State Circuits,* vol. 15, pp. 929–938, Dec. 1980.
44. D. Senderowitz, et al., A family of differential NMOS analog circuits for a PCM codec filter chip, *IEEE J. of Solid-State Circuits,* vol. 17, pp. 1014–1023, Dec. 1982.
45. R. Castello and P. R. Gray, A high-performance micropower switched-capacitor filter, *IEEE J. of Solid-State Circuits,* vol. 20, pp. 1122–1132, Dec. 1985.
46. T. B. Cho and P. R. Gray, A 10b, 20 Msample/s, 35 mW pipeline A/D converter, *IEEE J. of Solid-State Circuits,* vol. 30, pp. 166–172, Mar. 1995.
47. K. Nagaraj et al., Switched-capacitor integrator with reduced sensitivity to amplifier gain, *Electronics Letters,* vol. 22, p. 1103, Oct. 1986.
48. K. Huag, G. C. Temes, and L. Martin, Improved offset-compensation scheme for SC circuits, *1984 IEEE Intnl. Symp. on Circuits and Systems,* pp. 1054–1057, 1984.
49. P. J. Hurst and W. J. McIntyre, Double sampling on switched-capacitor delta-sigma A/D converters, *1990 IEEE Symp. on Circuits and Systems,* pp. 902–905, May 1990.
50. D. Senderowitz et al., Low voltage double-sampled sigma-delta converters, *IEEE J. of Solid-State Circuits,* vol. 32, pp. 1907–1919, Dec. 1997.
51. P. Furguson, et al., An 18b, 20kHz dual sigma-delta A/D converter, *1991 IEEE Intnl. Solid-State Circuits Conf.,* pp. 68–69, Feb. 1991.
52. P. R. Gray and R. G. Meyer, MOS operational amplifier design — A tutorial overview, *IEEE J. of Solid-State Circuits,* vol. 17, pp. 969–981, Dec. 1982.
53. A. Abidi, C. Viswanathan, J. Wu, and J. Wikstrom, Flicker noise in CMOS: A unified model for VLSI processes, *1987 Symp. VLSI Technology,* pp. 85–86, May 1987.
54. A. Yukawa, A CMOS 18-bit high speed A/D converter IC, *IEEE J. of Solid-State Circuits,* vol. 20, pp. 775–779, June 1985.

55. B. Kup, E. Dijkmans, P. Naus, and J. Sneep, A bit-stream digital-to-analog converter with 18-b resolution, *IEEE J. of Solid-State Circuits,* vol. 26, pp. 1757–1763, Dec. 1991.
56. R. Adams, K. Q. Nguyen, and K. Sweetland, A 113-dB SNR oversampled DAC with segmented noise-shaped scrambling, *IEEE J. of Solid-State Circuits,* vol. 33, pp. 1871–1878, Dec. 1998.
57. D. Su and B. Wooley, A CMOS oversampling D/A converter with a current-mode semidigital reconstruction filter, *IEEE J. of Solid-State Circuits,* vol. 28, pp. 1224–1233, Dec. 1993.
58. R. Schreier and B. Zhang, Delta-sigma modulators employing continuous-time circuitry, *IEEE Trans. on Circuits and Systems. I. Fundamental Theory and Applications,* vol. 44, pp. 324–332, Apr. 1996.
59. E. J. van der Zwan and E. C. Dijkmans, A 0.2-mW CMOS sigma-delta modulator for speech coding with 80dB dynamic range, *IEEE J. of Solid-State Circuits,* vol. 31, pp. 1873–1880, Dec. 1996.
60. Y.P. Tsividis, Integrated continuous-time filter design — an overview, *IEEE J. of Solid-State Circuits,* vol. 29, pp. 166–176, Mar. 1994.
61. K. C. Hsieh, Noise limitations in switched-capacitor filters, Ph.D. dissertation, Univ. California, Berkeley, Dec. 1981.
62. C. Gobet and A. Knob, Noise analysis of switched-capacitor networks, *1981 Intnl. Symp. on Circuits and Systems,* Apr. 1981.
63. C. Gobet and A. Knob, Noise generated in switched-capacitor networks, *Electronics Letters,* vol. 19, no. 19, 1980.
64. E. Säckinger and W. Guggenbühl, A high-swing, high-impedance MOS cascode circuit, *IEEE J. of Solid-State Circuits,* vol. SC-25, pp. 289–298, Feb. 1990.
65. K. Bult and G. J. G. M. Geelen, A fast-settling CMOS opamp for SC circuits with 90-dB DC gain, *IEEE J. of Solid-State Circuits,* vol. SC-25, no. 6, pp. 1379–1384, Dec. 1990.
66. D. Ribner, R. Baertsch, S. Garverick, D. McGrath, J. Krisciunas, and T. Fuji, A third-order multistage sigma-delta modulator with reduced sensitivity to nonidealities, *IEEE J. of Solid-State Circuits,* vol. 26, pp. 1764–1774, Dec. 1991.
67. B. Brandt and B. Wooley, A 50-MHz multibit sigma-delta modulator for 12-b 2-MHz A/D conversion, *IEEE J. of Solid-State Circuits,* vol. 26, pp. 1746–1756, Dec. 1991.
68. J. Fattaruso et al., Self-calibration techniques for a second-order multibit sigma-delta modulator, *IEEE J. of Solid-State Circuits,* vol. 28, pp. 1216–1223, Dec. 1993.
69. L. Carley, A noise-shaping coder topology for 15+ bit converters, *IEEE J. of Solid-State Circuits,* vol. 24, pp. 267–273, Apr. 1989.
70. F. Chen and B. Leung, A high resolution multibit sigma-delta modulator with individual level averaging, *IEEE J. of Solid-State Circuits,* vol. 30, pp. 453–460, Apr. 1995.
71. T. Kwan, R. Adams, and R. Libert, A stereo multi-bit $\Sigma\Delta$ D/A with asynchronous master-clock interface, *IEEE J. of Solid-State Circuits,* vol. 31, pp. 1881–1887, Dec. 1996.
72. B. Leung and S. Sutarja, Multibit Σ-Δ A/D converter incorporating a novel class of dynamic element matrching techniques, *IEEE Trans. on Circuits and Systems. II. Analog and Digital Signal Processing,* vol. 39, pp. 35–51, Jan. 1992.
73. L. Williams III, An audio DAC with 90 dB linearity using MOS to metal-metal charge transfer, *1998 IEEE Intnl. Solid-State Circuits Conf.,* pp. 58–59, Feb. 1998.

Steyaert, M., Borremans, M., Janssens, J., De Muer, B.
"RF Communication Circuits"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

56

RF Communication Circuits

Michiel Steyaert
Marc Borremans
Johan Janssens
Bram De Muer
Katholieke Universiteit Leuven, ESAT-MICAS

56.1 Introduction

A few years ago, the world of wireless communications and its applications started to grow rapidly. The main cause for this event was the introduction of digital coding and digital signal processing in wireless communications. This digital revolution is driven by the development of high-performance, low-cost, CMOS technologies that allow for the integration of an enormous amount of digital functions on a single die. This allows, in turn, for the use of sophisticated modulation schemes, complex demodulation algorithms, and high-quality error detection and correction systems, resulting in high-performance lossless communication channels.

Today, the digital revolution and the high growth of the wireless market also bring many changes to the analog transceiver front-ends. The front-ends are the interface between the antenna and the digital modem of the wireless transceiver. They have to detect very weak signals (μV) which come in at a very high frequency (1 to 2 GHz) and, at the same time, they have to transmit at the same high-frequency high power levels (up to 2 W). This requires high-performance analog circuits, like filters, amplifiers, and mixers which translate the frequency bands between the antenna and the A/D-conversion and digital signal processing. Low cost and a low power consumption are the driving forces and they make the analog front-ends the bottleneck for future RF design. Both low cost and low power are closely linked to the trend toward full integration. An even further level of integration renders significant space, cost, and power reductions. Many different techniques to obtain a higher degree of integration for receivers, transmitters, and synthesizers have been presented over the past years.[1-3] This chapter introduces and analyzes some advantages and disadvantages and their fundamental limitations.

Parallel to the trend to further integration, there is the trend to the integration of RF circuitry in CMOS technologies. The mainstream use for CMOS technologies is the integration of digital circuitry. The use of these CMOS technologies for high-performance analog circuits yields, however, many benefits. The technology is, of course — if used without any special adaptations toward analog design — cheap. This is especially true if one wants to achieve the ultimate goal of full integration: the complete transceiver system on a single chip, with both the analog front-end and the digital demodulator implemented on the same die. This can only be achieved in either a CMOS or a BiCMOS process. BiCMOS has better devices for analog design, but its cost will be higher, not only due to the higher cost per area, but also due to the larger area that will be needed for the digital part. Plain CMOS has the extra advantage that the performance gap between devices in BiCMOS and nMOS devices in deep sub-micron CMOS, and even nMOS devices in the same BiCMOS process, is becoming smaller and smaller due to the much higher investments in the development of CMOS than bipolar. The f_T's of the nMOS devices are getting close to the f_T's of npn devices.

Although some research had been done in the past on the design of RF in CMOS technologies,[4] it is only in the few years that real attention has been given to its possibilities.[5,6] Today several research groups at universities and in industry are researching this topic.[2,3,7,9] As bipolar devices are inherently better than CMOS devices, RF CMOS is by some seen as a possibility for only low-performance systems, with reduced specification (like ISM),[8,10] or that the CMOS processes need adaptations, like substrate etching under inductors.[7] Others feel, however, that the benefits of RF CMOS can be much bigger and that it will be possible to use plain deep sub-micron CMOS for the full integration of transceivers for high-performance applications, like GSM, DECT, and DCS 1800.[2,3] First, this chapter analyzes some trends, limitations, and problems in technologies for high-frequency design. Second, the down-converter topologies and implementation problems are addressed. Third, the design and trends toward fully integrated low-phase noise PLL circuits are discussed. Finally, the design of fully integrated up-converters is studied.

56.2 Technology

Active Devices

Due to the never-ending progress in technology and the requirement to achieve a higher degree of integration for DSP circuits, sub-micron technologies are nowadays considered standard CMOS technologies. The trend is even toward deep sub-micron technologies (e.g., transistor lengths of 0.1 μm. Using the square law relationship for MOS transistors to calculate the f_t of a MOS device no longer holds, due to the high electrical fields. Using a more accurate model, which includes the mobility degradation due to the high electrical fields, results in

$$f_t = \frac{g_m}{2\pi C_{gs}} = \frac{\mu}{2\pi 2/3L^2} \frac{\left(V_{gs} - V_t\right)}{\left(1 + 2\left(q + \frac{\mu}{v_{\max} L}\right)\left(V_{gs} - V_t\right)\right)} \tag{56.1}$$

Hence, by going to deep sub-micron technologies, the square law benefit in L for speed improvement drastically reduces due to the second term in the denominator of Eq. 56.1. Even for very deep sub-micron technologies, the small signal parameter g_m has no square law relationship anymore:

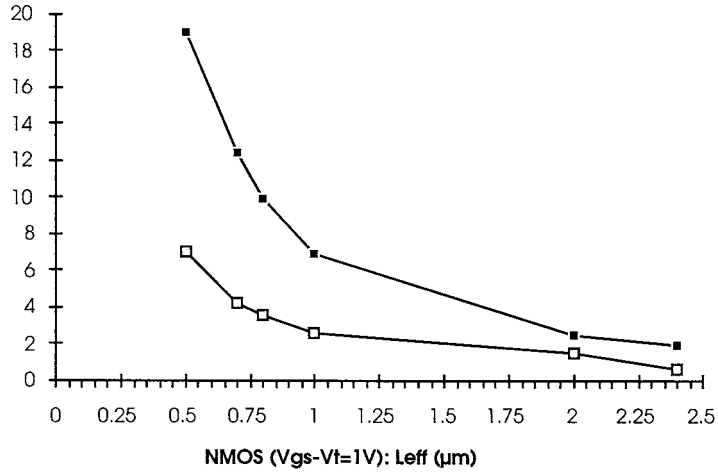


FIGURE 56.1 Comparison of f_t and f_{max}.

$$g_m = \frac{\mu C_{ox} W\left(V_{gs} - V_t\right)}{L\left(1 + 2\left(\theta + \frac{\mu}{v_{\max} L}\right)\left(V_{gs} - V_t\right)\right)} \tag{56.2}$$

with transistor lengths smaller than approximately

$$L < \frac{\mu}{v_{\max}} \frac{1}{\frac{1}{2\left(V_{gs} - V_t\right)} - q} \approx 0.12\,\mu\text{m} \tag{56.3}$$

with $\mu/v_{max} = 0.3$, $V_{gs} - V_t = 0.2$ (boundary of strong inversion), and $\theta = 0.06$, the transistor has only the weak inversion and the velocity saturation area. This will result in even higher biasing currents in order to achieve the required g_m and will result in higher distortion and intermodulation components, which will be further discussed in the tradeoff of low-noise amplifier designs.

Furthermore, the speed increase of deep sub-micron technologies is reduced by the parasitic capacitance of the transistor, meaning the gate-drain overlap capacitances and drain-bulk junction capacitances. This can clearly be seen in Fig. 56.1 in the comparison for different technologies of the f_t and the f_{max} defined as the 3-dB point of a diode-connected transistor.[11] The f_{max} is more important because it reflects the speed limitation of a transistor in a practical configuration. As can be seen, the f_t rapidly increases; but for real circuit designs (f_{max}), the speed improvement is only moderate.

Passive Devices

The usability of a process for RF design does not only depend on the quality of the active devices, but also, more and more, on the availability of good passive devices. The three passive devices (resistors, capacitors, and inductors) will be discussed.

Low-ohmic resistors are available today in all CMOS technologies and their parasitic capacitance is such that they allow for more than high enough bandwidth (i.e., more than 2 to 3 GHz). A more

important passive device is the capacitor. In RF circuits, capacitors can be used for ac-coupling. This allows dc-level shifting between different stages, resulting in a more optimal design of each stage and in the ability to use lower power supply voltages. The quality of a capacitor is mainly determined by the ratio between the capacitance value and the value of the parasitic capacitance to the substrate. Too high a parasitic capacitor loads the transistor stages, thus reducing their bandwidth, and it causes an inherent signal loss due to a capacitive division. Capacitors with ratios lower than 8 are, as a result, difficult to use in RF circuit design as coupling devices.

The third passive device, the inductor, is gaining more and more interest in RF circuit design on silicon. The use of inductors allows for a further reduction of the power supply voltage and for compensation of parasitic capacitances by means of resonance, resulting in higher operating frequencies. The problem is that the conducting silicon substrate under a spiral inductor reduces the quality of the inductor. Losses occur due to capacitive coupling to the substrates, and eddy currents induced in the substrate will also result in losses and in a reduction of the effective inductance value. This problem can be circumvented by using extra processing steps that etch away the substrate under the spiral inductor,[23] having the large disadvantage that it eliminates all the benefits of using a standard CMOS process. It is therefore important that in CMOS, spiral inductors are used without any process changes and that their losses are accurately modeled. In Ref. 12, it is shown that spiral inductors can be accurately modeled and that they can be used in CMOS RF circuit design. As an example, Section 56.4 discusses all the different possibilities for the use of inductors in the design of VCOs. It shows that high-performance VCOs can be integrated with spiral inductors, even on lossy substrates, without requiring any external component.

56.3 The Receiver

Receiver Topologies

The heterodyne or IF receiver is the best known and most frequently used receiver topology. In the IF receiver, the wanted signal is down-converted to a relatively high intermediate frequency. A high quality passive bandpass filter is used to prevent a mirror signal to be folded upon the wanted signal on the IF frequency. Very high performances can be achieved with the IF receiver topology, especially when several IF stages are used (e.g., 900 MHz to 300 MHz, 300 MHz to 70 MHz, 70 MHz to 30 MHz, 30 MHz to 10 MHz). The main disadvantage of the IF receiver is the poor degree of integration that can be achieved as every stage requires going off-chip and requires the use of a discrete bandpass filter. This is both costly (the cost of the discrete filters and the high pin-count for the receiver chip) and power consuming (often the discrete filters have to be driven by a 50-Ω signal source).

The homodyne or zero-IF receiver, introduced as an alternative to the IF receiver, can achieve a much higher degree of integration. The zero-IF receiver uses a direct, quadrature down-conversion of the wanted signal to the baseband. In this case, the wanted signal has itself as mirror signal and sufficient mirror signal suppression can therefore be achieved, even with a limited quadrature accuracy (e.g., 3 degrees phase accuracy and 1-dB amplitude accuracy). Theoretically, there is thus no discrete high-frequency bandpass filter required in the zero-IF receiver, allowing in this way the realization of a fully integrated receiver. The limited performance of the LNA and the mixers reveals, however, that — although not for mirror signal suppression — a high-frequency bandpass filter is still required. The reason why LNAs and mixers require bandpass filtering and how this can be prevented, is explained later.

In the zero-IF receiver, down-conversion can be performed in a single stage (e.g., directly from 900 MHz to the baseband), giving large benefits toward full integration, low cost, and low power consumption.[13] The problem with the zero-IF receiver, however, is its poor performance compared to IF receivers. The zero-IF receiver is intrinsically very sensitive to parasitic baseband signals like dc-offset voltages and crosstalk products caused by RF and LO self-mixing. It is precisely these drawbacks that have kept the zero-IF receiver from being used on a large scale in new wireless applications. The use of the zero-IF receiver has therefore been limited to either low-performance applications like pagers and ISM[10] or as a second stage in a combined IF–zero-IF receiver topology.[14,15] It has, however, been shown

that by using dynamic non-linear dc-correction algorithms, implemented in the DSP, the zero-IF topology can be used for high-performance applications like GSM and DECT.[1,16]

In recent years, new receiver topologies, like the quasi-IF receiver[3] and the low-IF receiver[2] have been introduced for use in high-performance applications. The quasi-IF receiver uses a quadrature down-conversion to an IF frequency, followed by a further quadrature down-conversion to the baseband. The channel selection is done with the second local oscillator on the IF frequency, giving the advantage that a fixed-frequency first local oscillator can be used. The disadvantages of the quasi-IF receiver are that, with a limited accuracy of the first quadrature down-converter (e.g., a phase error of 3 degrees), the mirror signal suppression is not good enough and an HF filter that improves the mirror signal suppression is still necessary. A second disadvantage is that a high IF is required in order to obtain a high enough ratio between the IF frequency and the full band of the application. Otherwise, the tunability of the second VCO has to be too large. Unfortunately, a high IF requires a higher power consumption. Moreover, the first stage of mixers cannot be true down-conversion mixers in the sense that they still need to have a relatively high output bandwidth. To conclude, multi-stage topologies inherently require more power.

The low-IF receiver performs a down-conversion from the antenna frequency directly down to — as the name already indicates — a low IF (i.e., in the range a few 100 kHz).[2] Down-conversion is done in quadrature and the mirror signal suppression is performed at low frequency, after down-conversion, in the DSP. The low-IF receiver topology is thus closely related to the zero-IF receiver. It can be fully integrated (it does not require an HF mirror signal suppression filter) and uses a single-stage direct-down-conversion. The difference is that the low-IF does not use baseband operation, resulting in a total immunity to parasitic baseband signals, resolving in this way the main disadvantage of the zero-IF receiver. The drawback is that the mirror signal is different from the wanted signal in the low-IF receiver topology; but by carefully choosing the IF frequency, an adjacent channel with low signal levels can be selected for which the typical mirror signal suppression (i.e., a phase accuracy of 3 degrees) is sufficient.

Full Integration

With newly developed receiver topologies such as the zero-IF receiver and the low-IF receiver, the need disappears for the use of external filters that suppress the mirror signal (see previous section). This does not mean, however, that there would not be any HF filtering required anymore. Filtering before the LNA is, although not for mirror signal suppression, still necessary to suppress the blocking signals. Moreover, between the LNA and the mixer, filtering may be necessary in order to suppress second and third harmonic distortion products that are introduced by the LNA. Due to the use of a switching down-converter or the non-linearities of the mixer and local oscillator harmonics, these distortion products will be down-converted to the same frequency as the wanted signal. The latter problem can be eliminated by using either a very good blocking filter before the LNA (resulting in small signals after the LNA) or by using a highly linear LNA. The use of linear down-converters (i.e., based on the multiplication with a sinusoidal local oscillator signal) reduces of course the problem as well.

In mobile communications systems, very high, out-of-band signals may be present. In order to prevent saturation of the LNA, these signals must be suppressed with an HF filter that passes only the signals in the band of the application. In the GSM system, for example, the ratio between the largest possible out-of-band signal and the lowest detectable signal is 107 dB. Without a blocking filter, the LNA and the mixer must be able to handle this dynamic range. For the LNA, this means that the input should be able to handle an input signal of 0 dBm (i.e., the –1 dB compression point P_{-1dB} should be about 0 dBm), while having a noise figure of 6 dB. Consequently, this means that the IP3 value should be +10 dBm (IP3 ≈ P_{-1dB} + 10.66 dB). The IMFDR3 (intermodulation free dynamic range) of an LNA or mixer for a given channel bandwidth is given by:

$$\mathrm{IMFDR3} = \frac{2}{3}\left[\mathrm{IP3} + 174\,\mathrm{dB} - 10\log\left(\mathrm{BW}\right) - \mathrm{NF}\right] \tag{56.4}$$

The required IMFDR3 for an LNA is thus (for a 200-kHz bandwidth) 80 dB. CMOS down-converters can be made very linear by using MOS transistors in the linear region,[2,6,17] much more linear than the bipolar cross-coupled multipliers. IP3 values of +45 dBm and noise figures of 18 dB have been demonstrated for CMOS realizations.[2,6] This results in an IMFDR3 for a 200-kHz bandwidth of more than 95 dB. The consequence is that the IMFDR3 spec of 80 dB (i.e., without blocking filter) is achievable for the mixer. In this manner, CMOS opens the way to the development of a true fully integrated single-chip receiver for wireless systems that does not require a single external component, not even a blocking filter. In order to achieve this goal, highly linear mixers that multiply with a single sine must be used. However, the noise performance of mixers is intrinsically worse than the noise of an amplifier, and the use of an LNA is still necessary. In order to be able to cope with the blocking levels, the LNA will have to be highly linear and its gain will have to be reduced from a typical value of, for example, 18 dB to 12 dB. The mixers' noise figure will then have to be lowered by about 6 dB too. This will require a higher power consumption from the down-conversion mixer, but the benefit would be that the receiver can then be fully integrated.

The Down-converter

The most-often used topology for a multiplier is the multiplier with cross-coupled variable transconductance differential stages. The use of this topology or related topologies (e.g., based on the square law) in CMOS is limited for high-frequency applications. Two techniques are used in CMOS: the use of the MOS transistor as a switch and the use of the MOS transistor in the linear region.

The technique often used in CMOS down-conversion for its ease of implementation is subsampling on a switched-capacitor amplifier.[5,18,19] Here, the MOS transistor is used as a switch with a high input bandwidth. The wanted signal is commutated via these switches. Subsampling is used in order to be able to implement these structures with a low-frequency op-amp. The switches and the switched capacitor circuit run at a much lower frequency (comparable to an IF frequency or even lower). The clock jitter must, however, be low so that the high-frequency signals can be sampled with a high enough accuracy. The disadvantage of subsampling is that all signals and noise on multiples of the sampling frequency are folded upon the wanted signal. The use of a high-quality HF filter in combination with the switched-capacitor subsampling topology is therefore absolutely necessary.

Figure 56.2 shows the block diagram of a fully integrated quadrature down-converter realized in a 0.7-μm CMOS process.[2] The proposed down-converter does not require any external components, nor does it require any tuning or trimming. It uses a newly developed double-quadrature structure, which renders a very high performance in quadrature accuracy (less than 0.3 degrees in a very large passband). The down-converter topology is based on the use of nMOS transistors in the linear region.[2,6] By using capacitors on the virtual ground, a low-frequency op-amp can be used for down-conversion. The MOS transistors in the linear region result in very high linearity (input-referred IP3 is +27 dBm) for both the RF and the LO input. The advantages of such high linearity on both inputs are, as explained in the previous section, that the mixer can handle a very high IMFDR3, resulting in no need for any kind of HF filtering. This opens the way to the implementation of a fully integrated receiver.

The LNA

As denoted in the previous section, the HF down-conversion mixer tends to have a high noise floor; if the mixer is positioned directly behind the antenna, small antenna signals will be drowned in noise and the overall receiver sensitivity will be low. In order to increase the receiver sensitivity and the SNR at minimum antenna input power, the antenna signal has to be pushed above the noise floor of the mixer by means of a low noise amplifier (LNA). As long as the output noise of the LNA is greater than the input noise of the mixer, the sensitivity is fully determined by the LNA noise figure (NF). This is illustrated in Fig. 56.3.

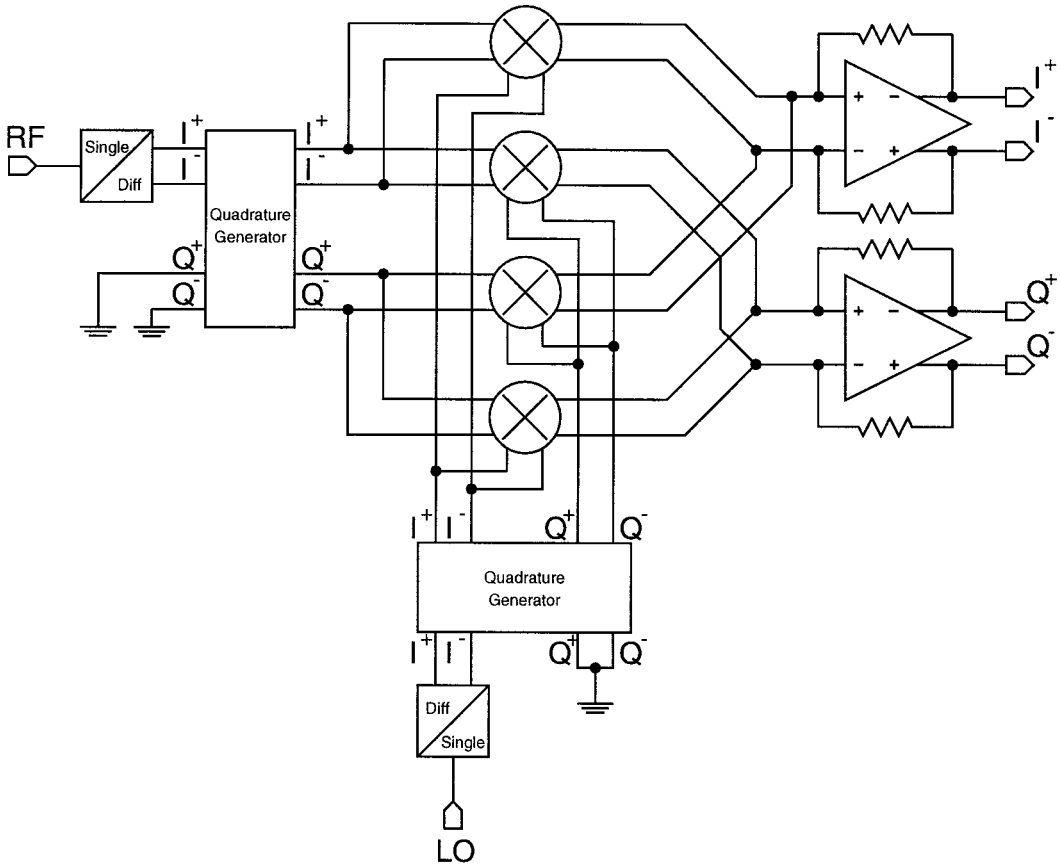


FIGURE 56.2 A double-quadrature down-conversion mixer.

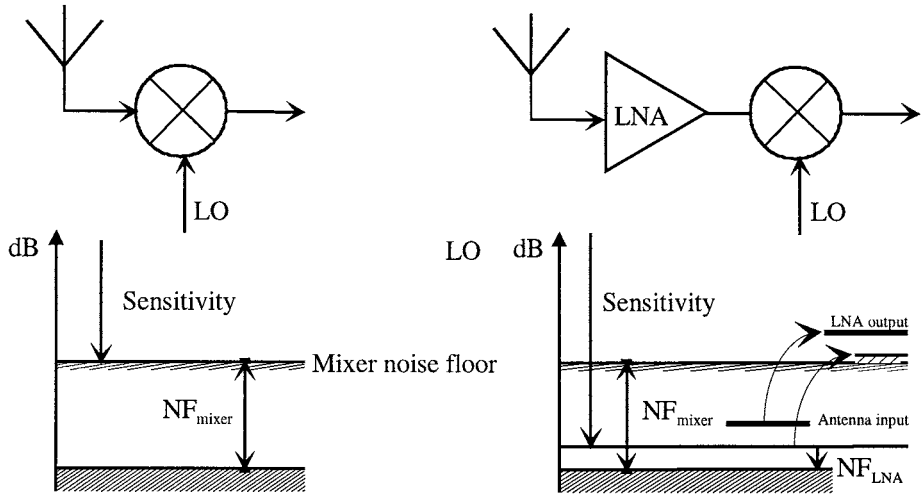


FIGURE 56.3 The benefit of using a low-noise amplifier.

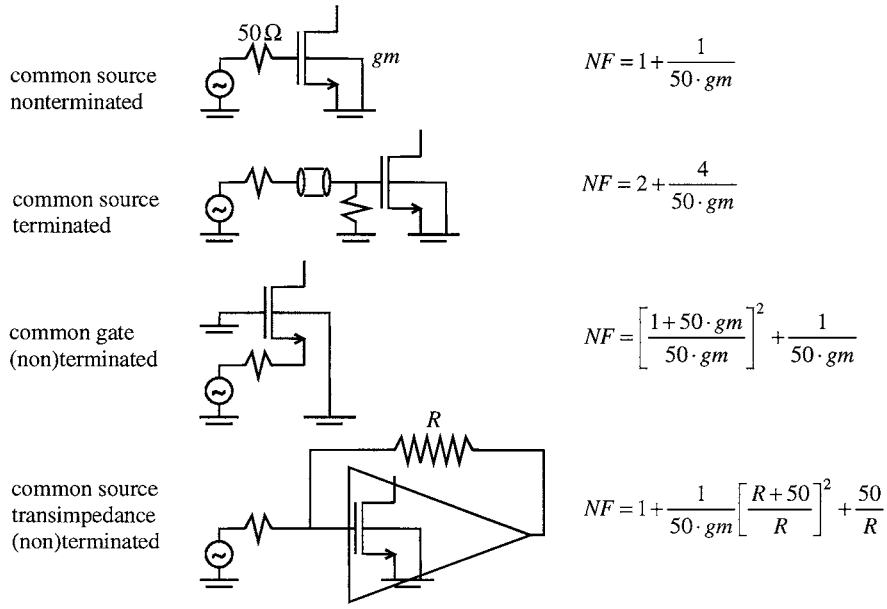


FIGURE 56.4 Noise figure of some common input structures.

The noise figure (NF) of a low-noise amplifier embedded in a 50-Ω system is defined as:

$$\mathrm{NF} = 10\log_{10}\left(\frac{\text{LNA output noise}}{\text{LNA output noise if the LNA itself was noiseless}}\right) \tag{56.5}$$

that is, the real output noise power (dv^2/Hz) of the LNA (consisting of the amplified noise of the 50-Ω source and including all the noise contributions of the amplifier itself to the output noise), divided by the amplified noise of the 50-Ω source only (dv^2/Hz). In this way, the noise figure can be seen as the deterioration of the SNR due to insertion of the non-ideal amplifier. The noise figure is generally dominated by the noise of the first device in the amplifier.

Figures 56.4 and 56.5 compare some common input structures regarding noise. As can be seen from the NF equations and the plotted noise figure as function of the g_m of the transistor for the different topologies, the non-terminated common-source input stage and the (terminated) transimpedance stage are superior as far as noise is concerned. For those circuits, the NF can be approximated as:

$$\left(NF - 1\right) = \frac{1}{50 \cdot g_m} = \frac{\left(V_{gs} - V_t\right)}{50 \cdot 2 \cdot I} \tag{56.6}$$

indicating that a low noise figure needs a high transconductance in the first stage. In order to generate this transconductance with high power efficiency, a low $V_{gs} - V_t$ is preferred. However, this will result in a large gate capacitance. Together with the 50-Ω source resistance in a 50-Ω antenna system, the achievable bandwidth is limited by:

$$f_{3dB} \cong \frac{1}{2\pi \cdot 50\Omega \cdot C_{gs}} \tag{56.7}$$

Together with Eq. 56.6, this results in (f_T is the cutoff frequency of the input transistor)

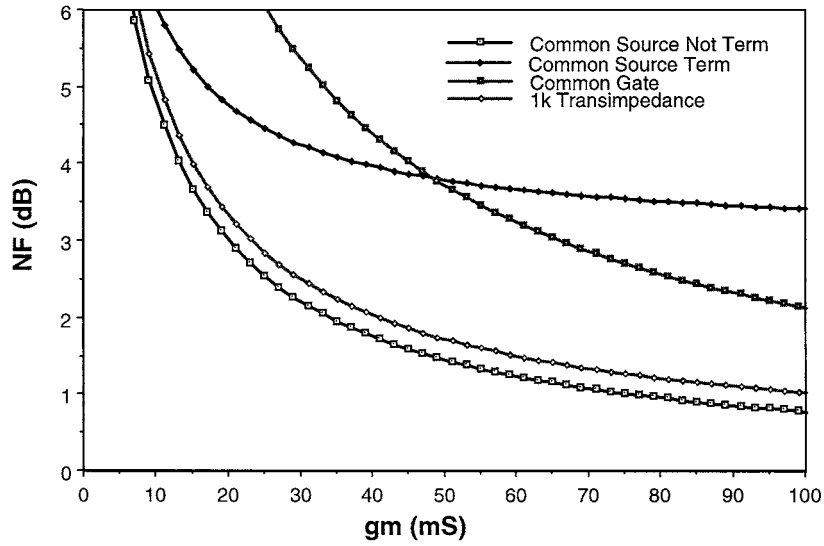


FIGURE 56.5 Performance comparison.

$$\left(\mathrm{NF}-1\right)=\frac{f_{3dB}}{f_T} \tag{56.8}$$

Due to overlap capacitances and Miller effect, this relationship becomes approximately (f_d is the 3-dB point of a transistor in diode configuration):[11]

$$\left(\mathrm{NF}-1\right)=\frac{f_{3dB}}{f_d} \tag{56.9}$$

This means that a low noise figure can only be achieved by making a large ratio between the frequency performance of the technology (f_d) and the working frequency (f_{3dB}). Because for a given technology, f_d is proportional to $V_{gs} - V_t$, this requires a large $V_{gs} - V_t$ and, associated with it, a large power drain. Only by going to real deep sub-micron technologies, will the f_d be high enough to achieve GHz working frequencies with low $V_{gs} - V_t$ values. Only then can the power drain be reduced to an acceptable value.

In practice, the noise figure and the power transfer from the antenna to the LNA is further optimized by doing, respectively, noise and source impedance matching. These matching techniques often rely on inductors to cancel out parasitics by a resonance phenomenon to boost up the maximum working frequency; the LNA works in "overdrive" mode. Although these aspects are not discussed in this chapter, they are very important when designing LNAs for practical boundary conditions like antenna termination, etc.

In contrast to what one might think, there are still some drawbacks in using short-channel devices for low noise. The large electric field at the drain of a sub-micron transistor may produce hot carriers, having a noise temperature significantly above the lattice temperature.[20] This indicates that a good LDD (lightly doped drain) is as crucial for low noise as it is for device reliability.

At high antenna input powers, the signal quality mainly degrades due to in-band distortion components that are generated by third-order intermodulation in the active elements. The linearity performance of LNAs is generally described by the input-referred third-order intercept point (IIP3), as can be seen in Fig. 56.6. IIP3 specifies the extrapolated input signal where third-order intermodulation products start to dominate the output.

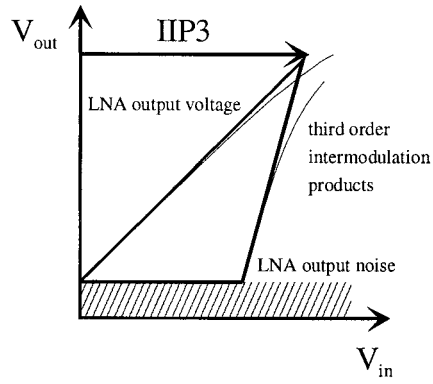


FIGURE 56.6 Linearity performance of an LNA.

As out-of band signals are, in general, orders of magnitude larger than the wanted signal ("blocking levels"), the mixing of out-of-band signals toward an in-band intermodulation product must be avoided by all means. Therefore, it is very important to know the limiting factors and the dynamics of the most important linearity spec, that is, IIP3. Because the core of an amplifier always contains one or more active elements, we will focus on their internal distortion mechanisms.

Long-channel transistors are generally described by a quadratic model. Consequently, a one-transistor common-source amplifier ideally suffers only from second-order distortion and produces no third-order intermodulation products. As a result, high IP3 values should easily be achieved. In fact,

$$\mathrm{IM2} = \frac{1}{2}\frac{v}{V_{gs} - V_t} \quad \text{and} \quad \mathrm{IM3} = 0 \tag{56.10}$$

where v denotes the input amplitude of the amplifier.

However, when the channel length decreases toward deep sub-micron, this story no longer holds; third-order intermodulation starts to become important. To understand the main mechanism behind third-order distortion in a sub-micron CMOS transistor, we start from the equation for the drain current of a short-channel transistor,

$$I_{ds} = \frac{\mu_0 C_{ox}}{2n} \cdot \frac{W}{L} \cdot \frac{\left(V_{gs} - V_t\right)^2}{1 + \Theta \cdot \left(V_{gs} - V_t\right)} \tag{56.11}$$

with

$$\Theta = \theta + \frac{\mu_0}{L_{eff} \cdot v_{\max} \cdot n} \tag{56.12}$$

where θ stands for the mobility degradation due to the transversal electrical field (surface scattering at the oxide–silicon interface) and the $\frac{\mu}{L_{eff} \cdot v_{\max} \cdot n}$-term models the degradation caused by the longitudinal electric field (electrons reaching the thermal saturation speed). As the $_{\theta}$-term is small in today's technologies (increasingly better quality of the oxide–silicon interface), it can often be neglected relative to the longitudinal term. For a typical 0.5 μm CMOS technology, the Θ-parameter equals about 0.9.

It can be seen from Eq. 56.11 that for large values of $V_{gs} - V_t$, the current becomes a linear function of $V_{gs} - V_t$. The transistor is then operating in the velocity saturation region. For smaller vales of $V_{gs} - V_t$, the effect of Θ consists apparently of "linearizing" the quadratic characteristic… but in reality, the effect results in an intermodulation behavior that is worse than in the case of quadratic transistors.

Indeed, we will have a slightly lower amount of second-order intermodulation, but it comes at the cost of third-order intermodulation.

The following equations for the intermodulation ratios *IMx* can be found[13] by calculating the Taylor expansion of the drain current around a certain $V_{gs} - V_t$ value:

$$\mathrm{IM2} = \frac{v}{V_{gs} - V_t} \cdot \frac{1}{(1+r)\cdot(2+r)} \tag{56.13}$$

and

$$\mathrm{IM3} = \frac{3}{4}\,\frac{v}{(V_{gs} - V_t)} \cdot \frac{v}{V_{sv}} \cdot \frac{1}{(1+r)^2 \cdot (2+r)} \tag{56.14}$$

where

$$V_{sv} = \frac{1}{\Theta} \tag{56.15}$$

represents the transit voltage between strong inversion and velocity saturation and

$$r = \frac{V_{gs} - V_t}{V_{sv}} \equiv \Theta \cdot (V_{gs} - V_t) \tag{56.16}$$

denotes the relative amount of velocity saturation. The transit voltage V_{sv} depends only on technology parameters and is about 2 V for a 0.7-μm CMOS technology. For deep sub-micron processes (e.g., a 0.1-micron technology), this voltage becomes even smaller than 300 mV, which is very close to the $V_{gs} - V_t$ at the boundary of strong inversion.

Based on Eq. 56.14, one can directly derive an expression for IIP3:

$$\mathrm{IIP3} \cong 11.25 + 10 \cdot \mathrm{Log}_{10}\left((V_{gs} - V_t)\cdot V_{sv} \cdot (1+r)^2 \cdot (2+r)\right) \tag{56.17}$$

This value is normalized to 0 VdBm, the voltage that corresponds to a power of 0 dBm in a 50-Ω resistor. For the 0.5-μm technology that was mentioned before and a $V_{gs} - V_t$ value of 0.2 V, IIP3 equals +9.5 VdBm. It is worth noting that for a given L_{eff}, the intrinsic IIP3-value of a transistor is only a function of the gate overdrive.

In Figure 56.7, the formula for IP3 is evaluated for a minimum-length transistor in three different technologies. As can be seen from the figure, for a given L_{eff}, the linearity becomes better with increasing gate overdrive. For small gate overdrives, the increase in IIP3 is proportional to the square root of $V_{gs} - V_t$. At high $V_{gs} - V_t$ values (near velocity saturation), the increase in IIP3 becomes even more pronounced. However, this region of operation exhibits a very low transconductance efficiency (g_m/I_{ds}), particularly for sub-micron transistors, where this parameter is given by

$$\frac{g_m}{I_{ds}} = \frac{2}{V_{gs} - V_t} \cdot \frac{1 + \Theta \cdot (V_{gs} - V_t)}{1 + 2\Theta \cdot (V_{gs} - V_t)} \tag{56.18}$$

The influence of L_{eff} on IIP3 can be seen in Fig. 56.7; For practical values of the gate overdrive, the linearity gets worse with decreasing gate length, because V_{sv} is proportional to L_{eff}. This may pose a

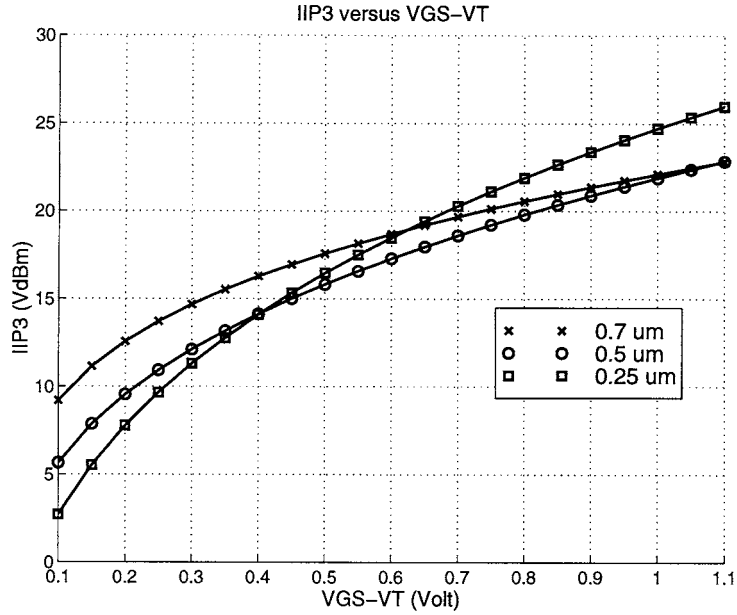


FIGURE 56.7 IIP3 versus $V_{gs} - V_t$ for three different technologies.

problem when very small transistor lengths are required to reduce the power drain and a high IP3 is necessary to avoid the blocking filters. For large values of the gate overdrive, there is a point where the intermodulation performance of a short-channel transistor gets better compared to a large-channel one, because the first already enters the velocity saturation region. As mentioned before, this region of operation is not highly recommended.

Nevertheless, when a certain IIP3 is required, there are basically two methods to ensure this: using a high enough $V_{gs} - V_t$ or using some kind of feedback mechanism (e.g., source degeneration). It can be shown that for the same equivalent g_m and the same distortion performance, the required dc current is lower when local feedback at the source is applied. It comes, however, at the cost of a larger transistor width, eventually compromising the amplifier bandwidth.

56.4 The Synthesizer

Synthesizer Topology

The *frequency synthesizer* generates the local oscillator signal, responsible for the correct frequency selection in the up- and down-converters. Since the frequency spectrum in modern wireless communication systems must be used as efficiently as possible, channels are placed very close together. The signal level of the desired receiving channel can be made very small, whereas adjacent channels can have very large power levels. Therefore, the phase noise specifications for the LO signal are very high, which makes the design of the frequency synthesizer very critical.

Meanwhile, mobile communication means low power consumption, low cost, and low weight. This implies that a completely integrated synthesizer is desirable, where integrated means a standard CMOS technology without any external components or processing steps. Usually, the frequency synthesizer is realized as a phase-locked loop (PLL) as shown in Fig. 56.8. The most critical building blocks of a PLL for integration in CMOS are the building blocks that operate at high frequency: the voltage-controlled oscillator (VCO) and the prescaler. Both will be discussed in the following sections.

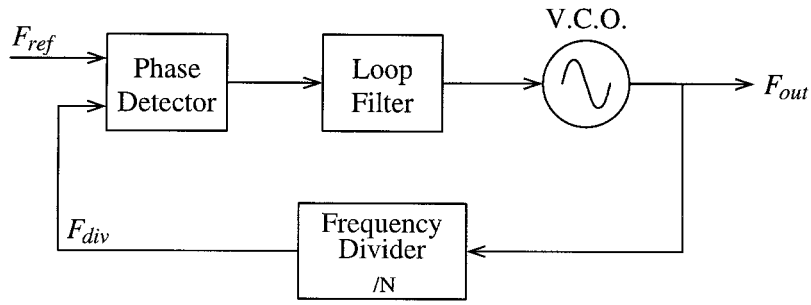


FIGURE 56.8 PLL-based frequency synthesizer.

As stated above, the most important specification of a frequency synthesizer is low phase noise. The following formulae indicate how the noise of the different building blocks is transferred toward the synthesizer's output.

$$\frac{\theta_{out}(s)}{\theta_{vco}(s)} = \frac{N \cdot s}{N \cdot s + K_{pd} \cdot G_{lf}(s) \cdot K_{vco}} \tag{56.19}$$

$$\frac{\theta_{out}(s)}{\theta_{other}(s)} = \frac{N \cdot K_{pd} \cdot G_{lf}(s) \cdot K_{vco}}{N \cdot s + K_{pd} \cdot G_{lf}(s) \cdot K_{vco}} \tag{56.20}$$

with θ_{out} the synthesizer's output noise, θ_{vco} the phase noise of the VCO, θ_{other} the phase noise of the reference signal, the prescaler, the phase detector, and the loop filter, N the prescaler division factor, K_{pd} the phase detector gain, $G_{lf}(s)$ the loop filter transfer function, and K_{vco} the VCO gain.

With $G_{lf}(s)$ as the transfer function of a low-pass filter, the phase noise of the VCO is high-passed toward the output. The phase noise of the other components is low-passed. In other words, the VCO is the main contributor for out-of-band phase noise, while the other building blocks account for the in-band noise.

As can be seen, the choice of the loop bandwidth is critical for phase noise performance. In order to have enough suppression of phase noise at frequency offsets, important for communication standards (e.g., 600kHz for GSM and DCS-1800) and of spurious due to the reference signal, the loop bandwidth cannot be chosen very large (a few kHz typically). Also, for stability reasons, the loop bandwidth has to be small compared to the PLL reference frequency. To realize relatively small loop bandwidths, large capacitor and resistor values are necessary to implement the large time constant. In current commercial designs, the capacitors are often implemented off-chip, to limit the chip area. To go to full integration of the frequency synthesizer, a way must be found to implement the loop filter without the need for large amounts of capacitance. Several possibilities exist.

The first possibility is to increase the resistance needed to create the large time constant of a narrow low-pass filter, which means a decrease of capacitance. The disadvantage of this approach is the increase of loop filter phase noise. Hence, a tradeoff between integrated capacitance and phase noise exists.

A more appealing approach is the one used in Ref. 31. Here, a type-II fourth-order PLL is integrated, using a dual-path filter topology. The loop filter consists of two filter paths: one active filter path and one passive filter path. By combining the signals of both paths, a zero is realized without the need for an extra capacitor and resistor. The zero is necessary to provide enough phase margin for loop stability. The principle is explained in Figure 56.9. In this way, the integrated capacitance is small enough to be integrated on-chip, without degrading the phase noise performance. The total chip area was only 1.9 μm by 1.6 μm.

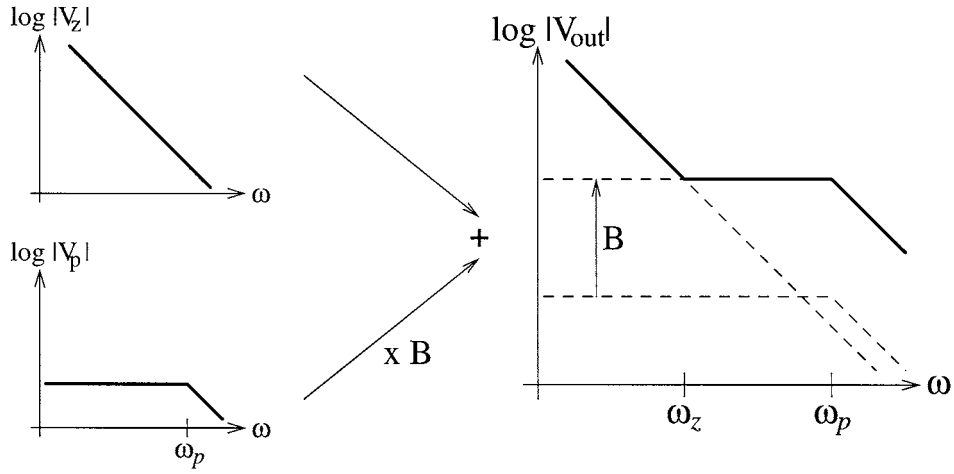


FIGURE 56.9 Dual-path filter principle.

A third possibility is increasing the reference frequency. As a consequence, the loop bandwidth can be made larger, with still enough suppression of the reference spurious noise. This means less integrated capacitance and a better PLL settling time. In addition, the prescaler's division factor *N* will be decreased. As can be seen in Eq. 56.20, the phase noise of the phase detector, the loop filter, the reference signal, and the prescaler is multiplied by *N*. In other words, increasing the reference frequency also results in better in-band phase noise. One disadvantage exists. The prescaler can only divide by integer values, so the smallest frequency step that can be synthesized is equal to the reference frequency. In GSM and DCS-1800 systems, this would mean that the reference frequency must be 200 kHz in order to select all possible channels. This problem can be circumvented by the use of the fractional-N technique. This technique allows the use of fractional division factors by very fast switching between different integer division factors. By combining fractional-N with delta-sigma modulation,[32] the spurs generated by the switching action are shaped to high-frequency noise. This noise can then be filtered by the PLL loop filter.

The Oscillator

As stated above, the *oscillator* will be the main source of out-of-band phase noise. Therefore, its design is one of the most critical parts in the integration of a frequency synthesizer for high-quality communication standards. For the realization of a gigahertz VCO in a sub-micron CMOS technology, two options exist: ring oscillators or oscillators based on the resonance frequency of an LC-tank. The inductor in this LC-tank can be implemented as an active inductor or a passive one. It has been shown that for ring oscillators[21] as well as active LC-oscillators,[22] the phase noise is inversely related to the power consumption.

$$\text{Ring osc. (Ref. 21):}\quad L\{\Delta\omega\} \sim kTR\left(\frac{\omega}{\Delta\omega}\right)^2 \quad \text{with} \quad g_m = \frac{1}{R}$$
$$\text{Active-LC (Ref. 22):}\quad L\{\Delta\omega\} \sim \frac{kT}{2\omega C}\cdot\left(\frac{\omega}{\Delta\omega}\right)^2 \quad \text{with} \quad g_m = 2\omega C \tag{56.21}$$

Therefore, the only viable solution to a low-power, low-phase-noise VCO is an LC-oscillator with a passive inductor. In this case, the phase noise changes proportionally with the power consumption:

$$\text{Passive-LC (Ref. 22):}\quad L\{\Delta\omega\} \sim kTR\left(\frac{\omega}{\Delta\omega}\right)^2 \quad \text{with} \quad g_m = R(\omega C)^2 \tag{56.22}$$

As could be expected, the only limitation in this oscillator is the integrated passive inductor. Equation 56.22 shows that for low phase noise, the resistance R (i.e., the equivalent series resistance in the LC-loop) must be as small as possible. A low resistance also means low losses in the circuit and thus low power needed to compensate for these losses. Capacitors are readily available in most technologies. But since the resistance R will be dominated by the contribution of the inductors' series resistance, the inductor design is critical. Three solutions exist.

Spiral inductors on a silicon substrate usually suffer from high losses in this substrate, which limit the obtainable Q-factor. Recently, techniques have been developed to etch this substrate away underneath the spiral coil in a post-processing step.[7,23] The cavity created by such an etching step can clearly be seen in Fig. 56.10. However, since there is an extra etching step required after normal processing of the ICs, this technique is not allowed for mass production.

For extremely low phase noise requirements, the concept of bondwire inductors has been investigated. Since a bondwire has a parasitic inductance of approximately 1 nH/mm and a very low series resistance, very-high-Q inductors can be created. Bondwires are always available in IC technology, and can therefore be regarded as being standard CMOS components. Two inductors, formed by four bondwires, can be combined in an enhanced LC-tank[22] to allow a noise/power tradeoff. A microphotograph of the VCO is shown in Fig. 56.11.[25] The measured phase noise is as low as –115 dBc/Hz at an offset frequency of 200 kHz from the 1.8-GHz carrier. The power consumption is only 8 mA at 3 V supply. Although chip-to-chip bonds are used in mass commercial products,[28] they are not characterized on yield performance for mass production. Therefore, the industry is reluctant with regard to this solution.

The most elegant solution is the use of a spiral coil on a standard silicon substrate, without any modifications. Bipolar implementations do not suffer from substrate losses because they usually have a

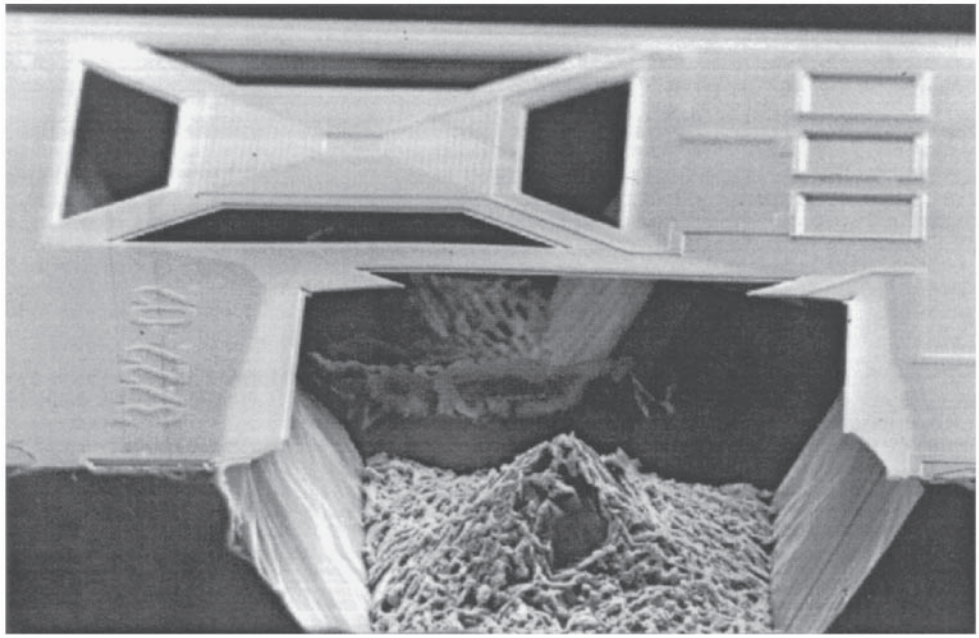


FIGURE 56.10 Etched spiral inductor.

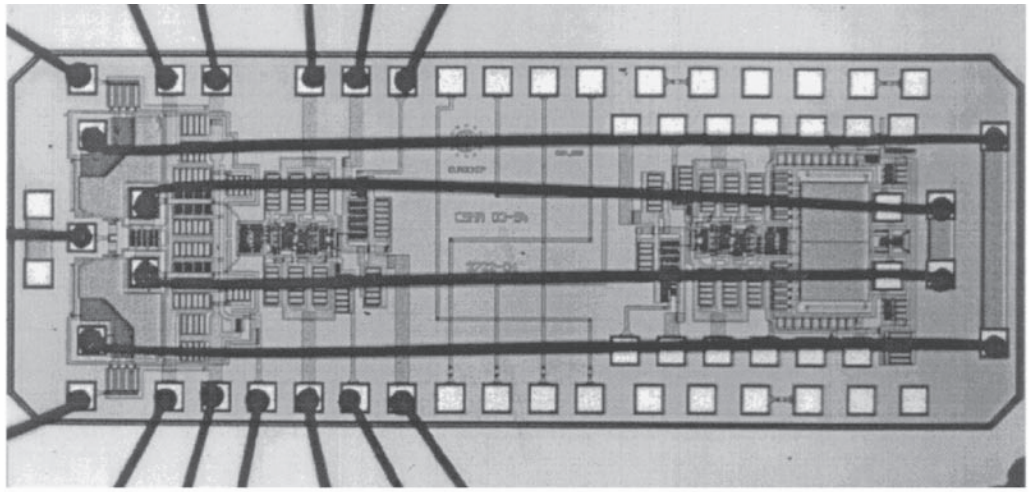


FIGURE 56.11 Microphotograph of the bondwire LC-oscillator.

high-ohmic substrate.[24] Most sub-micron CMOS technologies use a highly doped substrate, and therefore have large induced currents in the substrate, which is responsible for the high losses. The effects present in these low-ohmic substrates can be investigated with finite-element simulations. The finite-element simulations also take the losses in the metal conductors into account. Three phenomena contribute to these losses. The first is the dc series resistance of the metal. The others are high frequency effects, the skin effect, and eddy currents. Due to the skin effect, the metal turns are only partially used for conduction of the high frequency current. Eddy currents are generated by the changing magnetic field that crosses the metal lines, resulting in an increased resistance, especially in the inner turns of the inductors. This analysis can lead to a coil design optimized for a certain technology. A coil was implemented, using the above analysis, in a spiral-inductor LC-oscillator. The technology is a standard two-metal layer, 0.4-μm CMOS technology. With a power consumption of only 11 mW, a phase noise of –122.5 dBc/Hz at 600 kHz offset of the 1.8 GHz carrier has been obtained.[29] A microphotograph is shown in Fig. 56.12.

The Prescaler

To design a high-speed dual-modulus prescaler, a new architecture has been developed that is based on the 90-degrees phase relationship between the master and the slave outputs of an M/S toggle-flip-flop.[26] This architecture is shown in Fig. 56.13. No additional logic is present in the high frequency path to realize the dual-modulus division, as is the case in classic prescalers, based on synchronous counters. Here, the dual-modulus prescaler is as fast as an asynchronous fixed divider. Using this new principle, a 1.75-GHz input frequency has been obtained at a power consumption 24 mW and 3 V power supply. At 5 V power supply, input frequencies above 2.5 GHz can even be processed in a standard 0.7-μm CMOS technology. By going to sub-micron technologies, even higher frequencies can be obtained at low power consumption.

Fully Integrated Synthesizer

The fully integrated VCO and dual-modulus prescaler make it possible to integrate a complete LO synthesizer in a standard CMOS technology, without tuning, trimming, or post-processing, that achieves modern telecom specs. Using the dual-path filter topology for minimizing the necessary integrated capacitance, a type-II, fourth-order, fully integrated PLL frequency synthesizer for DCS-1800 applications has been realized. The PLL is implemented in a standard 0.4-μm CMOS, achieving a phase noise of –121 dBc/Hz at 600 kHz offset frequency, while consuming only 51 mW from a 3-V power supply. The

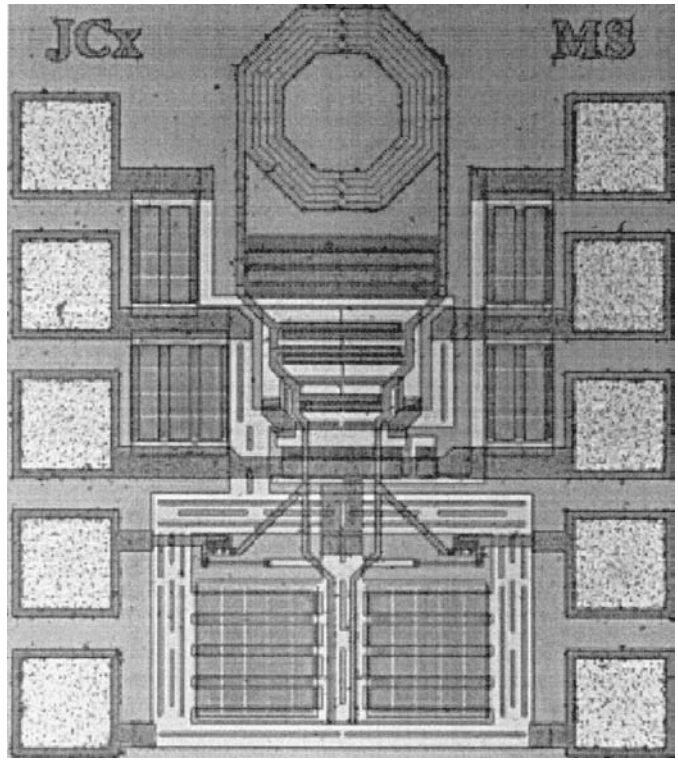


FIGURE 56.12 Microphotograph of the integrated spiral LC-oscillator.

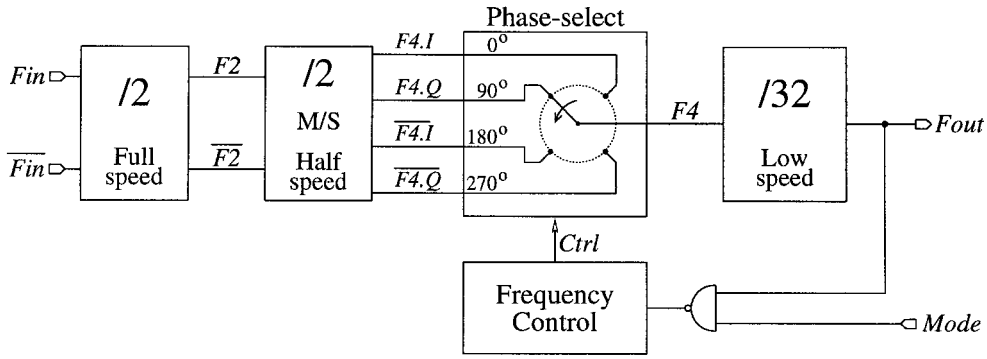


FIGURE 56.13 New dual-modulus prescaler architecture.

integrated capacitance could be decreased to less than 1 nF, for a loop bandwidth of 45 kHz, resulting in a chip area of 1.7 μm by 1.9 μm. A chip microphotograph is shown in Fig. 56.14.

56.5 The Transmitter

For communication systems like GSM, two-way communication is required and a transmitter circuit must be implemented to achieve a full transceiver system. In the open literature, most reported mixer circuits in CMOS are down-conversion mixers. However, as will be explained in the first section below, there is a huge difference between receivers. This implies that a lot of research for the development of CMOS transmitter circuits still needs to be done.

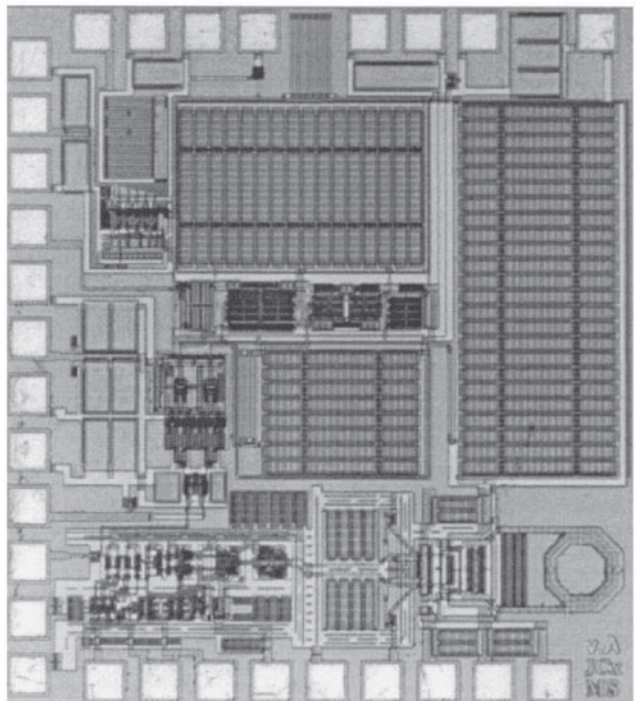


FIGURE 56.14 A fully integrated frequency synthesizer.

Down-conversion vs. Up-conversion

The modulation of the baseband signals on the local oscillator carrier frequency requires an up-conversion mixer topology. In classical bipolar transceiver implementations, the up- and down-converter mixer use typically the same four-quadrant topology. There are, however, some fundamental differences between up- and down-converters, which can be exploited to derive optimal dedicated mixer topologies.

In a down-conversion topology, the two input signals are at a high frequency (e.g., 900 MHz for GSM systems) and the output signal is a low-frequency signal of maximum a few MHz for low-IF or zero-IF receiver systems. This low-frequency output signal can easily be processed making optimal use of the advantages of feedback circuits. Also, high-frequency spurious signals (e.g., LO feedthrough) can be filtered.

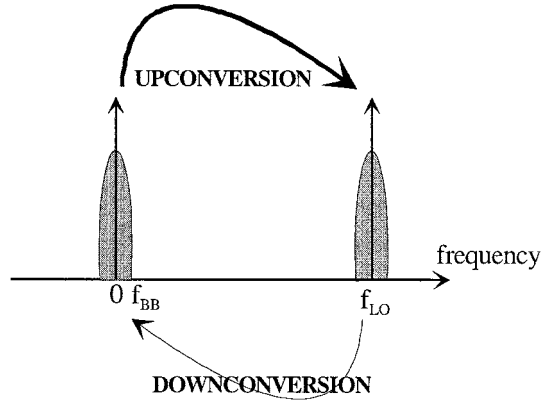


FIGURE 56.15 The difference between up-conversion and down-conversion.

For up-conversion mixers, the situation is totally different. The high frequent local oscillator (LO) and the low frequent baseband (BB) input signal are multiplied to form a high frequent (RF) output signal. All further signal processing has to be performed at high frequencies, which is very difficult and power consuming when using current sub-micron CMOS technologies. Furthermore, all unwanted signals (like the intermodulation products and LO-leakage) have to be limited to a specified level (e.g., below –30 dBc) as they cannot be filtered.

Also, the specifications have to be interpreted differently for both kinds of mixer circuits: for example, an important specification for down-conversion mixers is the conversion gain G_c, defined as the ratio between the output signal and the RF input signal. As for a down-conversion mixer, the RF input signal is fixed — and usually very small; G_c is a good measure of the circuit performance. For an up-conversion mixer, the situation is different. G_c would be the ratio between the RF output signal and the baseband signal. However, in this case, both the input ports, the baseband signal, and the LO signal are free design parameters. As it is easier and more power-friendly to amplify the low-frequency signal, a large baseband signal is preferred. Because of this extra design parameter, it should be better to compare up-conversion circuits based on, for example, the same distortion level or on the same output power level.

CMOS Mixer Topologies

Switching Modulators

Many published CMOS mixer topologies are based on the traditional variable transconductance multiplier with cross-coupled differential modulator stages. Since the operation of the classical bipolar cross-coupled differential modulator stages is based on the translinear behavior of the bipolar transistor, the MOS counterpart can only be effectively used in the modulator or switching mode. Large LO-signals have to be used to drive the gates and this results in a huge LO-feedthrough. In CMOS down-converters, this is already a problem; in Ref. 9, for example, the output signal level is –23 dBm with an LO-feedthrough signal of –30 dBm, which represents a suppression of only –7 dB. This gives rise to very severe problems in direct up-conversion topologies. Moreover, by using a square wave modulating LO signal, 30% of the signal power is present at the third-order harmonic. This unwanted signal can only be filtered with an extra external output blocking filter.

In CMOS, the variable transconductance stage is typically implemented using a differential pair biased in the saturation region. To avoid distortion problems, large $V_{gs} - V_t$ values or a large source degeneration resistance have to be used, which results in large power drain and noise problems, especially compared to the bipolar converter circuits. This can be avoided by replacing the bottom differential pair by a pseudo-differential topology with MOS transistors in the linear region.[17]

Linear MOS Mixers

Next, an intrinsically linear CMOS mixer topology is presented. The modulation is performed by biasing MOS transistors in the linear region. The circuit is focused on a real single-ended output topology, which avoids the use of external combining and the circuits have been optimized based on the analysis of the non-linearities of the mixer structure. The understanding of the origins of the distortion, results in a better compromise between the linearity, the output signal power, and the power consumption. Therefore, the results of the linearity analysis and some guidelines to optimize the circuit performance are also presented in this section. The general design ideas will be illustrated by numerical data from realized chips[33,34]

Figure 56.16 shows the four up-conversion mixers M1a, M1b, M1c, and M1d and the single-ended output current buffer. The realized mixer topology is based on an intrinsic linear mixer circuit that converts the baseband and LO voltages into modulated currents.

Each mixer transistor converts a quadrature LO voltage and baseband voltage to a linearly modulated current. The expression for the drain-source current for a MOS transistor in the linear region is given by:

$$I_{ds} = \beta\left[\left(V_{gs} - V_t\right)V_{ds} - \frac{V_{ds}^2}{2}\right] \tag{56.23}$$

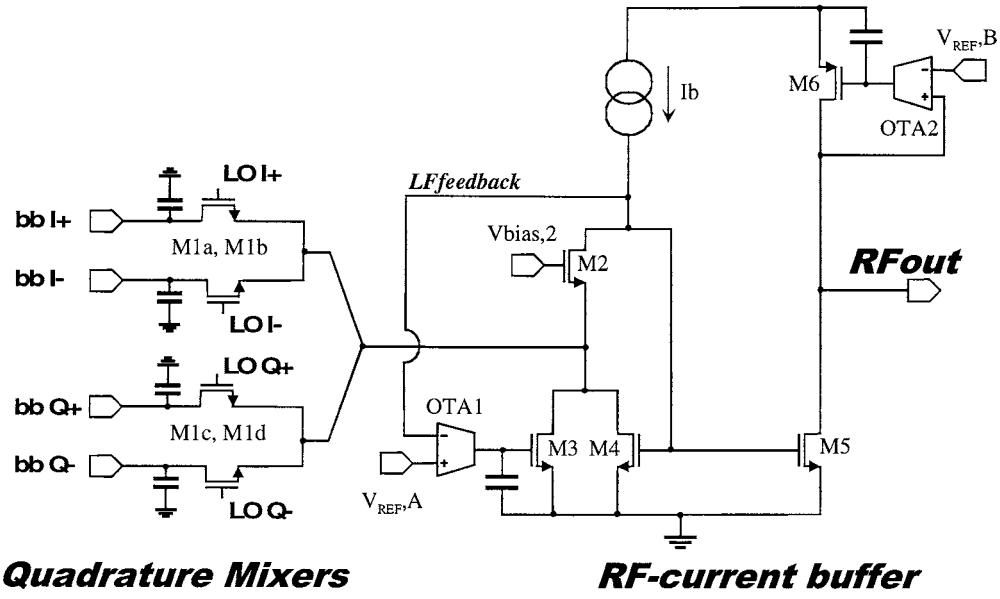


FIGURE 56.16 Schematic of the quadrature up-conversion mixers and output driver.

This equation is rewritten in terms of dc and ac terms as:

$$I_{ds} = \beta\left(V_{ds} + v_{ds}\right)\left(V_{gs} - V_t - \frac{V_d - V_s}{2} + v_g - \frac{v_d + v_s}{2}\right) \tag{56.24}$$

The differential baseband voltage is applied at the drain/source of the mixer transistors M1a and M1b, and the LO signal is applied at the gates of the mixers. Equation 56.24 shows that only two high-frequency components (products with the LO signal v_g) occur in the signal current of each mixer transistor: $\beta V_{DS} v_g$ and $\beta v_{ds} v_g$. The last term is the wanted mixed signal. The first term is proportional to the product of the dc drain-source voltage and the gate (LO) signal. Hence, it is situated at the oscillator frequency. This unwanted signal has been eliminated by applying zero dc drain-source voltage over the mixer transistor. In this way, only the wanted frequency component is formed by each mixer transistor.

The voltage-to-current conversion is performed balanced. The currents are immediately added at the common node, which is made virtual ground by the very low input impedance of the buffer stage (Fig. 56.16).

Quadrature modulation is performed by summing the four modulated currents of each mixer transistor. The resulting single-ended signal current is given by Eq. (56.25).

$$I_{mixer} = \delta\beta\left(v_{bb,I}^2 + v_{bb,Q}^2 + 2 v_{lo} v_{bb}\left(SSB\right)\right) \tag{56.25}$$

where:

δ	= A reduction factor due to the degeneration by the finite input conductance of the output stage
β	= $\mu C_{ox} W/L$
$v_{bb,I}$ and $v_{bb,Q}$	= the baseband I and Q voltage signals, respectively
v_{lo}	= the local oscillator voltage

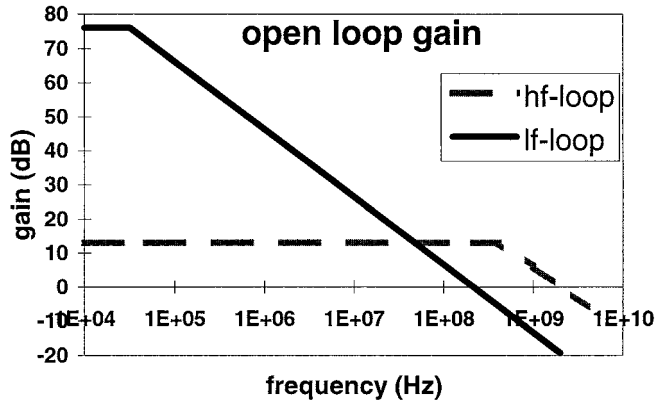


FIGURE 56.17 Open loop conversion gain of the feedback loops.

The modulated current contains a low frequent square baseband signal and the wanted RF single side band (SSB) mixing product.

The mixer has been designed to have all distortion and intermodulation components, including LO-feedthrough, lower than –30 dBc, which is typical for wireless GSM, DECT, and DCS applications.

An essential improvement in the RF mixer design is the implementation of large on-chip capacitors to the ground at the baseband nodes of the mixer transistors. In this way, the modulated RF current flows through the capacitor and not through the bonding wires, which form a considerable impedance at GHz frequency. The RF signal linearity and amplitude becomes independent of the bonding wire matching and length as the RF signal remains on-chip. Also, a degeneration of the high-frequency signal current by the output impedance of the baseband signal source is prevented by this low impedance.

The Low-Frequency Feedback Loop.

The low-frequency feedback loop, which consists of OTA1 and transistors M2 and M3, suppresses the low-frequency (square baseband) signals to enlarge the dynamic range of the output stage and to prohibit intermodulation products of the unwanted low frequent square baseband mixer current with the high-frequency signal. It also lowers the input impedance of the output stage at low frequencies. The operation can be further explained and illustrated by the numerical data given in Ref. 33 and Fig. 56.17. The LF loop has a gain bandwidth of 500 MHz and a high dc gain, which is obtained by using a cascoded OTA structure. The large gain results in a very small (<1 Ω) LF input impedance. In this way, no LF voltage signal is formed at the summing node. This is absolutely essential to obtain a linear operation of the mixer as an additional voltage signal is also up-converted. This structure offers the advantage that the LF and HF modulated current components are separated, and only the high-frequency component of the signal current is mirrored to the output. The dc current through M3 needs to be sufficiently large to reject all the LF current generated in the mixer transistors.

The High-Frequency Current Buffer.

The high-frequency current buffer (M2, M4) ensures a very low input impedance at high frequencies to realize the virtual ground node and mirrors the RF current component (~vlo·vbb) to the output. In Ref. 33, a 2.0-GHz GBW was obtained by designing gm_4 = 14 mS, consuming 5 mA dc current, and the parasitic capacitance at node 2 = 1.1 pF. As the loop gain is unity at 2 GHz, the input impedance of the output driver is determined by $1/g_m2$ = 12 Ω (Fig. 56.18). This low impedance also ensures the stability of the HF feedback loop. The pole at node 1, determined by the parasitic capacitance at the summing node and this input impedance, is at more than 3 times the GBW of HF feedback loop. For the second-order feedback system, this results in a phase margin of more than 68 degrees.

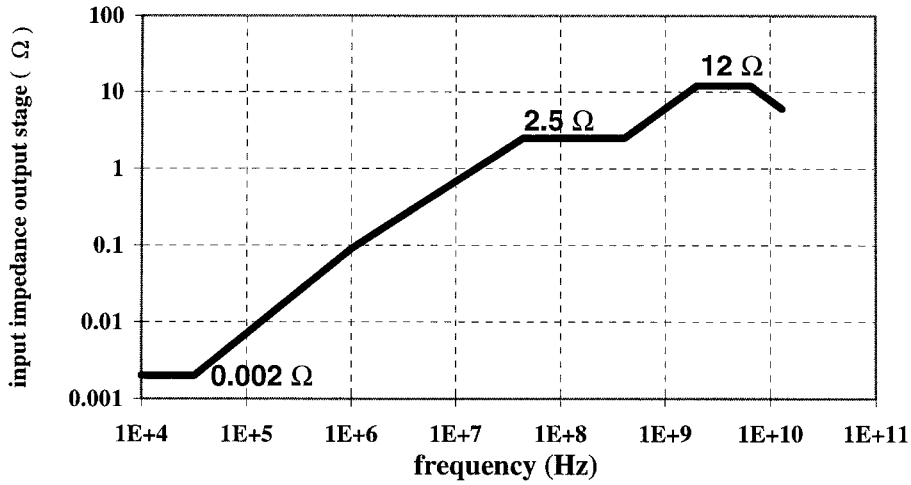


FIGURE 56.18 Input impedance on the virtual ground summing node.

Analysis of the Non-Linearities and the LO-Feedthrough

To obtain an optimal tradeoff between distortion, power consumption, frequency performance, signal power, and distortion, a profound analysis of the causes of the non-linearities is indispensable.

First, the intrinsic non-linearity of the voltage-to-current conversion by the mixer transistors is examined. As the LO feedthrough is also an unwanted signal in the signal band, it is also covered in this section.

A non-ideal virtual ground at the summing node has severe consequences for the modulated signal. A finite output conductance of the baseband signal has similar influences on the linearity of the output signal. These elements are investigated.

Intrinsic Non-Linearity of the Mixer Transistors.

Equations 56.24 and 56.25, which describe the generated current, are only valid if a very low impedance is seen at the drain and source nodes of the mixer transistors. If this condition is fulfilled, no unwanted high-frequency mixing components seem to appear in the modulated current. However, both in measurements and in simulations, a significant unwanted signal is noticed at oscillator frequency (f_{lo}) ± 3 times the baseband frequency (f_{bb}). As this signal originates from a product $v_{lo} \cdot v_{bb}^3$, the magnitude of this distortion component is expected to have the same third-order relationship to the baseband signal. However, only a second-order magnitude relationship is perceived if the virtual ground condition is fulfilled. An explanation for this effect has been searched and the result is described in the paragraphs that follow.

The observed phenomenon is a result of the short-channel effects in a MOS transistor. Both the mobility and the threshold voltage are affected by the gate-source and drain-source voltage. However, the calculated impact of the V_t dependency on the signal is an order of magnitude lower than the observed effect.

The distortion component can be explained by exploring the drain-source voltage dependency of μ_{eff} in the β factor. The expressions for β and μ_{eff} in the linear region are given by:

$$\beta = \mu_{eff} C_{ox} W/L, \text{ where } \mu_{eff} = \frac{\mu_0}{1 + \vartheta\left(V_{gs} - V_t\right) + \frac{\mu_0}{V_{max} L} V_{ds}} \tag{56.26}$$

where: μ_{eff} = The effective mobility
μ_0 = The surface mobility
ϑ = The empirical mobility modulation factor
V_{max} = The maximum drift velocity

If the summing node is made virtual ground and zero dc drain-source voltage is applied, drain and source of each mixer transistor alternate when a sinusoidal signal is applied. If the applied baseband signal is positive, the virtual ground node is the source of that mixer transistor; if the signal is negative, this node is the drain. Mathematically, the source and drain voltages of a mixer transistor can be described as:

$$v_s = \frac{v_{bb} - |v_{bb}|}{2} \qquad v_d = \frac{v_{bb} + |v_{bb}|}{2} \tag{56.27}$$

Equation 56.27 can be transformed to:

$$\mu_{eff} = \frac{\mu_0}{1 + \vartheta\left(V_{gs} - V_t\right)_{dc} + \vartheta\left(v_{lo} - \frac{v_{bb}}{2}\right) + \left(\frac{\mu_0}{V_{max}L} + \frac{\theta}{2}\right)|v_{bb}|} \tag{56.28}$$

Substituting v_{bb} by $A\sin(\omega_{bb}t)$ and making the Fourier series expansion of $|v_{bb}|$ results in Eq. 56.29:

$$\mu_{eff} = \frac{\mu_0}{B\left(1 + \frac{\vartheta}{B}v_{lo} - \frac{\vartheta A}{2B}\sin\left(\omega_{bb}t\right) + C\cos\left(2\omega_{bb}t\right) + D\cos\left(4\omega_{bb}t\right) + \ldots\right)}$$

where: $A\sin\left(\omega_{bb}t\right) =$ The baseband signal

$$\begin{aligned} B &= 1 + \left(V_{gs} - V_t\right)_{DC} + \frac{2}{\pi}A\left(\frac{\mu_0}{V_{max}L} + \frac{\theta}{2}\right) \\ &\approx 1 + \left(V_{gs} - V_t\right)_{DC} \\ C &= \frac{A}{B}\frac{4}{\pi 3}\left(\frac{\mu_0}{V_{max}L} + \frac{\theta}{2}\right) \\ D &= \frac{A}{B}\frac{4}{\pi \cdot 3 \cdot 5}\left(\frac{\mu_0}{V_{max}L} + \frac{\theta}{2}\right) \end{aligned} \tag{56.29}$$

The equation for μ_{eff} shows that a second-order baseband frequency component ($\cos(2\,\omega_{bb}t)$) appears. In the dc reduction factor B, the third term is an order of magnitude smaller than 1. Hence, it appears that the magnitude C of the second-order component has only a first-order relationship to the baseband signal amplitude A. In the mixer voltage-to-current relationship, μ_{eff} is multiplied with $v_{lo} \cdot v_{bb}$. As a result, a mixing component at $f_{lo} \pm 3f_{bb}$ occurs. The magnitude of this signal is only second-order related to the amplitude of the baseband signal. This explains the observed effect.

In the amplitude C of this distortion component, $\frac{\mu_0}{V_{max}L}$ is dominant to $\frac{\theta}{2}$ for most sub-micron CMOS technologies. It is important to notice that the distortion is inversely proportional to the gate length. This implies that this effect will become even more important when going to deeper sub-micron technologies.

Oscillator Feedthrough.

As a direct up-conversion topology is used, the oscillator feedthrough cannot be filtered and, consequently, it has to be as low as the other in-band unwanted frequency components. A signal at oscillator

frequency can have several origins: for example, capacitive feedthrough of the gate voltage or a component due to mixing of a dc drain-source voltage with the oscillator gate signal (see Eq. 56.23). The last is avoided by applying zero dc drain-source voltage over the mixer transistor, as stated earlier.

Due to the differential input voltages, the capacitive LO-feedthrough is canceled at the virtual ground node. However, this cancellation is never perfect, due to technology mismatch. The capacitive LO-feedthrough of one mixer transistor is given by (Eq. 56.30).

$$i_{lo} = \left(\frac{C_{ox}}{2} + \frac{C_{ov}}{L} \right) WLv_{lo} 2\pi f \tag{56.30}$$

where: C_{ox} = The oxide capacitance
C_{ov} = The gate-drain overlap capacitance
v_{lo} = The amplitude of the LO signal
f = The LO frequency

When the asymmetry caused by mismatches in the differential mixer structure is taken into account, the expression for the capacitive L.O.-feedthrough current is given by:

$$\Delta\left(i_{lo}\right) = \delta\left(i_{lo}\right) i_{lo} \tag{56.31}$$

where $\delta(i_{lo})$ is the relative difference in the LO-feedthrough for the differential mixer transistors.

Based on Eqs. 56.25, 56.30, and 56.31, the ratio between the LO feedthrough current and the modulated signal is given in Eq. 56.32.

$$\frac{i_{signal}}{\Delta\left(i_{lo}\right)} = \frac{2\mu C_{ox} \frac{W}{L} v_{bb} v_{lo}}{\delta\left(i_{lo}\right)\left(\frac{C_{ox}}{2} + \frac{C_{ov}}{L} \right) WLv_{lo} 2\pi f} \tag{56.32}$$

This formula can be simplified to:

$$\frac{i_{signal}}{\Delta\left(i_{lo}\right)} = \frac{\mu C_{ox} v_{bb}}{\delta\left(i_{lo}\right)\left(\frac{C_{ox}}{2} + \frac{C_{ov}}{L} \right) L^2 \pi f} \tag{56.33}$$

Even in the case where the two LO-feedthrough currents are added instead of canceled ($\delta(i_{lo}) = 1$), a ratio of 30 dBc signal to LO-feedthrough is achieved with a 1-GHz oscillator signal and a 316 mV baseband signal. If a relative inaccuracy ($\delta(i_{lo})$) of, for example, 10% on the parameters is taken into account, 50 dBc can easily be accomplished. This analysis illustrates the strategy not to rely on cancelling of RF signals to achieve the requirements.

56.6 Toward Fully Integrated Transceivers

Combining all of the above techniques has resulted in the recent development of single-chip CMOS transceiver circuits.[30] Figure 56.19 provides a microphotograph of the 0.25-μm CMOS transceiver test-circuit. The chip does not require a single external component, nor does it require any tuning or trimming. The objective of this design is to develop a complete system for wireless communications at 1.8 GHz that can be built with a minimum of surrounding components: only an antenna, a duplexer, a power amplifier,

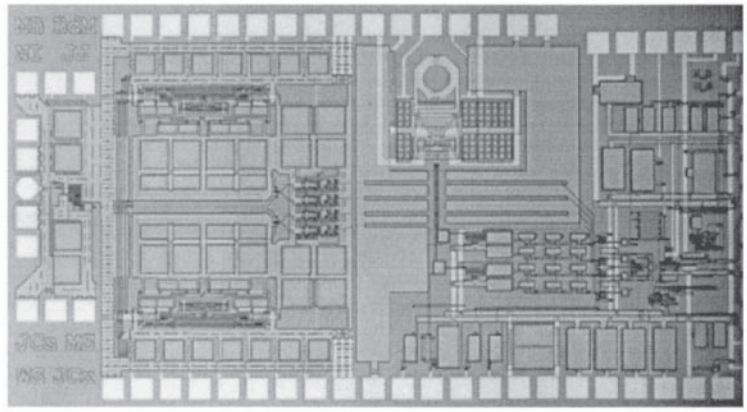


FIGURE 56.19 Microphotograph of a 0.25-μm CMOS transceiver test-circuit.

and a baseband signal processing chip. The high level of integration is achieved using a low-IF topology for reception, a direct quadrature up-conversion topology for transmission, and an oscillator with on-chip integrated inductor. The presented chip has been designed for the DCS-1800 system, but the broadband nature of the LNA, the down-converter, the up-converter, and the output amplifier makes the presented techniques equally suited for use at other frequencies, for example, for use in a DCS-1900 or a DECT system. The presented circuit consumes 240 mW from a 2.5-V supply and occupies a die area of 8.6 mm^2.

56.7 Conclusions

The trends toward deep sub-micron technologies have resulted in the exploration by several research groups of the possible use of CMOS technologies for the design of RF circuits. Especially the development of new receiver topologies, such as quasi-IF and low-IF topologies, in combination with highly linear down-converters, has opened the way to fully integrated down-converters with no external filters or components. However, due to the moderate speed performance of the present sub-micron technologies, lower noise circuits in combination with less power drain have to be worked out. The trends toward deep sub-micron technologies will allow us to achieve those goals as long as the short-channel effects do not limit the performance concerning linearity and intermodulation problems.

Concerning synthesizers, in the last few years, high-performance, low phase noise, low power drain, fully integrated VCO circuits have been demonstrated. Starting with difficult post-processing techniques, research has resulted in the use of standard CMOS technologies using bondwires as inductors. Today, even low phase noise performances with optimized integrated spiral inductors in standard CMOS technologies without any post-processing, tuning, trimming, or external components have been announced. This opens the way toward fully integrated receiver circuits.

However, telecommunication systems are usually two-way systems, requiring transmitter circuits as well. It is only recently that CMOS up-converters with moderate output power have been announced in open literature. Again, thanks to the trends toward deep sub-micron technologies, fully integrated CMOS transmitter circuits with an acceptable power consumption will be feasible. This opens the way for fully integrated transceiver circuits in standard CMOS technologies.

References

1. J. Sevenhans, A. Vanwelsenaers, J. Wenin, and J. Baro, An integrated Si bipolar transceiver for a zero IF 900 MHz GSM digital mobile radio front-end of a hand portable phone, *Proc. CICC*, pp.7.7.1-7.7.4, May 1991.
2. J. Crols and M. Steyaert, A single-chip 900 MHz CMOS receiver front-end with a high performance low-IF topology, *IEEE J. of Solid-State Circuits*, vol. 30, no.12, pp.1483-1492, Dec. 1995.

3. P. R. Gray and R. G. Meyer, Future directions in silicon ICs for RF personal communications, *Proc. CICC*, May 1995.
4. B.-S. Song, CMOS RF circuits for data communications applications, *IEEE J. of Solid-State Circuits*, vol. SC-21, no. 2, pp. 310-317, Apr. 1986.
5. P. Y. Chan, A. Rofougaran, K. A. Ahmed, and A. A. Abidi, A highly linear 1-GHz CMOS down-conversion mixer, *Proc. ESSCIRC*, pp.210-213, Sevilla, Sept. 1993.
6. J. Crols and M. Steyaert, A 1.5 GHz highly linear CMOS down-conversion mixer, *IEEE J. of Solid-State Circuits*, vol. 30, no. 7, pp. 736-742, July 1995.
7. J. Y.-C. Chang, A. A. Abidi, and M. Gaitan, Large suspended inductors on silicon and their use in a 2-μm CMOS RF amplifier, *IEEE Electron Device Letters*, vol. 14, no. 5, pp. 246-248, May 1993.
8. C. H. Hull, R. R. Chu, and J. L. Tham, A Direct-Conversion Receiver for 900 MHz (ISM Band) Spread-Spectrum Digital Cordless Telephone, *Proc. ISSCC*, pp. 344-345, San Francisco, Feb. 1996.
9. A. N. Karanicolas, A 2.7 V 900 MHz CMOS LNA and Mixer, *Proc. ISSCC*, pp. 50-51, San Francisco, Feb. 1996.
10. A. A. Abidi, Radio frequency integrated circuits for portable communications, *Proc. CICC*, San Diego, pp. 151-158, May 1994.
11. M. Steyaert and W. Sansen, Opamp design toward maximum gain-bandwidth, *Proc. of the AACD Workshop*, Delft, pp. 63-85, March 1993.
12. J. Crols, P. Kinget, J. Craninckx, and M. Steyaert, An analytical model of planar inductors on lowly doped silicon substrates for high frequency analog design up to 3 GHz, *Proc. VLSI Circuits Symposium*, June 1996.
13. D. Rabaey and J. Sevenhans, The challenges for analog circuit design in mobile radio VLSI chips, *Proc. of the AACD Workshop*, vol. 2, Leuven, pp. 225-236, Mar. 1993.
14. T. Stetzler, I. Post, J. Havens, and M. Koyama, A 2.7V to 4.5V single-chip GSM transceiver RF integrated circuit, *Proc. ISSCC*, San Francisco, pp. 150-151, Feb. 1995.
15. C. Marshall et al., A 2.7V GSM Transceiver ICs with On-chip Filtering, *Proc. ISSCC*, San Francisco, pp. 148-149, Feb. 1995.
16. J. Sevenhans et al., An analog radio front-end chip set for a 1.9 GHz mobile radio telephone application, *Proc. ISSCC*, San Francisco, pp. 44-45, Feb. 1994.
17. A. Rofougaran et al., A 1GHz CMOS RF front-end IC with wide dynamic range, *Proc. ESSCIRC*, Lille, pp. 250-253, Sept. 1995.
18. D. H. Shen, C.-M. Hwang, B. Lusignan, and B. A. Wooley, A 900 MHz integrated discrete-time filtering RF front-end, *Proc. ISSCC*, San Francisco, pp. 54-55, Feb. 1996.
19. S. Sheng et al., A low-power CMOS chipset for spread spectrum communications, *Proc. ISSCC*, San Francisco, pp. 346-347, Feb. 1996.
20. A. A. Abidi, High-frequency noise measurements on FETs with small dimensions, *IEEE Trans. Electron Devices*, vol. 33, no. 11, pp. 1801-1805, Nov. 1986.
21. B. Razavi, Analysis, modeling, and simulation of phase noise in monolithic voltage-controlled oscillators, *Proc. CICC*, pp. 323-326, May 1995.
22. J. Craninckx and M. Steyaert, Low-noise voltage controlled oscillators using enhanced LC-tanks, *IEEE Trans. on Circuits and Systems. II. Analog and Digital Signal Processing*, vol. 42, no. 12, pp. 794-804, Dec. 1995.
23. A. Rofourgan, J. Rael, M. Rofourgan, and A. Abidi, A 900-MHz CMOS LC-oscillator with quadrature outputs, *Proc. ISSCC*, pp. 392-393, Feb. 1996.
24. N. M. Nguyen and R. G. Meyer, A 1.8-GHz monoithic LC voltage-controlled oscillator, *IEEE Journal of Solid-State Circuits*, vol. 27, no. 3, pp. 444-450, March 1992.
25. J. Craninckx and M. Steyaert, A 1.8-GHz low-phase-noise voltage-controlled oscillator with prescaler, *IEEE Journal of Solid-State Circuits*, vol. 30, no. 12, pp. 1474-1482, Dec. 1995.
26. J. Craninckx and M. Steyaert, A 1.75-GHz/3-V dual modulus divide-by-128/129 prescaler in 0.7-μm CMOS, *Proc. ESSCIRC*, pp. 254-257, Sept. 1995.
27. P. Kinget and M. Steyaert, A 1 GHz CMOS upconversion mixer, *Proc. CICC*, session 10.4, May 1996.

28. –, AD 7886, a 12-bit, 750 kHz, sampling ADC, *Analog Devices Data Sheet*, Apr. 1991.
29. J. Craninckx and M. Steyaert, A fully integrated spiral-LC CMOS VCO set with prescaler for GSM and DCS-1800 systems, *Proc. CICC*, pp. 403-406, May 1997.
30. M. Steyaert et al. A single chip CMOS transceiver for DCS1800 wireless communications, *Proc. IEEE-ISSCC'98*, Feb. 1998.
31. J. Craninckx and M. Steyaert, A fully integrated CMOS DCS-1800 frequency synthesizer, *Proc. ISSCC*, San Francisco, pp. 372-373, Feb. 1998.
32. T. A. D. Riley, M. A. Copeland, and T. A. Kwasniewski, Delta-sigma modulation in fractional-N frequency synthesis, *IEEE J. of Solid-State Circuits*, vol. 28, no. 5, pp. 553-559, May 1993.
33. M. Borremans, M. Steyaert, and T. Yoshitomi, A 1.5V wide band 3GHz CMOS quadrature direct up-converter for multi-mode wireless communications, *Proc. CICC '98*, pp. 79-82.
34. M. Borremans and M. Steyaert, A 2V low distortion 1 GHz CMOS up-converter mixer, *IEEE J. of Solid-State Circuits,* vol. 33, no. 3, pp. 359-366, Mar. 1998.

Yuan, M., Wang, C. "PLL Circuits"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

57

PLL Circuits

Min-shueh Yuan
Chorng-kuang Wang
National Taiwan University

57.1 Introduction

What Is and Why Phase-Locked?

Phase-locked loop (PLL) is a circuit architecture that causes a particular system to track with another one. More precisely, PLL synchronizes a signal (usually a local oscillator output) with a reference or an input signal in frequency as well as in phase.

Phase locking is a useful technique that can provide effective synchronization solutions in many data transmission systems such as optical communications, telecommunications, disk drive systems, and local networks, in which data is transmitted in baseband or passband. In general, only data signals are transmitted in most of these applications; namely, clock signals are not transmitted in order to save hardware cost. Therefore, the receiver should have some schemes to extract the clock information from the received data stream in order to recover transmitted data. The scheme is called a *timing recovery* or *clock recovery* circuit.

The cost of electronic interfaces in communication systems is higher in conjunction with the higher data rate. Hence, high-speed circuits are the practical issue of the high data rate systems implementation, and advanced VLSI technology plays an important role in cost reduction for the high-speed communication systems.

Basic Operation Concepts of Phase-Locked Loops (PLLs)

Typically, a PLL consists of three basic functional blocks: a phase detector (PD), a loop filter (LF), and a voltage-controlled oscillator (VCO). The PD detects the phase difference between the VCO output and the input signal, and generates a signal proportional to the phase error. The PD output contains a dc component and an ac component; the former is accumulated and the latter is filtered by the loop filter. The loop filter output that is near a dc signal is applied to the VCO. This almost dc control voltage

changes the VCO frequency toward a direction to reduce the phase error between the input signal and the VCO. Depending on the type of loop filter used, the steady-state phase error will be reduced to zero or to a finite value.

PLL has an important feature: the ability to suppress both the noises superimposed on the input signal and generated by the VCO. The narrower the bandwidth of the PLL, the more effective jitter filtering the PLL can achieve. However, the error of the VCO frequency cannot be reduced rapidly. Although a narrow bandwidth is better for rejecting large amounts of input noise, it also prolongs the settling time in the acquisition process. So, there is a tradeoff between jitter filtering and fast acquisition.

Classification of PLL Types

Different PLL types are built from different classes of building blocks. The first PLL ICs appeared around 1965 and were purely analog devices. In the so-called "*linear PLLs*" (LPLLs), an analog multiplier (four-quadrant) is used as the phase detector, the loop filter is built from a passive or an active RC filter, and the voltage-controlled oscillator is used to generate the output signal of the PLL. In most cases, the input signal to this linear PLL is a sine wave, whereas the VCO output signal is a symmetrical square wave.

The classical *digital PLL* (DPLL) uses a digital phase detector such as an XOR gate, a JK-flipflop, or a phase-frequency detector (PFD), with the remaining blocks still being the same as LPLL. In many aspects, the DPLL performance is similar to the LPLL.

The function blocks of the *all-digital PLL* (ADPLL) is implemented by purely digital circuits, and the signals within the loop are digital too. Digital versions of the phase detector are the same as DPLL. The digital loop filter is built from an ordinary up/down counter, N-before-M counter, or K-counter.[1] The digital counterpart of the VCO is the digital-controlled oscillator (DCO).[2,3]

Analogous to filter designs, PLLs can be implemented by software such as a microcontroller, microcomputer, or digital signal processing (DSP); this type of PLL is called *software PLL* (SPLL).

57.2 PLL Techniques

Basic Topology

A phase-locked loop is a feedback system that operates and minimizes the phase difference between two signals. Figure 57.1 is the basic function block of a PLL, which consists of a phase detector (PD), a loop filter, and a VCO. The PD works as a phase error amplifier. It compares the phase of the VCO output signal $u_o(t)$ with the phase of the reference signal $u_i(t)$ and develops an output signal $u_d(t)$, that is proportional to the phase error θ_e. Within a limited range

$$u_d\left(t\right) = k_d\theta_e \tag{57.1}$$

where k_d represents the gain of the PD and the unit of k_d is volt/rad.

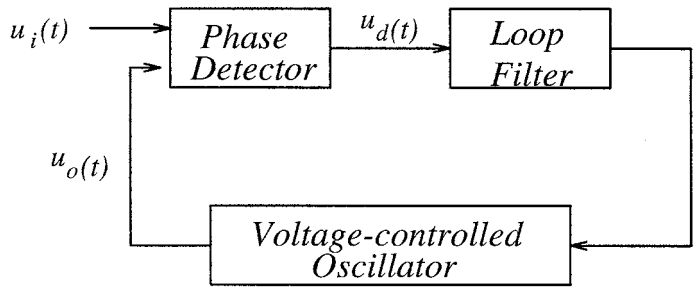


FIGURE 57.1 Basic block diagram of the PLL.

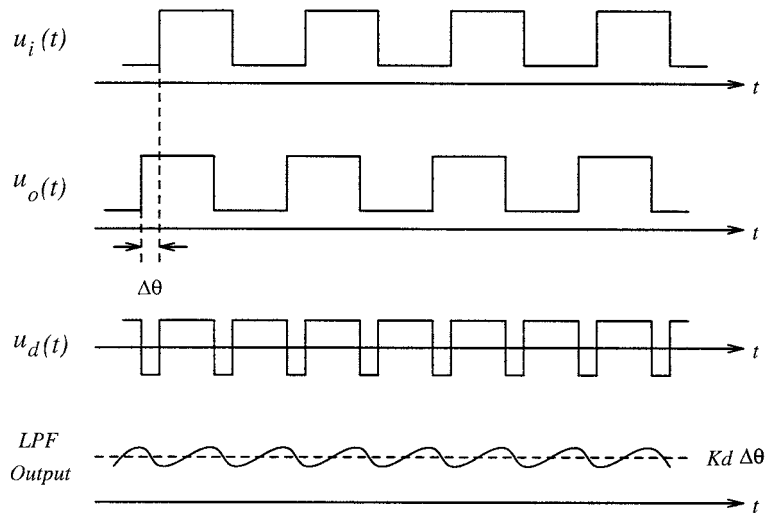


FIGURE 57.2 Waveforms in a PLL.

The output signal $u_d(t)$ of the PD consists of a dc component and a superimposed ac component. The latter is undesired and removed by the loop filter (LPF). Thus, the LPF generates an almost-dc control voltage for the VCO to oscillate at a frequency equal to the input frequency.

How the building blocks of the basic PLL work together will be explained in the following. First, the angular frequency ω_i of the input signal $u_i(t)$ is assumed equivalent to the central frequency ω_o of the VCO signal $u_o(t)$. Now, a small positive frequency step is applied to $u_i(t)$ at $t = t_o$ (shown in Fig. 57.2). $u_i(t)$ accumulates a phase faster than $u_o(t)$ of the VCO does. The PD then generates wider pulses increasingly, which results in a higher dc voltage at the LPF output to increase the VCO frequency. Depending on the type of the loop filter, the final phase error will be reduced to zero or to a finite value.

It is important to note from the descriptions in the above that the loop locks only after two conditions are satisfied: (1) ω_i and ω_o are equal and (2) the phase difference between the input $u_i(t)$ and the VCO output $u_o(t)$ settles to a steady-state value. If the phase error varies with time so fast that the loop is unlocked, the loop must keep on the transient process, which involves both *frequency acquisition* and *phase acquisition.*

To design a practical PLL system, it is required to know the status of the responses of the loop if (1) the input frequency is varied slowly (tracking process), (2) the input frequency is varied abruptly (lock-in process), and (3) the input and the output frequencies are not equal initially (acquisition process). Using linear PLL as an example, these responses will be shown in the following subsections.

Loop Order of the PLL

Second-Order Loop

The loop order of the PLL depends on the characteristics of the loop filter; therefore, the loop filter is a key component that affects the PLL dynamic behavior. Figure 57.3 shows three types of loop filters that are widely used. Figure 57.3(a) is a passive lead-lag filter; the associate transfer function $F(s)$ is given by

$$F(s) = \frac{1 + s\tau_2}{1 + s(\tau_1 + \tau_2)} \tag{57.2}$$

where $\tau_1 = R_1C$ and $\tau_2 = R_2C$. Figure 57.3(b) shows an active lead-lag filter; its transfer function is

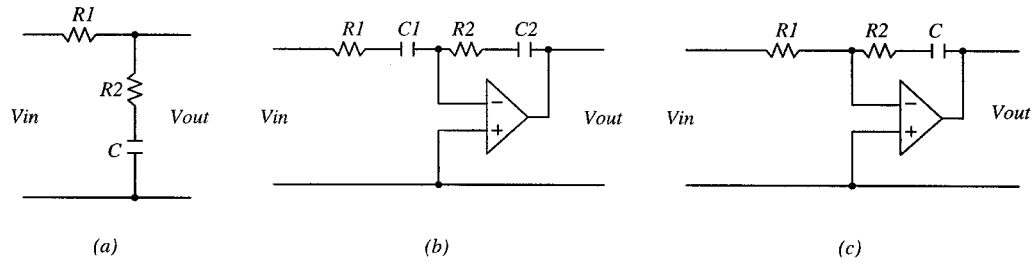


FIGURE 57.3 (a) Passive lead-lag filter, (b) active lead-lag filter, and (c) active PI filter.

$$F(s) = k_a \frac{1+s\tau_2}{1+s\tau_1} \tag{57.3}$$

where $\tau_1 = R_1C_1, \tau_2 = R_2C_2$ and $k_a = -\frac{C_1}{C}$. A PI filter is shown in Fig. 57.3(c), where PI stands for *Proportional and Integral* action. The transfer function is given by

$$F(s) = \frac{1+s\tau_2}{s\tau_1} \tag{57.4}$$

where $\tau_1 = R_1C$ and $\tau_2 = R_2C$. Their Bode diagrams are shown in Fig. 57.4 respectively. High-order filters could be used in some applications, but additional filter poles introduce a phase shift. High-order systems are not trivial generally to maintain a stable system.

Figure 57.5 shows the linear block diagram of the PLL. According to control theory, the closed loop transfer function of PLL can be derived as

$$H(s) \triangleq \frac{\theta_o(s)}{\theta_i(s)} = \frac{k_d k_o F(s)}{s + k_d k_o F(s)} \tag{57.5}$$

where k_d with units V/rad is called the phase-detector gain, k_o is the VCO gain factor and has units rad/s-V. In addition to the phase transfer function, a phase-error transfer function $H_e(s)$ is defined as follows:

$$H_e(s) \triangleq \frac{\theta_e(s)}{\theta_i(s)} = \frac{s}{s + k_d k_o F(s)} \tag{57.6}$$

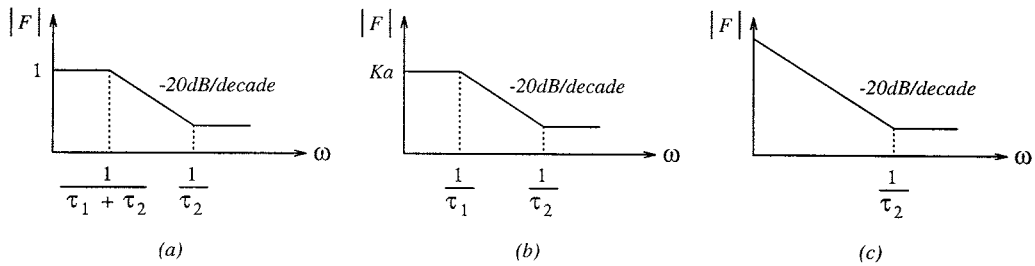


FIGURE 57.4 Bode diagrams of (a) passive lead-lag filter, (b) active lead-lag filter, and (c) active PI filter.

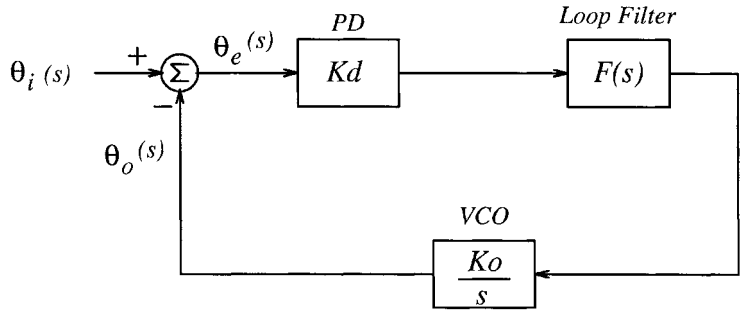


FIGURE 57.5 Linear model of PLL.

The transfer functions of the loop filters shown in Fig. 57.3 are substituted for $F(s)$ in Eq. 57.5 in order to analyze the phase transfer function. We obtain the phase transfer functions as follows:

For the passive lead-lag filter,

$$H(s) = \frac{k_d k_o \dfrac{1+s\tau_2}{\tau_1+\tau_2}}{s^2 + s\dfrac{1+k_d k_o \tau_2}{\tau_1+\tau_2} + \dfrac{k_d k_o}{\tau_1+\tau_2}} = \frac{\omega_n\left(2\zeta - \dfrac{\omega_n}{k_d k_o}\right)s + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{57.7}$$

For the active lead-lag filter,

$$H(s) = \frac{k_d k_a k_o \dfrac{1+s\tau_2}{\tau_1}}{s^2 + s\dfrac{1+k_d k_a k_o \tau_2}{\tau_1} + \dfrac{k_d k_a k_o}{\tau_1}} = \frac{\omega_n\left(2\zeta - \dfrac{\omega_n}{k_d k_o k_a}\right)s + \omega}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{57.8}$$

For the active PI filter,

$$H(s) = \frac{k_d k_o \dfrac{1+s\tau_2}{\tau_1}}{s^2 + s\dfrac{k_d k_o \tau_2}{\tau_1} + \dfrac{k_d k_o}{\tau_1}} = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{57.9}$$

where ω_n is the *natural frequency* and ξ is the *damping factor*. These two parameters are important to characterize a PLL. If the condition $k_d k_o \gg \omega_n$ or $k_d k_o k_o \gg \omega_n$ is true, this PLL system is called a *high-gain loop*. If the reverse is true, the system is a *low-gain loop*. Most practical PLLs are high-gain loops for good tracking performance. For a high-gain loop, Eqs. 57.7 to 57.9 become approximately

$$H(s) \approx \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{57.10}$$

Similarly, assuming a high-gain loop, the approximate expression of the phase-error transfer function $H_e(s)$ for all three loop filter types becomes

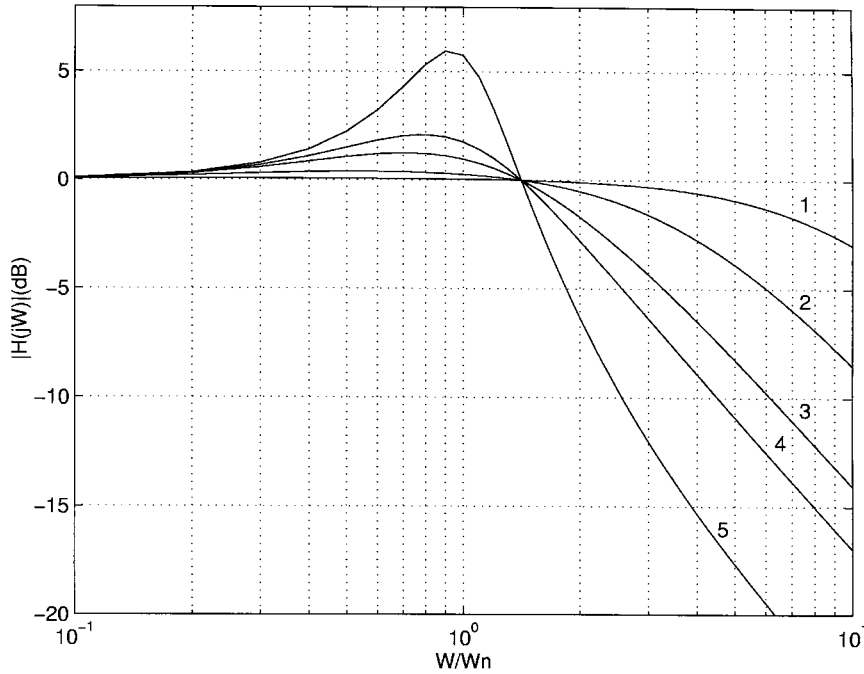


FIGURE 57.6 Frequency responses of the phase transfer function $H(j\omega)$ for different damping factors. Trace 1: ζ = 5, Trace 2: ζ = 2, Trace 3: ζ = 1, Trace 4: 5 = 0.707, Trace 5: 3 = 0.3.

$$H_e(s) \approx \frac{s^2}{s^2 + 2s\zeta\omega_n + \omega_n^2} \tag{57.11}$$

Because the highest power of s in the denominator of the transfer function is 2, the loop is known as the second-order loop.

The magnitude of the frequency responses of $H_e(s)$ for a high-gain loop with several values of damping factor are plotted in Fig. 57.6. It shows that the loop performs a low-pass filtering on the input phase signal. That is, the second-order PLL is able to track both phase and frequency modulations of the input signal as long as the modulation frequency remains within the frequency band roughly between zero and ω_n.

The magnitude of the frequency responses of $H_e(s)$ are plotted in Fig. 57.7. A high-pass characteristic is derived. It indicates that the second-order PLL tracks the low-frequency phase error but cannot track high-frequency phase error.

The transfer function $H(s)$ has a –3dB frequency, ω_{-3db}. ω_{-3db} stands for the closed-loop bandwidth of the PLL. The relationship between ω_{-3db} and ω_n is presented here to provide a comparison with the familiar concept of bandwidth. In the high-gain loop case, by setting $|H(j\omega)| = \frac{1}{\sqrt{2}}$ and solving for ω, we find that

$$\omega_{-3db} = \omega_n \left[2\zeta^2 + 1 + \sqrt{\left(2\zeta^2 + 1\right)^2} \right]^{\frac{1}{2}} \tag{57.12}$$

The relationship between ω_{-3db} and ω_n for different damping factors is plotted in Fig. 57.8.[4]

Other-Order Loop

The second-order PLL with a lead-lag loop filter is commonly used because the lead-lag filter has two time constants to determine the natural frequency and the damping factor independently. Therefore, a

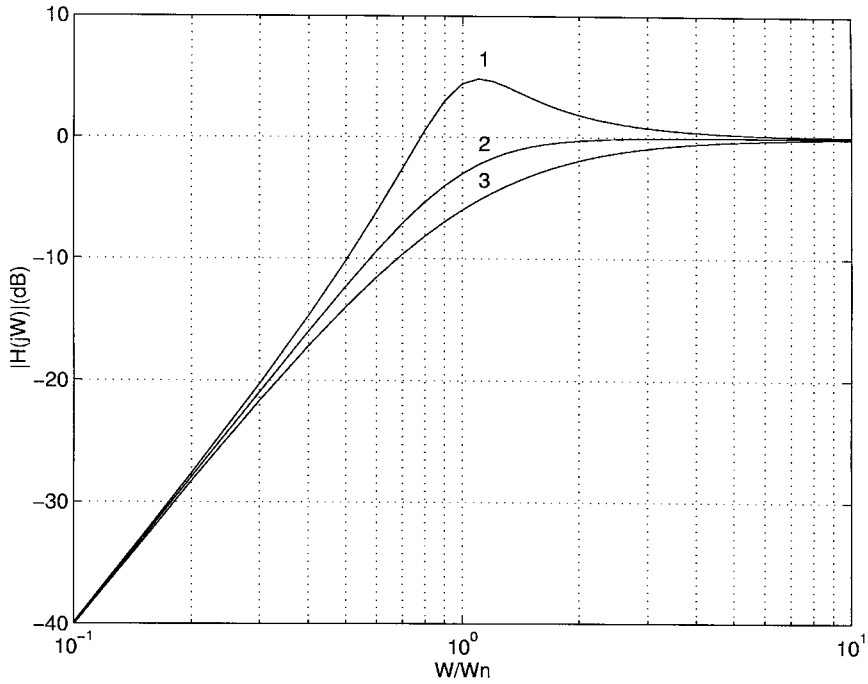


FIGURE 57.7 Frequency responses of the phase-error transfer function $H_e(j\omega)$ for different damping factors. Trace 1: $\zeta = 0.3$, Trace 2: $\zeta = 0.707$, Trace 3: $\zeta = 1$.

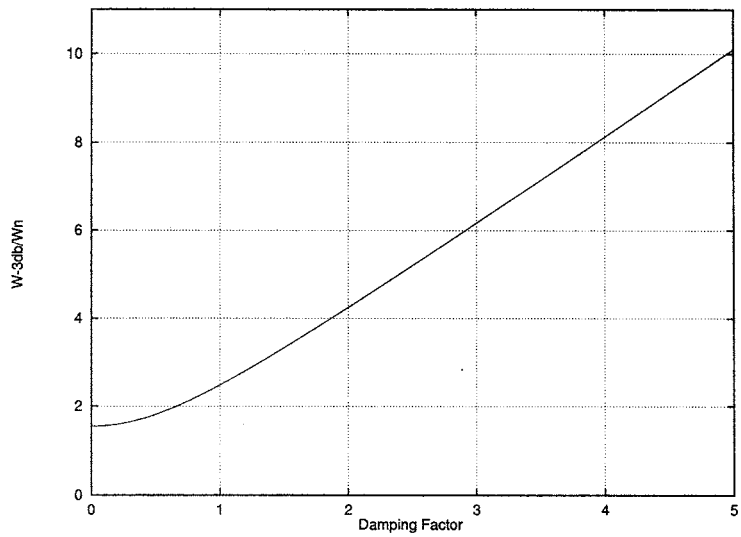


FIGURE 57.8 ω_{-3db} bandwidth of a second-order loop versus different damping factors.

high dc loop gain for good tracking, while maintaining an overdamped PLL system, can be achieved. No loop filter and single pole loop filter cases are discussed briefly in the following for completeness.

If the loop filter is left out, a first-order loop is obtained. As shown in Fig. 57.5, set $F(s) = 1$ and the closed-loop transfer function can be derived as

$$H(s) = \frac{k}{s+k} \tag{57.13}$$

where $k = k_d k_o$. If it is necessary to design a high dc gain loop for fast tracking, then the bandwidth of the PLL must be wide enough because the dc loop gain k is the only parameter available, which is not suitable for noise suppression. Therefore, fast tracking and narrow bandwidth are incompatible in a first-order loop.

Another commonly used loop filter is the passive lag filter. The transfer function is

$$F(s) = \frac{1}{1+s\tau} \tag{57.14}$$

The closed-loop transfer function can be derived as

$$H(s) = \frac{\frac{k_d k_o}{\tau}}{s^2 + \frac{1}{\tau}s + \frac{k_d k_o}{\tau}} = \frac{\tau\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{57.15}$$

where $\omega_n = \sqrt{\frac{k_d k_o}{\tau}}$ and $\zeta = \frac{1}{2\sqrt{\tau k_d k_o}}$. Although there are two parameters (τ and $k = k_o k_d$) available, three loop parameters must be determined (ω_n, ξ,k). That is if it is necessary to have a large dc loop gain and a narrow bandwidth, the loop will be severely underdamped and the transient response will be poor.

A high-order filter is used for critical applications because it provides better noise filtering. However, it is difficult to design a high-order loop due to some problems such as loop stability.

Tracking Process

The linear model of a PLL shown in Fig. 57.5 is suitable for analyzing the tracking performance of a PLL that is initially locked, but with small phase error. If the phase error changes so abruptly that the PLL loses lock, a large phase error will be induced, even if it happens only momentarily. The unlock condition is a non-linear process that cannot be analyzed by a linear model. The acquisition process will be described later.

At first, consider a step phase error applied to the input. The input phase signal is $\theta_i(t) = \Delta\theta u(t)$, the Laplace transform of the input is $\theta_i(s) = \frac{\Delta\theta}{s}$, which is substituted into Eq. 57.11 to get

$$\theta_e(s) = \frac{\Delta\theta}{s} \frac{s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{57.16}$$

According to the final value theorem of the Laplace transform,

$$\lim_{t\to\infty} \theta_e(t) = \lim_{s\to 0} s\theta_e(s) = 0$$

In other words, the loop will eventually track on the step phase change without steady-state phase error.

Applying a step change of frequency $\Delta\omega$ to the input, the input phase change is a ramp (i.e., $\theta_i(t) = \Delta\omega t$); therefore, $\theta_i(s) = \frac{\Delta\omega}{s^2}$. Substituting $\theta_i(s)$ in Eq. 57.6 and applying the final value theorem, one obtains

$$\begin{aligned} \theta_v &= \lim_{t\to\infty} \theta_e(t) = \lim_{s\to 0} s\theta_e(s) \\ &= \lim_{s\to 0} \frac{\Delta\omega}{s + k_d k_o F(s)} \end{aligned} \tag{57.17}$$

$$= \frac{\Delta\omega}{k_d k_o F(0)}$$

$$= \frac{\Delta\omega}{k_v}$$

θ_v is called the *velocity error* or *static phase error.*[4] Because the input frequency almost never agrees exactly with the VCO free-running frequency, there is a frequency difference $\Delta\omega$ between the two. From Eq. 57.17, if the PLL has a high dc loop gain, (i.e., $k_d k_o F(0) \gg \Delta\omega$), the steady-state phase error to a step frequency error input approaches zero. This is the reason why a high-gain loop has a good tracking performance. Now the advantage of a second-order loop using an active loop filter of high dc gain is evident. The active lead-lag loop filter with a high dc gain will make the steady-state phase error approach zero and the noise bandwidth narrow simultaneously, which is impossible in a first-order loop.

If the input frequency is changed linearly with time at a rate of $\Delta\dot{\omega}$, which is $\theta_i(t) = \frac{1}{2}\,\Delta\dot{\omega}\,t^2$, so $\theta_i(s) = \frac{\Delta\dot{\omega}}{s^3}$. According to a high-gain loop and applying the final value theorem of Laplace transform, then

$$\begin{aligned} \theta_a &= \lim_{t\to\infty} \theta_e(t) = \lim_{s\to 0} s\theta_e(s) \\ &= \frac{\Delta\dot{\omega}}{\omega_n^2} \end{aligned} \tag{57.18}$$

θ_a is called an *acceleration error* (sometimes called *dynamic tracking error* or *dynamic lag*).[4]

In some applications, PLL needs to track an accelerating phase error without static tracking error. The expression of the static phase error when frequency ramp is applied

$$\theta_e(s) = \lim_{s\to 0} \frac{\Delta\omega}{s\left(s + k_d k_o F(s)\right)} \tag{57.19}$$

For θ_e to be zero, it is necessary to make $F(s)$ be a form of $\frac{G(s)}{s^2}$, where $G(0) \neq 0$. $\frac{G(s)}{s^2}$ implies that the loop filter has two cascade integrators. This is a third-order loop. In order to eliminate the static acceleration error, a third-order loop is very useful for some special applications, such as satellites and missiles systems.

Based on Eq. 57.18, a large natural frequency ω_n is used to reduce the static tracking phase error in a second-order loop; however, a wide natural frequency has an undesired noise filtering performance. In contrast, the zero tracking phase error for a frequency ramp error is concordant with a small loop bandwidth in a third-order loop.

The preceding analysis on tracking process is under the assumption that the phase error is relatively small and the loop is linear. If the phase error is large enough to make the loop drop out of lock, the linear assumption is invalid. For a sinusoidal-characteristic phase detector, the exact phase expression of Eq. 57.17 should be

$$\sin\theta_v = \frac{\Delta\omega}{k_v} \tag{57.20}$$

The sine function has solutions when $\Delta\omega \le k_v$. Therefore, there is no solution if $\Delta\omega > k_v$. This is the case when the loop loses lock and the output of the phase detector will be beat notes signal rather than a dc control voltage. Therefore, k_v can be used to define the *hold range* of the PLL; that is

$$\Delta\omega_H = \pm k_v = k_o k_d F(0) \tag{57.21}$$

The hold range is the frequency range in which a PLL is able to maintain lock *statically*. Namely, if input frequency offset exceeds the hold range statically, the steady-state phase error would drop out of the linear range of the phase detector and the loop loses lock. The hold range expressed in Eq. 57.21 is not correct when some other components in PLL are saturated earlier than the phase detector. k_v is a function of k_o, k_d, and $F(0)$. The dc gain $F(0)$ of the loop filter depends on the filter type. Then hold range $\Delta\omega_H$ can be $k_o k_d$, $k_o k_d k_a$, and ∞ for passive lead-lag filter, active lead-lag filter, and active PI filter, respectively. When the PI filter is used, the real hold range is actually determined by the control range of the VCO.

Considering the dynamic phase error θ_a in a second-order loop, the exact expression for a sinusoidal characteristic phase detector is

$$\sin\theta_a = \frac{\Delta\dot{\omega}}{\omega_n^2} \tag{57.22}$$

which implies that the maximum change rate of the input frequency is ω_n^2. If the rate exceeds ω_n^2, the loop will fall out of lock.

Lock-in Process

The *lock-in* process is defined as PLL locks within one single beat note between the input and the output (VCO output) frequency. The maximum frequency difference between the input and the output that the PLL can lock within one single beat note is called the *lock-in range* of the PLL.

Figure 57.9 is the case in which a frequency offset $\Delta\omega$ is less than the lock-in range. Then, PLL will lock within one single beat note between ω_i and ω_o, and the lock-in process happens. In Fig. 57.10(b), the frequency offset $\Delta\omega$ between input (ω_i) and output (ω_o) is larger than the lock-in range; hence, the lock-in process will not take place, at least not instantaneously.

The magnitude of the lock-in range can be derived approximately in the following. Suppose the PLL is unlocked initially. The input frequency ω_i is $\omega_o + \Delta\omega$. If the input signal $v_i(t)$ is a sine wave and given by

$$v_i(t) = A_i \sin(\omega_o t + \Delta\omega t) \tag{57.23}$$

and the VCO output signal $v_o(t)$ is usually a square wave written as a Walsh function[5]

$$v_o(t) = A_o W(\omega_o t) \tag{57.24}$$

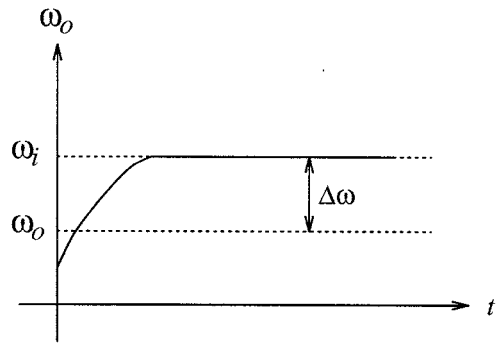


FIGURE 57.9 Lock-in process of the PLL.

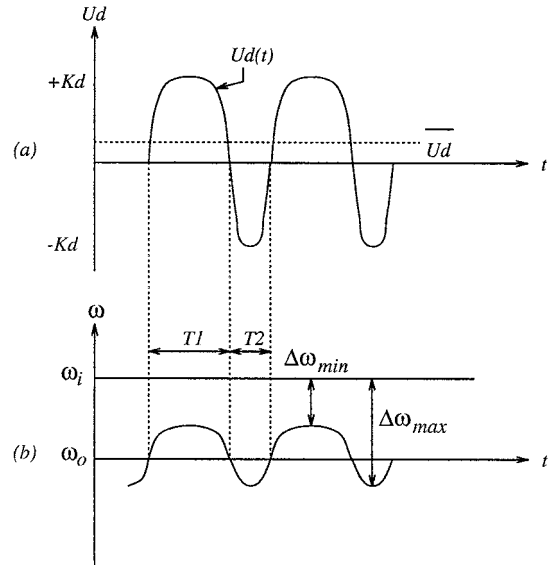


FIGURE 57.10 Pull-in process of the PLL.

$v_o(t)$ can be replaced by the Fourier Series,

$$v_o(t) = A_o\left[\frac{4}{\pi}\cos(\omega_o t) + \frac{4}{3\pi}\cos(3\omega_o t) + \ldots\right] \tag{57.25}$$

So, the phase detector output v_d is

$$\begin{aligned} v_d(t) = v_i(t)v_o(t) &= A_i A_o\left[\frac{2}{\pi}\sin(\Delta\omega t) + \ldots\right] \\ &= k_d \sin(\Delta\omega t) + \textit{high-frequency terms} \end{aligned} \tag{57.26}$$

The high-frequency components can be filtered out by the loop filter. The output of the loop filter is given by

$$v_f(t) \approx k_d \left|F(\Delta\omega)\right| \sin(\Delta\omega t) \tag{57.27}$$

The peak frequency deviation based on Eq. 57.27 is equal to $k_d k_o |F(\Delta\omega)|$. If the peak deviation is larger than the frequency error between ω_i and ω_o, the lock-in process will take place. Hence, the lock-in range is given by

$$\Delta\omega_L = k_d k_o \left|F(\Delta\omega_L)\right| \tag{57.28}$$

The lock-in range is always larger than the corner frequency $\frac{1}{\tau_1}$ and $\frac{1}{\tau_2}$ of the loop filter in practical cases. An approximation of the loop filter gain $F(\Delta\omega_L)$ is shown as follows:

For the passive lead-lag filter,

$$F\left(\Delta\omega_L\right) \approx \frac{\tau_2}{\tau_1 + \tau_2}$$

For the active lead-lag filter,

$$F\left(\Delta\omega_L\right) \approx k_a \frac{\tau_2}{\tau_1}$$

For the active PI filter,

$$F\left(\Delta\omega_L\right) \approx \frac{\tau_2}{\tau_1}$$

τ_2 is usually much smaller than τ_1, and the $F(\Delta\omega_L)$ can be further approximated as follows:

For the passive lead-lag filter,

$$F\left(\Delta\omega_L\right) \approx \frac{\tau_2}{\tau_1}$$

For the active lead-lag filter,

$$F\left(\Delta\omega_L\right) \approx k_a \frac{\tau_2}{\tau_1}$$

For the active PI filter,

$$F\left(\Delta\omega_L\right) \approx \frac{\tau_2}{\tau_1}$$

Substituting the above equations into Eq. 57.28 and assuming a high-gain loop,

$$\Delta\omega_L = 2\zeta\omega_n \tag{57.29}$$

can be obtained for all three types of loop filters shown in Fig. 57.3.

Acquisition Process

Suppose that the PLL does not lock initially, and the input frequency is $\omega_i = \omega_o + \Delta\omega$, where ω_o is the initial frequency of VCO. If the frequency error $\Delta\omega$ is larger than the lock-in range, the lock-in process will not happen. Consequently, the output signal $u_d(t)$ of the phase detector shown in Fig. 57.10(a) is a sine wave that has a frequency $\Delta\omega$. The ac phase detector output signal $u_d(t)$ passes through the loop filter. Then the output $u_f(t)$ of the loop filter modulates the VCO frequency. As shown in Fig. 57.10(b), when ω_o increases, the frequency difference between ω_i and ω_o becomes smaller and vice versa.Therefore,

the phase detector output $u_d(t)$ becomes asymmetric. That is, the duration of positve half-periods of the phase detector output is larger than the negative ones. The average value $\overline{u_d(t)}$ of the phase detector output therefore goes to slightly positive. Then the frequency of VCO will be pulled up until it reaches the input frequency. This phenomenon is called a *pull-in process.*

Because the pull-in process is a non-linear behavior, the mathematical analysis is quite complicated. According to the results in Ref. 1, the pull-in range and the pull-in time depend on the type of loop filter. For an active lead-lag filter with a high-gain loop, the pull-in range is

$$\Delta\omega_P \approx \frac{4\sqrt{2}}{\pi}\sqrt{\zeta\omega_n k_o k_d} \tag{57.30}$$

and the pull-in time is

$$T_P \approx \frac{\pi^2}{16}\frac{\Delta\omega_0^2 k_a}{\zeta\omega_n^3} \tag{57.31}$$

where $\Delta\omega 0$ is the initial frequency error. Equations 57.30 and 57.31 will be modified for different types of phase detectors.[1]

Aided Acquisition

The PLL bandwidth is always too narrow to lock a signal of large frequency error. Furthermore, the frequency acquisition is slow and impractical. Therefore, there are aided frequency-acquisition techniques to solve this problem, such as the frequency-locked loop (FLL) and bandwidth-widening methods.

The frequency-locked loop, which is very similar to a PLL, is composed of a frequency discriminator, a loop filter, and a VCO. PLL is a coherent mechanism to recover a signal buried in noise. FLL, in contrast, is a non-coherent scheme that cannot distinguish between signal and noise. Therefore, an FLL can only be useful to provide signal frequency, which usually implies that the input signal power must exceed the noise.

The major difference between PLL and FLL is the phase detector and the frequency discriminator. The frequency discriminator is the frequency detector in the FLL. It generates a voltage proportional to the frequency difference between the input and the VCO. The frequency difference will be driven to zero in a negative feedback fashion. If a linear frequency detector is employed, it can be shown that the frequency-acquisition time is proportional to the logarithm of the frequency error.[6] In the literature, some frequency detectors like the quadricorrelator,[7] balance quadricorrelator,[8] rotational frequency detector,[9] and frequency delimiter[10] are disclosed.

Delay-Locked Loop

Two major approaches for adjustable timing elements are VCO and voltage-controlled delay line (VCDL). Figure 57.11 shows a typical delay-locked loop (DLL)[11,12] that replaces the VCO of a PLL with a VCDL. The input signal is delayed by an integer multiple of the signal period because the phase error is zero when the phase difference between V_{in} and V_o approaches a multiple of the signal periods. The VCDL usually consists of a number of cascaded gain stages with variable delay. Delay lines, unlike ring oscillators, cannot generate a signal; therefore, it is difficult to make frequency multiplication in a DLL.

In a VCO, the output "frequency" is proportional to the input control voltage. The phase transfer function contains a pole, which is $H(s) = \frac{k_o}{s}$ (k_o is the VCO gain). In a VCDL, the output "phase" is proportional to the control voltage, and the phase transfer function is $H(s) = k_{VCDL}$. So, the DLL can be

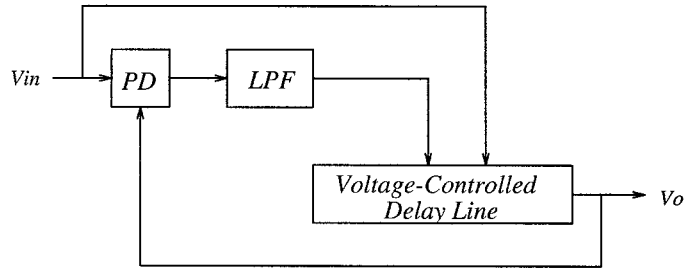


FIGURE 57.11 DLL block diagram.

easily stabilized with a simple first-order loop filter. Consequently, DLLs have much more relaxed tradeoffs among gain, bandwidth, and stability. This is one of the two important advantages over PLLs. Another advantage is that delay lines typically introduce much less jitter than a VCO.[13] Because a delay chain is not configured as a ring oscillator, there is no jitter accumulation because the noise does not contribute to the starting point of the next clock cycle.

Charge-Pump Phase-Locked Loop

A charge-pump PLL usually consists of four major blocks as shown in Fig. 57.12. The phase detector is a purely phase-frequency detector. The charge-pump circuit converts the digital signals UP, DN, and null (neither up nor down) generated by the phase detector into a corresponding charge-pump current I_p, $-I_p$, and zero. The loop filter is usually a passive RC circuit converting the charge-pump current into an analog voltage to control the VCO. So, the purpose of the "charge-pump" is to convert the logic state of the phase-frequency detector output into an analog signal suitable for controlling the voltage-controlled oscillator. The linear model of a charge-pump PLL is shown in Fig. 57.13. The k_d is the equivalent gain of a charge-pump circuit and a loop filter, which is shown in Fig. 57.14. If the loop bandwidth is much smaller than the input frequency, the detailed behavior within a single cycle can be ignored. Then, the state of a PLL can be assumed to be only changed by a small amount during each input cycle. Actually, the "average" behavior over many cycles is interesting. Then, the average current charging the capacitor is given by

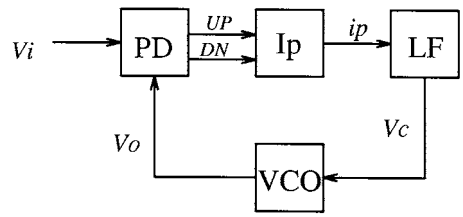


FIGURE 57.12 Charge-pump PLL diagram.

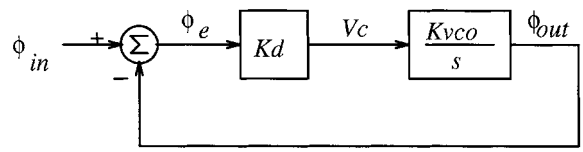


FIGURE 57.13 The linear model of charge-pump PLL.

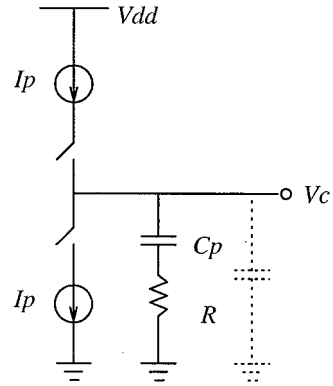


FIGURE 57.14 The schematic of a loop filter

$$\begin{aligned} I_{avg} &= \frac{Q}{T} = \frac{I\Delta t}{T} \\ &= \frac{I\left(\frac{\phi e}{2\pi}\right)T}{T} \\ &= \frac{I\phi_e}{2\pi} \end{aligned} \tag{57.32}$$

And the average k_d in Fig. 57.13 is

$$k_d \triangleq \frac{v_c}{\phi_e} = \frac{I}{2\pi}\left(R + \frac{1}{C_p s}\right) \tag{57.33}$$

The closed-loop transfer function can be obtained as

$$\begin{aligned} H(s) \triangleq \frac{\phi_{out}}{\phi_{in}} &= \frac{k_d \frac{k_{vco}}{s}}{1 + \frac{k_d k_{vco}}{s}} \\ &= \frac{\frac{I}{2\pi C_p}\left(RC_p s + 1\right)k_{vco}}{s^2 + \frac{I}{2\pi}k_{vco}Rs + \frac{Ik_{vco}}{2\pi C_p}} \end{aligned} \tag{57.34}$$

Generally, a second-order system is characterized by the natural frequency $f_n = \frac{\omega_n}{2\pi}$ and the damping factor ζ. So,

$$\begin{aligned} \omega_n &= \sqrt{\frac{I}{2\pi C_p}k_{vco}}\ rad\ \sec \\ \zeta &= \frac{RC_p}{2}\omega_n \end{aligned} \tag{57.35}$$

For the stability consideration, there is a limitation of a normalized natural frequency F_N:[14]

$$F_N \triangleq \frac{f_n}{f_i} < \frac{\sqrt{1+\zeta^2}-\zeta}{\pi} \tag{57.36}$$

In the single-ended charge pump, the resistor added in series with the capacitor shown in Fig. 57.14 can introduce "ripple" in the control voltage V_c even when the loop is locked.[15] The ripple control voltage modulates the VCO frequency and results in phase noise. This effect is especially undesired in frequency synthesizers. In order to suppress the ripple, a second-order loop filter, as shown in Fig. 57.14 with a shunt capacitor in dotted lines, is used. This configuration introduces a third pole in the PLL. Stability issues must be taken care of. Gardner[15] provides criteria for the stability of the third-order PLL.

An important property of any PLL is the static phase error that arises from a frequency offset $\Delta\omega$ between the input signal and the free-running frequency of the VCO. According to the analysis of Ref. 15, the static phase error is

$$\theta_v = \frac{2\pi\Delta\omega}{k_o I_p F(0)} rad \tag{57.37}$$

To eliminate the static phase error in conventional PLLs, an active loop filter with a high dc gain ($F(0)$ is large) is preferred. But the charge-pump PLL allows zero static phase error without the need for a large dc gain of the loop filter. This effect arises from the input open circuit during the "null" state (charge-pump current is zero). Real circuits will impose some resistive loading R_s in parallel to the loop filter. Therefore, the static phase error, from Eq. 57.37 will be

$$\theta_v = \frac{2\pi\Delta w}{k_o I_p R_s} rad \tag{57.38}$$

The shunt resistive loading most likely comes from the input of a VCO control terminal. Compared with the static phase error of a conventional PLL as expressed in Eq. 57.17, the same performance can be obtained from a charge-pump PLL without a high dc-gain loop filter.[16]

PLL Noise Performance

In high-speed data recovery applications, better performance of the VCO and the overall phase-locked loop itself is desired. The random variations of the sampling clock, so-called jitter, is the critical performance parameter.

Jitter sources of PLL, when using a ring voltage-controlled oscillator, mainly come from the input and the VCO itself. The ring oscillator jitter is associated with the power supply noise, the substrate noise, $1/f$ noise, and the thermal noise. The former two noise sources can be reduced by fully differential circuit structure. $1/f$ noise, on the other hand, can be rejected by the tracking capability of the PLL. Therefore, the thermal noise is the worst noise source. From the analysis in Ref. 17, the one-state RMS timing jitter error of the ring oscillator normalized to the time delay per stage can be shown as

$$\frac{\Delta\tau_{rms}}{t_d} \approx \sqrt{\frac{2KT}{C_L}}\left(\sqrt{1+\frac{2}{3}a_v}\right)\frac{1}{V_{pp}} \tag{57.39}$$

where C_L is the load capacitance, $\sqrt{1+\frac{2}{3}a_v}$ is called the noise contribution factor ς, a_v is the small-signal gain of the delay cell, and Vpp is the VCO output swing. From Eq. 57.39, for a fixed output bandwidth, higher gain contributes larger noise.

Because the ring oscillator is a feedback architecture, the noise contribution of a single delay cell may be amplified and filtered by the following stage. To consider two successive stages, Eq. 57.39 can be rearranged as:[17]

$$\frac{\Delta\tau_{rms}}{t_d} \approx \sqrt{\frac{2KT}{C_L}} \frac{1}{\left(V_{gs} - V_t\right)} \varsigma \tag{57.40}$$

Therefore, the cycle-to-cycle jitter of the ring oscillator in a PLL can be predicted[17] by

$$\overline{\left(\Delta\tau_N\right)^2} = \frac{KT}{I_{ss}} \frac{a_v \varsigma^2}{\left(V_{gs} - V_t\right)} T_o \tag{57.41}$$

where I_{ss} is the current of the delay cell, and T_o is the output period of the VCO. Based on Eq. 57.41, designing a low jitter VCO, $(V_{gs} - V_t)$ should be as large as possible. For fixed delay and fixed current, a lower gain of each stage is better for jitter performance, but the loop gain must satisfy the Barkhausen criterion. From the viewpoint of VCO jitter, a wide bandwidth of the PLL can correct the timing error of the VCO rapidly.[13] If the bandwidth is too wide, the input noise jitter may be so large that it dominates the jitter performance of the PLL. Actually, this is a tradeoff.

For a phase-locked loop design, the natural frequency and the damping factor are the key parameters to be determined by designers. If the signal-to-noise ratio $(\text{SNR})_i$ is defined, then the output signal-to-noise ratio $(\text{SNR})_o$ can be obtained:[4]

$$\left(\text{SNR}\right)_o = \left(\text{SNR}\right)_i \frac{B_i}{2B_L} \tag{57.42}$$

where B_i is the bandwidth of the prefilter and B_L is the noise bandwidth. Hence, the B_L can be derived using Eq. 57.42. And the relationship of B_L with ω_n and ζ is

$$B_L = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\zeta}\right) \tag{57.43}$$

Therefore, the ω_n and ζ can be designed to satisfy the $(\text{SNR})_o$ requirement.

Besides the system and circuit designs, jitter can be reduced in the board level design. Board jitter can be alleviated by better layout and noise decoupling schemes like such as appending proper decouple and bypass capacitances.

PLL Design Considerations

A PLL design starts with specifying the key parameters such as natural frequency ω_n, lock-in range $\Delta\omega_L$, damping factor ζ, and the frequency control range which depend significantly on applications. Design procedures based on a practical example will be described as follows:

Step 1. Specify the damping factor ζ. The damping factor of the PLL determines the responses of phase or frequency error steps applied to the input. ζ should be considered to achieve fast response, small overshoot, and minimum noise bandwidth B_L. If ζ is very small, large overshoot will occur and the overshoot causes phase jitter.[18] If ζ is too large, the response will become sluggish.

Step 2. Specify the lock-in range $\Delta\omega_L$ or the noise bandwidth B_L. As shown in Eqs. 57.29 and 57.43, the natural frequency ω_n depends on $\Delta\omega_L$and ζ (or B_L and ζ). If the noise is not the key

issue of the PLL, one can ignore the noise bandwidth and specify the lock-in range. Where noise is of concern, one should specify B_L first, and keep the lock-in range of the PLL.

Step 3. Calculate the ω_n according to Step 2. If the lock-in range has been specified, Eq. 57.29 indicates that

$$\omega_n = \frac{\Delta\omega_L}{2\zeta} \tag{57.44}$$

If the noise bandwidth has been specified, Eq. 57.43 indicates the natural frequency as

$$\omega_n = \frac{2B_L}{\zeta + \frac{1}{4}\zeta} \tag{57.45}$$

Step 4. Determine the VCO gain factor k_o and the phase detector gain k_d. k_o and k_d are both characterized by circuit architectures and they must achieve the requirement of the lock-in range specified in Step 2. For example, if k_o or k_d is too small, the PLL will fail to achieve the desired lock-in range.

Step 5. Choose the loop filter. Different types of loop filters are available, as shown in Fig. 57.3. Eqs. 57.7 to 57.9, ω_n and ζ (specified above) are used to derive the time constants of the loop filter.

57.3 Building Blocks of the PLL Circuit

Voltage-Controlled Oscillators

The function of a voltage-controlled oscillator (VCO) is to generate a stable and periodic waveform whose frequency can be varied by an applied control voltage. The relationship between the control voltage and the oscillation frequency depends upon the circuit architecture. A linear characteristic is generally preferred because of its wider applications. As a general classification, VCOs can be roughly categorized into two types by the output waveforms: (1) *harmonic oscillators* that generate nearly sinusoidal outputs, and (2) *relaxation oscillators* that provide square or triangle outputs. In general, a harmonic oscillator is composed of an amplifier that provides an adequate gain and a frequency-selective network that feeds a certain output frequency range back to the input. LC-tank oscillators and crystal oscillators belong to this type.

Relaxation oscillators are the most commonly used oscillator configuration in monolithic IC design because they can operate in a wide frequency range with a minimum number of external components. According to the mechanism of the oscillator topology employed, relaxation oscillators can be further categorized into three types: (1) grounded capacitor VCO,[19] (2) emitter-coupled VCO, and (3) delay-based ring VCO.[20] The operation of the first two oscillators is similar in the sense that time duration spent in each state is determined by the timing components and charge/discharge currents. The delay-based ring VCO operates quite differently since the timing relies on the delay in each of the gain stages that are connected in a ring configuration.

Generally, harmonic oscillators have the following advantages: (1) superior frequency stability, which includes the frequency stability with temperature, power supply, and noise; and (2) good frequency accuracy control, because the oscillation frequency is determined by a tank circuit or a crystal.

Nevertheless, harmonic oscillators are not compatible with monolithic IC technology and their frequency turning range is limited. On the contrary, relaxation oscillators are easy to implement in monolithic ICs. Since frequency is normally proportional to a controlled-current or -voltage and inversely proportional to timing capacitors, the frequency of oscillation can be varied linearly over a very wide

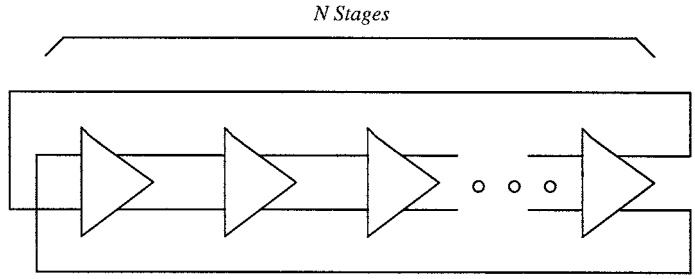


FIGURE 57.15 Ring oscillator.

range. Coming from the ease of frequency tuning, the drawbacks of such oscillators are poor frequency stability and frequency inaccuracy.

Recently, the ring oscillator has received considerable attentions in high-frequency PLL applications for clock synchronization and timing recovery. Since they can provide high-frequency oscillation with simple digital-like circuits that are compatible with digital technology, they are suitable for VLSI implementations.

In order to achieve high rejection of power supply and substrate noises, both the signal path and the control path of a VCO must be fully differential. A common ring oscillator topology in monolithic PLLs is shown in Figure 57.15. The loop oscillates with a period equal to $2NT_d$ where T_d is the delay of each stage. The oscillation can be obtained when the total phase shift is zero and the loop gain is greater or equal to unity at a certain frequency. To vary the frequency of oscillation, the effective number of stages or the delay of each stage must be changed. The first approach is called "delay interpolating" VCO,[20] where a shorter delay path and a longer delay path are used in parallel. The total delay is tuned by increasing the gain of one path and decreasing the other, and the total delay is a weighted sum of the two delay paths. The second approach is to vary the delay time of each stage to adjust the oscillation frequency. The delay of each stage is tuned by varying the capacitance or the resistance seen at the output node of each stage. Because the tuning range of the capacitor is small and the maximum oscillation frequency is limited by the minimum value of the load capacitor, it makes the "resistive tuning" a better alternative technique. The resistive tuning method provides a large, uniform frequency tuning range and lends itself easily to differential control. In Figure 57.16(a), the on-resistance of the triode PMOS loads are adjusted by V_{cont}. The more V_{cont} decreases, the more the delay of the stage drops; because the time constant at the output node decreased, the small-signal gain decreases too. The circuit eventually fails to oscillate when the loop gain at the oscillation frequency is less than unity. In Fig. 57.16(b), the delay of the gain stage is tuned by adjusting the tail current, but the small-signal gain remains constant. So, the circuit is better than Fig. 57.16(a). As shown in Fig. 57.16(c),[21] the PMOS current source with a pair of cross-coupled diode loads provides a differential load impedance that is independent of common-mode voltage. This makes the cell delay insensitive to common-mode noise. Figure 57.16(d) is a poor delay cell for a ring oscillator because the tuning range is very small.

The minimum number of stages that can be used while maintaining reliable operation is an important issue in a ring oscillator design. When the number of stages decreases, the required phase shift and dc gain per stage increases. Two-stage bipolar ring oscillators can be designed reliably,[22] but CMOS implementations cannot. Thus, CMOS ring oscillators typically utilize three or more stages.

Phase and Frequency Detectors

The phase detector type has influence on the dynamic range of PLLs. Hold range, lock-in range, and pull-in range are analyzed in Section 57.2, based on the multiplier phase detector. Most other types of phase detectors have a greater linear output span and a larger maximum output swing than a sinusoidal characteristic phase detector. A larger tracking range and a larger lock limit are available if the linear output

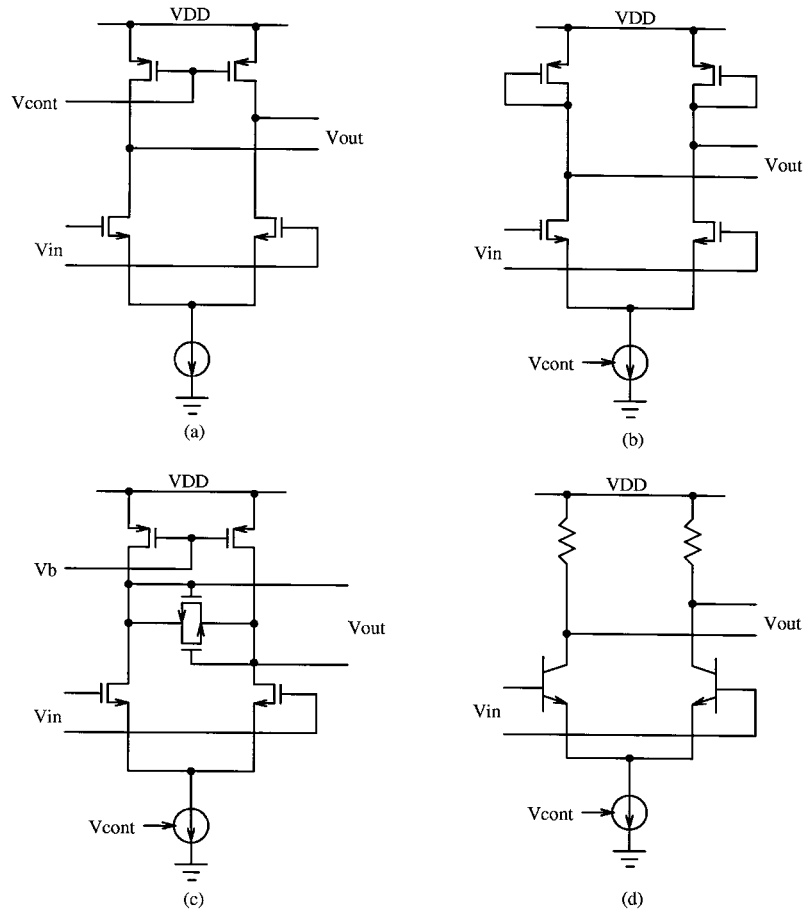


FIGURE 57.16 The gain stages using resistive tuning.

range of the PD increases. The three widely used phase detectors are XOR PD, edge-triggered JK-flipflop, and PFD (phase-frequency detector). The characteristics of these phase detectors are plotted in Fig. 57.17.

The XOR phase detector can maintain phase tracking when the phase error θ_e is confined in the range of

$$\frac{-\pi}{2} < \theta_e < \frac{\pi}{2}$$

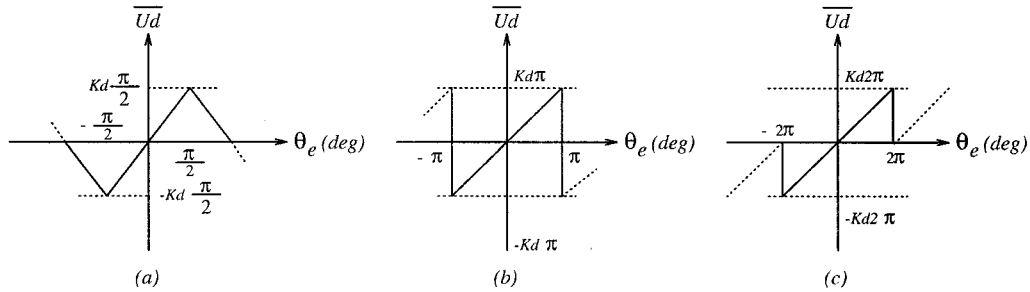


FIGURE 57.17 Phase detector characteristics of (a) XOR, (b) JK-flipflop, and (c) PFD.

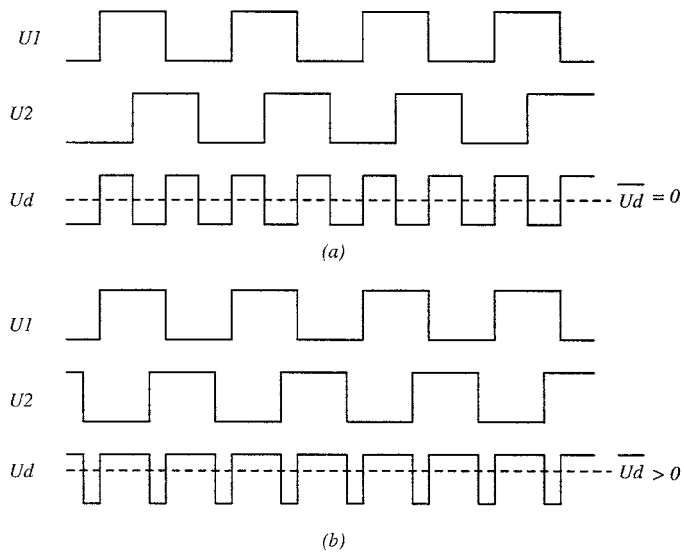


FIGURE 57.18 Waveforms of the signals for the XOR phase detector: (a) waveforms at zero phase error, and (b) waveforms at positive phase error.

as shown in Fig. 57.17(a). The zero phase error takes place when the input signal and the VCO output are quadrature in phase as shown in Fig. 57.18(a). As the phase difference deviates from $\frac{\pi}{2}$, the output duty cycle is no longer 50%, which provides a dc value proportional to the phase difference as shown in Fig. 57.18(b). But the XOR phase detector has a steady-state phase error if the input signal or the VCO output are asymmetric.

The JK-flipflop phase detector shown in Fig. 57.19, also called a two-state PD, is barely influenced by the asymmetric waveform because it is edge-triggered. The zero phase error happens when the input signal and the VCO output are out-of phase as illustrated in Fig. 57.19(a). As shown in Fig. 57.17(b), the JK-flipflop phase detector can maintain phase tracking when the phase error is within the range of

$$-\pi < \theta_e < \pi$$

Here, a positive edge appearing at the "J" input triggers the flipflop into "high" state (Q = 1), and the rising edge of u_2 drives Q to zero. Figure 57.19(b) shows the output waveforms of the JK-flipflop phase detector for $\theta_e > 0$.

The PFD output depends not only on the phase error, but also on the frequency error. The characteristic is shown in Fig. 57.17(c). When the phase error is greater than 2π, the PFD works as a frequency detector. The operation of a typical PFD is as follows, and the waveforms are shown in Fig. 57.20. If the frequency of input A, ω_A, is less than the frequency of input B, ω_B, then the PFD produces positive pulses at Q_A, while Q_B remains at zero. Conversely, if $\omega_A > \omega_B$, the positive pulses appear at Q_B while $Q_A = 0$. If $\omega_A = \omega_B$, then the PFD generates pulses at either $Q_A$or Q_B with a width equal to the phase difference between the two inputs. The outputs $Q_A$and Q_B are usually called the "up" and "down" signals, respectively. If the input signal fails, which usually happens at the NRZ data recovery applications during missing or extra transmissions, the output of the PFD would stick on the high state (or low state). This condition may cause the VCO to oscillate fast or slow abruptly, which results in noise jitter or even losing lock. This problem can be remedied by additional control logic circuits to make the PFD output toggle back and forth between the two logic levels with 50% duty cycle,[18] the loop is interpreted as zero phase error. The "rotational FD" described by Messerschmitt can also solve this issue.[9] The output of a PFD can be converted to a dc control voltage by driving a three-state charge-pump, as described in Section 57.2.

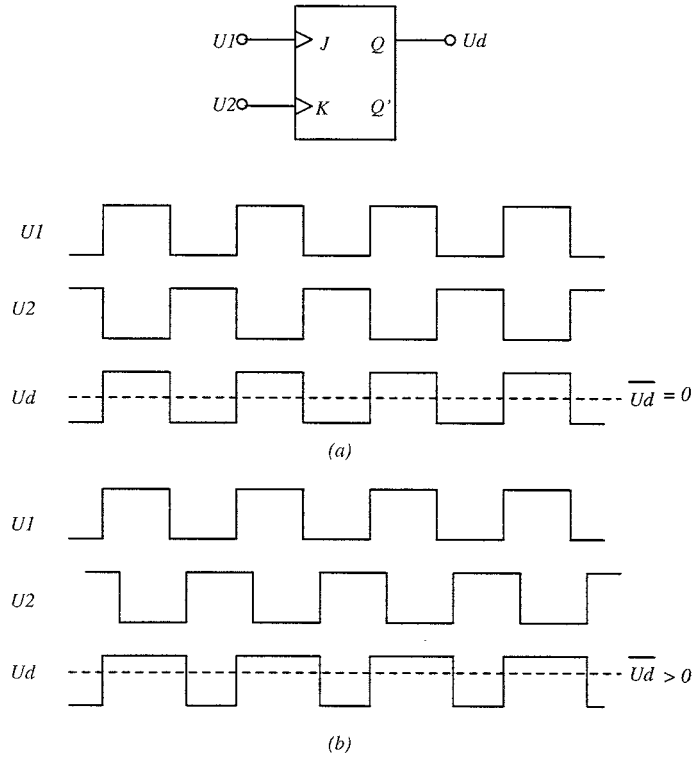


FIGURE 57.19 Waveforms of the signals for the JK-flipflop phase detector: (a) waveforms at zero phase error, (b) waveforms at positive phase error

57.4 PLL Applications

Clock and Data Recovery

In data transmission systems such as optical communications, telecommunications, disk drive systems, and local networks, data is transmitted on baseband or passband. In most of these applications, only data signals are transmitted by the transmitter; clock signals are not transmitted in order to save hardware cost. Therefore, the receiver should have some scheme to extract the clock information from the received data stream and regenerate transmitted data using the recovery clock. This scheme is called *timing recovery* or *clock recovery.*

To recover the data correctly, the receiver must generate a synchronous clock from the input data stream, and the recovered clock must synchronize with the bit rate (the baud of data). The PLL can be used to recover the clock from the data stream, but there are some special design considerations. For example, because of the random nature of data, the choice of phase-frequency detectors is restricted. In particular, a three-state PD is not proper; because of missing data transitions, the PD will interpret the VCO frequency to be higher than the data frequency, and the PD output stays on "down" state to make the PLL lose lock, as shown in Fig. 57.21. Thus, the choice of phase-frequency detector for random binary data requires a careful examination of their responses when some transitions are absent. One useful method is the rotational frequency detector described in Ref. 9. The random data also causes the PLL to introduce undesired phase variation in the recovered clock; this is called *timing jitter* and is an important issue of the clock recovery.

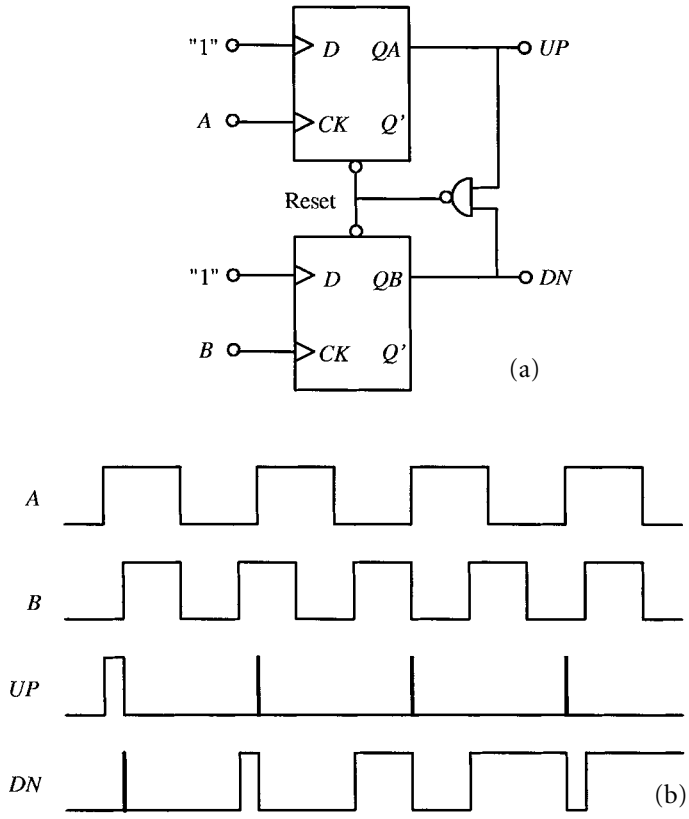


FIGURE 57.20 (a) PFD diagram and (b) input and output waveforms of PFD.

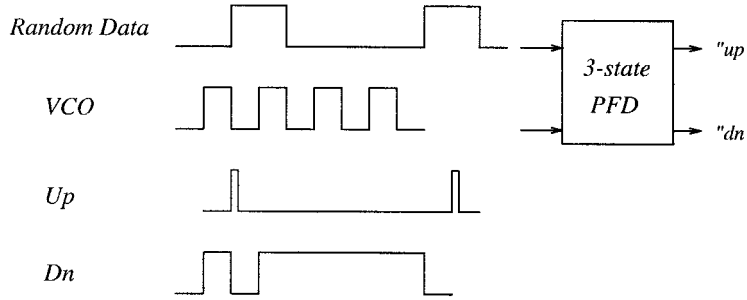


FIGURE 57.21 Response of a three-state PD to random data.

Data Format

Binary data is usually transmitted in a NRZ (Non-Return-to-Zero) format, as shown in Fig. 57.22(a), because of the consideration of bandwidth efficiency. In NRZ format, each bit has a duration of T_B (bit period). The signal does not go to zero between adjacent pulses representing 1's. It can be shown[23] in that the corresponding spectrum has no line component at $f_B = \frac{1}{T_B}$; most of the spectrum of this signal lines below $\frac{f_B}{2}$. The term "non-return-to-zero" distinguishes from another data type called "return-to-zero" (RZ), as shown in Fig. 57.22(b), in which the signal goes to zero between consecutive bits. Therefore,

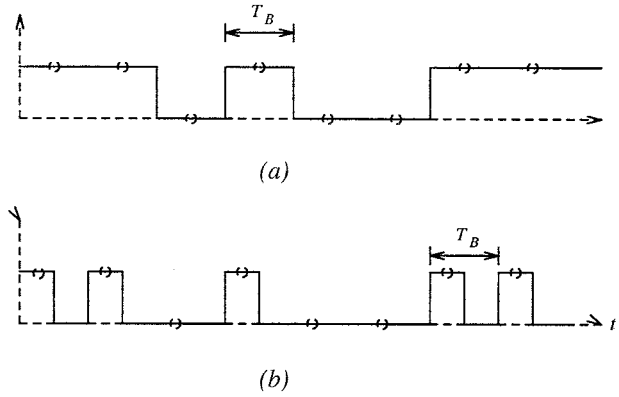


FIGURE 57.22 (a) NRZ data and (b) RZ data.

the spectrum of RZ data has a frequency component at f_B. For a given bit rate, RZ data needs wider transmitting bandwidth; therefore, NRZ data is preferable when channel or circuits bandwidth is a concern.

Due to the lack of a spectral component at the bit rate of NRZ format, a clock recovery circuit may lock to spurious signals or fail to lock at all. Thus, a non-linear process at NRZ data is essential to create a frequency component at the baud rate.

Data Conversion

One way to recover a clock from NRZ data is to convert it to RZ-like data that has a frequency component at bit rate, and then recover clock from data using a PLL. Transition detection is one of the methods to convert NRZ data to RZ-like data. As illustrated in Fig. 57.23(a), the edge detection requires a mechanism to sense both positive and negative data transitions. In Fig. 57.23(b), NRZ data is delayed and compared with itself by an exclusive-OR gate; therefore, the transition edges are detected. In Fig. 57.24, the NRZ data V_i is first differentiated to generate pulses corresponding to each transition. These pulses are made to be all positive by squaring the differentiated signal $\dot{v}_i$. The result is the signal V_i' that looks just like RZ data, where pulses are spaced at an interval of T_B.

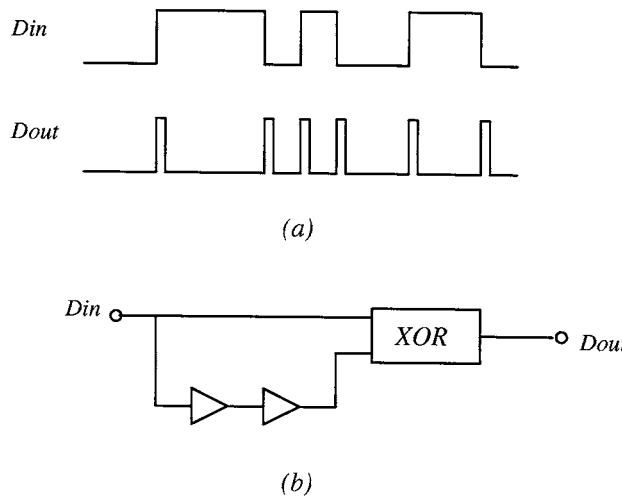


FIGURE 57.23 Edge detection of NRZ data.

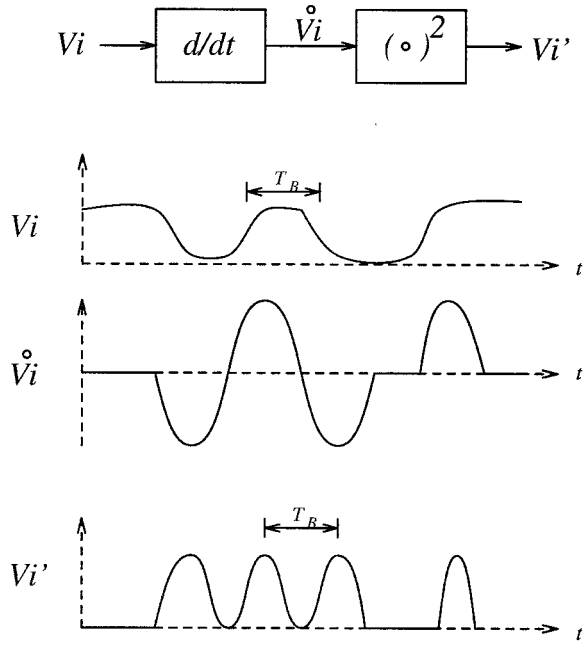


FIGURE 57.24 Converting NRZ to RZ-like signal.

Clock Recovery Architecture

Based on different PLL topologies, there are several clock recovery approaches. Here, the early-late and the edge-detector based methods are described.

Figure 57.25 shows the block diagram of the early-late method. If the input lags the VCO output, Fig. 57.26 shows the waveforms for this case. In Fig. 57.26, the early integrator integrates the input signal for the early-half period of the clock signal and holds it for remainder of the clock signal. On the other hand, the late integrator integrates the input signal for the late-half period of the clock signal and holds it for the next early-half period. The average difference between the absolute values of the late hold and the early hold voltage generated from a low-pass filter gives the control signal to adjust the frequency of the VCO. As mentioned above, this method is popular for rectangular pulses. However, there are some drawbacks to this method. Since this method relies on the shape of pulses, a static phase error can be introduced if the pulse shape is not symmetric. In high-speed applications, this approach requires a fast settling integrator that limits the operating speed of the clock recovery circuit and the acquisition time cannot be easily controlled.

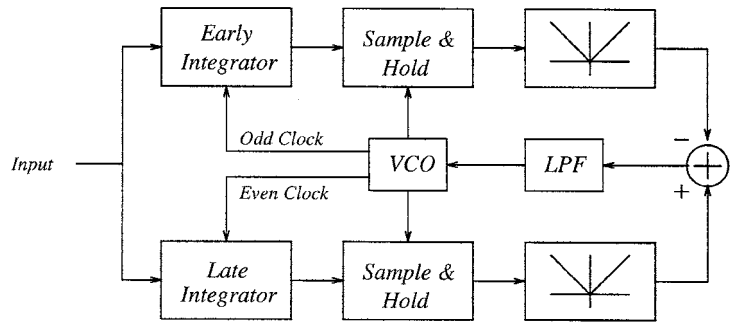


FIGURE 57.25 Early-late block diagram.

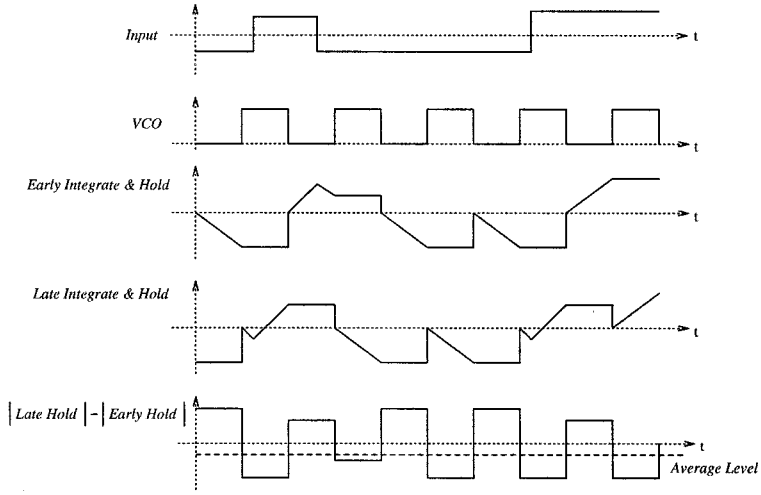


FIGURE 57.26 Clock waveforms for early-late architecture.

The most widely used technique for clock recovery in high-performance, wide-band data transmission applications is the edge-detection based method. The edge-detection method is used to convert data format such that the PLL can lock the correct band frequency. More details were given in the previous subsection. There are many variations of this method, depending on the exact implementation of each PLL loop component. The "quadricorrelator" introduced by Richman[7] and modified by Bellisio[24] is a frequency-difference discriminator and has been implemented in a clock recovery architecture. Figure 57.27 is a phase-recovery locked loop using edge-detection method and quadricorrelator to recover timing information from NRZ data.[25] As shown in Fig. 57.27, the quadricorrelator follows the edge-detector with a combination of three loops sharing the same VCO. Loop I and II form a frequency-locked loop that contains the quadricorrelator for frequency detection. Loop III is a typical phase-locked loop for phase alignment. The phase- and frequency-locked loops share the same VCO; the interaction between two loops is a very important issue. As described in Ref. 25, when $\omega_1 \approx \omega_2$, the dc feedback signal produced by loop I and II approaches zero, and loop III dominates the loop performance. A composite frequency- and phase-locked loop is a good method to achieve fast acquisition and a narrow PLL loop bandwidth

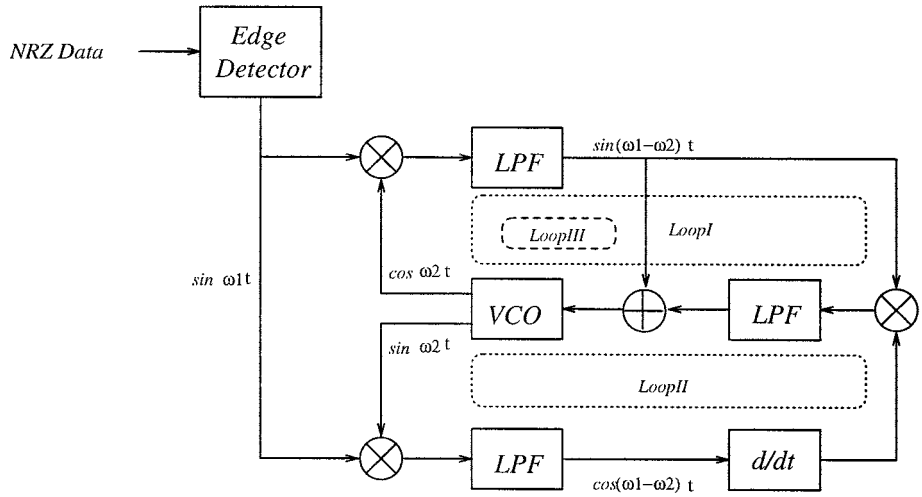


FIGURE 57.27 Quadricorrelator.

to minimize the VCO drift. Nevertheless, because the wide band frequency-locked loop can respond to noise and spurious components, it is essential to disable the frequency-locked loop when the frequency error gets into the lock-in range of the PLL to minimize the interaction. More clock recovery architectures are described in Refs. 18, 20, 22, 26–28.

Frequency Synthesizer

A frequency synthesizer generates any of a number of frequencies by locking a VCO to an accurate frequency source such as a crystal oscillator. For example, RF systems usually require a high-frequency local oscillator whose frequency can be changed in small and precise steps. The ability to multiply a reference frequency makes PLLs attractive for synthesizing frequencies.

The basic configuration used for frequency synthesis is shown in Fig. 57.28(a). This system is capable of generating an integer multiple frequency of a reference frequency. A quartz crystal is usually used as the reference clock source because of its low jitter characteristic. Due to the limited speed of a CMOS device, it is difficult to generate frequency directly in the range of GH_z or more. To generate higher frequencies, prescalers are used; they are implemented with other IC technologies such as ECL. Figure 57.28(b) shows a synthesizer structure using a prescaler V; the output frequency becomes

$$f_{out} = \frac{NVf_i}{M} \tag{57.46}$$

Because the scaling factor V is much greater than one, obviously, it is no longer possible to generate any desired integer multiple of the reference frequency. This drawback can be circumvented by using a so-called dual-modulus prescaler, as shown in Fig. 57.29. A dual-modulus prescaler is a divider whose division can be switched from one value to another by a control signal. The following shows that the dual-modulus prescaler makes it possible to generate a number of output frequencies that are spaced only by one reference frequency. The VCO output is divided by V/V+1 dual-modulus prescaler. The

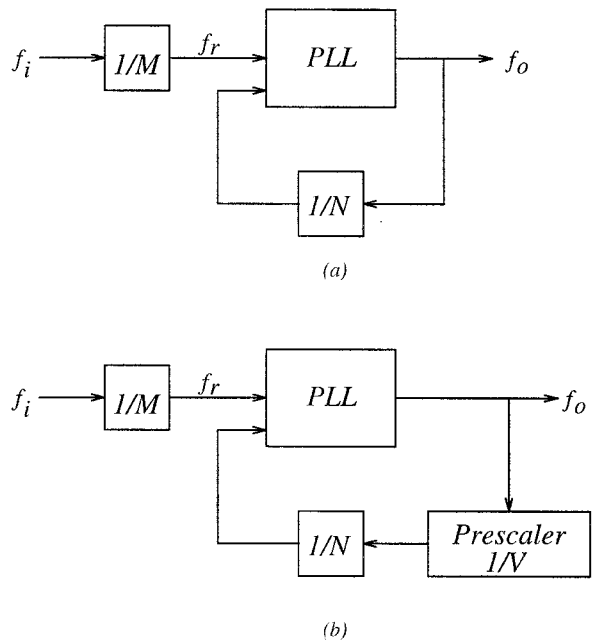


FIGURE 57.28 Frequency-synthesizer block diagrams: (a) basic frequency-synthesizer system; (b) system extends the upper frequency range by using an additional high-speed prescaler.

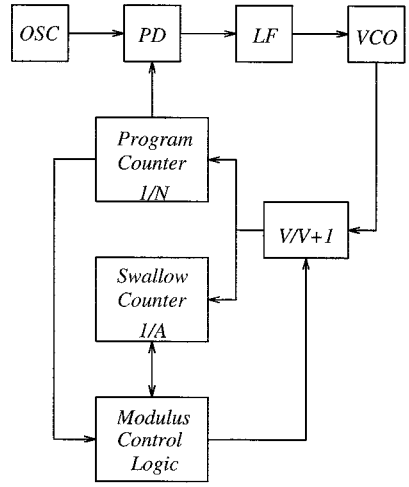


FIGURE 57.29 The block diagram of dual-modulus frequency synthesizer.

output of the prescaler is fed into a "program counter" $\frac{1}{N}$ and a "swallow counter" $\frac{1}{A}$. The dual-modulus prescaler is set to divide by V+1 initially. After "A" pulses out of the prescaler, the swallow counter is full and changes the prescaler modulus to V. After additional "N-A" pulses out of the prescaler, the program counter changes the prescaler modulus back to V+1, restarts the swallow counter, and the cycle is repeated. In this way, the VCO frequency is equal to $(V+1)A + V(N-A) = VN + A$ times the reference frequency. Note that N must be larger than A. If this is not the case, the program counter would be full earlier than $\frac{1}{A}$, and both counters would be reset. Therefore, the dual-modulus prescaler would never be switched from $V+1$ to V. For example, if V = 64, then A must be in the range of 0 to 63 such that $N_{min} = 64$. The smallest realizable division ratio is

$$\left(N_{tot}\right)_{min} = N_{min} V = 4096 \tag{57.47}$$

The synthesizer of Fig. 57.29 is able to generate all integer multiples of the reference frequency, starting from $N_{tot} = 4096$. For extending the upper frequency range of frequency synthesizers, but still allowing the synthesis of lower frequency, the four-modulus prescaler is a solution.[1]

Based on the above discussions, the synthesized frequency is an integer multiple of a reference frequency. In RF applications, the reference frequency is usually larger than the channel spacing for loop dynamic performance considerations, in which the wider loop bandwidth for a given channel spacing allows faster settling time and reduces the phase jitter requirements to be imposed on the VCO. Therefore, a "fractional" scaling factor is needed. Fractional division ratios of any complexity can be realized. For example, a ratio of 3.7 is obtained if a counter is forced to divide by 4 in seven cycles of each group of ten cycles and by 3 in the remaining three cycles. On the average, this counter effectively divides the input frequency by 3.7.

References

1. R. E. Best, *Phase-Locked Loops Theory, Design, Applications,* McGraw-Hill, New York, 1984.
2. D. G. Troha and J. D. Gallia, Digital phase-locked loop design using S-N54/74LS297, Application Note AN 3216, Texas Instruments Inc., Dallas, TX.
3. W. B. Rosink, All-digital phase-locked loops using the 74HC/HCT297, *Philips Components,* 1989.
4. F. M. Gardner, *Phaselock Techniques,* 2nd ed.

5. S. G. Tzafestas, *Walsh Functions in Signal and Systems Analysis and Design,* Van Nostrand, 1985.
6. F. M. Gardner, Acquisition of phaselock, *Conference Record of the International Conference on Communications,* vol. I, pp. 10-1 to 10-5, June 1976.
7. D. Richman, Color carrier reference phase synchronization accuracy in NTSC color television, *Proc. IRE,* vol. 42, pp. 106-133, Jan. 1954.
8. F. M. Gardner, Properties of frequency difference detector, *IEEE Trans. on Communication,* vol. COM-33, no. 2, pp. 131-138, Feb. 1985.
9. D. G. Messerschmitt, Frequency detectors for PLL acquisition in timing and carrier recovery, *IEEE Trans. on Communication,* vol. COM-27, no. 9, pp. 1288-1295, Sept. 1979.
10. R. B. Lee, Timing recovery architecture for high speed data communication system, Masters thesis, 1993.
11. M. Bazes, A novel precision MOS synchronous delay lines, *IEEE J. Solid-State Circuits,* vol. 20, pp. 1265-1271, Dec. 1985.
12. M. G. Johnson and E. L. Hudson, A variable delay line PLL for CPU-coprocessor synchronization, *IEEE J. Solid-State Circuits,* vol. 23, pp. 1218-1223, Oct. 1988.
13. B. Kim, T. C. Weigandt, and P. R. Gray, PLL/DLL systems noise analysis for low jitter clock synthesizer design, *ISCAS Proceedings,* pp. 31-35, 1994.
14. M. V. Paemel, Analysis of a charge-pump PLL: a new model, *IEEE Trans. on Comm.,* vol. 42, no. 7, pp. 131-138, Feb. 1994.
15. F. M. Gardner, Charge-pump phase-locked loops, *IEEE Trans. on Comm.,* vol. COM-28, pp. 1849-1858, Nov. 1980.
16. F. M. Gardner, Phase accuracy of charge pump PLL's, *IEEE Trans. on Comm.,* vol. COM-30, pp. 2362-2363, Oct. 1982.
17. T. C. Weigandt, B. Kim, and P. R. Gray, Analysis of timing recovery jitter in CMOS ring oscillator, *ISCAS Proceedings,* pp. 27-30, 1994.
18. T. H. Lee and J. F. Bulzacchelli, A 155-MHz clock recovery delay- and phase-locked loop, *IEEE J. of Solid-State Circuits,* vol. 27, no. 12, pp. 1736-1746, Dec. 1992.
19. M. P. Flyun and S. U. Lidholm, A 1.2 μm CMOS current-controlled oscillator, *IEEE J. Solid-State Circuits,* vol. 27, no. 7, pp. 982-987, July 1992.
20. S. K. Enam and A. A. Abidi, NMOS IC's for clock and data regeneration in gigabit-per-second optical-fiber receivers, *IEEE J. Solid-State Circuits,* vol. 27, no. 12, pp. 1763-1774, Dec. 1992.
21. M. Horowitz et al., PLL design for a 500MB/s interface, *ISSCC Digest Technical Paper,* pp. 160-161, Feb. 1993.
22. A. Pottbacker and U. Langmann, An 8GHz silicon bipolar clock-recovery and data-regenerator IC, *IEEE J. Solid-State Circuits,* vol. 29, no. 12, pp. 1572-1751, Dec. 1994.
23. B. P. Lathi, *Modern Digital and Analog Communication System,* HRW, Philadelphia, 1989.
24. J. S. Bellisio, A new phase-locked loop timing recovery method for digital regenerators, *IEEE Int. Comm. Conf. Rec.,* vol. 1, pp. 10-17-10-20, June 1976.
25. B. Razavi, A 2.5-Gb/s 15-m W clock recovery circuit, *IEEE J. Solid-State Circuits,* vol. 31, no. 4, pp. 472-480, Apr. 1996.
26. R. J. Baumert, P. C. Metz, M. E. Pedersen, R. L. Pritchett, and J. A. Young, A monolithic 50-200MHz CMOS clock recovery and retiming circuit, *IEEE Custom Integrated Circuits Conference,* pp. 14.5.5-14.5.4, 1989.
27. B. Lai and R. C. Walker, A monolithic 622Mb/s clock extraction data retiming circuit, *IEEE Inter. Solid-State Circuits Conference,* pp. 144-145, 1991.
28. B. Kim, D. M. Helman, and P. R. Gray, A 30MHz hybrid analog/digital clock recovery circuit in 2-μm CMOS, *IEEE J. Solid-State Circuits,* vol. 25, no. 6, pp. 1385-1394, Dec. 1990.

Khoury, J.M. "Continuous-Time Filters"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

58

Continuous-Time Filters

John M. Khoury
Lucent Technologies

58.1 Introduction

Modern Very Large Scale Integrated (VLSI) circuits realize complex mixed analog-digital systems on a monolithic semiconductor chip. These systems generally incorporate signal processing operations that can be performed either in the digital domain using digital signal processing (DSP) techniques or in the analog domain with analog signal processing (ASP) circuits. ASP techniques fall into two basic categories: continuous-time or sampled-data. The selection of DSP, continuous-time ASP, or sampled-data ASP approaches is highly dependent on the system requirements; however, continuous-time filters are generally preferable in applications that require low power, high-frequency operation, and moderate dynamic range.

Fully integrated continuous-time filters have found wide application in many VLSI systems that include modems, telephone circuits, disk drive read channels, video processing circuits, and others. The applications usually fall into one of the three basic configurations shown in Fig. 58.1. In the top two views, the continuous-time filter provides anti-aliasing and smoothing functions for sample-data signal processing operations that are either performed with switched-capacitor (SC), switched-current (SI), or DSP filters. Generally, for these applications, the precise signal processing functions are kept in the sampled-data domain. The continuous-time filter can then have non-stringent frequency response specifications provided that the ratio of half the sampling rate to the band edge is large for the sampled-data filter. Often for lower power operation or extremely high frequency operation, the entire signal processing is performed with the continuous-time ASP as shown in the bottom of Fig. 58.1.

When designing a system, the natural question arises as to which is the best approach for the performing the core signal processing operations: DSP or ASP. In general, DSP is usually the obvious choice if the application has the following attributes: (1) frequency response specifications that must be repeatable to within fractions of decibels (dB) over all manufacturing processes, (2) band edges are in the 100-kHz range and below, (3) dynamic range requirements exceed 80 dB, and (4) a high degree of programmability or coefficient adaptation is needed.

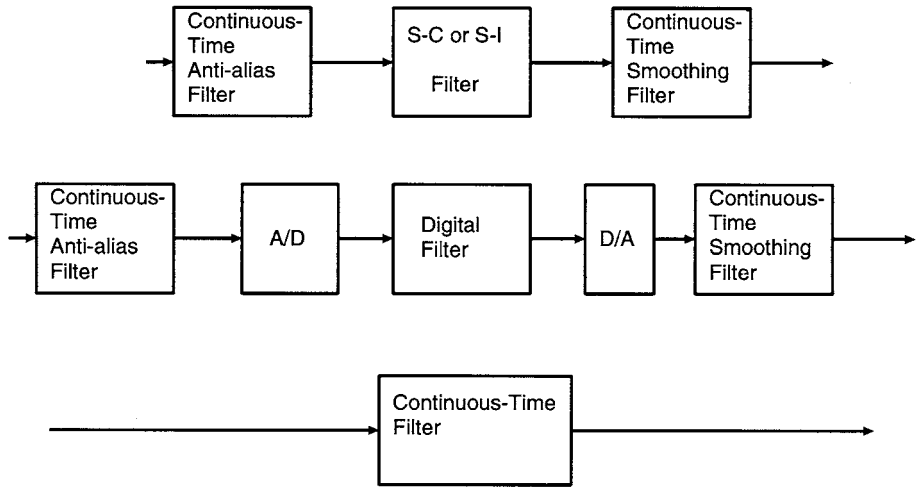


FIGURE 58.1 General uses of continuous-time filter.

ASP typically has a clear advantage when the critical frequencies in the filter exceed several hundred kHz. In today's 0.35-μm CMOS technologies, SC filters are generally limited to sampling rates below 100 MHz; hence, filter passbands will typically be below 25 MHz. SI filters have similar limitations. Continuous-time filters are then the only viable alterntive for passbands in the 10's of MHz and higher.

Continuous-time filters may be designed for the entire frequency range from audio to above 100 MHz. When used in audio and above audio applications, continuous-time filters can achieve the lowest possible power of all the filtering techniques; dynamic ranges above 90 dB can be obtained and linearity above 80 dB is possible with certain continuous-time design approaches.[1-4] Higher frequency continuous-time filters can also be designed to achieve excellent linearity.[5] In the 10- to 150-MHz range, continuous-time filters usually will achieve linearity and dynamic range performance in the 60-dB range and cutoff frequency accuracy in the range of a few percent.[6-11] Integrated continuous-time filters to date have not achieved the performance required in the high-frequency, high-Q, and high-linearity wireless application. The fundamental limitations of thermal noise present in all integrated continuous-time, SC, and SI filters severely limits the achievable dynamic range under high-Q conditions.[12]

Integrated continuous-time filters differ from discrete active filters and SC or SI filters in that a frequency tuning circuit is almost always required to obtain accurate frequency response characteristics.[1,7,9,10,13-15] In VLSI chips, the capacitor values can vary by ±15% for linear capacitors such as double-poly or metal-metal capacitors. Similarly, the resistance element, whether it is a diffused resistor, polysilicon resistor, or transistor will vary widely with processing and temperature. The combined effect often results in RC products that vary by as much as ±50%. In continuous-time filter applications such as anti-aliasing or reconstruction functions, such variation is often acceptable. However, to achieve corner frequency accuracy of a few percent, a tuning circuit is required. In addition to frequency tuning/scaling the filter, circuits to tune the quality factor of the filter are sometimes employed.[6,8,11,16-18]

The following sections in this chapter cover the state-variable implementation of continuous-time filters, the design of VLSI integrators, and the design of highly linear continuous-time filters. The chapter concludes with the design of tuning circuits.

58.2 State-Variable Synthesis Techniques

In the 1960s, considerable research was performed in the area of active filter design. At that time, the focus was on discrete circuit implementations that operated with single-ended circuitry. Although many creative and theoretically appealing approaches were invented and used commercially for discrete designs,

only a few of the circuit topologies are well suited to VLSI implementations. An extensive discussion of active filter realizations can be found in Ref. 19.

Of all the possible active filter topologies possible, the *state-variable filter* is the most general in form and most widely used in VLSI continuous-time filters today. The key advantage of state-variable filters is that they require only two basic building blocks: (1) integrators and (2) weighted summers. In VLSI solutions, the integrators are realized with on-chip capacitors, an active element such as an operational amplifier (op-amp), and a resistive element or transconductance amplifier. Signal summation is performed in the voltage or current domains, depending on the technique used.

The topology of state-variable filters can take on many varied forms. In the most general case, a linear system with N state variables would consist of N integrators with signal coupling between any and all integrators. In practice, coupling between integrators is limited to make the design realizable. In the biquadratic (biquad) filter structure, the N-th order filter is realized as a cascade of second-order circuits, followed by a first-order circuit, if N is odd. The biquad approach is widely used for its simplicity, ease of design, and ease of debugging.[19]

An alternate form of state-variable filters, called the "leapfrog" topology, is realized by simulating the equations that govern RLC ladder filters.[19] In the leapfrog topology, only the state variables (i.e., integrators) that are adjacent to one another are coupled. Leapfrog filters are more difficult to design, but they generally offer improved passband magnitude response accuracy and better dynamic range performance than the cascade of biquads.

Biquadratic Filters

The biquad structure realizes the filtering function as a cascade of second-order filters. The structure decouples the poles of the system and can ease the overall design approach.

The general equation governing the biquadratic filter is

$$V_{out}(s) = K \frac{s^2 + \frac{\omega_{oz}}{Q_z} s + \omega_{oz}^2}{s^2 + \frac{\omega_{op}}{Q_p} s + \omega_{op}^2} V_{in}(s) \tag{58.1}$$

where $V_{out}(s)$ and $V_{in}(s)$ are the output and input signals of the biquad, respectively, K is a gain constant, ω_p and ω_z are the frequencies of the poles and zeros, and Q_p and Q_z are the quality factors of the poles and zeroes, respectively. Although many methods of realizing this transfer function are possible, the state-variable approach uses a loop of two integrators connected with negative feedback to realize the poles. Damping around one (or both) integrator(s) makes the corresponding integrator lossy and implements the pole quality factor, Q_p. The zeroes of the biquad can be achieved by (1) creating an output signal $V_{out}(s)$ that is the weighted summation of the two integrator outputs, as well as the input signal, $V_{in}(s)$, or (2) by summing scaled values of the input signal into both integrators as well as directly to the output. The block diagram of the generalized biquad, shown in Fig. 58.2, places the zeroes and adjusts the overall gain of the filter with the K_1, K_2, and K_3 constants. The block diagram of Fig. 58.2 can be easily converted to an integrated VLSI filtering technique with a one-for-one substitution of the integrators and weighted summers with the corresponding VLSI circuits. Integrated implementations will be discussed in the sections on G_m-*C*, G_m-*OTA-C*, and *MOSFET-C* filters.

If the high-order transfer function is of the form:

$$H(s) = \frac{N(s)}{D(s)} \tag{58.2}$$

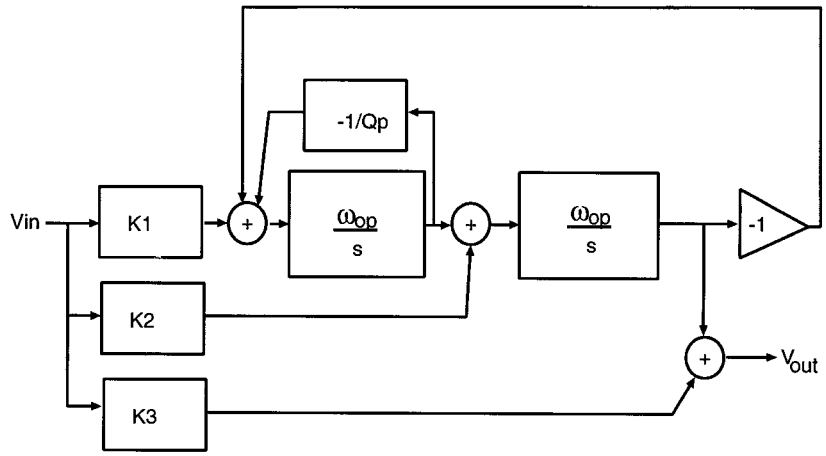


FIGURE 58.2 Biquad block diagram.

then $H(s)$ can be factored into second-order sections where the numerators and denominators of these biquads are at most second order, as in Eq. 58.1. Issues of how to arrange the cascade of second-order functions and how to pair the poles and zeroes can greatly affect the filter's performance in terms of signal swing, dynamic range, and dc offset accumulation. A few simple rules of how to realize a cascaded filter are enumerated here.

1. **Factor into Biquadratic Terms:** Split the numerator, $N(s)$, and the denominator, $D(s)$, into products of second-order functions. If either $N(s)$, or $D(s)$ is odd-order, a first-order term will be necessary. The transfer function is then in the following form:

$$H(s) = \frac{N_1(s)N_2(s)N_3(s)\ldots}{D_1(s)D_2(s)D_3(s)\ldots} \tag{58.3}$$

2. **Pair Poles and Zeroes:** Convert Eq. 58.3 into a product of second-order transfer functions, $H_A(s)$ $H_B(s)$ $H_C(s)$ …, by pairing each $N_i(s)$ with a $D_j(s)$ in such a way that $|H_A(j\omega)|$, $|H_B(j\omega)|$, $|H_C(j\omega)|$, etc. has as flat a magnitude response over the passband as possible. In this way, the signal at the various points in the cascade of the filter will be large and hence less susceptible to interference. Interference could be due to the thermal noise of the active and passive circuits, power supply noise, and crosstalk from digital circuits on-chip.

   To make $|H_A(j\omega)|$, $|H_B(j\omega)|$, $|H_C(j\omega)|$, etc. as flat as possible over the passband, pair the zeroes of $N_i(s)$ as close in frequency and Q as the poles of $D_j(s)$. This method minimizes the variation caused by $|N_i(j\omega)|/|D_j(j\omega)|$ because the effects of the pole and zero pairs tend to partially cancel.
3. **Choose Cascade Order:** The next decision is to order the biquads (and maybe a first-order term). Many practical factors influence the optimum ordering. A few examples are:
   a. Order the cascade to equalize signal swing as much as possible throughout the filter to maximize dynamic range.
   b. Choose the first biquad to be lowpass or bandpass to reject high-frequency noise, eliminating overload in the remaining stages.
   c. If the offset at the filter output is critical, the last stage should be a highpass or bandpass to block the dc of previous stages.
   d. Avoid high-Q biquads at the last stage because these biquads have higher fundamental noise and worse sensitivity to power supply noise than low-Q stages.[12]

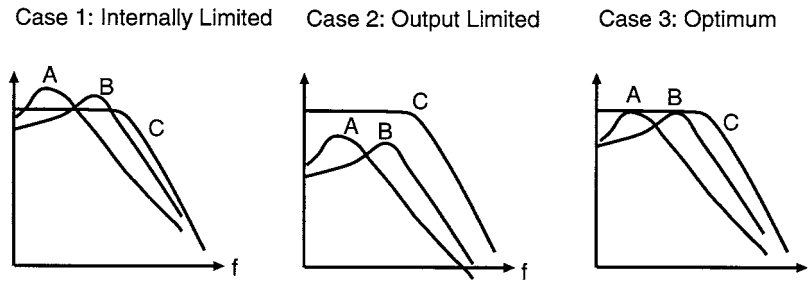


FIGURE 58.3 Dynamic range scaling.

e. In general, do not place allpass stages at the end of the cascade because these have wideband noise. It is usually best to place allpass stages near the beginning of the filter.
f. If several highpass or bandpass stages are available, one can place them at the beginning, middle, and end of the filter. This will prevent input dc offset from overloading the filter, will prevent internal offsets of the filter itself from accumulating (and hence decreasing available signal swing), and will provide a filter output that has low dc offset.
g. The effect of thermal noise at the filter output varies with ordering; therefore, several decibels (dB) of SNR can often be gained with biquad reordering.

4. **Dynamic Range Optimization:** Dynamic range optimization is simply the scaling of gains within the filter to make sure that the overload levels of the integrators (or summers) are equalized so that all elements will saturate at the same signal level.

If the frequency spectrum of the input signal is known, then dynamic range scaling of the filter's should be performed with this signal. The maximum amplitude input signal should be provided and the gains scaled until all integrator and summer outputs are at their maximum level. Note that gain scaling should be performed so as not to modify any loop gains in the filter; otherwise, the transfer function would be altered.

If the frequency spectrum of the input signal is unknown, the typical approach is to assume the input signal is a single sinusoid. The filter is then dynamic range scaled so that for the maximum amplitude input sinusoid, all integrator and summer outputs have the same maximum value for any possible sinusoidal frequency. Usually, the frequency of the input sinusoid is swept over the filter's passband and the maximum levels are then gain scaled. Pictorially this can be seen in Fig. 58.3. Here, the filter consists of a cascade of three biquads: A, B, C. The frequency response to each biquad output is shown. In case 1, the signal will clip at the output of biquads A and B first; whereas in case 2, the output, C, will saturate first. It is only in case 3 — where the maximum gains have been equalized — that clipping occurs in all three biquads at the same level.

Dynamic range scaling must be performed not only at the output of each biquad, but also at the output of the internal integrator. As an example, consider the classical Tow-Thomas biquad in Fig. 58.4. The derivation of this biquad from the block diagram in Fig. 58.2 should be self-evident. The frequency response shows that the internal node V_x will clip at a lower input amplitude level than the output. The signal amplitude at node V_x must be reduced by a factor F. The reduction in gain can be achieved by lowering the impedance in the feedback loop of the first integrator by F. However, to maintain constant loop gain around the two integrator loop, the input resistor of the second integrator must become R/F. The result of this dynamic range scaling is shown in Fig. 58.5.

Leapfrog Filters

The leapfrog filter topology uses active integrators and weighted summers to simulate all the equations governing RLC ladder filters.[19] The question naturally arises as to why passive *ladder* filters should be chosen. First, a wealth of knowledge and design tables exist for these filters. Designers can easily use

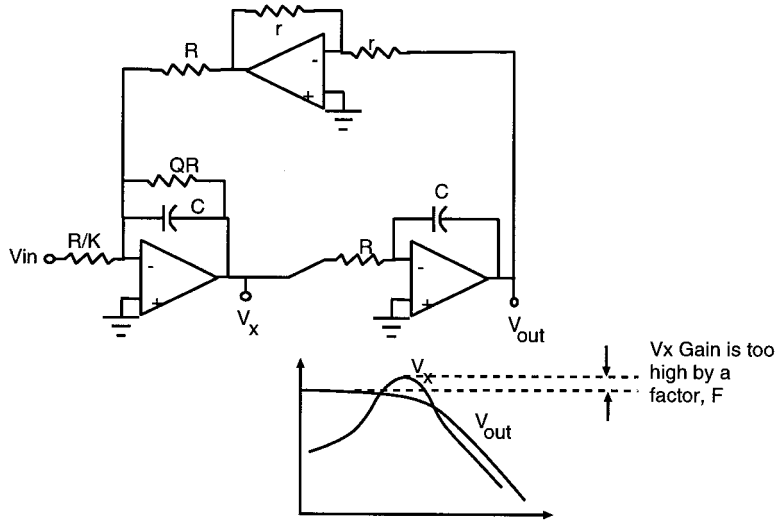


FIGURE 58.4 Tow-Thomas biquad prior to dynamic range scaling.

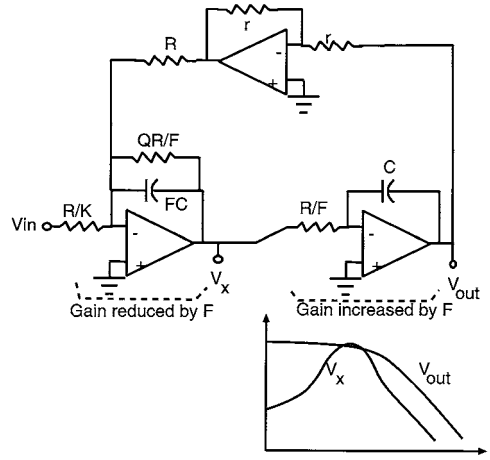


FIGURE 58.5 Tow-Thomas biquad after dynamic range scaling.

tabulated data to design classical ladder filters that implement Butterworth, Chebychev, Bessel, etc. responses. With a few simple steps, these ladders can be transformed into an active leapfrog topology with element values.[19] The second and more important reason to simulate ladder filters is that *in the passband*, the sensitivity of the filter's magnitude response to element value variation is extremely low. This low sensitivity is not true in the stopband, nor is it true for the phase response of the filter. Since leapfrog filters simulate all the equations governing the ladder filter, these sensitivity advantages carry over to the active realization. Finally, filters that are relatively insensitive to component errors *usually* have lower thermal noise. In most applications, leapfrog filters will have superior performance relative to biquadratic filters in terms of noise and passband magnitude response accuracy. Snelgrove and Sedra[20] analyzed biquad filters, leapfrog filters, and filters optimized for noise and magnitude response sensitivity. The leapfrog filters achieved performance close to the optimized design, but the biquad approach showed significantly degraded performance.

The design of leapfrog filters can be found in Ref. 19. Here, we will show by example the design of these filters. Consider the third-order lowpass doubly terminated ladder shown in Fig. 58.6. Since the

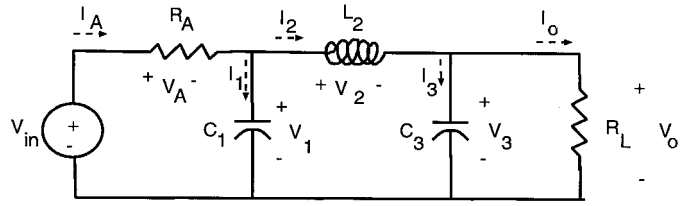


FIGURE 58.6 Third-order doubly terminated lowpass LC ladder filter.

active filter must simulate all equations governing the ladder, the first step is to write all the equations. The two-terminal branch relationships are:

$$I_A = \frac{V_A}{R_A}, \quad V_1 = \frac{I_1}{sC_1}, \quad I_2 = \frac{V_2}{sL_2}, \quad V_3 = \frac{I_3}{sC_3}, \quad I_O = \frac{V_O}{R_L} \tag{58.4}$$

The KVL equations are:

$$V_A = V_{in} - V_1, \quad V_2 = V_1 - V_3, \quad V_O = V_3 \tag{58.5}$$

The KCL equations are:

$$I_1 = I_A - I_2, \quad I_3 = I_2 - I_O \tag{58.6}$$

As can be seen in Eqs. 58.4 to 58.6, some variables are currents while others are voltages. In the implementations, usually all signal variables are either voltages or currents. Here, voltage signals will be assumed. To convert the current signals in Eqs. 58.4 to 58.6 to voltages, all currents can be scaled by a resistance r of arbitrary value (e.g., 1 Ω). After the scaling by r, Eqs. 58.4 to 58.6 become:

$$\begin{gathered} rI_A = \frac{rV_A}{R_A}, \quad V_1 = \frac{rI_1}{srC_1}, \quad rI_2 = \frac{V_2}{sL_2/r}, \quad V_3 = \frac{rI_3}{srC_3}, \quad rI_O = \frac{rV_o}{R_L}, \\ V_A = V_{in} - V_1, \quad V_2 = V_1 - V_3, \quad V_O = V_3, \quad rI_1 = rI_A - rI_2, \quad rI_3 = rI_2 - rI_O \end{gathered} \tag{58.7}$$

Using Eq. 58.7, the signal flow graph (SFG) shown in Fig. 58.7 can be obtained. Arrows flowing into a circle represent summation, and the values next to the arrows indicate a scaling operation. In the SFG, the KVL equations are implemented with the top two summation circles, while the KCL equations are

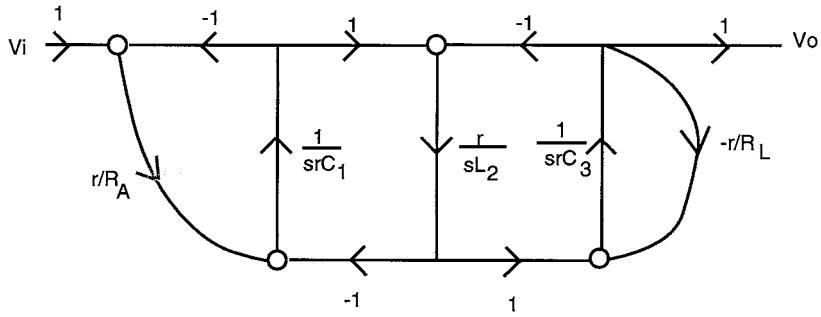


FIGURE 58.7 Signal flow graph representation of LC ladder filter.

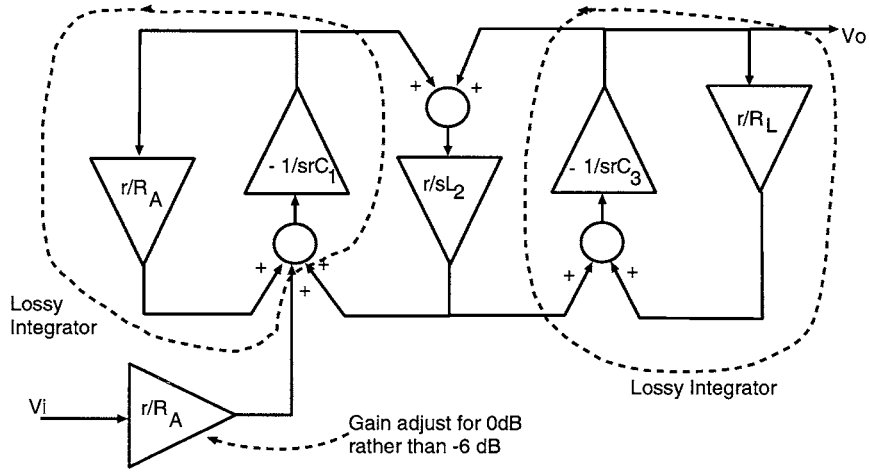


FIGURE 58.8 Leapfrog filter with gain scaling extracted.

on the bottom side. If one were to implement this SFG directly, the gain of the filter would be less than 0 dB (e.g., –6 dB for an equally terminated ladder). In fact, the gain would be the same as the original RLC ladder. By replicating the gain block, r/R_A, as shown in Fig. 58.8 an additional degree of freedom is obtained to implement arbitrary filter gains. Dotted lines are used to indicate that the integrators on the end of the filter are damped, while the inner integrator is lossless. In the realization of high-order ladder filters, the internal integrators will always be lossless, while the outside ones will be lossy due to the ladder terminations.

Highpass and bandpass leapfrog filters can be realized directly from the lowpass LC ladder with the use of the classical lowpass-to-highpass or lowpass-to-bandpass transformations.[19] For illustrative purposes, the bandpass case is considered here. Starting from a lowpass prototype with frequency domain variable *s,* a bandpass filter with bandwidth *BW* and center frequency ω_o can be realized in the frequency domain variable *p* with the following transformation:

$$s = \frac{p^2 + \omega_o^2}{pBW} \tag{58.8}$$

Applying the transformation element by element to a third-order lowpass ladder, one obtains the bandpass ladder shown in Fig. 58.9. An SFG can be generated directly from the bandpass ladder and the active filter realized. Alternatively, the lowpass-to-bandpass transformation can be applied directly to the lowpass active filter of Fig. 58.8, resulting in the bandpass active filter in Fig. 58.10. Notice that each integrator has been replaced with a bandpass biquad and that the biquads corresponding to the terminations are damped.

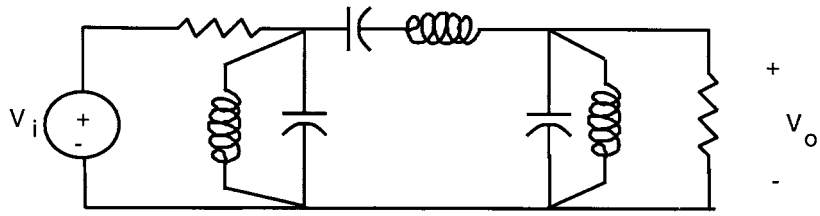


FIGURE 58.9 Bandpass ladder filter.

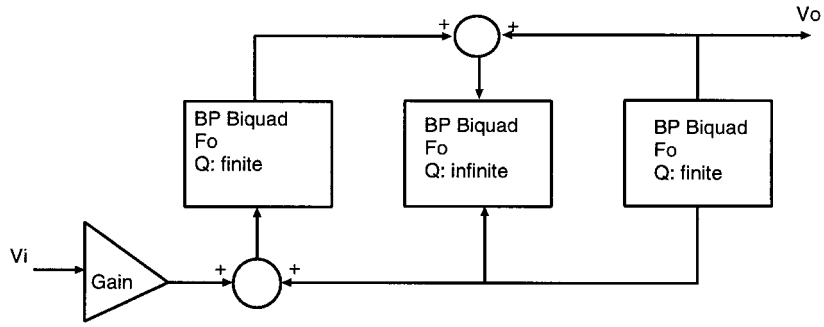


FIGURE 58.10 Bandpass leapfrog filter realization.

Designers proficient in the use of SFGs can readily transform the active leapfrog realization to include zeroes in the transfer function.

58.3 Realization of VLSI Integrators

Once the state-variable topology has been created, the VLSI filter realization is determined by the approach used for the integrator. This section describes the most common types of VLSI integrators and their corresponding summing circuits. The three most common types of implementations are the G_m-*C*, G_m-*OTA-C* and *MOSFET-C* filters. G_m-*C* filters are generally recognized to offer the highest possible frequency operation at the lowest power; however, the structures are sensitive to parasitic capacitances and generally have higher noise and offset than other techniques. G_m-*OTA-C* filters are far less parasitic sensitive than G_m-*C* designs, but at the cost of higher power. Finally, *MOSFET-C* filters generally are the most parasitic insensitive, and have the least noise and offset; however, the frequency of operation is *usually* the lowest of the three approaches. In BiCMOS technology where extremely wideband op-amps can be made, *MOSFET-C* techniques possess bandwidth capabilities approaching that of G_m-*C* and G_m-*OTA-C* filters.

G_m-*C* Integrators and Filters

G_m-*C* filters implement integrators with a transconductance amplifier loaded by a capacitor. As shown in Fig. 58.11 a differential transconductance amplifier (also called a transconductor) takes an input voltage, V_{ind}, and produces at its output a current $I_{out} = G_m V_{ind}$. This output current is integrated by the capacitor to produce the output voltage signal, V_{out}. The transfer function of the G_m-*C* integrator is

$$H(s) = \frac{G_m}{sC} = \frac{\omega_o}{s} \tag{58.9}$$

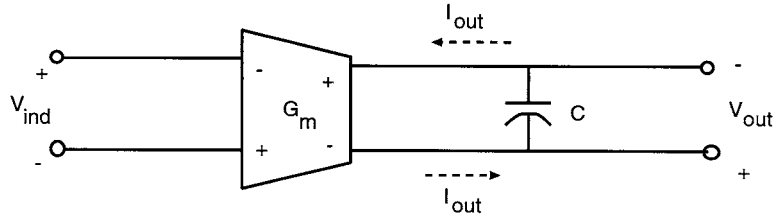


FIGURE 58.11 G_m-*C* integrator.

where ω_o is the unity-gain frequency of the integrator. The ideal integrator has infinite dc gain, a unity-gain frequency of ω_o, and a phase shift of –90° for all frequencies. Capacitors in VLSI technology are usually high quality, so all stringent integrator requirements fall on the transconductor design. Since the transconductor is a voltage-to-current ($V \rightarrow I$) converter, it should have: (1) high input impedance to accurately sense the input voltage signal, (2) high output impedance so the output signal appears as a current source, (3) high dc gain, (4) wide bandwidth so as not to create phase and magnitude errors in the integrator response, (5) large signal handling capability at the input and output for good dynamic range, and (6) a well-defined and tunable $V \rightarrow I$ mechanism to be used for frequency scaling the filter to remove process and temperature variations. In CMOS or BiCMOS technology, achieving high input impedance is simple due to the gate terminal of the MOSFET. Designing for high output impedance can be achieved with cascoding and with the use of regulated cascodes[21]; however, there is a tradeoff between high bandwidth and high output impedance. The most difficult aspect of G_m cell design is making the $V \rightarrow I$ mechanism tunable simultaneously achieving good linearity in the presence of large input signal swings.

Building state-variable filters with G_m-C filters follows directly from the block diagrams or SFGs. Signal summation is performed in the current domain by placing transconductor outputs in parallel. Consider the SFG in Fig. 58.12. The bandpass G_m-C filter is readily implemented as in Fig. 58.13. The loop of two

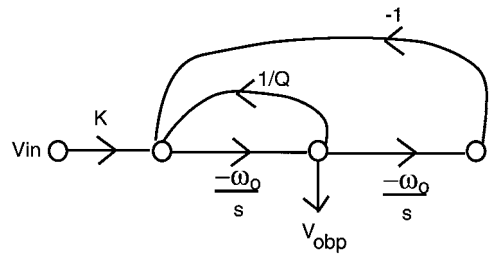


FIGURE 58.12 Signal flow graph representation of a state-variable biquad.

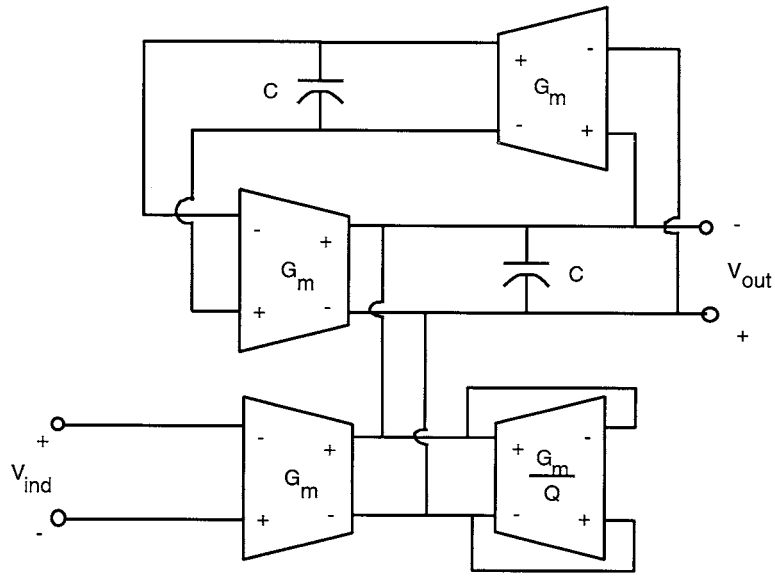


FIGURE 58.13 G_m-C realization of a bandpass biquad.

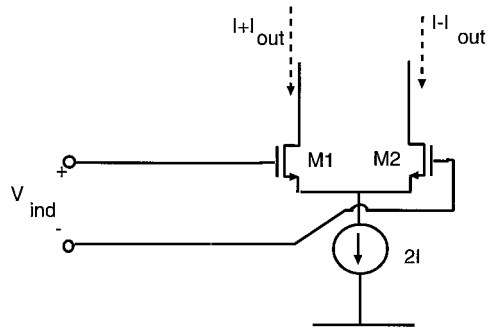


FIGURE 58.14 An MOS differential pair used as a $V \rightarrow I$ converter.

integrators is on the top of the figure, biquad damping is performed with the G_m/Q transconductor, and input signal scaling and summation are achieved with the remaining G_m cell. The G_m/Q transconductor connected in the negative feedback configuration on itself implements a resistor of value Q/G_m.

The key aspect in the design of G_m-C filters is the transconductor design and more specifically, the $V \rightarrow I$ converter. In the simplest case, the $V \rightarrow I$ converter can be a simple MOS differential pair, as shown in Fig. 58.14. The large signal differential output current, I_{outd}, is given by[22]:

$$I_{outd} = 2I_{out} = \mu_n \frac{C_{ox} W}{2L} V_{ind} \sqrt{\left(\frac{4I}{\mu_n C_{ox} W/(2L)} \right) - \left(V_{ind} \right)^2} \tag{58.10}$$

I_{outd} is a non-linear function of the input V_{ind}. The transconductance of the differential pair, $G_m = dI_{outd}/dV_{ind}$, is maximum at $V_{ind} = 0$ and falls off for increased signal swing. The maximum G_m is:

$$G_m = \sqrt{2I\mu_n C_{ox} W/L} = g_{m1} = g_{m2} \tag{58.11}$$

The transconductance can be tuned with the tail current $2I$ to frequency scale the G_m-C filter; however, the tuning range is small since G_m only varies as the square root of the current. In general, if the targeted filter application requires no programmability and the critical frequencies are nominally fixed, then roughly a 2:1 tuning range is needed to accommodate process and temperature variations. The tail current would then require a 4:1 variation, greatly impacting power dissipation. The more significant disadvantage of this $V \rightarrow I$ converter is its small linear input range. It can be shown that the linear differential input voltage range is much smaller than $\pm\sqrt{2}(V_{gs1,2-bias} - V_T)$, where $V_{gs1,2-bias}$ is the bias level of M1 and M2 for $V_{ind} = 0$. To maximize the linear input range $V_{gs1,2-bias}$ must be kept large by using small W/L ratios. Even with use of small W/L ratios, the input range is typically limited to less than ±200 mV for linearity of 40 to 60 dB.

Many linearization techniques have been invented using MOSFETs in the saturation and triode regions, as well as BiCMOS solutions. A few approaches are discussed here. In the basic MOS differential pair, the output current, I_{outd}, increases with V_{ind}; however, the rate of increase drops off at higher input amplitude levels. One solution is to have another source of current that is added to the transconductor's output. If that current is zero for low levels of V_{ind}, but increases with V_{ind}, the net effect is to linearize the overall G_m cell. Rather than adding a current, we can instead subtract a current from the G_m stages output. The amount to be subtracted would be maximum for $V_{ind} = 0$ and would drop to zero for large differential input signals. This technique uses an additional differential pair, as shown in Fig. 58.15.[10] Transistors M3 and M4 operate at lower current than M1 and M2 and are biased such that $(V_{gs3,4-bias} - V_T) \ll (V_{gs3,4-bias} - V_T)$. Detailed design equations for the sizing of M1-M4, I1, and I2 can be found in

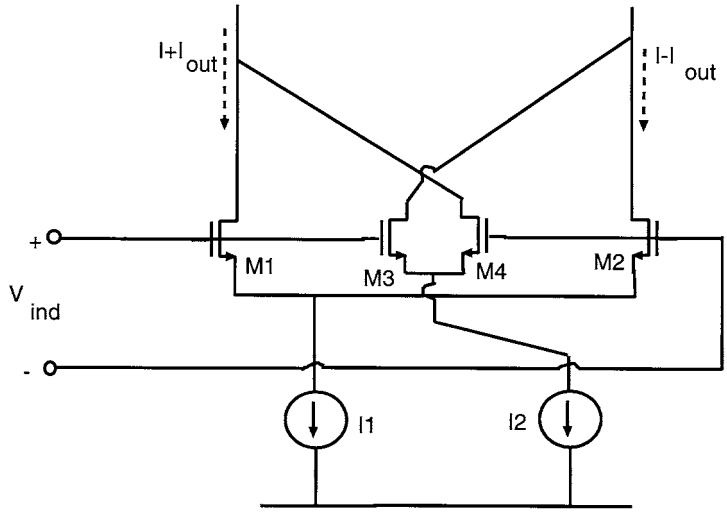


FIGURE 58.15 A cross-coupled linearized MOS $V \rightarrow I$ converter.

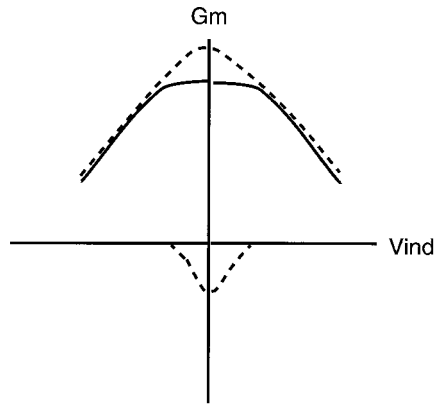


FIGURE 58.16 Individual differential pair G_m (dotted curves) and combined G_m (solid curve).

Ref. 10. The current through M3 and M4 will saturate at lower input voltages than the drain currents of M1 and M2. The concept is more clearly understood with Fig. 58.16. The dotted lines show the transconductance of the individual differential pairs versus input differential signal. The dotted lines for positive G_m correspond to the transistors M1 and M2, while the negative G_m refers to M3 and M4. Adding the G_m curves, the solid curve is obtained. Notice now that the transconductance curve is flat for small V_{ind}. It is possible to add the outputs of multiple differential pairs with slightly different non-linearities to broaden the region over which G_m is constant. A related approach is given in Ref. 23.

A classical linearization method is to use a differential pair with source degeneration. However, in most technologies, the resistor used for degeneration would vary with temperature and processing and be nominally fixed in value. To afford tunability, the degeneration resistor can be replaced with a MOSFET, M3, operating in the triode region, as shown in the G_m cell of Fig. 58.17.[24] M1 and M2 act as source followers and the transconductor's signal currect, I_{out}, is ideally the drain current of M3. The drain current of M3 can be expanded in a Taylor series for the case of zero drain-to-source voltage and obtains the following from the "3/2 Power" model[25] of the MOSFET:

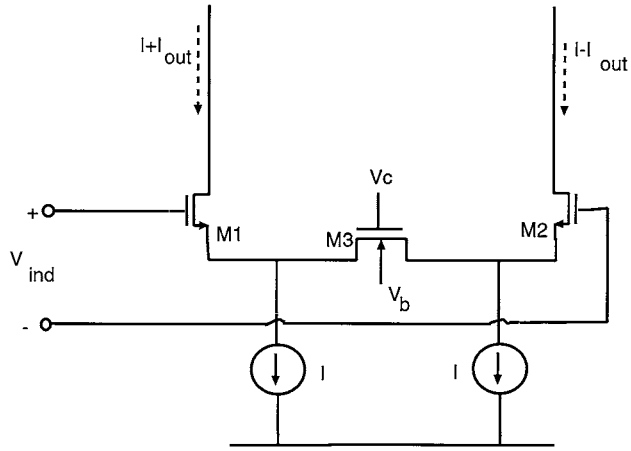


FIGURE 58.17 Source degenerated MOS $V \rightarrow I$ converter.

$$I = \left(W/L\right)\mu C_{ox}\left[\left(V_C - V_Q - V_T\right)\left(V_D - V_S\right) + \sum_{i=2}^{\infty} a_i\left(V_D^i - V_S^i\right)\right] \tag{58.12}$$

where V_Q is the bias level of the source and drain with respect to the body, V_S and V_D are the voltages on the source and drain terminals, respectively, and the a_i are constants. Notice that if the source and drain voltages are balanced around a common-mode voltage V_Q, such that $V_D = V_Q + V_{ind}/2$ and $V_S = V_Q - V_{ind}/2$, then as can be seen from Eq. 58.12,

$$I = \left(W/L\right)\mu C_{ox}\left[\left(V_C - V_Q - V_T\right)V_{ind} + \sum_{i>1,odd}^{\infty} a_i V_{ind}^i\right] \approx \left(W/L\right)\mu C_{ox}\left(V_C - V_Q - V_T\right)V_{ind} \tag{58.13}$$

For many applications, the remaining odd-order non-linearity is low enough (e.g., –65 dB) to be inconsequential. For applications requiring superior linearity, cross-coupled triode degenerated differential pairs can be used to theoretically cancel all high-order non-linearities.[24,26] Based on Eq. 58.13 the transconductance of the G_m cell is:

$$G_m = \left(W/L\right)\mu C_{ox}\left(V_C - V_Q - V_T\right) \tag{58.14}$$

assuming M1 and M2 are ideal source followers. As desired, the G_m is tunable with the control voltage, V_C, connected to the gate of M3. The linear input range can in practice be on the order of ± 1.0 *V*, provided that M3 remains in the triode region. The maximum input signal is thus equal to

$$V_{ind\text{–max}} = 2\left(V_C - V_Q - V_T\right) \tag{58.15}$$

The maximum input signal swing is a function of the tuning control voltage, V_C; consequently, the dynamic range is tightly coupled to the tuning range. In some situations where a programmable filter is required, multiple transistors can replace M3 and the triode devices can either be connected to V_C, or

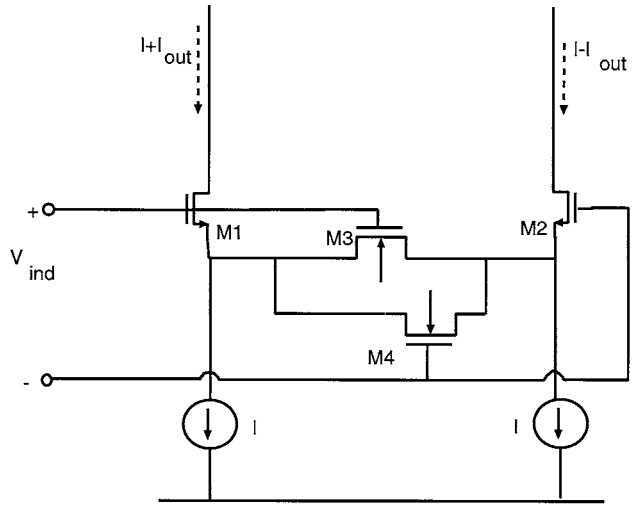


FIGURE 58.18 G_m cell operating in triode and saturation.

switched off to implement ranges in the filter.[7] In contrast to the transconductors in Figs. 58.14 and 58.15, power dissipation is unaffected by tuning. Finally, source followers M1 and M2 must be extremely low impedance (i.e., have high g_m) to drive the resistance of M3. Either large *W/L* devices can be used for M1 and M2, or negative feedback can be used around M1 and M2 to reduce their impedance at their source terminals.[5,27]

As an alternative to requiring low impedance source followers, the G_m cell shown in Fig. 58.18 can be used.[9] For small input signals, M3 and M4 operate in the triode region as in the previous design; however, the effective control voltage to tune the devices is set to the gate-to-source bias level of M1 and M2. For large positive input differential signals, more current flows in M1, increasing the V_{GS} of M1. If M3 and M4 had fixed gate voltages, the current through them would drop off, resulting in lower G_m as in the design of Fig. 58.17. However, the gate voltage of M3 increases under this input condition and helps to maintain a constant G_m. By scaling M1 and M2 to M3 and M4 (e.g., to a ratio of about 7:1) the linear range can be expanded. The linear range is also larger than that of Fig. 58.17 because M3 and M4 can operate in both the triode and saturation regions.

All transconductor designs discussed to this point have achieved an expanded linear range as a function of device matching or use of balanced input differential signals. Since matching and signal balancing can never be perfect, the achievable linearity is a strong function of layout and processing. In contrast, the transconductor of Fig. 58.19 achieves high linearity by maintaining constant drain-to-source voltage across triode devices M1 and M2.[8,28,29] Using the basic triode equation for a MOSFET,

$$I = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}\left[2\left(V_{gs} - V_t\right)V_{DS} - V_{DS}^2\right] \tag{58.16}$$

one can see that if the drain-to-source voltage is held constant, the relationship between V_{GS} and I is linear, except for an offset. Cascode devices Q1 and Q2 in Fig. 58.19 are used to hold $V_{DS1} = V_{DS2} = I_d R_d$, resulting in a linear transconductance of

$$g_m = \mu_n C_{ox}\frac{W}{L}R_d I_d \tag{58.17}$$

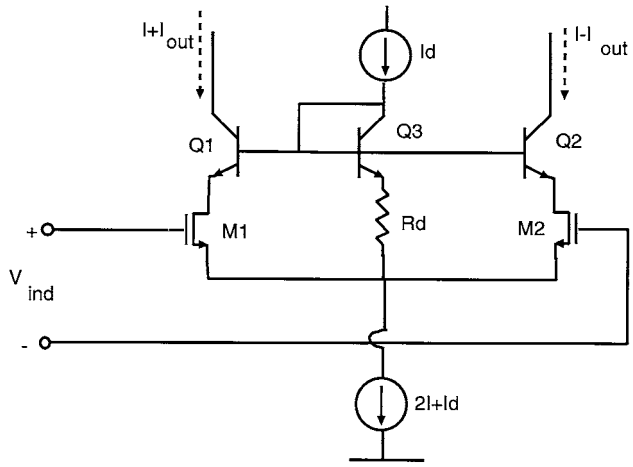


FIGURE 58.19 $V \rightarrow I$ converter with differential pair operating in triode with constant drain-to-source voltage.

The G_m cell is easily tuned with the collector current of Q3, I_d. Q1-Q3 could be replaced with MOSFETs; however, since the transconductance of bipolar devices is higher than MOSPETs, the BiCMOS solution provides superior cascoding to hold the drain-to-source voltages of M1 and M2 constant.

Many alternative linearization schemes exist for the $V \rightarrow I$ converters in CMOS, BiCMOS, and bipolar technology.[5,17,23,30-32] Invariably, nearly all the techniques require matching of transistors and/or balanced signals to achieve optimal linearity performance. Also, many of the techniques, particularly the MOSFET-based transconductors, rely on simplified large signal models (e.g., square law) of the transistors to model and cancel the non-linearity. In reality, more complex transistor equations, as found in Ref. 33, are needed to better predict performance. Ultimately, only experimental results over many process lots must be used for guaranteeing a specified linearity.

Once the $V \rightarrow I$ converter design has been determined, the entire G_m amplifier or integrator can be assembled using known op-amp structures. The design in Fig. 58.20 uses the $V \rightarrow I$ converter of Fig. 58.17 with a folded-cascode output stage. The cascoding raises the output impedance of the amplifier and

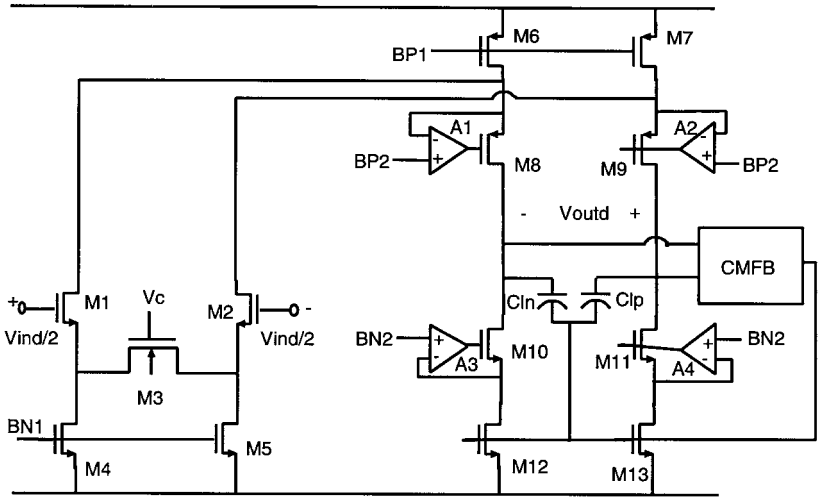


FIGURE 58.20 Folded cascode MOS G_m-C integrator.

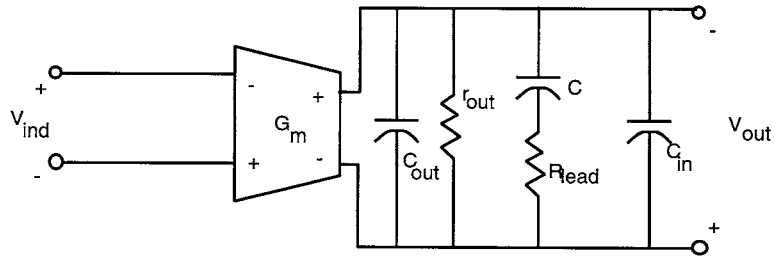


FIGURE 58.21 Non-idealities modeled in the G_m-C integrator.

optionally A1 through A4 create regulated cascodes to raise the output impedance further.[21] The nominally equal capacitors C_{lp} and C_{ln} serve two functions. For differential-mode output signals, the capacitors integrate the output current. For common-mode signals, they provide high frequency common-mode feedback (CMFB) via the gates of M12 and M13, while low frequency CMFB is poerformed with standard techniques.[34] Folding the cascode structure permits larger output voltage swings and equal input and output common-mode voltage levels. Use of folding as opposed to an unfolded cascode (i.e., telescopic structure) will generally result in increased input-referred noise and offset due to the addition of transistors M6 and M7.

G_m-C Integrator Frequency Response Errors

The G_m-C integrator will have magnitude and phase errors due to parasitic capacitances and resistances. Consider the non-ideal integrator shown in Fig. 58.21. C is the integrating capacitance, c_{in} is the parasitic capacitance due to wire routing and the input of the next stage, and c_{out} and r_{out} are the parasitic output capacitance and resistance of the transconductor, respectively. R_{lead} is a series resistance that is sometimes added to provide phase lead for correction of parasitic effects. Assuming that the G_m amplifier has a dc level of G_{mo}, a parasitic pole at ω_p and a parasitic zero at ω_z, the transfer function of the non-ideal integrator can be derived as:

$$H_a(s) = \frac{G_{mo}\left(1 + s/\omega_z\right)}{\left(1 + s/\omega_p\right)} \frac{r_{out}}{1 + sr_{out}\left(C + c_{out} + c_{in}\right)} \tag{58.18}$$

The ideal integrator has infinite dc gain, a rolloff of 6 dB/octave, a unity-gain frequency of $\omega_o = G_{mo}/C$ and a phase of 90° for all frequencies. Parasitics generally have a much stronger effect on the integrator's phase response than the magnitude response. The integrator phase errors in turn are the largest source of filter magnitude response errors. In general, the integrator phase accuracy at ω_o is most critical. The G_m-C integrator's phase error as a function of frequency is:

$$\phi_{I-error}(\omega) \approx \pi/2 + \left(\omega/\omega_z\right) - \left(\omega/\omega_p\right) - \arctan\left[G_{mo}r_{out}\left(\omega/\omega_{ox}\right)\right] \tag{58.19}$$

where ω_{ox} is the actual as opposed to the ideal unity-gain frequency. As an example, consider a G_m-C integrator with a unity-gain frequency of 20 MHz, C = 1 pF, $G_{m\sigma}$ = 125.7 μS, and r_{out} = 1 MΩ. If the transconductor has a parasitic pole at 300 MHz, but no parasitic zero, the resulting phase error at 20 MHz is –3.4°. Depending on the application, such an error may be acceptable. Two methods exist for correcting the phase error. The simplest approach is to add a zero to the transfer function to create phase lead of +3.4° at 20 MHz, resulting in zero net phase error at ω_o. The small value resistor, R_{lead}, in Fig. 58.21 is used for this purpose. Phase lead can also be created within the G_m amplifier with known feedforward techniques. If the accuracy of the phase is critical, tunable phase lead or lag can be performed.[6,8,11,16]

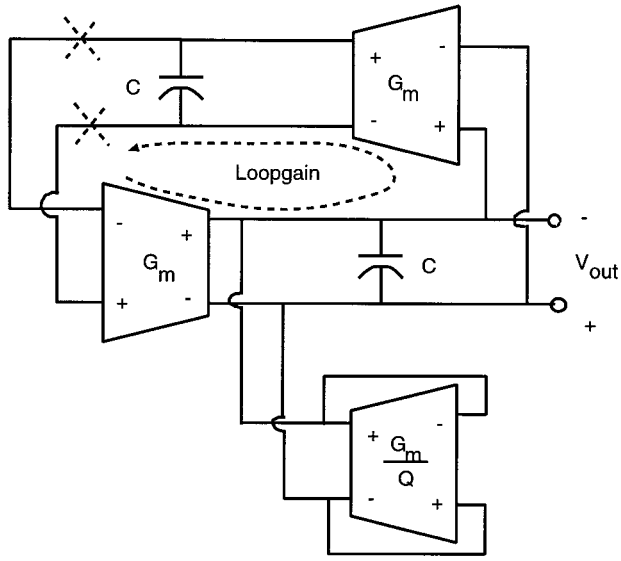


FIGURE 58.22 Integrator phase error impact on biquad filter Q.

The natural question arises as to what integrator phase accuracy is required. Although the requirement is dependent on the filter topology, and transfer function, a good estimate can be determined by considering the damped loop of two integrators shown in Fig. 58.22. This loop of integrators is found in cascade of biquad filters and leapfrog filters and is therefore quite general. The system poles are determined by the loop gain transfer function, $T(s)$,

$$T(s)=\left[\frac{G_m}{sC}\right]\left[\frac{G_m}{sC+G_m/Q}\right]=\left[\frac{1}{s/\omega_o}\right]\left[\frac{1}{s/\omega_o+1/Q}\right] \tag{58.20}$$

Under Ideal conditions and assuming a high value of Q, the loop gain phase shift is

$$\phi(\omega)\approx\frac{\omega_o}{\omega Q} \tag{58.21}$$

Notice that as Q increases, the net phase shift around the loop approaches zero; thus making any integrator phase errors a large source of damping errors. Consider two examples. First, the continuous-time filter used in hard disk drive read channels often has a Bessel response where all the poles are low Q. Assuming $Q \approx 2$, the nominal phase shift around the integrator loop, $\phi(\omega_o) \approx 26.6°$, is quite large. If the total integrator phase error is to be kept $<0.1\phi$, then each integrator phase error must be $<1°$. In contrast, for a bandpass response that might be found in wireless systems with $Q = 100$, the nominal loop phase is $\phi(\omega_o) \approx 0.57°$, and each integrator phase error must be kept $<0.03°$. The low-Q integrator requirements can be met with fixed or tunable phase lead networks, whereas the high-Q applications will require phase tuning (Q tuning) circuits. Although this discussion has centered around G_m-C filters, the results are equally applicable to G_m-*OTA*-C and *MOSFET*-C filters.

G_m-*OTA*-C Filters

The G_m-C integrator of Fig. 58.11 achieves high frequency operation because the circuit configuration is open loop; however, the requirement for high impedance at the output nodes of the G_m amplifier is

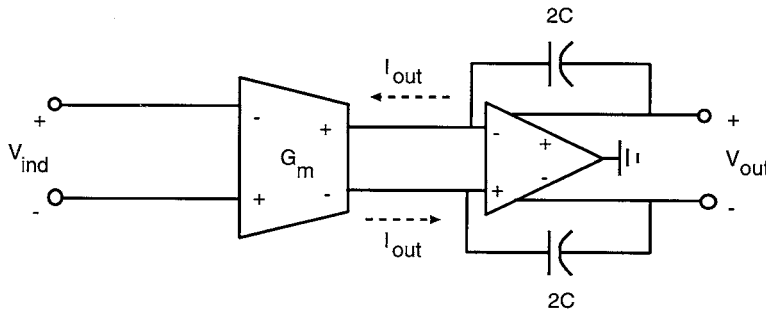


FIGURE 58.23 Fully balanced G_m-*OTA*-*C* integrator.

generally difficult to meet. Use of cascodes in the output stage is required, but the signal swing and resulting linearity often suffer. The second difficulty is that any parasitic capacitance due to routing or the input impedance of the next stage will lower the unity-gain frequency of the integrator. A portion of the parasitic capacitance is the non-linear diode capacitance of the drain diffusions. As the signal frequency increases, more current flows into these non-linear capacitors, reducing the overall linearity of the integrator.

The G_m-*OTA*-*C* integrator of Fig. 58.23 adds an operational transconductance amplifier (*OTA*) connected with negative feedback via the integration capacitors at the output of the G_m amplifier. For this configuration, the G_m amplifier drives into a virtual short-circuit and hence requires no signal swing. Parasitic capacitances at the output of the G_m amplifier are held at virtual ground and the capacitances at the output of the *OTA* are driven by a low impedance, so in both cases, their impact on the integrator's frequency response and linearity is negligible. Although the parasitic capacitances do not shift the ω_o of the integrator, high-frequency parasitic poles are created that can cause phase errors near ω_o. For this reason, the G_m-*OTA*-*C* approach operates at a lower frequency than the G_m filtering technique.

The dc gain of the G_m-*OTA*-*C* integrator is

$$\frac{V_{out}(s)}{V_{ind}(s)} = G_{mo} r_{out} A_o \tag{58.22}$$

where G_{mo} is the dc transconductance of the transconductor, r_{out} is its output resistance, and A_o is the dc voltage gain of the *OTA*. The G_m-*OTA*-*C* integrator has two stages of gain, and the increase in gain relative to the G_m-*C* approach by the factor A_o results in a more ideal integrator characteristic at low frequency. Since two gain stages are used, cascoding is often eliminated or simplified so the improvement in the G_m-*OTA*-*C* integrator's dc gain over that of the G_m-*C* integrator may be less dramatic. Elimination of cascoding simplifies the circuit design, often results in reduced power dissipation, and enables lower voltage operation.

Signal summation in G_m-*OTA*-*C* filters is performed in the current domain at the summing node of the *OTA*. Instead of one G_m amplfier as in Fig. 58.23, the outputs of multiple transconductors with different values of G_m can be paralleled and drive the *OTA* summing node to obtain weighted summation and integration.

Many G_m-*OTA*-*C* integrator designs exist, but the one in Fig. 58.24 is a good example.[8] The transconductor consists of M1-M4, Q1, and Q2. The $V \rightarrow I$ mechanism is realized with triode operated devices, M1 and M2, that have constant drain-to-source voltages. Varying the base voltage of transistors Q1 and Q2 tunes the $V \rightarrow I$ converter. Unlike the $V \rightarrow I$ converter in Fig. 58.19, devices M1 and M2 are connected to ground rather than to a current source to improve the headroom in the G_m amplifier. The input common-mode voltage level at the gates of M1 and M2 must be well controlled so the drain current bias levels are fixed. M3 and M4 serve as simple current sources and their gate voltages are varied by a CMFB

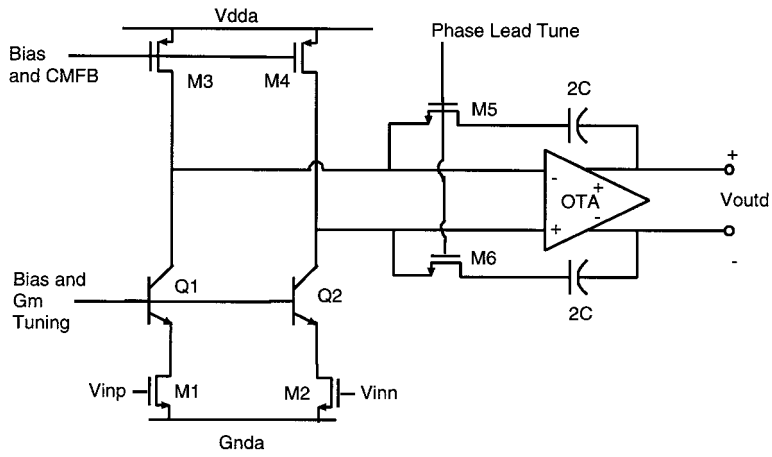


FIGURE 58.24 A BiCMOS G_m-*OTA*-*C* integrator with tunable phase lead.

circuit (not shown) to control the output common-mode voltage of the G_m stage. The second stage consists of the *OTA*, integrating capacitors, and phase lead transistors, M5 and M6. M5 and M6 operate in triode with a zero drain-to-source bias voltage and act as resistors. Their role is to provide phase lead in much the same way as the R_{lead} resistor does in the G_m-*C* design of Fig. 58.21. Here, the use of transistors instead of a fixed resistor permits the amount of phase lead to be adjusted for process and temperature variations. The phase lead tuning circuit in Ref. 8 is a dc control loop and does not use the more complex *Q* tuning techniques given in Refs. 6, 11, 16, and 18.

The *OTA* design can take the form of standard op-amp topologies. In Ref. 8 a simple one-stage BiCMOS design with cascoding was used, as shown in Fig. 58.25. CMFB control via the gates of M2 and M4 is required to control the average output voltages of V_{outp} and V_{outn}. Notice that although the G_m-*OTA*-*C* integrator requires two basic amplifiers (e.g., a G_m and an *OTA* amplifier), the amplifiers can often be simpler than the single G_m amplifier required in G_m-*C* filters. The power and bandwidth penalties of the G_m-*OTA*-*C* technique may therefore be less negative than at first view.

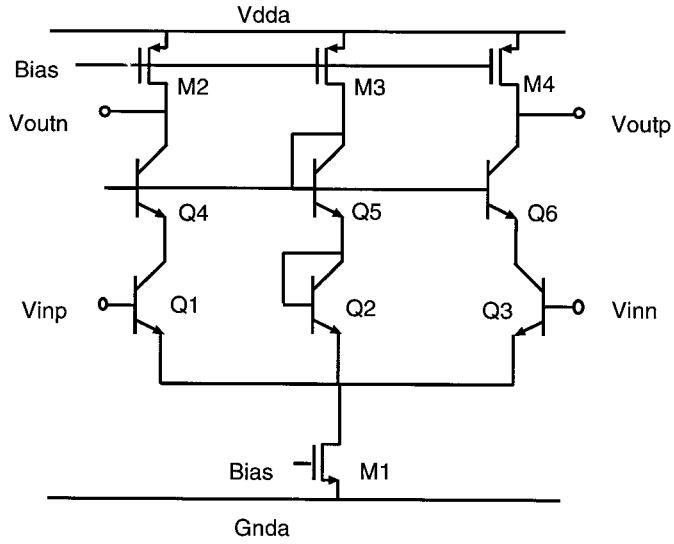


FIGURE 58.25 *OTA* for G_m-*OTA*-*C* integrator of Fig. 58.24.

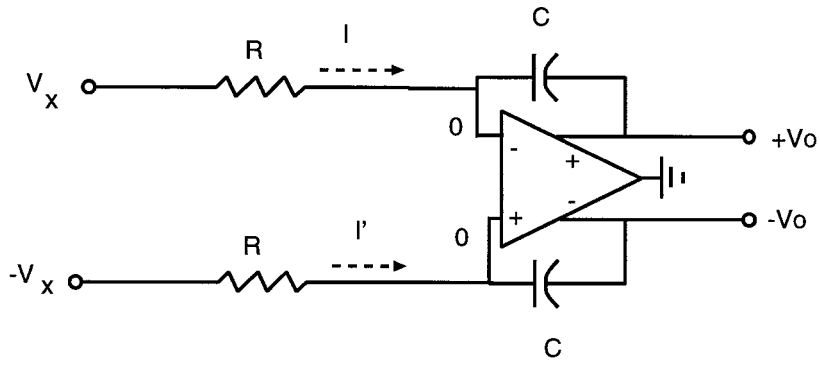


FIGURE 58.26 Balanced active RC integrator.

MOSFET-C Filters

MOSFET-C and G_m-*OTA-C* filters are similar, except that passive $V \rightarrow I$ devices are used to produce currents for summing into the *OTA* or op-amp in the *MOSFET-C* case, rather than an active $V \rightarrow I$ converter as in the G_m-*OTA-C* approach. Using two equal resistors for $V \rightarrow I$ conversion results in the fully balanced classical active-RC integrator of Fig. 58.26. For balanced inputs signals, V_x and $-V_x$, that have a common-mode level of zero, the input terminals of the op-amp are fixed at zero volts. The op-amp provides balanced outputs, $V_o(t)$ and $-V_o(t)$, that are derived to be:

$$V_o(t) - -V_o(t) = \frac{-1}{C}\int_{-\infty}^{t}\left[I(\tau) - I'(\tau)\right]d\tau \tag{58.23}$$

Therefore,

$$V_o(t) = \frac{-1}{RC}\int_{-\infty}^{t} V_x(\tau)\,d\tau \tag{58.24}$$

Obviously, using fixed resistors does not permit frequency scaling the integrator to remove temperature and process variations. If one replaces the resistors with MOSFETs operating in the triode region with zero drain-to-source bias, the *MOSFET-C* integrator shown in Fig. 58.27 results. Although each MOSFET is non-linear, operating with the fully balanced structure and assuming perfect matching of devices, the even-order non-linearities will cancel. Odd-order non-linearities are not canceled; however, they are low enough for many applications. Using Eq. 58.12 the output of the *MOSFET-C* integrator can be derived[35] as:

$$V_o(t) \approx \frac{-(W/L)\mu C_{ox}(V_C - V_Q - V_T)}{C}\int_{-\infty}^{t} V_x(\tau)\,d\tau \tag{58.25}$$

V_Q is the input common-mode bias level of the signals V_x and $-V_x$. The approximation results from the assumption that the odd-order non-linearities can be neglected. Notice that the op-amp input terminals are at voltage V_y, as opposed to zero. V_y will have a bias level of V_Q, but V_y will vary with the input signal V_x with a square-law characteristic.[36] For practical circuits, V_y will vary by a couple of hundred millivolts, resulting in modest input common-mode range requirements for the amplifier.

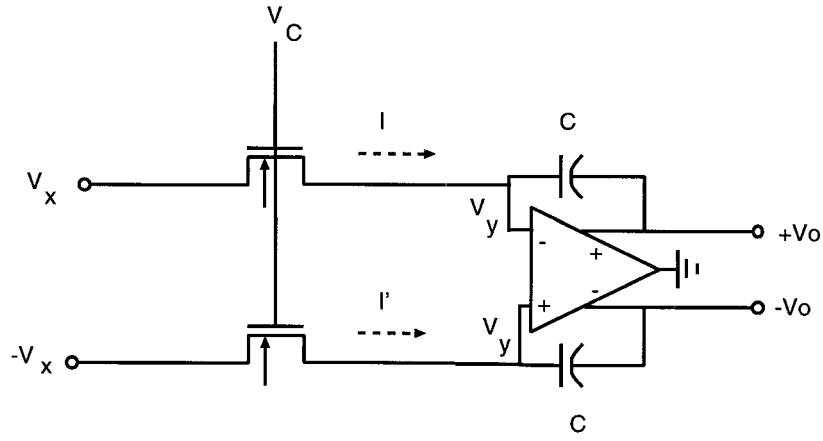


FIGURE 58.27 The *MOSFET-C* integrator.

The derivation of the *MOSFET-C* integrator linear input/output characteristic assumes that both transistors remain in the triode region. Such a requirement results in a coupling of the minimum permissible tuning voltage, V_{C-min} and the maximum input signal swing, V_{x-max}.

$$V_{x-max} = V_{C-min} - V_Q - V_T \tag{58.26}$$

Non-linearity cancellation in the *MOSFET–C* integrator depends on matched devices as well as fully balanced input signals. Since in a filter the signals that drive the integrator input come from a similar integrator stage, the op-amp used must have well-balanced outputs. If the signal balancing is imperfect in magnitude or if the signals are not exactly 180° out of phase, the even-order non-linearities will no longer be suppressed.[37] Magnitude errors of 1% and phase errors of 1° can be problematic.[37] The op-amp design must use a robust CMFB circuit consisting of an error amplifier and linear detector (i.e., two resistors) for the output common-mode detection, as described in Ref. 34. Many CMFB circuits that are adquate for differential switched-capacitor circuits do not provide adequate balancing for *MOSFET-C* filters.

Variations of the basic two-transistor fully balanced *MOSFET-C* integrator are possible.[26,38] In Ref. 38, the design of *MOSFET-C* filters with only single-ended output op-amps is described. In Ref. 26, the four-transistor *MOSFET-C* integrator shown in Fig. 58.28 in theory cancels even- *and odd*-order non-linearities.

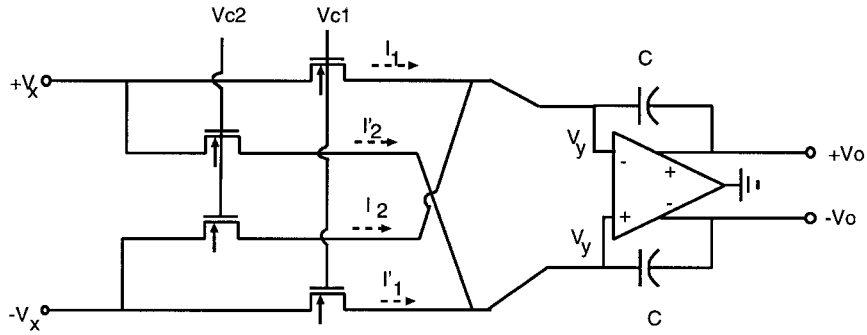


FIGURE 58.28 Four transistor *MOSFET-C* integrator.

Assuming perfectly matched transistors and constant mobility of carriers in the transistors,[33] the integrator output can be derived with Eq. 58.12 as:

$$V_o(t) = \frac{-(W/L)\mu C_{ox}(V_{C1} - V_{C2})}{C} \int_{-\infty}^{t} V_x(\tau)\, d\tau \tag{58.27}$$

Here, integrator tuning only depends on $V_{C1} - V_{C2}$, and is independent of the transistor threshold voltage, V_T, resulting in several advantages. First, in the basic two-transistor *MOSFET-C* integrator, body noise, V_B, can modulate the threshold voltage and couple into the integrator. The threshold voltage using the signal convention in the *MOSFET-C* integrator, is[33]:

$$V_T = V_{T0} + \gamma\left(\sqrt{V_Q - V_B + \phi_B} - \sqrt{\phi_B}\right) \tag{58.28}$$

where V_{T0} is the threshold voltage at zero source-to-body bias, and γ and ϕ_B are constants. Since the integrator's gain in the four-transistor *MOSFET-C* design is independent of V_T, immunity to body noise is achieved. The second advantage of the four-transistor *MOSFET-C* integrator is that very low integrator gains can be realized without loss of signal swing by making $V_{C1} - V_{C2}$ small, while simultaneously maximizing $V_{C1} + V_{C2}$. In contrast, to realize low gain, the basic *MOSFET-C* integrator requires $V_C - V_Q - V_T$ to be small, directly limiting the signal swing, as shown in Eq. 58.26. The primary disadvantage of the four-transistor *MOSFET-C* integrator is increased thermal noise and device sensitivity in comparison to the standard *MOSFET-C* circuit.[37] For most applications, the four-transistor integrator results in increased dynamic range (e.g., a few dB) and a larger tuning range.

Realizing *MOSFET-C* filters from SFGs, block diagrams, or classical active RC filters is straightforward. The *MOSFET-C* equivalent of the Tow-Thomas biquad (Fig. 58.4) is constructed as in Fig. 58.29. Each resistor in the active RC filter is replaced with a MOSFET, and the single-ended topology is converted to a balanced structure by mirroring all devices around the line of symmetry going between the op-amp input terminals and its output. Mirroring and use of a fully balanced structure is mandatory to obtain cancellation of even-order non-linearities in the triode-operated MOSFETs. Signal inversion is obtained by taking the opposite output terminal of the op-amp, so the inverting amplifier in the classical single-ended design is not required in the *MOSFET-C* approach. Weighted signal summation is achieved by

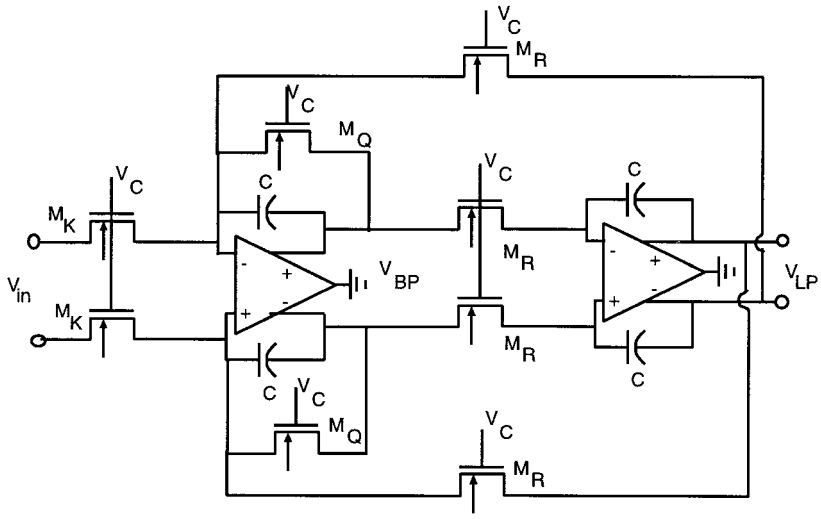


FIGURE 58.29 *MOSFET-C* version of Tow-Thomas biquad.

connecting multiple transistors to the op-amp summing node and scaling the relative W/*L* ratios of the devices. The gates of all MOSFETs are connected to a single control voltage, V_C, for frequency tuning the filter.

The frequency response of *MOSFET-C* filters can become distorted due to integrator gain and phase errors, as in the case of G_m-*C* and G_m-*OTA-C* filters. As before, phase errors are more problematical. Integrator phase errors in the *MOSFET-C* approach are due to the finite op-amp gain–bandwidth product as well as the distributed capacitance of the MOSFET. The triode-operated MOSFET acts as a linear tunable resistor for small signals; however, parasitic capacitance between the channel-to-gate terminal and from the channel-to-body terminal makes the transistor act as a uniformly distributed RC transmission line.[33,37] The transistor is then a lowpass filter between the integrator input and the op-amp summing node, creating phase lag in the integrator's response.

The channel resistance R_T and capacitance C_T of the MOSFET are given by

$$R_T = \frac{1}{\mu C_{ox}\left(W/L\right)\left[V_C - V_Q - V_T\right]} \tag{58.29}$$

$$C_T = WLC_{ox}\left(b+1\right) \tag{58.30}$$

where b is the backgate effect and is approximately 0.1. As discussed in Ref. 37, for most applications, the MOSFET can be modeled as a first-order lumped circuit resulting in the balanced small-signal *MOSFET-C* integrator model of Fig. 58.30. The transfer function of the integrator is then

$$\frac{V_o}{V_i} = \frac{-\omega_o}{s}\left[\frac{1}{1+\dfrac{s}{\omega_\tau}}\right] \tag{58.31}$$

where $\omega_o = 1/R_TC$ and $\omega_\tau = 1/R_TC_T$. Clearly, distributed capacitance adds extra phase lag. Note that ω_o is proportional to $1/L$, but ω_τ is proportional to $1/L^2$, where L is the channel length. Therefore, *MOSFET-C* filters will in general have worse distributed effects for *low* frequency filters because long channel lengths are used.

MOSFET-C integrator phase errors due to distributed effects and op-amp finite gain-bandwidth product can be compensated by introducing a high frequency zero by either adding a small series resistor to each integrating capacitor or by adding capacitor C_c in parallel with the MOSFETs, as shown in

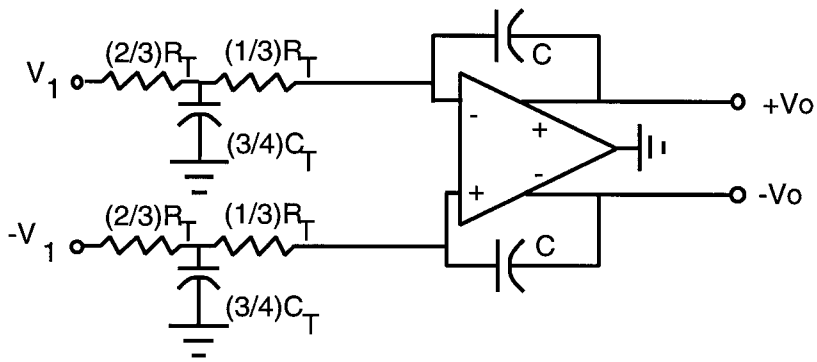


FIGURE 58.30 Small-signal model of *MOSFET-C* integrator with MOSFET channel capacitance.

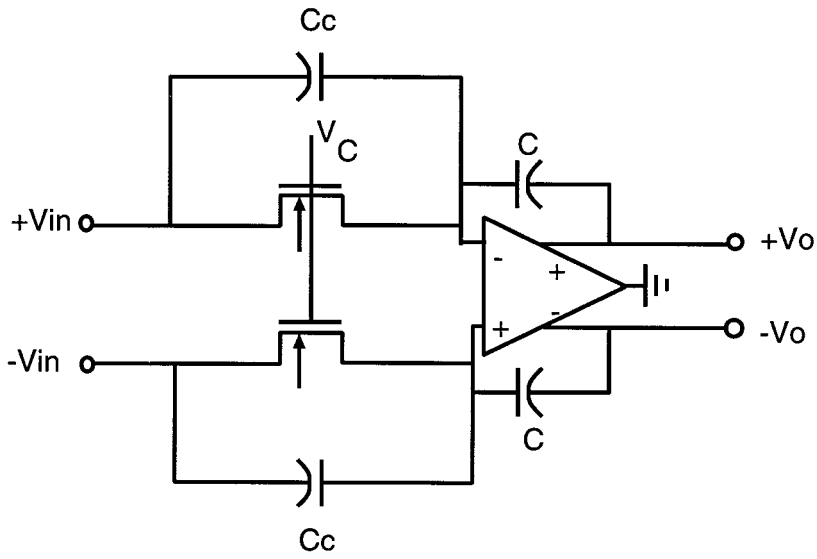


FIGURE 58.31 Phase compensated *MOSFET-C* integrator.

Fig. 58.31. Assuming an op-amp frequency response model of $A(s) = \omega_T/s$, C_c should be selected as follows to null the integrator phase error at the unity-gain frequency[37]:

$$C_c \approx \frac{C_T}{6} + \frac{\omega_o}{\omega_T} C \tag{58.32}$$

The use of a fixed phase compensation capacitor, such as C_c, or a fixed series resistor is often appropriate for low-quality factor filters, but not for high-Q filters (e.g., $Q > 10$).

Alternate Continuous-Time Filter Techniques

Although the vast majority of integrated continuous-time filters fall into the G_m-*C*, G_m-*OTA-C*, or *MOSFET-C* approaches, other techniques have been developed for applications requiring high linearity. The approaches described so far permit continuous frequency scaling of the filter and hence require MOS or bipolar transistors as variable resistors or transconductors. Since transistors are inherently non-linear, use of nominally matched devices, balanced signals, etc. must be employed to achieve overall linear operation. Practical issues generally limit the linearity to 60 dB for signal swings on the order of 1 V. To achieve linearity in the 80-dB range and higher, use of linear passive devices (e.g., resistors) has helped.[2-4,39,40]

The linearity of the *MOSFET-C* integrator can be vastly improved if less drain-to-source voltage is dropped across the MOSFETs for maximum input signals. Adding resistors in series with the input signals will improve linearity, but decrease the tuning range. With the addition of two transistors connected to a second control voltage V_{c2}, as in Fig. 58.32, the tuning range can be restored.[4] The majority of the input voltage is dropped across the resistors for excellent linearity, and the four transistors are used to simply steer signal current to ground or to the op-amp summing node. As V_{c2} is increased, more current is diverted to ground, leaving less for the summing node, so the integrator gain is reduced. This configuration has been used to achieve in excess of 90-dB linearity for digital audio applications.[4] In this technique, the integrator loop gain is low, restricting the signal bandwidth to levels significantly below standard *MOSFET-C* filters.

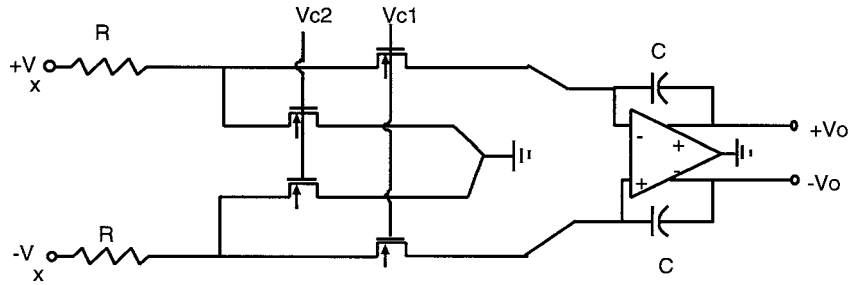


FIGURE 58.32 *R-MOSFET-C* integrator.

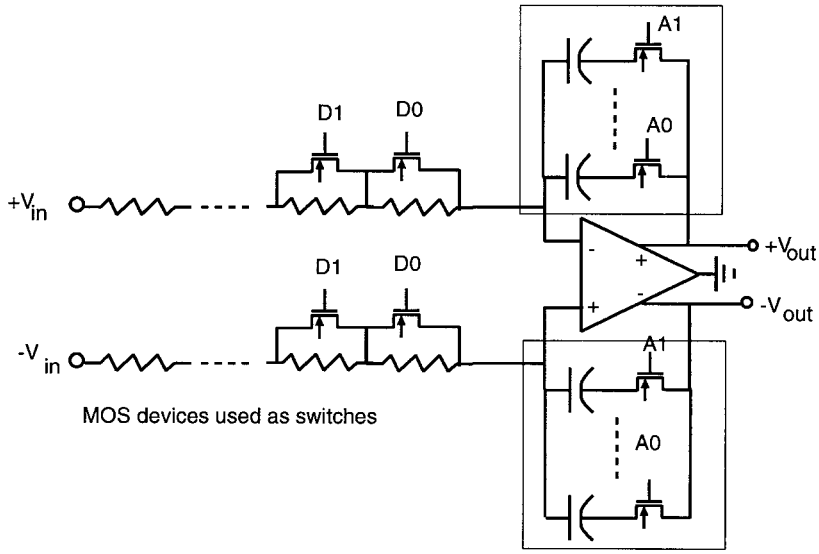


FIGURE 58.33 Programmable active RC integrator.

If continuous frequency scaling can be replaced with discretized tuning, then switchable active RC filters can be used to maximize linearity.[2,3,39,40] The basic concept, as shown in Fig. 58.33, is to build the integrator with linear resistors and capacitors. MOS switches are then used to program the values of capacitors and/or resistors. In the figure, a series connection of resistors is shown; however, parallel combinations are also feasible.[2] This technique has no inherent bandwidth restrictions, but in practice the MOS switches add parasitic capacitance that cause phase lag, disturbing the integrator's frequency response. A design tradeoff is then tuning resolution versus switch parasitics. To provide adequate tuning range to cover process and temperature variations (e.g., ±50%) and a minimization of switch parasitics, typically results in tuning resolution of ±2 to ±5%. Filter tuning can use microprocessor control[3] or a replica master integrator or delay circuit with up/down counters.[2,39,40] Finally, since the components are inherently linear, use of balanced structures is not mandatory and a larger class of classical active RC filters can be implemented on-chip. Single-amplifier biquad structures[19,40] then become attractive.

58.4 Filter Tuning Circuits

Integrated continuous-time filters require on-chip tuning circuits to control the corner or center frequencies in the face of process and temperature variations. Only in some non-critical applications, such as anti-alias or reconstruct filters, can the ±50% variation in filter time constants be tolerated. Filter

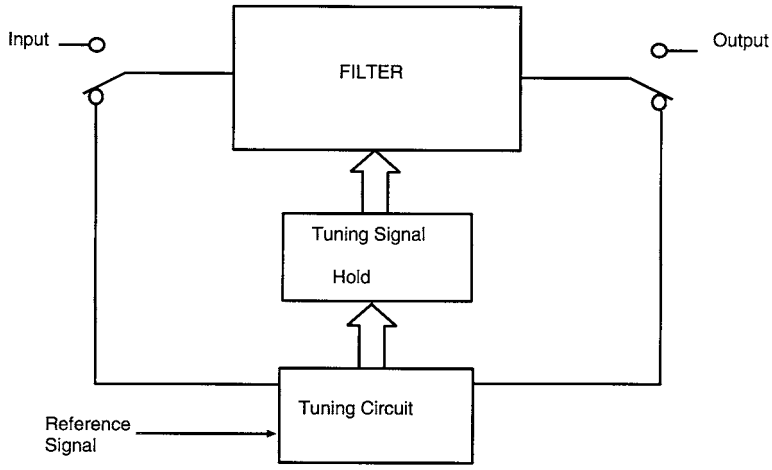


FIGURE 58.34 A direct-tuned filter.

tuning, or frequency scaling of the filter, can be accomplished with either direct or indirect tuning methods.

Direct Tuning

Direct tuning, illustrated conceptually in Fig. 58.34, is described in Ref. 41. The filter is periodically removed from the signal path and, after measurements are made with respect to a reference signal, the required adjustments are applied to the filter and held with analog or digital storage means. The filter is then returned to the signal path. The advantage of this approach is that the filter is measured directly and excellent accuracy is possible. Since many applications cannot tolerate interruption of the signal path, it is conceptually possible to use two filters[41]: one processes the signal while the other is tuned. However, the complexity and chip area overhead cause this alternative to be rarely used. Direct tuning is seldom used, but a production example does exist.[42]

Master-Slave Tuning

By far the most prevalent tuning method is the indirect or master-slave tuning technique. In this aproach, a master circuit that is made of the same components as the main filter is tuned with respect to a reference signal or component. The tuning signal that adjusts the master is also used to tune the slave or main filter. Since components in the master and slave circuits are on the same chip, they will match accurately. The time constants of the slave will then track the master over all process and temperature variations. If the master and slave components are placed closely in the layout, and good layout techniques are used, then matching of a few percent is possible.

One simple dc master-slave frequency tuning technique, shown in Fig. 58.35, uses a precision off-chip resistor. A small voltage V is applied to the MOSFET and $-V$ is applied to the resistor. The circuit converges when $I_1 = I_2$, so that $r_{ds} = 1/R_s$. The process and temperature variations of the MOSFET resistance are then removed; however, capacitor values in the filter will need trimming during manufacture. An ingenious variation[43] replaces the external precision resistor with an on-chip switched-capacitor equivalent resistance as shown in Fig. 58.36. If switched-capacitor C_s is clocked at a rate f_s, the charge transfer from the input voltage V to the summing node is equivalent to that which would be obtained with a resistor of value

$$R_{sc} = \frac{1}{f_s C_s} \tag{58.33}$$

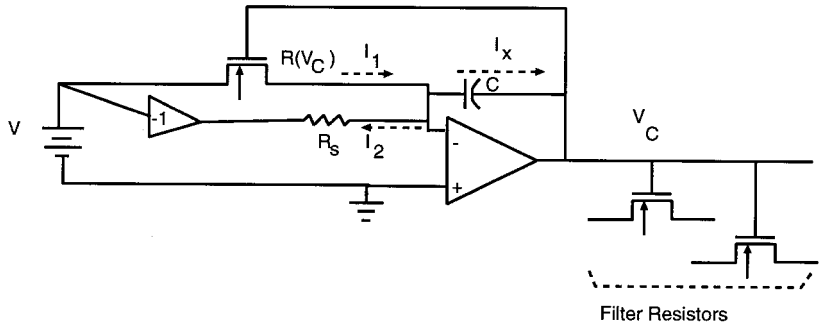


FIGURE 58.35 Reference resistor-based master-slave tuning circuit.

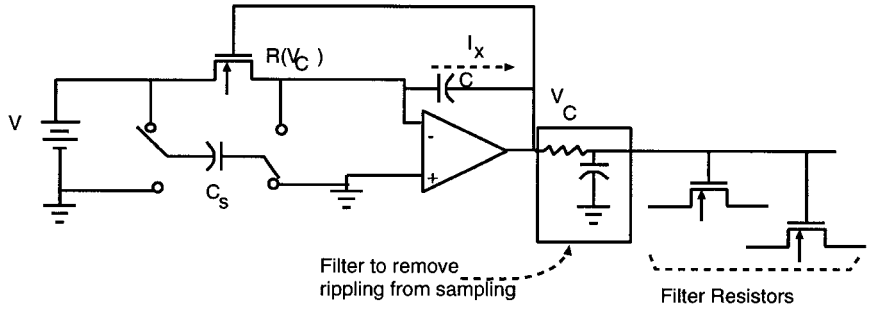


FIGURE 58.36 Switched-capacitor-based master-slave tuning circuit.

The average current flowing through the switched-capacitor will be forced equal to the drain current in the MOSFET due to the negative feedback resulting in $I_x = 0$. The voltage V_C that tunes the MOSFET in the master cell will then reach an average dc value with some ripple due to the switched-capacitor circuit operation. After additional filtering of V_C to remove the ripple, the tuning signal is routed to the slave filter. The slave filter will possess many time constants and critical frequencies. A critical frequency, f_x, will be determined by a MOSFET channel resistance r_{ds-x} and a capacitance, C_x. The tuning circuit then ensures that

$$f_x = f_s\left(\frac{r_{ds}}{r_{ds-x}}\right)\left(\frac{C_s}{C_x}\right) \tag{58.34}$$

The filter's critical frequency, f_x, is then accurate since it is determined by the clock reference, f_s, the ratio of the channel resistance of transistors, and the ratio of capacitors. Clearly, the switched-capacitor tuning technique has the benefit that it tunes time constants, not just resistance values. The master-slave dc tuning methods shown here are equally applicable to G_m-C and G_m-OTA-C filters.

Often, the master-slave tuning approach uses a phase-lock loop (PLL) and a master cell that is either a voltage-controlled filter (VCF) or a voltage-controlled oscillator (VCO). First, consider the case of a master VCF.[6,10] Many options exist for the design of the VCF, but a reasonable design uses a lowpass biquadratic filter. The transfer function of a lowpass biquad is given by

$$V_{o-lp}(s) = \frac{K\omega_o^2}{s^2 + \frac{\omega_o}{Q}s + \omega_o^2}V_{in}(s) \tag{58.35}$$

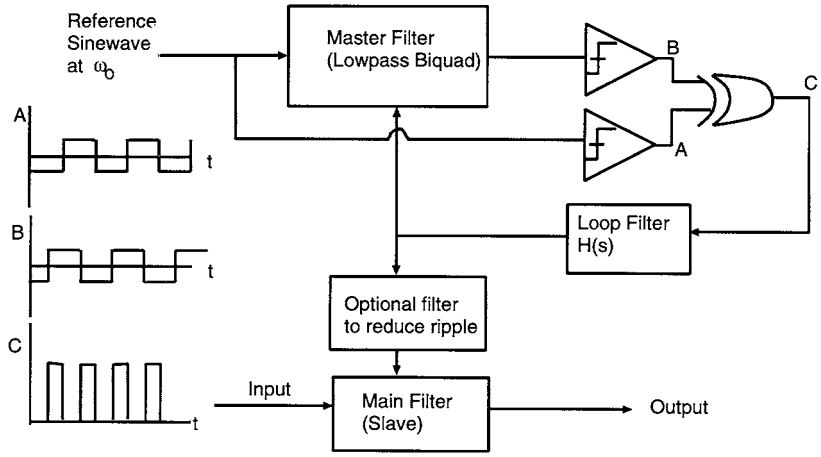


FIGURE 58.37 A VCF-based Master-slave tuning system.

where K is a gain constant. If a sinewave at frequency ω_o is applied to the filter's input, the phase of the output will be –90° with respect to the input, provided that the filter is accurately tuned. The PLL tuning circuit will adjust the RC products of the VCF until the phase shift is –90°. One embodiment of the VCF master-slave tuning method is shown in Fig. 58.37. The exclusive-OR (XOR) gate phase detector is preceded by two slicers that convert the filter's sinewave input and output signals to logic levels. The loop filter, $H(s)$, has a first-order lowpass response and the feedback loop forces the tuning voltage (or current) at the output of the loop filter to converge to a dc value, which simultaneously frequency tunes the master biquad and slave filter. Since the loop filter output will contain ripple, extra lowpass filtering can be applied outside the feedback loop so as not to compromise loop stability while eliminating the ripple on the tuning signal in the slave filter.

The tuning accuracy of this approach is directly linked to the loop gain of the PLL, which is derived as

$$\text{Loop gain} = K_D H(s) K_{VCF} = K_D H(s)\left(\frac{\partial \omega_o}{\partial V_C}\right)\left(\frac{-2Q}{\omega_o}\right) \tag{58.36}$$

K_D is the phase detector gain in volts/radian, K_{VCF} is the master filter's gain in radians/volt, V_C is the filter's tuning signal, and Q and ω_o are the quality factor and corner frequency of the master biquad, respectively. To maximize the tuning accuracy, the loop gain should be maximized. The gain is kept high by using a relatively high-Q master filter (e.g., $Q > 5$) and an integrator for the loop filter. Frequency tuning errors will occur if there are static phase errors in the PLL. Static phase errors will not cause PLL convergence difficulty; however, the relative phase of the biquad's input and output signals may differ from –90°. Static phase errors can occur due to delay mismatches in the slicers or rise and fall time asymmetries in the XOR phase detector.[6]

The PLL with VCF master-slave tuning approach is appealing because the master circuit is simply a biquad that closely resembles a portion of the slave filter. The only drawback is that this tuning method requires a sinusoidal reference signal that is not usually available in most system applications. Unfortunately, use of a commonly available square wave reference signal is unacceptable because the large harmonic content shifts the zero crossings at the master biquad's output, creating a systematic error in phase measurements, resulting in a fixed frequency tuning error. Triangular input signals work better since the harmonic content is lower.[6]

Since sinusoidal reference signals are rarely available, the more common PLL tuning approach uses a VCO in a classical PLL circuit.[44] Use of a master VCO with PLL has been widely applied to master-slave continuous-time filter tuning.[1,7,9,13,14,27] This master-slave tuning technique, depicted in Fig. 58.38, uses

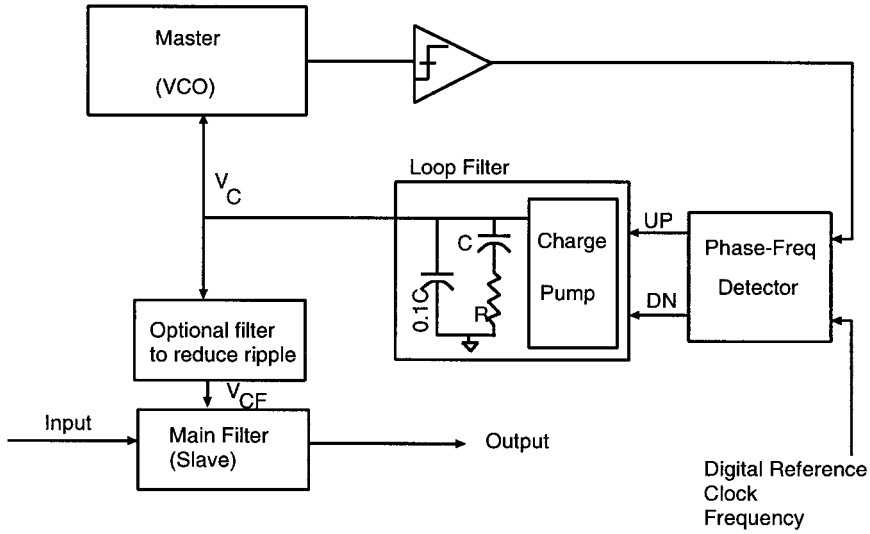


FIGURE 58.38 A VCO-based master-slave tuning system.

a master oscillator that is constructed with the same type of capacitors and resistors (or transconductors) used in the slave filter. The VCO produces a sinusoidal output, so a slicer converts the signal to a square wave for application to the digital phase-frequency detector. Some implementations use an XOR gate, but the phase-frequency detector enhances the capture range of the PLL and prevents harmonic locking.[44] In Fig. 58.38 the loop filter is implemented with a charge pump that has fixed source and sink current levels coupled to a resistor and capacitor in series to ground. The capacitor and charge pump form an integrator and the series resistor introduces a zero to stabilize the loop. As in the previous case, the output of the loop filter tunes both the master oscillator and the slave filter. However, unlike the PLL with VCF tuning method, the PLL will lock the VCO to *exactly* the same frequency as the input reference signal even if static phase errors occur. In this application, static phase errors between the input reference and the VCO output are inconsequential.

The primary drawback with the PLL and VCO master-slave tuning method is that the VCO design is difficult. Virtually any poorly designed VCO can be placed in the PLL and lock to the reference frequency; however, the issue is whether the control voltage, V_C, that is developed will correctly tune the slave filter. In other words, the voltage-to-frequency conversion curve for the VCO and the slave filter must be identical or at least they must match to better than about ±1% over the full range of tuning levels. Designing a slave filter that has a well-defined tuning voltage-to-corner frequency curve is relatively easy. In contrast, unless extreme care is taken in the VCO design, the amplitude of oscillation and hence VCO linearity can be coupled to the tuning signal level. The amplitude of oscillations in the VCO must be well controlled in the linear region so that the MOSFETs or transconductors in the master operate at the same levels as in the slave so that the voltage-to-frequency characteristics are nominally the same. This issue has been addressed in Ref. 7, but more work needs to be done.

Q Tuning Loops

Frequency tuning circuits hold the filter's time constants fixed over process and temperature variations or, equivalently, hold constant an integrator's unity-gain frequency. In contrast, Q tuning loops adjust an integrator's phase response to achieve –90° over all process and temperature conditions. Rather than directly tune the phase response of an integrator, the Q of a bandpass biquad can be adjusted instead[6,11,16] as shown in Fig. 58.39. If the integrators have phase lag, the Q will be enhanced; if they have phase lead, the Q will be reduced. The input-output relationship of the bandpass biquad is

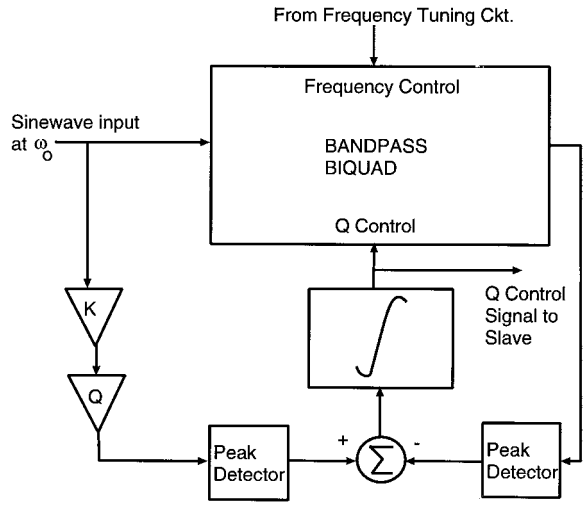


FIGURE 58.39 A master-slave Q tuning loop.

$$V_{o-bp}(s) = \frac{K\omega_o s}{s^2 + \frac{\omega_o}{Q}s + \omega_o^2} V_{in}(s) \tag{58.37}$$

If a sinusoid of frequency ω_o is applied to the biquad input and the biquad is properly frequency scaled by a tuning loop (not shown), then the sinusoid at the biquad output will have an amplitude *KQ*. Since *K* will be defined by well-matched components, the output amplitude error can be directly attributed to errors in *Q*. By subtracting the amplitude of the biquad's input and output and integrating, the *Q* error can be driven to zero with the negative feedback shown. Amplitude measurements can use r.m.s.-type detectors or peak amplitude detecting circuits, as shown in the figure. Many variations of Q tuning loops exist.[6,11,16,18]

Master-slave Q tuning loops should be used with caution for two reasons. First, if the tuning loop incorrectly adjusts the integrator's phase response, even momentarily during a transient, the slave filter can break into oscillation. Second, integrator phase errors are in large part due to parasitic capacitances. For the master-slave Q tuning to be successful, parasitics in the master and slave must match and track, suggesting that meticulous attention to layout detail is required. Additionally, excellent modeling and extraction of the parasitics from the layout are necessary to precisely design these circuits.

58.5 Conclusion

Integrated continuous-time filters are now found in a variety VLSI chips spanning the frequency range from a few kHz for ultra-low power hearing aid applications to over 150 MHz for the hard disk drive read channel application. The continuous-time filtering techniques described in this chapter have become integral parts of production quality chips and now represent well-established analog filter design practices.

References

1. M. Banu, and Y. Tsividis, An elliptic continuous-time CMOS filter with on-chip automatic tuning, *IEEE Journal of Solid-State Circuits*, vol. SC-20, no. 6, pp. 1114-1121, Dec. 1985.
2. A. Durham, J. Hughes, and W. Redman-White, Circuit architectures for high linearity monolithic continuous-time filtering, *IEEE Transactions on Circuits and Systems,* pp. 651-657, Sept. 1992.

3. H. Khorramabadi, M. Tarsia, and N. Woo, Baseband filters for IS-95 CDMA receiver applications featuring digital automatic frequency tuning, *1996 International Solid State Circuits Conference*, pp. 172-173.
4. U.-K. Moon and B.-S. Song, Design of a low-distortion 22-kHz fifth-order bessel filter, *IEEE Journal of Solid State Circuits,* vol. 28, no. 12, pp. 1254-1264, Dec. 1993.
5. S. Willingham, K. Martin, and A. Ganesan, A BiCMOS low distortion 8-MHz low-pass filter, *IEEE Journal of Solid State Circuits,* vol. 28, no. 12, pp. 1234-1245, Dec. 1993.
6. V. Gopinathan, Y. Tsividis, K-S Tan, R. Hester, Design considerations for high-frequency continuous-time filters and implementation of an antialiasing filter for digital video, *IEEE Journal of Solid State Circuits,* vol. SC-25, no. 6, pp. 1368-1378, Dec. 1990.
7. J. Koury, Design of a 15 MHz continuous-time filter with on-chip tuning, *IEEE Journal of Solid-State Circuits,* vol. 26, no. 12, pp. 1988-1997, Dec. 1991.
8. C. Laber and P. Gray, A 20MHz 6th order BiCMOS parasitic insensitive continuous-time filter and second order equalizer optimized for disk-drive read channels, *IEEE Journal of Solid State Circuits,* vol. 28, pp. 462-470, Apr. 1993.
9. F. Krummenacher and N. Joehl, A 4 MHz CMOS continuous-time filter with on-chip automatic tuning, *IEEE Journal of Solid-State Circuits,* vol. 23, no. 3, pp. 750-758, June 1988.
10. H. Khorramabadi and P. R. Gray, High-frequency CMOS continuous-time filters, *IEEE Journal of Solid-State Circuits,* vol. SC-19, no. 6, pp. 939-948, Dec. 1984.
11. C. Chiou and R. Schaumann, Design and performance of a fully integrated bipolar 10.7 MHz analog bandpass filter, *IEEE Transactions on Circuits and Systems,* vol. CAS-33, no. 2, pp. 116-124, Feb. 1986.
12. B-S Song, and P. R. Gray, Switched-capacitor high-Q bandpass filters for IF applications, *IEEE Journal of Solid State Circuits,* vol. SC-21, no. 6, pp. 924-933, Dec. 1986.
13. K. W. Moulding, J. R. Quartly, P. J. Rankin, R. S. Thompson, and G. A. Wilson, Gyrator video filter IC with automatic tuning, *IEEE Journal of Solid State Circuits*, vol. SC-15, no. 6 pp. 963-968, Dec. 1980.
14. K. S. Tan and P. R. Gray, Fully integrated analog filters using bipolar FET technology, *IEEE Journal of Solid-State Circuits*, vol. SC-13, no. 6, pp. 814-821, Dec. 1978.
15. K. R. Rao, V. Sethuraman, and P. K. Neelakantan, A novel 'follow the master' filter, *Proceedings of the IEEE,* vol. 65, pp. 1725-1726, 1977.
16. D. Senderowicz, D. Hodges, and P. Gray, An NMOS integrated vector-locked loop, *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 1164-1167, May 1982.
17. J. Silva-Martinez, M. Steyaert, and W. Sansen, A 10.7-MHz 68-dB SNR CMOS continuous-time filter with on-chip automatic tuning, *IEEE Journal of Solid State Circuits*, vol. 27, no. 12, pp. 1843-1853, Dec. 1992.
18. S. Pavan and Y. Tsividis, An analytical solution for a class of oscillators, and its application to filter tuning, *IEEE Transactions on Circuits and Systems. I*, vol. 45, no. 5, May 1998.
19. A. Sedra, and P. Brackett, *Filter Theory and Design: Active and Passive*, Matrix Publishers, Inc., Beaverton, OR, 1978.
20. W. Snelgrove and A. Sedra, Synthesis and analysis of state-space active filters using intermediate transfer functions, *IEEE Transactions on Circuits and Systems*, vol. CAS-33, no. 3, pp. 287-301, March 1986.
21. K. Bult and G. Geelen, A fast-settling CMOS op amp with 90 dB DC-gain and 116 MHz unity-gain frequency, *International Solid State Circuits Conference*, pp. 108-109, 1990.
22. P. Gray and R. Meyer, *Analysis and Design of Analog Integrated Circuits*, John Wiley & Sons, second edition, 1984.
23. A. Nedungadi and T. R. Viswanathan, Design of linear CMOS transconductance elements, *IEEE Transactions on Circuits and Systems*, vol. CAS31, pp. 891-894, October 1984.
24. Y. Tsividis, Z. Czarnul, and S. C. Fang, MOS transconductors and integrators with high linearity, *Electronics Letters*, vol. 22, pp. 245-246, Feb. 1986.

25. M. Banu and Y. Tsividis, Detailed analysis of nonidealities in MOS fully integrated active RC filter based on balanced networks, *Proceedings of the Institute of Elect. Eng.*, vol. 131, Pt. G, no. 5, pp. 190-196, Oct. 1984.
26. Z. Czarnul, Modification of the banu-tsividis continuous-time integrator structure, *IEEE Transactions on Circuits and Systems*, vol. CAS-33, no. 7, pp. 714-716, July 1986.
27. D. Welland, S. Phillip et al., A digital read/write channel with EEPR4 detection, *IEEE International Solid-State Circuits Conference*, San Francisco, pp. 276-277, 352, 1994.
28. J. Pennock, CMOS triode transconductor for continuous-time active integrated filters, *IEE Electronic Letters*, vol. 21, no. 18, pp. 817-818, Aug. 1985.
29. R. Alini, A. Baschirotto, and R. Castello, Tunable BiCMOS continuous-time filter for high-frequency applications, *IEEE Journal of Solid State Circuits*, vol. 27, no. 12, pp. 1905-1915, Dec. 1992.
30. Y. Tsividis and J. Voorman, *Integrated Continuous-Time Filters: Principles, Design and Applications*, IEEE Press, New York, 1993.
31. D. Calder, Audio frequency gyrator filters for an integrated paging receiver, *IEEE Conference, Mobile Radio Systems and Techniques*, no. 238, 1984.
32. J. Voorman, W. Bruls, and P. Barth, Integration of analog filters in a bipolar process, *IEEE Journal of Solid-State Circuits*, vol. SC-17, no. 4, pp. 713-722, Aug. 1982.
33. Y. Tsividis, *Operation and Modeling of the MOS Transistor*, McGraw-Hill, New York, 1987.
34. M. Bani, J. Khoury, and Y. Tsividis, Fully differential operational amplifiers with accurate output balancing, *IEEE Journal of Solid-State Circuits*, vol. SC-23, no. 6, pp. 1410-1414, Dec. 1988.
35. M. Banu and Y. Tsividis, Fully integrated active RC filters in MOS technology, *IEEE Journal of Solid-State Circuits*, vol. SC-18, pp. 644-651, Dec. 1983.
36. Y. Tsividis, M. Banu, and J. Khoury, Continuous-time MOSFET-C filters in VLSI, *IEEE Journal of Solid State Circuits*, vol. SC-21, no. 1, Feb. 1986, pp. 15-30; and *IEEE Transactions on Circuits and Systems*, vol. CAS-33, no. 2, pp. 125-140, Feb. 1986.
37. J. Khoury, and Y. Tsividis, Analysis and compensation of high frequency effects in integrated MOSFET-C continuous-time filters, *IEEE Transactions on Circuits and Systems*, vol. CAS-34, no. 8, pp. 862-875, Aug. 1987.
38. M. Ismail, A new MOSFET capacitor integrator, *IEEE Transactions on Circuits and System*, vol. CAS-32, no. 11, pp. 1194-1196, Nov. 1985.
39. A. Durham and W. Redman-White, Integrated continuous-time balanced filters for 16-b DSP interfaces, *IEEE Journal of Solid State Circuits* vol. 28, no. 7, pp. 835-839, July 1993.
40. R. Shariatdoust, K. Nagaraj, J. Khoury, S. Daubert, and D. Fasen, An integrating servo demodulator for hard disk drives, *IEEE 1993 Custom Integrated Circuits Conference*, pp. 10.6.1-10.6.5, 1993.
41. Y. Tsividis, Self-tuned filters, *Electronics Letters*, vol. 17, no. 12, pp. 406-407, June 1981.
42. G. Smolka, U. Riedle, U. Grehl, B. Jahn, F. Parzefall, W. Veit, and H. Werker, A low-noise trunk interface circuit with continuous-time filters and on-chip tuning, in *Integrated Continuous-Time Filters: Principles, Design and Applications*, Edited by Y. Tsividis and J. Voorman, IEEE Press, New York, 1993.
43. T. R. Viswanathan, S. Murtuza, V. Syed, J. Berry, and M. Staszel, Switched-capacitor frequency control loop, *IEEE Journal of Solid-State Circuits*, vol. SC-17, no. 4, pp. 775-778, Aug. 1982.
44. F. Gardner, *Phase-Lock Techniques*, John Wiley & Sons, New York, 1966.

Baschirotto, A. "Swithced-Capacitor Filters"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

59

Switched-Capacitor Filters

Andrea Baschirotto
Università di Pavia

59.1 Introduction

The accuracy of the absolute value of integrated passive devices (R and C) is very poor. As a consequence, the frequency response accuracy of integrated active-RC filters is poor and they are not feasible when high-accuracy performance is needed. A possible solution for the implementation of analog filters with accurate frequency response was given by the switched-capacitor (SC) technique since the late 1970s.[1,2] Their popularity has further increased since they can be realized with the same standard CMOS technology used for digital circuits. In this way, fully integrated low-cost high-flexibility mixed-mode systems have become possible. The main reasons of the large popularity of SC networks can be summarized as follows:

1. The basic requirements of SC filters fit the popular MOS technology features. In fact, the infinite input impedance of the operational amplifier (op-amp) is obtained using a MOS input device;

MOS transconductance amplifiers can be used since only-capacitive load is present; precise switches are realized with MOS transistor; and capacitors are available in the MOS process.

2. SC filter performance accuracy is based on the matching of integrated capacitors (and not on their absolute values). In a standard CMOS process, the capacitor matching error can be less than 0.2%. As a consequence, SC systems guarantee very accurate frequency response without component trimming. For the same reason, temperature and aging coefficients track reducing performance sensitivity to temperature and aging variations.
3. It is possible to realize SC filters with long time constants without using large capacitors and resistors. This means a chip area saving, with respect to active-RC filter implementations.
4. SC systems operate with closed-loop structures; this allows one to process large swing signals and to achieve large dynamic range.

On the other hand, the major drawbacks of the SC technique can be summarized in the following points:

1. To process a fully analog signal, an SC filter has to be preceded by an anti-aliasing filter and followed by a smoothing filter, which complicate the overall system and increase power and die size.
2. The op-amps embedded in an SC filter have to perform a large dc-gain and a large unity-gain bandwidth, much larger than the bandwidth of the signal to be processed. This limits the maximum signal bandwidth.
3. The power of noise of all the sources in the SC filter is folded in the band [0–Fs/2]. Thus, their nose power density is increased by the factor (Fs/2)/Fb, where Fs is the sampling frequency and Fb is the noise bandwidth at the source.

From its first proposals, the SC technique has been deeply developed. Many different circuits solutions have been realized with the SC technique not only in analog filtering, but also in analog equalizers, analog-to-digital and digital-to-analog conversion (including in particular the oversampled SD converters), Sample&Hold, Track&Hold, etc.

In this chapter, an overview of the main aspects of the SC technique is given, leaving to the reader to study the large literature for details more related to his necessity. A few advanced solutions feasible for future SC systems are given in the last section.

59.2 Sampled-Data Analog Filters

An SC filter is a continuous-amplitude, sampled-data system. This means that the amplitude of the signals can assume any value within the possible range in a continuous manner. On the other hand, these values are assumed at certain time instants and then they are held for the entire sampling period. Thus, the resulting waveforms are not continuous in time but look like a staircase.

In an SC filter, the continuous-time input signal is firstly sampled at sampling frequency Fs and then processed through the SC network. This sampling operation results in a particular feature of the frequency response of the SC system. In the following, the aspects relative to the sampling action are illustrated from an intuitive point of view, while a more rigorous description can be found in Ref. 3.

The sampling operation extracts from the continuous-time waveform the values of the input signal at the instant k·Ts (k = 1,2,3, …), where Ts is the sampling period (Ts = 1/Fs). This is shown in Fig. 59.1 for a single-input sinewave at $fo = 0.16 \cdot Fs$ (i.e., with $fo < Fs/2$).

If the input sinewave is at $Fs + fo$, the input sequence of analog samples is exactly equal to that previously obtained with *fo* as input frequency (see Fig. 59.2). Both sequences should then be processed in exactly the same way by the SC network, and the overall filter output sequence should then result to be again identical. The two input sinewaves result then to be indistinguishable after the sampling action. This effect is called *aliasing*. It can be demonstrated that a sinewave at frequency *fo* in the range [0–Fs/2] is aliased by the components at frequency f_{al} given by:

$$f_{al} = k \cdot Fs \pm fo \qquad (k = 1, 2, 3, \ldots) \tag{59.1}$$

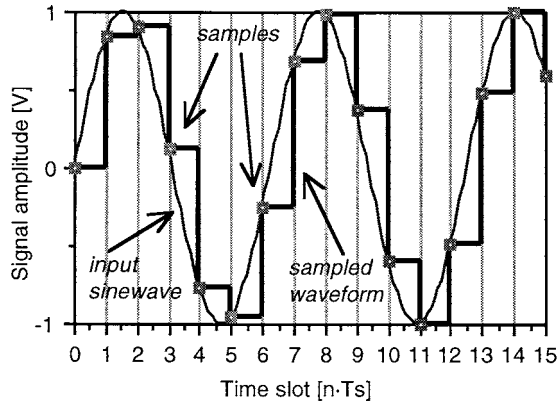


FIGURE 59.1 Sampling of the input signal.

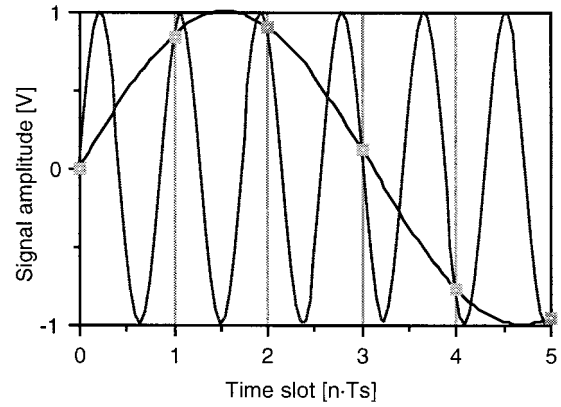


FIGURE 59.2 Aliasing between fo and FS + fo.

As a consequence, in order to avoid frequency aliasing (which means signal corruption), the input signal band of a sample data system must be limited to the [0–Fs/2] range. The range [0–Fs/2] is called *baseband* and the above limitation is an expression of the Nyquist theorem.

After the sampling, the SC network processes the sequence of samples, independently of how they have been produced. Since all the frequencies given in Eq. 59.1 produce the same sequence of samples, the gain for all of them is the same. This concept results in the fact that the transfer function of a sampled-data system is periodical with period equal to Fs, and it is symmetrical in its period. For instance, in Fig. 59.3 the frequency response amplitude for a lowpass filter is shown for frequencies higher than Fs.

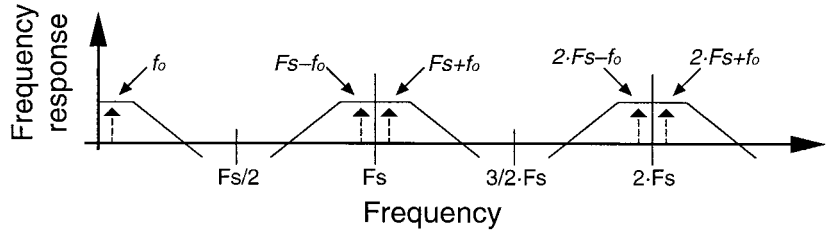


FIGURE 59.3 Periodicity of the sampled-data system transfer function.

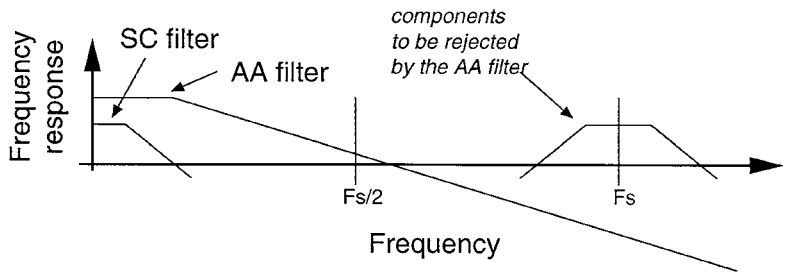


FIGURE 59.4 Transfer functions of swtiched-capacitor and anti-aliasing filters.

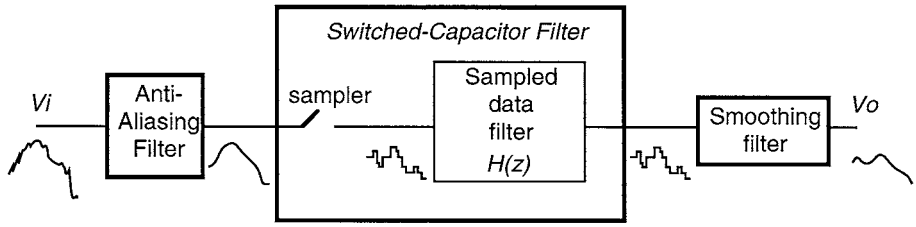


FIGURE 59.5 Overall SC filtering structure.

As stated above, in order to avoid the aliasing effect to corrupt the signal, it is necessary to limit the input signal bandwidth. This function is performed by the anti-aliasing (AA) filter, which is placed in front of the SC filter and operates in the continuous-time domain. From a practical point of view, the poles of the SC filter are typically much smaller than Fs/2, and only in the passband is the frequency response required to be accurate. On the other hand, the AA filter transfer function is not required to be accurate. Thus, the AA filter is usually implemented with active-RC filters (see Fig. 59.4).

At the output of the SC network, a staircase signal is produced. If a continuous-time output waveform is needed, a continuous-time smoothing filter must be added. The overall SC filter processing chain then results as shown in Fig. 59.5. Of course, in some cases, the input signal spectrum is already limited to Fs/2 and then the AAF filter is not necessary; while in other cases, the final smoothing filter is no longer necessary, like when the SC filter is used in front of a sampled-data system (like an ADC, for instance).

59.3 The Principle of the SC Technique

The principle of the SC technique consists in simulating the resistor behavior with a switched-capacitor structure. In the structure of Fig. 59.6, where an ideal op-amp is used, the resistor Req is connected between Vi and a zero-impedance zero-voltage node (as a virtual ground is). This means that a continuous-time current I flows from Vi, through Req, into the virtual ground. This current is equal to:

$$I = Vi/Req \qquad (59.2)$$

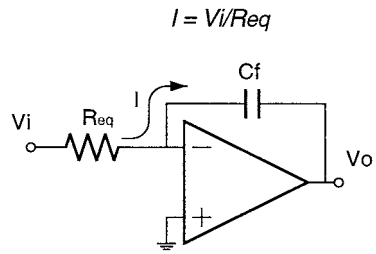


FIGURE 59.6 Basic RC integrator.

The alternative SC structure is shown in Fig. 59.7(a). It is composed of an input sampling capacitor CS (connected through four switches to the input signal VI, to the op-amp input node, and to two ground nodes), an op-amp, and a feedback (integrating) capacitor Cf. The clock phases driving the switches are shown in Fig. 59.7(b). A switch is closed (conductive) when its driving phase is high. It is

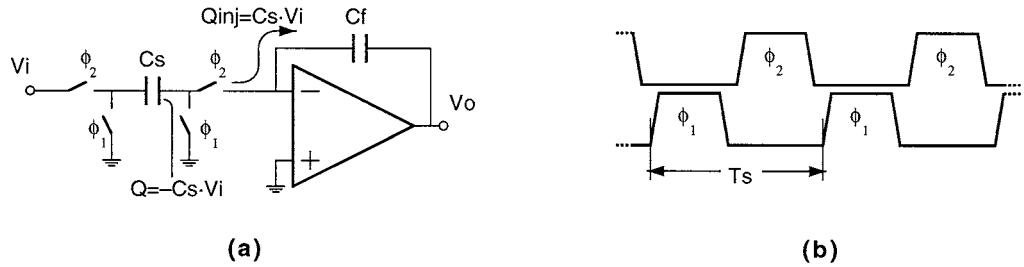


FIGURE 59.7 Basic SC integrator.

necessary that the two clock phases are non-overlapping, in order to connect each capacitor plate to only one low-impedance node for each time slot.

During phase ϕ1, capacitor Cs is discharged. During phase ϕ2, Cs is connected between Vi and the virtual ground. So, a charge Q = –Cs·Vi is collected on its right-hand plate. Due to the charge conservation law applied at the virtual ground node, this charge collection corresponds to an injection in virtual ground of the same amount of charge but with the opposite sign, given by:

$$Qinj = Cs \cdot Vi \tag{59.3}$$

Notice that this charge injection is independent of the component in the op-amp feedback path. This charge injection occurs every clock period. Observing this effect for a long time slot T, the total charge injection Qtot is given by:

$$Qtot = Cs \cdot Vi \cdot \frac{T}{Ts} \tag{59.4}$$

This corresponds to a mean current (Imean) equal to:

$$Imean = \frac{Qtot}{T} = Cs \cdot \frac{Vi}{Ts} \tag{59.5}$$

Equating Eq. 59.2 with Eq. 59.5, the following relationship holds:

$$Req = \frac{Ts}{Cs} \tag{59.6}$$

This means that the structure composed of Cs and the four switches operated at Fs is equivalent to a resistor Req. This approximation is valid for Vi equal to a dc-value, as it is the case of the proposed example, and it is still valid for Vi *slowly variable with respect to the clock period*; otherwise, the quantitiative average operation of Eq. 59.5 is no longer valid. The limits of the approximation between a resistor and an SC structure as expressed in Eq. 59.6 implies fundamental differences in the exact design of SC filters when derived from active-RC filters in a one-to-one correspondence.

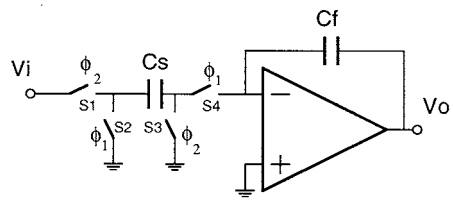
The synthesis of active filters is based on the use of some elementary blocks interconnected in different ways, depending on the type of adopted design philosophy. The different strategies and approaches for designing analog filters are well known from the continuous-time domain and they are also used in the case of SC filters, although the sampled-data nature of the SC filters can be profitably used either for simplifying or improving the design itself. In any case, basic building blocks are used to compose high-order filters.

In the following, the main basic blocks are described. They implement first-order (active integrators, undamped and damped, summers) and second-order (biquads) transfer functions in the *z*-domain (*z* is the state variable in the sampled data domain).

59.4 First-Order SC Stages

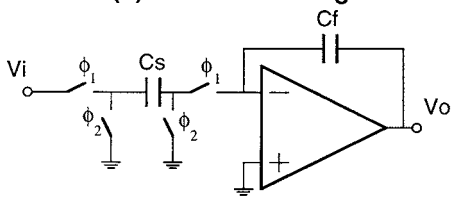
The Active SC Integrators

In Fig. 59.8(a-c), the standard integrators normally used in SC designs are shown. For each integrator, the transfer function in the z-domain is reported, assuming that the input signal is sampled during phase $\phi 1$ and is held to this value until the end of phase $\phi 2$, while the output is read during phase $\phi 2$.



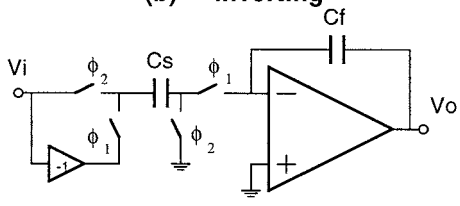
(a) - non-inverting

$$H_a(z) = \frac{V_o(z)}{V_i(z)} = \frac{Cs}{Cf} \cdot \frac{z^{-1}}{1 - z^{-1}} \tag{59.7a}$$



(b) - inverting

$$H_b(z) = \frac{V_o(z)}{V_i(z)} = -\frac{Cs}{Cf} \cdot \frac{1}{1 - z^{-1}} \tag{59.7b}$$



(c) - bilinear

$$H_c(z) = \frac{V_o(z)}{V_i(z)} = 2 \cdot \frac{Cs}{Cf} \cdot \frac{1 + z^{-1}}{1 - z^{-1}} \tag{59.7c}$$

FIGURE 59.8 SC integrators.

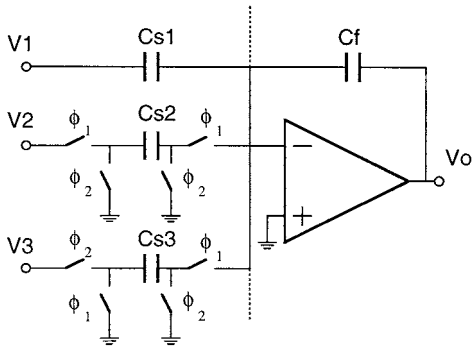
The third integrator is called bilinear since it implements the bilinear mapping of the s-to-z transformation (s is the state variable in the continuous-time domain).

In all the above transfer functions, only capacitor ratios appear. For this reason, the SC filter transfer functions are sensitive only to the capacitor ratios (i.e., to the capacitor matching) and are independent of absolute capacitor value. This is a remarkable advantage of all SC netowrks.

An important feature of all these integrators is their insensitivity to parasitic capacitance. This can be verified by observing that any parasitic capacitance connected to the capacitor left-hand plate is not connected to the virtual ground and therefore does not contribute to the amount of injected charge. On the other hand, the stray capacitance connected to the capacitor right-hand armature could contribute to the charge transfer, but this capacitance is switched between two nodes (ground and virtual ground) at the same potential and thus no charge injection results.

The Summing Integrator

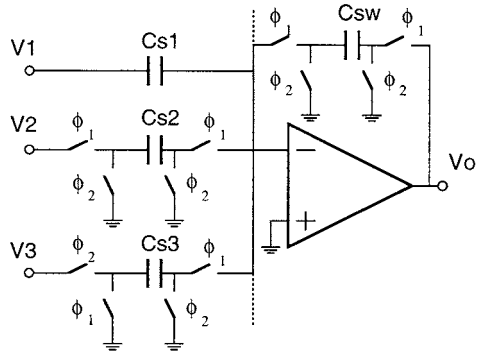
The SC operation is based on charge transfer. It is therefore easy to make weighted sum of multiple inputs by connecting different input branches to the same virtual ground. This concept is shown in the summing integrator of Fig. 59.9. The transfer function from the three input signals to the output is given in Eq. 59.8.



$$Vo = \frac{Cs1 \cdot \left(1 - z^{-1}\right) \cdot V1 - Cs2 \cdot V2 + Cs3 \cdot z^{-1} \cdot V3}{Cf \cdot \left(1 - z^{-1}\right)} \quad (59.8)$$

FIGURE 59.9 The SC summing integrator.

If the integrating feedback capacitor (Cf) is replaced by a feedback switched-capacitor (Csw), the structure does not maintain memory of its past evolution and a simple summing amplifier is obtained. The resulting structure is shown in Fig. 59.10, with the corresponding transfer function given in Eq. 59.9. This is the basic building block for the construction of SC filters implementing FIR frequency response.[4]

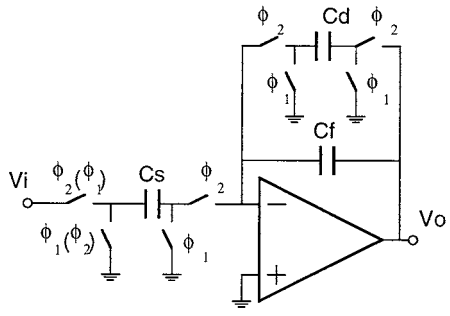


$$Vo = \frac{1}{Csw} \cdot \left(Cs1 \cdot \left(1 - z^{-1}\right) \cdot V1 - Cs2 \cdot V2 + Cs3 \cdot z^{-1} \cdot V3\right) \quad (59.9)$$

FIGURE 59.10 The SC summing amplifier.

The Active Damped SC Integrator

A damped integrator can be realized by connecting a damping switched capacitor (Cd) in parallel to the integrating capacitor (Cf), as shown in Fig. 59.11. Both inverting and non-inverting circuits are possible, depending on the type of input sampling structure. Equation 59.10a is valid for the clock phases out of parentheses, while Eq. 59.10b is valid for the clock phases within parentheses.



$$H_1(z) = \frac{Cd \cdot z^{-1}}{\left(Cd + Cf\right) - Cf \cdot z^{-1}} \quad (59.10a)$$

$$H_2(z) = -\frac{Cd}{\left(Cd + Cf\right) - Cf \cdot z^{-1}} \quad (59.10b)$$

FIGURE 59.11 Damped SC integrator.

A Design Example

As an example, the design of a damped SC integrator (Fig. 59.11) is given. A possible design approach is to derive the capacitor values of the SC structure from the R and C values in the equivalent continuous-time protptype, which is shown in Fig. 59.12, by the relationship of Eq. 59.6. It results that: Cs = Ts/Rs, and Cd = Ts/Rd. For instance, to have the pole frequency at 10 kHz with unitary dc-gain, a possible solution is: Rs = Rd = 159. 15 kΩ, and Cf = 10 pF. Using Fs = 1 MHz, it is obtained that Cs = Cd = 0.628 pF.

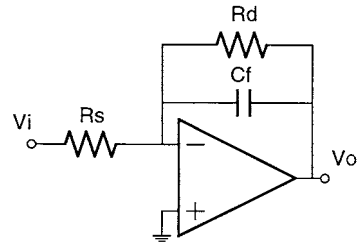


FIGURE 59.12 Continuous-time damped RC integrator.

The frequency response of the continuous-time and the SC damped integrator are shown in Fig. 59.13(a). Line I refers to the damped integrator of Fig. 59.11. Line II refers to a damped integrator with bilinear input branch (see Fig. 59.8(c)), while line III refers to the active-RC integrator of Fig. 59.12. In the passband, the frequency responses track very well. By increasing the input frequency, a difference becomes evident (as stated by the fact that Eq. 59.6 is valid for slowly variant signals). This is more pronounced if the frequency response is plotted up to 2·Fs (see Fig. 59.13(b)), where the periodic behavior of the sampled-data system frequency response is evident. Moreover, the key point of a sampled-data filter is the fact that the frequency response fixes the ratio between sampling-frequency and pole-frequency, i.e., with the above capacitor values, the pole frequency (fp) is 10 kHz for Fs = 1 MHz, while it decreases to 1 kHz if Fs = 100 kHz is used (i.e., the ratio fp/Fs remains constant). For this reason, the frequency response is plotted as a function of the normalized frequency f/Fs.

A limited stopband attenuation results for line I. This attenuation is improved using the bilinear input branch which implements a zero at Fs/2. This does not affect the frequency response in the passband, while a larger attenuation is obtained in the stopband (line II).

For the SC networks, the frequency response depends on capacitor ratios. Thus, the above extracted capacitor values can be normalized to the lowest one (which will be the unit capacitance). For this first-order cell example, the normalized capacitor values are: Cf = 15.92, and Cs = Cd = 1. The chosen value of the unit capacitance will not change the transfer function, while it will affect other filter features like die size, power consumption, and output noise.

From a general point of view, using Cf much larger than Cd corresponds to having a small damping impedance, as is the case in high-Q filters. This results in a large capacitor spread. On the other hand, to have a small time constant requires Cs much smaller than Cf; and also in this case, a large capacitor spread occurs. Since the unit capacitance cannot be smaller than a minimum technological value (to

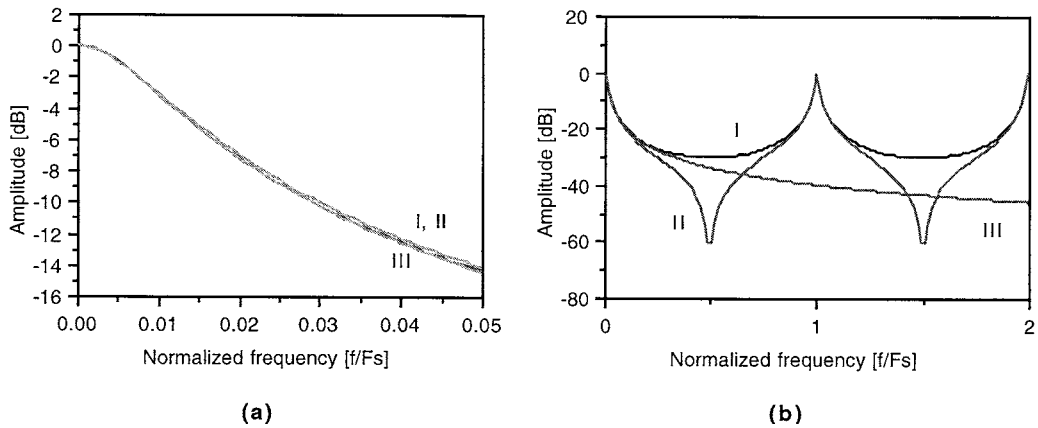


FIGURE 59.13 Frequency response comparison for different integrators.

achieve a certain matching accuracy), a large capacitor spread means to have a large capacitor to be driven and thus a large power to the op-amp to operate. In addition, a large chip area is needed. Thus, in order to avoid large capacitor spread, possible solutions will be proposed in the follwing.

59.5 Second-Order SC Circuit

The general expression of a second-order (biquadratic) z-transfer function can be written in the form:

$$H(z)=\frac{\gamma+\varepsilon\cdot z^{-1}+\delta\cdot z^{-2}}{1+\alpha\cdot z^{-1}+\beta\cdot z^{-2}} \tag{59.11}$$

The denominator coefficients (α, β) fix the pole frequency and quality factor, while the numerator coefficients (γ, ε, δ) define the types of filter frequency response. Several SC biquadratic cells have been proposed in literature to implement the above transfer function. In the following two of them are presented: the Fleischer & Laker one and another one useful for high sampling frequency.

The Fleischer & Laker Biquad

A popular biquadratic cell, proposed by Fleischer & Laker,[5] is shown in Fig. 59.14 in its most general form. The cell is composed by:

- an input branch (capacitor G, H, I, and J) which allows to select the zero positions, and therefore the kind of frequency response;
- a resonating loop (capacitor A, B, C, and D in conjuntion with the damping capacitors E, and F) which sets the pole frequency and the pole quality factor. The two damping capacitors are not usually adopted together. In general E-capacitor is used for high-Q filters, while F-capacitor is preferred for low-Q circuits.

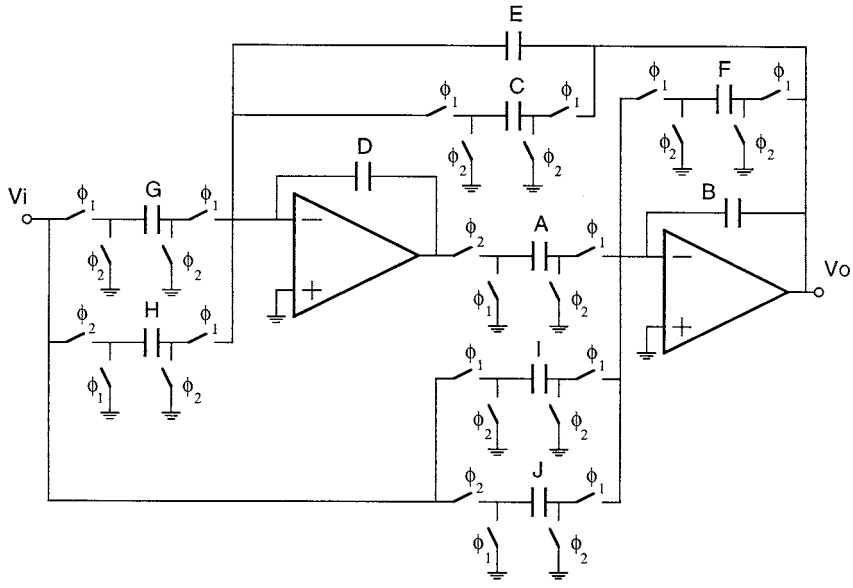


FIGURE 59.14 Fleischer & Laker biquadratic cell.

The transfer function of the Fleischer & Laker biquad cell can be written as:

$$H(z)=\frac{Vo}{Vi}=-\frac{D\cdot I+\left(A\cdot G-D\cdot\left(I+J\right)\right)z^{-1}+\left(J\cdot D-A\cdot H\right)z^{-2}}{D\cdot\left(B+F\right)-\left(D\cdot\left(2\cdot B+F\right)-A\cdot\left(C+E\right)\right)z^{-1}+\left(D\cdot B-A\cdot E\right)z^{-2}} \tag{59.12}$$

This cell allows one to synthetize any kind of transfer function by using parasitic insensitive structures, which ensures performance accuracy. The key observation related to this biquad cell is that the first op-amp operates in cascade to the second one; therefore, its settling is longer than the second one is.

Design Methodology

At the first order, the SC circuits can be derived from continuous-time circuits implementing the desired frequency response, by a proper substitution of each resistor with the equivalent SC structures, as shown for the first-order cell. An alternative approach is to optimize the transfer function in the *z*-domain, and to numerically fit the desired transfer function with the transfer function implemented by the second-order cell. A third possibility is to adapt the *s*-domain transfer function to the *z*-domain signal processing of the SC structures. This procedure will be used in the following.

Consider the case of a given transfer function in *s*-domain, for instance, when an approximation table (Butterworth, Chebichev, Bessel, etc.) is used. The *s*-domain transfer function to be implemented is written as:

$$H(s)=\frac{a_2\cdot s^2+a_1\cdot s+a_0}{s^2+b_1\cdot s+b_0} \tag{59.13}$$

This *s*-domain transfer function is transformed into a *z*-domain transfer function through the use of the bilinear *s*-to-*z* transformation:

$$s=\frac{2}{Ts}\cdot\frac{1-z^{-1}}{1+z^{-1}} \tag{59.14}$$

This transformation produces a warping of the frequency of interest. To avoid this error, a characteristic frequency for the given design (i.e., the frequencies ω_i of interest for the final filter mask) should be 'prewarped' according to the relationship between the angular frequencies in the *s*-domain (Ω_i) and in the *z*-domain (ω_i):

$$\Omega_i=\frac{2}{Ts}\cdot\tan\left(\frac{\omega_i\,Ts}{2}\right) \tag{59.15}$$

as it is indicated in Fig. 59.15 for a bandpass-type response.

The characteristic frequency can be the –3dB frequency for a low-pass filter. The H'(s) that will satisfy the 'prewarped' filter mask will be automatically transformed by Eq. 59.14 into a *z*-domain transfer function whose frequency response satisfies the desired filter mask in the ω-domain. Obviously, if $\omega_i\cdot Ts \ll 1$, no predistortion is needed, being $\Omega_i \approx \omega_i$. Assuming that $\omega_{i\cdot Ts} \ll 1$, H′(s) ≈H(s) and Eq. 59.14 is substituted directly into Eq. 59.13; the resulting coefficients of terms z^{-i} in the denominator are then equated to the corresponding ones in Eq. 59.12. Assuming A = B = D = 1, the capacitor values for the E-type and F-type can be extracted as follows:

$$\mathrm{F}=0 \qquad E=\frac{b_1Ts}{1+\frac{b_1Ts}{2}+\frac{b_0Ts^2}{4}} \qquad C=\frac{b_0Ts^2}{1+\frac{b_1Ts}{2}+\frac{b_0Ts^2}{4}} \tag{59.16a}$$

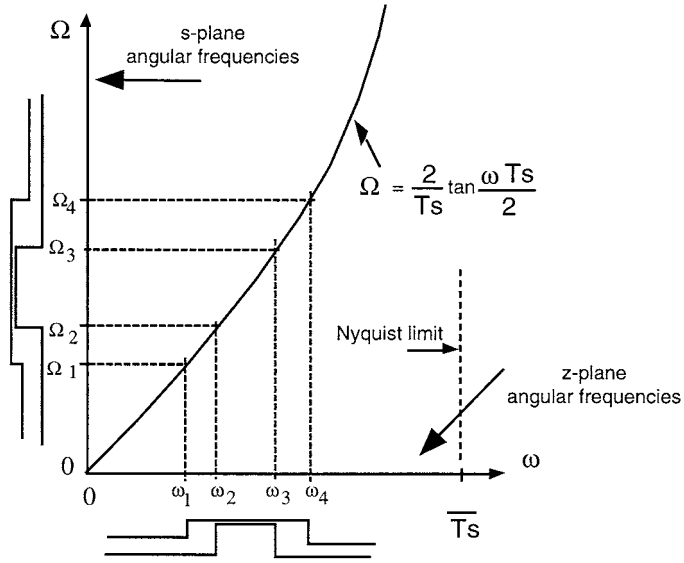


FIGURE 59.15 Bilinear mapping between continuous-time (Ω) and sampled-data (ω) frequency.

$$\mathrm{E}=0 \qquad E=\frac{b_1 Ts}{1-\frac{b_1 Ts}{2}+\frac{b_0 Ts^2}{4}} \qquad C=\frac{b_0 Ts^2}{1-\frac{b_1 Ts}{2}+\frac{b_0 Ts^2}{4}} \tag{59.16b}$$

When the bilinear transform is used, a second-order numerator is obtained, which can be written as given in Eq. 59.17, where simple solutions for the input capacitors are also given.

$$\text{Lowpass} \qquad K\left(1+z^{-1}\right)^2 \qquad I=J=IKI \qquad G=4IKI;\ H=0 \tag{59.17a}$$

$$\text{Bandpass} \qquad K\left(1+z^{-1}\right)\cdot\left(1-z^{-1}\right) \qquad I=G=H=IKI;\quad J=0 \tag{59.17b}$$

$$\text{Highpass} \qquad K\left(1-z^{-1}\right)^2 \qquad I=J=IKI \qquad G=H=0 \tag{59.17c}$$

Design Example

As design example for the second-order cell, a bandpass response with Q = 5, *fo* = 20 kHz, and *Fs* = 1 MHz is considered (i.e., *fo*/*Fs* = 0.02). The transfer functions in the *s*-domain and in the *z*-domain (using bilinear transformation) are the following:

$$H(s)=\frac{s}{s^2+\frac{2\cdot\pi\cdot fo}{Q}\cdot s+\left(2\cdot\pi\cdot fo\right)^2}=\frac{2.5133\cdot 10^4\cdot s}{s^2+2.5133\cdot 10^4\cdot s+1.5791\cdot 10^{10}} \tag{59.18}$$

$$H(z)=\frac{z^{-2}-1}{z^{-2}-1.9719\cdot z^{-1}+0.98755} \tag{59.19}$$

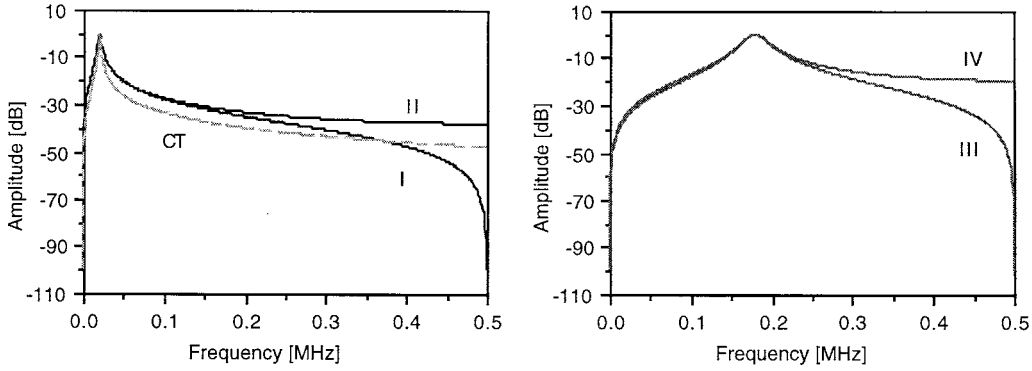


FIGURE 59.16 Frequency responses for different designs.

No frequency has been prewarped, since, applying Eq. 59.15 to *fo*, the prewarped pole frequency should result 19.765 kHz, with a negligible deviation of about 0.1%.

For the bandpass response, using the bilinear *s*-to-*z* mapping, the zero positions are at $\{z = 1, z = -1\}$. The frequency response is shown in Fig. 59.16 with line I. The normalized capacitance value, obtained equating the transfer function of Eq. 59.19 with the transfer function of the biquadratic cell of Eq. 59.12, are given in Table 59.1, for the E-type and F-type structures, in column I. A very large capacitor spread (>78) is needed. This results in large die area, and large power consumption. The capacitor spread could be reduced with a slight modification of the transfer function to the one given in the following:

$$H(z) = \frac{z^{-1} - 1}{z^{-2} - 1.9719 \cdot z^{-1} + 0.98755} \tag{59.20}$$

With respect to the bilinear transformation of Eq. 59.9, in this case, the zero at dc is maintained, while the zero at Nyquist frequency (at Fs/2, i.e., at z = –1) is eliminated. The normalized capacitor values are indicated again in Table 59.1, in Column II. It can be seen that a large reduction of the capacitor spread is obtained (from 80 to 8, for the E-type). The obtained frequency response is reported in Fig. 59.16 with line II. In the passband no significant changes occur; on the other hand in the stopband, the maximum signal attenuation is about –35 dB. In some applications, this solution is acceptable, also in consideration of the considerable capacitor spread reduction. For this reason, if not strictly necessary, the zero at Fs/2 can be eliminated. However reducing the factor fo/Fs results in reducing the stopband attenuation. For instance, for *fo* = 200 kHz (i.e., *fo*/Fs = 0.2), the frequency response with and without the Nyquist zero are reported in Fig. 59.16 with line III and IV, respectively. In this case, the stopband attenuation is reduced to –22 dB and therefore the Nyquist zero could be strongly needed. The relative normalized capacitor values are indicated in Table 59.1 in Column III (with zeros at $\{z = 1, z = -1\}$), and in Column IV (with zeros at $z = 1$).

A Biquadratic Cell for High Sampling Frequency

In the previous biquadratic cell, the two op-amps operate in cascade during the same clock phase. This requires that the second op-amp in cascade wait for the complete settling of the first op-amp to complete its settling. This, of course, reduces the maximum achievable sampling frequency, or, alternatively for a given sampling frequency increases the required power consumption since op-amps with larger bandwidth are needed. In order to avoid this aspect, the biquad shown in Fig. 59.17 can be used. In this scheme, the two op-amps settle in different clock phase and thus they have the full clock phase time slot to settle.

The transfer function of the biquadratic cell is given in Eq. 59.21. As it can be seen a limitation occurs in the possible transfer function, since the term in z^{-2} is not present.

TABLE 59.1 Capacitor Values for Different Designs

E-type	I	II	III	IV
A	10.131	1	12.366	1.0690
B	80.885	7.963	12.094	1.0000
C	1.2565	1	12.561	7.4235
D	10.131	8.0838	12.366	7.6405
E	1.9998	1.5915	1.9991	1.1815
F	0	0	0	0
G	1	1.5885	1	1
H	1	1.5885	1	1
I	1	0	1	0
J	0	0	0	0
F-type	**I**	**II**	**III**	**IV**
A	10.006	5.1148	11.305	5.9919
B	78.885	39.446	10.095	5.0497
C	1.2565	1	12.561	7.4235
D	10.006	8.1404	11.305	7.0793
E	0	0	0	0
F	1.9998	1	1.9991	1
G	1	1.5885	1	1
H	1	1.5885	1	1
I	1	0	1	0
J	0	0	0	0

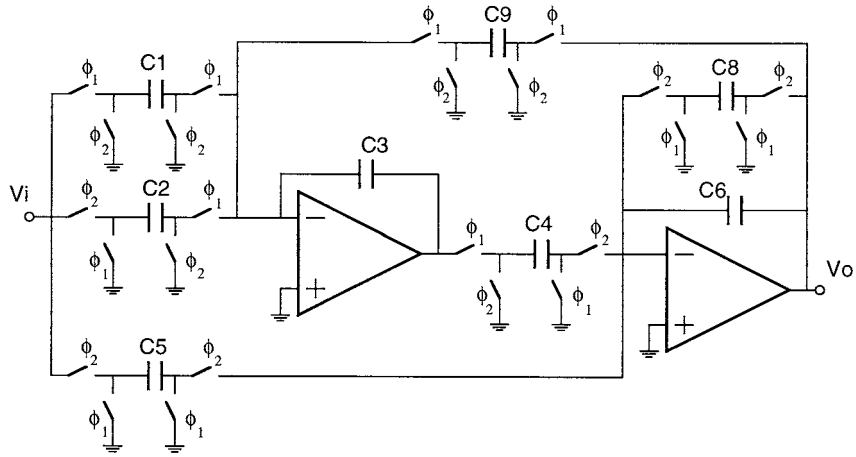


FIGURE 59.17 High-frequency biquadratic cell.

$$H(z) = -\frac{C3\cdot C5 + C1\cdot C4 - \left(C3\cdot C5 - C2\cdot C4\right) z^{-1}}{C3\cdot\left(C8 + C6\right) + \left(C4\cdot C9 - C3\cdot C8 - 2\cdot C3\cdot C6\right)\cdot z^{-1} + C3\cdot C6\cdot z^{-2}} \tag{59.21}$$

High-Order Filters

The previous first- and second-order cells can be used to build up high-order filters. The main architectures are taken from the theory for the active RC filters. Some of the most significant ones are: ladder[6] (with good amplitude response robustness with respect to component spread), cascade of first- and second-order cells (with good phase response robustness with respect to component spread), follow-the-leader feedback (for low-noise systems, like reconstruction filters in oversampled DAC).

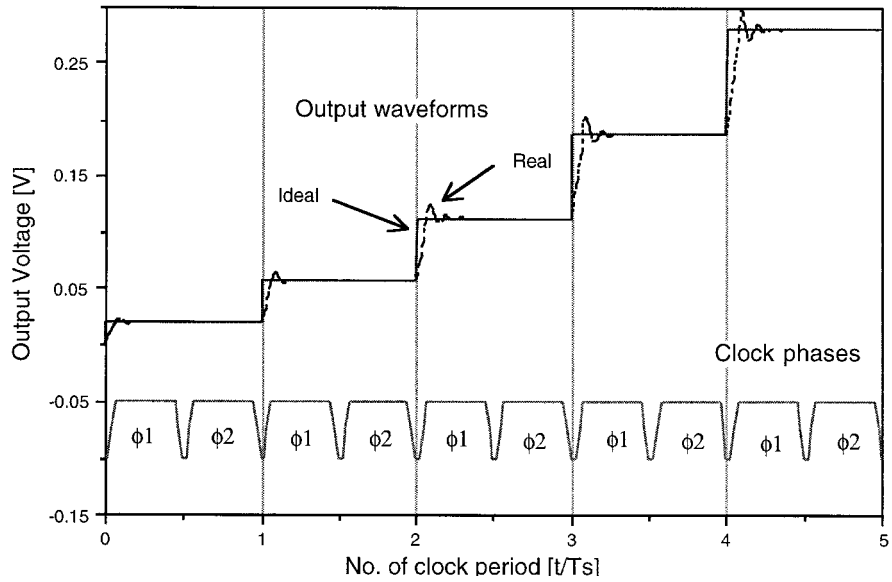


FIGURE 59.18 Output waveform evolution.

59.6 Implementation Aspects

The arguments presented so far have to be implemented in actual integrated circuits. Such implementations have to minimize the effects of the non-idealities of the actual blocks, which are capacitors, switches, and op-amps. The capacitor behavior is quite stable, apart from capacitance non-linearities which affect the circuit performance only as a second-order effect.

On the other hand, switches and op-amps must be properly designed to operate in the SC system. The switches must guarantee a minimum conductance to ensure a complete charge transfer within the available timeslot. For the same reason, the op-amps must ensure large-dc gain, large unity-gain bandwidth, and large slew rate. For instance, in Fig. 59.18, the ideal output waveform of an SC network is shown with a solid line, while the more realistic actual waveform is illustrated with the dotted line. The output sample is updated during phase $\phi 1$, while it is held (at the value achieved at the end of phase $\phi 1$ during phase $\phi 2$. In phase $\phi 1$, the output value moves from its initial to its final value. The slowness of this movement is affected by switch conductance, op-amp slew rate, and op-amp bandwidth.

The transient response of the system can be studied using the linear model of Fig. 59.19, where the conductive switches are replaced by their on-resistance Ron and the impulsive charge injection is replaced by a voltage step. The assumption of a complete linear system should allow one to exactly study the system evolution. In this case, the circuit time-constants depend on input branch ($\tau_{in} = 2 \cdot Ron \cdot Cs$), op-amp frequency response, and feedback factor.

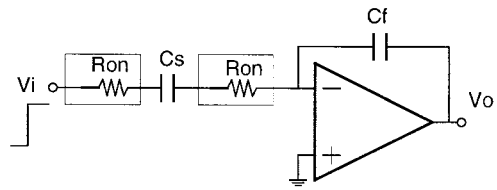


FIGURE 59.19 Linear model for transient analysis.

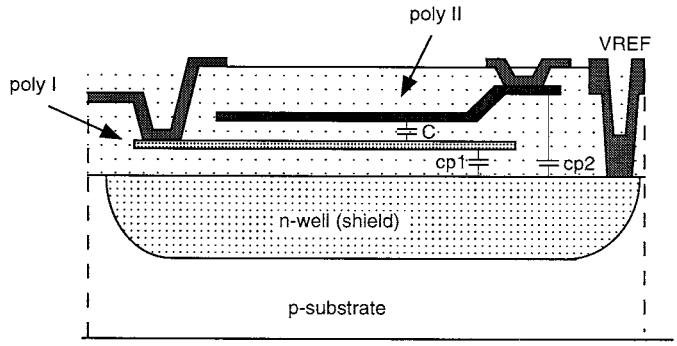


FIGURE 59.20 Poly1-poly2 capacitor cross-section.

Non-linear analysis is, however, necessary when op-amp slew rate occurs. This analysis is difficult to carry out and optimum performance can be achieved using computer simulations. Usually, for typical device models, 10% of the available timeslot (i.e., Ts/2) is used for slew rate, while 40% is used for linear settling.

Integrated Capacitors

Integrated capacitors in CMOS technology for SC circuits are mainly realized using poly1-poly2 structure, whose cross-section is shown in Fig. 59.20. This capacitor implementation guarantees linear behavior over a large signal swing. The main drawbacks of integrated capacitors are related to their absolute and relative inaccuracy, and to their associated parasitic capacitance.

The absolute value of integrated capacitors can change ±30% from their nominal values. However, the matching between equal capacitors can be on the order of 0.2%, provided that proper layout solutions are adopted (in close proximity, with guard rings, with common centroid structure). The matching of two capacitors of different value C can be expressed with the standard deviation of their ratio σ_C, which is correlated with the standard deviation of the ratio between two identical capacitors σ_{C1} by Eq. 59.22.[7]

$$\sigma_C = \frac{\sigma_{C1}}{\sqrt{C/C1}} \tag{59.22}$$

This model can be used to evaluate the robustness of the SC system performance with respect to random capacitor variations using a Monte Carlo analysis.

The plates of poly1-poly2 capacitor of value C present a parasitic capacitance toward the substrate, as shown in Fig. 59.18. Typically, this capacitance is about 10% of C for the bottom plate (cp1 = C/10), and it is 1% of C for the top plate (cp2 = C/100). In order to reduce the effect of these parasitic capacitances in the transfer function of the SC systems, it is useful to connect the top plate to the op-amp input node, and the bottom plate to low impedance nodes (op-amp output nodes or voltage sources). In addition, in Fig. 59.20, an n-well, biased with a clean voltage VREF; is placed under the poly1-poly2 capacitor in order to reduce noise coupling from the substrate, through parasitic capacitance.

MOS Switches

The typical situation during sampling operation is shown in Fig. 59.21(a) (this is the input branch of the integrator of Fig. 59.7(a)). The input signal Vi is sampled on the sampling capacitor Cs in order to have Vc = Vi.

FIGURE 59.21 (a) ideal sampling structure, (b) sampling structure with NMOS switches.

In Fig. 59.21(b), the switches are replaced by a single-nMOS device which operates in the triode region with an approximately zero voltage drop between drain and source. The switch on-resistance Ron can be expressed as:

$$Ron = \frac{1}{\mu n \cdot Cox \cdot \frac{W}{L} \cdot \left(V_{gs} - V_{th}\right)} = \frac{1}{\mu n \cdot Cox \cdot \frac{W}{L} \cdot \left(V_G - Vi - V_{th}\right)} \tag{59.23}$$

where V_G is the amplitude of the clock driving phase, μn is the electron mobility, Cox is the oxide capacitance, and W and L are the width and length of the MOS device. Using V_{DD} = 5 V (i.e., V_G = 5 V), the dependence of Ron on the input voltage is plotted in Fig. 59.22(a). This means that if Ron is required by the capacitor value be lower than a given value (to implement a low Ron·Cs time constant), a limitation in the possible input swing is given. For instance, if the maximum possible Ron is 2.5 kΩ, the maximum input signal swing is [0 V–3.5 V].

To avoid this limitation, a complementary switch can be used. It consists of an NMOS and a PMOS device in parallel, as shown in Fig. 59.23. The PMOS switch presents a Ron behavior complementary to that of the NMOS, as plotted in Fig. 59.22(b). The complete switch Ron is then given by the parallel of the two contributions which is sufficiently low for all the signal swing.

Using this solution requires one to distribute double clock lines controlling the NMOS and the PMOS. This could be critical for SC filters operating at high-sampling frequency, also in consideration of the synchronization of the two phases and of the digital noise from the distributed clocks which could reduce the dynamic range.

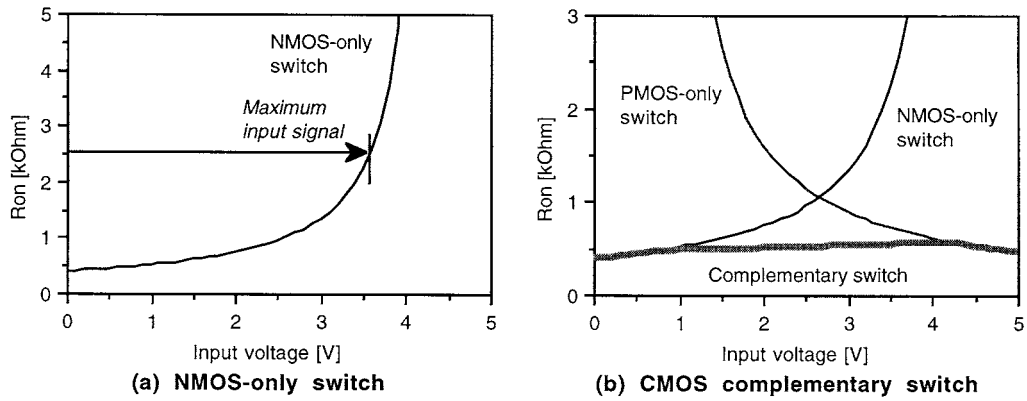


FIGURE 59.22 Switch on-resistance.

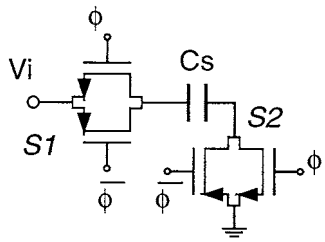


FIGURE 59.23 Sampling structure with complementary switches.

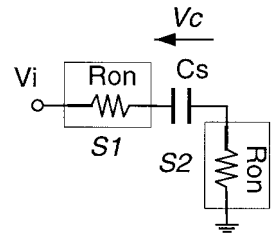


FIGURE 59.24 Sampling operation linear model.

Once a minimum conductance is guaranteed, the structure can be studied using the linear model for the MOS devices S1 and S2 which operate in the triode region, resulting in the circuit of Fig. 59.24. In this case, Vc follows Vi, through an exponential law with a time constant τ_{in} = Cs·2·Ron. Typically, at least 6·τ_{in} must be guaranteed in the sampling timeslot to ensure sufficient accuracy. For a given sampling capacitance value, this is achieved using switches with sufficiently low on-resistance and no voltage drop across its nodes. Large on-resistance results in a long time constant and incomplete settling, while a voltage drop results in an incorrect final value. MOS technology allows the implementation of analog switches satisfying both the previous requirements.

Transconductance Amplifier

The SC technique appears the natural application of available CMOS technology design features. This is true also for the case of the op-amp design. In fact, SC circuits require an infinite input op-amp impedance, as in the case of op-amp using a MOS input device. On the other hand, CMOS op-amps are particularly efficient when the load impedance is not resistive and low, but only capacitive, as in the case of SC circuits. In addition, SC circuits allow one to process a full swing (rail-to-rail) signal and this is possible for CMOS op-amps. The main requirements to be satisfied by the op-amp remain the bandwidth, the slew rate, and the dc-gain.

The bandwidth and the slew rate must be sufficiently large to guarantee accurate settling for all the signal steps. The op-amp gain must be sufficiently large to ensure a complete charge transfer. A tradeoff between large dc-gain (achieved with low-current and/or multistage structure) and large bandwidth (obtained at high-current and/or simple structure) must be optimized. For this case, the use of mixed technology (like BiCMOS) could help the proper design optimization.

59.7 Performance Limitations

The arguments described thus far are valid assuming an ideal behavior of the devices in the SC network (i.e., op-amp, switches, and capacitor). However, in actual realization, each of them presents non-idealities which reduce the performance accuracy of the complete SC circuit. The main limitations and their effects are described in the following. Finally, considerations about noise in SC systems conclude this section.

Limitation Due to the Switches

As described before, CMOS switches satisfy both low on-resistance and zero voltage-drop requirements. However, they introduce some performance limitations due to their intrinsic CMOS realization. The cross-section of an NMOS switch in its on-state is shown in Fig. 59.25. The connection between its nodes N1 and N2 is guaranteed by the presence of the channel, made up of the charge Qch. The amount of charge Qch can be written as:

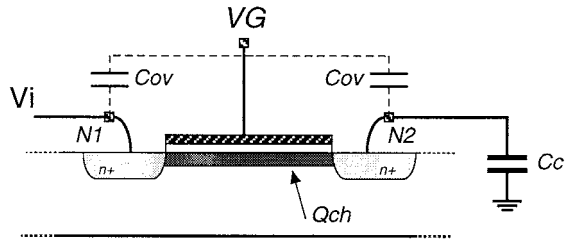


FIGURE 59.25 Switch charge profile of an NMOS switch in the on-state.

$$Qch = \left(W \cdot L\right) \cdot Cox \cdot \left(V_G - Vi - V_{TH}\right) \tag{59.24}$$

where Vi is the channel (input) voltage. Both nodes N1 and N2 are at voltage Vi (no voltage drop between the switch nodes). In addition, the gate oxide which guarantees infinite MOS input impedance constitutes a capacitive connection between gate and both source and drain. This situation results in two non-ideal effects: charge injection and clock feedthrough.

Charge Injection

At the switch turn-off, the charge Qch given in Eq. 59.24 is removed from the channel and it is shared between the two nodes connected to the switch, with a partition depending on the node impedance level.

The charge k·Qch is injected in N2 and collected on a capacitor Cc. A voltage variation nVc across the capacitor arises, which is given by:

$$\Delta Vc = k \cdot \frac{Qch}{Cc} \tag{59.25}$$

For all the switches of a typical SC integrator, as shown in Fig. 59.26(a), this effect is important. For instance, for the switch S4 connected to the op-amp virtual ground, the charge injection into the virtual ground is collected in the feedback capacitor and it is processed as an input signal. The amount of this charge injection depends on different parameters (see Eq. 59.24 and Eq. 59.25). Charge Qch depends on switch size W, which however cannot be reduced beyond a certain level; otherwise, the switch on-resistance should increase. Thus, a tradeoff between charge injection and on-resistance is present. In addition, charge Qch depends on the voltage Vi which the switch is connected to. For the switches S2, S3, and S4, the injected charge is proportional to (V_G – Vgnd) and is always fixed; as a consequence, it can be considered like an offset. On the other hand, for the switch S1 connected to the signal swing, the channel charge Qch is dependent on (V_G – Vi), i.e., on the signal amplitude and thus also the charge injection is signal dependent. This creates an additional signal distortion.

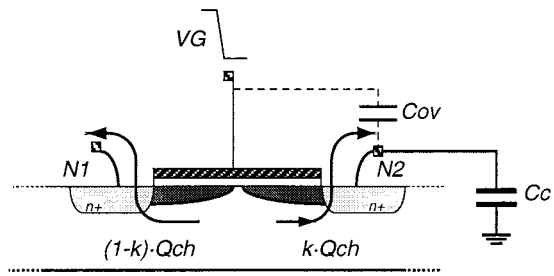


FIGURE 59.26 Charge displacement during turn-off.

Possible solutions for the reduction of the charge injection are: use of dummy switches, use of slowly variable clock phase, use of differential structures, use of delayed clock phases,[8] and use of signal-dependent charge pump.[9]

Dummy switches operate with complementary phases in order to sink the charge rejected by the original switches. The use of differential structures reduces the offset charge injection to the mismatch of the two differential paths. For the signal-dependent charge injection, the delayed phases of Fig. 59.26(b) are applied to the integrator of Fig. 59.26(a). This clock phasing is based on the concept that at the turn-off, S3 is open before S1. In such a way, when S1 opens, the impedance toward Cs is infinite and no signal-dependent charge injection occurs into Cs.

Clock Feedthrough

The clock feedthrough is the amount of signal that is injected in the sampling capacitor Cc from the clock phase through the MOS overlap capacitor (Cov) path, shown in Fig. 59.27, which is then proportional to the area of the switches. Using large switches, to reduce on-resistance, results in large charge injection and large clock feedthrough. This error is typically constant (it depends from capacitance partition) and therefore it can be greatly reduced by using differential structures. The voltage error ΔVc across a capacitance Cc due to the feedthrough of the clock amplitude ($V_{DD} - V_{SS}$) can be written as:

$$\Delta Vc = \left(V_{DD} - V_{SS}\right) \cdot \frac{Cov}{Cov + Cc} \tag{59.26}$$

Limitation Due to the Op-amp

The operation of SC networks is based on the availability of a "good" virtual ground which ensures a complete charge transfer from the sampling capacitors to the feedback capacitor. Whenever this charge transfer is uncompleted, the SC network performance derives from its nominal behavior. The main non-ideality causes from the op-amp are: finite dc-gain, finite bandwidth, finite slew-rate, and gain non-linearity.

Finite Op-amp dc-Gain Effects[10,11]

The op-amp finite gain results in a deviation of output voltage at the end of the sampling period from the ideal one, as shown in Fig. 59.28(a). This output sample deviation can be translated in the SC system performance deviation. For the finite gain effect, an analysis which correlates the op-amp gain Ao with SC network performance deviation, can be carried out under the hypothesis that the op-amp bandwidth is sufficiently large to settle within the available timeslot.

For the case of the summing amplifier of Fig. 59.10, it can be demonstrated that the effect of the finite op-amp dc-gain (AO) is only in an overall gain error. For this reason SC FIR filters (based on this scheme) exhibit a low sensitivity to op-amp finite dc-gain. On the other hand, for the case of SC integrators, the

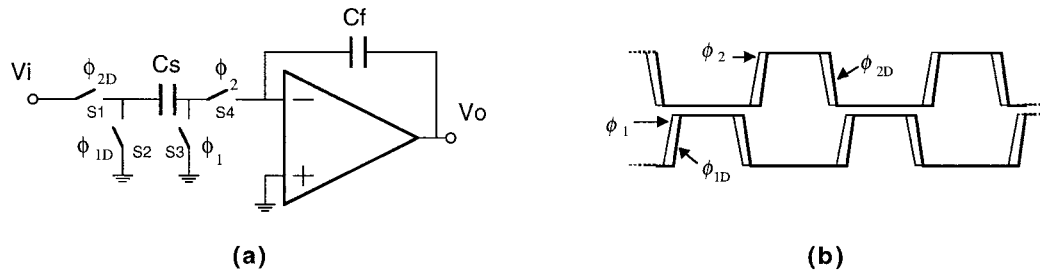


FIGURE 59.27 Clocking scheme for signal-dependent charge injection reduction.

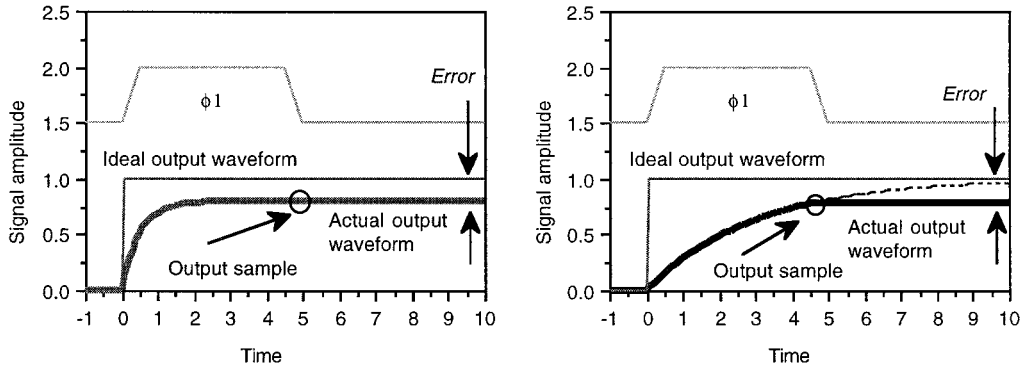


FIGURE 59.28 Op-amp induced errors.

finite gain effect results in pole and gain deviation. For instance, the transfer function of the integrator of Fig. 59.8(a) becomes:

$$H_a(z)=\frac{V_o(z)}{V_i(z)}=\frac{Cs}{Cf}\cdot\frac{z^{-1}}{1+\frac{1}{Ao}\cdot\left(1+\frac{Cs}{Cf}\right)-\left(1+\frac{1}{Ao}\right)\cdot z^{-1}} \tag{59.27}$$

For a biquadratic cell, the op-amp finite gain results in pole frequency, pole quality factor deviations. The actual frequency and quality factor of the pole (f_oA and Q_A) are correlated to their nominal values (fo and Q) by the relationship:

$$f_{oA}=\frac{Ao}{1+Ao}\cdot f_o \qquad Q_A=\frac{1}{\frac{1}{Q}+\frac{2}{Ao}}\approx\left(1-\frac{2\cdot Q}{Ao}\right)\cdot Q \tag{59.28}$$

Finite Bandwidth and Slew-Rate[10,11]

Also, op-amp finite bandwidth and slew-rate result in incomplete charge transfer, which still corresponds with deviation of the output sample with respect to its nominal value. For the case of only finite bandwidth, the effect is shown in Fig. 59.28(b). An analysis similar to that of the finite gain for the finite bandwidth and slew-rate effect is not easily extracted. This is due to the fact that incomplete settling is caused by the correlation of a linear effect (e.g., the finite bandwidth) and a non-linear effect (e.g., the slew rate). In addition, this case is worsened by the fact that in some structures, several op-amps are connected in cascade and then each op-amp (a part of the first one) has to wait for the operation conclusion of the preceding one.

Op-amp Gain Non-linearity

Since the SC structure allows one to process large swing signals, for this signal swing, the op-amp has to perform constant gain. When the op-amp gain is not constant for all the necessary output swing, distortion arises. An analysis can be carried out for the case of the integrator of Fig. 59.8(a).[12] Assuming an op-amp input (vi)-to-output (vo) relationship expressed in the form:

$$vo=a_1\cdot vi+a_2\cdot vi^2+a_3\cdot vi^3+\ldots \tag{59.29}$$

The resulting harmonic components (for $\omega o \cdot Ts \ll 1$) are given by:

$$HD2 = \frac{a_2}{2 \cdot a_1^3 \beta} \quad V_o \sqrt{1 + \left(\frac{Vo}{2 \cdot Vi} \right)^2} \tag{59.30a}$$

$$HD3 = \frac{a_3}{2 \cdot a_1^4 \beta} \quad V_o^2 \sqrt{1 + \left(\frac{Vo}{3 \cdot Vi} \right)} \tag{59.30b}$$

The distortion can then be reduced or making constant low-gain (i.e., reducing a_2 and a_3) or using a very large op-amp gain (i.e., increasing a_1). This second case is usually the adopted strategy.

Noise in SC Systems[13,14]

In SC circuits, the main noise sources are in the switches (thermal noise) and in the op-amp (thermal noise and 1/*f* noise). These noise sources are processed by the SC structure as an input signal, i.e., they are sampled (with consequent folding) and transferred to the output with a given transfer function. As explained for the signal, the output frequency range of an SC filter is limited in the band [0–Fs/2]. This means that for any noise source its sampled noise band, independently on how large it is at the source before sampling, is limited in the [0–Fs/2] range. On the other hand, the total power of noise remains constant after sampling; this means that the power density of sampled noise is increased by the factor Fb/(Fs/2), where Fb is the noise band at its source. This can be seen for the switch noise in the following simple example. Consider the structure of Fig. 59.29, where the resistance represents the switch on-resistance. Its associated noise spectral density is given by $v_n^2 = 4kT \cdot Ron$ (where *k* is the Boltzmann's constant and T is the absolute temperature). The transfer function to the output node (the voltage over the capacitor) is $H(s) = \frac{1}{1 + s \cdot Ron \cdot Cs}$. The total output noise can be calculated from the expression:

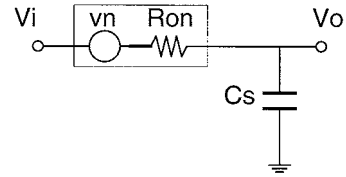


FIGURE 59.29 Sampling the noise on a capacitor.

$$n_o^2 = \int_0^\infty v_n^2 \cdot \left| H(s)^2 \right| \cdot df = \frac{kT}{Cs} \tag{59.31}$$

The total sampled noise is then given by *k*T/Cs, and presents a bandwidth of Fs/2. This means that the output noise power density is *k*T/Cs·2/Fs. (See Fig. 59.30.)

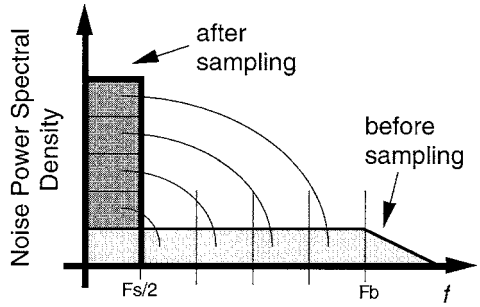


FIGURE 59.30 Folding of the noise power spectral density.

The same folding concept can be applied to the op-amp noise. For the op-amp 1/*f* noise, the corner frequency is usually lower than Fs/2. Therefore, the 1/f noise is not modified by the sampling. On the other hand, the white noise presents a bandwidth Fb larger than Fs/2. This means that this noise component is modified in a noise source of bandwidth Fs/2 and noise power density multiplied by the factor Fb/(Fs/2). When the noise sources are evaluated in this way, the output noise of an SC cell can be evaluated by summing the different components properly weighted by their transfer functions from their source position to the output node.

A few considerations follow for the noise performance of an SC cell. Switch noise is independent of Ron, since its dependence is cancelled in the bandwidth dependence. Thus, this noise source is dependent only on Cs. Noise reduction is achieved by increasing Cs. This, however, trades with the power increase necessary to drive the enlarged capacitance. Of course, even if Ron does not appear in the noise expression, as the capacitor is enlarged, the Ron must be adequately decreased in order to guarantee a proper sampling accuracy.

For the op-amp noise, the noise band is usually correlated with the signal bandwidth. Therefore, a good op-amp settling (achieved with a large signal bandwidth) is in contrast to low-noise performance (achieved with reduced noise bandwidth). Therefore, in low-noise systems, the bandwidth of the op-amp is designed to be the minimum that guarantees proper settling.

59.8 Compensation Technique (Performance Improvements)

SC systems usually operate with a two-phase clock in which the op-amp is 'really' active only during one phase, and during the other phase is 'sleeping.' Provided that the op-amp output node is not read during the second phase, this non-active phase could be used to improve the performance of the SC system, as shown in the following.[15] 1/f noise and offset can be reduced with Correlated Double Sampling (CDS) or the chopper technique. Similar structures are also able to compensate for the error due to a finite gain of the operational amplifier. On the other hand, proper structures are able to reduce the capacitor spread occurring in particular situations (high-Q or large time constant filters). Finally, the double-sampled-technique can be used to increase, by a factor of two, the sampling frequency of the SC system.

CDS Offset-Compensated SC Integrator

The extra phase available in a two-phase SC system can be used to reduce op-amp offset and 1/f noise effects at the output. A possible scheme is shown in Fig. 59.31,[16] and operates as follows. Capacitor Cof is used to sample, during $\phi 1$, the offset voltage Voff as it appears in the inverting node of the op-amp with close to unitary feedback. During $\phi 2$, the inverting node is still at a voltage very close to Voff, since

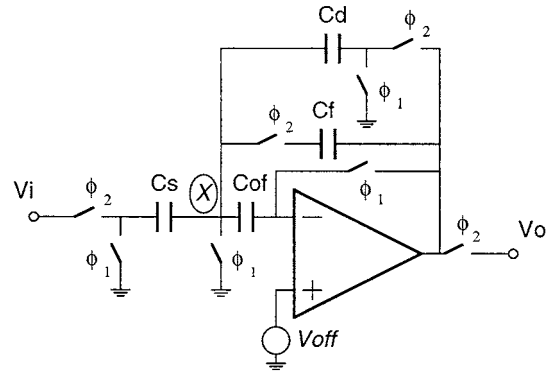


FIGURE 59.31 Offset-compensated SC integrator.

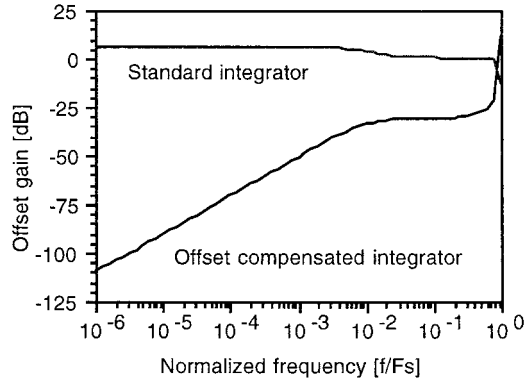


FIGURE 59.32 Offset-compensated SC integrator performance.

the bandwidth of Voff is assumed to be very small with respect to the sampling frequency. Capacitor Cof maintains the charge on its armatures and acts like a battery. Thus, node X is a good virtual ground, independent on the op-amp offset. In the same way, the output signal, read only during $\phi 2$, is offset independent. The effect of this technique can be simulated using the value of the first-order cell of the previous example (i.e., Cf = 15.92, and Cs = Cd = 1), and Cof = 1. The transfer function Vo/Voff is shown in Fig. 59.32. At low frequency, the Voff is highly rejected, while this is not the case of the standard (uncompensated) integrator. The main problem with this solution is due to the unity feedback operation of the structure during phase $\phi 1$. This requires the stability of the op-amp, which could require a very high power consumption.

Chopper Technique

An alternative solution to reduce offset 1/f noise at the output is given by the chopper technique. It consists of placing one SC mixer for frequency Fs/2 at the op-amp input and a similar one at the op-amp output. This action does not affect white noise. On the other hand, offset and 1/f noise are shifted to around Fs/2, not affecting the frequencies around dc, where the signal to be processed is supposed to be.

This concept is shown for a fully differential op-amp in Fig. 59.33. In Fig.59.34, the input-referred noise power spectral density (PSD) without and with chopper modulation is shown. The white noise level (wnl) is not affected by the chopper operation and remains constant. It will be modified by the folding of the high-frequency noise, as previously described.

This technique is particularly advantageous for SC systems since the mixer can be efficiently implemented with the SC technique as shown in Fig. 59.35.

Finite-Gain Compensated SC Integrator

In the op-amp design, a tradeoff between op-amp dc-gain and bandwidth exists. Therefore, when a large bandwidth is needed, a finite dc-gain necessarily occurs, reducing SC filter performance accuracy. To

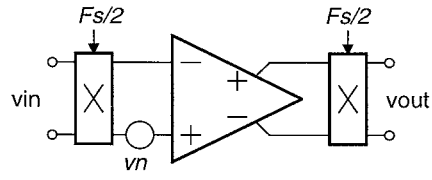


FIGURE 59.33 Op-amp with chopper.

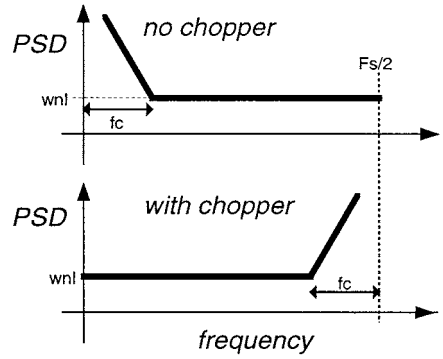


FIGURE 59.34 Input-referred noise power spectral density (PSD) without and with chopper technique.

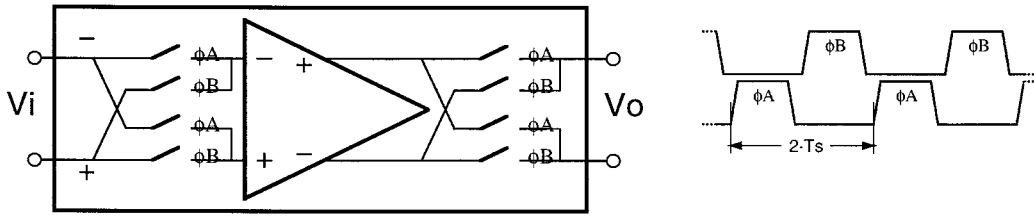


FIGURE 59.35 Op-amp with SC chopper configuration.

avoid this, the available extra phase can be used to self-calibrate the structure with respect to the error due to the op-amp finite gain. In the literature, several techniques have been proposed. The majority of them are based on the concept of using a preview of the future output samples to pre-charge a capacitor placed in series to the op-amp inverting input node in order to create a 'good' virtual ground (as for offset cancellation). The various approaches differ on how they get the preview and how they calibrate the new virtual ground. For the different cases, they can be effective for a large bandwidth,[17,18] for a small bandwidth,[19,20] or for a passband bandwidth.[21] As an example of this kind of compensation, one of the earliest proposed schemes is depicted in Fig. 59.36.

The op-amp finite gain makes the op-amp inverting input node different from the virtual ground ideal behavior and assume the value –Vo/Ao, where Vo is the output value and Ao is the op-amp dc-gain. In the scheme of Fig. 59.36, the future output sample is assumed to be close to the previous sample, sampled of Cg1. This limits the effectiveness of this scheme to signal frequencies f for which this

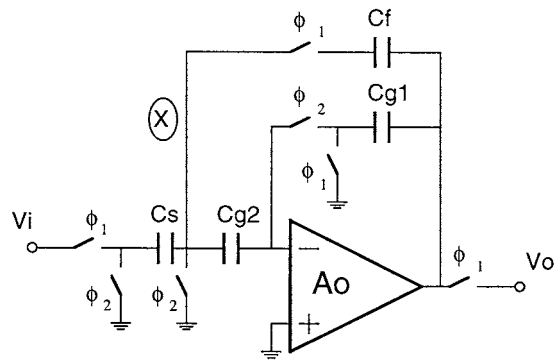


FIGURE 59.36 Finite-gain-compensated SC integrator.

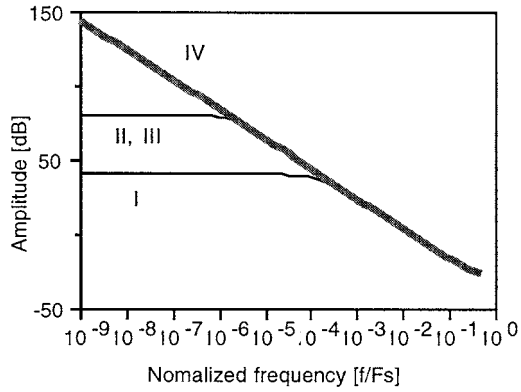


FIGURE 59.37 Finite-gain-compensated SC integrator performance.

assumption is valid (i.e., for *f*/Fs ≪ 1). The circuit operates as follows. During ϕ1, auxiliary capacitor Cg1 samples the output; while during ϕ2, Cg1 is used to precharge Cg2 to –Vo/Ao, generating a good virtual ground at node X.

In Fig. 59.37, the frequency response of different integrators are compared. Line I refers to an uncompensated integrator with Ao = 100; line II refers to the uncompensated integrator with Ao = 10,000. This line matches with line III, which corresponds to the compensated integrator with Ao = 100. Finally, line IV is the frequency response of the ideal integrator. From this comparison, the compensation effect is to achieve an op-amp gain Ao performance similar to those achieved with an op-amp gain Ao^2.

Alternative solution to the op-amp gain compensation are based on the use of a replica amplifier matched with the main one. Also in this way the effectiveness of the solution is to achieve performance accuracy relative to an op-amp dc-gain of Ao^2.

The Very-Long Time-Constant Integrator

In the design of Very-Long Time-Constant integrators using the scheme of Fig. 59.8(a), typical key points to be considered are:

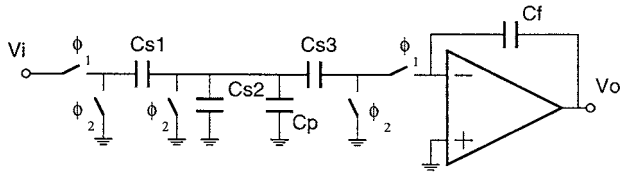
- The capacitor spread: if the pole frequency *fp* is very low with respect to the sampling frequency Fs, then the capacitor spread S = Cf/Cs of a standard integrator (Fig. 59.8(a)) will be very large. This results in large die area and reduce performance accuracy for poor matching.
- The sensitivity to the parasitic capacitances: proper structure can reduce capacitor spread. They, however, suffer from the presence of parasitic capacitance. Parasitic-insensitive or at least parasitic-compensated designs should then be considered.
- The offset of the operational amplifier: offset-compensated op-amps are needed when the op-amp offset contribution cannot be tolerated.

In the literature, several SC solutions have been proposed, more oriented toward reducing the capacitor spread than toward compensating either the parasitics or the op-amp offset.

A first solution is based on the use of a capacitive T-network in a standard SC integrator, as shown in Fig. 59.38.[22] The operation of the sampling T-structure is to realize a passive charge partition with the capacitors Cs1, and Cs2+Cs3. The final result is that only the charge on Cs3 is injected into the virtual ground. Therefore, the effect of this scheme is that Cs is replaced with the Cs_equiv, given by the expression:

$$Cs\_equiv = Cs3 \cdot \frac{Cs1}{Cs1 + Cs2 + Cs3} \tag{59.32}$$

The net gain of this approach is that, using Cs2 = $\sqrt{S}$·Cs1 = $\sqrt{S}$·Cs3, the capacitor spread is reduced to $\sqrt{S}$. For example, an integrator with Cs = 1 and Cf = 40, can be realized with Cs1 = 1, Cs2 = 6, Cs3 = 1, and Cf = 5, i.e., with the capacitor spread reduced to 6.

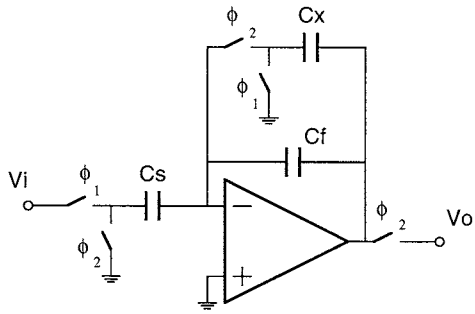


$$\frac{Vo}{Vi} = \frac{Cs1}{Cs1+Cs2+Cs3} \cdot \frac{Cs3}{Cf} \cdot \frac{1}{1-z^{-1}} \tag{59.33}$$

FIGURE 59.38 A T-network long-time-constant SC integrator.

The major problem of the circuit of Fig. 59.38 is due to the fact that the T-network is sensitive to the parasitic capacitance Cp (due to Cs1, Cs2, and Cs3) in the middle node of the T-network, which is added to Cs2, reducing frequency response accuracy.

A parasitic-insensitive circuit is the one proposed by Nararaj[23] and shown in Fig. 59.39. In this case, the transfer function is given by Eq. 59.34. Also, in this case, Cs = Cx = $\sqrt{S}$·Cf are usually adopted to reduce the standard spread from S to $\sqrt{S}$. However, for the Nagaraj integrator, the op-amp is used on both phases, disabling the possibility of using double-sampled structure.



$$\frac{Vo}{Vi} = \frac{Cf}{Cs} \cdot \frac{Cx}{Cf+Cx} \cdot \frac{1}{1-z^{-1}} \tag{59.34}$$

FIGURE 59.39 The Nagaraj's long-time-constant SC integrator.

It is also possible to combine a long-time-constant scheme with an offset-compensated scheme to obtain a long-time-constant offset-compensated SC integrator.[15]

Double-Sampling Technique

If the output value of the SC integrator of Fig. 59.8(b) is read only at the end of ϕ2, the requirement for the op-amp to settle can be relaxed. For the integrator of Fig. 59.8(b), the time available for the op-amp to settle is $T_s/2$. The equivalent double-sampled structure is shown in Fig. 59.40. The capacitor values for the two structures are the same, and thus they implement the same transfer function. The time evolution for the two structures are compared in Fig. 59.41. For the double-sampled SC integrator, the time available for the op-amp to settle is doubled.

This advantage can be used in two ways. First, when a high sampling frequency is required, if the op-amp cannot settle in Ts/2, the extra time allows it to reach the speed requirement (i.e., the double-sampling technique is used to increase the sampling frequency). Second, at low sampling frequency when the power consumption must be strongly reduced, a smaller bandwidth guaranteed by the op-amp reduces its power consumption.

The cost of the double-sampled structure is the doubling of all the switched capacitors. In addition, in the case of a small mismatch between the two parallel paths, mismatch energy could be present around Fs/4.[24]

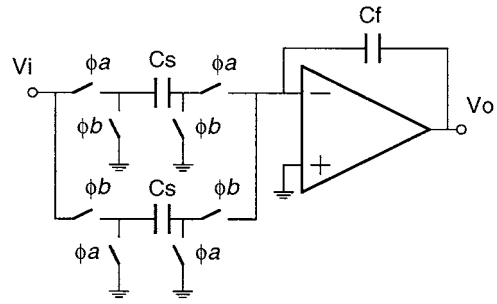


FIGURE 59.40 Double-sampled SC integrator.

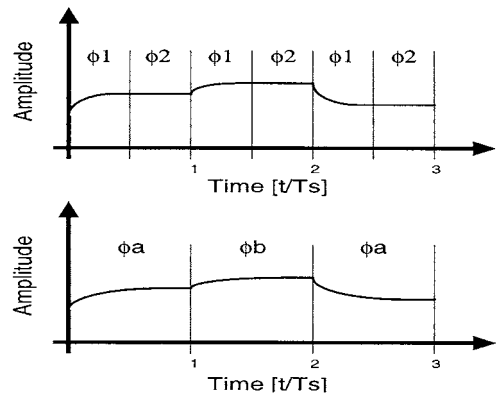


FIGURE 59.41 Double-sampled operation.

59.9 Advanced SC Filter Solutions

In this section, alternative solutions able to overcome some basic limitations to SC system performance improvement are proposed. They deal with the tradeoff between bandwidth-vs.-gain in the op-amp design for high-frequency SC filter, and the implementation of low-voltage SC filters.

Precise Op-amp Gain (POG) for High-Speed SC Structures[25,26]

Standard design of SC networks assumes operation with infinite gain and infinite bandwidth op-amps. However, in the op-amp design, a tradeoff exists between the speed and the gain. As a consequence with a high sampling frequency, the needed large bandwidth limits the op-amp gain to low values, thus limiting the achievable accuracy. Thus, standard design is less feasible for high-sampling frequency. A possible solution to this limitation is the Precise Op-amp Gain (POG) design approach, which consists of designing high-frequency SC networks, taking into account the precise gain value of the op-amps as a parameter in the capacitor design. The standard design op-amp tradeoff between *speed-and-gain* is then changed into the POG design tradeoff between *speed-and-gain precision*, which is more affordable in high-frequency op-amps.

If the op-amp dc-gain is Ao, the transfer function of the first-order cell shown in Fig. 59.42 is given by:

$$H_{POG}(z) = -\frac{Cs_{POG}}{\left(Cf_{POG} + Cd_{POG} + \dfrac{Cs_{POG} + Cf_{POG} + Cd_{POG}}{Ao}\right) - Cf_{POG}\left(1 + \dfrac{1}{Ao}\right)z^{-1}} \tag{59.35}$$

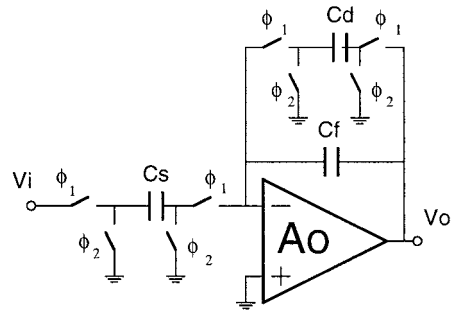


FIGURE 59.42 Dumped SC integrator.

This expression for Ao equal to infinite becomes the standard case t.f HST, which is given by:

$$H_{ST}(z) = -\frac{Cs}{(Cf + Cd) - Cf\ z^{-1}} \tag{59.36}$$

To obtain the same t.f., the POG capacitor values are obtained from the standard values as given in the following:

$$Cs_{POG} = Cs \qquad Cf_{POG} = Cf \cdot \left(1 + \frac{1}{Ao}\right) \qquad Cd_{POG} = Cd\left(1 + \frac{1}{Ao}\right) + \frac{Cs}{Ao} \tag{59.37}$$

The concept here applied to the SC integrator can be applied to higher-order SC filters. It can be demonstrated that using the POG approach with an op-amp nominal gain Ao and an actual gain in the range [Ao(1 – ε), Ao(1 + ε)] achieves the same response accuracy as the standard approach with an infinite op-amp nominal gain with an actual gain given by:

$$A_{eff} = \frac{(Ao+1)(1 \pm \varepsilon)}{\varepsilon} \approx \frac{Ao}{\varepsilon} \tag{59.38}$$

The value A_{eff} can then be defined as the effective gain of the POG approach. For example, for the op-amp with Ao = 100 and ε = 0.08, the same performance accuracy is achieved as when using standard design with op-amp gain A = 1250. This value of A_{eff} can then be used in Eq. 59.27 to evaluate filter performance accuracy.

Low-Voltage Switched-Capacitor Solutions

In the last few years, the interest in low-power, low-voltage integrated systems has consistently grown due to the increasing importance of portable equipment and to the reduction of the supply voltage of modern standard CMOS scaled-down technology ICs. For the design of SC filters operating at reduced supply voltages,[27,28] capacitor properties are quite stable. On the other hand, at low supply, it is difficult to properly operate the MOS switches and the op-amps. With the supply voltage reduction, the MOS switches' overdrive voltage is lowered, inhibiting proper operation of classical transmission gate (complementary switches). The switch conductance for different input voltages changes, depending on the supply voltage V_{DD}. In Fig. 59.22, the case for V_{DD} = 5 V was shown. In Fig. 59.43, the case for V_{DD} = 1 V is reported for comparison. In this case, there is a critical voltage region centered around $V_{DD}/2$ for which both switches are not conducting. In SC circuits, to achieve rail-to-rail swing, the output of the op-amp

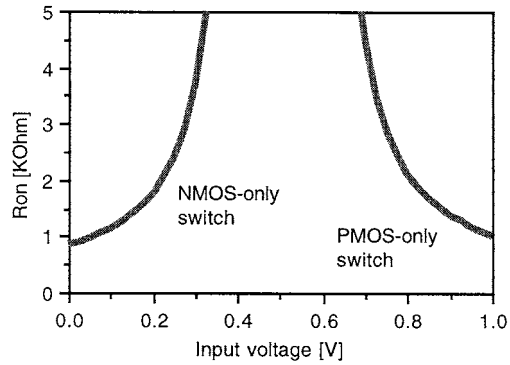


FIGURE 59.43 Switch Ron with V_{DD} = 1 V.

must necessarily cross this critical region, where the switches connected to the op-amp output node will not properly operate at V_{DD} = 1 V.

On the other hand, op-amp operation can be achieved with proper design using a supply voltage as low as $V_{TH}+2\cdot V_{OV}$, with some modifications at system level. Switches and op-amp sections may use different supply voltages. In fact, using voltage multiplier (a possible scheme is shown in Fig. 59.44[29]), it is possible to generate on-chip a voltage higher than the supply voltage. This "multiplied" voltage can then be used to power the entire SC circuit (op-amp and switches) or to drive only the switches.

If the higher supply is used to bias the op-amp and switches, standard design solutions can be implemented. In addition, the op-amp powered with a higher supply voltage can manage a larger signal swing, with a consequential larger dynamic range. However, in a scaled-down technology, the maximum acceptable electric field between gate and channel (for gate oxide breakdown) and between drain and source (for hot electron damage) must be reduced. This puts an absolute limit on the value of the multiplied supply voltage. In addition, the need to supply a dc-curent to the op-amp from the multiplied supply forces one must use an external capacitor, which is an additional cost.

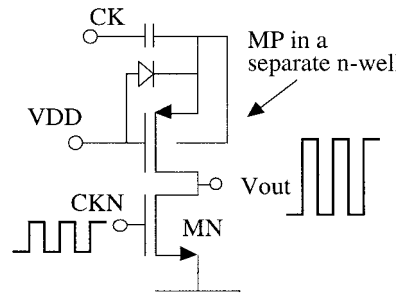


FIGURE 59.44 Charge-pump for on-chip voltage multiplication.

An alternative approach consists of using the multiplied supply to drive only the switches. In this case, the voltage multiplier does not supply any dc-current, thus avoiding any external capacitor. This solution, like the previous one, must not exceed the limit of the technology associated with the gate oxide breakdown. Nonetheless, this approach is largely used because it allows the filter to operate at high sampling frequency.

In order to avoid any kind of voltage multiplier, the Switched-OpAmp (SOA) approach was proposed[30,31] and is based on the following considerations:

1. The best condition for the switches driven with a low supply voltage is to be connected either to ground or to V_{DD}. Thus, to properly operate, S2, S3, and S4 in Fig. 59.8(a) have to be referred to ground or to V_{DD} (i.e., the op-amp input dc-voltage has to be either ground or V_{DD}). This allows one to minimize the required op-amp supply voltage. On the other hand, the op-amp dc output voltage has to be at $V_{DD}/2$ in order to have rail-to-rail output swing.
2. The switch S1 connected to the signal swing cannot operate properly for the full signal swing, as explained before.

The resulting SOA SC integrator is shown in Fig. 59.45. The SOA approach uses an op-amp, which can operate in a tri-state mode. In this way, the critical output switch S1 is no longer necessary and it can be eliminated, moving the critical problem to the op-amp design. The function of the eliminated

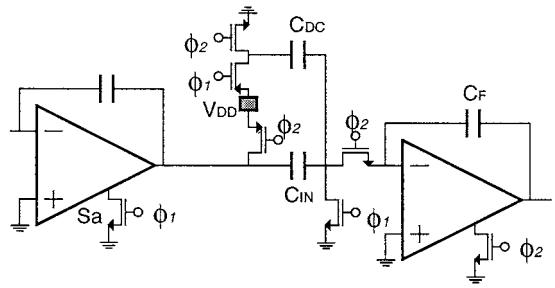


FIGURE 59.45 Switched-op-amp SC integrator.

critical switch S1 is implemented by turning on and off the op-amp through Sa, which is connected to ground.

The input dc-voltage is set to ground. Therefore, all the switches are connected to ground (and realized with a single NMOS device) or to V_{DD} (and realized with a PMOS device), and are driven with the maximum overdrive, given by $V_{DD} - V_{TH}$. Capacitor C_{DC} in Fig. 59.45 gives a fixed charge injection into virtual ground, producing a voltage level shift between input (V_{in_DC}) and output (V_{out_DC}) op-amp dc-voltage. Since V_{in_DC} is set to ground, using $C_{DC} = C_{IN}/2$ sets $V_{out\_DC} = V_{DD}/2$. C_{DC} has no effect on the signal transfer function given by:

$$H(z) = \frac{C_{IN} \cdot z^{-1/2}}{C_F \cdot (1 - z^{-1})} \tag{59.39}$$

A fully differential architecture of the scheme of Fig. 59.45 provides both signal polarities at each node, useful to build up high-order structures without any extra elements (e.g., inverting stage). In addition, any disturbance (offset or noise) injected by C_{DC} results in a common-mode signal, which is largely rejected by the fully differential operation.

The SOA approach suffers from the following problems:

1. SOA structure operates with an op-amp, which is turned on and off. Its turn-on time becomes the main limitation in the possible maximum sampling frequency.
2. In an SOA structure, the output signal is available only during one clock phase; while during the other clock phase, the output is set to zero (return-to-zero), as shown in Fig. 59.46. If the output signal is read as a continuous-time waveform, the return to zero has two effects: a loss of 6 dB in the transfer function, and an increased distortion due to the large output steps. On the other

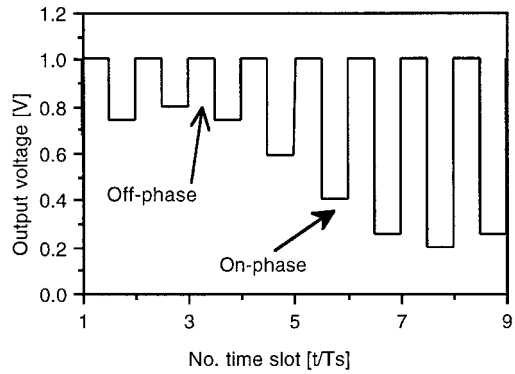


FIGURE 59.46 Switched-op-amp output waveform.

hand, when the SOA integrator is used in front of a sampled-data system (like an ADC), the output signal is sampled only when it is valid and both the above problems are cancelled.

A comparison between the three different low-voltage SC filter design approaches is given in Table 59.2.

TABLE 59.2 Comparison Between Different Low-voltage SC Designs

		Supply multiplier	Clock multiplier	Switched-op-amp
$V_{DDswith}$		V_{DDmult}	V_{DDmult}	V_{DD}
$V_{DDop\text{-}amp}$		V_{DDmult}	V_{DD}	V_{DD}
New op-amp design		No	No/Yes	Yes
New switch design		No	No	Yes
Output swing		+	–	–
Gate Break-down (V_{GS} limit)		–	–	+
Hot electron (V_{DS} limit)		–	+	+
Sampling frequency limitation		+	+	–
Power consumption		–	–	+
External component		Yes	No	No
Continuous waveform:	Gain loss	1	1	1/2
	Return-to-zero distortion	+	+	—

References

The literature about SC filters is so wide that any list of referred publications should be considered incomplete. In the following, just few papers related to the discussed topics are indicated.

1. B. J. Hosticka, R. W. Brodersen, and P. R. Gray, MOS sampled-data recursive filters using switched-capacitor integrators, *IEEE J. Solid-State Circuits,* vol. SC-12, pp. 600-608, Dec. 1997.
2. J. T. Caves, M. A. Copeland, C. F. Rahim, and S. D. Rosenbaum, Sampled analog filtering using switched capacitors as resistor equivalents, *IEEE J. Solid-State Circuits,* vol. SC-12, pp. 592-599, Dec. 1977.
3. R. Gregorian and G. C. Temes, *Analog MOS Integrated Circuits for Signal Processing,* John Wiley & Sons, 1986.
4. G. T. Uehara, and P. R. Gray, A 100MHz output rate analog-to-digital interface for PRML magnetic disk read channels in 1.2μ CMOS, *IEEE Int. Solid State Circ. Conf. 1994, Digest of Tech. Papers,* pp. 280-281, 1994.
5. P. E. Fleischer and K. R. Laker, A family of active switched-capacitor biquad building blocks, *Bell Syst. Tech. J.,* vol. 58, pp. 2235-2269, 1979.
6. G. M. Jacobs, D. J. Allstot, R. W. Brodersen, and P. R. Gray, Design techniques for MOS switched-capacitor ladder filters, *IEEE Trans on Circ. and Syst,* vol. CAS-25, pp. 1014-1021, Dec. 1978.
7. J. B. Shyu, G. C. Temes, and F. Krummenacher, Random error effects in matched MOS capacitors and current sources, *IEEE J. of Solid-State Circuits,* vol. SC-19, pp. 948-955, 1984.
8. D. G. Haigh and B. Singh, A switching scheme for switched-capacitor filters which reduces the effects of parasitic capacitances associated with switch control terminals, *IEEE Intern. Symp. on Circ. and Systems,* ISCAS 1993.
9. T. Brooks, D. H. Robertson, D. F. Kelly, A. DelMuro, and S. W. Harston, A cascaded sigma-delta pipeline A/D converter with 1.25MHz signal bandwidth and 89dB SNR, *IEEE J. of Solid-State Circuits,* vol. SC-32, pp. 1896-1906, Dec. 1997.
10. G. C. Temes, Finite amplifier gain and bandwidth effects in switched capacitor filters, *IEEE J. Solid-State Circuits,* vol. SC-15, pp. 358-361, June 1980.
11. K. Martin and A. S. Sedra, Effects of the op amp finite gain and bandwidth on the performance of switched-capacitor filters, *IEEE Trans. Circ. Syst.,* vol. CAS-28, no. 8, pp. 822-829, Aug. 1981.

12. K. Lee and R. G. Meyer, Low-distortion switched-capacitor filter design techniques, *IEEE Journal of Solid State Circuits,* Dec. 1985.
13. C. A. Gobet and A. Knob, Noise analysis of switched-capacitor networks, *IEEE Trans. on Circ. and Systems,* vol. CAS-30, pp. 37-43, Jan. 1983.
14. J. H. Fischer, Noise sources and calculation techniques for switched-capacitor filters, *IEEE Journal of Solid State Circuits,* vol. SC-17, pp. 742-752, Aug. 1982.
15. C. Enz and G. C. Temes, Circuit techniques for reducing the effects of opamp imperfections: autozeroing, correlated double sampling, and chopper stabilization, *Proceedings of IEEE,* vol. 84, no. 11, pp. 1584-1614, Nov. 1996.
16. K. K. K. Lam and M. A. Copeland, Noise-cancelling switched-capacitor (SC) filtering technique, *Electronics Letters,* vol. 19, pp. 810-811, Sept. 1983.
17. K. Nagaraj, T. R. Viswanathan, K. Singhal, and J. Vlach, Switched-capacitor circuits with reduced sensitivity to amplifier gain, *IEEE Trans. on Circuits and Systems,* vol. CAS-34, pp. 571-574, May 1987.
18. L. E. Larson and G. C. Temes, Switched-capacitor building-blocks with reduced sensitivity to finite amplifier gain, bandwidth, and offset voltage, *IEEE International Symposium on Circuits and Systems (ISCAS '87),* pp. 334-338, 1987.
19. K. Haug, F. Maloberti, and G. C. Temes, Switched-capacitor integrators with low finite-gain sensitivity, *Electronics Letters,* vol. 21, pp. 1156-1157, Nov. 1985.
20. K. Nagaraj, J. Vlach, T. R. Viswanathan, and K. Singhal, Switched-capacitor integrator with reduced sensitivity to amplifier gain, *Electronics Letters,* vol. 22, pp. 1103-1105, Oct. 1986.
21. A. Baschirotto, R. Castello, and F. Montecchi, Finite gain compensated double-sampled switched-capacitor integrator for high-Q bandpass filters, *IEEE Trans. Circuits Syst.,* vol. CAS-I 39, no. 6, June 1992.
22. T. Huo and D. J. Allstot, MOS SC highpass/notch ladder filter, *Proc. IEEE Int. Symp. Circ. Syst.,* pp. 309-312, May 1980.
23. K. Nagaraj, A parasitic insensitive area efficient approach to realizing very large time constant in switched-capacitor circuits, *IEEE Trans. on Circuits and Systems,* vol. CAS-36, pp. 1210-1216, Sept. 1989.
24. J. J. F. Rijns and H. Wallinga, Spectral analysis of double-sampling switched-capacitor filters, *IEEE Trans. on Circ. and Systems,* vol. 38, no. 11, pp. 1269-1279, Nov. 1991.
25. A. Baschirotto, F. Montecchi, and R. Castello, A 15MHz 20mW BiCMOS switched-capacitor biquad operating with 150Ms/s sampling frequency, *IEEE Journal of Solid State Circuits,* pp. 1357-1366, Dec. 1995.
26. A. Baschirotto, Considerations for the design of switched-capacitor circuits using precise-gain operational amplifiers, *IEEE Transaction on Circuits and Systems. II.,* vol. 43, no. 12, pp. 827-832, Dec. 1996.
27. R. Castello, F. Montecchi, F. Rezzi, and A. Baschirotto, Low-voltage analog filter, *IEEE Transactions on Circuits and Systems. II.,* pp. 827-840, Nov. 1995.
28. A. Baschirotto and R. Castello, 1V switched-capacitor filters, *Workshop on Advances in Analog Circuit Design,* Copenaghen, 28-30 Apr. 1998.
29. J. F. Dickson, On-chip high-voltage generation in MNOS integrated circuits using an improved voltage multiplier technique, *IEEE J. of Solid-State Circuits,* vol. SC-11, no. 3, pp. 374-378, June 1976.
30. J. Crols and M. Steyaert, Switched-opamp: an approach to realize full CMOS switched-capacitor circuits at very low power supply voltages, *IEEE J. of Solid-State Circuits,* vol. SC-29, no. 8, pp. 936-942, Aug. 1994.
31. A. Baschirotto and R. Castello, A 1V 1.8MHz CMOS Switched-opamp SC filter with rail-to-rail output swing, *IEEE Journal of Solid State Circuits,* pp. 1979-1986, Dec. 1997.

Dharchoudhury, A., et al "Timing and Signal Integrity Analysis"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

60

Timing and Signal Integrity Analysis

Abhijit Dharchoudhury
David Blaauw
Motorola, Inc.

Stantanu Ganguly
Intel Corp.

60.1 Introduction

Microprocessors are rapidly moving into deep submicron dimensions, gigahertz clock frequencies, and transistor counts in excess of 10 million transistors. This trend is being fueled by the ever-increasing demand for more powerful computers on one side and by rapid advances in process technology, architecture, and circuit design on the other side. At these small dimensions and high speeds, timing and signal integrity analyses play a critical role in ensuring that designs meet their performance and reliability goals.

Timing analysis is one of the most important verification steps in the design of a microprocessor because it ensures that the chip is meeting speed requirements. Timing analysis of multi-million transistor microprocessors is a very challenging task. This task is made even more challenging because in the deep submicron regime, transistor-level and interconnect-centric analyses become vital. Therefore, timing analysis must satisfy the two conflicting requirements of accurate low-level analysis (so that deep submicron designs can be handled) and efficient high-level abstraction (so that large designs can be handled).

The term *signal integrity* typically refers to analyses that check that signals to not assume unintended values due to circuit noise. *Circuit noise* is a broad term that applies to phenomena caused by unintended circuit behavior such as unintentional coupling between signals, degradation of voltage levels due to leakage currents and power supply voltage drops, etc. Circuit noise does not encompass physical noise effects (e.g., thermal noise) or manufacturing faults (e.g., stuck-at faults). Signal integrity is also becoming a very critical verification task. Among the various signal integrity-related issues, noise induced by coupling between adjacent wires is perhaps the most important one. With the scaling of process technologies, coupling capacitances between wires are become a larger fraction of the total wire capacitances.

Coupling capacitances are also larger because a larger number of metal layers are now available for routing, and more and more wires are running longer distances across the chip. As operating frequencies increase, noise induced on signal nets due to coupling is much greater. Noise-related functional failures are increasing as dynamic circuits become more prevalent, with circuit designers looking for increased performance at the cost of noise immunity.

Another important problem in submicron high-performance designs is the integrity of the power grid that distributes power from off-chip pads to the various gates and devices in the chip. Increased operating frequencies result in higher current demands from the power and ground lines, which in turn increases the voltage drops seen at the devices. Excessive voltage drops reduce circuit performance and inject noise into the circuit, which may lead to functional failures. Moreover, with reductions in supply voltages, problems caused by excessive voltage drops become more severe. The analysis of the power and ground distribution network to measure the voltage drops at the points where the gates and devices of the chip connect to the power grid is called *IR-drop* or *power grid analysis*.

In this chapter, we will briefly discuss the important issues in static timing analysis, noise analysis with particular emphasis on coupling noise, and IR-drop analysis methods. Additional information on these topics is available in the literature and the reader is encouraged to look through the list of references.

60.2 Static Timing Analysis

Static timing analysis (TA)[1-4] is a very powerful technique for verifying the timing correctness of a design. The power of this technique comes from the fact that it is pattern independent, implicitly verifies all signal propagation paths in the design, and is applicable to very large designs. Further, it lends itself easily to higher levels of abstraction, which makes it even more computationally feasible to perform full-chip timing analysis. The fundamental idea in static timing analysis is to find the *critical paths* in the design. Critical paths are those signal propagation paths that determine the maximum operating frequency of the design. It is easiest to think of critical paths as being those paths from the inputs to the outputs of the circuit that have the longest delay. Since the smallest clock period must be larger than the longest path delay, these paths dictate the operating frequency of the chip. In very simple terms, static TA determines these long paths using breadth-first search as follows. Starting at the inputs, the latest time at which signals arrive at a node in the circuit is determined from the arrival times at its fan-in nodes. This latest arrival time is then propagated toward the primary outputs. At each primary output, we obtain the latest possible arrival time of signals and the corresponding longest path. If the longest path does not meet the timing constraints imposed by the designer, then a violation is detected. Alternatively, if the longest path meets the timing constraints, then all other paths in the circuit will also satisfy the timing constraints. By propagating only the latest arrival time at a node, static TA does not have to explicitly enumerate all the paths in the design.

Historically, simulation-based or *dynamic timing analysis* techniques had been the most common timing analysis technique. However, with increasing complexity and size of recent microprocessor designs, static timing analysis has become an indispensable part of design verification and much more popular than dynamic approaches. Compared to dynamic approaches, static TA offers a number of advantages for verifying the timing correctness of a design. Dynamic approaches are pattern-dependent. Since the possible paths and their delays are dependent on the state of the circuit, the number of input patters that are required to verify all the paths in a circuit is exponential with the number of inputs. Hence, only a subset of paths can be verified with a fixed number of input patterns. Only moderately large circuits can be verified because of the computational cost and size limitations of transient simulators. Static TA, on the other hand, implicitly verifies all the longest paths in the design without requiring input patterns. Dynamic timing analysis is still heavily used to verify complex and critical circuitry such as PLLs, clock generators, and the like. Dynamic simulation is also used to generate timing models for block-level static timing analysis. Dynamic timing analysis technique rely on a circuit simulator (e.g., SPICE[5]) or on a fast timing simulator (e.g., ILLIADS,[6] ACES,[7] TimeMill[8]) for performing the simulations. Because

of the importance of static techniques in verifying the timing behavior of microprocessors, we will restrict the discussion below to the salient points of static TA.

DCC Partitioning

The first step in transistor-level static TA is to partition the circuit into *dc connected components* (DCCs), also called *channel-connected components*. A DCC is a set of nodes which are connected to each other through the source and drain terminals of transistors. The transistor-level representation and the DCC partitioning of a simple circuit is shown in Fig. 60.1. As seen in the diagram, a DCC is the same as the gate for typical cells such as inverters, NAND and NOR gates. For more complex structures such as latches, a single cell corresponds to multiple DCCs. The inputs of a DCC are the primary inputs of the circuit or the gate nodes of the devices that are part of the DCC. The outputs of a DCC are either primary outputs of the circuit or nodes that are connected to the gate nodes of devices in other DCCs. Since the gate current is zero and currents flow between source and drain terminals of MOS devices, a MOS circuit can be partitioned at the gates of transistors into components which can then be analyzed independently. This makes the analysis computationally feasible since instead of analyzing the entire circuit, we can analyze the DCCs one at a time. By partitioning a circuit into DCCs, we are ignoring the current conducted by the MOS parasitic capacitances that couple the source/drain and gate terminals. Since this current is typically small, the error is small. As mentioned above, DCC partitioning is required for transistor-level static TA. For higher levels of abstraction, such as gate-level static TA, the circuit has already been partitioned into gates, and their inputs are known. In such cases, one starts by constructing the timing graph as described in the next section.

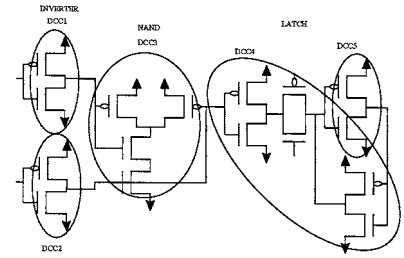


FIGURE 60.1 Transistor-level circuit partitioned into DCCs.

Timing Graph

The fundamental data structure in static TA is the *timing graph*. The timing graph is a graphical representation of the circuit, where each vertex in the graph corresponds to an input or an output node of the DCCs or gates of the circuit. Each edge or timing arc in the graph corresponds to a signal propagation from the input to the output of the DCC or gate. Each timing arc has a polarity defined by the type of transition at the input and output nodes. For example, there are two timing arcs from the input to the output of an inverter: one corresponds to the input rising and the output falling, and the other to the input falling and the output rising. Each timing arc in the graph is annotated with the propagation delay of the signal from the input to the output. The gate-level representation of a simple circuit is shown in Fig. 60.2(a) and the corresponding timing graph is shown in Fig. 60.2(b). The solid line timing arcs correspond to falling input transitions and rising output transitions, whereas the dotted line arcs represent rising input transitions and falling output transitions.

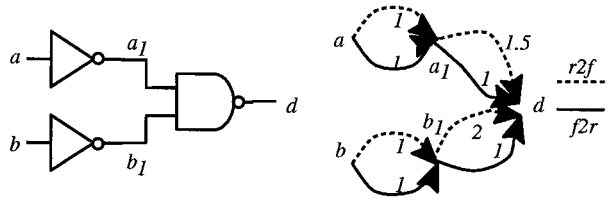


FIGURE 60.2 A simple digital circuit: (a) gate-level representation, and (b) timing graph.

Note that the timing graph may have cycles which correspond to feedback loops in the circuit. Combinational feedback loops are broken and there are several strategies to handle sequential loops (or cycles of latches).[5] In any event, the timing graph becomes acyclic and the vertices of the graph can be arranged in topological order.

Arrival Times

Given the times at which the signals at the primary inputs or *source nodes* of the circuit are stable, the minimum (earliest) and maximum (latest) arrival times of signals at all the nodes in the circuit can be calculated with a single breadth-first pass through the circuit in topological order. The early arrival time $a(v)$ is the smallest time by which signals arrive at node v and is given by

$$a(v) = \min_{u \in FI(v)} \left[a(u) + d_{uv} \right] \tag{60.1}$$

Similarly, the late arrival time $A(v)$ is the latest time by which signals arrive at node v and is given by

$$A(v) = \max_{u \in FI(v)} \left[A(u) + d_{uv} \right] \tag{60.2}$$

In the above equations, $FI(v)$ is the set of all fan-in nodes of v, i.e., all nodes that have an edge to v and d_{uv} is the delay of an edge from u to v. Equations 60.1 and 60.2 will compute the arrival times at a node v from the arrival times of its fan-in nodes and the delays of the timing arcs from the fan-in nodes to v. Since the timing graph is acyclic (or has been made acyclic), the vertices in the graph can be arranged in topological order (i.e., the DCCs and gates in the circuit can be *levelized*). A breadth-first pass through the timing graph using Eqs. 60.1 and 60.2 will yield the arrival times at all nodes in the circuit.

Considering the example of Fig. 60.2, let us assume that the arrival times at the primary inputs a and b are 0. From Eq. 60.2, the maximum arrival time for a rising signal at node a_1 is 1, and the maximum arrival time for a falling signal is also 1. In other words, $A_{a1,r} = A_{a1,f} = 1$, where the subscripts r and f denote the polarity of the signal. Similarly, we can compute the maximum arrival times at node b_1 as $A_{b1,r} = A_{b1,f} = 1$, and at node d as $A_{d,r} = 2$ and $A_{d,f} = 3$.

In addition to the arrival times, we also need to compute the *signal transition times* (or slopes) at the output nodes of the gates or DCCs. These transition times are required so that we can compute the delay across the fan-out gates. Note that there are many timing arcs that are incident at the output node and each gives rise to a different transition time. The transition time of the node is picked to be the transition time corresponding to the arc that causes the latest (earliest) arrival time at the node.

Required Times and Slacks

Constraints are placed on the arrival times of signals at the primary output nodes of a circuit based on performance or speed requirements. In addition to primary output nodes, timing constraints are automatically placed on the clocked elements inside the circuit (e.g., latches, gated clocks, domino logic gates, etc.). These timing constraints check that the circuit functions correctly and at-speed. Nodes in the circuit where timing checks are imposed are called *sink nodes*.

Timing checks at the sink nodes inject required times on the earliest and latest signal arrival times at these nodes. Given the required times at these nodes, the required times at all other nodes in the circuit can be calculated by processing the circuit in reverse topological order considering each node only once. The late required time $R(v)$ at a node v is the required time on the late arriving signal. In other words, it is the time by which signals are required to arrive at that node and is given by

$$R(v) = \max_{u \in FO(v)} \left[R(u) - d_{uv} \right] \tag{60.3}$$

Similarly, the early required time $r(v)$ is the required time on the early arriving signal. In other words, it is the time after which signals are required to arrive at node v and is given by

$$r(v) = \min_{u \in FO(v)} \left[r(u) - d_{uv} \right] \tag{60.4}$$

In these equations, $FO(v)$ is the set of fan-out nodes of v (i.e., the nodes to which there is a timing arc from node v) and d_{uv} is the delay of the timing arc from node u to node v. Note that $R(v)$ is the time *before* which a signal must arrive at a node, whereas $r(v)$ is the time *after* which the signal must arrive.

The difference between the late arrival time and the late required time at a node v is defined as the *late slack* at that node and is given by

$$S_l(v) = R(v) - A(v) \tag{60.5}$$

Similarly, the *early slack* at node v is defined by

$$S_e(v) = a(v) - r(v) \tag{60.6}$$

Note that the late and early slacks have been defined in such a way that a negative value denotes a constraint violation. The overall slack at a node is the smaller of the early and late slacks; that is,

$$S(v) = \min \; S_l(v), S_e(v) \tag{60.7}$$

Slacks can be calculated in the backward traversal along with the required times. If the slacks at all nodes in the circuit are positive, then the circuit does not violate any timing constraint. The nodes with the smallest slack value are called *critical nodes*. The most *critical path* is the sequence of critical nodes that connect the source and sink nodes.

Continuing with the example of Fig. 60.2, let the maximum required time at the output node d be 1. Then, the late required time for a rising signal at node a_1 is $R_{a1,r} = -0.5$ since the delay of the rising-to-falling timing arc from a_1 to d is 1.5. Similarly, the late required time for a falling signal at node a_1 is $R_{a1,f} = R_{d,r} - 1 = 0$. The required times at the other nodes in the circuit can be calculated to be: $R_{b1,r} = -1$, $R_{b1,f} = 0$, $R_{a,r} = -1$, $R_{a,f} = -1.5$, $R_{b,r} = -1$, and $R_{b,f} = -2$. The slack at each node is the difference between the required time and the arrival time and are as follows: $S_{d,r} = -1.5$, $S_{d,}$f $= -2$, $S_{a1,r} = -1.5$, $S_{a1,f} = -1$, $S_{b1,r} = -2$, $S_{b1,f} = -1$, $S_{a,r} = -1$, $S_{a,f} = -1.5$, $S_{b,r} = -1$, and $S_{b,f} = -2$. Thus, the critical path in this circuit is b falling — $b1$ rising — d falling, and the circuit slack is –2.

Clocked Circuits

As mentioned earlier, combinational circuits have timing checks imposed only at the circuit primary outputs. However, for circuits containing clocked elements such as latches, flip-flops, gated clocks, domino/precharge logic, etc., timing checks must also be enforced at various internal nodes in the circuit to ensure that the circuit operates correctly and at-speed. In circuits containing clocked elements, a separate recognition step is required to detect the clocked elements and to insert constraints. There are two main techniques for detecting clocked elements: pattern recognition and clock propagation.

In *pattern recognition*-based approaches, commonly used sequential elements are recognized using simple topological rules. For example. back-to-back inverters in the netlist are often an indication of a latch. For more complex topologies, the detection is accomplished using templates supplied by the user. Portions of a circuit are typically recognized in the graph of the original circuit by employing subgraph isomorphism algorithms.[9] Once a subcircuit has been recognized, timing constraints are automatically

inserted. Another application of pattern-based subcircuit recognition is to determine logical relationships between signals. For example, in pass-gate multiplexors, the data select lines are typically one-hot. This relationship cannot be obtained from the transistor-level circuit representation without recognizing the subcircuit and imposing the logical relationships for that subcircuit. The logical relationship can then be used by timing analysis tools. However, purely pattern recognition-based approaches can be restrictive and may necessitate a large number of templates from the user for proper functioning.

In *clock propagation*-based approaches, the recognition is performed automatically by propagating clock signals along the timing graph and determining how these clock signals interact with data signals at various nodes in the circuit. The primary input clocks are identified by the user and are marked as (simple) clock nodes. Starting from the primary clock inputs and traversing the timing arcs in the timing graph, the type of the nodes are determined based on simple rules. These rules are illustrated in Fig. 60.3,

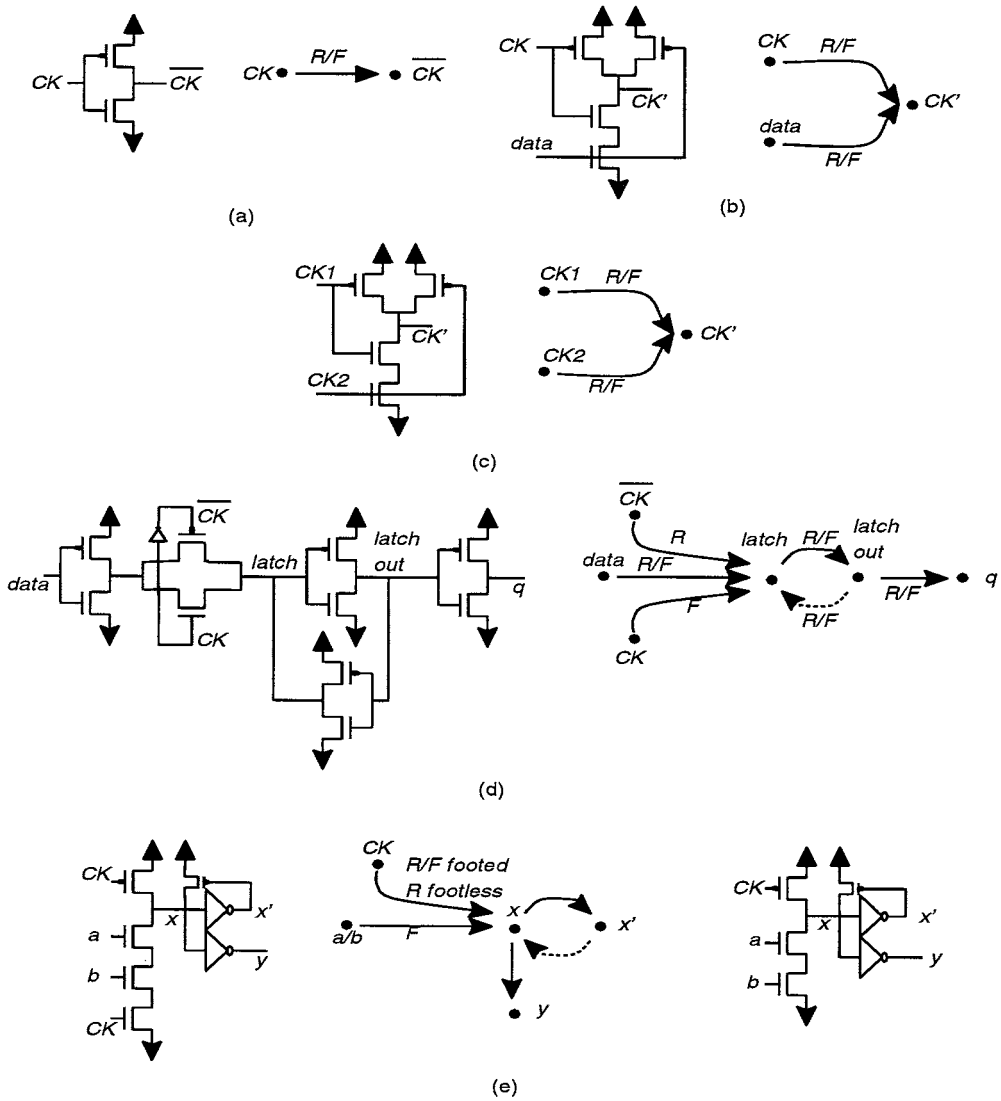


FIGURE 60.3 Sequential element detection: (a) simple clock, (b) gated clock, (c) merged clock, (d) latch node, and (e) footed and footless domino gates. Broken arcs are shown as dotted lines. Each arc is marked with the type of output transition(s) it can cause (e.g., R/F: rise and fall, R: rise only, and F: fall only).

where we show the transistor-level subcircuits and the corresponding timing subgraphs for some common sequential elements.

- A node that has only one clock signal incident on it and no feedback is classified as a *simple clock node* (Fig. 60.3(a)).
- A node that has one clock and one or more data signals incident on it, but no feedback, is classified as a *gated clock node* (Fig. 60.3(b)).
- A node that has multiple clock signals (and zero or more data signals) incident on it and no feedback is classified as a *merged clock node* (Fig. 60.3(c)).
- A node that has at least one clock and zero or more data signals incident on it and has a feedback of length two (i.e., back-to-back timing arcs) is classified as a *latch node* (Fig. 60.3(d)). The other node in the two node feedback is called the *latch output node*. A latch node is of type data. The timing arc(s) from the latch output node to the latch is (are) broken.
  Latches can be of two types: *level-sensitive* and *edge-triggered*. To distinguish between edge-triggered and level-sensitive latches, various rules may be applied. These rules are usually design-specific and will not be discussed here. It is assumed that all latches are level-sensitive unless the user has marked certain latches to be edge-triggered.

  Note that the domino gates of Fig. 60.3(e) also satisfy the conditions for a latch node. For a latch node, both data and clock signals cause rising and falling transitions at the latch node. For domino gates, data inputs *a* and *b* cause only falling transitions at the *domino node x*. This condition can be used to distinguish domino nodes from latch nodes. Footed and footless domino gates can be distinguished from each other by looking at the clock transitions on the domino node. Since the footed gate has the clocked nMOS transistor at the "foot" of the evaluate tree, the clock signal at *CK* causes both rising and falling transitions at node *x*. In the footless domino gate, *CK* causes only a rising transition at node *x*.

Clock propagation stops when a node has been classified as a data node. This type of detection can be easily performed with a simple breadth-first search on the timing graph.

Once the sequential elements have been recognized, timing constraints must be inserted to ensure that the circuit functions correctly and at-speed.[10] These are described below and illustrated in Figs. 60.4 and 60.5.

- *Simple clocks*: In this case, no timing checks are necessary. The arrival times and slopes at the simple clock node are obtained just as in normal data node.
- *Gated clocks*: The basic purpose of a gated clock is to enable or disable clock transitions at the input of the gate from propagating to the output of the gate. This is done by setting the value of the data input. For example, in the gated clock of Fig. 60.3(b), setting the data input to 1 will allow the clock waveform to propagate to the output, whereas setting the data input to 0 will disable transitions at the gate output. To make sure that this is indeed the behavior of the gated clock, the timing constraints should be such that transitions at the data input node(s) do not create transitions at the output node. For the gated NAND clock of Fig. 60.3(b), we have to ensure that the data can transition (high or low) only when the clock is low, i.e., data can transition after the clock turns low (short path constraint) and before the clock turns high (long path constraint). This is shown in Fig. 60.4(a). In addition to imposing this timing constraint, we also break the timing arc from the data node to the gated clock node since data transitions cannot create output clock transitions.
- *Merged clocks*: Merged clocks are difficult to handle in static TA since the output clock waveform may have a different clock period compared to the input clocks. Moreover, the output clock waveform depends on the logical operation performed by the gate. To avoid these problems, static TA tools typically ask the user to provide the waveform at the merged clock node and the merged clock node is treated as a (simple) clock input node with that waveform. Users can obtain the clock waveform at the merged clock node by using dynamic simulation with the input clock waveforms.

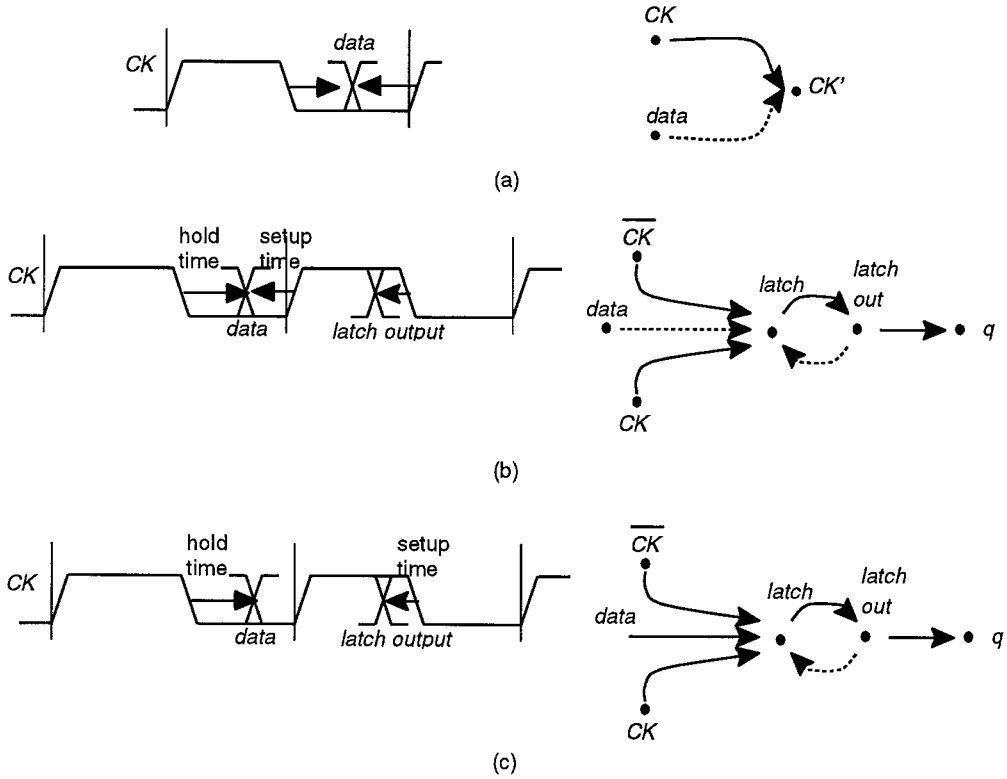


FIGURE 60.4 Timing constraints and timing graph modifications for sequential elements: (a) gated clock, (b) edge-triggered latch, and (c) level-sensitive latch. Broken arcs are shown as dotted lines.

- *Edge-triggered latches*: An edge-triggered latch has two types of constraints: *set-up constraint* and *hold constraint*. The set-up constraint requires that the data input node should be ready (i.e., the rising and falling signals should have stabilized) before the latch turns on. In the latch shown in Fig. 60.3(d), the latch is turned on by the rising edge of the clock. Hence, the data should arrive some time before the rising edge of the clock (this time margin is typically referred to as the *set-up time* of the latch). This constraint imposes a required time on the latest (or maximum) arrival time at the data input of the latch and is therefore a long path constraint. This is shown in Fig. 60.4(b). The hold constraint ensures that data meant for the current clock cycle does not accidentally appear during the on-phase of the previous clock cycle. Looking at Fig. 60.4(b), this implies that the data should appear some time after the falling edge of the clock (this time margin is called the *hold time* of the latch). The hold time imposes a required time on the early (or minimum) arrival time at the data input node and is therefore a short path constraint. As the name implies, in edge-triggered latches, the on-edge of the clock causes data to be stored in the latch (i.e., causes transitions at the latch node). Since the data input is ready before the clock turns on, the latest arrival time at the latch node will be determined only by the clock signal. To make sure that this is indeed the behavior of the latch, the timing arc from the data input node to the latch node is broken, as shown in Fig. 60.4(b). One additional set of timing constraints is imposed for an edge-triggered latch. Since data is stored at the latch (or latch output) node, we must ensure that the data gets stored before the latch turns off. In other words, signals should arrive at the latch output node before the off-edge of the clock.
- *Level-sensitive latches*: In the case of level-sensitive latches, the data need not be ready before the latch turns on, as is the case for edge-triggered latches. In fact, the data can arrive after the on-edge of the clock — this is called *cycle stealing* or *time borrowing*. The only constraint in this case

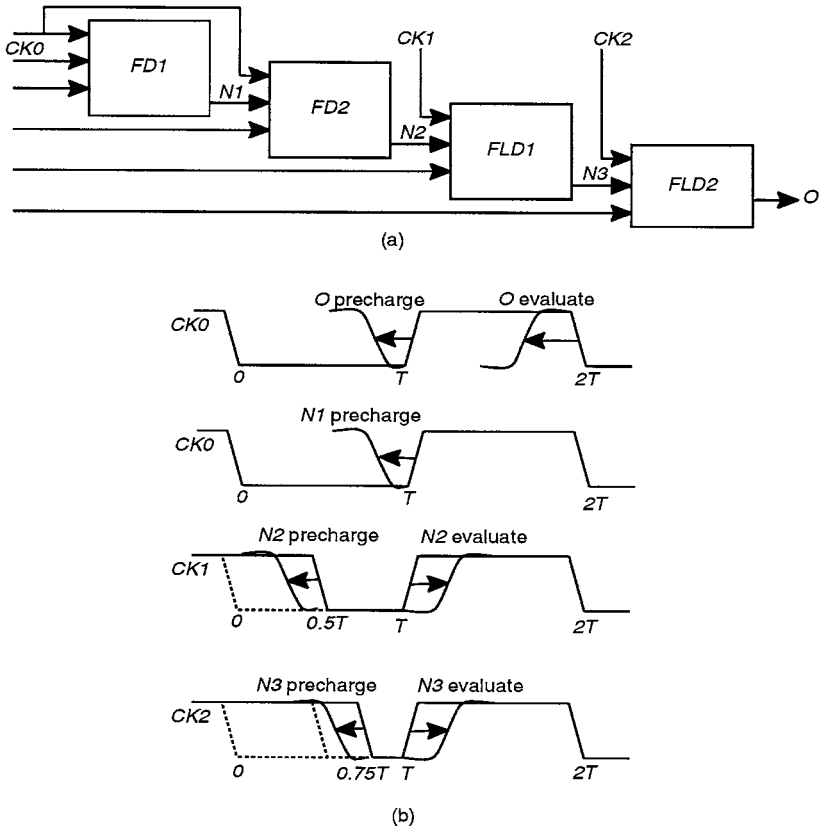


FIGURE 60.5 Domino circuit: (a) block diagram, and (b) clock waveforms and precharge and evaluate constraints. Note precharge implies the phase of operation (clock); the signals are falling.

is that the data gets latched before the clock turns off. Hence, the set-up constraint for a level-sensitive latch is that signals should arrive at the latch output node (not the latch node itself) before the falling edge of the clock, as shown in Fig. 60.4(c). The hold constraint is the same as before; it ensures that data meant for the current clock cycle arrives only after the latch was turned off in the previous clock cycle. This is also shown in Fig. 60.4(c). Since the latest arriving signal at the latch node may come from either the data or the clock node, timing arcs are not broken for a level-sensitive latch. Since data can flow through the latch, level-sensitive latches are also referred to as *transparent latches*.

- *Domino gates*: Domino circuits have two distinct phases of operation: *precharge* and *evaluate*.[11] Looking at the domino gate of Fig. 60.3(e), we see that in the precharge phase, the clock signal is low and the domino node x is precharged to a high value and the output node y is pre-discharged to a low value. During the evaluate phase, the clock is high and if the values of the gate inputs establish a path to ground, domino node x is discharged and output node y turns high. The difference between *footed* and *footless domino gates* is the clocked nMOS transistor at the "foot" of the nMOS evaluate tree. To demonstrate the timing constraints imposed on domino circuits, consider the domino circuit block diagram and the clock waveforms shown in Fig. 60.5. The footed domino blocks are labeled *FD*1 and *FD*2, and the footless blocks are labeled *FLD*1 and *FLD*2. From Fig. 60.5(b), note that all three clocks have the same period $2T$, but the falling edge of *CK*2 is $0.25T$ after the falling edge of *CK*1 which in turn is $0.5T$ after the falling edge of *CK*0. Therefore, the precharge phase for *FD*1 and *FD*2 is T, for *FLD*1 is $0.5T$, and for *FLD*2 is $0.25T$. The various timing constraints for domino circuits are illustrated in Fig. 60.5 and discussed below.

1. We want the output O to evaluate (rise) before the clock starts falling and to precharge (fall) before the clock starts rising.
2. Consider node $N1$, which is an output of $FD1$ and an input of $FD2$. $N1$ starts precharging (falling) when $CK0$ falls, and the constraint on it is that it should finish precharging before $CK0$ starts rising.
3. Next, consider node $N2$, which is an input to $FLD1$ clocked by $CK1$. Since this block is footless, $N2$ should be low during the precharge phase to avoid short-circuit current. $N2$ starts precharging (falling) when $CK0$ starts falling and should finish falling before $CK1$ starts falling. Note that the falling edges of $CK0$ and $CK1$ are $0.5T$ apart, and the precharge constraint is on the late or maximum arrival time of $N2$ (long path constraint). Also, $N2$ should start rising only after $CK1$ has finished rising. This is a constraint on the early or minimum arrival time of $N2$ (short path constraint). In this example, $N2$ starts rising with the rising edge of $CK0$ and, since all the clock waveforms rise at the same time, the short path constraint will be satisfied trivially.
4. Finally, consider node $N3$. Since $N3$ is an input of $FLD2$, it must satisfy the short-circuit current constraints. $N3$ starts precharging (falling) when $CK1$ starts falling and it should fall completely before $CK2$ starts falling. Since the two clock edges are $0.25T$ apart, the precharge constraint on $N3$ is tighter than the one on $N2$. As before, the short path constraint on $N3$ is satisfied trivially.

The above discussion highlights the various types of timing constraints that must be automatically inserted by the static TA tool.

Note that each relative timing constraint between two signals is actually composed of two constraints. For example, if signal d must rise before clock CK rises, then (1) there is a required time on the late or maximum rising arrival time at node d (i.e., $A_{d,r} < A_{CK,r}$), and (2) there is a required time on the early or minimum rising arrival time at the clock node CK (i.e., $a_{CK,r} < a_{d,r}$). There is one other point to be noted. Set-up and hold constraints are fundamentally different in nature. If a hold constraint is violated, then the circuit will not function at any frequency. In other words, hold constraints are *functional constraints*. Set-up constraints, on the other hand, are *performance constraints*. If a set-up constraint is violated, the circuit will not function at the specified frequency, but it will function at a lower frequency (lower speed of operation). For domino circuits, precharge constraints are functional constraints, whereas evaluate constraints are performance constraints.

Transistor-Level Delay Modeling

In transistor-level static TA, delays of timing arcs have to be computed on the fly using transistor-level delay estimation techniques. There are many different transistor-level delay models which provide different tradeoffs between speed and accuracy. Before reviewing some of the more popular delay models, we define some notations. We will refer to the delay of a timing arc as being its *propagation delay* (i.e., the time difference between the output and the input completing half their transitions). For a falling output, the fall times is defined as the time to transition from 90% to 10% of the swing; similarly, for a rising output, the rise time is defined as the time to transition from 10% to 90% of the swing. The transition time at the output of the timing arc is defined to be either the rise time or the fall time. In many of the delay models discussed below, the transition time at the input of a timing arc is required to find the delay across the timing arc. At any node in the circuit, there is a transition time corresponding to each timing arc that is incident on that node. Since for long path static TA, we find the latest arriving signal at a node and propagate that arrival time forward, the transition time at a node is defined to be the output transition time of the timing arc which produced the latest arrival time at the node. Similarly, for short path analysis, we find the transition time as the output transition time of the timing arc that produced the earliest arrival time at the node.

Analytical closed-form formulae for the delay and output transition times are useful for static TA because of their efficiency. One such model was proposed in Hedenstierna et al.,[12] where the propagation

delay across an inverter is expressed as a function of the input transition time s_{in}, the output load C_L, and the size and threshold voltages of the NMOS and PMOS transistors. For example, the inverter delay for a rising input and falling output is given by

$$t_d = k_0 \frac{C_L}{\beta_n} + s_{in}\left(k_1 + k_2 V_{tn}\right) \tag{60.8}$$

where β_n is the NMOS transconductance (proportional to the width of the device), V_{tn} is the NMOS threshold voltage, and k_0, k_1, and k_2 are constants. The formula for the rising delay is the same, with PMOS device parameters being used. The output transition time is considered to be a multiple of the propagation delay and can be calibrated to a particular technology. More accurate analytical formulae for the propagation delay and output transition time for an inverter gate have been reported in the literature.[13,14] These methods consider more complex circuit behavior such as short-circuit current (both NMOS and PMOS transistors in the inverter are conducting) and the effect of MOS parasitic capacitances that directly couple the input and outputs of the inverter. More accurate models of the drain current and parasitic capacitances of the transistor are also used. The main shortcoming of all these delay models is that they are based on an inverter primitive; therefore, arbitrary CMOS gates seen in the circuit must be mapped to an equivalent inverter.[15] This process often introduces large errors.

A simpler delay model is based on replacing transistors by linear resistances and using closed-form expressions to compute propagation delays.[16,17] The first step in this type of delay modeling is to determine the charging/discharging path from the power supply rail to the output node that contains the switching transistor. Next, each transistor along this path is modeled as an effective resistance and the MOS diffusion capacitances are modeled as lumped capacitances at the transistor source and drain terminals. Finally, the Elmore time constant[18] of the path is obtained by starting at the power supply rail and adding the product of each transistor resistance and the sum of all downstream capacitances between the transistor and the output node. The accuracy of this method is largely dependent on the accuracy of the effective resistance and capacitance models. The effective resistance of a MOS transistor is a function of its width, the input transition time, and the output capacitance load. It is also a function of the position of the transistor in the charging/discharging path. The position variable can have three values: *trigger* (when the input at the gate of the transistor is switching), *blocking* (when the transistor is not switching and it lies between the trigger and the output node), and *support* (when the transistor is not switching and lies between the trigger and the power supply rail). The simplest way to incorporate these effects into the resistance model is to create a table of the resistance values (using circuit simulation) for various values of the transistor width, the input transition, and the output load. During delay modeling, the resistance value of a transistor is obtained by interpolation from the calibration table. Since the position is a discrete variable, a different table must be stored for each position variable. The effective MOS parasitic capacitances are functions of the transistor width and can also be modeled using a table look-up approach. The main drawbacks of this approach are the lack of accuracy in modeling a transistor as a linear resistance and capacitance, as well as not considering the effect of parallel charging/discharging paths and complementary paths. In our experience, this approach typically gives 10–20% accuracy with respect to SPICE for standard gates (inverters, NANDs, NORs, etc.); for complex gates, the error can be greater. These methods do not compute the transition time or slope at the output of the DCC. The transition time at the output node is considered to be a multiple of the propagation delay. Note that the propagation delay across a gate can be negative; this is the case, for example, if there is a slow transition at the input of a strong but lightly loaded gate. As a result, the transition time would become negative, giving a large error compared to the correct value.

Yet another method of modeling the delay from an input to an output of a DCC (or gate) is based on running a circuit simulator such as SPICE,[5] or a fast timing simulator such as ILLIADS[6] or ACES.[7] Since the waveform at the switching input is known, the main challenge in this method is to determine the assertions (whether an input should be set to a high or low value) for the side inputs which gives rise to

a transition at the output of the DCC.[19] For example, let us consider a rising transition at the input causing a falling transition at the output. In this case, a valid assertion is one that satisfies the following two conditions: (1) before the transition, there should be no conducting path between the output node and *Gnd*, and (2) after the transition, there should be at least one conducting path between the output node and *Gnd* and no conducting path between the output node and V_{dd}. The sensitization condition for a rising output transition is exactly symmetrical. The valid assertions are usually determined using a binary decision diagram.[20] For a particular input-output transition, there may be many valid assertions; these valid assertions may have different delay values since the primary charging/discharging path may be different or different node capacitances in the side paths may be charged/discharged. To find the assertion that causes the worst-case (or best-case) delay, one may resort to explicit simulations of all the valid assertions or employ other heuristics to prune out certain assertions. The main advantage of this type of delay modeling is that very accurate delay and transition time estimates can be obtained since the underlying simulator is accurate. The added accuracy is obtained at the cost of additional runtime.

Since static timing analyzers typically use simple delay models for efficiency reasons, the top few critical paths of the circuit should be verified using circuit simulation.[21,22]

Interconnects and Static TA

As is well known, interconnects are playing a major role in determining the performance of current microprocessors, and this trend is expected to continue in the next generation of processors.[23] The effect of interconnects on circuit and system performance should be considered in an accurate and efficient manner during static timing analysis. To illustrate interconnect modeling techniques, we will use the example shown in Fig. 60.6(a) of a wire connecting a driving inverter to three receiving inverters.

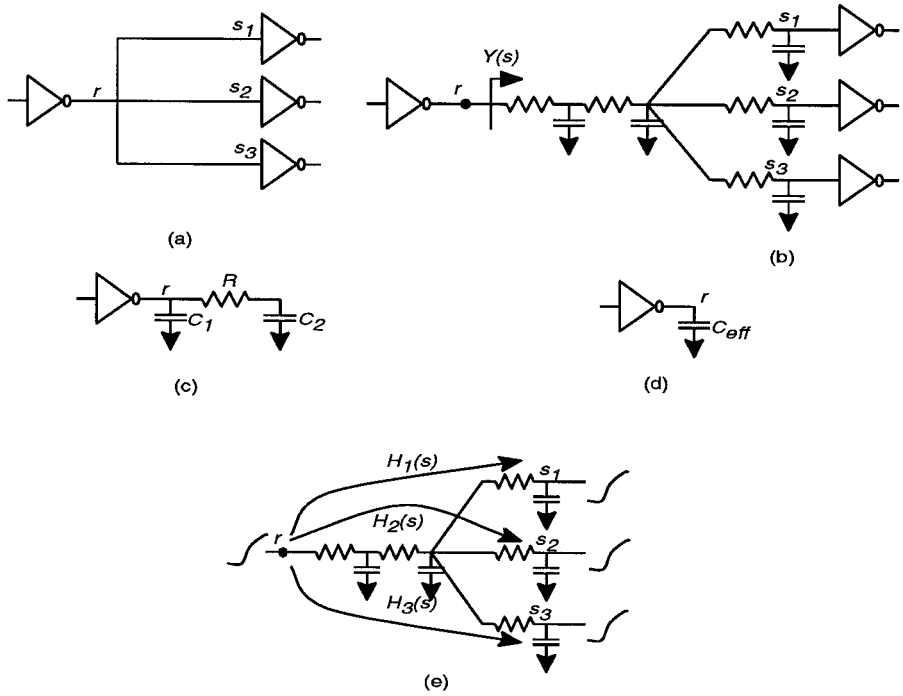


FIGURE 60.6 Handling interconnects in static TA: (a) a typical interconnect, (b) distributed RC model of interconnect, (c) reduced π-model to represent the loading of the interconnect, (d) effective capacitance loading, and (e) propagation of waveform from root to sinks.

The simplest interconnect model is to lump all the interconnect and receiver gate capacitances at the output of the driver gate. This approximation may greatly overestimate the delay across the driver gate since, in reality, all of the downstream capacitances are not "seen" by the driver gate because of resistive shielding due to line resistances. A more accurate model of the wire as a distributed RC line is shown in Fig. 60.6(b). This is the wire model output by most commercial RC extraction tools. In Fig. 60.6(b), node *r* is called the *root* of the interconnect and is driven by the driver gate, and the other end points of the wire at the inputs of the receiver gate are called *sinks* of the interconnect and are labeled s_1, s_2, and s_3. Interconnects have two main effects: (1) the interconnect resistance and capacitance determines the effective load seen by the driving gate and therefore its delay, and (2) due to non-zero wire resistances, there is a non-zero delay from the root to the sinks of the interconnect — this is called the *time-of-flight delay*.

To model the effect of the interconnect on the driver delay, we first replace the metal wire with a π-model load as shown in Fig. 60.6(c).[24] This is done by finding the first three moments of the admittance $Y(s)$ of the interconnect at node *r*. It can be shown that the admittance is given by $Y(s) = m_1 s + m_2 s^2 + m_3 s^3 + \ldots$. Next, we obtain the admittance of the π-load as $\hat{Y}(s) = s(C_1 + C_2) - s^2RC^2_2 + s^3R^2C^3_2 + \ldots$, where R, C_1, and C_2 are the parameters of the π-load model. To obtain the parameters of the π-load, we equate the first three moments of $Y(s)$ and $\hat{Y}(s)$. This gives us the following equations for the parameters of the π-load model:

$$C_2 = \frac{m_2^2}{m_3}, \quad C_1 = m_1 - \frac{m_2^2}{m_3}, \quad \text{and} \quad R = -\frac{m_3^2}{m_2^3} \tag{60.9}$$

Now, if we are using a transistor-level delay model or a pre-characterized gate-level delay model that can only handle purely capacitive loading and not π-model loads, we have to determine an effective capacitance C_{eff} that will accurately model the π-load. The basic idea of this method[25,26] is to equate the average current drawn by the π-model load to the average current drawn by the C_{eff} load. Since the average current drawn by any load is dependent on the transition time at the output of the gate and the transition time is itself a function of the load, we have to iterate to converge to the correct value of C_{eff}. Once the effective capacitance has been obtained, the delay across the driver gate and the waveform at node *r* can be obtained.

The waveform at the root node is then propagated to the sink nodes s_1, s_2, s_3 across the transfer functions $H_1(s)$, $H_2(s)$, and $H_3(s)$, respectively. This procedure is illustrated in Fig. 60.6(e). If the driver waveform can be simplified as a ramp, the output waveforms at the sink nodes can be computed easily using reduced-order modeling techniques like AWE[27] and the time-of-flight delay between the root node and the sink nodes can be calculated.

Process Variations and Static TA

Unavoidable variations and disturbances present in IC manufacturing processes cause variations in device parameters and circuit performances. Moreover, variations in the environmental conditions (of such parameters are temperature, supply voltages, etc.) also cause variations in circuit performances.[28] As a result, static TA should consider the effect of process and environmental variations. Typically, statistical process and environmental variations are considered by performing analysis at two process corners: *best-case* corner and *worst-case* corner. These process corners are typically represented as different device model parameter sets, and as the name implies, are for the fastest and slowest devices. For gate-level static TA, gate characterization is first performed at these two corners yielding two different gate delay models. Then, static TA is performed with the best-case and worst-case gate delay models. Long path constraints (e.g., latch set-up and performance or speech constraints) are checked with the worst-case models and short path constraints (e.g., latch hold constraints) are checked with the best-case models.

Timing Abstraction

Transistor-level timing analysis is very important in high-performance microprocessor design and verification since a large part of the design is hand-crafted and cannot be pre-characterized. Analysis at the transistor level is also important to accurately consider interconnect effects such as gate loading, charge-sharing, and clock skew. However, full-chip transistor-level analysis of large microprocessor designs is computationally infeasible, making timing abstraction a necessity.

Gate-Level Static TA

A straightforward extension of transistor-level static TA is to the gate level. At this level of abstraction, the circuit has been partitioned into gates, and the inputs and outputs of each gate have been identified. Moreover, the timing arcs from the inputs to the outputs of a gate are typically pre-characterized. The gates are characterized by applying a ramp voltage source at the input of the gate and an explicit load capacitance at the output of the gate. Then, the transition time of the ramp and the value of the load capacitance is varied, and circuit simulation (e.g., SPICE) is used to compute the propagation delays and output transition times for the various settings. These data points can be stored in a table or abstracted in the form of a curve-fitted equation. A popular curve-fitting approach is the *k*-factor equations,[26] where the delay t_d and output transition time t_{out} are expressed as non-linear functions of the input transition time s_{in} and the capacitive output load C_L:

$$t_d = \left(k_1 + k_2 C_L\right) s_{in} + k_3 C_L^2 + k_4 C_L + k_5 \tag{60.10}$$

$$t_{out} = \left(k_1' + k_2' C_L\right) s_{in} + k_3' C_L^2 + k_4' C_L + k_5'. \tag{60.11}$$

The various coefficients in the *k*-factor equations are obtained by curve fitting the data. Several modifications, including more complex equations and dividing the plane into a number of regions and having equations for each region, have been proposed.

The main advantage of gate-level static TA is that costly on-the-fly delay and output transition time calculations can be replaced by efficient equation evaluations or table look-ups. This is also a disadvantage since it requires that all the timing arcs in the design are pre-characterized. This may be a problem when parts of the design are not complete and the delays for some timing arcs are not available. This problem can be avoided if the design flow ensures that at early stages of a design, estimated delays are specified for all timing arcs which are then replaced by characterized numbers when the design gets completed. To apply gate-level TA to designs that contain a large amount of custom circuits, timing rules must be developed for the custom circuits also. Gate-level static TA is still at a fairly low level of abstraction and the effects of interconnects and clock skew can be considered. Moreover, at the gate level, the latches and flip-flops of the design are visible and so timing constraints can be inserted directly at those nodes.

Black-Box Modeling

At the next higher level of abstraction, gates are grouped together into blocks and the entire design (or chip) now consists of these blocks or "boxes." Each box contains combinational gates as well as sequential elements such as latches as shown in Fig. 60.7(a). Timing checks inside the block can be verified using static TA at the transistor or gate level. At the chip level, the internal nodes of the box are no longer visible and its timing behavior must be abstracted at the input, output, and clock pins of the box. In black-box modeling, we assume that the first and last latch along any path from input to output of the box are edge-triggered latches; in other words, cycle stealing is not allowed across these latches (cycle stealing may be allowed across other transparent latches inside the box). The first latch along a path from input to output is called an *input latch* and the last latch is called an *output latch*. With this assumption, there can be two types of paths to the outputs of the box. First, paths that originate at box inputs and end at box outputs without traversing through any latches. These paths are represented as input-output arcs in the block-box with the path delays annotated on the arcs. Second, there are paths that originate

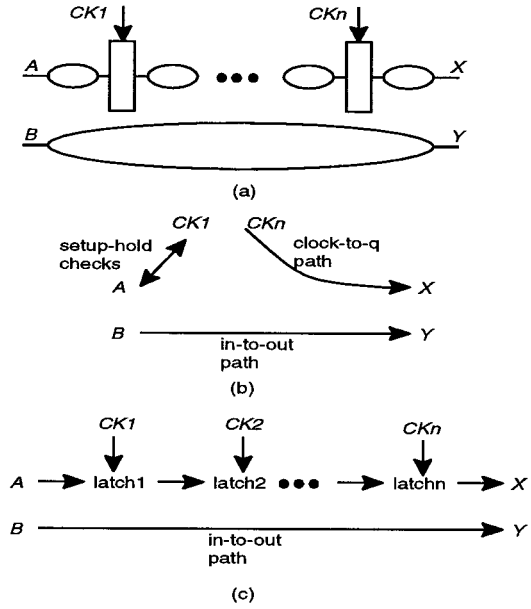


FIGURE 60.7 High-level timing abstraction: (a) a block containing combinational and sequential elements, (b) black-box model, and (c) gray-box model.

at the clock pins of the output edge-triggered latches and end at the box outputs.These paths are represented as clock-to-input arcs in the black-box and the paths delays are annotated on the arcs. Finally, the set-up and hold time constraints of the input latches are translated to constraints between the box inputs and clock pins. These constraints will be checked at the chip-level static TA. The constraints and the arcs are shown in Fig. 60.7(b). Note that the timing checkpoints inside a block have been verified for a particular set of clocks when the black-box model is generated. Since these timing checkpoints are no longer available at the chip level, a black-box model is valid only for a particular frequency. If a different clock frequency (or different clock waveforms) is used, then the black-box model must be regenerated.

Gray-Box Modeling

Gray-box modeling removes the edge-triggered latch restrictions of black-box modeling. All latches inside the box are allowed to be level-sensitive and therefore have to be visible at the top level so that the constraints can be checked and cycle-stealing is allowed through these latches. As shown in Fig. 60.7(c), the gray-box model consists of timing arcs from the box inputs to the input latches, from latches to latches, and from the output latches to the box outputs. The clock pins of each of the latches are also visible at the chip level, and so the set-up and hold timing constraints for each latch in the box is checked at the chip level. In addition to these timing arcs, there can also be direct input-output timing arcs. Note that since the timing checkpoints internal to the box are available at the chip level, the gray-box model is frequently independent — unlike the black-box model.

False Paths

To find the critical paths in the circuit, static TA propagates the arrival times from the timing inputs to the timing outputs. Then, it propagates the required times from the outputs back to the inputs and computes the slacks along the way. During propagation, static TA does not consider the logical functionality of the circuit. As a result, some of the paths that it reports to the user may be such that they cannot be activated by any input vector. Such paths are called false paths.[29-31] An example of a false path is shown in Fig. 60.8(a). For x to propagate to a, we must set $y = 1$, which is the non-controlling value of the NAND gate. Similarly, for a to propagate to b, we set $z = 1$. Now, since $y = z = 1$, $e = 0$ (the controlling

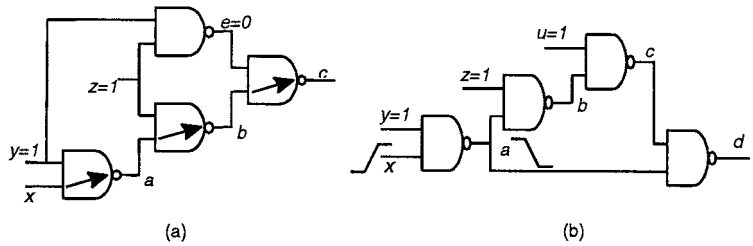


FIGURE 60.8 False path examples: (a) static false path, and (b) dynamic false path.

value for a NAND gate), and there can be no signal propagation from b to c. Therefore, there can be no propagation from x to c (i.e., $x - a - b - c$ is a false path). False paths that arise due to logical correlations are called *static false paths* to distinguish them from dynamic false paths, which are caused by temporal correlations.

A simple example of a *dynamic false path* is shown in Fig. 60.8(b). Suppose we want to find the critical path from node x to the output d. It is clear that there are two such paths, $x - a - d$ and $x - a - b - c - d$, of which the latter has a larger delay. In order to sensitize the longer path $x - a - b - c - d$, we would set the other inputs of the circuit to the non-controlling values of the gates (i.e., $y = z = u = 1$). If there is a rising transition on node x, there will be a falling transition on nodes a and c. However, because of the propagation delay from a to c, node a will fall well before node c. As soon as node a falls, it will set the primary output d to be 1 (since the controlling value of a NAND gate is 0). Because node a always reaches the controlling value before node c, it is not possible for a transition at node c to reach the output. In other words, the path x rising – a falling – b rising – c falling – d rising is a dynamic false path. Note that if we add some combinational logic between the output of the first NAND gate and the input of the last NAND gate to slow the signal a down, then the transition on c could propagate to the output. The example shown above is for purposes of illustration only and may appear contrived. However, dynamic false paths are very common in carry-lookahead adders.[32]

Finding false paths in a combinational circuit is an NP-complete problem. There are a number of heuristic approaches that find the longest paths in a circuit while determining and ignoring the false paths.[29-31] Timing analysis techniques that can avoid false paths specified by the user have also been reported.[33,34]

60.3 Noise Analysis

In digital circuits, nodes that are not switching are at the nominal values of the supply (logic 1) and ground (logic 0) rails. In a digital system, noise is defined as a deviation of these node voltages from their stable high or low values. Digital noise should be distinguished from physical noise sources that are common in analog circuits (e.g., shot noise, thermal noise, flicker noise, and burst noise).[35] Since noise causes a deviation in the stable logic voltages of a node, it can be classified into four categories: (1) *high undershoot* noise reduces the voltage of a node that is supposed to be at logic 1; (2) *high overshoot* noise which increases the voltage of a logic 1 node above the supply level (V_{dd}); (3) *low overshoot* noise increases the voltage of a node that is supposed to be at logic 0; and (4) *low undershoot* noise which reduces the voltage of a logic 0 node below the ground level (*Gnd*).

Sources of Digital Noise

The most common sources of noise in digital circuits are crosstalk noise, power supply noise, leakage noise and charge-sharing noise.[36]

Crosstalk Noise

Crosstalk noise is the noise voltage induced on a net that is at a stable logic value due to interconnect capacitive coupling with a switching net. The net or wire that is supposed to be at a stable value is called the *victim net*. The switching nets that induce noise on the victim net are called *aggressor nets*. Crosstalk noise is the most common source of noise in deep submicron digital designs because, as interconnect wires get scaled, coupling capacitances become a larger fraction of the total wire capacitances.[23] The ratio of the width to the thickness of metal wires reduces with scaling, resulting in a larger fraction of the total capacitance of the wire being contributed by coupling capacitances. Several examples of functional failures caused by crosstalk noise are given in the next section.

Power Supply Noise

This refers to noise on the power supply and ground nets of a design that is passed onto the signal nets by conducting transistors. Typically, the power supply noise has two components. The first is produced by IR-drop on the power and ground nets due to the current demands of the various gates in the chip (discussed in the next section). The second component of the power supply noise comes from the RLC response of the chip and package to current demands that peak at the beginning of a clock cycle. The first component of power supply noise can be reduced by making the wires that comprise the power and ground network wider and denser. The second component of the noise can be reduced by placing on-chip decoupling capacitors.[37]

Charge-Sharing Noise

Charge-sharing noise is the noise induced at a dynamic node due to charge redistribution between that node and the internal nodes of the gate.[32] To illustrate charge-sharing noise, let us again consider the two-input domino NAND gate of Fig. 60.9(a). Let us assume that during the first evaluate phase shown in Fig. 60.9(b), both nodes x and x_1 are discharged. Then, during the next precharge phase, let us assume that the input a is low. Node x will be precharged by the PMOS transistor *MP*, but x_1 will not and will remain at its low value. Now, suppose *CK* turns high, signaling the beginning of another evaluate phase. If during this evaluate phase, a is high but b is low, nodes x and x_1 will share charge resulting in the waveforms shown in Fig. 60.9(b): x will be pulled low and x_1 will be pulled high. If the voltage on x is reduced by a large amount, the output inverter may switch and cause the output node y to be wrongly set to a logic high value. Charge-sharing in a domino gate is avoided by precharging the internal nodes in the NMOS evaluate tree during the precharge phase of the clock. This is done by adding an anti-charge sharing device such as *MNc* in Fig. 60.9(c) which is gated by the clock signal.

Leakage Noise

Leakage noise is due to two main sources: *subthreshold conduction* and *substrate noise*. Subthreshold leakage current[32] is the current that flows in MOS transistors even when they are not conducting (off).

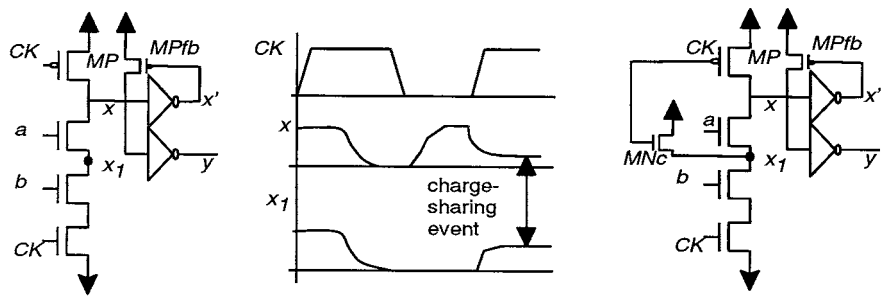


FIGURE 60.9 Example of charge-sharing noise: (a) a two-input domino NAND gate, (b) waveforms for charge-sharing event, and (c) anti charge-sharing device.

This current is a strong function of the threshold voltage of the device and the operating temperature. Subthreshold leakage is an important design parameter in portable devices since battery life is directly dependent on the average leakage current of the chip. Subthreshold conduction is also an important noise mechanism in dynamic circuits where, for a part of the clock cycle, a node does not have a strong conducting path to power or ground and the logic value is stored as a charge on that node. For example, suppose that the inputs *a* and *b* in the two-input domino NAND gate of Fig. 60.9(a) are low during the evaluate phase of the clock. Due to subthreshold leakage current in the NMOS evaluate transistors, the charge on node *x* may be drained away, leading to a degradation in its voltage and a wrong value at the output node *y*. The purpose of the half latch device *MPfb* is to replenish the charge that may be lost due to the leakage current.

Another source of leakage noise is minority carrier back injection into the substrate due to bootstrapping. In the context of mixed analog-digital designs, this is often referred to as *substrate noise*.[38] Substrate noise is often reduced by having guard bands, which are diffusion regions around the active region of a transistor tied to supply voltages so that the minority carriers can be collected.

Crosstalk Noise Failures

In this section, we provide some examples of functional failures caused by crosstalk noise. Functional failures result when induced noise voltages cause an erroneous state to be stored at a memory element (e.g., at a latch node or a dynamic node). Consider the simple latch circuit of Fig. 60.10(a) and let us

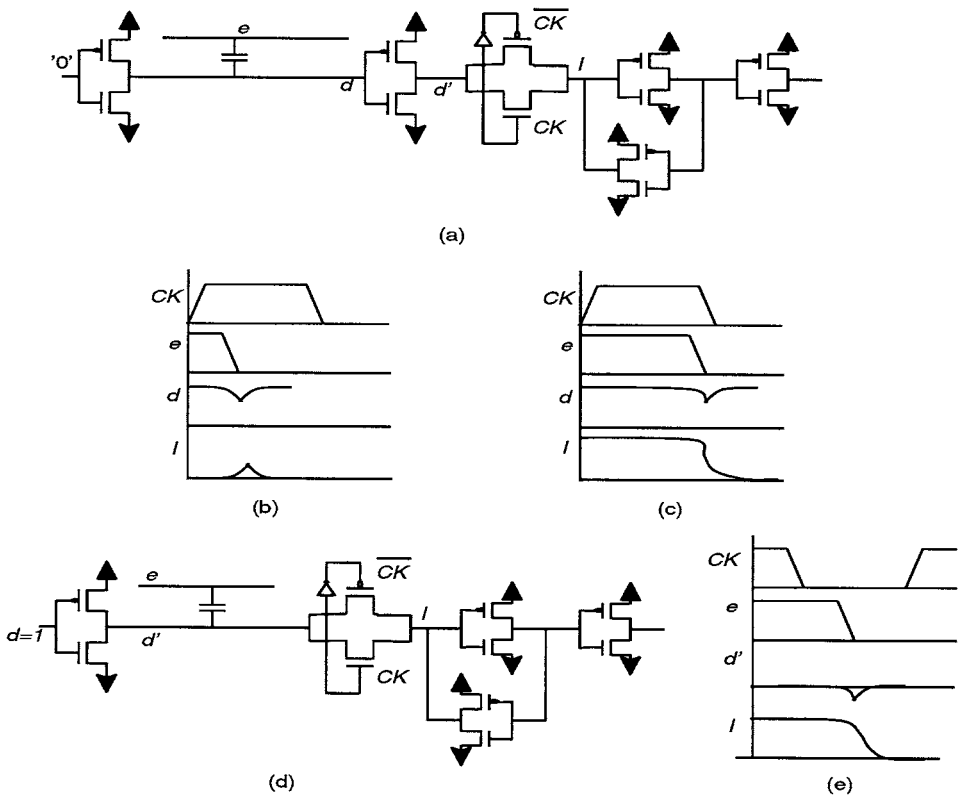


FIGURE 60.10 Crosstalk noise-induced functional failures: (a) latch circuit; (b) high undershoot noise on *d* does not cause functional failure in (b) but does cause failure in (c); (d) same latch circuit with noise induced on an internal node; and (e) low undershoot noise causing a failure.

assume that the data input *d* is a stable high value and the latch *l* has a stable low value. If the net corresponding to node *d* is coupled to another net *e* and there is a high to low transition on net *e*, net *d* will be pulled low. When *e* has finished switching, *d* will be pulled back to a high value by the PMOS transistor driving net *d* and the noise on *d* will dissipate. Thus, the transition on net *e* will cause a noise pulse on *d*. If the amplitude of this noise pulse is large enough, the latch node *l* will be pulled high. Depending on the conditions under which the noise is injected, it may or may not cause a wrong value to be stored at the latch node. For example, let us consider the situation depicted in Fig. 60.10(b), where *CK* is high and the latch is open. If the noise pulse on *d* appears near the middle of the clock phase, then the latch node will be pulled high; but as the noise on *d* dissipates, latch node *l* will return to its correct value because the latch is open. However, if the noise pulse on *d* appears near the end of the clock phase as shown in Fig. 60.10(c), the latch may turn off before the noise on *d* dissipates, the latch node may not recover, and a wrong value will be stored. A similar unrecoverable error may occur if noise appears on the clock net turning the latch on when it was meant to be off. This might cause a wrong value to be latched.

Now, let us consider the latch circuit of Fig. 60.10(d) where the wire between the input inverter and the pass gate of the latch is long and subject to coupling capacitances. Suppose the latch is turned off (*CK* is low), the data input is high so that the node *d′* is low, and a high value is stored at the latch node. If net *e* transitions from a high to a low value, a low undershoot noise will be introduced on *d′*. If this noise is sufficiently large, the NMOS pass transistor will turn on even through its gate voltage is zero (since its gate-source voltage will become greater than its threshold voltage). This will discharge the latch node *l*, resulting in a functional failure.

In order to push performance, domino circuits are becoming more and more prevalent.[88] These circuits trade performance for noise immunity and are susceptible to functional noise failures. A noise-related functional failure in domino circuits is shown in Fig. 60.11. Again, let us consider the two-input domino NAND gate shown in Fig. 60.11(a). Let us assume that during the evaluate phase, *a* is held to a low value by the driving inverter, but *b* is high. Then, *x* should remain charged and *y* should remain low. If an unrelated net *d* switches high, and there is sufficient coupling between signals *a* and *d*, then a low overshoot noise pulse will be induced on node *a*. If the pulse is large enough, a path to ground will be created and node *x* will be discharged. As shown in Fig. 60.11(b), this will erroneously set the output node of the domino gate to a high value. When the noise on *a* dissipates, it will return to a low value, but *x* and *y* are not able to recover from the noise event, causing a functional failure.

As the examples above demonstrate, functional failures due to digital noise cause circuits to malfunction. Noise analysis is becoming an important failure mechanism in deep submicron designs because of several technology and design trends. First, larger die sizes and greater functionality in modern chips result in longer wires, which makes the circuit more susceptible to coupling noise. Second, scaling of interconnect geometries has resulted in increased coupling between adjacent wire.[23] Third, the drive for faster performance has increased the use of faster non-restoring logic families such as domino logic.

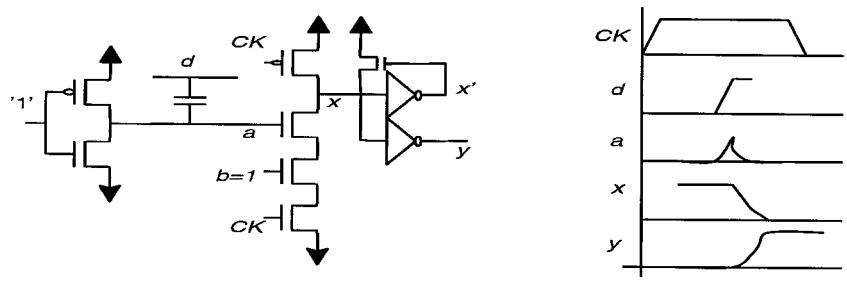


FIGURE 60.11 Functional failure in domino gates: (a) two-input NAND gate, and (b) voltage waveforms when input noise causes a functional failure.

These circuit families have faster switching speeds at the expense of reduced noise immunity. False switching events at the inputs of these gates are catastrophic since precharged nodes may be discharged and these nodes cannot recover their original state when the noise dissipates. Fourth, lower supply voltage levels reduce the magnitudes of the noise margins of circuits. Finally, in state-of-the-art microprocessors, many functional units located in different parts of the chip are operating in parallel and this causes a lot of switching activity in long wires that run across different parts of the chip. All of these factors make noise analysis a very important task to verify the proper functioning of digital designs.

Modeling of Interconnect and Gates for Noise Analysis

Let us consider the example of Fig. 60.12(a) where three wires are running in parallel and are capacitively coupled to each other. Suppose that we are interested in finding the noise that is induced on the middle net by the adjacent nets switching. The middle net is called the *victim net* and the two neighboring nets are called *aggressors*. Consider the situation when the victim net is held to a stable logic zero value by the victim driver and both the aggressor nets are switching high. Due to the coupling between the nets, a low overshoot noise will be induced on the victim net as shown in Fig. 60.12(a). If the noise pulse is large and wide enough, the victim receiver may switch and cause a wrong value at the output of the inverter.

The circuit-level models for this system are explained below and shown in Fig. 60.12(b).

1. The (net) complex consisting of the victim and aggressor nets is modeled as a coupled distributed RC network. The coupled RC lines are typically output by a parasitic extraction tool.
2. The non-linear victim driver is holding the victim net to a stable value. We model the non-linear driver as a linear holding resistance. For example, if the victim driver holds the output to logic 0 (logic 1), we determine an effective NMOS (PMOS) resistance. The value of the holding resistance for a gate can be obtained by pre-characterization using SPICE.
3. The aggressor driver is modeled as a Thevenin voltage source in series with a switching resistance. The Thevenin voltage source is modeled as a shifted ramp, where the ramp starts switching at time t_0 and the transition time is Δt. The switching resistance is denoted by R_s.
4. The victim receiver is modeled as a capacitor of value equal to the input capacitance of the gate.

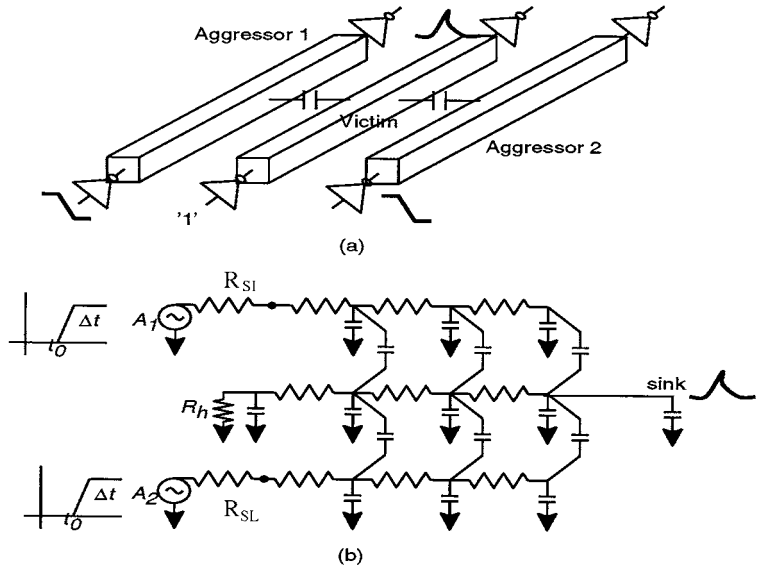


FIGURE 60.12 (a) A noise pulse induced on the victim net by capacitive coupling to adjacent aggressor nets, and (b) linearized model for analysis.

These models convert the non-linear circuit into a linear circuit. The multiple sources in this linear circuit can now be analyzed using *linear superposition*. For each aggressor, we get a noise pulse at the sink(s) of the victim net, while shorting the other aggressors. These noise pulses have different amplitudes and widths; the amplitude and width of the *composite noise waveform* is obtained by aligning these noise pulses so that their peaks line up. This is a conservative assumption to simulate the worst-case noise situation.

Input and Output Noise Models

As mentioned earlier, noise creates circuit failures when it propagates to a charge-storage node and causes a wrong value to be stored at the node. Propagating noise across non-linear gates[39] makes the noise analysis problem complex. In this discussion, a more conservative simple model will be discussed. With each input terminal of a victim receiver gate, we associate a noise rejection curve.[40] This is a curve of the noise amplitude versus the noise width that produces a predefined amount of noise at the output. If we assume a triangular noise pulse at the input of the victim receiver, the noise rejection curve defines the amplitude-width combination that produces a fixed amount of noise at the output of the receiver. A sample noise rejection curve is shown in Fig. 60.13. As the width becomes very large, the noise amplitude tends toward the dc noise margin of the gate. Due to the lowpass nature of a digital gate, very sharp noise pulses are filtered out and do not cause any appreciable noise at the output. When the noise pulse at the sink(s) of the victim net have been obtained, the pulse amplitude and width are compared against the noise rejection curve to determine if a noise failure occurs.

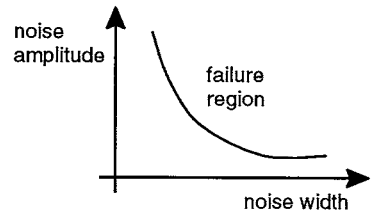


FIGURE 60.13 A typical noise rejection curve.

Since we do not propagate noise across gates, noise injected into the victim net at the output of the victim driver must model the maximum amount of noise that may be produced at the output of a gate. The output noise model is a dc noise that is equal to the predefined amount of output noise that was used to determine the input noise rejection curve above. Contributions from other dc noise sources such as IR-drop noise may be added to the output noise. If we assume that there is no resistive dc path to ground, this output noise appears unchanged at the sink(s) of the victim net.

Linear Circuit Analysis

The linear circuit that models the net complex to be analyzed can be quite large since the victim and aggressor nets are modeled as a large number of RC segments and the victim net can be coupled to many aggressor nets. Moreover, there are a large number of nets to be analyzed. Since general circuit simulation tools such as SPICE can be extremely time-consuming for these networks, fast linear circuit simulation tools such as RICE[41] can be used to solve these large net complexes. RICE uses reduced-order modeling and asymptotic waveform evaluation (AWE) techniques[27] to speed up the analysis while maintaining sufficient accuracy. Techniques that overcome the stability problems in AWE, such as Pade via Lancszos (PVL),[42] Arnoldi-based techniques,[43] congruence transform-based techniques (PACT),[44] or combinations (PRIMA),[45] have been proposed recently.

Interaction with Timing Analysis

Calculation of crosstalk noise interacts tightly with timing analysis since timing analysis lets us determine which of the aggressor nets can switch at the same time. This reduces the pessimism of assuming that for a victim net, all the nets it is coupled to can switch simultaneously and induce noise on it. Timing analysis defines timing windows by the earliest and latest arrival times for all signals. This is shown in Fig. 60.14 for three aggressors $A1$, $A2$, and $A3$ of a particular victim net of interest. Based upon these timing windows, we can define five different scenarios for noise analysis where different aggressors can

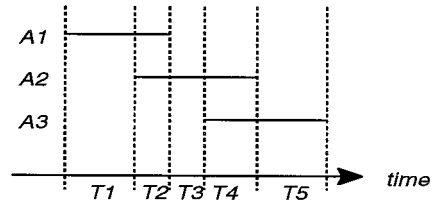


FIGURE 60.14 Effect of timing windows on aggressor selection for noise analysis.

switch simultaneously. For example, in interval $T1$, only $A1$ can switch; in $T2$, $A1$, and $A2$ can switch; in $T3$, only $A2$ can switch; and so on. Note that in this case, all three aggressors can never switch at the same time. Without considering the timing windows provided by timing analysis, we would have overestimated the noise by assuming that all three aggressors could switch at the same time.

Fast Noise Calculation Techniques

Any state-of-the-art microprocessors will have many nets to be analyzed, but typically only a small fraction of the nets will be susceptible to noise problems. This motivates the use of extremely fast techniques that provably overestimate the noise at the sinks of a net. If a net passes the noise test under this quick analysis, then it does not need to be analyzed any further; if a net fails the noise test, then it can be analyzed using more accurate techniques. In this sense, these fast techniques can be considered to be *noise filters*. If these noise filters produce sufficiently accurate noise estimates, then the expectation is that a large number of nets would be screened out quickly. This combination of fast and detailed analysis techniques would therefore speed up the overall analysis process significantly. Note that noise filters must be provably pessimistic and that multiple noise filters with less and less pessimism can be used one after the other to successively screen out nets.

Let us consider the net complex shown in Fig. 60.15(a), where we have modeled the net as distributed RC lines, the victim driver as a linear holding resistance, and the aggressors as voltage ramps and linear

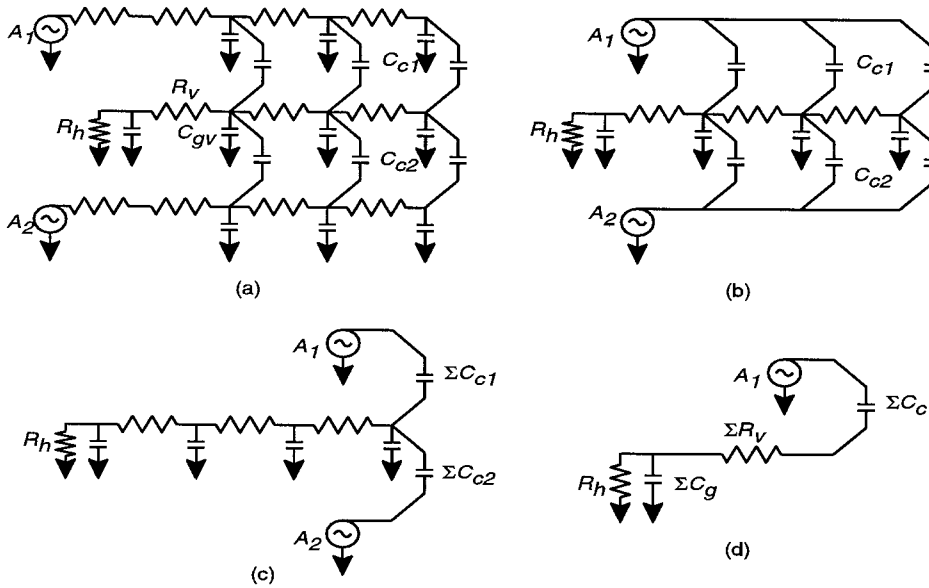


FIGURE 60.15 Noise filters: (a) original net complex with distributed RC models for aggressors and victims, (b) aggressor lines have only coupling capacitances to victim, (c) aggressors are directly coupled to sink of victim, and (d) single (strongest) aggressor and all grounded capacitors of victim moved away from sink.

resistances. The grounded capacitances of the victim net is denoted as C_{gv}, and the coupling capacitances to the two aggressors are denoted as C_{c1} and C_{c2}. In Figs. 60.15(b-d), we show the steps through which we can obtain a circuit which will provide a provably pessimistic estimate of the noise waveform. In Fig. 60.15(b), we have removed the resistances of the aggressor nets. This is pessimistic because, in reality, the aggressor waveform slows down as it proceeds along the net. By replacing it with a faster waveform, more noise will be induced on the victim net. In Fig. 60.15(c), the aggressor waveforms are capacitively coupled directly into the sink net; for each aggressor, the coupling capacitance is equal to the sum of all the coupling capacitances between itself and the victim net. Since the aggressor is directly coupled to the sink net, this transformation will result in more induced noise. In Fig. 60.15(d), we have made two modifications; first, we replaced the different aggressors by one capacitively coupled aggressor and, second, we moved all the grounded capacitors on the victim net away from the sink node. The composite aggressor is just the fastest aggressor (i.e., the aggressor that has the smallest transition time) and it is coupled to the victim net by a capacitor whose value is equal to the sum of all the coupling capacitances in the victim net. To simplify the victim net, we sum all the grounded capacitors and insert it at the root of the victim net and sum all the net resistances. By moving the grounded (good) capacitors away from the sink net, we increase the amount of coupled noise. This simple network can now be analyzed very quickly to compute the (pessimistic) noise pulse at the sink.

An efficient method to compute the peak noise amplitude at the sink of the victim net is described by Devgan.[46] Under infinite ramp aggressor inputs, the maximum noise amplitude is the final value of the coupled noise. For typical interconnect topologies, these analytical computations are simple and quick.

Noise, Circuit Delays, and Timing Analysis

Circuit noise, especially crosstalk noise, significantly affects switching delays. Let us consider the example of Fig. 60.16(a), where we are concerned about the propagation delay from *A* to *C*. In the absence of any coupling capacitances, the rising waveform at *C* is shown by the dotted line of Fig. 60.16(b). However, if net 2 is switching in the opposite direction (node *E* is rising as in Fig. 60.16(b)), then additional charge is pumped into net 1 due to the coupling capacitors causing the signals at nodes B_1 and B_2 to slow down. This in turn causes the inverter to switch later and causes the propagation delay from *A* to *C* to be much larger, as shown in the diagram. Note that if net 2 switched in the same direction as net 1, then the delay from *A* to *C* would be reduced. This implies that delays across gates and wires depend on the switching activity on adjacent coupled nets. Since coupling capacitances are a large fraction of the total capacitance of wires, this dependence will be significant and timing analysis should account for this behavior. Using the same terminology as crosstalk noise analysis, we call the net whose delay is of primary interest (net 1 in the above example) the *victim net* and all the nets that are coupled to it are called *aggressor nets.*

A model that is commonly used to approximate the effect of coupling capacitors on circuit delays is to replace each coupling capacitor by a grounded capacitor of twice the value. This model is accurate

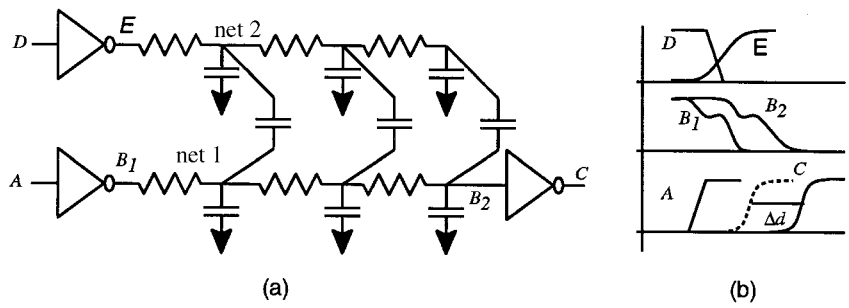


FIGURE 60.16 Effect of noise on circuit delays: (a) victim and aggressor nets, and (b) typical waveforms.

only when the victim and aggressor nets are identical and the waveforms on the two nets are identical, but switching in opposite directions. For some cases, doubling the coupling capacitance may be pessimistic, but in many cases it is not — the effective capacitance is much more than twice the coupling capacitance. Note that the effect on the propagation delay due to coupling will be strongly dependent on how the aggressor waveforms are aligned with respect to each other and to the victim waveform. Hence, one of the main issues in finding the effect of noise on delay is to determine the aggressor alignments that cause the worst propagation delay.

A more accurate model for considering the effect of noise on delay is described by Dartu et al.[47] In this approach, the gates are replaced by linearized models (e.g., the Thevenin model of the gate consists of a shifted ramp voltage source in series with a resistance). Once the circuit has been linearized, the principle of linear superposition is applied. The voltage waveform at the sink of the victim net is first obtained by assuming that all aggressors are "quiet." Then the victim net is assumed to be quiet and each aggressor is switched one at a time and the resultant noise waveforms at the victim sink node is recorded. These noise waveforms are offset with respect to each other because of the difference in the delays between the aggressors to the victim sink node. Next, the aggressor noise waveforms are shifted such that the peaks get lined up and a composite noise waveform is obtained by adding the individual noise waveforms. The remaining issue is to align the composite noise waveform with the noise-free victim waveform to obtain the worst delay. This process is described in Fig. 60.17, where we show the original noise-free waveform V_{orig} and the (composite) noise waveform V_{noise} at the victim sink node. Then, the worst case is to align the noise such that its peak is at the time when $V_{orig} = 0.5V_{dd} - V_N$, where V_N is the peak noise.[47,48] The final waveform at C is marked V_{final}.

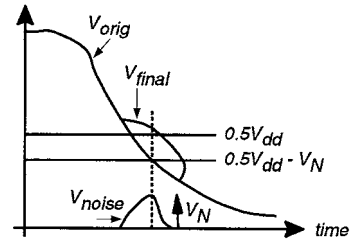


FIGURE 60.17 Aligning the composite noise waveform with the original waveform to produce worst-case delay.

The impact of noise on delays and the impact of timing windows on noise analysis implies that one has to iterate between timing and noise analysis. There is no guarantee that this process will converge; in fact, one can come up with examples when the process diverges. This is one of the open issues in noise analysis.

60.4 Power Grid Analysis

The power distribution network distributes power and ground voltages to all the gates and devices in the design. As the devices and gates switch, the power and ground lines conduct current and due to the resistance of the lines, there is an unavoidable voltage drop at the point of distribution. This voltage drop is called *IR-drop*. As device densities and switching currents increase, larger currents flow in the power distribution network causing larger IR-drops. Excessive voltage drops in the power grid reduce switching speeds of devices (since it directly affects the current drive of devices) and noise margins (since the effective rail-to-rail voltage is lower). Moreover, as explained in the previous section, IR-drops inject dc noise into circuits which may lead to functional or performance failures. Higher average current densities lead to undesirable wear-and-tear of metal wires due to electromigration.[49] Considering all these issues, a robust power distribution network is vital in meeting performance and reliability goals in high-performance microprocessors. This will achieve good voltage regulation at all the consumption points in the chip, notwithstanding the fluctuations in the power demand across the chip. In this section, we give a brief overview of various issues involved in power grid analysis.

Problem Characteristics

The most important characteristic of the power grid analysis problem is that it is a global problem. In other words, the voltage drop in a certain part of the chip is related to the currents being drawn from that as well as other parts of the chip. For example, if the same power line is distributing power to several

functional units in a certain part of the chip, the voltage drop in one functional unit depends on the currents being drawn by the other functional units. In fact, as more and more of the functional units switch together, the IR-drop in all the functional units will increase because the current supply demand on the power line is more.

Since IR-drop analysis is a global problem and since power distribution networks are typically very large, a critical issue is the large size of the network. For a state-of-the-art microprocessor, a number of nodes in the power grid is on the order of millions. An accurate IR-drop analysis would simulate the non-linear devices in the chip, together with the non-ideal power grid, making the size of the network even more unmanageable. In order to keep IR-drop analysis computationally feasible, the simulation is done in two steps. First, the non-linear devices are simulated assuming perfect supply voltages, and the power and ground currents drawn by the devices are recorded (these are called *current signatures*). Next, these devices are modeled as independent time-varying current sources for simulating the power grid and the voltage drops at the consumption points (where transistors are connected to power and ground rails) are measured. Since voltage drops are typically less than 10% of the power supply voltage, the error incurred by ignoring the interaction between the device currents and the actual supply voltage is usually small. The linear power and ground network is still very large and hierarchy has to be exploited to reduce the size of the analyzed network. Hierarchy will be discussed in more detail later.

Yet another characteristic of the IR-drop analysis problem is that it is dependent on the activity in the chip, which in turn is dependent on the vectors that are supplied. An important problem in IR-drop analysis is to determine what this input pattern should be. For IR-drop analysis, patterns that produce maximum instantaneous currents are required. This topic has been addressed by a few papers,[50-52] but will not be discussed here. However, the fact that vectors are important means that transient analysis of the power grid is required. Since each solution of the network is expensive and since many simulations are necessary, dynamic IR-drop analysis is very expensive. The speed and memory issues related to linear system solution techniques becomes important in the context of transient analysis. An important issue in transient analysis is related to the capacitances (both parasitic and intentional decoupling) in the power grid. Since capacitors prevent instantaneous changes in node voltages, IR-drop analysis without considering capacitors will be more pessimistic. A pessimistic analysis can be done by ignoring all power grid capacitances, but a more accurate analysis with capacitances may require additional computation time for solving the network.

Yet another issue is raised by the vector dependence. As mentioned earlier, the non-linear simulation to determine the currents drawn from the power grid is done separately (from the linear network) using the supplied vectors. Since the number of transistors in the whole chip is huge, simultaneous simulation of the whole chip may be infeasible because of limitations in non-linear transient simulation tools (e.g., SPICE or fast timing simulators). This necessitates partitioning the chip into blocks (typically corresponds to functional units, like floating point unit, integer unit, etc.) and performing the simulation one block at a time. In order to preserve the correlation among the different blocks, the blocks must be simulated with the same underlying set of chip-wide vectors. To determine the vectors for a block, a logic simulation of the chip is done, and the signals at the inputs of the block are monitored and used as inputs for the block simulation.

Since dynamic IR-drop analysis is typically expensive (especially since many vectors are required), techniques to reduce the number of simulations are often used. A commonly used technique is to compress the current signatures from the different clock cycles into a single cycle. The easiest way to accomplish this is to find the maximum envelope of the multi-cycle current signature. To find the maximum envelope over N cycles, the single-cycle current signature is computed using

$$i_{sc}(t) = \max i_{orig}(t + kT),\ 1 \le k \le N,\ 0 \le t \le T \tag{60.12}$$

where $i_{sc}(t)$ is the single-cycle, $i_{orig}(t)$ is the original current signature, and T is the clock period. Since this method does not preserve the correlation among different current sources (sinks), it may be overly pessimistic.

A final characteristic of IR-drop analysis is related to the way in which the analysis is typically done. Typically, the analysis is done at the very last stages of the design when the layout of the power network is available. However, IR-drop problems that could be revealed at this stage are very expensive or even impossible to fix. IR-drop analysis that is applicable to all stages of a microprocessor design has been addressed by Dharchoudhury et al.[53]

Power Grid Modeling

The power and ground grids can be extracted by a parasitic extractor to obtain an R-only or an RC network. Extraction implies that the layout of the power grid is available. To insert the transistor current sources at the proper nodes in the power grid, the extractor should preserve the names and locations of transistors. Power grid capacitances come from metal wire capacitances (coupling and grounded), device capacitances, and decoupling capacitors inserted in the power grid to reduce voltage fluctuations. Several interesting issues are raised in the modeling of power grid capacitances. The power or ground net is coupled to other signal nets and since these nets are switching, the effective grounded capacitance is difficult to compute. The same is true for capacitances of MOS devices connected to the power grid. Making the problem worse, the MOS capacitances are voltage dependent. These issues have not been completely addressed as yet. Typically, one resorts to worst-case analysis by ignoring coupling capacitances to signal nets and MOS device capacitances, but considering only the grounded capacitances of the power grid and the decoupling capacitors.

There are three other issues related to power grid modeling. First, for electromigration purposes, via arrays should be extracted as resistance arrays so that current crowding can be modeled. Electromigration problems are primarily seen in the vias and if the via array is modeled as a single resistance, such problems could be masked. Second, the inductance of the package pins also creates a voltage drop in the power grid. This drop is created by the time-varying current in the pins ($v = Ldi/dt$). This effect is typically handled by adding a fixed amount of drop on top of the on-chip IR-drop estimate. Third, a word of caution about network reduction or crunching. Most commercial extraction tools have options to reduce the size of an extracted network. This reduction is typically performed using reduced-order modeling techniques with interconnect delay being the target. This reduction is intended for signal nets and is done so that errors in the interconnect delay is kept below a certain threshold. For IR-drop analysis, such crunching should not be done since we are not interested in the delay. Moreover, during the reduction the nodes at which transistors hook up to the power grid could be removed.

Block Current Signatures

As mentioned above, accurate modeling of the current signatures of the devices that are connected to the power grid is important. At a certain point in the design cycle of a microprocessor, different blocks may be at different stages of completion. This implies that multiple current signature models should be available so that all the blocks in the design can be modeled at various stages in the design.[53]

The most accurate model is to provide transient current signatures for all the devices that are connected to the supply or ground grid. This assumes that the transistor-level representation of the entire block is available. The transient current signatures are obtained by transistor-level simulation (typically with a fast transient simulator) with user-specified input vectors. As mentioned earlier, in order to maintain correlation with other blocks, the input vectors for each block must be derived from a common chip-wide input vector set. At the chip-level, the vectors are usually hot loops (i.e., the vectors try to turn on as many blocks as possible). The block-level inputs for the transistor-level simulation are obtained by monitoring the signal values at the block inputs during a logic simulation of the entire chip with the hot loop vectors.

At the other end of the spectrum, the least accurate current model for a block is an area-based dc current signature. This is employed at early stages of analysis when the block design is not complete. The average current consumption per unit area of the block can be computed from the average power consumption

specification for the chip and the normal supply voltage value. Since the peak current can be larger than the average current, some multiple of the average per-unit-area current is multiplied by the block area to compute the current consumption for the block.

An intermediate current model can be derived from a full-chip gate-level power estimation tool. Given a set of input vectors, this tool computes the average power consumed by each block over a cycle. From the average power consumption, an average current can be computed for each cycle. Again, to account for the difference between the peak and average currents, the average current can be multiplied by a constant factor. Hence, one obtains a multi-cycle dc current signature for the block in this model.

Matrix Solution Techniques

The large size of power grids places very stringent demands on the linear system solver, making it the most important part of an IR-drop analysis tool. The power grids in typical state-of-the-art microprocessors usually contain multiple layers of metal (processes with up to six layers of metal are currently available) and the grid is usually designed as a mesh. Therefore, the network cannot usually be reduced significantly using a tree-link type of transformation. In older-generation microprocessors, the power network was often "routed" and therefore more amenable to tree-link type reductions. In networks of this type, significant reduction in the size can typically be obtained.[54]

In general, matrix solution techniques can be categorized into two major types: direct and iterative.[55] The size and structure of the conductance matrix of the power grid is important in determining the type of linear solution technique that should be used. Typically, the power grid contains millions of nodes, but the conductance matrix is very sparse (typically, less than five entries per row or column of the matrix). Since it is a conductance matrix, the matrix will also be symmetric positive definite — for a purely resistive grid, the conductance matrix may be ill-conditioned.

Iterative solution techniques apply well to sparse systems, but their convergence can be slowed down by ill-conditioning. Convergence can usually be improved by applying pre-conditioners. Another important advantage of iterative methods is that they do not suffer from size limitations as much as direct techniques. Iterative techniques usually need to store the sparse matrix and a few iteration vectors during the solution. The disadvantage of iterative techniques is in transient solution. If constant time steps are used during transient simulation, the conductance matrix remains the same from one time point to another and only the right-hand side vector changes. Iterative techniques depend on the right-hand side and so a fresh solution is required for each time point during transient simulation. The solution from previous time points cannot be reused. The most widely used iterative solution technique for IR-drop analysis is the conjugate gradient solution technique. Typically, a pre-conditioner such as incomplete Cholesky pre-conditioning is also used in conjunction with the conjugate gradient scheme.

Direct techniques rely on first factoring the matrix and then using these factors with the right-hand side vector to find the solution. Since the matrix is symmetric positive definite, one can apply specialized direct techniques such as Cholesky factorization. The main advantage of direct techniques in the context of IR-drop analysis is in transient analysis. As explained earlier, transient simulation with constant time steps will result in the linear solution of a fixed matrix. Direct techniques can factor this matrix once and the factors can be reused with different right-hand side vectors to give some efficiency. The main disadvantage of direct techniques is memory usage to store the factors of the conductance matrix. Although the conductance matrix is sparse, its factors are not and this means that the memory usage will be $O(n^2)$, were n is the size of the matrix.

Exploiting Hierarchy

From the discussions above, it is clear that IR-drop analysis of large microprocessor designs can be limited by size restrictions. The most effective way to reduce the size is to exploit the hierarchy in the design. In this discussion, we will assume a two-level hierarchy consisting of the chip and its constituent blocks. This hierarchy in the blocks also partitions the entire power distribution grid into two parts: the global

grid and the intra-block grid. The global grid distributes power from the chip pads to tap points in the various blocks (these are called block ports) and the intra-block grid distributes power from these tap points to the transistors in the block. This partitioning allows us to apply hierarchical analysis. First, the intra-block power grid can be analyzed to find the voltages at the transistor tap points. This analysis assumes that the voltages at the block ports are equal to ideal supply (V_{dd}) or ground (0). The intra-block analysis must also determine a macromodel for the block which is then used for analyzing the global grid. A block admittance macromodel will consist of a current source at each port and an admittance matrix relating the currents and voltages among the ports. The size of the admittance matrix will be equal to the number of ports and each entry will model the effect of the voltage at one port to the current at some other port. In other words, the off-diagonal entries in the admittance matrix will model current redistribution between the ports of the block. Note that, in general, the admittance matrix will be dense and have p^2 entries if p is the number of ports. If n is the number of nodes in the intra-block grid, this block would have contributed a sparse submatrix of size n to the global grid during flat analysis. For hierarchical analysis, this block contributes a dense submatrix of size p. If $p << n$, hierarchical analysis will be more efficient than a flat analysis, both in terms of computational time and memory usage.

For exact equivalence with flat analysis, the admittance between every pair of ports must be modeled, resulting in a dense admittance matrix for the block. This will reduce the sparsity of the global conductance matrix and adversely affect solution speed. However, if a block is large, the effective resistance between two ports that are far away will be very large and so the corresponding entry in the admittance matrix can be zeroed with very little loss in accuracy. In fact, the simplest block model will consist of current sources at the ports and a diagonal admittance matrix. For chip-level analysis, the error from this assumption can be kept small if the blocks themselves are small. There is one other source of error in hierarchical analysis and that is the dependence of the block currents on the port voltages. Again, if the voltage drops to the blocks are small (as it will be in a well-designed grid), the error due to this assumption will be small.

References

1. R.B. Hitchcock, G.L. Smith and D.D. Cheng, Timing analysis of computer hardware, *IBM J. Res. Develop.*, 26(1), 100-105, Jan. 1982.
2. N.P. Jouppi, Timing analysis and performance improvement of MOS VLSI designs, *IEEE Trans. Computer-Aided Design*, 6(4), 650-665, July 1987.
3. K.A. Sakallah, T.N. Mudge, and O.A. Olukotun, checkT_c and minT_c: Timing verification and optimal clocking of synchronous digital circuits, *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp. 552-555, Nov. 1990.
4. T. Burks, K.A. Sakallah, and T.N. Mudge, Critical paths in circuits with level-sensitive latches, *IEEE Trans. Very Large Scale Integration Systems*, 3(2), 273-291, June 1995.
5. L.W. Nagel, SPICE 2: A computer program to simulate semiconductor circuits, Technical Report ERL-M520, Univ. of California, Berkeley, May 1975.
6. Y.H. Shih, Y. Leblebici, and S.M. Kang, ILLIADS: A fast timing and reliability simulator for digital MOS circuits, *IEEE Trans. Computer-Aided Design*, pp. 1387-1402, Sept. 1993.
7. A. Devgan and R.A. Rohrer, Adaptively controlled explicit simulation, *IEEE Trans. Computer-Aided Design*, pp. 746-762, June 1994.
8. TimeMill Reference Manual, Epic Design Technology, 1996.
9. Generalized recognition of gates, Bull Worldwide Information Systems, Sept. 1994.
10. N. Weste and K. Eshragian, *Principles of CMOS VLSI Design*, Addison-Wesley, 1990.
11. A. Dharchoudhury, D. Blaauw, J. Norton, S. Pullela, and J. Dunning, Transistor-level sizing and timing verification of domino circuits in the PowerPC™ microprocessor, *Proc. Intl. Conf. Computer Design*, pp. 143-148, 1997.
12. N. Hedenstierna and K.O. Jeppson, CMOS circuit speed and buffer optimization, *IEEE Trans. Computer-Aided Design*, 6(2), 270-281, Mar. 1987.

13. T. Sakurai and A.R. Newton, Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas, *IEEE J. Solid-State Circuits*, 25(2), 584-594, April 1990.
14. A.I. Kayssi, K.A. Sakallah, and T.M. Burks, Analytical transient response of CMOS inverters, *IEEE Trans. Circuits. Syst.*, 39(1), 42-45, Jan. 1992.
15. A. Nabavi-Lishi and N.C. Rumin, Inverter models of CMOS gates for supply current and delay evaluation, *IEEE Trans. Computer-Aided Design*, 13(10), 1271-1279, Oct. 1994.
16. J. Rubinstein, P. Penfield, and M.A. Horowitz, Signal delay in RC tree networks, *IEEE Trans. Computer-Aided Design*, 2(3), 202-211, July 1983.
17. J. Cherry, Pearl: A CMOS timing analyzer, *Proc. ACM/IEEE Design Automation Conf.*, pp. 148-153, 1988.
18. W.C. Elmore, The transient response of damped linear networks with particular regard to broadband amplifiers, *J. Applied Physics*, 19(1), 55-63, Jan. 1948.
19. T. Burkes and R.E. Mains, Incorporating signal dependencies into static transistor-level delay calculation, *Proc. TAU 97*, pp. 110-119, Dec. 1997.
20. R. Bryant, Graph-based algorithms for boolean function manipulation, *IEEE Trans. Computers*, 35(8), 677-691, Aug. 1986.
21. M. Desai and Y.T. Yen, A systematic technique for verifying critical path delays in a 300 MHz Alpha CPU design using circuit simulation, *Proc. Design Automation Conf.*, pp. 125-130, 1996.
22. S. Savithri, D. Blaauw, and A. Dharchoudhury, A three tier assertion technique for SPICE verification of transistor-level timing analysis, *Proc. Intl. VLSI'99*, Jan. 1999.
23. H. Bakoglu, *Circuits, Interconnection and Packaging for VLSI*, Addison-Wesley, Reading, MA, 1990.
24. P.R. O'Brien and T.L. Savarino, Modeling the driving point characteristics of resistive interconnect for accurate delay estimation, *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp. 512-515, Nov. 1989.
25. J. Qian, S. Pullela, and L.T. Pillage, Modeling the effective capacitance for the RC interconnect of CMOS gates, *IEEE Trans. Computer-Aided Design*, pp. 1526-1555, Dec. 1994.
26. F. Dartu, N. Menezes, J. Qian, and L.T. Pileggi, A gate-delay model for high-speed CMOS circuits, *Proc. ACM/IEEE Design Automation Conf.*, 1994.
27. L.T. Pillage and R.A. Rohrer, Asymptotic waveform evaluation for timing analysis, *IEEE Trans. Computer-Aided Design*, 9, 352-366, April 1990.
28. J.C. Zhang and M.A. Styblinski, *Yield and Variability Optimization of Integrated Circuits*, Kluwer Academic: Boston, 1995.
29. D.H.C. Du, S.H.C. Yen, and S. Ghanta, On the general false path problem in timing analysis, *Proc. Design Automation Conf.*, pp. 555-560, 1989.
30. P.C. McGeer and R.K. Brayton, Efficient algorithms for computing the longest viable path in a combinational network, *Proc. Design Automation Conf.*, pp. 561-567, 1989.
31. Y. Kukimoto, W. Gost, A. Saldanha, and R. Brayton, Approximate timing analysis of combinational circuits under the XBD0 model, *Proc. ACM/IEEE Conf. Computer-Aided Design*, pp. 176-181, 1997.
32. M. Shoji, *CMOS Digital Circuit Technology*, Prentice-Hall: Englewood Cliffs, NJ, 1988.
33. K.P. Belkhale and A.J. Seuss, Timing analysis with known false sub graphs, *Proc. ACM/IEEE Intl. Conf. Computer-Aided Design*, pp. 736-740, Nov. 1995.
34. D. Blaauw and T. Edwards, Generating false path free timing graphs for circuit optimization, *Proc. TAU99*, March 1999.
35. D.A. Hodges and H.G. Jackson, *Analysis and Design of Digital Integrated Circuits*, McGraw-Hill: New York, NY, 1988.
36. K.L. Sheppard and V. Narayanan, Noise in deep submicron digital design, *Proc. ACM/IEEE Design Automation Conf.*, pp. 524-531, 1996.
37. H.C. Chen, Minimizing chip-level simultaneous switching noise for high-performance microprocessor design, *Proc. IEEE Intl. Symp. Circuits Syst.*, 4, 544-547, 1996.
38. P.K. Su, M.J. Loinaz, S. Masui, and B.A. Wooley, Experimental results and modeling techniques for substrate noise in mixed-signal integrated circuits, *IEEE J. Solid-State Circuits*, 28(4), 420-430, 1993.

39. K.L. Sheppard, V. Narayana, P.C. Elmendorf, and G. Zheng, Global harmony: Coupled noise analysis for full-chip RC interconnect networks, *Proc. Intl. Conf. Computer-Aided Design*, pp. 139-146, 1997.
40. J. Lohstroh, Static and dynamic noise margins of logic circuits, *IEEE J. Solid-State Circuits*, SC-14, 591-598, June 1979.
41. C.L. Ratzlaff, N. Gopal and L.T. Pillage, RICE: Rapid interconnect circuit evaluator, *IEEE Trans. Computer-Aided Design*, 13(6), 763-776, 1994.
42. P. Feldman and R.W. Freund, Efficient linear circuit analysis by Pade approximation via the Lanczos process, *IEEE Trans. Computer-Aided Design*, 14(5), 639-649, May 1995.
43. L.M. Elfadel and D.D. Ling, Block rational Arnoldi algorithm for multipoint passive model-order reduction of multiport RLC networks, *Proc. IEEE/ACM Intl. Conf. Computer-Aided Design*, pp. 66-71, Nov. 1997.
44. K.J. Kerns, I.L. Wemple, and A.T. Yang, Stable and efficient reduction of substrate model networks using congruence transforms, *Proc. IEEE/ACM Intl. Conf. Computer-Aided Design*, pp. 207-214, 1995.
45. A. Odabasioglu, M. Celik, and L.T. Pileggi, PRIMA: Passive reduced-order interconnect macromodeling algorithm, *Proc. Intl. Conf. Computer-Aided Design*, pp. 58-65, 1997.
46. A. Devgan, Efficient coupled noise estimation for on-chip interconnects, *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp. 147-151, Nov. 1997.
47. F. Dartu and L.T. Pileggi, Calculating worst-case gate delays due to dominant capacitance coupling, *Proc. ACM/IEEE Design Automation Conf.*, pp. 46-51, June 1997.
48. P. Gross, R. Arunachalam, K. Rajgopal, and L.T. Pileggi, Determination of worst-case aggressor alignment for delay calculation, *Proc. Intl. Conf. Computer-Aided Design*, pp. 212-219, Nov. 1998.
49. J.R. Black, Electromigration failure modes in aluminum metalization for semiconductor devices, *Proc. IEEE*, pp. 1587-1594, Sept. 1969.
50. S. Chowdhury and J.S. Barkatullah, Estimation of maximum currents in MOS IC logic circuits, *IEEE Trans. Computer-Aided Design*, 9(6), 642-654, June 1990.
51. H. Kriplani, F. Najm, and I. Hajj, Pattern independent maximum current estimation in power and ground buses of CMOS VLSI circuits, *IEEE Trans. Computer-Aided Design*, 14(8), 998-1012, Aug. 1995.
52. A. Krstic and K.T. Cheng, Vector generation for maximum instantaneous current through supply lines for CMOS circuits, *Proc. ACM/IEEE Design Automation Conf.*, pp. 383-388, 1997.
53. A. Dharchoudhury, R. Panda, D. Blaauw, R. Vaidyanathan, B. Tutuianu, and D. Bearden, Design and Analysis of power distribution networks in Power PC™ microprocessors, *Proc. ACM/IEEE Design Automation Conf.*, pp. 738-743, 1998.
54. D. Stark, Analysis of power supply networks in VLSI circuits, Research Report 91/3, Western Research Lab, Digital Equipment Corp., Apr. 1991.
55. G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press: Baltimore, MD, 1989.

Iyengar, V. "Microprocessor Design Verification"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

61

Microprocessor Design Verification

Vikram Iyengar
University of Illinois at Urbana-Champaign

Elizabeth M. Rudnick
University of Illinois at Urbana-Champaign

61.1 Introduction

The task of verifying that a microprocessor implementation conforms to its specification across various levels of design hierarchy is a major part of the microprocessor design process. Design verification is a complex process which involves a number of levels of abstraction (e.g., architectural, RTL, and gate), several aspects of design (e.g., timing, speed, functionality, and power), as well as different design styles.[22] With the high complexity of present-day microprocessors, the percentage of the design cycle time required for verification is often greater than 50%.

The increasing complexity of designs has led to a number of approaches being used for verification. Simulation and formal verification are widely recognized as being at opposite ends of the design verification spectrum, as shown in Fig. 61.1.[6] Simulation is the process of stimulating a software model of the design in an enviornment that models the actual hardware system. The values of internal and output signals are obtained for a given set of inputs and are compared with expected results to determine whether

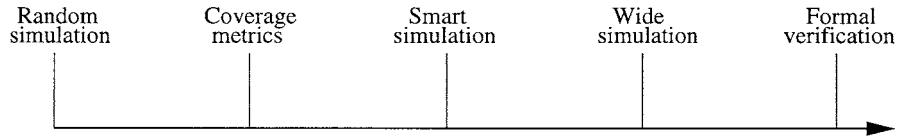


FIGURE 61.1 The spectrum of design verification techniques, which range from simulation to formal verification. (From Ref. 6. With permission.)

the design is behaving as specified. Formal verification, on the other hand, uses mathematical formulae on an abstracted version of the design to *prove* that the design is correct or that particular aspects of the design are correct.

Formal verification includes equivalence checking, model checking, and theorem proving. Equivalence checking verifies whether one description of a design is functionally equivalent to another. Model checking verifies that specified properties of a design are true, that is, that certain aspects of the design always work as intended. In theorem proving, the entire design is expressed as a set of mathematical assumptions. Theorems are expressed using these assumptions and are then proven. Formal verification is particularly useful at lower levels of abstraction, for example, to verify that a gate-level model matches its RTL specification. Formal verification is becoming popular as a means of achieving 100% coverage, at least for specific areas of the design, and is described more fully elsewhere in this book.

There are several problems inherent in applying formal verification to large microprocessor designs. While equivalence checking ensures that no functional errors are inserted from one design iteration to the next, it does not guarantee that the design meets the designer's specifications. Model checking is useful to check consistency with specifications; however, the assertions to be verified must be manually written in most cases. The size of the circuit or state machine that can be formally verified is severely limited due to the problem of state-space explosion. Last, formal techniques cannot be used for performance validation because timing-dependent circuits, such as oscillators, rely on analog behavior that is not handled by mathematical representations.

Simulation is therefore the primary commercial design verification methodology in use, especially for large microprocessor designs. Simulation is performed at various levels in the design hierarchy, including at the register transfer, gate, transistor, and electrical levels, and is used for both functional verification and performance analysis. Timing simulation is becoming critical for ultra-deep submicron designs because the problems of power grid IR-drops, interconnect delays, clock skews, and electromigration intensify with shrinking process geometries and adversely affect circuit performance.[23] Timing verification involves performing 2-D or 3-D parasitic RC extraction on the layout, followed by back-annotating the capacitance values obtained onto the netlist. A combination of static and dynamic timing analyses is performed to find critical paths in the circuit. Static analysis involves analyzing delays using a structural model of the circuit, while dynamic analysis uses vectors to simulate the design to locate critical paths.[23] Accurate measurements of the critical path delays can then be obtained using SPICE. Techniques for timing verification are described elsewhere in this book.

Pseudo-random vector generation is the most popular form of generating instruction sequences for functional simulation. Random test generators provide the ability to generate test programs that lead to multiple simultaneous events, which would be extremely time-consuming to write by hand.[18] Furthermore, the amount of computation required to generate random instruction sequences is low. However, random simulation often requires a very long time to achieve a suitable level of confidence in the design. This has given rise to the use of a number of semiformal metrics to estimate and improve simulation coverage. These methods combine the advantages of simulation and formal verification to achieve a higher coverage, while avoiding the scaling and methodology problems inherent in formal verification. In this chapter, we focus on the tools and techniques used to generate instruction sequences for a simulation-based verification environment.

The chapter is organized as follows. We begin with a description of the design verification environment in Section 61.2. Random and biased-random instruction generation, which lie at the simulation end of the spectrum, are discussed in Section 61.3. Section 61.4 describes three popular correctness checking methods that are used to determine the success or failure of a simulation run. Coverage metrics, which are used to estimate simulation coverage, are presented in Section 61.5. In Section 61.6, we move to the middle of the design verification spectrum and discuss *smart simulation*, which is used to generate vectors satisfying semiformal metrics. *Wide simulation*, which refers to the use of formal assertions to derive vectors for simulation is described in Section 61.7. Having covered the spectrum of semiformal verification methods, we conclude with a description of hardware emulation in Section 61.8. Emulation uses dynamically configured hardware to implement a design, which can be simulated at high speeds.

61.2 Design Verification Environment

In this section, we present a design verification environment that is representative of many commercial verification methodologies. This environment is illustrated in Fig. 61.2, and the typical levels of design abstraction are shown in Fig. 61.3. We describe the different parts of the environment and the role each part plays in the verification process.

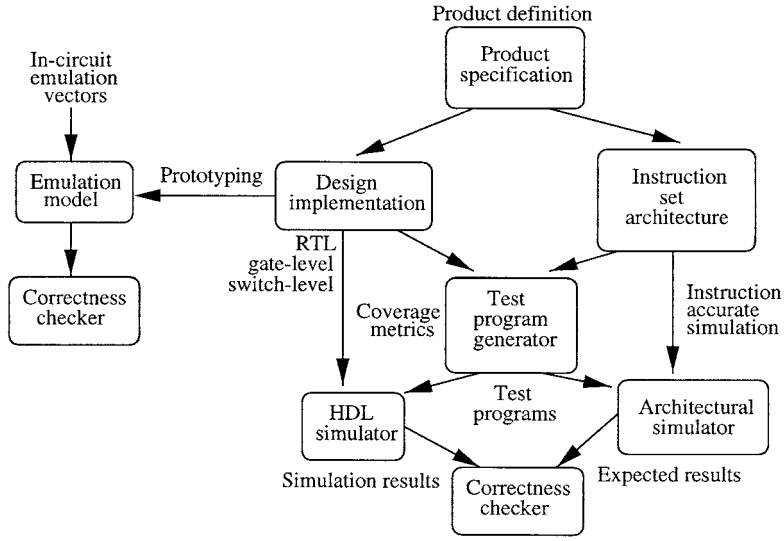


FIGURE 61.2 A representative design verification environment and verification process flow.

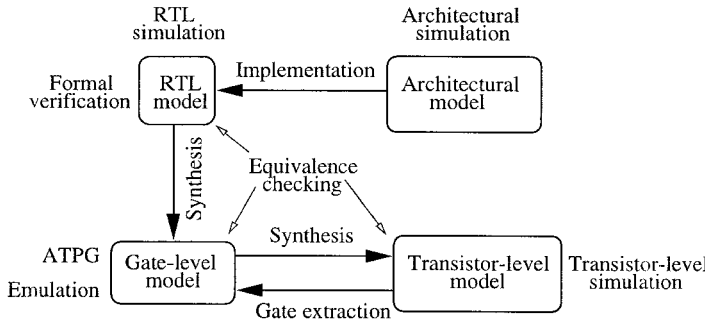


FIGURE 61.3 Different levels of design abstraction.

Architectural Model

A high-level specification of the microprocessor is first derived from the product requirements and from the requirement of compatibility with previous generations. An architectural simulation model and an RTL model are then implemented based on the product specification. The architectural model, often written in C or C++, includes the programmer-visible registers and the capability to simulate the execution of an instruction sequence. This model emphasizes simulation speed and correctness over implementation detail and therefore does not represent pipeline stages, parallel functional units, or caches. This model is instruction accurate but not clock cycle accurate.[22] A typical architectural model executes over 100 times faster than a detailed RTL model.[18]

RTL Model

The RTL model, implemented in a hardware description language (HDL) such as VHDL or Verilog, is more detailed than the architectural model. Data is stored in register variables, and transformations are represented by arithmetic and logical operators. Details of pipeline implementation are included. The RTL model is used to synthesize a gate-level model of the design, which may be used to formally verify equivalence between the RTL and transistor-level implementations or for automatic test pattern generation (ATPG) for manufacturing tests. Circuit extraction can also be performed to derive a gate-level model from the transistor-level implementation. In many methodologies, the RTL represents the *golden model* to which other models must conform. Equivalence checking is commonly used to verify the equivalence of RTL, gate-level, and transistor-level implementations.

Test Program Generator

The combination of simulation and formal methods is an emerging paradigm in design verification. A test program generator may therefore use a combination of random, hand-crafted, and deterministic instruction sequences generated to satisfy certain semiformal measures of coverage. These measures include the coverage of statements in the HDL description and coverage of transitions between control states in the design's behavior. The RTL model is simulated with these test vectors using an HDL simulator, and the results are compared with those obtained from the architectural simulation. Since the design specification (architectural-level) and design implementation (RTL or gate-level) are at different levels of abstraction, there can be no cycle-equivalent comparison. Instead, comparisons are made at special checkpoints, such as at the completion of a set of instructions.[12] Sections 61.3, 61.6, and 61.7 discuss the most popular techniques used for test generation.

HDL Simulator

HDL simulation consists of two stages. In the *compile* stage, the design is checked for errors in syntax or semantics and is converted to an intermediate representation. The design representation is then reduced to a collection of signals and processes. In the *execute* stage, the model is simulated by initializing values on signals and executing the sequential statements belonging to the various processes. This can be achieved in two ways: *event-driven* simulation and *cycle-based* simulation. Event-driven simulation is based on determining changes (events) in the value of each signal in a clock cycle and may incorporate various timing models. A process is first simulated by assigning a change in value to one or more of its inputs. The process is then executed, and new values for other signals are calculated. If an event occurs on another signal, other processes that are sensitive to that signal are executed. Events are processed in the order of the time at which they are expected to occur according to the timing model used. In this manner, all events occurring in a clock cycle are calculated. Cycle-based simulators, on the other hand, limit calculations by determining simulation results only at clock edges and ignoring inter-phase timing. Cycle-based simulators focus only on the design functionality by performing zero-delay, two-valued simulation (memory elements are assumed to be initialized to known values) and they offer an improvement

in speed of up to 10X while utilizing a fifth of the memory required for event-driven simulation. However, cycle-based simulators are inefficient in verifying asynchronous designs, and event-driven simulators must be used to derive initializing sequences and for timing calculations. Simulation techniques used at various levels of design abstraction are discussed more fully in this book.

Emulation Model

Hardware emulation is a means of embedding a dynamically configured prototype of the design in its final environment. This hardware prototype, known as the *emulation model*, is derived from the gate-level implementation of the design. The prototype can execute both random vectors and software application programs faster than conventional software logic simulators. It is also connected to a hardware environment, known as an *in-circuit* facility, to provide it with a high throughput of test vectors at appropriate speeds. Hardware emulation executes from three to six orders of magnitude faster than simulation and subsequently requires considerably less verification time. However, hardware emulators have limitations on the sizes of the circuits they can handle.

Table 61.1 presents the results of a survey conducted by 0-In Design Automation on verification techniques currently used in industry.[1] Columns 1 and 3 in the table list the different techniques, while columns 2 and 4 show the percentage of surveyed engineers currently using a particular approach. While formal methods are becoming popular as a means to more exhaustively cover the design, psuedo-random simulation is still a vital part of the verification engineer's repertoire. In Section 61.3 we review some conventional verification techniques that use psuedo-random and biased-random test programs for simulation.

TABLE 61.1 0-In Bug Survey Results: Percentages of Various Validation Techniques Used by Design Verification Engineers (May 1997–May 1998)

Stimulus Techniques	Percentage Use	Advanced Verification Techniques	Percentage Use
System stimulation	94%	Cycle-based simulation	25%
Directed tests	89%	Equivalence checking	19%
Regression tests	88%	Hardware/software co-design	15%
Pseudorandom	82%		
Prototype silicon	58%	Model checking	13%
Emulation	49%		

Source: From Ref. 1. With permission.

61.3 Random and Biased-Random Instruction Generation

Random vector simulation is the primary verification methodology used for microprocessors today. New designs, as well as changes made to existing designs, are subjected to a battery of simulation and regression tests involving billions of pseudo-random vectors before focused testing is performed. Random test generation, also known as black-box testing, produces more complex combinations of instructions than can be manually written by the design verification engineer. A large number of test programs are generated randomly. Each test program consists of a set of register and memory initializations and a sequence of instructions. It may also contain the expected contents of the registers and memory after execution of the instructions, depending on the implementation. The expected contents of the registers and memory are obtained using an architectural model of the design. The test programs are translated to assembler or machine-language code that is supported by the HDL simulator, and are simulated on the RTL model. However, purely random test programs are not ideal because the instruction sequences developed may not exercise a sufficient number of corner cases; thus, millions of vectors and days of simulation are

required before reasonable levels of coverage can be achieved. In addition, random vectors may violate constraints on memory addressing, thus causing invalid instruction execution.

Biased-Random Testing

Biasing is the manipulation of the probability of selecting instructions and operands during instruction generation. Biased-random instruction generation is used to create test programs that have a higher probability of leading to execution hazards for the processor. For example, the biasing scheme in Ref. 24 utilizes knowledge of the Alpha 21264 architecture to favor the generation of instructions that test architecture-specific corner cases, specifically those affecting control-flow, out-of-order processing, superscalar structures, cache transactions, and illegal instructions.

Constraint solving, another biasing technique, identifies output conditions or intermediate values that are important to verify.[4] The instruction generator identifies input values that would lead to these conditions and generates instructions that utilize these "biased" input values. Constraint solving is useful because it improves the probability of exercising certain corner cases. Both of these schemes have biases hard-coded into the test generation algorithm based on the instruction type.

Static and Dynamic Biasing

Biasing can be classified as being either static or dynamic. *Static biasing* of test vectors involves randomly initializing the registers and memory, generating the biased-random test program and applying it to the architectural and RTL models (e.g., the RIS tool from Motorola[8]). A major complication of this method is that the test generator must construct a test that does not violate the acceptable ranges for data and memory addresses. The solution to this problem is to constrain biasing within a restricted set of choices that define a constrained model of the environment; for example, to reserve certain registers for indexed addressing.[22]

Dynamically-biased test generators use knowledge of the current processor states, memory state, and user bias preferences to generate more effective test programs. In dynamic instruction generation, the states of the programmer model in the test generator are updated to reflect the execution of the instruction after each instance of instruction generation.[2,4] The test generator interacts with a tightly coupled functional model of the design to update current state information.

Drawbacks of random and biased-random testing include the vast amount of simulation time required to achieve acceptable levels of coverage and the lack of effective biasing methodologies. Determining when an acceptable level of coverage has been achieved is a major concern. Semiformal verification techniques have therefore become popular as a means to monitor simulation coverage, as well as improve coverage by generating vectors to cover test cases that have not been exercised by random simulation.

In Section 61.4, we discuss several correctness checking techniques that are used to determine whether the simulation test was successful. Later, in Section 61.5, we review some of the common metrics used to evaluate the coverage of test programs.

61.4 Correctness Checking

Correctness checking is the process of isolating a design error by determining whether the simulation test was successful. In this section, we discuss three techniques for correctness checking: self-checking, reference model comparison, and assertion checking. The three methods are complementary and are often used in conjunction to achieve the highest coverage. Figure 61.4 illustrates the three correctness checkers in the verifiction flow of the Alpha 21164 microprocessor.[18]

Self-checking

Self-checking is the simplest way to determine success for focused, hand-coded tests. The test program sets up a combination of conditions and then checks to see if the RTL model reacted correctly to the

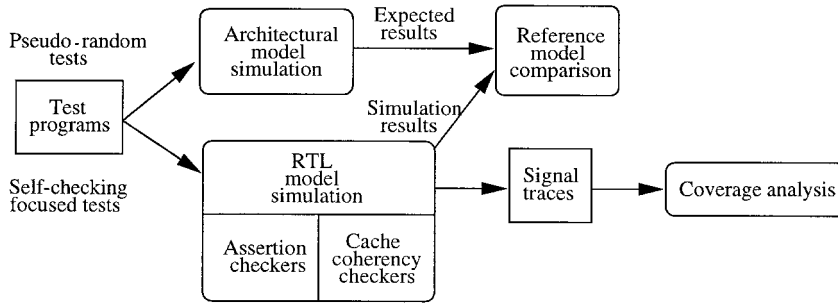


FIGURE 61.4 Verification flow and correctness checking for the Alpha 21164 microprocessor. (From Ref. 18.)

simulated situation.[15] However, this approach is time-consuming, prone to error, and intrusive at the register transfer level. The test generator may be required to maintain an extensive amount of state information. Furthermore, the technique is often not useful beyond a single focused test.

Reference Model Comparison

An alternative to self-checking is to compare the traces generated by the RTL model with the simulation traces of an architectural reference model, as illustrated in Fig. 61.2. This technique, known as reference model comparison, obviates the need for constantly checking the state of the processor being simulated. The reference model is an abstraction of the design architecture written in a high-level language such as C++. It represents all features visible to software, including the instruction set and support for memory and I/O space.[18]

Several correctness checks may be performed using the reference model, of which the simplest is *end-of-state* comparison. When simulation completes, the contents of memory locations are accessed, and the final states of the register files are compared. However, end-of-state comparison is not very useful for lengthy simulations because it may be difficult to identify incorrect intermediate results, which are overwritten during simulation. Comparing intermediate results during simulation is a solution; however, this requires the reference model to match the timing of the RTL model, and is not easily implemented. Additional comparisons that can be made include checking the PC flow and checking writes to integer and floating-point registers. Incorrect values here will signal problems with control-flow and data manipulation instructions.

Assertion Checking

Assertion checking, another popular means to check correctness, is the process of adding segments of code to the RTL model to verify that certain properties of design behavior always hold true under simulation. Examples of simple assertion checking include monitoring illegal states and invalid transitions. More complex checking involves monitoring queue overruns and violation of the bus protocol.[24] An example of a specialized assertion checker is the cache coherency checker used in the verification of the Alpha 21164 microprocessor.[18] The system supports three levels of caching, with the second and third-level caches being writeback. Cache coherency checking was activated at regular intervals during simulation to ensure that coherency rules were not violated. Table 61.2 presents the origins of bugs introduced into the design and the percentages of bugs detected by the various correctness checking mechanisms for the Alpha 21264 microprocessor.[24] Assertion checkers were the most successful; however, when viewed collectively, 44% of errors were found by reference model comparison.

With the correctness checking problem examined, the next major issue in simulation-based verification is determining whether acceptable levels of coverage have been achieved by the simulation vectors. In the next section, we look at several techniques for coverage analysis.

TABLE 61.2 Effectiveness of Correctness Checks Used in the Verification of the Alpha 21264 Microprocessor

Origin of Bug	Bugs Introduced[a]	Correctness Checker	Bugs Detected[a]
Implementation error	78%	Assertion checker	25%
Programming mistake	9%	Register miscompare	22%
Matching model to schematics	5%	Simulation "no progress"	15%
Architectural conception	3%	PC miscompare	14%
Other	5%	Memory state miscompare	8%
		Manual inspection	6%
		Self-checking test	5%
		Cache coherency check	3%
		SAVES check	2%

[a] Percentage of total design erros.
Source: From Ref. 24. With permission.

61.5 Coverage Metrics

Coverage analysis provides information on how thoroughly a design has been exercised during simulation. Coverage metrics are used to evaluate the effectiveness of random simulation vectors, as well as guide the generation of deterministic tests. A number of coverage metrics have been proposed, and verification engineers often use a variety of metrics simultaneously to determine test completeness. The simplest metrics used are based on the HDL description of the design. Examples are statement coverage, conditional branch coverage, toggle coverage, and path coverage.[6,18]

HDL Metrics

Statement coverage determines the number of statements in the HDL description that are executed. Conditional branch coverage metrics compute the number of conditional expressions that are toggled during simulation. Each of the important conditional expressions (e.g., **if** and **case** statements) is assigned to a monitored variable. If the variable is assigned to both 0 and 1 during simulation, both paths of the conditional branch are considered activated. Toggle coverage is the ratio of the number of signals that experienced 1-to-0 and 0-to-1 transitions during simulation, to the total number of effective signals. The number of effective signals is adjusted to include only those that can possibly be toggled in the fault-free model. Another recently proposed HDL metric is based on error observability at the primary outputs of the design.[7] Observability is computed by tagging variables during simulation and checking whether the tags are propagated to the outputs. A tag calculus, similar to that of the D-algorithm used for ATPG, is introduced. Coverage is measured as the percentage of tags visible at the design outputs. The method provides a stricter measure of coverage than does HDL-line coverage. However, while HDL-based metrics are useful, they are generally not effective measures of whether logic is being functionally exercised.

Manufacturing Fault Models

A second class of coverage metrics is based on manufacturing fault models.[25,26] These metrics characterize a class of design errors analogous to faults in hardware testing and measure coverage through fault simulation. Logic design errors, such as gate substitution, missing gates, and extra inverters, are injected randomly into the design. The design is then simulated using a stuck-at fault simulator. This approach has been used for measuring the coverage of ATPG tests for embedded arrays in microprocessors. ATPG is the process of automatically generating test patterns for manufacturing tests and is typically performed at the gate level. The problems with using manufacturing fault models to estimate coverage are that fault simulation is often computationally intensive, and faults introduced during the manufacturing process do not always model design errors.

Sequence and Case Analysis

Other metrics widely used in industry include sequence analysis, occurrence analysis, and case analysis.[18,24] *Sequence* analysis monitors sequences of events during simulation, for example, request-acknowledge sequences, interrupt assertions, and traps. *Occurrence* analysis determines the presence of one-time events, such as a carry-out being generated for every bit in an adder. The absence of such an event could signal a failure. Finally, *case* analysis consists of collecting and studying simulation statistics, such as exerciser type, cycles simulated, issue rate, and instruction types.[24]

State Machine Coverage

A more formal way to evaluate coverage is to look at an abstraction of the design in the form of a finite state machine (FSM).[5,12,14,20] The control logic of the design is extracted as an FSM, which has a smaller state space than the original design but which exhibits the design's control behavior. Coverage is typically estimated by the fraction of different states reached by the FSM or the fraction of state transitions exercised during simulation. FSM coverage metrics are also used to generate test programs with high coverage, as described in Section 61.6. Binary decision diagrams (BDDs), borrowed from formal verification, are used to describe and traverse the implementation state space. A BDD is a way of efficiently representing a set of binary-valued decisions (scalar Boolean functions) in the form of a tree or directed acyclic graph.

A method of transforming the high-level description of the circuit into a reduced FSM that has far fewer states than the original design, is proposed in Ref. 5. Simulation coverage is estimated by relating the fraction of transitions in the state graph traversed by this reduced FSM, to the number of HDL statements exercised in the high-level description.

More recently, Moundanos et al.[20] have described the extraction of the control logic of the design from its HDL description. The control logic is extracted in the form of an FSM, which represents the control space of the entire circuit. The vectors whose coverage is to be evaluated are simulated on the FSM. Simulation coverage is estimated by the following two ratios.

$$\text{State coverage metric}\left(\text{SCM}\right) = \frac{\text{Number of states visited}}{\text{Total number of reachable states}}$$

$$\text{Transition coverage metric}\left(\text{TCM}\right) = \frac{\text{Number of transitions visited}}{\text{Total number of reachable transitions}}$$

A similar approach to evaluating coverage is used in Ref. 14. Since only a subset of state variables directly controls the datapath, the non-controlling independent state variables are removed from the state graph of the FSM. This reduced state graph is called a *control event* graph, and each reachable state is a control event. Coverage is evaluated in terms of the number of control events visited and the number of transitions exercised in the control event graph.

Further along the spectrum toward formal verification lie techniques dubbed "smart simulation,"[6] which not only evaluate coverage of the given vectors, but also *generate* new functional tests using coverage metrics. In Section 61.6, we discuss several such techniques that use semiformal coverage metrics to derive simulation vectors. We begin with techniques based on identifying hazard patterns and later discuss more formal methods that use state machine coverage.

61.6 Smart Simulation

Deterministic or *smart* simulation uses vectors that cover a certain aspect of the design's behavior using details of its implementation. We first describe ad hoc techniques, such as hazard-pattern enumeration, which target specific blocks in the processor, and then describe more general techniques aimed at verifying the entire chip.

Hazard-Pattern Enumeration

Ad hoc techniques typically target a specific block in the design, such as a pipeline[16,19] or cache controller.[21] Errors in the pipeline mechanism represent only a small fraction of the total errors. In a study undertaken in Ref. 19, it was shown that only 2.79% of the total errors in a commercial 32-bit CPU design were related to the pipeline interlock controller. However, these errors are widely acknowledged as being the hardest to detect and are therefore targeted by ad hoc methods.

Pipeline hazards are situations that prevent the next instruction from executing in its designated clock cycle. These are classified as structural hazards, data hazards, and control hazards. *Structural* hazards occur when two instructions in different pipeline stages attempt to access the same physical resource simultaneously. *Data* hazards are of three types: read-after-write (RAW), write-after-write (WAW), and write-after-read (WAR) hazards. The most common are RAW hazards, in which the second instruction attempts to read the result of the first instruction before it is written. *Control* hazards are treated as RAW hazards in the program counter (PC). An algorithm that enumerates all the structural, data, and control hazard patterns for each common resource in the pipeline is presented in Ref. 16. Test programs that include all the patterns that can cause the pipeline to stall are then generated.

Lee and Siewiorek[19] define the set of state variables read by an instruction as its *read state* and the set written by the instruction as its *write state*. A conflict exists between two instructions if at least one of them is a write and the intersection of their read/write or write/write states is not empty. A dependency graph is constructed with nodes representing all the possible read/write instructions and edges (or dependency arcs) representing conflicts between instructions. Test programs are generated to cover all the dependency arcs in the graph, and the dependency arc coverage is calculated.

In Ref. 21, a cache controller is verified using a model of the memory hierarchy, a set of cache coherence protocols, and enumeration capabilities to generate test programs for the design.

The problem inherent in ad hoc techniques is that pipeline behavior after the detection of a hazard is usually not considered.[17] Test cases reachable only after a hazard has occurred are therefore not covered. We next discuss more general test generation techniques, which are applicable to a larger part of the design.

ATPG

An important class of verification techniques is based on the use of test programs generated by ATPG tools. Coverage is measured as the fraction of design errors detected. These methods have been used in industry to verify the equivalence between the gate-level and transistor-level models; for example, in the verification of PowerPC™ arrays.[25,26] In this approach, a gate-level model is created from the transistor-level implementation, and tests generated at the gate level are simulated at the transistor level to verify equivalence. However, these techniques, while effective at lower levels of abstraction, do not provide a good measure of the extent to which the design has been exercised.

State and Transition Traversal

Tests generated by traversing the design's state space work on the principle that verification will be close to complete if the processor either visits all the states or exercises all the transitions of its state graph during simulation.[5,13,20] Since memory limitations make it impossible to examine the state graph of the entire design, the design behavior is usually abstracted in the form of a reduced state graph. Test sequences are then generated which cause this reduced FSM to exercise all the transitions. Figure 61.5 illustrates the verification flow for this technique. The first step is to extract the control logic of the design in the form of an FSM. The datapath is usually not considered because most designs have datapaths of substantial size, which can lead to an unmanageable state space. Furthermore, errors in the datapath usually result from incorrect implementation — not incorrect design — and can be easily tested by conventional simulation.[20]

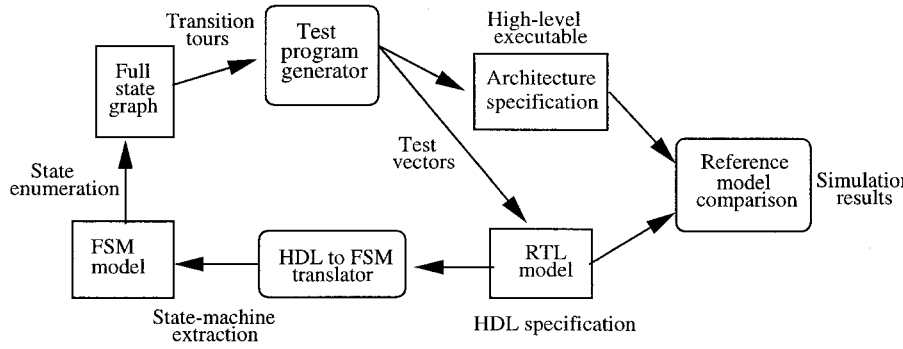


FIGURE 61.5 Verification flow for a representative state machine traversal technique. (From Ref. 13. With permission.)

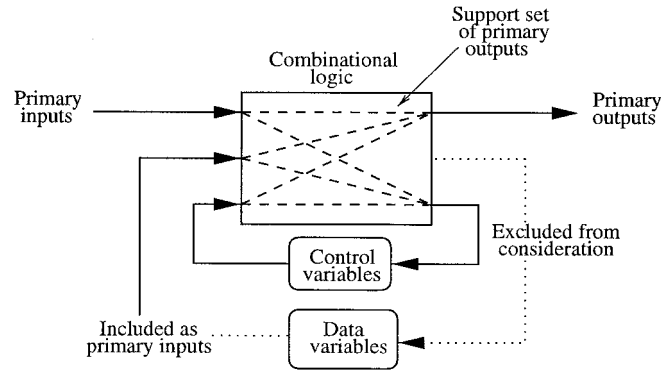


FIGURE 61.6 Extraction of the control flow machine. (From Ref. 20. With permission.)

A method to extract the control logic of the design in the form of an FSM can be found in Ref. 20. This is illustrated in Fig. 61.6. The data variables in the design are made nondeterministic by including them in the set of primary inputs to the FSM. Since the datapath is to be excluded from consideration, the inputs to the data variables are excluded. This is represented by the dotted lines in Fig. 61.6. The support set of the primary outputs and control state variables is now determined in terms of the primary inputs, control state variables, and data variables. This support set forms the new set of primary inputs to the FSM. Data variables that are not a part of the support set are excluded from the FSM. In this manner, the effect of the data variables on the control flow is taken into account, even though the data registers are abstracted.

After the FSM has been extracted, state enumeration is performed to determine the reachable states, and a state graph which details the behavior of the FSM is generated. Since coverage is typically evaluated by the number of states visited or the number of transitions exercised, a *state* or *transition tour* of the state graph is found. A state (transition) tour of a directed state graph is a sequence of transitions that traverses every state (transition) at least once. Several polynomial-time algorithms have been developed for finding transition tours in nonsymmetric, strongly connected graphs, since this problem (the Chinese Postman problem) is frequently encountered in protocal conformance testing.[13] The transition tour is translated into an instruction sequence which will cause the FSM to exercise all transtions.

Cheng and Krishnakumar[5] use exhaustive coverage of the reduced FSM to generate test programs guaranteeing that all statements in the original HDL representation are exercised. A test generation technique based on visiting all states in the state graph is presented in Ref. 17. Test cases are developed based on enumerating hazard patterns in the pipeline and are translated into sequences of states in the

state graph. Simulation vectors that satisfy all test cases are generated. A more general transition-traversal technique is given in Ref. 13. A translator is used to convert the HDL representation to a set of interacting FSMs. A full state enumeration of the FSMs is performed to find all reachable states from reset. This produces a complete state graph, which is used to generate vectors that cause the processor to take a transition tour.

Finally, several classes of processors for which transition coverage is effective are identified in Ref. 12. The authors demonstrate that under a given set of conditions, transition tours of the state graph can be used to completely validate a large class of processors.

State-space explosion is currently a key problem in computing state machine coverage. As designs get larger and considerably more complex, the maximum size of the state machine that can be handled is the major limiting factor in the use of formal methods. However, research is currently being undertaken to deal with state explosion, and we foresee an increasing use of formal coverage metrics in the future.

61.7 Wide Simulation

Near the formal end of the verification spectrum, *wide simulation* is performed by representing the FSM behavior as a set of transitions between valid control states and symbolically representing large sets of states in relatively few simulations. Assertions covering all the transitions in the state graph are written and are used to derive vectors for simulation.

Partitioning FSM Variables

The authors in Ref. 11 first focus on specific parts of the design by partitioning the FSM variables into three sets — *coverage Co, ignore Ig,* and *care CA* — based on their respective importance. Using these sets, the number of transitions in the graph that need to be exercised can be reduced.

For example, a state in the FSM is viewed as the 3-tuple $\{X, Y, Z\}$, where $X \in Co$, $Y \in Ig$, and $Z \in Ca$. Two transitions, $T_1((X_1Y_1Z_1), (X_2Y_2Z_2))$ and $T_2((X_3Y_3Z_3), (X_4Y_4Z_4))$, which differ in the value of a coverage variable, are distinct and require separate tests; for example, if $X_1 \neq X_3$ or $X_2 \neq X_4$, then T_1 and T_2 require different tests. However, two transitions that differ only in the value of an ignore variable are equivalent. Therefore, if $X_1 = X_3$, $X_2 = X_4$, $Z_1 = Z_3$, and $Z_2 = Z_4$, then T_1, T_2 are equivalent and a vector that tests T_1 will also test T_2. Finally, two transitions that differ in the value of a care variable do not necessarily require different tests.[11]

In this manner, the state graph is represented as the set of all valid transitions T, of which only a few must be exercised, based on the equivalence relations. Next, formal assertions are written for each transition. An assertion is a temporal logic expression of the form *antecedent* $\rightarrow$ *consequent*, where both antecedent and consequent can consist of complex logical expressions.[25] The first step in the test generation process is to choose a valid transition $T_1(v,v') \in T$ and write an assertion of the form $state(v) \rightarrow next(\neg state(v'))$, which means that if the FSM is in state v, then the next state cannot be v'.

Deriving Simulation Tests from Assertions

A model checker can be used to generate sequences of input vectors which satisfies the assertion.[11] A *model checker* is a formal verification tool that is used to either prove that a certain property is satisfied by the system or generate a counterexample to show that the property does not always hold true. The model checker reports that the assertion $state(v) \rightarrow next(\neg state(v'))$ is false and that the transition is indeed valid. The model checker also outputs a symbolic sequence of states and input patterns which lead to state v. This symbolic (high-level) sequence of patterns is then translated into a simulation vector sequence and is used to verify the design. The transition T_1 and all transitions equivalent to T_1 are removed from T, and the process is repeated.[11]

Wang and Abadir[25,26] use tools to automatically generate formal assertions for PowerPC™ arrays from the RTL model. *Symbolic trajectory evaluation*, a descendant of symbolic simulation, is used to formally

prove that all assertions are true. After the design has been formally verified, simulation vectors are derived from the assertions and are used for simulating the design. The methods used to derive these vectors are as follows. The symbolic values used in the antecedent of each assertion are replaced with a set of vectors based on each condition specified in the consequent. First, symbolic address comparison expressions are replaced with address marching sequences (e.g., to test large decoders). Next, symbolic data comparison expressions are replaced with data marching sequences (e.g., in testing comparators). Stand-alone symbolic values representing sets of states or input patterns are replaced with random vectors. Assertion decision trees are constructed and tests are generated to cover all branches (e.g., in testing control logic). Finally, control signal decision trees are constructed in order to generate tests that cover abnormal functional space.[25]

We have now reached the "formal" end of our discussion on verification techniques, which range from random simulation to semiformal verification. Formal verification, which uses mathematical formulae to prove correctness, is described by Levitan. In Section 61.8, we describe emulation, which is a means to implement a design using programmable hardware, with performance several orders of magnitude faster than conventional software simulators. Emulation has become popular as a means to test a processor against real-world application programs, which are impossibly slow to run using simulation.

61.8 Emulation

The fundamental difference between simulation and emulation is that simulation models the design in software on a general-purpose host computer, while emulation actually implements the design using dynamically configured hardware. Emulation, performed in addition to simulation has several advantages. It provides up to six orders of magnitude improvement in execution performance and enables several tests that are too complex to simulate to be performed prior to tapeout. These include power-on-self-tests, operating system boots, and running software applications (e.g., Open Windows).[10] Finally, emulation reduces the number of silicon iterations that are needed to arrive at the final design, because errors caught by emulation can be corrected before committing the design to silicon.

Pre-configuration

The emulation process consists of four major phases: pre-configuration, configuration, testbed preparation, and in-circuit emulation (ICE).[10] In the pre-configuration phase, the different components of the design are assembled and converted into a representation that is supported by the emulation vendor. For example, in the K5 emulation, each custom library cell was expressed in terms of primitives that could be mapped to a field-programmable gate array (FPGA).[9] An FPGA is a simple programmable logic device that allows users to implement multi-level logic. Several thousand FPGAs must be connected together to prototype a complex microprocessor. Once the cell libraries have been translated, the various gate-level netlists are converted to a format acceptable to the configuration software. This can be complicated because the netlists obtained from standard-cell and datapath designers are often in a variety of formats.[10]

There is often no FPGA equivalent for complex transistor-level megacells, which are commonly used in full custom processors. Gate-level emulation models for megacells must therefore be created. These gate-level blocks are implemented in the programmable hardware and are verified against simulation vectors to ensure that each module performs correctly according to the simulation model.

Full-Chip Configuration

In this phase, the design netlists and libraries are combined with control and specification files and downloaded to program the emulation hardware. In the first stage of configuration, the netlists are parsed for semantic analysis and logic optimization.[10] The design is then partitioned into a number of logic board modules (LBMs) in order to satisfy the logic and pin constraints of each LBM. The logic assigned to each LBM is flattened, checked for timing and connectivity and further partitioned into clusters to

allow the mapping of each cluster to an individual FPGA.[9] Finally, the interconnections between the LBMs are established, and the design is downloaded to the emulator.

Testbed and In-circuit Emulation

The testbed is the hardware environment in which the design to be emulated will finally operate. This consists of the target ICE board, logic analyzer, and supporting laboratory equipment.[10] The target ICE board contains PROM sockets, I/O ports, and headers for the logic analyzer probes.

Verification takes place in two modes: the simulation mode and ICE. In the simulation mode, the emulator is operated as a fast simulator. Software is used to simulate the bus master and other hardware devices, and the entire simulation test suite is run to validate the emulation model.[9] An external monitor and logic analyzer are used to study results at internal nodes and determine success. In the ICE mode, the emulator pins are connected to the actual hardware (application) environment. Initially, diagnostic tests are run to verify the hardware interface. Finally, application software provides the emulation model with billions of vectors for high-speed functional verification.

In Section 61.9, we conclude our discussion on design verification and review some of the areas of current research.

61.9 Conclusion

Microprocessor design teams use a combination of simulation and formal verification to verify pre-silicon designs. Simulation is the primary verification methodology in use, since formal methods are applicable mainly to well-defined parts of the RTL or gate-level implementation. The key problem in using formal verification for large designs is the unmanageable state space.

Simulation typically involves the application of a large number of psuedo-random or biased-random vectors in the expectation of exercising a large portion of the design's functionality. However, random instruction generation does not always lead to certain highly improbable (corner case) sequences, which are the most likely to cause hazards during execution. This has led to the use of a number of semiformal methods, which use knowledge-derived from formal verification techniques to more fully cover the design behavior. For example, techniques based on HDL statement coverage ensure that all statements in the HDL representation of the design are executed at least once. At a more formal level, a state graph of the design's functionality is extracted from the HDL description, and formal techniques are used to derive test sequences that exercise all transitions between control states. Finally, formal methods based on the use of temporal logic assertions and symbolic simulation can be used to automatically generate simulation vectors. We next describe some current directions of research in verification.

Performance Validation

With an increasing sophistication in the art of functional validation, ensuring the lack of performance bugs in microprocessors has become the next focus of verifiction. The fundamental hurdle to automating performance validation for microprocessors is the lack of formalism in the specification of error-free pipeline execution semantics.[3] Current validation techniques rely on focused, handwritten test cases with expert inspection of the output. In Ref. 3, analytical models are used to generate a controlled class of test sequences with *golden* signatures. These are used to test for defects in latency, bandwidth, and resource size coded into the processor model. However, increasing the coverage to include complex, context-sensitive parameter faults and generating more elaborate tests to cover the cache hierarchy and pipeline paths remain open problems.

Design for Verification

Design for verification (DFV) is the new buzzword in microprocessor verification today. With the costs of verification becoming prohibitive, verification engineers are increasingly looking to designers for

easy-to-verify designs. One way to accomplish DFV is to borrow ideas from design for testability (DFT), which is commonly used to make manufacturing testing easier. Partitioning the design into a number of modules and verifying each module separately is one such popular DFT technique. DFV can also be accomplished by adding extra modes to the design behavior, in order to suppress features such as out-of-order execution during simulation. Finally, a formal level of abstraction, which expresses the microarchitecture in a formal language that is amenable to assertion checking, would be an invaluable aid to formal verification.

References

1. 0-In Design Automation: Bug Survey Results, *http://www.In.comsurvey_results.html.*
2. A. Aharon, A. Bar-David, B. Dorfman, E. Gofman, M. Leibowitz, and V. Schwartzburd, Verification of the IBM RISC system/6000 by a dynamic biased pseudo-random test program generator, *IBM Systems Journal*, vol. 30, no. 4, pp. 527, 1991.
3. P. Bose, Performance test case generation for microprocessors, *Proc. VLSI Test Symp.*, pp. 54, 1998.
4. A. Chandra, V. Iyengar, D. Jameson, R. Jawalekar, I. Nair, B. Rosen, M. Mullen, J. Yoon, R. Armoni, D. Geist, and Y. Wolfsthal, AVPGEN–A test generator for architecture verification, *IEEE Trans. on Very Large Scale Integrated Systems*, vol. 3, no. 2, pp. 188, June 1995.
5. K-T Cheng and A.S. Krishnakumar, Automatic functional test generation using the extended finite state machine model, *Proc. Design Automation Conf.*, pp. 86, 1993.
6. D.A. Dill, What's between simulation and formal verification?, *Proc. Design Automation Conf.*, pp. 328-329, 1998.
7. F. Fallah and S. Devadas, OCCOM: Efficient computation of observability-based code coverage metrics for functional verification, *Proc. Design Automation Conf.*, pp. 152, 1998.
8. J. Freeman, R. Duerden, C. Taylor, and M. Miller, The 68060 microprocessor function design and verification methodology, *Proc. On-Chip Systems Design Conf.*, pp. 10-1, 1995.
9. G. Ganapathy, R. Narayan, G. Jorden, D. Fernandez, M. Wang, and J. Nishimura, Hardware emulation for functional verification of K5, *Proc. Design Automation Conf.*, pp. 315, 1996.
10. J. Gateley et al., UltraSPARC™-I emulation, *Proc. Design Automation Conf.*, pp. 13, 1995.
11. D. Geist, M. Farkas, A. Landver, Y. Lichtenstein, S. Ur, and Y. Wolfsthal, Coverage-directed test generation using symbolic techniques, *Proc. Int. Test Conf.*, pp. 143, 1996.
12. A. Gupta, S. Malik, and P. Ashar, Toward formalizing a validation methodology using simulation coverage, *Proc. Design Automation Conf.*, pp. 740, 1997.
13. R.C. Ho, C.H. Yang, M.A. Horowitz, and D.A. Dill, Architecture validation for processors, *Proc. Int. Symp. on Computer Architecture*, pp. 404, 1995.
14. R.C. Ho and M.A. Horowitz, Validation coverage analysis for complex digital designs, *Proc. Int. Conf. on Computer Aided Design*, pp. 146, 1996.
15. A. Hosseini, D. Mavroidis, and P. Konas, Code generation and analysis for the functional verification of microprocessors, *Proc. Design Automation Conf.*, pp. 305, 1996.
16. H. Iwashita, T. Nakata, and F. Hirose, Integrated design and test assistance for pipeline controllers, IEICE *Trans. on Information and Systems*, vol. E76-D, no. 7, pp. 747, 1993.
17. H. Iwashita, S. Kowatari, T. Nakata, and F. Hirose, Automatic test program generation for pipelined processors, *Proc. Int. Conf. on Computer-Aided Design*, pp. 580, 1994.
18. M. Kantrowitz and L.M. Noack, I'm done simulating; now what? Verification coverage analysis and correctness checking of the DECchip 21164 Alpha microprocessor, *Proc. Design Automation Conf.*, pp. 325, 1996.
19. D.C. Lee and D.P. Siewiorek, Functional test generation for pipelined computer implementations, *Proc. Int. Symp. on Fault-Tolerant Computing*, pp. 60, 1991.
20. D. Moundanos, J.A. Abraham, and Y.V. Hoskote, Abstraction techniques for validation coverage analysis and test generation, *IEEE Trans. on Computers*, vol. 47, no. 1, pp. 2, Jan. 1998.

21. B. O'Krafka, S. Mandyam, J. Kreulen, R. Raghavan, A. Saha, and N. Malik, MTPG: A portable test generator for cache-coherent multiprocessors, *Proc. Phoenix Conf. on Computers and Communications*, pp. 38, 1995.
22. C. Pixley, N. Strader, W. Bruce, J. Park, M. Kaufmann, K. Shultz, M. Burns, J. Kumar, J. Yuan, and J. Nguyen, Commercial design verification: Methodology and tools, *Proc. Int. Test Conf.*, pp. 839, 1996.
23. R. Saleh, D. Overhauser, and S. Taylor, Full-chip verification of UDSM designs, *Proc. Int. Conf. on Computer-Aided Design*, pp. 254, 1998.
24. S. Taylor, M. Quinn, D. Brown, N. Dohm, S. Hildebrandt, J. Huggins, and C. Ramey, Functional verification of a multiple-issue, out-of-order, superscalar alpha processor — The Alpha 21264 microprocessor, *Proc. Design Automation Conf.*, pp. 638, 1998.
25. L.-C. Wang and M.S. Abadir, A new validation methodology combining test and formal verification for PowerPC™ microprocessor arrays, *Proc. Int. Test Conf.*, pp. 954, 1997.
26. L.-C. Wang and M.S. Abadir, Measuring the effectiveness of various design validation approaches for PowerPC™ microprocessor arrays, *Proc. Design in Automation and Test Europe*, pp. 273, 1998.

Karnik, T. "Microprocessor Layout Method"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

62

Microprocessor Layout Method

Tanay Karnik
Intel Corporation

62.1 Introduction

This chapter presents various concepts and strategies employed to generate a layout of a high-performance, general-purpose microprocessor. The layout process involves generating a physical view of the microprocessor that is ready for manufacturing in a fabrication facility (fab) subject to a given target frequency. The layout of a microprocessor differs from ASIC layout because of the size of the problem, complexity of today's superscalar architectures, convergence of various design styles, the planning of large team activities, and the complex nature of various, sometimes conflicting, constraints.

In June 1979, Intel introduced the first 8-bit microprocessor with 29,000 transistors on the chip with 8-MHz operating frequency.[1] Since then, the complexity of microprocessors has been closely following Moore's law, which states that the number of transistors in a microprocessor will double every 18 months.[2] The number of execution units in the microprocessor is also increasing with generations. The increasing die size poses a layout challenge with every generation. The challenge is further augmented by the ever-increasing frequency targets for microprocessors. Today's microprocessors are marching toward the GHz frequency regime with more than 10 million transistors on a die. Table 62.1 includes some statistics of today's leading microprocessors[1]:

TABLE 62.1 Microprocessor Statistics

Manufacturer	Part Name	# Transistors (millions)	Frequency (MHz)	Die Size (mm^2)	Technology (μm)
Compaq	Alpha 21264	15.2	600	314	0.35
IBM	PowerPC	6.35	250	66.5	0.3
HP	PA-8000	3.8	250	338	0.5
Sun	Ultrasparc-I	5.2	167	315	0.5
Intel	Pentium II	7.5	450	118	0.25

[1]The reader may refer to Refs. 3 through 10 for further details about these processors.

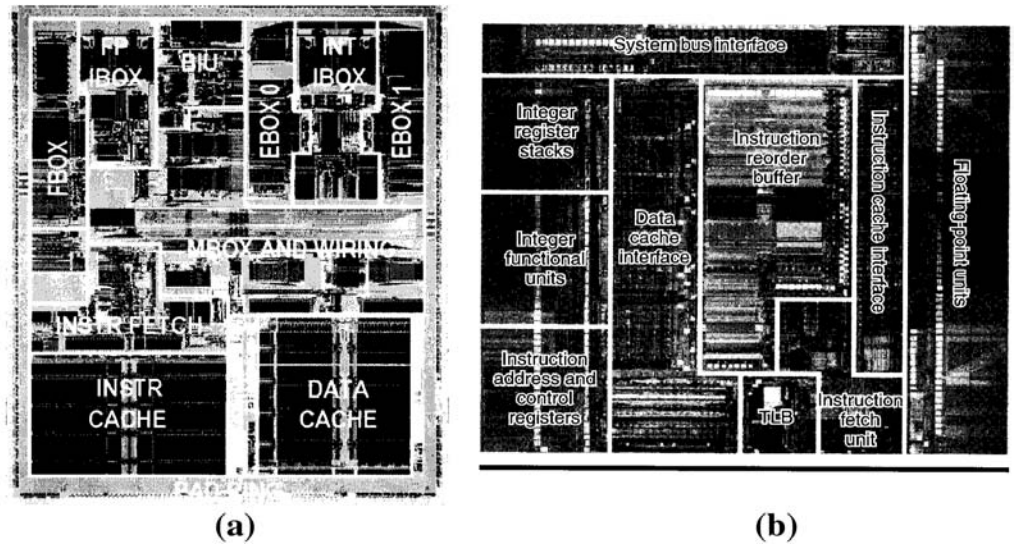


FIGURE 62.1 Chip micrographs. (a) Compaq Alpha 21264; (b) HP PA-8000. ((a) Courtesy of IEEE, Ref. 4; (b) Courtesy of IEEE, Ref. 6. With permission.)

In order to understand the magnitude of the problem of laying out a high-performance microprocessor, refer to the sample chip micrographs in Fig. 62.1. Various architectural modules, such as functional blocks, datapath blocks, memories, memory management units, etc. are physically separated on the die. There are many layout challenges apparent in this figure. The floorplanning of various blocks on the chip to minimize chip-level global routing is done before the layout of the individual blocks is available. The floorplanning has to fit the blocks together to minimize chip area and satisfy the global timing constraints. The floorplanning problem is explained in Section 62.4 (Floorplanning). As there are millions of devices on the die, routing power and ground signals to each gate involves careful planning. The power routing problem is described in Section 62.4 (Clock Planning). The microprocessor is designed for a particular frequency target. There are three key steps to high performance. The first step involves designing a high-performance circuit family, the second one involves design of fast storage elements, and the third is to construct a clock distribution scheme with minimum skew. Many elements need to be clocked to achieve synchronization at the target frequency. Routing the global clock signal exactly from an initial generator point to all of these elements within the given delay and skew budgets is a hard task. Section 62.4 (Power Planning) includes the description of clock planning and routing problems. There are various signal buses routed inside the chip running among chip I/Os and blocks. A 64-bit datapath bus is a common need in today's high-performance architectures, but routing that wide a bus in the presence of various other critical signals is very demanding, as explained in Section 62.4 (Bus Routing).

The problems identified by looking at the chip micrographs are just a glimpse of a laborious layout process. Before any task related to layout begins, the manufacturing techniques need to be stabilized and the requirements have to be modeled as simple design rules to be strictly obeyed during the entire design process. The manufacturing constraints are caused by the underlying process technology (Section 62.3, Technology Process) or packaging (Section 62.2, Packaging).

Another set of decisions to be taken before the layout process involve the circuit style(s) to be used during the microprocessor design. Examples of such styles include full custom, semi-custom, and automatic layout. They are described in Section 62.2. The circuit styles represent circuit layout styles, but there is an orthogonal issue to them, namely, circuit family style. The examples of circuit families include

static CMOS, domino, differential, cascode, etc. The circuit family styles are carefully studied for the underlying manufacturing process technology and ready-to-use cell libraries are developed to be used during the block layout. The library generation is illustrated in Section 62.5.

Major layout effort is required for the layout of functional blocks. The layout of individual blocks is usually done by parallel teams. The complex problem size prompts partitioning inside the block and reusability across blocks. Cell libraries as well as shared mega-cells help expedite the process. Well-established methodologies exist in various microprocessor design companies. Block-level layout is usually done hierarchically. The steps for block-level layout involve partitioning, placement, routing, and compaction. They are detailed in Section 62.6.

CAD Perspective

The complexity of microprocessor design is growing, but there is no proportional growth in design team sizes. Historically, many tasks during the microprocessor layout were carefully hand-crafted. The reasons were twofold. The size of the problem was much smaller than what we face today. The second reason was that computer-aided design (CAD) was not mature. Many CAD vendors today are offering fast and accurate tools to automatically perform various tasks such as floorplanning, noise analysis, timing analysis, placement, and routing. This computerization has enabled large circuit design and fast turnaround times. References to various CAD tools with their capabilities have been added throughout this chapter.

CAD tools do not solve all of the problems during the microprocessor layout process. The regular blocks, like datapath, still need to be laid out manually with careful management of timing budgets. Designers cannot just throw the netlist over the wall to CAD to somehow generate a physical design. Manual effort and tools have to work interactively. Budgeting, constraints, connectivity, and interconnect parasitics should be shared across all levels and styles. Tools from different vendors are not easily interoperable due to a lack of standardization. The layout process may have proprietary methodology or technology parameters that are not available to the vendors. Many microprocessor manufacturers have their own internal CAD teams to integrate the outside tools into the flow or develop specific point tools internally. This chapter attempts to explain the advantages as well as shortcomings of CAD for physical layout.

Invaluable information about Physical Design Automation and related algorithms is provided in Refs. 11 and 12. These two textbooks cover a wide range of problems and solutions from the CAD perspective. They also include detailed analyses of various CAD algorithms. The reader is encouraged to refer to Refs. 13 to 15 for a deep understanding of digital design and layout.

Internet Resources

The Internet is bringing the world together with information exchange. Physical design of microprocessors is a widely discussed topic on the internet. The following web sites are a good resource for advanced learning of this field.

The key conference for physical design is the International Symposium on Physical Design (ISPD), held annually in April. The most prominent conference in Electronic Design Automation (EDA) community is the ACM/IEEE Design Automation Conference (DAC), (www.dac.com). The conference features an exhibit program consisting of the latest design tools from leading companies in design automation. Other related conferences are International Conference on Computer Aided Design (ICCAD) (www.iccad.com), IEEE International Symposium on Circuits and Systems (ISCAS) (www.iscas.nps.navy.mil), International Conference on Computer Design (ICCD), IEEE Midwest Symposium on Circuits and Systems (MSCAS), IEEE Great Lakes Symposium on VLSI (GLSVLS) (www.eecs.umich.edu/glsvlsi), European Design Automation Conference (EDAC), International Conference on VLSI Design (vcapp.csee.usf.edu/vlsi99/), and Microprocessor Forum. There are several journals which are dedicated to the field of VLSI Design Automation and include broad coverage of all topics in

physical design. They are *IEEE Transactions on CAD of Circuits and Systems* (akebono.stanford.edu/users/nanni/tcad), *Integration*, *IEEE Transactions on Circuits and Systems*, *IEEE Transactions on VLSI Systems*, and the *Journal of Circuits, Systems and Computers*. Many other journals occasionally publish articles of interest to physical design. These journals include *Algorithmica, Networks, SIAM Journal of Discrete and Applied Mathematics*, and *IEEE Transactions on Computers*.

An important role of the Internet is through the forum of newsgroups. comp.lsi.cad is a newsgroup dedicated to CAD issues, while specialized groups such as comp.lsi.testing and comp.cad.synthesis discuss testing and synthesis topics. The reader is encouraged to search the Internet for the latest topics. *EE Times* (www.eet.com) and *Integrated System Design* (www.isdmag.com) magazines provide latest information about physical design and they are both online publications. Finally, the latest challenges in physical design are maintained at (www.cs.virginia.edu/pd_top10/). The current benchmark problems for comparison of PD algorithms are available at www.cbl.ncsu.edu/www/.

We describe various problems involved throughout the microprocessor layout process in the Section 62.2.

62.2 Layout Problem Description

The design flow of a microprocessor is shown in Fig. 62.2. The architectural designers produce a high-level specification of the design, which is translated into a behavioral specification using function design, structural specification using logic design, and a netlist representation using circuit design. In this chapter, we discuss the microprocessor layout method called *physical design*. It converts a netlist into a mask layout consisting of physical polygons, which is later fabricated on silicon. The boxes on the right side of Fig. 62.2 depict the need for verification during all stages of the design. Due to high frequencies and shrinking die sizes, estimation of eventual physical data is required at all stages before physical design during the microprocessor design process. The estimation may not be absolutely necessary for other types of designs.

Let us consider the physical design process. Given a netlist specification of a circuit to be designed, a layout system generates the physical design either manually or automatically and verifies that the design conforms to the original specification. Figure 62.3 illustrates the microprocessor physical design flow.

Various specifications and constraints have to be handled during microprocessor layout. Global specs involve the target frequency, density, die size, power, etc. Process specs will be discussed in Section 62.3. The chip planner is the main component of this process. It partitions the chip into blocks, assigns blocks

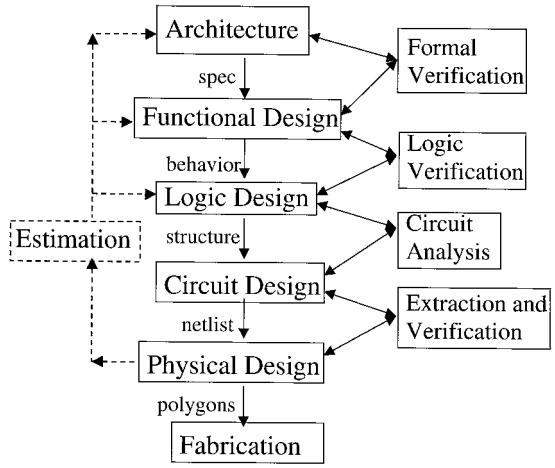


FIGURE 62.2 Microprocessor design flow.

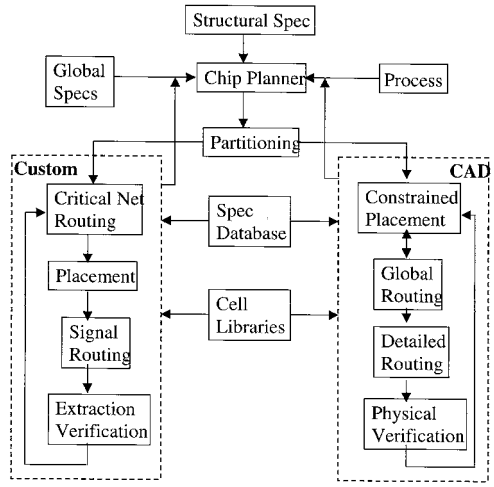


FIGURE 62.3 Microprocessor physical design flow.

for either full custom (manual) layout or CAD (automatic) layout and assembles the chip after block-level layout is finished. It may also iterate this process for better results. Full custom and CAD layout differ in the approach to handle critical nets. In the custom layout, critical nets are routed as a first step of block layout. In the CAD approach, the critical net requirements are translated into a set of constraints to be satisfied by placement and routing tools. The placement and global routing have to work in an iterative fashion to produce a dense layout. The double-sided arrow in CAD box represents this iteration. In both layout styles, iterations are required for block layout to completely satisfy all the specs. Some microprocessor teams employ a semi-custom approach which takes advantage of careful hand-crafting and power savings on the full custom side, and the efficiency and scalability of the CAD side.

Global Issues

The problems specific to individual stages of physical design are discussed in the following sections. This section attempts to explain the problems that affect the whole design process. Some of them may be applicable to the pre-layout design stages and post-layout verification.

Planning

There has to be a global flow to the layout process. The flow requires consistency across all levels and support for incremental re-design. A decision at one level affects almost all the other levels. The chip planning and assembly are the most crucial tasks in the microprocessor layout process. The chip is partitioned into blocks. Each block is allotted some area for layout. The allotment is based on estimation based on past experience. When the blocks are actually laid out, they may not fit in the allotted area. The full microprocessor layout process is long. One cannot wait until the last moment to assemble the blocks inside the chip. The planning and assembly team has to continuously update the flow, chip plans, and block interfaces to conform to the changing block data.

Estimation

New product generations rely on technology advances and providing the designer with a means of evaluating technology choices early in the product design.[16] Today's fine-line geometries jeopardize timing. Massive circuit density coupled with high clock rates, is making routed interconnects hardest to gauge early in the design process. A solid estimation tool or methodology is needed to handle today's complex microprocessor designs. Due to the uncertain effects of interconnect routing, the wall between logical and physical design is beginning to fall.[17] In the past, many microprocessor layout teams resorted

to post-layout updates to resolve interconnect problems. This may cause major re-design and another round of verification, and is therefore not acceptable. We cannot separate logical design and physical design engineers. Chip planners have to minimize the problems that interconnect effects may cause. Early estimation of placement, signal integrity, and power analysis information is required at the floorplanning stage even before the structural netlist is available.

Changing Specifications

Microprocessor design is a long process. It is driven by market conditions, which may change during the course of the design. So, architectural specs may be updated during the design. During physical design, the decisions taken during the early stages of the design may prove to be wrong. Some blocks may have added functionalities or new circuit families, which may need more area. The global abstract available to block-level designers may continuously change, depending on sibling blocks and global specs. Hence, the layout process has to be very flexible. Flexibility may be realized at the expense of performance, density, or area — but it is well worth it.

Die Shrinks and Compactions

The easiest way to achieve better performance is process shrinks. Optical shrinks are used to convert a die from one process to a finer process. Some more engineering is required to make the microprocessor work for the new process. A reduction in feature size from 0.50 μm to 0.35 μm results in an increase of approximately 60% more devices on a similarly sized die.[3] Layouts designed for a manufacturing process should be scalable to finer geometries. The decisions taken during layout should not prohibit further feature shrinks.

Scalability

CAD algorithms implemented in automatic layout tools must be applicable to large sizes. The same tools must be useful across generations of microprocessor. Training the designers on an entirely new set of CAD tools for every generation is impractical. The data representation inside the tools should be symbolic so that the process numbers can be updated without a major change in tools.

Explanation of Terms

There are many terms related to microprocessor layout used in the following sections. The definitions and explanation of those terms is provided in this section.

Capacitance: A time-varying voltage across two parallel metal segments exhibits capacitance. The voltage (v) and current (i) relation across a capacitor (C) is

$$i = C\frac{dv}{dt}$$

Closely spaced unconnected metal wires in layout can have significant cross-capacitance. Capacitance is very significant at 0.5-μm process and beyond.[18]

Inductance: A time-varying current in a wire loop exhibits inductance. If the current through a power grid or large signal buses changes rapidly, this can have inductive effects on adjacent metal wires. The voltage (v) and current (i) relation across an inductor (L) is

$$v = L\frac{di}{dt}$$

Inductance is not a local phenomenon like capacitance.

Parasitics: The shrinking technology and increasing frequencies are causing analog physical behavior in digital microprocessors.[19] The electrical parameters associated with final physical routes are called interconnect parasitics. The parasitic effects in the metal routes on the final silicon need to be estimated in the early phases of the design.

Design rules: The process specification is captured in an easy-to-use set of rules called design rules.

Spacing: If there is enough spacing between metal wires, they do not exhibit cross-capacitance. Minimum metal spacing is a part of the design rules.

Shielding: The power signal is routed on a wide metal line and does not have time-varying properties. In order to reduce external effects like cross-capacitance, on a critical metal wire, it is routed between or next to a power wire. This technique is called shielding.

Electromigration: Also known as metal migration, it results from a conductor carrying too much current. The result is a change in conductor dimensions, causing high resistive spots and eventual failure. Aluminum is the most commonly used metal in microprocessors. Its current density (current per width) threshold for electromigration is

$$2\frac{mA}{\mu m}$$

62.3 Manufacturing

Manufacturing involves taking the drawn physical layout and fabricating it on silicon. A detailed description of fabrication processes is beyond the scope of this book. Elaborate descriptions of the fabrication process can be found in Refs. 11 and 13. The reader may be curious as to why manufacturing has to be discussed before the layout process. The reality is that all of the stages in the layout flow need a clear specification of the manufacturing technology. So, the packaging specs and design rules must be ready before the physical design starts.

In this section, we present a brief overview of chip packaging and technology process. The reader is advised to understand the assessment of manufacturing decisions (see Ref. 16). There is a delicate balancing of the system requirements and the implementation technology. New product generation relies on technology advances and providing the designer with a means of evaluating technology choices early in the product design.

Packaging

ICs are packages into ceramic or plastic carriers usually in the form of a pin grid array (PGA) in which pins are organized in several concentric rectangular rows. These days, PGAs have been replaced by surface-mount assemblies such as ball grid arrays (BGAs) in which an array of solder balls connects the package to the board. There is definitely a performance loss due to the delays inside the package. In many microprocessors, naked dies are directly attached to the boards. There are two major methods of attaching naked dies. In wire bonding, I/O pads on the edge of the die are routed to the board. The active side of the die faces away from the board and the I/Os of the die lie on the periphery (peripheral I/Os). The other die attachment, control collapsed chip connection (C4) is a direct connection of die I/Os and the board. The I/O pins are distributed over the die and a solder ball is placed over each I/O pad (areal I/Os). The die is flipped and attached to the board. The technology is called C4 flip-chip. Figure 62.4 provides an abstract view of the two styles.

There is a discussion about practical issues related to packaging available in Ref. 20. According to the Semiconductor Industry Association's (SIA) roadmap, there should be 600 I/Os per package in 2507 rows, 7 μm package lines/space, 37.5 μm via size, and 37.5 μm landing pad size by the year 1999. The SIA roadmap lists the following parameters that affect routing density for the design of packaging parameters:

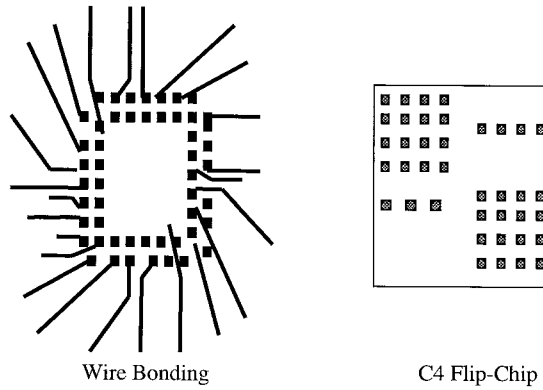


FIGURE 62.4 Die attachment styles.

- **Number of I/Os**: This is a function of die size and planned die shrinks. The off-chip connectivity requires more pins.
- **Number of rows**: The number of rows of terminals inside the package.
- **Array shape**: Pitch of the array, style of the array (i.e., full array, open at center, only peripheral).
- **Power delivery**: If the power and ground pins are located in the middle, the distribution can be made with fewer routing resources and more open area is available for signals. But then, the power cannot be used for shielding the critical signals.
- **Cost of package**: This includes the material, processing cost, and yield considerations. The current trend in packaging indicates a package with 1500 I/O on the horizon and there are plans for 2000 I/Os.

There is a gradual trend toward the increased use of areal I/Os. In the peripheral method, the I/Os on the perimeter are fanned out until the routing metal pitch is large enough for the chip package and board to handle it. There may be high inductance in the wire bonding. Inductance causes current time delay at switching, slow rise time, and ground bounce in which the ground plane moves away from 0 V, noise, and timing problems. These effects have to be handled during a careful layout of various critical signals. Silicon array attachments and plastic array packages are required for high I/O densities and power distribution. In microprocessors, the packaging technology has to be improvised because of the growth in bus widths, additional metal layers, less current capacity per wire, more power to be distributed over the die, and the growing number of data and control lines due to bus widths. The number of I/Os has exceeded the wire bonding capacity. Additionally, there is a limit to how much a die can be shrunk in the wire bonding method. High operating frequencies, low supply voltage, and high current requirements manifest themselves into a difficult power distribution across the whole die. There are assembly issues with fine pitches for wire bonds. Hence, the microprocessor manufacturers are employing C4 flip-chip technologies. Areal packages reduce the routing inside the die but need more routing on the board.

The effect of area packaging is evident in today's CAD tools.[21] The floorplanner has to plan for areal pads and placement of I/O buffers. Area interconnect facilitates high I/O counts, shorter interconnect rates, smaller power rails, and better thermal conductivity. There is a need for an automatic area pad planner to optimize thousands of tightly spaced pads. A separate area pad router is also desired. The possible locations for I/O buffers should be communicated top-down to the placement tool and the placement info should be fed back to the I/O pad router. After the block level layout is complete and the chip is assembled, the area pad router should connect the power pads to inner block-level power rails.

Let us discuss some industry microprocessor packaging specs. The packaging of DEC/Compaq's Alpha 21264 has 587 pins.[4] This microprocessor contains distributed on-chip decoupling capacitors (decap) as well as a 1-μm package decap. There are 144-bit (128-bit data, 16-bit ECC) secondary cache data interface and 72-bit system data interface. Cache and system data pins are interleaved for efficient multiplexing. The vias have to arrayed orthogonal to the current flow. HP's PA-8000 has a flip-chip package, which

enables low resistance, less inductance, and larger off-chip cache support. There are 704 I/O signals and 1200 power and ground bumps in the 1085-pin package. Each package pin fans out to multiple bumps.[6] PowerPC™ has a 255-pin CBGA with C4 technology.[7] 431 C4's are distributed around the periphery. There are 104 VDD and GND internal C4's. The C4 placement is done for optimal L2 cache interface.

There is a debate about moving from high-cost ceramic to low-cost plastic packaging. Ceramic ball grid arrays suffer from 50% propagation speed degradation due to high dielectric constant (10). There is a trend to move toward plastic. However, ceramic is advantageous in thermal conductivity and it supports high I/O flip-chip packaging.

Technology Process

The whole microprocessor layout is driven by the underlying technology process. The process engineers decide the materials for dielectric, doping, isolation, metal, via, etc. and design the physical properties of various lithographic layers. There has to be close cooperation between layout designers and process engineers. Early process information and timely updates of technology parameters are provided to the design teams, and a feedback about the effect of parameters on layout is provided to the process teams. Major process features are managed throughout the design process. This way, a design can be better optimized for process, and future scaling issues can be uncovered.

The main process features that affect a layout engineer are metal width, pitch and spacing specs, via specs, and I/O locations. Figure 62.5(a) shows a sample multi-layer routing inside a chip. Whenever two metal rails on adjacent layers have to be connected, a via needs to be dropped between them. Figure 62.5(b) illustrates how a via is placed. The via specs include the type of a via (stacked, staggered), coverage of via (landed, unlanded, point, bar, arrayed), bottom layer enclosure, top layer enclosure, and the via width. In today's microprocessors, there is a need for metal planarization. Some manufacturers are actually adding planarization metal layers between the usual metal layers for fabrication as well as shielding. Aluminum was the most common metal for fabrication. IBM has been successful in getting copper to work instead of aluminum. The results show a 30% decrease in interconnect delay.

The process designers perform what-if analyses and design sensitivity studies of all of the process parameters on the basis of early description of the chip with major datapath and bus modeling, net constraints, topology, routing, and coupled noise inside the package. The circuit speed is inversely proportional to the physical scale factor. Aggressive process scaling makes manufacturing difficult. On the other hand, slack in the parameters may cause the die size to increase. We have listed some of the process numbers in today's leading microprocessors in this section. The feature sizes are getting very small and many unknown physical effects have started showing up.[22] The processes are so complicated to correctly obey during the design, an abstraction called design rules is generated for the layout engineers. Design rules are constraints imposed on the geometry or topology of layouts and are derived from basic physics of circuit operation such as electromigration, current carrying capacity, junction breakdown, or

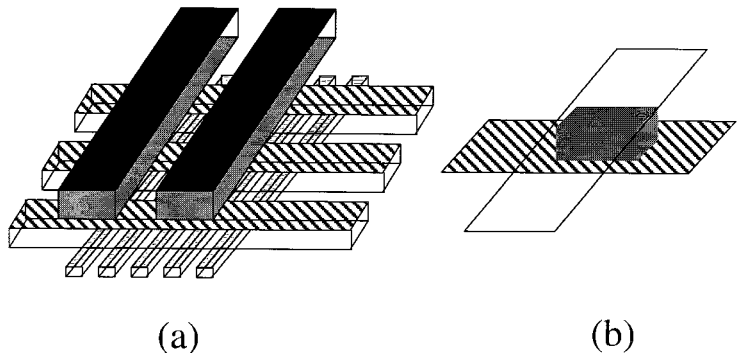


FIGURE 62.5 A view of multi-layer routing and a simple via.

punch-through, and limits on fabrication such as minimum widths, spacing requirements, misalignments during processing, and planarization. The rules reflect a compromise between fully exploiting the fabrication process and producing a robust design on target.[5]

As feature sizes are decreasing, optical lithography will need to be replaced with deep-UV, X-ray, or electron beam techniques for features sizes below 0.15 μm.[20] It was feared that quantum effects will start showing up below 0.1 μm. However, IBM has successfully fabricated a 0.08-μm chip in the laboratory without seeing quantum effects. Another physical limit may be the thickness of the gate oxide. The thickness has dropped to a few atoms. It is soon going to hit a fundamental quantum limit.

Alpha 21264 has 0.35-μm feature size, 0.25-μm effective channel length, and 6-nm gate oxide. It has four metal layers with two reference planes. All metal layers are AlCu. Their width/pitches are 0.62/1.225, 0.62/1.225, 1.53/2.8, and 1.53/2.8 μm, respectively.[4] Two thick aluminum planes are added to the process in order to avoid cycle-to-cycle current variations There is a ground reference plane between metal2 and metal3, and a VDD reference plane above metal4. Nearly the entire die is available for power distribution due to the reference planes. The planes also avoid inductive and capacitive coupling.[8]

PowerPC™ has 0.3-μm feature size, 0.18-μm effective channel length, 5-nm gate oxide thickness, and a five-layer process with tungsten local interconnect and tungsten vias.[7] The metal widths/pitches are 0.56/0.98, 0.63/1.26, 0.63/1.26, 0.63/1.26, and 1.89/3.78 μm, respectively.

HP-8000 has 0.5-μm feature size and 0.29-μm effective channel length.[6] There is a heavy investment in the process design for future scaling of interconnect and devices. There are five metal layers, the bottom two for local fine routing, metal3 and metal4 for global low resistive routing, and metal5 reserved for power and clock. The author could not find published detailed metal specs for this microprocessor.

Intel Pentium II is fabricated with a 0.25-μm CMOS four-layer process.[23] The metal width/pitches are 0.40/1.44, .64/1.36, .64/1.44, and 1.04/2.28 μm, respectively. The two lower metal layers are usually used in block-level layout, metal3 is primarily used for global routing, and metal4 is used for top-level chip power routing.

62.4 Chip Planning

As explained in Section 62.2, chip planning is the master step during the layout of a microprocessor. During the early stages of design, the planning team has to assign area, routing, and timing budgets to individual blocks on the basis of some estimation methods. Top-down constraints are imposed on the individual blocks. During the block layout, continuous bottom-up feedback to the planner is necessary in order to validate or update the imposed constraints and budgets. Once all the blocks have been laid out and their accurate physical information is available, the chip planning team has to assemble the full chip layout subject to the architectural and process specs.

Chip planning involves partitioning the microprocessor into blocks. The finite state machines are considered random control logic and partitioned into automatically synthesizable blocks. Regular structures like arrays, memories, and datapath require careful signal routing and pitch matching. They have to be partitioned into modular and regular blocks that can be laid out using full-custom or semi-custom techniques.

IBM adopted a two-level hierarchical approach for the G4 processor.[24] They identified groups of 10,000 to 20,000 non-array transistors as macros. Macros were individually laid out by parallel teams. The macro layouts were simplified and abstracted for floorplanning, place and route, and global extraction. The shapes of individual blocks varied during the design process. The chip planner performed the layouts for global interconnects and physical design of the entire chip. The global environment was abstracted down to the block level. A representation of global wires was added overlaying a block. That included global timing at block interfaces, arrival times with phase tags at primary inputs (PI), required times with phase tags at primary outputs (PO), PI resistances, and PO capacitances. Capacitive loading at the outputs was based on preliminary floorplan analysis. Each block was allowed sufficient wiring and cell area. The control logic was synthesized with high-performance standard cell library; datapaths were designed with semi-custom macros. Caches, Memrory Management Unit (MMU) arrays, branch unit

arrays, Phase-Locked Loop (PLL), and Delay-Locked Loop (DLL) were all full-custom layouts.[7] There were three distinct physical design styles optimizing for different goals, namely, full custom for high performance and density, structured custom for datapath, and fully automated for control logic. The floorplan was flexible throughout the methodology. There are 44% memory arrays, 21% datapath, 15% control, 11% I/O, 9% miscellaneous blocks on the die. Final layout was completely hierarchical with no limits on the levels of hierarchy involved inside a block. The block layouts had to conform to a top abstracted global shadow of interconnects and blockages. The layout engineers performed post-placement re-tuning and post-placement optimization for clock and scan chains.

For the 1-GHz integer PowerPC™ microprocessor, the planning team at IBM enforced strict partitioning on latch boundaries for global timing closure.[5] The planning team constructed a layout description view of the mega-cells containing physical shape data of the pads, power buses, clock spine, and global interconnects. At the block level, pin locations, capacitance, and blockages were available. The layouts were created by hand due to the very high-performance requirements of the chip.

We describe the major steps during the planing stages, namely, floorplanning, power planning, clock planning, and bus routing. These steps are absolutely essential during microprocessor design. Due to the complicated constraints, continuous intelligent updates, and top-down/bottom-up communication, manual intervention is required.

Floorplanning

Floorplannig is the task of placing different blocks in the chip so as to fit them in the minimum possible area with minimum empty space. It must fill the chip as close to the brim as possible. Figure 62.6 shows an example of floorplanning. The blocks on the left hand side are fitted inside the chip on the right. The reader can see that there is very little empty space on the chip. The blocks may be flexible and their orientation not fixed. Due to the dominance of interconnect in the overall delay on the chip, today's floorplanning techniques also try to minimize global connectivity and critical net lengths.

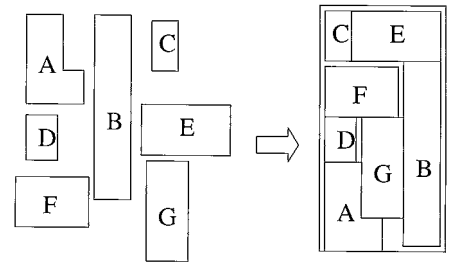


FIGURE 62.6 An example of floorplanning.

There are many CAD tools available for floorplanning from the EDA vendors. The survey of all such tools is available.[25] The tools are attempting to bridge the gap between synthesis and layout. All of the automatic tools are independent of IC design style. There are two types of floorplanners. Functional floorplanners operate at the RTL level for timing management and constraints generation. The goal of physical floorplanners is to minimize die size, maximize routability, and optimize pin locations. Some physical floorplanners perform placement inside floorplanning. As explained in the routing section, when channel routing is used, the die size is unpredictable. The floorplanners cannot estimate routing accurately. Hence, channel allocation on the die is very difficult. Table 62.2 summarizes the CAD tools available for floorplannning.

TABLE 62.2 CAD Tools Available for Floorplanning

Company	Internet	Product	Description
Avant!	www.avanticorp.com	Planet	Timing-driven hierarchical floorplanner
Cadence	www.cadence.com	Preview	Mixed-level floorplanning and analysis environment
Compass	www.compass-da.com	ChipPlanner-RTL	Timing constraint satisfaction before logic synthesis
HLD	www.hlds.com	Physical DP	Constraint-driven floorplanning
HLD	www.hlds.com	Top-down DP	RTL-level timing analysis for pre-synthesis; internal estimation tool
SVR	www.svri.com	FloorPlacer	Timing and routability analysis with floorplanning

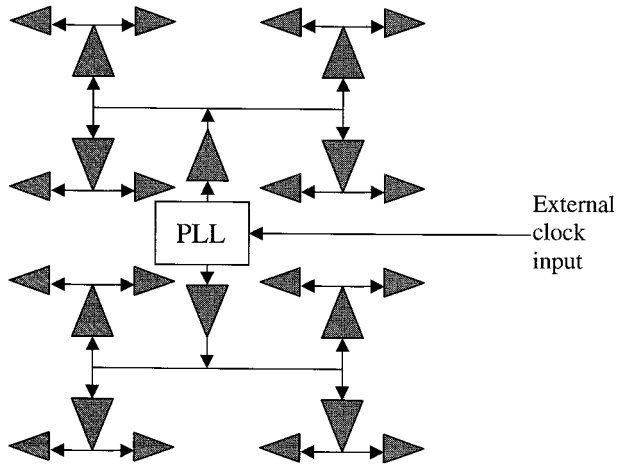


FIGURE 62.7 A sample global clock buffered H-tree.

Clock Planning

Clock is a global signal and clock lines have to be very long. Many elements in high-frequency microprocessors are continuously being clocked. Different blocks on the same die may operate at different frequencies. Multiple clocks are generated internally and there is a need for global synchronization. Clock methodology has to be carefully planned and the individual clocks have to be generated and routed from the chip's main phase-locked loop (PLL) to the individual sink elements. The delays and skews (defined later) have to exactly match at every sink point. There are two major types of clock networks, namely, trees and grids. Figure 62.7 illustrates a modified H-tree with clock buffers. Figure 62.8 shows a clock grid used in Alpha processors. Most of the power consumption inside today's high-frequency processors is in their clock networks. In order to reduce the chip power, there are architectural modifications to shut off some part of the chip. This is achieved by clock gating. The clock gator routing has become an integral part of clock routing.

Let us explain the some terms used in clock design. Clock skew is the temporal variation of the same clock edge arriving at various locations on the die. Clock jitter is the temporal variation of consecutive clock edges arriving at the same location. Clock delay is the delay from the source PLL to the sink element. Both skew and jitter have a direct relation to clock delay. Globally synchronous behavior dictates minimum skew, minimum jitter, and equal delay.

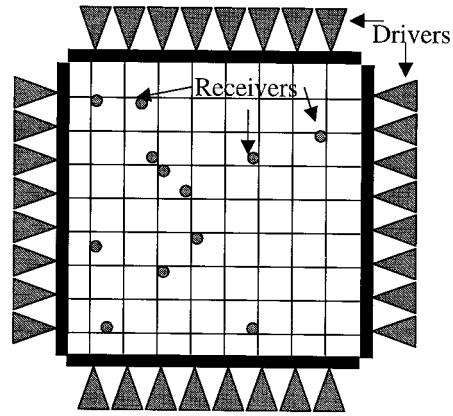


FIGURE 62.8 A sample clock grid.

Clock grids, being perfectly symmetric, achieve very low skews, but they need high routing resources and stacked vias, and cause signal reflections. The wire loading on driving buffers feeding to the grid is also high. This requires large buffer arrays that occupy significant device area. Electrical analysis of grids is more difficult than trees. Buffered trees are preferred in high-performance microprocessors because they achieve acceptable skews and delays with low routing resource usage.

Ideally, the skew should be 0. However, there are many unknowns due to processing and randomness in manufacturing. Instead of matching the clock receivers exactly, a skew budget is assigned. In high-performance microprocessor designs, there is usually a global clock routing scheme (GCLK) that spawns into multiple matched clock points in various regions on the chip. Inside the region, careful clock routing is performed to match the clock delay within assigned skew budgets.

Alpha 21264 has a modified H-tree. On-chip PLL dissipates power continuously, 40% of the chip power dissipation was measured to be in the clocking network. Reduction of clock power was a primary concern to reduce overall chip power.[26] There is a GCLK network that distributes clock to local clock buffers. GCLK is shielded with VCC or VSS throughout the die.[4] GCLK skew is 70 ps, with 50% duty cycle and uniform edge rate.[8] The clock routing is done on metal3 and metal4. In earlier Alpha designs, a clock grid was used for effective skew minimization. The grid consumed most of the metal3 and metal4 routing resources. In 21264, there is a savings of 10 W power over previous grid techniques. Also, significantly less metal3 and metal4 is used for clock routing. This proved that a less aggressive skew target can be achieved with a sparser grid and smaller drivers. The new technique also helped power and ground networks by spreading out the large clock drivers across the die.

HP-8000 also has a modified H-tree for clock routing.[6,18] External clock is delivered to the chip PLL through a C4 bump. The microprocessor has a three-level clock network. There is a modified H-tree that routes GCLK from PLL to 12 secondary buffers strategically placed at various critical locations in various regions on the chip. The output of the receiver is routed to matched wire lengths to a second level of clock buffers. The third level involves 7000 clock gators that gate the clock routing from the buffers to local clock receivers. There are many flavors of gated clocks on the chip. There is a 170-ps skew across the die. Due to a large die, PA8000 buffers were designed to minimize process variations.

In PowerPC™, a PLL is used for internal GCLK and a DLL is used for external SRAM L2 interface.[7] There is a semi-balanced H-tree network from PLL to local regenerators. Semi-balanced means the design was adjusted for variable skew up to 55 ps from main PLL to H-tree sinks. There are three variations of masking 486 local clock regenerators. The overall skew across the die was 300 ps.

Many CAD vendors have attempted to provide clock routing technologies. The microprocessor community is very paranoid about clock and clocking power. The designers prefer hand-crafting the whole clock network.

Power Planning

Every gate on the die needs the power and ground signals. Power arrives at many chip-level input pins or C4 bumps and is directly connected to the topmost metal layer. Routing power and ground from the topmost layer to each and every gate on the die without consuming too many routing resources, not causing voltage drops in the power network, and using effective shielding techniques constitutes the power planning problem. A high-performance power distribution scheme must allow for all circuits on the die to receive a constant power reference. Variation in the reference will cause noise problems, subthreshold conduction, latch-up, and variable voltage swings

The switching speed of CMOS circuits in the first order is inversely proportional to the drain-to-source current of the transistor (I_{ds}),in the linear region:

$$t = C\int \frac{dV}{I_{ds}}$$

where, C is the loading capacitance, V is the output voltage, and t is the switching delay. I_{ds}, in turn, depends on the IR drop (V_{drop}) as:

$$I_{ds} \propto \left(V_{gs} - V_t - V_{drop}\right)$$

where V_{gs} is the gate to source voltage and V_t is the threshold voltage of the MOS transistor. Therefore, achieving the highest switching speed requires distributing the power network from the pads at the periphery of the die or C4 bumps to the sources of the transistors with minimal IR drop due to routing. The problem of reducing V_{drop} is modeled in terms of minimum allowable voltage at the source and the difference between Vdd and Vss acceptable at the sinks. All physical stages from pads to pins have to be considered. Some losses, like tolerance of the power supply, the tester guardband, and power drop in the package, are out of the designer's control. The remaining IR drop budget is divided among global and local power meshes.

The designers at Motorola have provided a nice overview of power routing in Ref. 27. Their design of PowerPC™ power grid continued across all design stages. A robust grid design was required to handle the possible switching and large current flow into the power and ground networks. Voltage drops in power grid cause noise, degrading performance, high average current densities, and undesirable wearing of metal. The problem was to design a grid achieving perfect voltage regulation at all demand points on the chip, irrespective of switching activities and using minimum metal layers. The PowerPC™ processor family has a hierarchy of five or six metal layers for power distribution. Structure, size, and layout of the power grid had to be done early in the design phase in the presence of many unknowns and insufficient data. The variability continued until the end of design cycle. All commercial tools depend on post-layout power grid analysis after the physical data is available. One cannot change the power plan at that stage because too much is at stake toward the end. Hence, Motorola designers used power analysis tools at every stage. They generated applicable constant models for every stage. There are millions of demand points in a typical microprocessor. One cannot simulate all non-linear devices with a non-ideal power grid. Therefore, the approach was as follows. They simulated non-linear devices with fixed power, converted all devices to current sources, and then analyzed the power grid. There was still a large linear system to handle. So, a hierarchical approach was used. Before the floorplaning stage, the locations of clean VCC/GND pads and power grid widths/pitches were decided on the basis of design rules and via styles (point or bar vias). After the floorplan was fixed, all blocks were given block power service terminals. Wires that connect global power to block power were also modeled in the service terminals. Power was routed inside the blocks and PowerMill simulations were used for validation.

Alpha 21264 operates at a high frequency and has a large die as listed in Table 62.1. The large die and high frequency lead to high power supply currents. This has a serious effect on power, clock, and ground networks.[3,4] Power dissipation was the sole factor limiting chip complexity and size; 198 out of 587 chip-level pins are VDD and VSS pins. Supply current has doubled during every generation of Alpha microprocessor. Hence, a very complex power distribution was required. In order to meet very large cycle-to-cycle current variations, two thick low-resistance aluminum planes were added to the process.[8] One plane was placed between metal2 and metal3 connected to VSS, and the other above the topmost metal4 connected to VDD. Nearly the entire die area was available for power distribution. This helped in inductive and capacitive decoupling, reduced on-chip crosstalk, and presented excellent current returns paths for analysis and minimized inductive noise.

UltraSparce-I™ has 288 power and ground pins out of 520.[9] The methodology involved an early identification of excessive voltage drop points and seamless integration of power distribution and CAD tools. Correct-by-construction power grid design was done throughout the design cycle. The power networks were designed for cell libraries and functional blocks. They were reliability-driven designs before mask generation. This enabled efficient distribution of the Vdd and Vss networks on a large die. Minimization of area overhead, as well as IR drop for power distribution, was considered throughout the design cycle. Parts of power distribution network are incorporated into the standard cell library layouts.

CAD tools were used for the composition of standard cell and datapath with correct-by-construction power interconnections. The methodology was designed to be scalable to future generations. Estimation and budgeting of IR drops was done across the chip. Metal4 was the only over-the-block routing layer. It was used for routing power from peripheral I/O pads to individual functional units. It was the primary means of distributing power. The power distribution should not constrain the floorplan. Hence, two meshes were laid out: a top-down global mesh and an in-cell local mesh. This enabled block movement during placement because they have only local mesh. As long as the local power mesh crosses the global mesh, the power can be distributed inside the block. Metal3 local power routes have to be orthogonal to global metal4 power. The direction of metal1 and metal2 do not matter. The global chip is divided into two parts. In part 1, metal3 was vertical and metal4 was horizontal. The opposite directions were selected for the second part. A block could be moved half the die distance because of two types of regions for power on the chip. The power grid on three metal layers with interconnections, number of vias, and via types was simulated using HSPICE to determine the widths, spacings, and number of vias of the power grid. Vias had to be arrayed orthogonal to the current flow. There was a 90-mV IR drop from M3-M4 via to the source of a cell. Additional problems existed because the metal2 width is fixed in UltraSparc™. Up to a certain drive strength, the metal2 power rail was 2.5 μm. Beyond that, additional rail of 1 μm was added. The locations of clock receivers changed throughout the design process. They had to be shifted to align power.

Bus Routing

The author considers bus routing a critical problem and it needs the same attention as power or clock routing. The problem arises due to today's superscalar, large bit-width microprocessor architectures. The chip planners design the clock and power plans and floorplan the chip very efficiently to minimize empty space on the die, but leave limited routing resources on the top layers to route busses. There is a simple analogy to understand this problem. Whenever a city is being planned, the roads are constructed before the individual buildings. In microprocessor layout, busses must be planned before the blocks are laid out.

A bus, by nature, is bi-directional and must have matching characteristics at all data bits. There should be a matching RC delay viewed from both ends. It connects a wide datapath to another. If it is routed straight from one datapath block to another, then the characteristics match; but it is not always feasible on the die to achieve straight routes. Whenever there is a directional change, via delay comes into picture. The delays due to via and uneven lengths for all the bit-lines in the bus cause a mismatch across the bits of the bus. Figure 62.9 depicts a simple technique called bus interleaving, employed in today's microprocessors, to achieve matching lengths.

The problems do not end there. Bus interleaving may match the lengths across the bit-widths, but it does not guarantee matching environment for all the bit-lines. Crosstalk due to adjacent layers or busses may cause mismatch among the bit-lines. In differential circuits, very low voltage busses are routed with long routing lengths. Alpha designers had to carefully route low swing busses in 21264 to minimize all differential noise effects.[3] These types of busses need shielding to protect the low-voltage signals. If all bits in a bus switch simultaneously, large current variations inject inductive noise into the neighboring signal lines. Hence, other signals also need to be shielded from active busses.

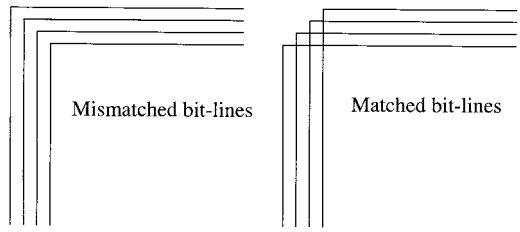


FIGURE 62.9 Bus interleaving.

Cell Libraries

A major step toward high performance is the availability of a fast ready-to-use circuit library. Due to large and complex circuit sizes, transistor-level layout is formidable. All microprocessor teams design a family of logic gates to perform certain logic operations. These gates become the bottom level units in the netlist hierarchy. They serve as a level of abstraction higher than a basic transistor. Predefined logic functions help in automatic synthesis. The gates may differ in their circuit family, logic functions, drive strength, power consumption, internal layout, placement of cell interface ports, power rails, etc. The number of different cells available in the design libraries can be as high as 2000. The libraries offer the most common predefined building blocks of logic and low-level analog and I/O functions. Complex designs require multiple libraries. The libraries enable fast time to market, aid synthesis in logic minimization, and provide an efficient representation of logic in hardware description languages.

Block-level layout tools support cell-based layout. They need the cells to be of a certain height and perform fast row-based layout. The block-level layout tools are very mature and fast. Many microprocessor design teams design their libraries to be directly usable by block-level layout tools. There are many CAD tools available for cell designs and cell-based block designs. The most common approach is to develop a different library for each process and migrate the design to match the library. Process-specific libraries lead to small die size with high performance. There are tools available on the market for automatic process porting, but the portability across processes causes performance and area degradation.

Microprocessor manufacturers have their in-house libraries designed and optimized for proprietary processes. The cell libraries have to be designed concurrently with the process design and they must be ready before the block-level design begins. The libraries for datapath and control can differ in styles, size, and routing resource utilization. As datapath is considered crucial to a microprocessor, datapath libraries may not support porosity, but the control logic library has to provide porosity for neighboring datapath cells to use some of its routing resources. Thus, datapath libraries are designed for higher performance than control. In UltraSparc-I™ processor, the design team at Sun Microsystems used separate standard cells for datapath and control.[9]

In this section, we present various layout aspects of cell library design. The reader is requested to refer to Refs. 13-15 for circuit aspects of libraries.

Circuit Family

The most common circuit family is CMOS. They are very popular because of the static nature. It is a fully restored logic in which output either sets at Vdd or Vss. The rise and fall times are of the same order. This family has almost zero static power dissipation. The main advantage in layout is its symmetric nature, nice separation of n and p transistors, and ability to produce regular layouts. Figure 62.10 shows a three-input CMOS NOR library cell.

The other popular circuit family in high-performance microprocessors is that of dynamic circuits. The inputs feed into the n-stack and not the p-stack. There is a precharge p-transistor and a smaller keeper p-transistor in the p-stack. So, the number of transistors in p-stack is exactly 2. The dynamic circuits need careful analysis and verification, but allow wide OR structures, less fan-in and fan-out capacitance. The switching point is determined by the nMos threshold and there is no crossover current during output transition. As there is less loading on the inputs, this circuit family is very fast. As one can see in Fig. 62.7, the area occupied by the p-stack is very large compared to the n-stack in static CMOS. Domino logic families have a significant area advantage over static if the same static netlist can be synthesized in monotonic domino gates. However, layout of domino gates is not trivial. Every gate needs a clock routed to it. As the family does not support fully restoring logic, the domino gate output needs to be shielded from external noise sources. Additional circuitry may be required to avoid charge-sharing and noise problems.

Other circuit families include BiCMOS, in which bipolar transistors are used for high speed and CMOS transistors are used for low power, high-density gates; differential cascode voltage switch logic (DVSL),

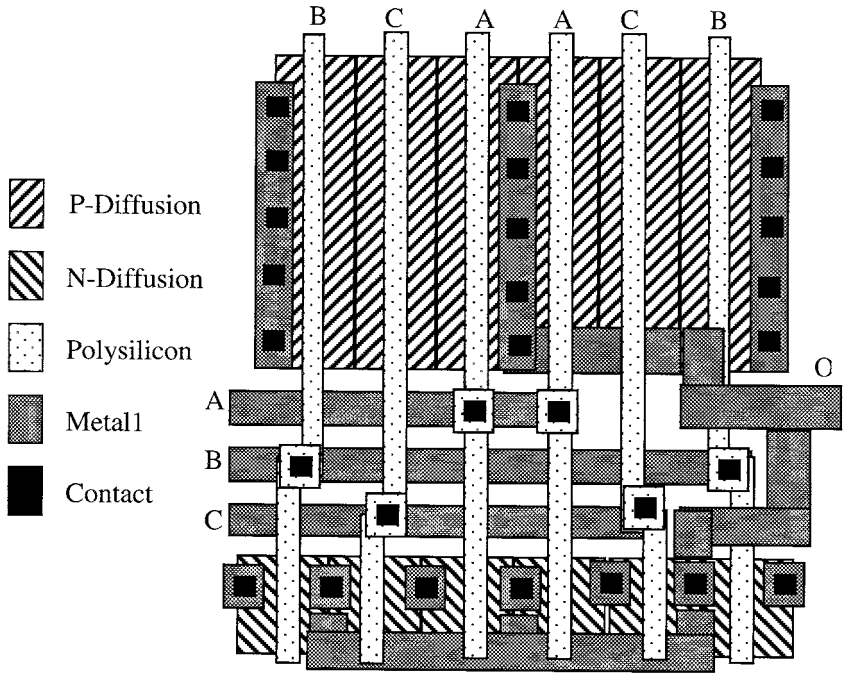


FIGURE 62.10 A three-input CMOS NOR layout.

in which differential output logic uses positive feedback for speed-up; differential split-level logic (DSL), in which load is used to reduce output voltage swing; and pass transistor logic (PTL), in which complex logic such as muxing is easily supported.

Cell Layout Architecture

There are various issues involved in deciding how a cell should be laid out. Let us look at some of the issues.

Cell height: If row-based block layout tools are going to be used, then the cells should be designed to have standard heights. This approach also helps in placement during full-custom layout. Basically, constraining one dimension (height) enables better optimization for the other one (width). However, snapping to a particular height may cause unnecessary waste of active transistor area for cells with small drive strengths.

Diffusion orientation: Manufacturing may cause some variation in cell geometries. In order to achieve consistent variations across all transistors inside a cell, process technology may dictate fixed orientation of transistors.

Metal usage: Cells are part of a larger block. They should allow block-level over-the-cell routing. Guidelines for strict metal usage must be followed while laying out cells. Some cell guidelines may force single-metal usage inside the cell.

Power: Cells must adhere to the block-level power grid. They should either instantiate power pins internally and include the power pins in the interface view, or should enable block-level power routing by abutment. In UltraSparc-I™, there was a clear separation of metal usage between datapath and control standard cells. The power in control was distributed on horizontal metal1 with adjacent cells abutting the rails. Metal2 was only used to connect metal1 to metal3 power. Metal2 power hook-up could have been longer for better power delivery, but it would consume routing resources. The datapath library had vertical metal2 abutting for power and it was directly connected to metal3 power grid.[9]

Cell abstraction: Internal layout details of a cell are not required at the block level. Cells should be abstracted to provide a simplified view of interface pins (ports), power pins, and metal obstructions. Design guidelines may have requirements for coherent cell abstract views. Multiple cell families may differ in their internal layout, but there may be a need for generating consistent abstract views for easy placement and routing.

Port placement: If channel routers are used, then interface ports must lie at the cell boundaries. For area routers, the ports can be either at the boundary or at internal locations where there is enough space to drop a via from a higher metal layer passing over the cell.

Gridding: All geometries inside the cell must lie on the manufacturing grid. Some automatic tools may enforce gridding for cell abstracts. In that case, the interface ports must be on a layout routing grid dictated by the tools.

Special requirements: These can include family-specific constraints. A domino cell may need specific clock placement; a different logic cell may need strict layout matching for differential signals, etc.

Stretchability: Consider two versions of the CMOS NOR3 gate as shown in Fig. 62.11. As we can see, the widths of the transistors changed, but the overall layout looks very similar. This is the idea behind stretchability and soft libraries. Generate new cells from a basic cell, depending on the drive strength required. In the G4 processor, IBM design team used a continuously tunable, parameterized standard cell library with logic functions chosen for performance.[24] The cells were available in discrete levels or sizes. The rules were continuously tunable. Parameterization was done for delay, not size. They also had a parameterized domino library. Beta and gain tuning enabled delay optimization during placement, even after initial placement. Changes due to actual routing were handled as engineering change orders (ECOs). The cell layouts were generated from soft libraries. The automatic generator concentrated on simple static cells. The most complex cell was a 2×2 AO/OA. The soft library also allowed customization of cell images. The cell generator generated a standard set of sizes, which were selected and used over the entire chip. This approach loses the cell library notion. So, the layout was completely flattened. Some cells were also non-parameterized. Schematics were generated on the basis of tuned library and flattened layout. This basically led to a block-level mega-cell just like a standard cell.

Characterization: As we mentioned before, circuit aspects of cell design are out of the scope of this section. However, we briefly explain characterization of the cell because it impacts layout. The detailed electrical parasitics of cell layout are extracted and the behavior of each library cell is

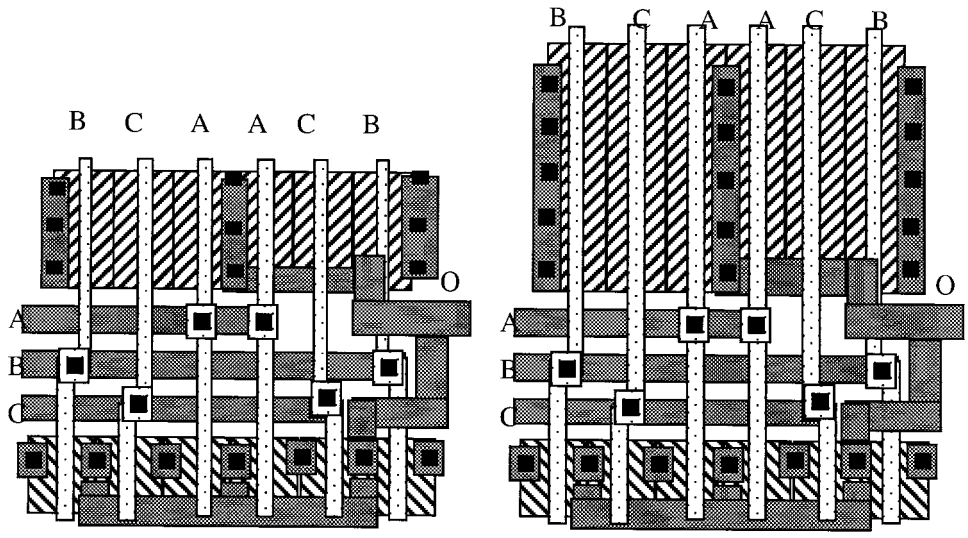


FIGURE 62.11 Cell stretching.

individually characterized over a range of output loads and input rise/fall times. The parameters tracked during this process are propagation delay, output rise/fall times, and peak/average current. The characterization can be represented as a closed-form equation of input rise/fall times, output loading, and device characteristics inside the cell. Another popular method involves generating look-up table models for the equations. The tables need interpolation methods. Using the process data and electromigration limits, the width of signal/supply rails and minimum number of contacts were determined in UltraSparc-I™. These values are formulated as a set of layout verification rules for post-layout checks.[9] In the PowerPC microprocessor, all custom circuits and library elements were simulated over various process corners and operating conditions to guarantee reliable operation, sufficient design margin, and sufficient scalability.[7]

Mega-cells: Today's superscalar microprocessors have regular and modular architectures. Not only standard cells, but large layout blocks such as clock drivers, ROMs, and ALUs can also be repeated at several locations on the die. Mega-cells is a concept that generalizes standard cells to a larger size. This automatically converts logic function to a datapath function. Automatic layout is not recommended for mega-cells because of the internal irregularity. Layout optimization of a mega-cell is done by full-custom technique, which is time-consuming; but if it is used multiple times on the die, the effort pays off.

Cell Synthesis

As mentioned earlier in this section, there are CAD vendors supporting library generation tools. Cadabra (www.cadabratech.com) is a leading vendor in this area with its CLASSIC tool suite. Another notable vendor tool is Tempest-Cell from Sycon Design Inc. (www.sycon-design.com). A very good overview of such tools and external library vendors is available in Ref. 28. The idea of external libraries originated from IC databooks. In the past, ready-to-use ICs were available from various vendors with fully detailed electrical characteristics. Now, the same concept is applied to cell libraries, which are not ICs, but ready-to-use layouts that can be included in bigger circuits. The libraries are designed specific to a particular process and gate family, but they can be ported to other architectures. Automatic process migration tools are available on the market. Complex combinational and sequential functions are available in the libraries with varying electrical characteristics comprising of strengths, fan-out, load matching, timing, power, area attributes, and different views. The library vendors also provide synthesis tools that work with logic design teams and enable usage of new cells.

Block-Level Layout

A block is a physically and logically separated circuit inside a microprocessor that performs a specific arithmetic, logic, storage, or control function. Roughly speaking, a full-custom technique is used for layout of regular structures, like arrays and datapath; whereas, automatic tools are used for random control logic consisting of finite state machines. Block-level layout is a very thoroughly researched and mature area. The author has biased the presentation in this section toward automation and CAD tools. Full-custom techniques accept more constraints, but approximately follow the same methodology.

Block-level layout needs careful tracking of all pieces.[29] Due to its hierarchical nature, strict signal and net naming conventions must be followed. The blocks' interface view may be a little fuzzy. Where does a block design end? At the output pin of the current block or at the input pin of the block it is feeding to? There may be some logic that cannot be classified into any of the types and it is not large enough to be considered a separate block of its own. Such logic is called glue logic. Glue logic at the chip level may actually be tightly coupled to lower-level gates. It needs physical proximity to the lower level. Every block may be required to include some part of such glue logic during layout.

In IBM's G4 microprocessor, custom layout was used for dataflow stacks and arrays. A semi-custom cell-based technique was used for control logic.[24] Capacitive loading at the block outputs was based on preliminary floorplan analysis. During the early phase of the design, layout-dependent device models were used for block-level optimization. For UltraSparc™, layout of mega-cells and memory cells was

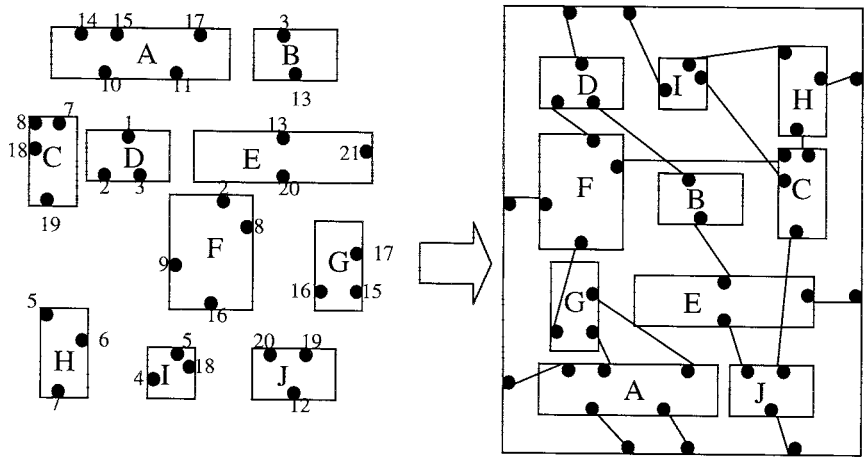


FIGURE 62.12 Example of placement.

done in parallel with RTL design.[30] Initial layout iterations were performed with estimated area and boundaries. There were concurrent chip and block-level designs as well as concurrent datapath and standard cell designs. The concurrency yielded faster turn-around time for logical-physical design iterations. Critical net routing and detailed routing was done after the block-level layout iterations converged.

A survey of CAD tools available on the market for block-level layout is included in Table 62.3. The author presents various steps in the block-level layout process in the following sections. Constraints associated with different block types are also included in the individual sections, wherever applicable.

Placement

The chip planner partitions the circuit into different blocks. Each block consists of a netlist of standard cells or subblocks, whose physical and electrical characteristics are known. For the sake of simplicity, let us only consider a netlist of cells inside the block. The area occupied by each block can be estimated and the number of block-level I/Os (pins) required by each block is known. During the placement step, all of the movable pins of the block and internal cells are positioned on the layout surface, in such fashion that no two cells are overlapping and enough space is left for interconnection among the cells.

Figure 62.12 illustrates an example placement of a netlist. The numbers next to the pins of the cells on the left side specify the nets they are connected to. The placement problem is stated as follows: given an electrical circuit consisting of cells, and a netlist interconnecting terminals on these cells and on the periphery of the block itself, construct a layout indicating positions of these blocks such that all the nets can be routed and the total layout area of the block is minimized. For high-performance microprocessors, an alternative objective is chosen where the placement is optimized to minimize the total delay of the circuit by minimizing lengths of all critical paths subject to a fixed block area constraint. In full-custom style, the placement problem is a packing problem where cells of different sizes and shapes are packed inside the block area.

Various factors affect the decisions taken during placement. We discuss some of the factors. All microprocessor designers may face many additional constraints due to the circuit families, types of libraries, layout methodology, and schedule.

Shape of the cells: In automatic placement tools, the cell are assumed to be rectangular. If the real cell is not rectangular, it may be snapped to an overlapping rectangle. The snapping tends to increase block area. Cells may be flexible and different aspect ratios may be available for each cell. Row-based placement approaches also need standardized height for all the cells.

Routing considerations: All of the tools and algorithms for placement are routing driven. Their objective is to estimate routing lengths and congestions at the placement stage and avoid

unroutability. The cells have to be spaced to allow routing completion. If over-the-cell (OTC) routes are used, then the spacing may be avoided.

Performance: For high-performance circuits, critical nets must be routed within their timing budgets. The placement tool has to operate with a fast and accurate timing analyzer to evaluate various decisions taken during placement. This approach is called performance-driven placement. It forces cells connected to critical nets to be placed very close to each other, which may leave less space for routing that critical net.

Packaging: When the circuit is operational, all cells generate heat. The heat dissipated should be uniform over the entire layout surface of the block. The high power-consuming cells will have to be spaced apart. This approach may directly conflict with performance-driven placement. C4 bumps and power grids may cause some restrictions on allowable locations for some of the cells.

Pre-placed cells: In some cases, the locations of some cells may be fixed or a region may be specified for their placement. For instance, a block-level clock buffer must be at the exact location specified by the clock planner to achieve minimum skew. The placement approach must follow these restrictions.

Special considerations: In microprocessor designs, the placement methodology may be expected to place and sometimes reorder the scan chain. Parts of blocks may be allowed to overlap. Block-level pins may be ordered but not fixed. If the routing plan separates chip and block-level routing layers, there may be areal block-level I/Os in the middle of the layout area.

The CAD algorithms for placement have been thoroughly studied over many decades. The algorithms are classified into simulated annealing-based, partitioning-based, genetic algorithm-based, and mathematical programming-based approaches. All of these algorithms have been extended to performance-driven techniques for microprocessor layouts. For an in-depth analysis of these algorithms, please refer to Refs. 11 and 12.

Global Routing

The placement step determines the exact locations of cells and pins. The nets connecting to those pins have to be routed. The input at a general routing stage consists of a netlist, timing budgets for critical nets, full placement information, and the routing resource specs. Routing resources include available metal layers with obstructions/porosity and their specs include RC delay per unit length on each metal layer and RC delay for each type of via. The objective of routing a block in a microprocessor is to achieve routing completion and timing convergence. In other words, the net loads presented by the final routes must be within the timing budgets. In microprocessor layout, routing also involves special treatment for clock nets, power, and ground lines.

The layout area of the block can be divided into smaller regions. They may be the open spaces not occupied by the cells. These open spaces are called channels. If the routing is only allowed in the open spaces, it is called a channel routing problem. Due to multiple layers available for routing and areal I/Os, over-the-cell routing has become popular. The approach where the whole region is considered for routing with pins lying anywhere in the layout area is called area routing.

Traditionally, the routing problem is divided into two phases. The first phase is called global routing and generates an approximate route for each net. It assigns a list of routing regions to each net without specifying the actual geometric layout of wires. The second phase, called detailed routing, will be discussed in the next subsection.

Global routing consists of three phases: region definition, region assignment, and pin assignment. During definition, the regions are decided by partitioning the routing space into different regions. Each region has a capacity, which means the maximum number of nets that can pass through that region on a layer in a direction. The routing capacity of a region is a function of design rules and wire geometries. During the second phase, nets or parts if the nets, are assigned to various regions, depending on the current occupancy and the net criticality. This phase identifies a sequence of regions through which a net will be routed. Once the region assignment is done, pins are assigned at the boundary of the regions

so that the detailed routing can proceed on each region independently. As long as the pins are fixed at the region boundaries, the whole layout area will be fully connected by abutment.

There is a slight difference between full-custom and automatic layout styles for global routing. In full custom, since regions can be expanded, some violations of region capacities is allowed. However, too many violations may enforce a re-placement.

Some of the factors affecting the decisions taken at global routing are:

Block I/O: Location of block I/Os and their distribution along the periphery may affect region definitions. Areal I/Os need special considerations because they may not lie at a region boundary.

Nets: Multi-terminal nets need special consideration during global routing. There is a different class of algorithms to handle such nets.

Pre-routes: There may be pre-routed nets, like clock, already occupying region capacities. A completely unconnected bus may be passing through the block. Such pre-routes have to be correctly modeled in the region definition.

Performance: Critical nets may have a length and via bound. The number of vias must be minimized for such nets. Critical nets may also need shielding, so they have to be routed next to a power route. Some nets may have spacing requirements with respect to other nets. Some nets may be wider than others, and the region occupancy must include the extra resources required for wide routes.

Detailed router: The type and style of detailed routing affects the decisions taken during the global routing. The detailed router may be a channel router, for which pins must be placed on the opposite sides of the region. In some cases, the detailed router may need information about via bounds from the global router.

Global routing is typically studied as a graph problem. There are three types of graph models to represent regions and their capacities, namely, the grid graph model, the checker board model, and the channel intersection graph model. For two terminal nets, there are three types of global routing algorithms: maze routing, line-probe, and shortest path based. For multi-terminal routing, Steiner tree-based approaches are very popular. There are some mathematical formulations for global routing; however, they provide solutions on small blocks only.

Detailed Routing

Global routing uses the original net information and separates the routing problem into a set of restricted region routing problems. A routing region can be a channel (pins on opposite sides), a 2-D switchbox (pins on all sides in 2-D), or a 3-D switchbox (pins on all faces in 3-D). The detailed router places the actual wire segments within the regions, thus completing the required connection between the cells. There is a limited scope for the regions to expand into other regions. A detailed router has to intelligently order the regions to be routed, depending on the occupancy and criticality. Factors affecting detailed routing are:

Metal layers: Traditionally, two or three routing layers were available at the block-level detailed routing. There are numerous techniques published for two- or three-layer detailed routing. Today's microprocessors consist of four or five metal layers. The number of layers is likely to increase to ten in the near future. A detailed router should fully utilize the available layers. Their widths, spacing, pitch, and electrical requirements must be obeyed. Obstructions must be handled on all metal layers.

Via: The via count is of major concern in detailed routing and must be minimized to improve performance and area. Vias impact manufacturability, cause RC delays, signal reflections, and transmission line effects. They also make post-layout compaction difficult.

Nets: Traditionally, a multi-terminal net is decomposed into a set of two terminal nets for ease of routing. Current approaches handle multi-terminal nets directly. Variable-width nets need special attention during detailed routing. In high-performance designs, nets may also be tapered, that is,

the same routing segment of a net may have variable widths. The detailed router should support tapering. Due to the criticality, some nets may be required to be routed across all the regions before the rest of the nets. This breaks the paradigm for sequential region routing, unless such nets are modeled as pre-routes.

Region specs: Depending on the type of the region, pins may be located at various boundaries or faces. Regions may be flexible to some extent. However, the detailed router must try not to exceed the region bounds.

Gridding: A detailed router may assume wire gridding, implying that the pitch of wires on any metal layer is considered fixed. All pins in the regions and on the cell are on the routing grid specified by the detailed router. The layout area can be modeled as an array of grid points. Hence, the routing is very fast. Gridding hinders routing with variable-width variable spacing of metal layers. It can be accomplished at the cost of area. Hence, non-gridded routers are used in microprocessors for critical net routing.

Until the process technology advanced to the point when over-the-cell (OTC) routing became feasible, channel routing was the most popular area of research for CAD. The channel routing approaches are classified into algorithms for a single layer, a single row, two layers, and three layers. Multi-layer channel routing algorithms have also been published. Channel routing approaches can also be extended to switchboxes. The switchbox routing is not guaranteed to complete. A rip-up and re-route utility is added to the detailed routers for switchboxes.

Let us understand some of the routing tools and methodologies followed internal to various microprocessor companies. IBM developed a grid-based router to connect blocks together.[5] For the G4 processor, they employed two strategies. In the first method, chip-level routing was performed without any blockages from the block level.[24] Then, the block level routes tap the chip-level shadows appropriately. This approach was used only where wiring resources were limited. In the alternative method, the wiring tracks were divided between chip and block level. The negative image of each level was available at the other level. Pre-routes were also supported. The second method enables parallel routing effort while the first enables efficient use of wiring resources. Long routes were split at appropriate places and buffers (repeaters) were placed to minimize delays.

In HP's PA-8000, the block router is really pushing the limits of technology. It achieves high routing completion, supports multi-width wires, optimizes the ratio of wire area/block area, has a fast turnaround time, and strictly follows a rigid placement model.[31] The router was originally a channel router with blocks and channels, but it was modified for multiple layers. The placement of C4 I/O bumps is fixed. Changes in locations of bumps may cause alpha-particle emission. Hence, metal5 was not included with other layers during automatic routing. Routing channels were not expandable, but they could be moved. An electrical model of the block I/Os was supplied to the router. The area routing problem was converted to channels with blockages so that an in-house channel router could be used. L-shaped blocks were cut into two rectangular blocks, but intelligent port placement and constraints bound them together so that the same block router was used. In earlier HP processors, the ports were at the block boundary. In PA-8000, over-the-block (OTB) routing was supported. Blocks were considered black-boxes at the chip level and no internals were supplied to the router; however, an abstract virtual grid model of each block was available. The grid model enabled the lowest cost path of a global net to traverse through any region over a block. The router minimized jogging and distributed unavoidable jogs to reduce congestion. A sophisticated net flow optimizer was developed for obstacles, ports inside the block, jog allocation, and optimal exit points to avoid jogging. A density estimator was used for close estimation of detailed routing. It had port models and net characteristics for multi-terminal net routing. The topology of ports and obstacles was negotiated between the chip and block layouts. The OTB router supported variable widths and spacing. A graph theoretic approach was used to allocate trunks in channels with obstacles. The routers did not support crosstalk or delay modeling. When these violations occurred, jog insertion and wrong-side segmenting was employed. The router always finished routing under constrained placement and reported spacing problems.

Compaction

The original idea behind compaction was to improve layout productivity. The designers were free to explore alternative layout strategies and generate a topological design without geometrical details. The compaction tool was expected to produce a correct geometrical design from the topological design that completely satisfied all of the design rules of the manufacturing process.[32] The approaches employing hierarchical compaction helped in chip planning and assembly because the compactors had flexibility to choose interconnections, abutment, routing area, etc.

Today, compactors are used to minimize layout area after detailed routing. They are used as automatic tools or layout aids. Due to excessive area allotment by the chip planner, sub-optimal layout algorithms, or local optimization of internal layout, some vacant space is present in the block layout area. The goal of compaction is to minimize layout without violating design rules, without significant changes to the existing layout topology, and without violating the designer specified constraints.[11] The main idea is to reduce the space between features as much as possible without violating spacing design rules. Compaction can also be used when scaling down a design to a new set of process rules. The features can be regenerated to the new process spec and the empty area around the features can be recovered using compaction.[12]

A compactor needs three things: the initial layout representation, technology information, and a compaction strategy. The same approach can be applied to full-custom and automatic layout styles because there is no apparent difference between the three inputs generated by both styles.

The initial layout is represented as a constraint graph or a virtual grid. The former represents connection and separation rules as linear inequalities, which can be modeled as a weighted directed graph. A separation constraint leads to one inequality, while a connection constraint leads to two. Shadow propagation and scanlines are two examples of techniques to generate constraint graphs. The latter representation requires that each component be attached to a grid line on the layout grid. The minimum distance between grid lines is the maximum separation required between any two features occupying the grid lines. This representation leads to very fast and simple algorithms, but does not produce as good results as the constraint graph representation. All compactors allow the designers to specify additional constraints specific to a circuit.

The most popular strategy is 1-D compaction. The layout is compacted along the *x*-direction, followed by a compaction in the *y*-direction. Longest path or network flow methods are commonly used for 1-D compaction. As the full 2-D view is not available, the results may be inferior to 2-D strategy. The reader should note that the 2-D compaction problem is proven to be NP-complete. The 2-D problem is solved by an integer linear programming technique, whose complexity is exponential. So the 2-D approach is impractical even for moderate-sized circuits. There are 1½-D approaches employing zone refinement techniques, but they change the original topology of the layout.

Hierarchical compaction strategies are used to compact a full chip or large blocks. In this approach, hierarchical input representation is generated at each level of the hierarchy from the bottom up. Initially, leaf-level individual blocks or subblocks are compacted and then layout of group of blocks is compacted. Finally, a flat level compactor can also be used for generating a compact cell library.

CAD Tools

Surveys of the latest CAD tools for block-level layout are available in Refs. 25 and 33. The routers are classified into three stages. Stage 1 routing means point-to-point single-width routing without any electrical info; stage 2 means routing with geometric data and design rules, and stage 3 means interconnect RC aware routing. All tools interact with the floorplan. They consider length, timing, routability, and use automatic cell padding to minimize congestion. Some tools also perform scan chain reordering. Placement with estimated global routing is a very common feature. The tools are very mature and widely used. However, some physical design problems stem less from the technical challenge than from the lack of industry standards. Except for GDSII, there are no standard data formats. One cannot easily represent block boundaries, dimensions, ports, channel locations, connection points, open spaces for OTC across all the tools. Microprocessor layout teams go through strenuous processes to integrate point tools from various vendors to work as a common tool suite.

TABLE 62.3 Currently Available Block-Level Tools

Company	Internet	Tool	Block Type	Description
Arcadia Design Systems	www.arcadiadesign.com	Mustang	Datapath	Regularity extraction and placement
Avant! Corp.	www.avanticorp.com	Apollo	Control, mega-blocks	All path timing-driven place and route
Cadence	www.cadence.com	Silicon Ensemble	Control, mega-blocks	Timing-driven place and route
Cadence	www.cadence.com	IC Craftsman	All	Detailed routing
Duet Technologies	www.duettech.com	Epoch	Control	Placement and timing-driven routing
Everest Design Automation	www.everest-da.com	(Under development)	Control	Interconnect design, physical floorplannig, gridless routing
Gambit Automated Design	www.gambit.com	Grandmaster	Control	Parallel processing-based place and route
Mentor Graphics Corp.	www.mentorg.com	IC Station	Control, mega-blocks	Cell-based place and route
Snaketech, Inc.	www.snaketech.com	Cellsnake	Control	For cell-based ICs
Stanza Systems, Inc.	www.stanzas.com	PolarSLE	All	Custom layout editor with router
Sycon Design, Inc.	www.sycon-design.com	Tempest-Cell	All	Layout synthesis, structured custom style or block-level place and route
Tanner EDA	www.tanner.com	Tanner Tools Pro	Control	Editing, placement, routing, simulation
Timberwolf Systems, Inc.	www.twolf.com	TimberWolf	Control	Placement, global routing, detailed routing

There are three types of constraint driven routing tools: channel routing, area routing, and hybrid routing. In channel routing, the die size is unknown. Hence, they force an additional floorplanning iteration. Area routers try to finish routing even if they violate design rules

The major vendor for block-level placement and routing tools is Cadence (www.cadence.com). It is supplying fundamentally new engines. There is a new timing-driven flow with no need to re-synthesize. Buffer optimization is done during placement. It will soon include an extraction capability and analysis of crosstalk, electromigration, and hot electron effects. The new Warp router eliminates clock skew. Cadence also supplies a detailed router, IC craftsman, capable of shape-based routing. It is a stage 3 router. The warp router will have the same capability soon. Currently available block-level layout tools are presented in Table 62.3. The reader should note that all of the automatic tools also support manual editing. So they can be used as layout editors for full custom techniques.

Physical Verification

Let us re-visit the physical design flow described earlier. The chip planner partitions the chip into blocks, the blocks are floorplanned, critical signals are routed, the blocks are laid out, and finally the chip is assembled. A large database of polygons representing the physical features inside the chip is generated. The chip layout represented in the database must be verified against the high-level architectural goals of the microprocessor, such as frequency, power, manufactuarability, etc. Post-silicon debug is an expensive process. In some cases, editing the manufactured die may be impossible. Physical verification is the last, but very important step during microprocessor layout method. If a serious design rule or timing violation is observed, the entire layout process may have to be re-visited, followed by re-verification.

The reader may be aware of commonly used terms during physical verification: post-layout performance verification (PLPV), design rule checking (DRC), electrical rule checking (ERC), and layout verification system (LVS). ERC and PLPV involve extracting the layout in the form of electrical elements and analyzing the electrical representation of the circuit by simulation methods. Some CAD vendors and

microprocessor design teams are investing in new tools to reveal the full effects of a circuit's parasitic coupling, delays, degradation, signal integrity, crosstalk, IR drops, hot spots from thermal build-up, charge accumulation, electromigration, etc. Simulation and electrical analysis is beyond the scope of this chapter.

There are two types of design rules checked during DRC. The first type are composition rules, which describe how to construct components and wires from the layers that can be fabricated. The other type are spacing rules, which describe how far apart objects in the layout must be for them to be reliably built.[32] Adherence to both types is required during DRC. The rules are checked by expanding the components and wires into rectangles as specified by their design rule views.

Due to the confidential nature of manufacturing process, the exact details of the verification methods are proprietary to the microprocessor manufacturers. There is a significant gap between silicon capabilities and CAD tools on the market.[29] The high-performance requirements need verification to be done at greater levels of detail and accuracy. Due to the large number of transistors in a microprocessor, there is an explosion of layout data. To solve this problem, verification should provide a close interaction between front-end design and back-end layout. It should be able to operate on approximate data available at various stages of the layout to identify potential problems related to power, signal integrity, electromigration, electromagnetic interference, reliability, and thermal effects.

The challenges involved in physical verification and available vendor tools for automatic verification are presented in Ref. 33. These tools are modified inside the microprocessor design teams to conform to the confidential manufacturing and architectural specification. The basic problem suffered by all tools is too much data from accurate physical analysis. In a typical microprocessor, there may be 500,000 nets, which lead to 21 million coupling capacitors and 2.5 million resistances. Hence, fast and accurate verification is a problem. The number of parasitic effects and circuit data is growing with every microprocessor generation. Unless efficient physical verification tools are available, over-engineering will continue to compensate for the uncertainty in final parasitics. Process shrinks are causing more layers, more interconnect, 3-D capacitive effects, and even inductive effects. The lack of efficient verification tools prohibits further feature shrinks. Verification has to be a complex set of algorithms handling large data. There is a need for incremental and hierarchical systems that have new parasitic extractors, circuits analyzers, and optimizers. Some microprocessor layout designers have employed automatic updates of routed edges, non-uniform etching, and remedies for the antenna effect.

Let us discuss some verification approaches followed by leading microprocessor manufacturers. Alpha 21264 includes very high-speed circuits and the layout was full-custom.[8] It needed careful and detailed post-layout electrical verification. No CAD tools capable of handling this were available. Therefore, an internally developed simulator was used. It is non-logic; that is, it checks timing behavior, electrical hazards, reliability, charge sharing, IR noise, interconnect capacitance, noise-induced minority carrier injection, circuit topology violations, dynamic nodes, latches, stack height minimization, leaker usage, fan-in-fan-out restrictions, wireability, beta ratios, races, edge rates, and delays.

The verification for the G4 microprocessor at IBM was divided between chip level and block level.[24] The modeling had three levels of accuracy: namely, statistical, Steiner, and detailed RC. Pathmill[2] was used for timing analysis. The verification tool extracted and analyzed the layout and inserted decoupling capacitors, wide wires, and repeaters automatically. If a full-chip long net was found not to meet its timing, a repeater had to be inserted on the net. IBM observed a problem with the repeater insertion methodology. What if the die does not have a space at the location of the repeater to be inserted? Some space had to be deliberately created for this problem.

In UltraSparc-I™, the power network was extensively verified using an internal tool called PGRID.[9] The block-level layout was translated into a schematic model for the chip-level verification. The voltages at four corners of a block were extracted from HSPICE runs. Finally, a graphical error map for electromigration and IR drop violations was generated at all levels of the layout.

[2]A tool from Synopsys.

References

1. T. Jamil, Fifth-generation microprocessors, *IEEE Potentials,* 15(5), 33, Dec. 1996-Jan. 1997.
2. R.N. Noyce, Microelectronics, *Scientific American,* 237(3), 65, Sept. 1977.
3. M.K. Gowan, L.L. Biro, and D.B. Jackson, Power considerations in the design of the alpha 21264 microprocessor, *Proceedings of Design Automation Conference,* pp. 726-731, 1998.
4. M. Matson et al., Circuit Implementation of a 600 MHz Superscalar RISC Microprocessor, *ICCD 98,* pp. 104-110, 1998.
5. S. Posluszny et al., Design Methodology for a 1.0 GHz microprocessor, *ICCD,* pp. 17-23, 1998.
6. A. Kumar, The HP PA-8000 RISC CPU, *IEEE Micro.,* 17, 27, 1997.
7. G. Gerosa, A 250 MHz 5-W PowerPC microprocessor with on-chip L2 cache controller, *IEEE Journal of Solid State Circuits,* 32, 11, 1997.
8. Gronowski et al., High-performance microprocessor design, *IEEE Journal of Solid-State Circuits,* 33(5), 676, 1998.
9. A. Dala, L. Lev, and S. Mitra, Design of an efficient power distribution network for the UltraSPARC-I™ Microprocessor, *Proceedings of ICCD,* pp. 118-123, 1995.
10. K. Diefendorff, K7 Challenges *Intel. Microprocessor Report,* 12, Oct. 26, 1998.
11. N. Sherwani, *Algorithms for VLSI Physical Design Automation,* 2nd ed., Kluwer Academic Publishers, 1995.
12. S.M. Sait and H. Youssef, *VLSI Physical Design Automation Theory and Practice,* McGraw-Hill, 1995.
13. N.H.E. Weste and K. Eshraghian, *Principles of CMOS VLSI Design — A Systems Perspective,* 2nd ed., Addison Wesley, 1993.
14. S.M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits Analysis and Design,* McGraw-Hill, 1996.
15. R.J. Baker, H.W. Li, and D.E. Boyce, *CMOS Circuit Design, Layout and Simulation,* IEEE Press, 1998.
16. D.P. LaPotin, Early assessment of design, packaging and technology tradeoffs, *International Journal of High Speed Electronics,* 2(4), 209, 1991.
17. G. Bassak, Focus Report: IC physical design tools, *Integrated System Design Magazine,* Nov. 1998.
18. P.J. Dorweiler, F.E. Moore, D.D. Josephson, and G.T. Colon-Bonet, Design methodologies and circuit design tradeoffs for the HP PA 8000 processor, *Hewlett-Packard Journal,* 48, 16, Aug. 1997.
19. E. Malavasi, E. Charbon, E. Feit, and A. Sangiovanni-Vincentelli, Automation of IC layout with analog constraints, *IEEE Transactions on CAD,* 15, 923, Aug. 1996.
20. D. Trobough, IC design drives array packages, *Integrated System Design Magazine,* Aug. 1998.
21. Farbarik et al., CAD tools for area-distributed I/O pad packaging, *Proceedings of 1997 IEEE Multi-Chip Module Conference,* pp. 125-129, 1997.
22. B.T. Preas and M.J. Lorenzetti, Physical design automation of VLSI Systems, *Introduction to Physical Design Automation,* Benjamin Cummings, Menlo Park, CA, 1988.
23. N. Sherwani, Panel Discussion, *International Symposium on Physical Design,* Monterey, CA, Apr. 1998.
24. K.L. Sheperd et al., Design methodology for the high performance G4 S/390 microprocessor, *ICCAD,* pp. 232-240, 1997.
25. [Schultz 97].
26. H. Fair and D. Bailey, Clocking design and analysis for a 600 MHz alpha microprocessor, *ISSCC Digest of Technical Papers,* pp. 398-399, Feb. 1998.
27. A. Dharchoudhury, R. Panda, D. Blauuw, and R. Vaidyanathan, Design and analysis of power distribution networks in PowerPC microprocessors, *Proceedings of Design Automation Conference,* pp. 738-743, 1998.
28. R.T. Maniwa, Focus report: design libraries, *Integrated System Design Magazine,* Aug. 1997.
29. T. Maniwa, Physical verification: challenges and problems for new designs, *Integrated System Design Magazine,* Nov. 1998.

30. A. Cao et al., CAD Methodology for the design of UltraSPARC™-I Microprocessor at Sun Microsystems, Inc., *Proceedings of 32nd Design Automation Conference,* pp. 19-22, 1995.
31. J.C. Fong, H.K. Chan, and M.D. Kruckenberg, Solving IC interconnect routing for an advanced PA-RISC processor. *Hewlett-Packard Journal,* 48(4), 40, Aug. 1997.
32. W.J. Wolf and A.E. Dunlop, Symbolic layout and compaction, Chapter 6 in *Physical Design Automation of VLSI Systems,* Benjamin Cummings, Menlo Park, CA, 1988.
33. G. Bassak, Focus report: physical verification tools, *Integrated System Design Magazine,* Feb. 1998.

Connors, D.A., Hwu, W. "Architecture"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

63 Architecture

Daniel A. Connors
Wen-mei W. Hwu
University of Illinois at Urbana-Champaign

63.1 Introduction

The microprocessor industry is divided into the computer and embedded sectors. Both computer and embedded microprocessors share aspects of computer design, instruction set architecture, organization, and hardware. The term "computer architecture" is used to describe these fundamental aspects and, more directly, refers to the hardware components in a computer system and the flow of data and control information among them. In this chapter, various types of microprocessors will be described, fundamental architecture mechanisms relevant in the operation of all microprocessors will be presented, and microprocessor industry trends discussed.

63.2 Types of Microprocessors

Computer microprocessors are designed for use as the Central Processing Units (CPU) of computer systems such as personal computers, workstations, servers, and supercomputers. Although microprocessors started as humble programmable controllers in the early 1970s, virtually all computer systems built in the 1990s use microprocessors as their central processing units. The dominating architecture in the computer microprocessor domain today is the Intel 32-bit architecture, also known as IA-32 or X86. Other high-profile architectures in the computer microprocessor domain include Compaq-Digital Alpha, HP PA-RISC, Sun Microsystems SPARC, IBM/Motorola PowerPC, and MIPS.

Embedded microprocessors are increasingly used in consumer and telecommunications products to satisfy the demands for quality and functionality. Major product areas that require embedded microprocessors include digital TV, digital cameras, network switches, high-speed modems, digital cellular phones, video games, laser printers, and automobiles. Future improvements in energy consumption, fabrication cost, and performance will further enable new applications such as the hearing aid. Many experts expect that embedded microprocessors will form the fastest growing sector of the semiconductor business in the next decade.[1]

Embedded microprocessors have been categorized into DSP processors and embedded CPUs due to historic reasons. DSP processors have been designed and marketed as special-purpose devices that are mostly programmed by hand to perform digital signal processing computations. A recent trend in the DSP market is to use compilers to alleviate the need for tedious hand-coding in DSP development. Another recent trend in the DSP market is toward integrating a DSP processor core with application-specific logic to form a single-chip solution. This approach is enabled by the fast increasing chip density technology. The major benefit is reduced system cost and energy consumption. Two general types of DSP cores are available to application developers today. Foundry-captive DSP cores and related application-specific logic design services are provided by major semiconductor vendors such as Texas Instruments, Lucent Technologies, and SGS-Thompson to application developers who commit to their fabrication lines. A very large volume commitment is usually required to use the design service. Licensable DSP cores are provided by small to medium design houses to application developers who want to be able to choose fabrication lines.

There are several ways that the needs of embedded computing differ from those of more traditional general-purpose systems. Constraints on the code size, weight, and power consumption place stringent requirements on embedded processors and the software they execute. Also, constraints rooted in real-time requirements are often a significant consideration in many embedded systems. Furthermore, cost is a severe constraint on embedded processors.

Embedded CPUs are used in products where the computation involved resembles that of general-purpose applications and operating systems. Embedded CPUs have been traditionally derived from out-of-date computer microprocessors. They often reuse the compiler and related software support developed for their computer cousins. Recycling the microprocessor design and compiler software minimizes engineering cost. A trend in the embedded CPU domain is similar to that in the DSP domain: to provide embedded CPU cores and application specific logic design services to form single-chip solutions. For example, MIPs customized its embedded CPU core for use in Nintendo64, in return for engineering fees and royalty streams. ARM, NEC, and Hitachi offer similar products and services. Due to an increasing need to perform DSP computation in consumer and telecommunication products, an increasing number of embedded CPUs have extensions to enable more effective DSP computation.

Contrary to the different constraints and product markets, both computer and embedded microprocessors share traditional elements of computer architecture. These main elements will be described. Additionally, over the past decade, substantial research has gone into the design of microprocessors embodying parallelism at the instruction level, as well as aggressive compiler optimization and analysis techniques for harnessing this opportunity. Much of this effort has since been validated through the proliferation of mainstream general-purpose computers based on these technologies. Nevertheless, growing demand for high performance in embedded computing systems is creating new opportunities to leverage these techniques in application-specific domains. The research of Instruction-Level Parallelism (ILP) has developed a distinct architecture methodology referred to as Explicitly Parallel Instruction Computing (EPIC) technology. Overall, these techniques represent fundamental substantial changes in computer architecture.

63.3 Major Components of a Microprocessor

The main hardware of a microprocessor system can be divided into sections according to their functionalities. A popular approach is to divide a system into four subsystems: the central processor, the memory subsystem, the input/output (I/O) subsystem, and the system interconnection. Figure 63.1 shows the connection between these subsystems. The main components and characteristics of these subsystems will be described.

Central Processor

A modern microprocessor's central processor system can typically be further divided into control, data path, pipelining, and branch prediction hardware.

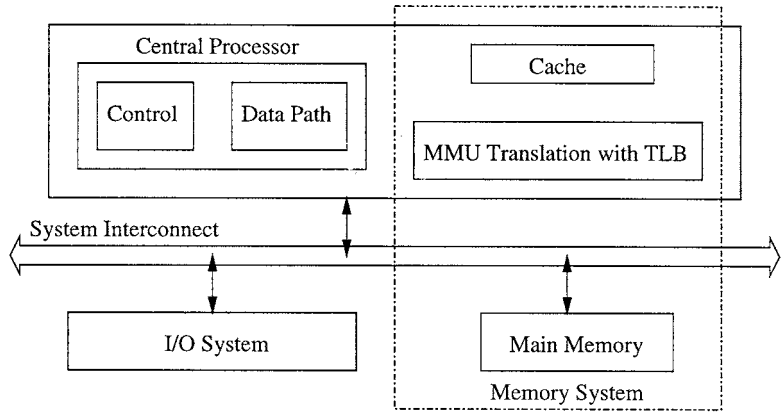


FIGURE 63.1 Architecture subsystems of a computer system.

Control Unit

The *control unit* of a microprocessor generates the control signals to orchestrate the activities in the data path. There are two major types of communication lines between the control unit and the data path: the control lines and the condition lines. The *control lines* deliver the control signals from the control unit to the data path. Different signal values on these lines trigger different actions in the data path. The *condition lines* carry the status of the execution from data path to the control unit. These lines are needed to test conditions involving th registers in the data path in order to make future control decisions. Note that the decision is made in the control unit but the registers are in the data path. Therefore, the conditions regarding the register contents are formed in the data path and then shipped to the control unit for decision-making. A control unit can be implemented with hardwiring, microprogramming, or a combination of both.

In a hardwired design, each control unit is viewed as an ordinary sequential circuit. The design goals are to minimize the component count and to maximize the operation speed. The finite state machine is realized with registers, logic, and wires. Once constructed, the design can be changed only through physically rewiring the unit. Therefore, the resulting circuits are called *hardwired control units.* Due to design optimizations, the resulting circuits often exhibit little structure. The lack of structure makes it very difficult to design and debug complicated control units with this technique. Therefore, hardwiring is normally used when the control unit is relatively simple.

Most of the design difficulties in the hardwired control units are due to the effort of optimizing the combinational circuit. If there is a method that does not attempt to optimize the combinational circuit, the design complexity could be significantly reduced. One obvious option is to use either Read-Only Memory (ROM) or Random Access Memory (RAM) to implement the combinational circuit. A control unit whose combinational circuit is simplified by the use of ROM or RAM is called a *microprogrammed control unit.* The memory used is called *Control Memory* (CM). The practice of realizing the combinational circuit in a control unit with ROM/RAM is called *microprogramming.* The concept of microprogramming was first introduced by Wilkes.

The idea of using a memory to implement a combinational circuit can be illustrated with a simple example. Assume that we are to implement a logic function with three input variables, as described in the truth table illustrated in Fig. 63.2(a). A common way to realize this function is to use Karnaugh maps to derive highly optimized logic and wiring. The result is shown in Fig. 63.2(b). The same function can also be realized in memory. In this method, a memory with eight 1-bit locations can be used to retain the eight possible combinations of the three-input variable. Location *i* contains an F value corresponding to the *i*-th input combination. For example, location 3 contains the F value (0) for the input combination 011. The three input variables are then connected to the address input of the memory to complete the design (Fig. 63.2(c)). In essence, the memory implicitly contains the entire truth table. Considering the

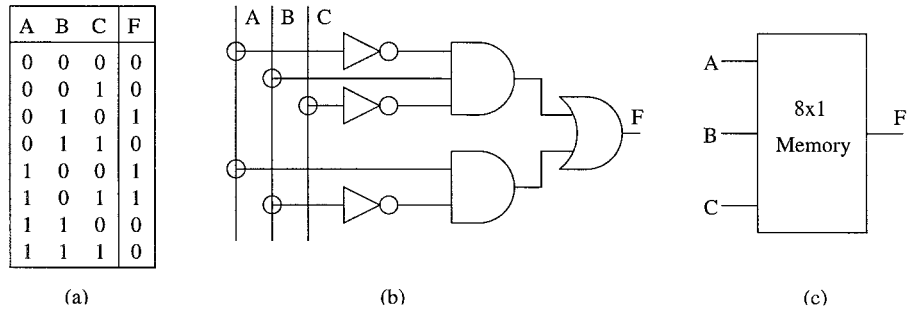


FIGURE 63.2 Using memory to simplify logic design: (a) Karnaugh map, (b) logic, (c) memory.

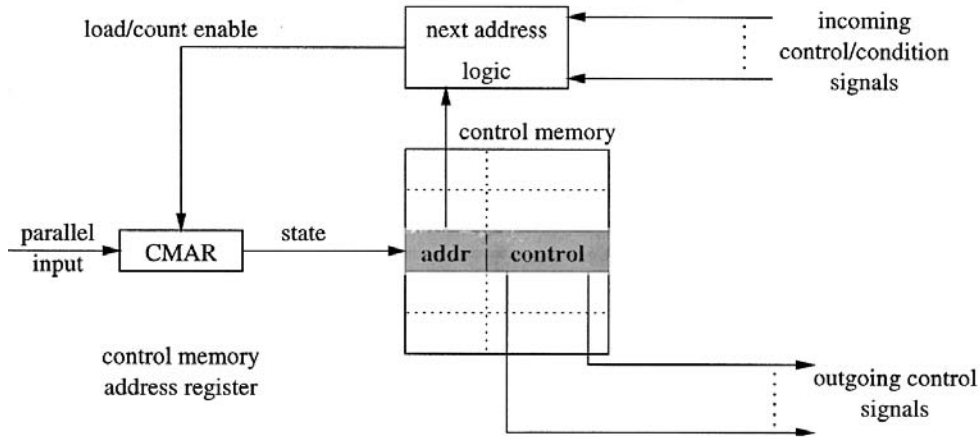


FIGURE 63.3 Basic model of microprogrammed control units.

decoding logic and storage cells involved in a 8×1 memory, it is obvious that the memory approach uses a lot more hardware components than the Karnaugh map approach. However, the design is much simpler in the memory approach.

Figure 63.3 illustrates the general model of a microprogrammed control unit. Each control memory location consists of an address field and some control fields. The address field plus the next address logic implements the combinational circuit for generating the next state value. The control fields implement the combinational circuit for generating the control signal. Both the control memory and the next addrss logic will be studied in detail in this section. The state register/counter has been renamed the *Control Memory Address Register* (CMAR) for an obvious reason: the contents of the register are used as the address input to the control memory. An important insight is that the CMAR stores the state of the control unit.

Data Path

The data path of a microprocessor contains the main arithmetic and logic execution units required to execute instructions. Designing the data path involves analyzing the function(s) to be performed, then specifying a set of hardware registers to hold the computation state, and designing computation steps to transform the contents of these registers into the final result. In general, the functions to be performed will be divided into steps, each of which can be done with a reasonable amount of logic in one clock cycle. Each step brings the contents of the registers closer to the final result. The data path must be equipped with a sufficient amount of hardware to allow these computation steps in one clock cycle. The data path of a typical microprocessor contains integer and floating-point register files, ten or more

functional units for computation and memory access, and pipeline registers. One must understand the concept of pipelining in order to understand the data paths of today's microprocessors.

Pipelining

In the 1970s, only supercomputers and mainframe computers were pipelined. Today, most commercial microprocessors are pipelined. In fact, pipelining has been a major reason why microprocessors today outperform supercomputers built less than 10 years ago. Pipelining is a technique to coordinate parallel processing of operations.[2] This technique has been used in assembly lines of major industries for more than a century. The idea is to have a line of workers specializing in different pieces of work required to finish a product. A conveying belt carries each product through the line of workers. Each worker will do a small piece of work on each product. Each product is finished after it is processed by all the workers in the assembly line.

The obvious advantage of pipelining is to allow one worker to immediately start working on a new product after finishing the work on a current product. The same methodology is applied to instruction processing in microprocessors. Figure 63.4(a) shows an example five-stage pipeline dividing instruction execution into Fetch (F), Decode (D), Execute (E), Memory (M), and Write-back (W) operations, each requiring various stage-specific logic. Between each stage is a stage register (SR) used to hold the instruction information necessary to control the instruction. A very basic principle of pipelining is that the work performed by each stage must take about the same amount of time. Otherwise, the efficiency will be significantly reduced because one stage becomes a bottleneck of the entire pipeline. Similarly, the time duration of the slowest pipeline stage determines the overall clock frequency of the processor. Due to this constraint and the characteristics of memory speeds, the five-stage pipeline model often requires some of the principle five stages to be divided into smaller stages. For instance, the memory stage may be divided into three stages, allowing memory accesses to be pipelined and the overall processor clock speed to be a function of a fraction of the memory access latency.

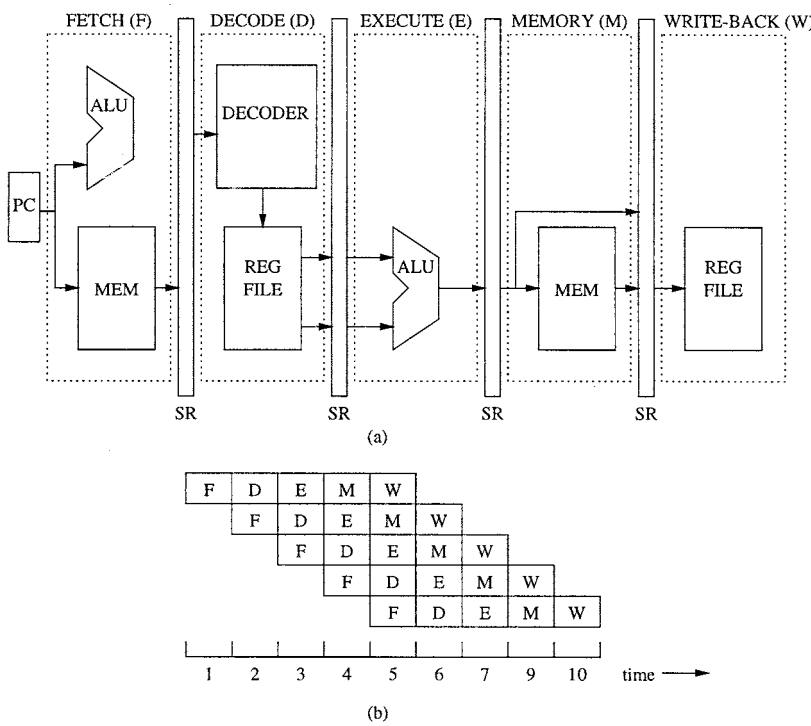


FIGURE 63.4 Pipeline architecture: (a) machine, (b) overlapping instructions.

The time required to finish N instructions in a pipeline with K stages can be calculated. Assume a cycle time of T for the overall instruction completion, and an equal T/K processing delay at each stage. With a pipeline scheme, the first instruction completes the pipeline after T, and there will be a new instruction out of the pipeline per stage delay T/K. Therefore, the delays of executing N instructions with and without pipelining, respectively, are:

$$T*\left(N\right) \tag{63.1}$$

$$T+\left(T/k\right)*\left(N-1\right) \tag{63.2}$$

There is an initial delay in the pipeline execution model before each stage has operations to execute. The initial delay is usually called *pipeline start-up delay* (P), and is equal to total execution time of one instruction. The speed-up of a pipelined machine relative to a non-pipelined machine is calculated as:

$$\frac{P*N}{P+\left(N-1\right)} \tag{63.3}$$

When N is much larger than the number of pipestages P, the ideal speed-up approaches P. This is an intuitive result since there are P parts of the machine working in parallel, allowing the execution to go about P times faster in ideal conditions.

The overlap of sequential instructions in a processor pipeline is shown in Fig. 63.4(b). The instruction pipeline becomes full after the pipeline delay of $P = 5$ cycles. Although the pipeline configuration executes operations in each stage of the processor, two important mechanisms are constructed to ensure correct functional operation between dependent instructions in the presence of data hazards. Data hazards occur when instructions in the pipeline generate results that are necessary for later instructions that are already started in the pipeline. In the pipeline configuration of Fig. 63.4(a), register operands are initially retrieved during the decode stage. However, the execute and memory stage can define register operands and contain the correct current value but are not able to update the register file until the later write-back execution stage. Forwarding (or bypassing) is the action of retrieving the correct operand value for an executing instruction between the initial register file access and any pending instruction's register file updates. Interlocking is the action of stalling an operation in the pipeline when conditions cause necessary register operand results to be delayed. It is necessary to stall early stages of the machine so that the correct results are used, and the machine does not proceed with incorrect values for source operands. The primary causes of delay in pipeline execution are initiated due to instruction fetch delay and memory latency.

Branch Prediction

Branch instructions pose serious problems for pipelined processors by causing hardware to fetch and execute instructions until the branch instructions are completed. Executing incorrect instructions can result in severe performance degradation through the introduction of wasted cycles into the instruction stream.

There are several methods for dealing with pipeline stalls caused by branch instructions. The simplest performance scheme handles branches by treating every branch as either *taken* or *not-taken*. This treatment can be set for every branch or determined by the branch opcode. The designation allows the pipeline to continue to fetch instructions as if the branch was a normal instruction. However, the fetched instruction may need to be discarded and the instruction fetch restarted when the branch outcome is incorrect. *Delayed branching* is another scheme which treats the set of sequential instructions following a branch as delay slots. The delay-slot instructions are executed whether or not the branch instruction is taken. Limitations on delayed branches are caused by the compiler and program characteristics being unable to support numerous instructions that execute independent of the branch direction. Improvements have

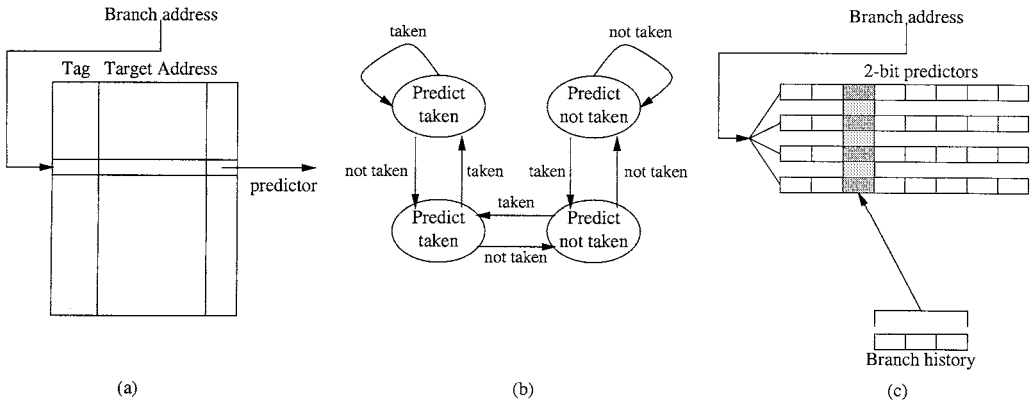


FIGURE 63.5 Branch prediction.

been introduced to provide *nullifying* branches, which include a predicted direction for the branch. When the prediction is incorrect, the delay-slot instructions are nullified.

A more modern approach to reducing branch penalties uses hardware to dynamically predict the outcome of a branch. Branch prediction strategies reduce overall branch penalties by allowing the hardware to continue processing instructions along the predicted control path, thus eliminating wasted cycles. Efficient execution can be maintained while branch targets are correctly predicted. However, a large performance penalty is incurred when a branch is mispredicted. Branch target buffer is a cache structure that is accessed in parallel with the instruction fetch. It records the past history of branch instructions so that a prediction can be made while the branch is fetched again. This prediction method adapts the branch prediction to the run-time program behavior, generating a high prediction accuracy. The target addresses of the branch is also saved in the buffer so that the target instruction can be fetched immediately if a branch is predicted taken.

Several methodologies of branch target prediction have been constructed.[3] Figure 63.5 illustrates several general branch prediction schemes. The most common implementation retains history information for each branch as shown in Fig. 63.5(a). The history includes the previous branch directions for making predictions on future branch directions. The simplest history is last-taken, which uses 1-bit to recall whether the branch condition was taken or not-taken. A more effective branch predictor uses a 2-bit saturating state history counter to determine the future branch outcome similar to Fig. 63.5(b). Two bits rather than one bit allows each branch to be tagged as strongly or weakly taken or not-taken. Every correct prediction reinforces the prediction, while an incorrect prediction weakens it. It takes two consecutive mispredictions to reverse the direction (whether taken or not taken) of the prediction.

Recently, more complex two-level adaptive branch prediction schemes have been built which use two levels of branch history to make predictions, as shown in Fig. 63.5(c). The first level is the branch outcome history of the last branches encountered. The second level is the branch behavior for the last occurrences of a specific pattern of branch histories. There are alternative ways of constructing both levels of adaptive branch prediction schemes, the mechanisms can contain information that is either based on individual branches, groups (set-based), and all (global). Individual formation contains the branch history for each branch instruction. Set-based information groups branches according to their instruction address, thereby forming sets of branch history. Global information uses a global history containing all branch outcomes. The second level containing branch behaviors can also be constructed using any of the three types. In general, the first-level branch history pattern is used as an index into the second-level branch history.

Memory Subsystem

The *memory system* serves as a repository of information in a microprocessor system. The processing unit retrieves information stored in memory, operates on the information, and returns new information

back to memory. The memory system is constructed of basic semiconductor DRAM units called modules or banks.

There are several properties of memory, including speed, capacity, and cost that play an important role in the overall system performance. The speed of a memory system is the key performance parameter in the design of the microprocessor system. The *latency* (L) of the memory is defined as the time delay from when the processor first requests data from memory until the processor receives the data. *Bandwidth* is defined as the rate which information can be transferred from the memory system. Memory bandwidth and latency are related to the number of outstanding requests (R) that the memory system can service:

$$BW = \frac{L}{R} \tag{63.4}$$

Bandwidth plays an important role in keeping the processor busy with work. However, technology tradeoffs to optimize latency and improve bandwidth often conflict with the need to increase the capacity and reduce the cost of the memory system.

Cache Memory

Cache memory, or simply cache, is a small, fast memory constructed using semiconductor SRAM. In modern computer systems, there is usually a hierarchy of cache memories. The top-level cache is closest to the processor and the bottom level is closest to the main memory. Each higher level cache is about 5 to 10 times faster than the next level. The purpose of a cache hierarchy is to satisfy most of the processor memory accesses in one or a small number of clock cycles. The top-level cache is often split into an instruction cache and a data cache to allow the processor to perform simultaneous accesses for instructions and data. Cache memories were first used in the IBM mainframe computers in the 1960s. Since 1985, cache memories have become a standard feature for virtually all microprocessors.

Cache memories exploit the principle of locality of reference. This principle dictates that some memory locations are referenced more frequently than others, based on two program properties. *Spatial locality* is the property that an access to a memory location increases the probability that the nearby memory location will also be accessed. Spatial locality is predominantly based on sequential access to program code and structured data. *Temporal locality* is the property that access to a memory location greatly increases the probability that the same location will be accessed in the near future. Together, the two properties ensure that most memory references will be satisfied by the cache memory.

There are several different cache memory designs: direct-mapped, fully associative, and set-associative. Figure 63.6 illustrates the two basic schemes of cache memory, direct-mapped and set-associative. Direct-mapped cache, shown in Fig. 63.6(a) allows each memory block to have one place to reside within a cache. Fully associative cache, shown in Fig. 63.6(b), allows a block to be placed anywhere in the cache. Set associative cache restricts a block to a limited set of places in the cache.

Cache misses are said to occur when the data requested does not reside in any of the possible cache locations. Misses in caches can be classified into three categories: conflict, compulsory, and capacity. Conflict misses are misses that would not occur for fully associative caches with LRU (Least Recently Used) replacement. Compulsory misses are misses required in cache memories for initially referencing a memory location. Capacity misses occur when the cache size is not sufficient to contain data between references. Complete cache miss definitions are provided in Ref. 4.

Unlike memory system properties, the latency in cache memories is not fixed and depends on the delay and frequency of cache misses. A performance metric that accounts for the penalty of cache misses is *effective latency*. Effective latency depends on the two possible latencies, hit latency (L_{HIT}), the latency experienced for accessing data residing in the cache, and miss latency (L_{MISS}), the latency experienced when accessing data not residing in the cache. Effective latency also depends on the *hit rate* (H), the percentage of memory accesses that are hits in the cache, and the *miss rate* (M or $1 - H$), the percentage of memory accesses that miss in the cache. Effective latency in a cache system is calculated as:

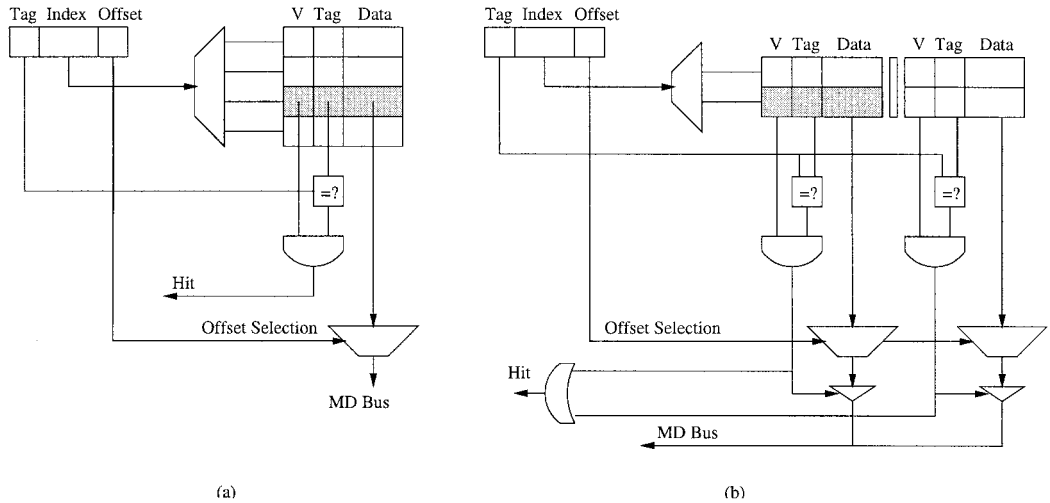


FIGURE 63.6 Cache memory: (a) direct-mapped design, (b) two-way set-associative design.

$$L_{effective} = L_{HIT} * H + L_{MISS} * \left(1 - H\right) \tag{63.5}$$

In addition to the base cache design and size issues, there are several other cache parameters that affect the overall cache performance and miss rate in a system. The main memory update method indicates when the main memory will be updated by store operations. In *write-through* cache, each write is immediately reflected to the main memory. In *write-back* cache, the writes are reflected to the main memory only when the respective cache block is replaced. Cache block allocation is another parameters and designates whether the cache block is allocated on writes or reads. Last, block replacement algorithms for associative structures can be designed in various ways to extract additional cache performance. These include LRU (least recently used), LFU (least frequently used), random, and FIFO (first-in, first-out). These cache management strategies attempt to exploit the properties of locality. Spatial locality is exploited by deciding which memory block is placed in cache, and temporal locality is exploited by deciding which cache block is replaced. Traditionally, when cache service misses, they would *block* all new requests. However, *non-blocking* cache can be designed to service multiple miss requests simultaneously, thus alleviating delay in accessing memory data.

In addition to the multiple levels of cache hierarchy, additional memory buffers can be used to improve cache performance. Two such buffers are a streaming/prefetch buffer and a victim cache.[2] Figure 63.7 illustrates the relation of the streaming buffer and victim cache to the primary cache of a memory system. A streaming buffer is used as a prefetching mechanism for cache misses. When a cache miss occurs, the streaming buffer begins prefetching successive lines starting at the miss target. A victim cache is typically a small, fully associative cache loaded only with cache lines that are removed from the primary cache. In the case of a miss in the primary cache, the victim cache may hold additional data. The use of a victim cache can improve performance by reducing the number of conflict misses. Figure 63.7 illustrates how cache accesses are processed through the streaming buffer into the primary cache on cache requests, and from the primary cache through the victim cache to the secondary level of memory on cache misses.

Overall, cache memory is constructed to hold the most important portions of memory. Techniques using either hardware or software can be used to select which portions of main memory to store in cache. However, cache performance is strongly influenced by program behavior and numerous hardware design alternatives.

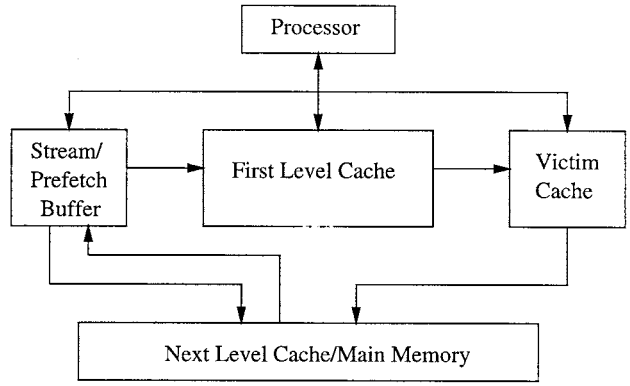


FIGURE 63.7 Advanced cache memory system.

Virtual Memory

Cache memory illustrated the principle that the memory address of data can be separate from a particular storage location. Similar address abstractions exist in the two-level memory hierarchy of main memory and disk storage. An address generated by a program is called a *virtual address*, which needs to be translated into a *physical address* or location in main memory. Virtual memory management is a mechanism which provides the programmers with a simple, uniform method to access both main and secondary memories. With virtual memory management, the programmers are given a virtual space to hold all the instructions and data. The virtual space is organized as a linear array of locations. Each location has an address for convenient access. Instructions and data have to be stored somewhere in the real system; these virtual space locations must correspond to some physical locations in the main and secondary memory. Virtual memory management assigns (or maps) the virtual space locations into the main and secondary memory locations. The mapping of virtual space locations to the main and secondary memory is managed by the virtual memory management. The programmers are not concerned with the mapping.

The most popular memory management scheme today is demand paging virtual memory management, where each virtual space is divided into pages indexed by the page number (PN). Each page consists of several consecutive locations in the virtual space indexed by the page index (PI). The number of locations in each page is an important system design parameter called page size. Page size is usually defined as a power of two so that the virtual space can be divided into an integer number of pages. Pages are the basic unit of virtual memory management. If any location in a page is assigned to the main memory, the other locations in that page are also assigned to the main memory. This reduces the size of the mapping information.

The part of the secondary memory to accommodate pages of the virtual space is called the swap space. Both the main memory and the swap space are divided into page frames. Each page frame can host a page of the virtual space. If a page is mapped into the main memory, it is also hosted by a page frame in the main memory. The mapping record in the virtual memory management keeps track of the association between pages and page frames.

When a virtual space location is requested, the virtual memory management looks up the mapping record. If the mapping record shows that the page containing requested virtual space location is in main memory, the management performs the access without any further complication. Otherwise, a secondary memory access has to be performed. Accessing the secondary memory is usually a complicated task and is usually performed as an operating system service. In order to access a piece of information stored in the secondary memory, an operating system service usually has to be requested to transfer the information into the main memory. This also applies to virtual memory management. When a page is mapped into the secondary memory, the virtual memory management has to request a service in the operating system

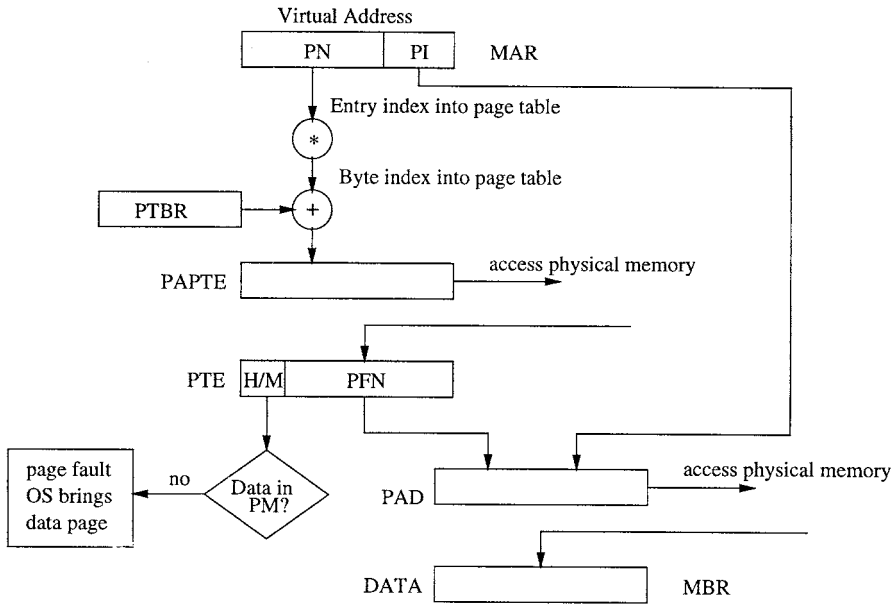


FIGURE 63.8 Virtual memory translation.

to transfer the requested virtual space location into the main memory, update its mapping record, and then perform the access. The operating system service thus performed is called the page fault handler.

The core process of virtual memory management is a memory access algorithm. A one-level virtual address translation algorithm is illustrated in Fig. 63.8. At the start of the translation, the memory access algorithm receives a virtual address in a Memory Address Register (MAR), looks up the mapping record, requests an operating system service to transfer the required page if necessary, and performs the main memory access. The mapping is recorded in a data structure called the Page Table located in main memory at a designated location marked by the Page Table Base Register (PTBR).

The page table index and the PTBR form the physical address (PAPTE) of the respective Page Table Entry. Each PTE keeps track of the mapping of a page in the virtual space. It includes two fields: a hit/miss bit and a page frame number. If the hit/miss (H/M) bit is set (hit), the corresponding page is in main memory. In this case, the page frame hosting the requested page is pointed to by the page frame number (PFN). The final physical address (PAD) of the requested data is then formed using the PFN and PI. The data is returned and placed in the Memory Buffer Register (MBR) and the processor is informed of the completed memory access. Otherwise (miss), a secondary memory access has to be performed. In this case, the page frame number should be ignored. The fault handler has to be invoked to access the secondary memory. The hardware component that performs the address translation algorithm is called the Memory Management Unit (MMU).

The complexity of the algorithm depends on the mapping structure. A very simple mapping structure is used in this section to focus on the basic principles of the memory access algorithms. However, more complex two-level schemes are often used due to the size of the virtual address space. The size of the page table designated may be quite large for a range of main memory sizes. As such, it becomes necessary to map portions of page table into a second page table. In such designs, only the second-level page table is stored in a reserved region of main memory, while the first page table is mapped just like the data in the virtual spaces. There are also requirements for such designs in a multiprogramming system, where there are multiple processes active at the same time. Each processor has its own virtual space and therefore its own page table. As a result, these systems need to keep multiple page tables at the same time. It usually take too much main memory to accommodate all the active page tables. Again, the natural solution to this problem is to provide other levels of mapping.

Translation Lookaside Buffer

Hardware support for a virtual memory system generally includes a mechanism to translate virtual addresses into the real physical addresses used to access main memory. A Translation Lookaside Buffer (TLB) is a cache structure which contains the frequently used page table entries for address translation. With a TLB, address translation can be performed in a single clock cycle when TLB contains the required page table entries (TLB hit). The full address translation algorithm is performed only when the required page table entries are missing from the TLB (TLB miss).

Complexities arise when a system includes both virtual memory management and cache memory. The major issue is whether address translation is done before accessing the cache memory. In *virtual* cache systems, the virtual address directly accesses cache. In a *physical* cache system, the virtual address is translated into a physical address before cache access. Figure 63.9 illustrates both the *virtual* and *physical* cache translation approaches.

A virtual cache system typically overlaps the cache memory access and the access to the TLB. The overlap is possible when the virtual memory page size is larger than the cache capacity divided by the degree of cache associativity. Essentially, since the virtual page index is the same as the physical address index, no translation for the lower indexes of the virtual address is necessary. Thus, the cache can be accessed in parallel with the TLB, or the TLB can be accessed after the cache access for cache misses. Typically, with no TLB logic between the processor and the cache, access to cache can be achieved at lower cost in virtual cache systems and multi-access per cycle cache systems can avoid requiring a multiported TLB. However, the virtual cache translation alternative introduces virtual memory consistency problems. The same virtual address from two different processes mean different physical memory locations. Solutions to this form of aliasing are to attach a process identifier to the virtual address or to flush cache contents on context switches. Another potential alias problem is that different virtual addresses of the same process may be mapped into the same physical address. In general, there is no easy solution; and it involves a reverse translation problem.

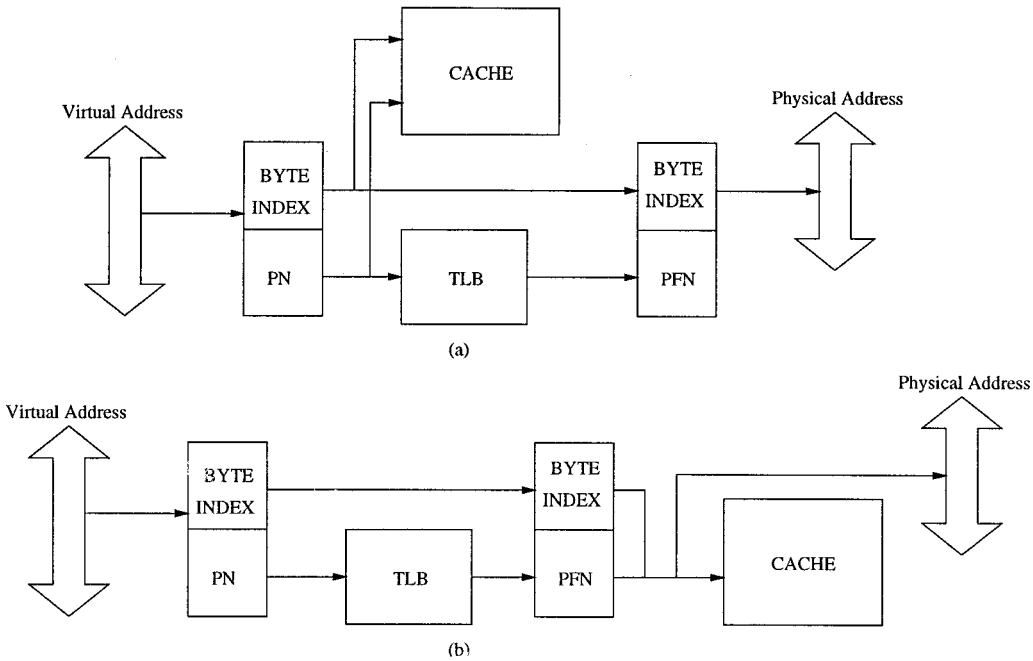


FIGURE 63.9 Translation Lookaside Buffer (TLB) architectures: (a) virtual cache, (b) physical cache.

Physical cache designs are not always limited by the delay of the TLB and cache access. In general, there are two solutions to allow large physical cache design. The first solution, employed by companies with past commitments to page size, is to increase the set associativity of cache. This allows the cache index portion of the address to be used immediately by the cache in parallel with virtual address translation. However, large set associativity is very difficult to implement in a cost-effective manner. The second solution, employed by companies without past commitment, is to use a larger page size. The cache can be accessed in parallel with the TLB access similar to the other solution. In this solution, there are fewer address indexes that are translated through the TLB, potentially reducing the overall delay. With larger page sizes, virtual caches do not have advantage over physical caches in terms of access time.

Input/Output Subsystem

The Input/Output (I/O) subsystem transfers data between the internal components (CPU and main memory) and the external devices (disks, terminals, printers, keyboards, scanners).

Peripheral Controllers

The CPU usually controls the I/O subsystem by reading from and writing into the I/O (control) registers. There are two popular approaches for allowing the CPU to access these I/O registers; I/O instructions and memory-mapped I/O. In an I/O instruction approach, special instructions are added to the instruction set to access I/O status flags, control registers, and data buffer registers. In a memory-mapped I/O approach, the control registers, the status flags, and the data buffer registers are mapped as physical memory locations. Due to the increasing availability of chip area and pins, microprocessors are increasingly including peripheral controllers on-chip. This trend is especially clear for embedded microprocessors.

Direct Memory Access Controller

A DMA controller is a peripheral controller that can directly drive the address lines of the system bus. The data is directly moved from the data buffer to the main memory, rather than from data buffer to a CPU register, then from CPU register to main memory.

System Interconnection

System interconnection is the facilities that allow the components within a computer system to communicate with each other. There are numerous logical organizations of these system interconnect facilities.

Dedicated links or point-to-point connections enable dedicated communication between components. There are different system interconnection configurations based on the connectivity of the system components. A complete connection configuration, requiring $N \cdot (N-1)/2$ links, is created when there is one link between every possible pair of components. A *hypercube* configuration assigns a unique n-tuple {1,0} as the coordinate of each component and constructs a linke between components whose coordinates differ only in one dimension, requiring $N \cdot \log N$ links. A *mesh* connection arranges the system components into an N-dimensional array and has connections between immediate neighbors, requiring $2 \cdot N$ links.

Switching networks are a group of switches that determine the existence of communication links among components. A cross-bar network is considered the most general form of switching network and uses a $N \times M$ two dimensional array of switches to provide an arbitrary connection between N components on one side to M components on another side using $N \cdot M$ switches and $N + M$ links. Another switching network is the multistage network, which employs multiple stages of shuffle networks to provide a permutation connection pattern between N components on each side by using $N \cdot \log N$ switches and $N \cdot \log N$ links.

Shared buses are single links which connect all components to all other components and are the most popular connection structure. The sharing of buses among the components of a system requires several

aspects of bus control. First, there is a distinction between bus masters, the units controlling bus transfers (CPU, DMA, IOP) and bus slaves, the other units (memory, programmed I/O interface).

Bus interfacing and bus addressing are the means to connect and disconnect units on the bus. Bus arbitration is the process of granting the bus resource to one of the requesters. Arbitration typically uses a selection scheme similar to interrupts; however, there are more fixed methods of establishing selection. Fixed-priority arbitration gives every requester a fixed priority, and round-robin ensures every requester the most favorable at one point in time. Bus timing refers to the method of communication among the system units and can be classified as either synchronous or asynchronous. Synchronous bus timing uses a shared clock that defines the time other bus signals change and stabilize. Clock sharing by all units allows the bus to be monitored at agreed time intervals and action taken accordingly. However, the synchronous system bus must operate at the speed of the slowest component. Asynchronous bus timing allows units to use different clocks, but the lack of a shared clock makes it necessary to use extra signals to determine the validity of bus signals.

63.4 Instruction Set Architecture

There are several elements that characterize an instruction set architecture, including word size, instruction encoding, and architecture model.

Word Size

Programs often differ in the size of data they prefer to manipulate. Word processing programs operate on 8-bit or 16-bit data that correspond to characters in text documents. Many applications require 32-bit integer data to avoid frequent overflow in arithmetic calculation. Scientific computation often requires 64-bit floating-point data to achieve desired accuracy. Operating systems and databases may require 64-bit integer data to represent a very large name space with integers. As a result, the processors are usually designed to access multiple-byte data from memory systems. This is a well-known source of complexity in microprocessor design.

The endian convention specifies the numbering of bytes with a memory word. In the little endian convention, the least significant byte in a word is numbered byte 0. The number increases as the positions increase in significance. The DEC VAX and X86 architectures follow the little endian convention. In the big endian convention, the most significant byte in a word is numbered 0. The number decreases as the positions decrease in significance. The IBM 360/370, HP PA-RISC, Sun SPARC, and Motorola 680X0 architectures follow the big endian convention. The difference usually manifests itself when users try to transfer binary files between machines using different endian conventions.

Instruction Encoding

Instruction encoding plays an important role in the code density and performance of microprocessors. Traditionally, the cost of memory capacity was the determining factor in designing either a fixed-length or variable-length instruction set. Fixed-length instruction encoding assigns the same encoding size to all instructions. Fixed-length encoding is generally a characteristic of modern microprocessors and the product of the increasing advancements in memory capacity.

Variable-length instruction set is the term used to describe the style of instruction encoding that uses different instructions lengths according to addressing modes of operands. Common addressing modes included either register or methods of indexing memory. Figure 63.10 illustrates two potential designs found in modern use of decoding variable length instructions. The first alternative, in Fig. 63.10(a) involves an additional instruction decode stage in the original pipeline design. In this model, the first stage is used to determine instruction lengths and steer the instructions to the second stage, where the actual instruction decoding is performed. The second alternative, in Fig. 63.10(b) involves pre-decoding and marking instruction lengths in the instruction cache. This design methodology has been effectively used in decoding X86 variable instructions.[5] The primary advantage of this scheme is the simplification of the number of decode stages in the pipeline design. However, the method requires a larger instruction cache structure for holding the resolved instruction information.

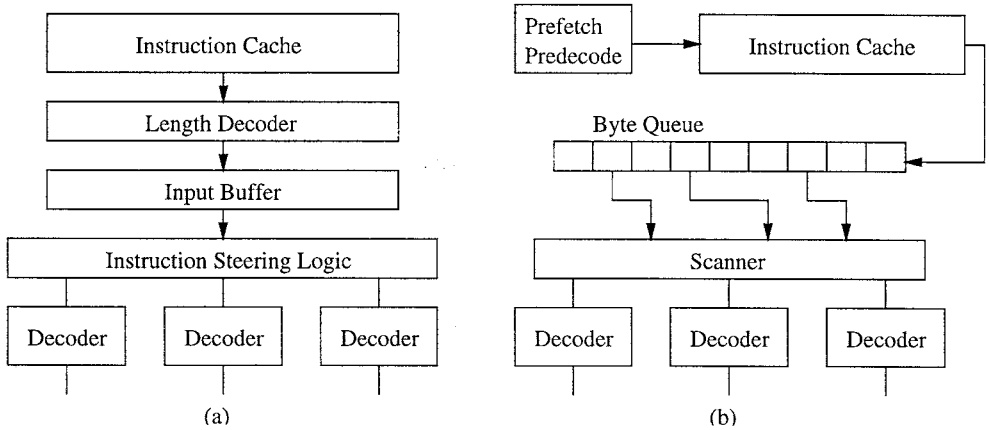


FIGURE 63.10 Variable-sized instruction decoding: (a) staging, (b) predecoding.

Architecture Model

Several instruction set architecture models have existed over the last three decades of computing. First, CISC (Complex Instruction Set Computers) characterized designs with variable instruction formats, numerous memory addressing modes, and large numbers of instruction types. The original CISC philosophy was to create instructions sets that resembled high-level programming languages in an effort to simplify compiler technology. In addition, the design constraint of small memory capacity also led to the development of CISC. The two primary architecture examples of the CISC model are the Digital VAX and Intel X86 architecture families.

RISC (Reduced Instruction Set Computers) gained favor with the philosophy of uniform instruction lengths, load-store instruction sets, limited addressing modes, and reduced number of operation types. RISC concepts allow the microarchitecture design of machines to be more easily pipelined, reducing the processor clock cycle frequency and the overall speed of a machine. The RISC concept resulted from improvements in programming languages, compiler technology, and memory size. The HP PA-RISC, Sun SPARC, IBM Power PC, MIPS, and DEC Alpha machines are examples of RISC architectures.

Architecture models allowing multiple instructions to issue in a clock cycle are VLIW (Very Long Instruction Word). VLIWs issue a fixed number of operations conveyed as a single long instruction and place the responsibility of creating the parallel instruction packet on the compiler. Early VLIW processors suffered from code expansion due to instructions. Examples of VLIW technology are the Multiflow Trace and Cydrome Cydra machines. EPIC (Explicitly Parallel Instruction Computing) is similar in concept to VLIW in that both use the compiler to explicitly group instructions for parallel execution. In fact, many of the ideas for EPIC architectures come from previous RISC and VLIW machines. In general, the EPIC concept solves the excessive code expansion and scalability problems associated with VLIW models by not completely eliminating its functionality. Also, the trend of compiler controlled architecture mechanisms are generally considered part of the EPIC-style architecture domain. The Intel IA-64, Philips Trimedia, and Texas Instruments ‘C6X are examples of EPIC machines.

63.5 Instruction Level Parallelism

Modern processors are being designed with the ability to execute many parallel operations at the instruction level. Such processors are said to exploit ILP (Instruction-Level Parallelism). Exploiting ILP is recognized as a new fundamental architecture concept in improving microprocessor performance, and there are a wide range of architecture techniques that define how an architecture can exploit ILP.

Dynamic Instruction Execution

A major limitation of pipelining techniques is the use of in-order instruction execution. When an instruction in the pipeline stalls, no further instructions are allowed to proceed to insure proper execution of in-flight instruction. This problem is especially serious for multiple issue machines, where each stall cycle potentially costs work of multiple instructions. However, in many cases, an instruction could execute properly if no data dependence exists between the stalled instruction and the instruction waiting to execute. Static scheduling is a compiler-oriented approach for scheduling instructions to separate dependent instructions and minimize the number of hazards and pipeline stalls. Dynamic scheduling is another approach that uses hardware to rearrange the instruction execution to reduce the stalls. The concept of dynamic execution uses hardware to detect dependences in the in-order instruction stream sequence and rearrange the instruction sequence in the presence of detected dependences and stalls.

Today, most modern superscalar microprocessors use dynamic out-of-order scheduling techniques to increase the number of instructions executed per cycle. Such microprocessors use basically the same dynamically scheduled pipeline concept, all instructions pass through an issue stage in-order, are executed out-of-order, and are retired in-order. There are several functional elements of this common sequence which have developed into computer architecture concepts. The first functional concept is *scoreboarding*. Scoreboarding is a technique for allowing instructions to execute out-of-order when there are available resources and no data dependences. Scoreboarding originates from the CDC 6600 machine's issue logic, named the scoreboard. The overall goal of scoreboarding is to execute every instruction as early as possible.

A more advanced approach to dynamic execution is *Tomasulo's approach*. This scheme was employed in the IBM 360/91 processor. Although there are many variations on this scheme, the key concept of avoiding Write-After-Read (WAR) and Write-After-Write (WAW) dependences during dynamic execution is attributed to Tomasulo. In Tomasulo's scheme, the functionality of the scoreboarding is provided by the *reservation stations*. Reservation stations buffer the operands of instructions waiting to issue as soon as they become available. The concept is to issue new instructions immediately when all source operands become available instead of accessing such operands through the register file. As such, waiting instructions designate the reservation station entry that will provide their input operands. This action removes WAW dependences caused by successive writes to the same register by forcing instructions to be related by dependences instead of by register specifiers. In general, renaming of register specifiers for pending operands to the reservation station entries is called *register renaming*. Overall, Tomasulo's scheme combines scoreboarding and register renaming. *An Efficient Algorithm for Exploring Multiple Arithmetic Units*[6] provides the complete details of Tomasulo's scheme.

Predicated Execution

Branch instructions are recognized as a major impediment to exploiting (ILP). Branches force the compiler and hardware to make frequent predictions of branch directions in an attempt to find sufficient parallelism. Misprediction of these branches can result in severe performance degradation through the introduction of wasted cycles into the instruction stream. Branch prediction strategies reduce this problem by allowing the compiler and hardware to continue processing instructions along the predicted control path, thus eliminating these wasted cycles.

Predicated execution support provides an effective means to eliminate branches from an instruction stream. Predicated execution refers to the conditional execution of an instruction based on the value of a Boolean source operand, referred to as the predicate of the instruction. This architectural support allows the compiler to use an *if-conversion* algorithm to convert conditional branches into predicate defining instructions, and instructions along alternative paths of each branch into predicated instructions.[7] Predicated instructions are fetched regardless of their predicate value. Instructions whose predicate value is true are executed normally. Conversely, instructions whose predicate is false are nullified, and thus are prevented from modifying the processor state. Predicated execution allows the compiler to trade instruction fetch efficiency for the capability to expose ILP to the hardware along multiple execution paths.

Predicated execution offers the opportunity to improve branch handling in microprocessors. Eliminating frequently mispredicted branches may lead to a substantial reduction in branch prediction misses. As a result, the performance penalties associated with the eliminated branches are removed. Eliminating branches also reduces the need to handle multiple branches per cycle for wide issue processors. Finally, predicated execution provides an efficient interface for the compiler to expose multiple execution paths to the hardware. Without compiler support, the cost of maintaining multiple execution paths in hardware grows rapidly.

The essence of predicated execution is the ability to suppress the modification of the processor state based upon some execution condition. Full predication cleanly supports this through a combination of instruction set and microarchitecture extensions. These extensions can be classified as a support for suppression of execution and expression of condition. The result of the condition which determines if an instruction should modify state is stored in a set of 1-bit registers. These registers are collectively referred to as the predicate register file. The values in the predicate register file are associated with each instruction in the extended instruction set through the use of an additional source operand. This operand specifies which predicate register will determine whether the operation should modify processor state. If the value in the specified register is 1, or true, the instruction is executed normally; if the value is 0, or false, the instruction is suppressed.

Predicate register values may be set using predicate define instructions. The predicate define semantics used are those of the HPL Playdoh architecture.[8] There is a predicate define instruction for each comparison opcode in the original instruction set. The major difference with conventional comparison instructions is that these predicate defines have up to two destination registers and that their destination registers are predicate registers. The instruction format of a predicate define is shown below.

$$\text{pred\_<}cmp\text{>}\ \text{Pout1}_{<type>},\ \text{Pout2}_{<type>},\ \text{src1},\ \text{src2}\ (\text{P}_{in})$$

This instruction assigns values to *Pout1* and *Pout2* according to a comparison of *src1* and *src2* specified by <*cmp*>. The comparison <*cmp*> can be: equal (eq), not equal (ne), greater than (gt), etc. A predicate <*type*> is specified for each destination predicate. Predicate defining instructions are also predicated, as specified by P_{in}.

The predicate <*type*> determines the value written to the destination predicate register based upon the result of the comparison and of the input predicate, P_{in}. For each combination of comparison result and P_{in}, one of three actions may be performed on the destination predicate: it can write 1, write 0, or leave it unchanged. There are six predicate types which are particularly useful, the unconditional (*U*), *OR*, and *AND* type predicates and their complements. Table 63.1 contains the truth table for these predicate definition types.

TABLE 63.1 Predicate Definition Truth Table

		P_{out}					
P_{in}	Comparison	U	$\overline{U}$	OR	$\overline{OR}$	AND	$\overline{AND}$
0	0	0	0	—	—	—	—
0	1	0	0	—	—	—	—
1	0	0	1	—	1	0	—
1	1	1	0	1	—	—	0

Unconditional destination predicate registers are always defined, regardless of the value of P_{in} and the result of the comparison. If the value of P_{in} is 1, the result of the comparison is placed in the predicate register (or its compliment for $\overline{U}$). Otherwise, a 0 is written to the predicate register. Unconditional predicates are utilized for blocks which are executed based on a single condition.

The *OR*-type predicates are useful when execution of a block can be enabled by multiple conditions, such as logical AND (&&) and OR (||) constructs in C. *OR*-type destination predicate registers are set if P_{in} is 1 and the result of the comparison is 1 (0 for $\overline{OR}$); otherwise, the destination predicate register is

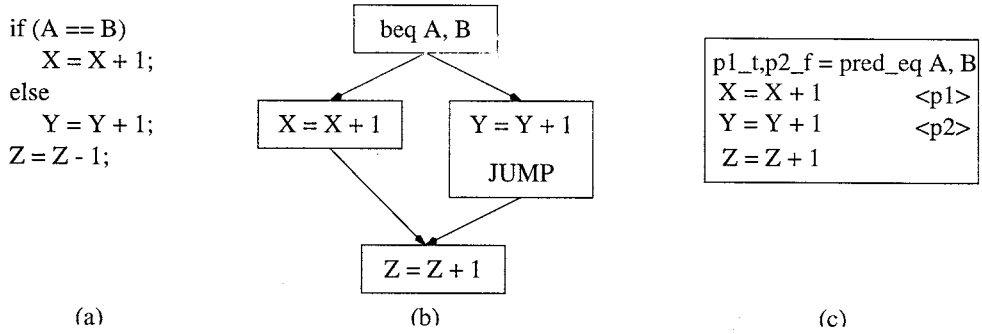


FIGURE 63.11 Instruction sequence: (a) program code, (b) traditional execution, (c) predicated execution.

unchanged. Note that *OR*-type predicates must be explicitly initialized to 0 before they are defined and used. However, after they are initialized, multiple *OR*-type predicate defines may be issued simultaneously and in any order on the same predicate register. This is true since the *OR*-type predicate either writes a 1 or leaves the register unchanged, which allows implementation as a wired logical *OR* condition. *AND*-type predicates are analogous to the *OR* type predicate. *AND*-type destination predicate registers are cleared if P_{in} is 1 and the result of the comparison is 0 (1 for *AND*); otherwise, the destination predicate register is unchanged.

Figure 63.11 contains a simple example illustrating the concept of predicated execution. Figure 63.11(a) shows a common programming if-then-else construction. The related control flow representation of that programming code is illustrated in Fig. 63.11(b). Using if-conversion, the code in Fig. 63.11(b) is then transformed into the code shown in Fig. 63.11(c). The original conditional branch is translated into a *pred_eq* instructions. Predicate register *p1* is set to indicate if the condition ($A = B$) is true, and *p2* is set if the condition is false. The "then" part of the if-statement is predicated on *p1* and the "else" part is predicated on p_2 The *pred_eq* simply decides whether the addition or subtraction instruction is performed and ensures that one of the two parts is not executed. There are several performance benefits for the predicated code. First, the microprocessor does not need to make any branch predictions since all the branches in the code are eliminated. This removes related penalties due to misprediction branches. More importantly, the predicated instructions can utilize multiple instruction execution capabilities of modern microprocessors and avoid the penalties for mispredicting branches.

Speculative Execution

The amount of ILP available within basic blocks is extremely limited in non-numeric programs. As such, processors must optimize and schedule instructions across basic block code boundaries to achieve higher performance. In addition, future processors must content with both long latency load operations and long latency cache misses. When load data is needed by subsequent dependent instructions, the processor execution must wait until the cache access is complete.

In these situations, out-of-order machines dynamically reorder the instruction stream to execute non-dependent instructions. Additionally, out-of-order machines have the advantage of executing instructions that follow correctly predicted branch instructions. However, this approach requires complex circuitry at the cost of chip die space. Similar performance gains can be achieved using static compile-time speculation methods without complex out-of-order logic. Speculative execution, a technique for executing an instruction before knowing its execution is required, is an important technique for exploiting ILP in programs. Speculative execution is best known for hiding memory latency. These methods utilize instruction set architecture support of special speculative instructions.

A compiler utilizes speculative code motion to achieve higher performance in several ways. First, in regions of code where insufficient ILP exists to fully utilize the processor resources, useful instructions

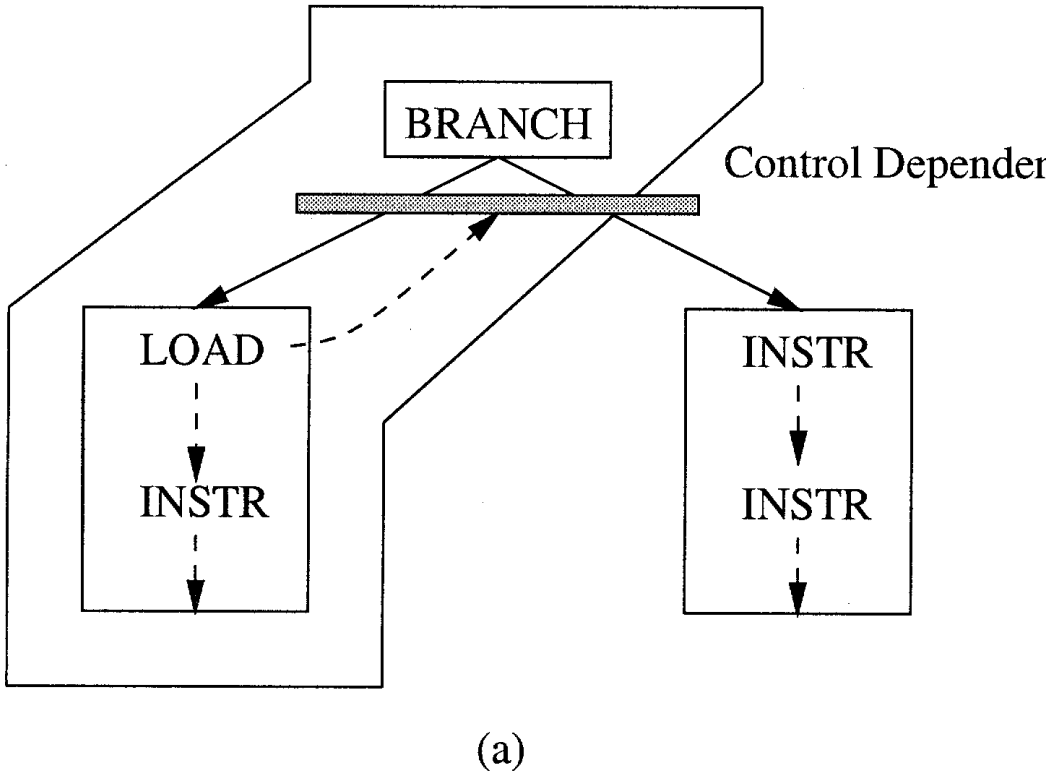


FIGURE 63.12 Instruction sequence: (a) traditional execution, (b) speculative execution.

may be executed. Second, instructions at the beginning of long dependence chains may be executed early to reduce the computation's critical path. Finally, long latency instructions may be initiated early to overlap their execution with other useful operations. Figure 63.12 illustrates a simple example of code before and after a speculative compile-time transformation is performed to execute a load instruction above a conditional branch.

Figure 63.12(a) shows how the branch instruction and its implied control flow define a control dependence that restricts the load operation from being scheduled earlier in the code. Cache miss latencies would halt the processor unless out-of-order execution mechanisms were used. However, with speculation support, Fig. 63.12(b) can be used to hide the latency of the load operation.

The solution requires the load to be speculative or nonfaulting. A speculative load will not signal an exception for faults such as address alignment or address space access errors. Essentially, the load is considered silent for these occurrences. The additional check instruction in Fig. 63.12(b) enables these signals to be detected when the original execution does reach the original location of the load. When the other path of branch's execution is taken, such silent signals are meaningless and can be ignored. Using this mechanism, the load can be placed above all existing control dependences, providing the compiler with the ability to hide load latency. Details of compiler speculation can be found in Ref. 9.

63.6 Industry Trends

The microprocessor industry is one of the fastest moving industries today. Healthy demands from the market-place have stimulated strong competition, which in turn resulted in great technical innovations.

Computer Microprocessor Trends

The current trends of computer microprocessors include deep pipelining, high clock frequency, wide instruction issue, speculative and out-of-order execution, predicated execution, natural data types, large on-chip caches, floating point capabilities, and multiprocessor support. In the area of pipelining, the Intel Pentium II processor is pipelined approximated twice as deeply as its predecessor Pentium. The deep pipeline has allowed the clock Pentium II processor to run at a much higher clock frequency than Pentium.

In the area of wide instruction issue, the Pentium II processor can decode and issue up to three X86 instructions per clock cycle, compared to the two-instruction issue bandwidth of Pentium. Pentium II has dedicated a very significant amount of chip area to Branch Target Buffer, Reservation Station, and Reorder Buffer to support speculative and out-of-order execution. These structures together allow the Pentium II processor to perform much more aggressive speculative and out-of-order execution than Pentium. In particular, Pentium II can coordinate the execution of up to 40 X86 instructions, which is several times larger than Pentium.

In the area of predicated execution, Pentium II supports a conditional move instruction that was not available in Pentium. This trend is furthered by the next generation IA-64 architecture where all instructions can be conditionally executed under the control of predicate registers. This ability will allow future microprocessors to execute control intensive programs much faster than their predecessors.

In the area of data types, the MMX instructions from Intel have become a standard feature of all X86 microprocessors today. These instructions take advantage of the fact that multimedia data items are typically represented with a smaller number of bits (8 to 16 bits) than the width of an integer data path today (32 to 64 bits). Based on an observation, the same operation is often repeated on all data items in multimedia applications, the architects of MMX specify that each MMX instruction performs the same operation on several multimedia data items packed into one integer word. This allows each MMX instruction to process several data items simultaneously to achieve significant speed-up in targeted applications. In 1998, AMD proposed the 3DNow! instructions to address the performance needs of 3-D graphics applications. The 3DNow! instructions are designed based on the concept that 3-D graphics data items are often represented in single precision floating-point format and they do not required the sophisticated rounding and exception handling capabilities specified in the IEEE Standard format. Thus, one can pack two graphics floating-point data into one double-precision floating-point register for more efficient floating-point processing of graphics applications. Note that MMX and 3DNow! are similar in concepts applied to integer and floating-point domains.

In the area of large on-chip caches, the popular strategies used in computer microprocessors are either to enlarge the first-level caches or to incorporate second-level and sometimes third-level caches on-chip. For example, the AMD K7 microprocessor has a 64KB first-level instruction cache and a 64-KB first-level data cache. These first-level caches are significantly larger than those found in the previous generations. For another example, the Intel Celeron microprocessor has a 128-KB second level combined instruction and data cache. These large caches are enabled by the increased chip density that allows many more transistors on the chip. The Compaq Alpha 21364 microprocessor has both: a 64-KB first-level instruction cache, a 64-KB first-level data cache, and a 1.5-MB second-level combined cache.

In the area of floating-point capabilities, computer microprocessors in general have much stronger floating-point performance than their predecessors. For example, the Intel Pentium II processor achieves several times the floating-point performance improvements of the Pentium processor. For another example, most RISC microprocessors now have floating-point performances that rival supercomputer CPUs built just a few years ago.

Due to the increasing demand of multiprocessor enterprise computing servers, many computer microprocessors now seamlessly support cache coherence protocols. For example, the AMD K7 microprocessor provides direct support for seamless multiprocessor operation when multiple K7 microprocessors are connected to a system bus. This capability was not available in its predecessor, the AMD K6.

Embedded Microprocessor Trends

There are three clear trends in embedded microprocessors. The first trend is to integrate a DSP core with an embedded CPU/controller core. Embedded applications increasingly require DSP functionalities such as data encoding in disk drives and signal equalization for wireless communications. These functionalities enhance the quality of services of their end computer products. At the *1998 Embedded Microprocessor Forum*, ARM, Hitachi, and Siemens all announced products with both DSP and embedded microprocessors.[10]

Three approaches exist in the integration of DSP and embedded CPUs. One approach is to simply have two separate units placed on a single chip. The advantage of this approach is that it simplifies the development of the microprocessor. The two units are usually taken from existing designs. The software development tools can be directly taken from each unit's respective software support environments. The disadvantage is that the application developer needs to deal with two independent hardware units and two software development environments. This usually complicates software development and verification.

An alternative approach to integrating DSP and embedded CPUs is to add the DSP as a co-processor of the CPU. This CPU fetches all instructions and forwards the DSP instructions to the co-processor. The hardware design is more complicated than the first approach due to the need to more closely interface the two units, especially in the area of memory accesses. The software development environment also needs to be modified to support the co-processor interaction model. The advantage is that the software developers now deal with a much more coherent environment.

The third approach to integrating DSP and embedded CPUs is to add DSP instructions to a CPU instruction set architecture. This usually requires brand-new designs to implement the fully integrated instruction set architecture.

The second trend in embedded microprocessors is to support the development of single-chip solutions for large-volume markets. Many embedded microprocessor vendors offer designs that can be licensed and incorporated into a larger chip design that includes the desired input/output peripheral devices and Application-Specific Integrated Circuit (ASIC) design. This paradigm is referred to as system-on-a-chip design. A microprocessor that is designed to function in such a system is often referred to as a licensable core.

The third major trend in embedded microprocessors is aggressive adoption of high-performance techniques. Traditionally, embedded microprocessors are slow to adopt high-performance architecture and implementation techniques. They also tend to reuse software development tools, such as compilers from the computer microprocessor domain. However, due to the rapid increase of required performance in embedded markets, the embedded microprocessor vendors are now making fast moves in adopting high-performance techniques. This trend is especially clear in the DSP microprocessors. Texas Instruments, Motorola/Lucent, and Analog Devices have all announced aggressive EPIC DSP microprocessors to be shipped before the Intel/HP IA-64 EPIC microprocessors.

Microprocessor Market Trends

Readers who are interested in market trends for microprocessors are referred to *Microprocessor Report*, a periodical publication by MicroDesign Resources (www.MDRonline.com). In every issue, there is a summary of microarchitecture features, physical characteristics, availability, and pricing of microprocessors.

References

1. J. Turley, RISC volume gains but 68K still reigns, *Microprocessor Report*, vol. 12, pp. 14-18, Jan. 1998.
2. J.L. Hennessy and D.A. Patterson, *Computer Architecture A Quantitative Approach*, Morgan Kaufman, San Francisco, CA, 1990.
3. J.E. Smith, A study of branch prediction strategies, *Proceedings of the 8th International Symposium on Computer Architecture*, pp. 135-14, May 1981.

4. W.W. Hwu and T.M. Conte, The susceptibility of programs to context switching, *IEEE Transactions on Computers*, vol. C-43, pp. 993-1003, Sept. 1994.
5. L. Gwennap, Klamath extends P6 family, *Microprocessor Report*, Vol. 1, pp. 1-9, February 1997.
6. R.M. Tomasulo, An efficient algorithm for exploiting multiple arithmetic units, *IBM Journal of Research and Development*, vol. 11, pp. 25-33, Jan. 1967.
7. J.R. Allen et al. Conversion of control dependence to data dependence, *Proceedings of the 10th ACM Symposium on Principles of Programming Languages*, pp. 177-189, Jan. 1983.
8. V. Kathail, M.S. Schlansker, and B.R. Rau, HPL PlayDoh architecture specification: Version 1.0, Tech. Rep. HPL-93-80, Hewlett-Packard Laboratories, Palo Alto, CA, Feb. 1994.
9. S.A. Mahlke et al. Sentinel scheduling: A model for compiler-controlled speculative execution, *ACM Transactions on Computer Systems*, vol. 11, Nov. 1993.
10. *Embedded Microprocessor Forum* (San Jose, CA), Oct. 1998.

Gupta, S., Gupta, R.K. "ASIC Design"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

64

ASIC Design

Sumit Gupta
Rajesh K. Gupta
University of California at Irvine

64.1 Introduction

Microelectronic technology has matured considerably in the past few decades. Systems which until the start of the decade required a printed circuit board for implementation are now being developed on a single chip. These systems-on-a-chip (SOCs) are becoming a reality due to vast improvements in chip fabrication and process technology. A key component in SOC and other semiconductor chips are *Application-Specific Integrated Circuits* (ASICs). These are specialized circuit blocks or entire chips which are designed specifically for a given application or an application domain. For instance, a video decoder circuit may be implemented as an ASIC chip to be used inside a personal computer product or in a range of multimedia appliances. Due to the custom nature of these designs, it is often possible to squeeze in more functionality under performance requirements — while reducing system size, power, heat, and cost — than possible with standard IC parts. Due to cost and performance advantages, ASICs and semiconductor chips with ASIC blocks are used in a wide range of products, from consumer electronics to space applications.

Traditionally, the design of ASICs has been a long and tedious process because of the different steps in the design process. It has also been an expensive process due to the costs associated with ASIC manufacturing for all but applications requiring more than tens of thousands of IC parts. Lately, the

situation has been changing in favor of increased use of ASIC parts, in part helped by robust design methodologies and increased use of automated circuit synthesis tools. These tools allow designers to go from high-level design descriptions, all the way to final chip layouts and mask generation for the fabrication process. These developments, coupled with an increasing market for semiconductor chips in nearly all every-day devices, have led to a spur in the demand for ASICs and chips which have ASICs in them.

ASIC design and manufacturing span a broad range of activities, which includes product conceptualization, design and synthesis, verification, and testing. Once the product requirements have been finalized, a high-level design is done from which the circuit is synthesized or successively refined to the lowest level of detail. The design has to be verified for functionality and correctness at each stage of the process to ensure that no errors are introduced and the product requirements are met. Testing here refers to manufacturing test, which involves determining if the chip has no manufacturing defects. This is a challenging problem since it is difficult to control and observe internal wires in a manufactured chip and it is virtually impossible to repair the manufactured chips. At the same time, volume manufacturing of semiconductors requires that the product be tested in a very short time (usually less than a second). Hence, we need to develop a test methodology which allows us to check if a given chip is functional in the shortest possible amount of time. In this chapter, we focus on ASIC design issues and their relationship to other ASIC aspects, such as testability, power optimization, etc. We concentrate on the design flow, methodology, synthesis, and physical issues, and relate these to the computer-aided design (CAD) tools available.

The rest of this chapter is organized in the following manner. Section 64.2 introduces the notion of a design style and the ASIC design methodologies. Section 64.3 outlines the steps in the design process followed by a discussion of the role of hierarchy and design abstractions in the ASIC design process. Following sections on architectural design, logic synthesis, and physical design give examples to demonstrate the key ideas. We elucidate the availability and the use of appropriate CAD tools at various steps of the ASIC design.

64.2 Design Styles

ASIC design starts with an initial concept of the required IC part. Early in this product conceptualization phase, it is important to decide the *design style* that will be most suitable for the design and validation of the eventual ASIC chip. A design style refers to a broad method of designing circuits which uses specific techniques and technologies for the design implementation and validation. In particular, a design style determines the specific design steps and the use of library parts for the ASIC part. Design styles are determined, in part, by the economic viability of the design, as determined by tradeoffs between performance, pricing, and production volume. For some applications, such as defense systems and space applications, although the volume is low, the cost is of little concern due to the time-criticality of the application and the requirements of high performance and reliability. For applications such as consumer electronics, the high volume can offset high production costs.

Design styles are broadly classified into *custom* and *semi-custom* designs.[1] Custom designs, as the name suggests, involve the complete design to be hand-crafted so as to optimize the circuit for performance and/or area for a given application. Although this is an expensive design style in terms of effort and cost, it leads to high-quality circuits for which the cost can be amortized over a large volume production.

The semi-custom design style limits the circuit primitives and uses predesigned blocks which cannot be further fine-tuned. These predesigned primitive blocks are usually optimized, well-designed, and well-characterized, and ultimately help raise the level of abstraction in the design. This design style leads to reduced design times and facilitates easier development of CAD tools for design and optimization. These CAD tools allow the designer to choose among the various available primitive blocks and interconnect them to achieve the design functionality and performance. Semi-custom design styles are becoming the norm due to increasing design complexity. At the current level of circuit complexity, the loss in quality by using a semi-custom design style is often very small compared to a custom design style.

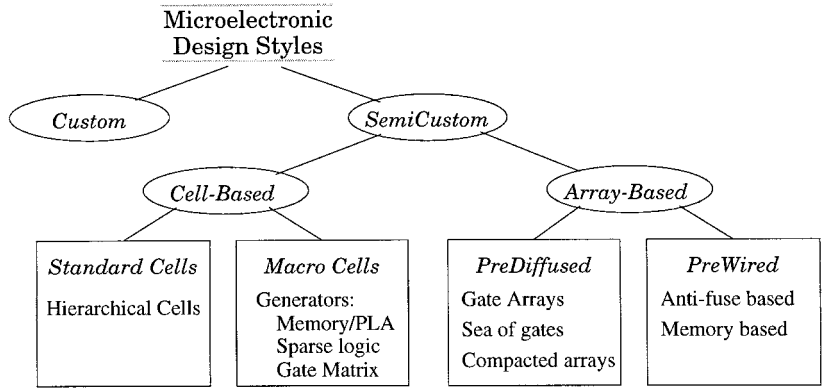


FIGURE 64.1 Classification of custom and semi-custom design styles.

Semi-custom designs can be classified into two major classes: *cell-based design* and *array-based design*, which can further be further subdivided into subclasses as shown in Fig. 64.1.[1] Cell-based designs use *libraries* of predesigned cells or *cell generators*, which can synthesize cell layouts given their functional description. The predesigned cells can be characterized and optimized for the various process technologies that the library targets.

Cell-based designs can be based on *standard-cell* design, in which basic primitive cells are designed once and, thereafter, are available in a library for each process technology or foundry used. Each cell in the library is parameterized in terms of area, delay, and power. These libraries have to be updated whenever the foundry technology changes. CAD tools can then be used to map the design to the cells available in the library in a step known as *technology mapping* or *library binding*. Once the cells are selected, they are placed and wired together.

Another cell-based design style uses *cell generators* to synthesize primitive building blocks which can be used for *macro-cell*-based design (see Fig. 64.1). These generators have traditionally been used for the automatic synthesis of memories and programmable logic arrays (PLAs), although recently module generators have been used to generated complex datapath components such as multipliers.[2] Module generators for macro-cell generation are *parameterizable*, that is, they can be used to generate different instances of a module such as a 8×8 and a 16×8 multiplier.

In contrast to cell-based designs, *array-based* designs use a prefabricated matrix of non-connected components known as *sites*. These sites are wired together to create the circuit required. Array-based circuits can either be *pre-diffused* or *pre-wired*, also known as *mask programmable* and *field programmable gate arrays*, respectively (MPGAs and FPGAs). In MPGAs, wafers consisting of arrays of unwired sites are manufactured and then the sites are programmed by connecting them with wires, via different routing layers during the chip fabrication process. There are several types of these pre-diffused arrays, such as gate arrays, sea-of-gates, and compacted arrays (see Fig. 64.1).

Unlike MPGAs, pre-wired gate arrays or FPGAs are programmed outside the semiconductor foundry. FPGAs consist of programmable arrays of modules implementing generic logic. In the *anti-fuse* type of FPGAs, wires can be connected by programming the anti-fuses in the array. Anti-fuses are open-circuit devices that become a short-circuit when an appropriate current is applied to them. In this way, the circuit design required can be achieved by connecting the logic module inputs appropriately by programming the anti-fuses. On the other hand, *memory-based* FPGAs store the information about the interconnection and configuration of the various generic logic modules in memory elements inside the array.

The use of FPGAs is becoming more and more popular as the capacity of the arrays and their performance are improving. At present, they are used extensively for circuit prototyping and verification. Their relative ease of design and customization leads to low cost and time overheads. However, FPGA is still an expensive technology since the number of gate arrays required to implement a moderately complex

design is large. The cost per gate of prototype design is decreasing due to continuous density and capacity improvement in FPGA technology.

Hence, there are several design styles available to a designer, and choosing among them depends upon tradeoffs using factors such as cost, time-to-market, performance, and reliability. In real-life applications, nearly all designs are a mix of custom and semi-custom design styles, particularly cell-based styles. Depending on the application, designers adopt an approach of embedding some custom designed blocks inside a semi-custom design. This leads to lower overheads since only the critical parts of the design have to be hand-crafted. For example, a microprocessor typically has a custom designed data path and the control logic is synthesized using a standard cell-based technique. Given the complexity of microprocessors, recent efforts in CAD are attempting to automate the design process of data path blocks as well.[3] Prototyping and circuit verification using FPGA-based technologies has become popular due to high costs and time overruns in case of a faulty design once the chip is manufactured.

64.3 Steps in the Design Flow

An important decision for any design team is the design flow that they will adopt. The design flow defines the approach used to take a design from an abstract concept through the specification, design, test, and manufacturing steps.[28] The *waterfall model* has been the traditional model for ASIC development. In this model, the design goes through various steps or phases while it is constantly refined to the highest level of detail. This model involves minimal interaction between design teams working on different phases of the design.

The design process starts with the development of a *specification* and high-level design of the ASIC, which may include requirements analysis, architecture design, executable specification or C model development, and functional verification of the specification. The design is then coded at the register transfer level (RTL) in hardware description languages such as VHDL[12] or Verilog.[13] The functionality of the RTL code is verified against the initial specification (e.g., C model), which is used as the *golden model* for verifying the design at every level of abstraction (see Section 64.5). The RTL is then synthesized into a *gate-level netlist* which is run through a *timing verification* tool which verifies that the ASIC meets the timing constraints specified. The physical design team subsequently develops a floorplan for the chip, places the cells, and routes the interconnects, after which the chip is manufactured and tested (see Fig. 64.2).

The disadvantage with this design methodology is that as the complexity of the system being designed increases, the design becomes more error prone. The requirements are not properly tested until a working system model is available, which only becomes available late in the design cycle. Errors are hence discovered late in the design process and error correction often involves a major redesign and rerun through the steps of the design again. This leads to several design reworks and may even involve multiple chip fabrication runs.

The steps and different levels of detail that the design of an integrated circuit goes through as it progresses from concept to chip fabrication, are shown in Fig. 64.2. The requirements of a design are represented by a behavioral model which represents

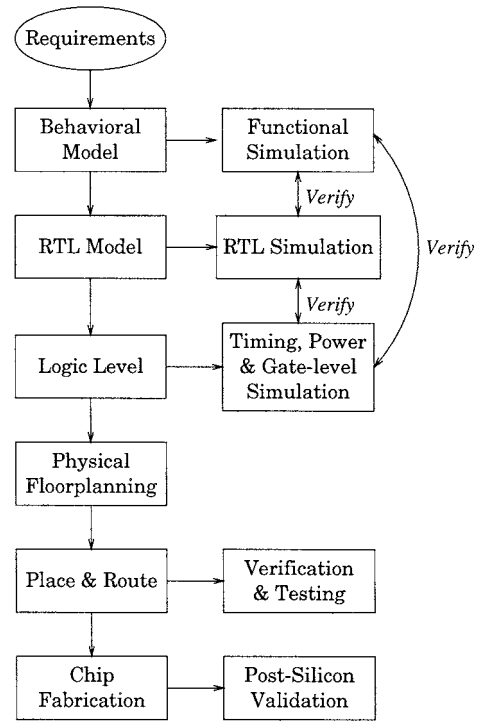


FIGURE 64.2 A typical ASIC design flow.

the functions the design must implement with the timing, area, power, testing, etc. constraints. This behavioral model is usually captured in the form of an executable functional specification in a language such as C (or C++). This functional specification is simulated for a wide set of inputs to verify that all the requirements and functionalities are met.

For instance, when developing a new microprocessor, after the initial architectural design, the design team develops an instruction set architecture. This involves making decisions on issues such as the number of pipeline stages, width of the data path, size of the register file, number and type of components in the data path, etc. An instruction set simulator is then developed so that the range of applications being targeted (or a representative set) can be simulated on the processor simulator. This verifies that the processor can run the application or a benchmark suite within the required timing performance. The simulator also verifies that the high-level design is correct and attempts to identify data and pipeline hazards in the data path architecture. The feedback from the simulator may be used to refine the instruction set of the processor.

The functional specification (or behavioral model) is converted into a register transfer level (RTL) model, either manually or by using a behavioral or high-level synthesis tool.[27] This RTL model uses register-level components like adders, multipliers, registers, multiplexors, etc. to represent the structural model of the design with the components and their interconnections. This RTL model is simulated, typically using event-driven simulation (see Section 64.7) to verify the functionality and coarse-level timing performance of the model. The tested and verified software functional model is used as the *golden model* to compare the results against. The RTL model is then refined to the logic gate level using logic synthesis tools which implement the components with gates or combination of gates, usually using a cell-library-based methodology. The gate-level netlist undergoes the most extensive simulation. Besides functionality, other constraints such as timing and power are also analyzed. Static timing analysis tools are used to analyze the timing performance of the circuit and identify critical paths in the design. The gate-level netlist is then converted into a physical layout, by floorplanning the chip area, placement of the cells, and routing of the interconnects. The layout is used to generate the set of masks[1] required for chip fabrication.

Logic synthesis is a design methodology for the synthesis and optimization of gate-level logic circuits. Before the advent of logic synthesis, ASIC designers used a *capture-and-simulate* design methodology.[4] In this methodology, a team of design architects starts with the requirements for the product and produces a rough block diagram of the chip architecture. This architecture is then refined to ensure completeness and functionality and then given to a team of logic and layout designers who use logic and circuit schematic design tools to capture the design and each of its functional blocks and their interconnections. Layout, placement, and routing tools are then used to map this schematic into the technology library or to another custom or semi-custom design style.

However, the development of logic synthesis in the last decade has raised the ante to a *describe-and-synthesize* methodology. Designs are specified in hardware description languages (HDL) such as *VHDL*[12] and *Verilog*,[13] using Boolean equations and finite-state machine descriptions or diagrams, in a technology-independent form. *Logic synthesis* tools are then used to synthesize these Boolean equations and finite-state machine descriptions into functional units and control units, respectively.[5,6,23]

Behavioral or *high-level synthesis* tools work at a higher level of abstraction and use programs, algorithms, and dataflow graphs as inputs to describe the behavior of the system and synthesize the processors, memories, and ASICs from them.[24,27] They assist in making decisions that have been the domain of chip architects and have been based mostly on experience and engineering intuition.

The relationship of the ASIC design flow, synthesis methodologies, and CAD tools is shown in Fig. 64.3. This figure shows how the design can go from behavior to register to gate to mask level via several paths which may be manual or automated or may involve sourcing out to another vendor. Hence, at any stage of the design, the design refinement step can either be performed manually or with the help of a synthesis

[1]Masks are the geometric patterns used to etch the cells and interconnects onto the silicon wafer to fabricate the chip.

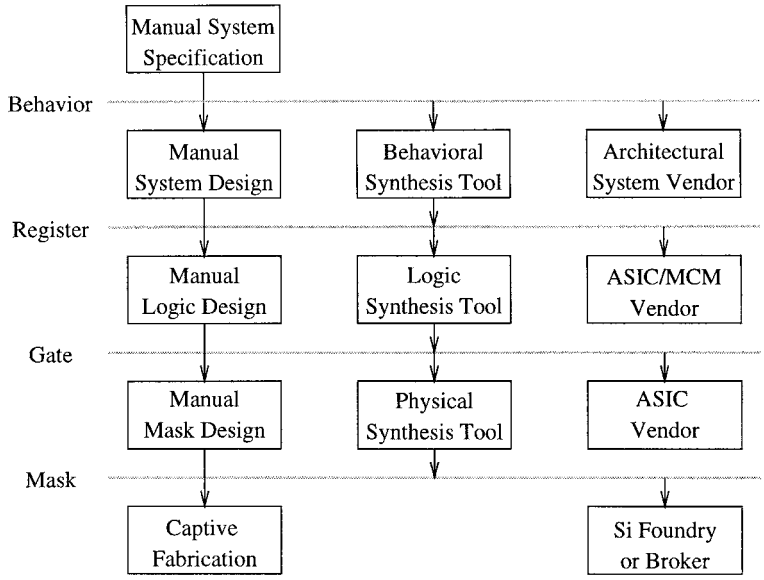


FIGURE 64.3 Manual design, automated synthesis, and outsourcing.

CAD tool or the design at that stage can be sent to a vendor who refines the current design to the final fabrication stage. This concept has been popular among *fab-less* design companies that use technology libraries from foundries for logic synthesis and send out the logic gate netlist design for final mask generation and manufacturing to the foundries. However, in more recent years, vendors are specializing in design of reusable blocks which are sold as *intellectual property* (IP) to other design houses, who then assemble these blocks together to create systems-on-a-chip.[28]

Frequently, large semiconductor design houses are structured around groups which specialize in each one of these stages of the design. Hence, they can be thought of as independent vendors: the architectural design team defines the blocks in the design and their functionality, the logic design team refines the system design into a logic level design for which the masks are then generated by the physical design team. These masks are used for chip fabrication by the foundry. In this way, the design style becomes modular and easier to manage.

64.4 Hierarchical Design

Hierarchical decomposition of a complex system into simpler subsystems and further decomposition into subsystems of ever-more simplicity is a long established design technique. This *divide-and-conquer* approach attempts to handle the problem's complexity by recursively breaking it down into manageable pieces which can be easily implemented.

Chip designers extend the same hierarchical design technique by structuring the chip into a hierarchy of components and subcomponents. An example of hierarchical digital design is shown in Fig. 64.4.[25] This figure shows how a 4-bit adder can be created using four single-bit *full adders* (FAs) which are designed using logic gates such as AND, OR, and XOR gates. The FAs are composed into the 4-bit adder by interconnecting their pins appropriately; in this case, the carry out of the previous FA is connected to the carry-in of the next FA in a ripple-carry manner.

In the same manner, a system design can be recursively broken down into components, each of which is composed of smaller components until the smallest components can be described in terms of gates and/or transistors. At any level of the hierarchy, each component is treated as a black-box with a known input-output behavior, but how that behavior is implemented is unknown. Each black-box is designed

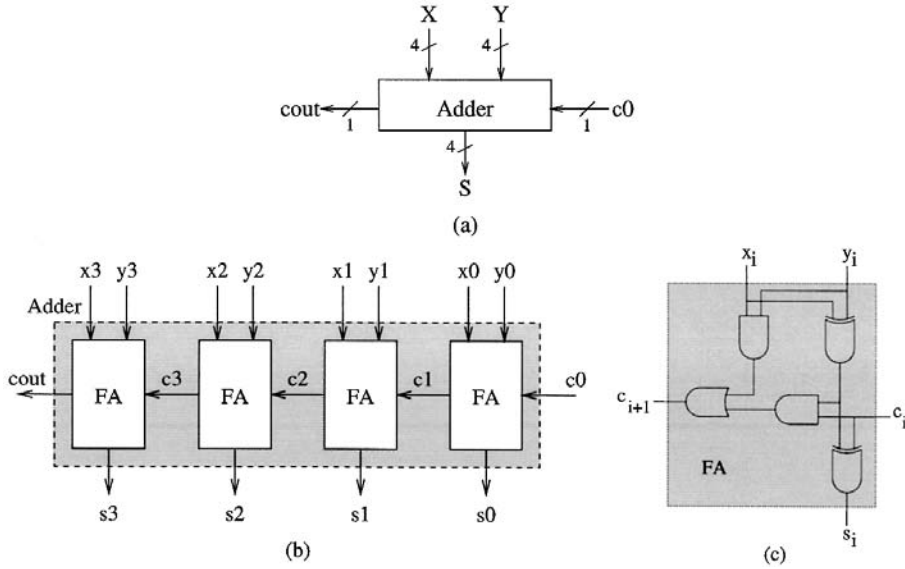


FIGURE 64.4 An example of hierarchical design: (a) a 4-bit ripple-carry adder; (b) internal view of the adder composed of full adders (FAs); (c) full-adder logic schematic.

by building simpler and simpler black-boxes based on the behavior of the component. The smallest primitive components (such as gates and transistors) are used at the lowest level of hierarchy.

Besides assisting in breaking down the complexity of a large system, hierarchy also allows easier conceptualization of the design and its functionality. At higher levels of the hierarchy, it is easier to understand the functionality at a behavioral level without having to worry about lower-level details. Hierarchical design also enables the reuse of components with little or no modification to the original design.

The design approach described above is a *top-down design approach* to hierarchy. The top-down design approach is a recursive process that takes a high-level specification and successively decomposes and refines it to the lowest level of detail and ends with integration and verification. This is in contrast to a *bottom-up approach*, which starts by designing and building the lowest-level components and successively using these components to build components of ever-increasing complexity until the final design requirements are met.

Since a top-down approach assumes that the lowest-level blocks specified can, in fact, be designed and built, the whole process has to be repeated if a low-level block turns out to be infeasible. Current design teams use a mixture of top-down and bottom-up methodologies, wherein critical low-level blocks are built concurrently as the system and block specifications are refined. The bottom-up approach attempts to abstract parameters of the low-level components so that they can be used in a generic manner to build several components of higher complexity.

64.5 Design Representation and Abstraction Levels

Another hierarchical approach is based on the concept of *design abstraction*. This approach views the design with different degrees of resolution at different levels of abstraction. In the design process, the design goes through several levels of abstraction as it progresses from concept to fabrication — namely, *system*, *register-transfer*, *logic*, and *geometrical*.[1] The *system-level* description of the design consists of a behavioral description in terms of functions, algorithms, etc. At the *register transfer level*, the circuit is represented by arithmetic and storage units and corresponds to the register transfer level (RTL) discussed earlier. The register-level components are selected and interconnected so as to achieve the functionality

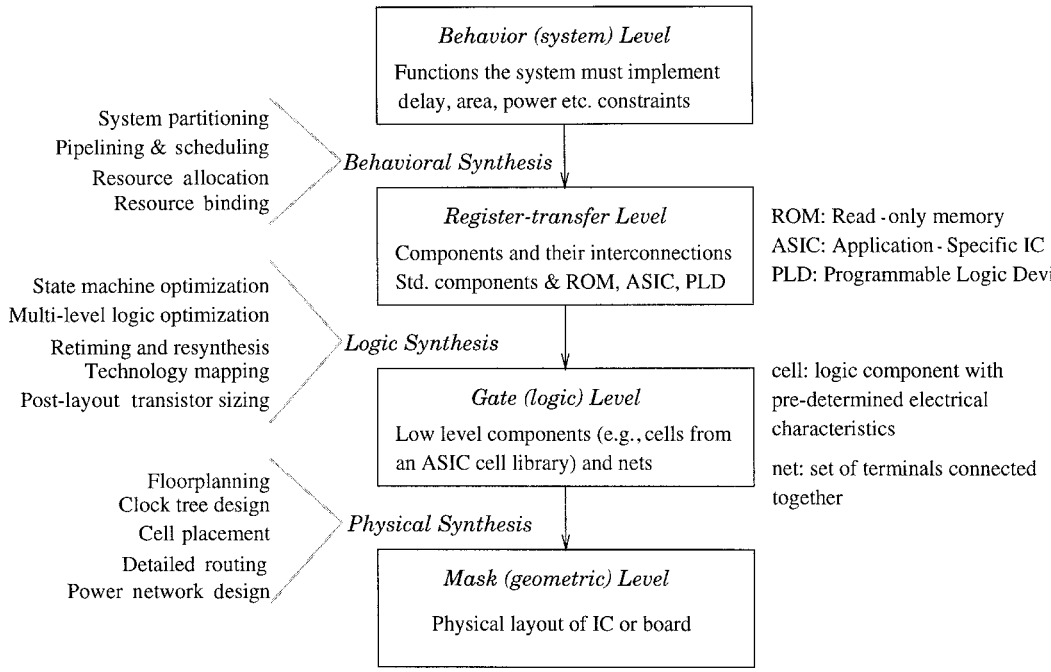


FIGURE 64.5 Simplified ASIC design flow: the progress of the design from the behavior to mask level and the synthesis processes and steps involved.

of the design. The *logic* level describes the circuit in terms of logic gates and flip-flops and the behavior of the system can be described in terms of a set of logic functions. These logic components are represented at the *geometric* level by a layout of the cells and transistors using geometric masks.

These levels of abstraction can be further understood with the help of the simplified ASIC design flow shown in Fig. 64.5.[26] This figure shows *behavior* as the initial abstraction level which represents the system level functionality of the design. The *register-transfer level* comprises components and their interconnections and, for more complex systems, may also comprise standard components such as ROMs (read-only memory), ASICs, etc. The logic level corresponds to the *gate* level representation and the set of *masks* of the physical layout of the chip correspond to the geometric level.

This figure also shows the synthesis processes and the steps involved in each process. These synthesis processes help refine the design from one level of detail to the next finer level of detail. These synthesis processes are known as behavioral synthesis, logic synthesis, and physical synthesis, and each of these synthesis processes are discussed in detail in later sections. It is possible to go from one level of detail to the next by following the steps within the synthesis process, either manually or with the help of CAD tools.

The circuit can also be viewed at different levels of design detail as the design progresses from concept to fabrication. These different design representations or views are differentiated by the type of information that they capture. These representations can be classified as *behavioral*, *structural*, and *physical*.[4] In a *behavioral representation*, only the functional behavior of the system is described and the design is treated as a black-box. A *structural representation* refines the design by adding information about the components in the system and their interconnection. The detailed physical characteristics of the components are specified in the *physical representation*, including the placement and routing information.

The relationships between the different abstraction levels and design representations or views is captured by the Y-chart shown in Fig. 64.6.[10] This chart shows how the same design at the system level can have a behavioral view and a structural view. Whereas the behavioral view would conceptualize the design in terms of flowcharts and algorithms, the structural view would represent the design in terms of processors, memories, and other logic blocks. Similarly, the behavioral view at the register-transfer level

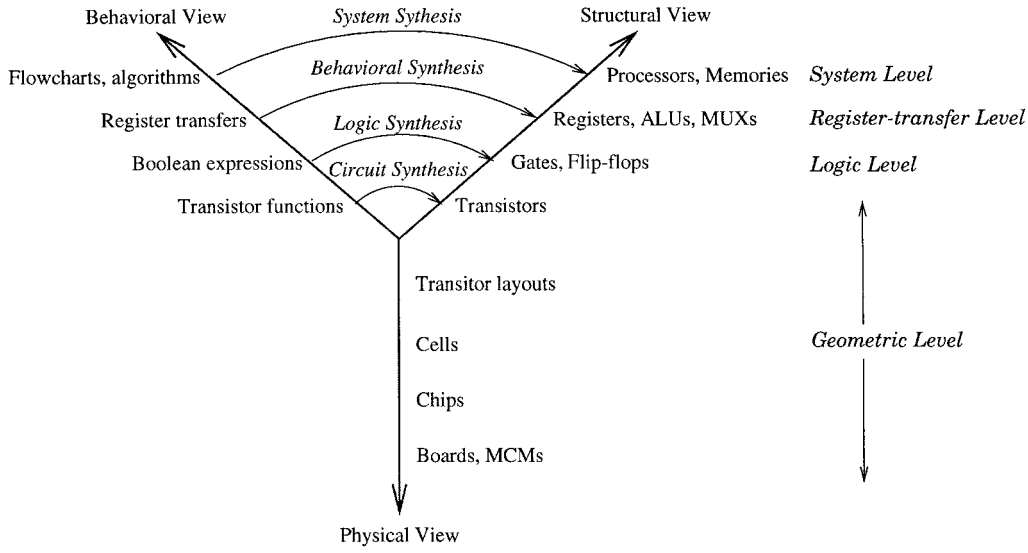


FIGURE 64.6 Y-chart: relationship of different abstraction levels and design representations.

would represent the register transfer flow by a set of behavioral statements, whereas the structural view would represent the same flow by a set of components and their interconnections. At the logic level, a circuit can be represented with Boolean equations or finite-state machines in the behavioral view, or it can be represented as a network of interconnected gates and flip-flops in the structural view. The geometric level is represented as transistor functions in the behavioral level, as transistors in the structural view, and as layouts, cells, chips, etc. in the physical view. In this way, the Y-chart model helps to understand the various phases, levels of detail, and views of a design. There have been many extensions to this model, including adding aspects such as testing and design processes.[11]

64.6 System Specification

In the following sections, we will discuss each of the steps in the design process of an ASIC. Any design or product starts with determining and capturing the requirements of the system. This is typically done in the form of a *system requirements specification* document. This specification describes the end-product requirements, functionality, and other system-level issues that impose requirements such as environment, power consumption, user acceptance requirements, and system testing. This leads to more specific requirements on the device itself, in terms of functionality, interfaces, operating modes, operating conditions, performance, etc.

At this stage, an initial analysis is done on the system requirements to determine the feasibility of the specification. It is determined which design style will be used (see Section 64.2) and the foundry, process, and library are also selected. Some other parameters such as packaging, operating frequency, number of pins on the chip, area, and memory size are also estimated.

Traditionally, for simple designs, design entry is done after the high-level architecture design has been completed. This design entry can be in the form of *schematics* of the blocks that implement the architecture. However, with increasing complexity of designs, concerns about system modeling and verification tools are becoming predominant. System designers want to ensure hardware design quality and quickly produce a working hardware model, simulate it with the rest of the system, and synthesize and formally verify it for specific properties. Hence, designers are adopting high-level hardware description languages (HDLs) for the initial specification of the system. These HDLs are simulatable and, hence, the functionality and architectural design can be simulated to verify the correctness and fulfillment of end-product

requirements. In present ASIC design methodologies used in the industry, HDLs are typically used to capture designs at a register-transfer level and logic synthesis tools are then used to synthesize the design.

However, recently the use of executable specifications for capturing system requirements is becoming popular, as proposed in the *Specify-Explore-Refine* (SER) methodology for system design.[4] After this *specify* phase, the *explore* phase consists of evaluating various different system components to implement the system functionality within the design constraints specified. The specification is updated with the design decisions made during the exploration phase in the *refine* phase. This methodology leads to a better understanding of the system functionality at a very early stage in the process. An *executable specification* is particularly useful to validate the product functionality and correctness and for the automatic verification of various design properties. Executable specifications can be easily simulated and the same model can be used for synthesis. Current design methodologies produce functional verification models in C or C++ and these are then thrown away and the design is manually entered again for the design tools.

The selection of a language to capture the system specification is an area of active research. The language must be easy to understand and program, and must be able to capture all the system's characteristics besides having the support of CAD tools which can synthesize the design from the specification. Many languages have been used to capture system descriptions, including VHDL,[12] Verilog,[13] HardwareC,[14] Statecharts,[15] Silage,[16] Esterel,[17] and SpecSyn.[18] More recently, there has been a move toward the use of programming languages for digital design due to their ability to easily express executable behaviors and allow quick hardware modeling and simulation and also due to system designers' familiarity with general-purpose, high-level programming languages such as C and C++.[53]

These languages have raised the level of abstraction at which the designer specifies the design to being closer to the conceptual model. The conceptual behavioral design can then be partitioned and structured and components can be allocated. In this manner, the design progresses from a purely functional specification to a structural implementation in a series of steps known as *refinement*. This methodology leads to lower design times, more efficient exploration of a larger design space, and lower re-design time.

64.7 Specification Simulation and Verification

Once a design has been captured in a hardware description language or a schematic capture tool, the functionality of the specification needs to be verified. The most popular technique for design verification is *simulation*, in which a set of input values are applied to the design and the output values are compared to the expected output values. Simulation is used at every stage of the design process and at various levels of design description: behavioral, functional, logic, circuit, and switch level.

Formal verification tools attempt to do equivalence checks between different stages of a design. Currently, in the industry, once the requirements of a design have been finalized, a functional specification is captured by a software model of the design in C or C++, which also models other design properties and architectural decisions. This software model is extensively simulated to verify that the design meets the system requirements and to verify the correctness of the architectural design. Often, a C or C++ model is used as the *golden model* against which the hardware model is verified at every stage of the design. The functional specification is translated (usually manually) into a structural RTL description, and their outputs are compared by simulation to verify that their functionality is equivalent. This is typically done by applying a set of input patterns to both the models and comparing their outputs on a cycle-by-cycle basis. As the design is further refined from RTL to logic level to physical layout, at each stage, the circuit is simulated to verify functional correctness and some other design properties, such as timing and area constraints.

The simulations of the RTL, logic, and physical level descriptions are done by different kind of simulators.[19] *Logic-level* simulators simulate the circuit at the logic gate level and are used extensively to verify the functional correctness of the design. *Circuit-level* simulation, which is the most accurate simulation technique, operates at a circuit level. The SPICE program is the foremost circuit simulation and analysis tool.[20] SPICE simulates the circuit by solving the matrix differential equations for circuit

currents, voltages, resistances, and conductances. Switch-level simulators, on the other hand, model transistors as switches and, unlike logic simulators, wires are not assumed to be ideal but instead are assumed to have some capacitance. Another simulator, *RSIM*, is a switch-level simulator with timing, which models CMOS gates as pull-down or pull-up structures and calculates their resistance to power or ground, so that it can be used with output capacitance to determine rise and fall times.[21]

Logic-level simulators are typically *event-driven*. These model the system in a discrete event system by defining appropriate events of interest and how the events are propagated throughout the model.[6,22] Hardware description languages (HDLs) such as VHDL and Verilog[12,13] have been designed based on event-driven simulation semantics. They have constructs to represent hardware features such as concurrency, hierarchy, and timing. Extensive simulation and functional verification techniques are used by designers at every stage of the design to ensure that no *bugs* are introduced in the process of refining the design from the behavioral level to the final layout.

64.8 Architectural Design

After the design specification has been captured, the system is partitioned into blocks with clearly defined functionality and the interfaces and interaction between the blocks are defined. This structuring of the design is known as *architectural* design. Besides partitioning, architectural decisions include deciding number and type of components and their interconnects such as adders, multipliers, ALUs, buses, etc., whether the design will be pipelined[2], number of pipeline stages, and the operations in each pipeline stage. These high-level architectural decisions have traditionally been done by a few experienced system architects in the design team. However, in the last decade, CAD tools such as high-level synthesis have been introduced which automatically or interactively make many of these architectural decisions and schedule the design, allocate components for it and interconnect them to create a register transfer level design optimized for different parameters.[24,27]

Behavioral Synthesis

Behavioral or *high-level synthesis*, which is the automated synthesis of systems from behavioral descriptions, has received a lot of attention recently due to its ability to provide the low turn-around time required for an ASIC design. High-level synthesis accepts a behavioral description of a system and generates a data path for this description at a register-transfer level.[29-31] High-level synthesis tools allow designers to work at a system level closer to the original conceptual model of the system. High-level synthesis tools can be targeted to optimize the area, performance, power, and testability of the final design. The tasks in high-level synthesis can be broadly classified into allocation, scheduling, and binding. *Allocation* consists of determining the number and type of components and other resources that are required for the implementation of the design. These components and resources are at the register-transfer level (RTL) and are taken from a library of available modules, which includes components such as ALUs, adders, multipliers, register files, registers, and multiplexers. Allocation also determines the number, width, and type of each bus in the system.

Scheduling assigns each of the operations in the behavioral description to time intervals, also known as control steps. The data flows from one stage of registers to the next during each control step and may be operated upon by a functional unit. The control steps are usually the length of a clock-cycle. The operations in each control step are then assigned to particular register-level components by the *binding* task. Hence, operations are assigned to functional units, variables to storage units, and the interconnect between the various units are also established.

Consider the sample data flow graph shown in Fig. 64.7(a) and its corresponding data path shown in Fig. 64.7(b). This data path was synthesized using a high-level synthesis system.[30] The data flow graph

[2]Pipelining is a technique where a series of operations are done in a pipeline or assembly-line fashion so as to increase concurrency among different types of operations.

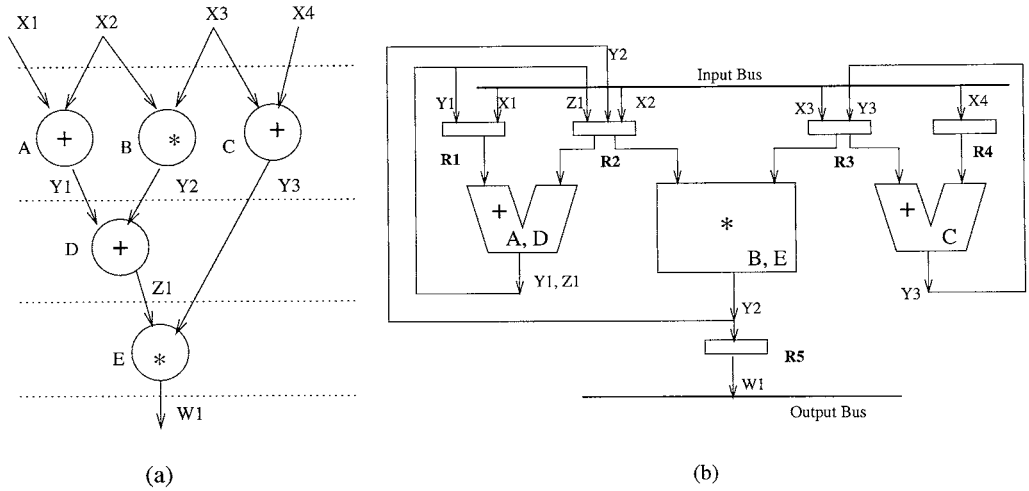


FIGURE 64.7 High-level synthesis: (a) a sample data flow graph, (b) corresponding data path.

shows the variables *X*1, *X*2, *X*3, *Y*1, *Y*2, *Y*3, *Z*1, and *W*1, and the operations *A* to *E*. The data path in Fig. 64.7(b) shows the mapping of the variables to the registers and the operations to the functional units. Multiplexers are not shown in this figure. This example demonstrates the ability of CAD tools to synthesize behavioral descriptions into data paths. These CAD tools can also synthesize the control logic and make high-level decisions, such as number of pipeline stages, etc.[27]

Testable Design

Testability of digital circuits has become a major concern with the increasing complexity of designs. Testability refers to the ability to detect manufacturing faults in a fabricated chip. Designers are increasingly using a *design for testability* (DFT) methodology to ensure that the circuit is testable. DFT attempts to modify the circuit during the design phase without affecting its functionality so as to make it testable. There are several approaches and techniques that are used to make chips and the individual components in them testable. Additional test hardware and pins are added to the chip, such as *boundary scan* test hardware[37] which enable testing the chip, introduce test modes to the chip functionality, and provide pins dedicated to shifting in and out of the test vectors and their responses. The testability of the internal components of the chip is enhanced primarily by two techniques: serial scan and built-in self-test (BIST). In the first approach, the components within a chip are tested by applying test vectors to the input pins of the chip and shifting out the output patterns and checking for correctness. In the second approach, known as the *built-in self-test* (BIST) technique, the chip is tested by specialized hardware built-in within the chip that self-tests the components in the chip. The former approach is known as the *full-scan* or *partial-scan* test technique since all or some of the registers in the chip are connected in a test scan chain.

Full-Scan Testing

In practice, the *full-scan* technique for testing the data path in a chip is more popular among designers. This technique improves the observability and controllability of the circuit by using *scan registers*.[37] A scan register has both serial shift and parallel-load capability and has additional serial-in and serial-out pins over a standard register. All the scan registers in the circuit are tied together in a chain by connecting the serial-out of a register to the serial-in of the next register.

During normal circuit operation mode, the scan registers behave as parallel load registers. However, in the test mode, a test pattern is serially scanned into all the registers of the circuit and then the circuit is clocked and the values in the registers are serially shifted out. The output bit vector values are compared with the expected results to verify that the circuit is functioning correctly. In this way, only one serial-in

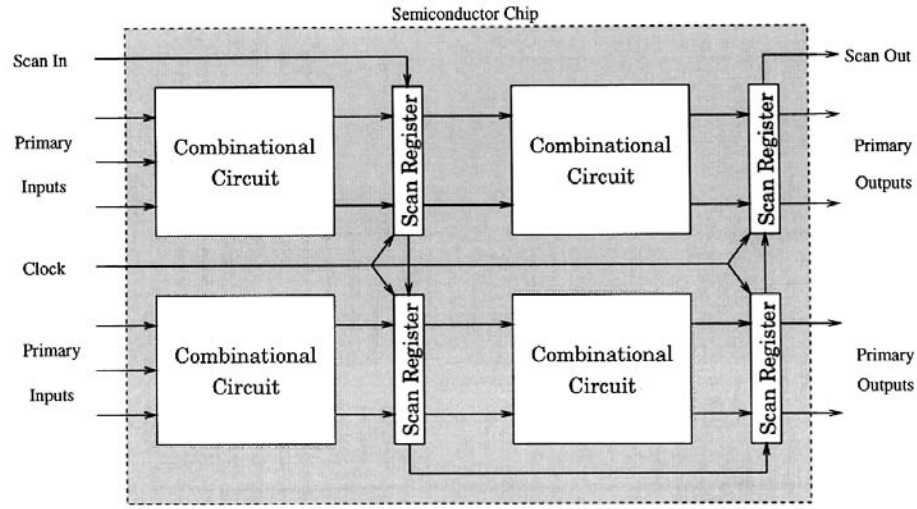


FIGURE 64.8 Full-scan register-based design.

pin and one serial-out pin has to be assigned at the chip level. However, since for each test vector that is applied to the chip, it has to be scanned in serially and then the output has to be serially scanned out, this approach is very slow. The slow speed of testing using full-scan is its main disadvantage. The overhead of scan-based test techniques comprises area overhead and performance slow-down. However, the overhead is relatively low compared to other schemes such as BIST.

The full-scan technique is demonstrated in Fig. 64.8. In this figure, there are four combinational blocks each of which feeds into registers which have been modified to be scan registers. There is a scan-in pin and a scan-out pin at the chip level and all the scan registers are tied together to form a scan chain.

Built-In Self-testing

The built-in self-test (BIST) methodology has gained popularity over the past decade and techniques have been demonstrated to incorporate it into behavioral synthesis tools.[30,40] Memory blocks such as RAMs (random access memories) are usually tested by inserting built-in self-test (BIST) logic in the memory design. These BIST circuits apply pseudo-random patterns to the memory and test it by several techniques such as writing data into an address location and then reading it back out and comparing the two.

Datapath units can also be tested by BIST techniques by applying a set of test vectors to the inputs of the units and doing a *signature analysis* of the output bit stream.[37,39] This signature analysis is enough to ensure that the unit is not faulty. The input test vectors are generated in a pseudo-random manner using registers which are configured as *pseudo-random pattern generators* (PRPGs). Similarly, signature analysis is done by configuring registers as *signature analyzers* (SAs). Registers which can be configured in this manner are known as *built-in logic block observers* (BILBOs). One way, then, of ensuring testability of a functional unit is by creating a *n:m* embedding for the functional unit, where *n* is the number of inputs to the functional unit and *m* is the number of outputs. In such an embedding, it is ensured that each functional unit is fed by at least *n* registers and the functional unit feeds at least *m* registers which are different from the input registers. The input registers are configured as PRPGs and the output registers as SAs. In the *test mode* of the chip, the input PRPGs generate a test vector, a clock cycle is applied to the functional unit's embedding, at the end of which the outputs of the unit are analyzed by the output registers configured as SAs.

In this way, each functional unit can be tested by running the chip in test mode. However, to reduce the test time of the chip, multiple functional units can be tested simultaneously provided that any input PRPG register of one unit is not the output SA register of another. A test schedule or plan can be generated for testing the various units in as few test sessions as possible.[38]

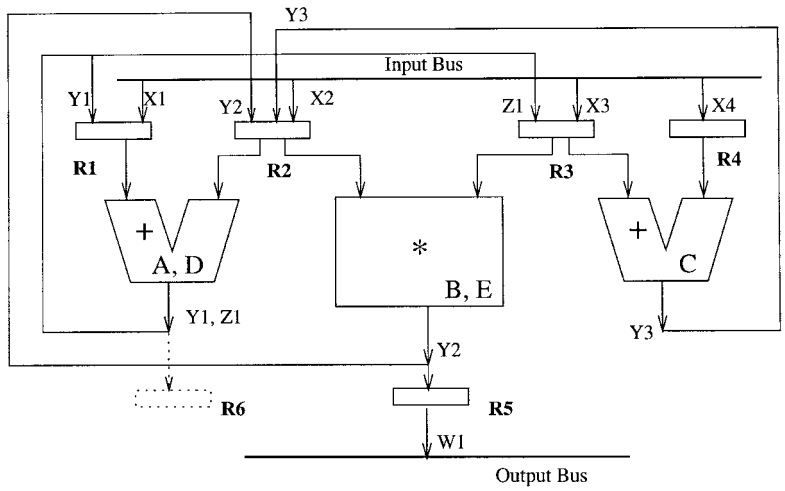


FIGURE 64.9 Built-in self-test (BIST)-based testable data path for sample data flow graph.

Consider the example of the data path of the sample data flow graph shown earlier in Fig. 64.7(b). In this figure, the multiplier module is part of a 2-1 embedding consisting of registers *R*2, *R*3, and *R*5. In the test mode, *R*2 and *R*3 are configured as pseudo-random test pattern generators, whereas *R*5 is configured as a signature register. However, both the adders cannot be part of a 2-1 embedding since their outputs are stored in the same registers as their inputs. By adding a register *R*6 (shown dotted in Fig. 64.9) at the output of the left adder, we can make this adder testable since it becomes part of a 2-1 embedding consisting of input registers *R*1 and *R*2 and output register *R*6. The other adder can be made testable by changing the binding of variables to registers such that *Z*1 is mapped to *R*3 and *Y*3 is mapped to *R*2, along with the necessary changes in the interconnect. If the modified embedding is used, the second adder will be the part of a 2-1 embedding which consists of input registers *R*3 and *R*4 and output register *R*2. The modified testable data path is shown in Fig. 64.9. There are several other ways that this circuit can be modified to make it testable.

Some of the main challenges in this BIST-based methodology for testing data path units are ensuring that each functional unit is part of a *n*:*m* embedding while at the same time converting as few registers into BILBOs (since these are more expensive in terms of area) and generating an efficient test schedule such that the total test time is minimum.

Although in this section we have attempted to introduce the issues in testability and design for testability, it is by no means a complete picture of the field of testing. Several test issues such as delay faults, mixed-signal test, partial scan have not been discussed. There are several techniques and test styles which can be adopted, depending on the characteristics of the system under design. Other chapters in this book offer a more detailed discussion on the subject.

64.9 Logic Synthesis

Logic synthesis deals with the synthesis and optimization of circuits at the logic gate level.[5,7-9] Digital circuits typically have sequential and combinational components. These can be specified by finite-state machines, state transition diagrams or tables, Boolean equations, schematic diagrams, or HDL descriptions. Finite-state machine representations are optimized by state minimization and encoding and Boolean functions are optimized either by two-level optimization techniques which are exact or by heuristic multi-level optimization techniques.

Logic synthesis includes a range of optimizations and techniques like state machine optimization, multi-level logic optimization, retiming, re-synthesis, technology mapping, or post-layout transistor sizing. The optimization steps are selected and ordered according to the chosen optimization metric,

whether it may be area, speed, power, or a tradeoff between these. These steps are divided into two phases: the *technology-independent* phase, where the logic circuit is optimized by Boolean or algebraic manipulation or state minimization, and the *technology-mapping* phase, in which the logic network is mapped into a technology library of cells and then, transistor-level optimizations are performed.

Since circuits are usually a combination of combinational and sequential parts and the techniques to optimize the two differ a lot, we discuss each one separately.

Combinational Logic Optimization

Combinational circuits can be modeled by two-level sum-of-products expressions. These expressions can be optimized by two-level minimization tools such as Espresso, Mini, or Presto.[1,41] Two-level logic networks can be easily mapped onto macrocell-based design styles such as PLAs (programmable logic arrays). However, in practice, logic networks are usually multi-level and, hence, multi-level logic optimization tools such as MIS[42] are becoming popular. Unlike two-level logic networks, multi-level network graphs can be mapped onto cell libraries with complex n-level gates, thereby allowing more complex cell and array-based design styles.

To demonstrate the steps in technology-independent steps in combinatorial logic optimization, we show the optimization of Boolean functions representing two-level logic networks in a sum-of-products format of the logic variables. Boolean functions can be optimized by minimizing the number of operators using either *map-based* or *table-based* methods. The map-based method uses *Karnaugh maps* to minimize a Boolean function as shown in the example below. Consider the Boolean function:

$$F = a'b'c'd' + a'b'c'd + a'b'cd' + a'b'cd + a'bc'd + a'bcd' + ab'cd' + a'bcd + ab'cd + abcd$$

where a, b, c, and d are single-bit Boolean variables. The Karnaugh map corresponding to this example is shown in Fig. 64.10(a).[25] This map represents the terms in the Boolean expression by assigning a 1 in the squares that correspond to a term in the expression. Each term in a Boolean function is called a *minterm*. For any Boolean function with n-variables or literals, it has 2^n possible minterms and a *n-cube* is defined as a minterm with all n-variables. A *subcube* is a minterm with fewer variables than n in it. From the Karnaugh map shown, we determine that the *prime implicants* (PIs), which are the subcubes not contained in any other subcube, are $a'b'$, $a'c$, $a'd$, cd, $b'c$. These are marked in the figure by dashed boxes. The dashed boxes were created by grouping together the maximal set of minterms in groups of multiples of 2 (i.e., 2, 4, 8, etc.). *Essential primary implicants* are the prime implicants which include a minterm that is not included in any other subcube. For this example, all the prime implicants are also essential prime implicants. A *cover* is a set of prime implicants such that each minterm in the Boolean

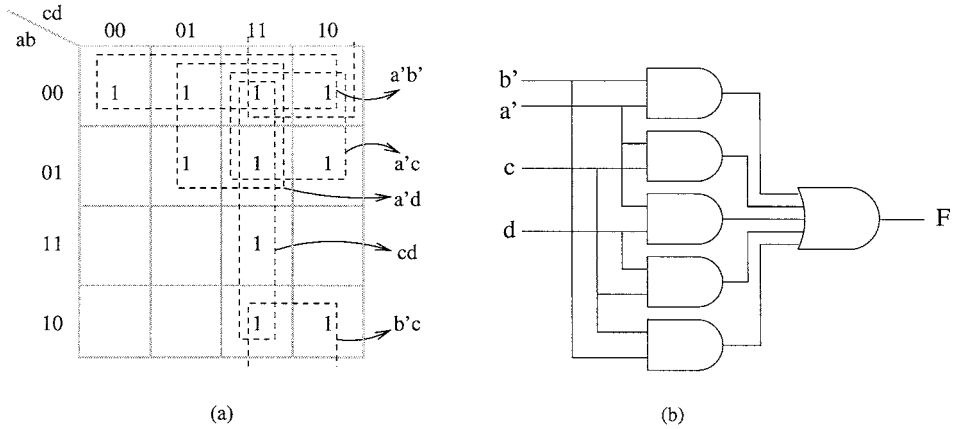


FIGURE 64.10 An example function: (a) Karnaugh map, (b) circuit implementation.

function is contained in at least one prime implicant. A *minimal* cover is a selection of the minimum number of prime implicants that form a cover over all the minterms in the function. For this example, a minimal cover is $a'b'$, $a'c$, $a'd$, cd, $b'c$. Hence, the reduced Boolean function is:

$$F = a'b' + a'c + a'd + cd + b'c$$

The circuit corresponding to this function is shown in Fig. 64.10(b). The 5-input OR gate at the end of the circuit can be implemented by splitting it into several 2-input OR gates.

The same minimization can be done using tabular methods such as the Quine-McCluskey method.[25] This method represents the same information in tables which then reduce the minterms by iteratively finding subcubes with fewer variables. The reader is referred to standard texts on digital design for further discussion on this method.

The Karnaugh map shown in Fig. 64.10(a) conceptually demonstrates the combinational logic optimization process. However, in practice, two-level optimizers such as *Espresso* are used for logic optimization. Espresso uses an expand-irredundant-reduce iterative algorithm to reduce the size of the given Boolean function.[41] A n-variable function can be represented by a set of points in n-dimensional space. The function then has an *on-set*, which is the set of points for which the function's value is 1, an off-set, which is the set of points for which the function's value is 0 and a *don't-care* or *dc-set*, which is the set of points for which the function's value is don't care. The basic Espresso algorithm first *expands* each cube in the on-set to make it as large as possible, without covering a point in the off-set (points in the dc-set may be covered). Then, for points covered by several cubes, the smaller cubes are removed in favor of the larger covering cubes in the *irredundant* step. Finally, the cubes are *reduced* so as to minimize the variables in the cubes.

The example and strategies discussed above demonstrate the two-level optimization methodology. The final circuit implementation for the example, (see Fig. 64.10(b)) has two stages of logic. However, cell libraries used to map the gates in the logic circuit to the gates available from the foundry, usually have more complex gates which are a combination of several gates such as AND-OR, OR-AND, or NOR-AND gates. To fully utilize these cell libraries, multi-level logic optimization techniques are used. These techniques are not restricted to two-level logic networks but instead deal with multiple-level logic circuits. This provides the necessary flexibility required to map the logic network to complex cells in the technology library, hence optimizing area and delay. However, multi-level optimization techniques are not exact, i.e., only heuristics exist for modeling and optimizing multiple-level networks. For further discussion on this subject, the reader is referred to Ref. 1.

Sequential Logic Optimization

Sequential circuits are usually represented by a finite-state machine (FSM) model. This consists of a combinational circuit and a set of registers as shown in Fig. 64.11. The model has a set of inputs, I, a set of outputs O, the state S, and a clock signal. The clock signal defines the *clock cycle*, which is a time

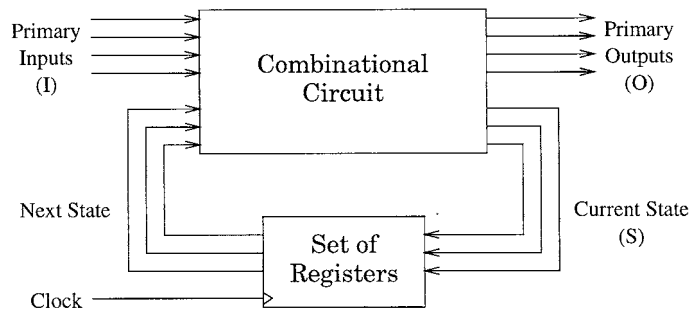


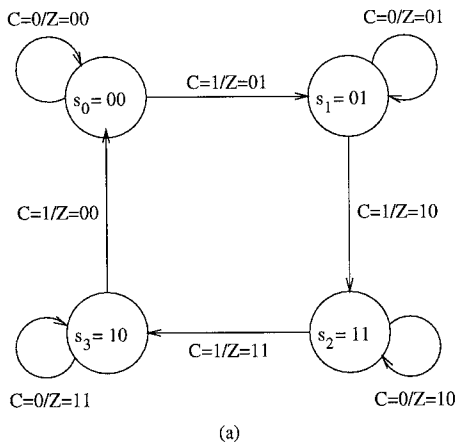
FIGURE 64.11 Finite state machine model.

interval in which the combinational circuit analyzes the inputs and the state to calculate the outputs and the next state. At every clock cycle, the data computed by the combinational circuit is stored in the registers along with other state and control information.

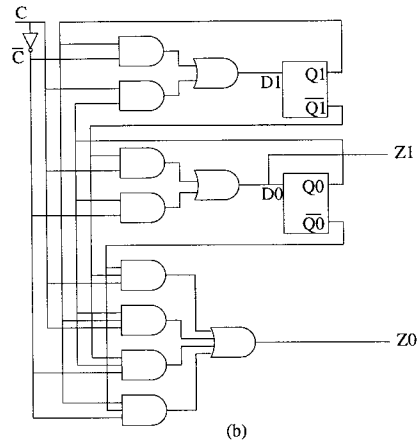
A finite-state machine (FSM) is defined by the quintuple $<S,I,O,f,h>$ where S, I, and O are the set of states, inputs, and outputs, respectively, and f and h represent the next state and output calculation functions. The next state function f can be represented as $f: S \times I \rightarrow S$ and the output function h can be either represented as $h: S \times I \rightarrow O$ or as $h: S \rightarrow O$, depending on whether the finite-state machine is implemented as a *Mealy* machine or a *Moore* machine. In the Mealy machine, the output function is dependent on the inputs and the state, whereas in the Moore machine the output is state based only.

In a sequential circuit represented by an FSM, the set of states, inputs and outputs, S, I, and O correspond to k flip-flops, $Q_0, \ldots, Q_{k-1}$, n input signals, $I_0, \ldots, I_{n-1}$ and m output signals, $O_0, \ldots, O_{m-1}$. Each of these correspond to a single bit in the implementation. The finite-state machine model is usually represented using state transition diagrams or state tables.[1,25] State transition diagrams are mainly optimized by *state minimization* and *state encoding* (explained in the next subsection).

Let us first discuss an example to demonstrate the design of sequential circuits. Consider the example of a modulo-4 counter shown in Fig. 64.12. Figure 64.12(a) shows the finite-state machine transition graph for the counter. The counter counts from 0 to 3 back to 0 whenever the count signal C is 1. When the count signal C is 0, the counter stays in the same state. The counter outputs the count Z at each clock cycle. Hence, the state transition graph has four states S_0 to S_3 corresponding to the count states 0 to 3. There is a transition from one state to the next if $C = 1$ and the output Z is the count at that time. If $C = 0$, the state does not change and the output Z is the same as when entering the state. The states S_0 to S_3 have been encoded as 00, 01, 11, 10, respectively. This is an example of an input-based or Mealy-type FSM.



(a)



(b)

Present State	Next State/Output D1 D0 / Z1 Z0	
Q1 Q0	C = 0	C = 1
00	00/00	01/01
01	01/01	11/10
11	11/10	10/11
10	10/11	00/00

(c)

C / Q1 Q0	D1 D0: 0	1
00	0 0	0 1
01	0 1	1 1
11	1 1	1 0
10	1 0	0 0

(d)

C / Q1 Q0	Z1 Z0: 0	1
00	0 0	0 1
01	0 1	1 0
11	1 0	1 1
10	1 1	0 0

(e)

FIGURE 64.12 Sequential circuit example: modulo-4 counter (a) FSM for counter, (b) circuit for the counter, (c) state transition table, (d) next state Karnaugh map, (e) output Karnaugh map.

The information from the FSM can be captured in a state transition table as shown in Fig. 64.12(c). In this figure, the present and the next states are shown using their encoding and are marked by bit variables Q_1 Q_0 and D_1 D_0, respectively. The output Z is a two-bit variable Z_1 Z_0 which goes from 0 to 3 (or 00 to 11). The Karnaugh maps corresponding to the next state and the output bit vectors are shown in Figs. 64.12(d) and 64.12(e) respectively. The maximal coverings for all the bits in the next state variables and the output variable are shown in these Karnaugh maps by dotted boxes. Note that although the Karnaugh Maps for D_1 D_0 and Z_1 Z_0 have been grouped together, their coverings and optimizations are independent. From these coverings, we get the following reduced Boolean equations for the bit variables:

$$D_1 = Q_1\overline{C} + Q_0C \qquad Z_1 = Q_1\overline{C} + Q_0C$$
$$D_0 = Q_0\overline{C} + \overline{Q}_1C \qquad Z_0 = \overline{Q}_1Q_0\overline{C} + Q_1\overline{Q}_0\overline{C} + \overline{Q}_1\overline{Q}_0C + Q_1Q_0C$$

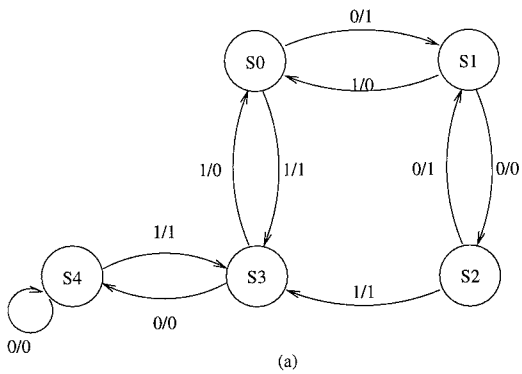
The circuit diagram corresponding to these equations is shown in Fig. 64.12(b). The circuit has two D-flip-flops which correspond to the two-bit variables in the state, and the combinational part has been implemented using simple AND, OR, and NOT gates. Note that in this example, the state minimization and encoding steps are assumed to have already been done.

State Minimization and Encoding

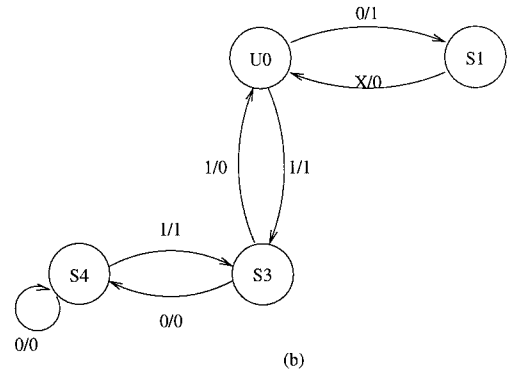
State minimization aims at reducing the number of machine states used to represent an FSM. Since the minimum number of bits required to encode n states is $[\log_2 n]$, reducing the number of states can lead to a reduced number of bits and, hence, flip-flops required to encode the states. It also leads to fewer transitions, fewer logic gates, and fewer inputs per gate. These reductions not only lead to lower area cost but also speed up the design and reduce the power consumption.

State minimization can be done by finding *equivalent* states and by using *don't-care* information to remove states. Two states are equivalent if and only if, for every input, both the states produce the same output and the corresponding next states are equivalent.

Consider the example state transition graph shown in Fig. 64.13(a). The state transition table corresponding to this graph is shown in Fig. 64.13(c). State minimization can be done in two steps. The first



(a)



(b)

Pres. State	Next State/Out	
	Inp I=0	Inp I=1
s_0	$s_1/1$	$s_3/1$
s_1	$s_2/0$	$s_0/0$
s_2	$s_1/1$	$s_3/1$
s_3	$s_4/0$	$s_0/0$
s_4	$s_4/0$	$s_3/1$

(c)

Pres. State	Next State/Out	
	Inp I=0	Inp I=1
$u_0 = \{s_0, s_2\}$	$u_1, u_1/1$	$u_1, u_1/1$
$u_1 = \{s_1, s_3\}$	$u_0, u_2/0$	$u_0, u_0/0$
$u_2 = \{s_4\}$	$u_2/0$	$u_1/1$

(d)

Pres. State	Next State/Out	
	Inp I=0	Inp I=1
u_0	$s_1/1$	$s_3/1$
s_1	$u_0/0$	$u_0/0$
s_3	$s_4/0$	$u_0/0$
s_4	$s_4/0$	$s_3/1$

(e)

FIGURE 64.13 An example of state minimization: (a) original state transition graph, (b) minimized state transition graph, (c) original state transition table, (d) states grouped based on their outputs, (e) minimized state transition table.

step is finding the states with the same outputs for the same inputs. We group these states such that states in the same group have the same output for each input. This is shown in Fig. 64.13(d). There are three groups u_0, u_1, and u_2 which, respectively, give output 1, 0, and 0 when the input 0 is applied and give output 1, 0, and 1 when the input 1 is applied. In the next step, we compare the next states for each state in a group for all inputs. If the next state for two states within a group is in the same group, then the two states are considered equivalent. In this example, we find the states s_0 and s_2 in the group u_0 are equivalent since all the next states of these two states are in the same group. Hence, these two states can be combined into one state and the minimized state transition table is shown in Fig. 64.13(e). The corresponding minimized state transition graph for the example is shown in Fig. 64.13(b). Note, that the transition from $s1$ to $u0$ is denoted as X/0 since for all inputs, when in state s_1, the next state is u_0 and the output is 0.

After the states have been minimized, *state encoding* is performed to assign a binary representation to the states of the finite-state machine. In the example shown earlier in Fig. 64.13(b), the minimized state transition graph has four states, whereas the original state transition graph had five states (see Fig. 64.13(a)). Hence, whereas it would have taken 3 bits to encode the five states in the original FSM, the reduced FSM requires only 2 bits for the encoding. Fewer encoding bits implies fewer flip-flops in the circuit and, hence, reduced area and increased speed of the final design. There are several other encoding methodologies such as gray encoding, NRZ encoding, etc., which are used to reduce circuit switching, bus switching, etc.[1]

Technology Mapping

Technology mapping forms the link between logic synthesis and physical design. After logic synthesis, a circuit-level schematic or netlist of the design is created using a vendor-independent logic library. This library has elements such as low-level gates, flip-flops, latches, and at times, multiplexers, counters, and adders. The schematic entry tool then generates a netlist of the elements with their interconnections. Typically, a *netlist translator* along with a vendor-specific library is used to replace the vendor-independent generic elements and generate the netlist in a particular vendor's netlist format. This allows the schematic entry or netlist generation to be independent of the vendor-specific library.

The process of transforming the generic cell based logic network into a vendor library-specific network is known as *library binding* or *technology mapping*. This step allows us to retarget the same design to different technologies and implementation styles. The library contains a set of parameterized logic cells. These cells may be primitive or a combination of a set of cells to produce a commonly used functionality such as adders, shifters, etc. Typically, the cell library vendor provides different libraries optimized for area, performance, power, and/or testability.

Each cell in the vendor library contains a physical layout of the cell, its timing model (delay characteristics and capacitances on each input), a wire load model, a behavioral model (VHDL/Verilog model), circuit schematic, cell icon (for schematic tools), and for bigger cells, its routing and testing strategy. CAD tools use the timing characteristics to analyze the circuit and determine the capacitances at each node in the netlist, and use the delay formulas along with the timing characteristics of each element to compute the delays for each node. Wiring capacitances are included by estimating a wire-load model initially and then later using the back-annotation information from the floorplanning and place-and-route tools (see Section 64.10).

Cell-Library Binding

Cell-library binding is the process of transforming the set of Boolean equations or the Boolean network into a logic gate network with the gates in the cell library. Cell-library binding approaches are classified into two types: *rule-based* and *tree-based* approaches. Rule-based approaches iteratively replace parts of the logic network with equivalent cells from the cell library. This is done using local transformations which do not affect the behavior of the circuit. The tree-based approach does either *structural covering* and *matching* or *Boolean covering* and *matching*. In the structural approach, the logic network is expressed

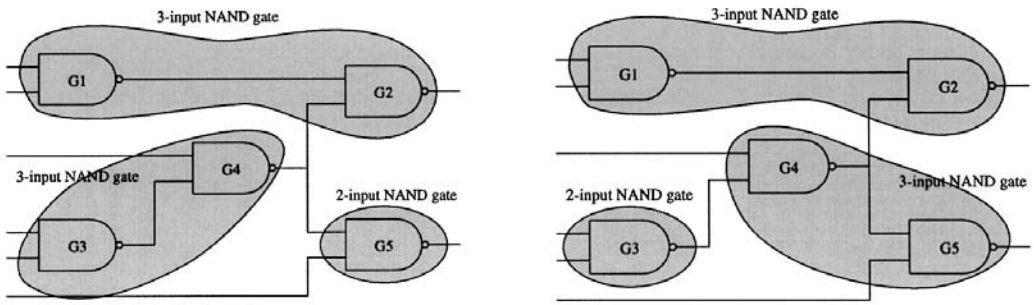


FIGURE 64.14 Two different network coverings for the same 2-input NAND logic subnetwork.

as an algebraic expression which is represented as a graph. Similarly, the cells in the library are also represented by graphs and the problem is reduced to one of subgraph matching and graph covering. The Boolean approach is similar but uses the matching of Boolean functions instead of graphs.

Tree-based matching is similar to pattern matching.[33] The cells in the library are represented as *pattern graphs* and then the aim is to find an optimal covering of the nodes in the logic network so as to optimize for the cost function (which may be area, power, etc.). This problem then reduces to a tree matching and covering problem which can be solved in linear time. One approach is to transform the logic network into a canonical form using only 2-input NAND gates and represent it as a logic graph. The cells in the library are also represented as pattern graphs in the canonical 2-input NAND gate format along with their area and delay costs. The pattern matching algorithm then attempts to find a cover of all the gates in the given logic graph using the cell-library pattern graphs so as to minimize the area and/or delay costs. An illustrative example is shown in Fig. 64.14. In this figure, two different network coverings are shown for the same logic subnetwork. Both these coverings use 3-input NAND gates from the cell library; however, a simple covering could have bound each node with a 2-input NAND gate.

Rule-based library binding techniques apply simple rules to identify circuit patterns and replace them with an equivalent pattern from the library. The cells from the library are characterized and rules derived from them. For example, a simple rule might replace two 2-input AND gates in series with a 3-input AND gate. More complex rules can even restructure a subnetwork of the given logic network so as to replace it with a more optimal subnetwork in terms of area and/or delay. Rule-based approaches are heuristic since the quality of results are affected to a great extent by the sequence in which the rules are applied. However, rule-based approaches allow complex transformations such as replacing nodes with high loads by high-drive cells or by inserting buffers. Also, rule-based approaches allow stepwise refinement and rebinding of cells to search for globally optimal results.

Static Timing Analysis

Timing analysis is required to verify the correctness and the timing performance of a circuit by ensuring that the timing constraints such as set-up and hold times of the flip-flops are met and the *critical paths*[3] in the circuit meet the timing budgets set for them. *Static timing analysis* exhaustively analyzes all the paths in the circuit netlist to check if they meet the timing requirements of the design. It computes the delay along the various paths and times all of them and determines the critical paths in the circuit.

The timing analysis is done using the gate delay, rise time, fall time, capacitance, and load values in the cell library to determine the delay of each gate and the interconnect delay. Delay across a gate (or any other node) depends on the delay through the gate, the loading on the gate, the number of fan-outs, and load due to the interconnect. The delay through a path (i.e., a chain of nodes) is also affected by the

[3]A critical path is a path in the circuit which has the maximum delay among all the paths in the circuit from its input to the output of the circuit.

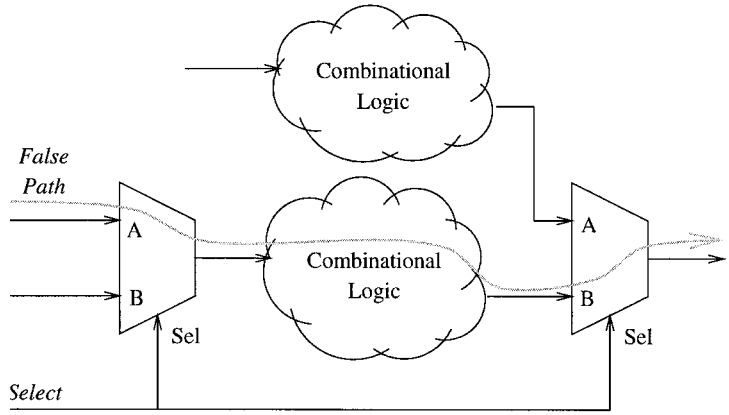


FIGURE 64.15 An example of a *false path* (i.e., a path which can never be activated).

skew or path delays due to the interconnect capacitances. In deep submicron designs, interconnect delays dominate over gate delays. For computing the path delays during static timing analysis, it is very important to have accurate estimates of the interconnect capacitances and wire-load model of the chip. Early floorplanning techniques are adopted to obtain these accurate estimates (see Section 64.10).

In this way, by timing all the paths in the circuit, the timing analyzer can determine all the critical paths in the circuit. However, the circuit may have *false paths*, which are paths in the circuit which are never exercised during normal circuit operation for any set of inputs. An example of a false path is shown in Fig. 64.15. The path going from the *A* input of the first multiplexor through the combinational logic out through the *B* input of the second multiplexor to the output is a false path. This path can never be activated since if the *A* input of the first multiplexor is activated, then the *Sel* line will also select the *A* input of the second multiplexor. Static timing analysis tools are able to identify simple false paths; however, they are not able to identify all the false paths and sometimes report false paths as the critical paths. For hard-to-detect false paths, the designer has to explicitly mark the known false paths as such before running the static timing analysis tool.

Circuit Emulation and Verification

Since testing and correcting a chip once it has been manufactured is a difficult and expensive task, it is essential to verify functional and timing characteristics of the design. As mentioned earlier in Section 64.2, FPGAs (field-programmable gate arrays) are increasingly being used for circuit prototyping and verification due to their ease of reconfigurability and programming. Once the netlist of the circuit design has been generated, it is used to program an FPGA-based circuit consisting of several FPGAs (depending on the size of the design).[32] Test patterns are then applied to this design to check its functionality in such a way, so as to exercise all the functions possible and all the inputs possible. The outputs of the emulation circuit are compared with the responses expected as per the functionality as described in the system specification. If design errors are found, the FPGA boards can easily be reprogrammed after the design has been fixed, and it is this ease of reconfigurability that makes FPGAs an attractive — albiet expensive — prototyping system.

64.10 Physical Design

The physical design process consists of specification of area and power of each block, floorplanning, placement, routing, and clock tree design.[34,35] The flow of the entire process is shown in Fig. 64.16, starting from logic synthesis to layout, parasitic extraction, and delay calculation. The physical design process starts during the logic synthesis process with the block circuit design, optimization and characterization steps, along with transistor resizing for taking care of loading and timing anamolies.

Floorplanning is a chip-level layout process where the layout cells, blocks, and inputs/outputs (I/Os) are placed on the chip to create a map of the location of the various blocks and devices. The layout program places the blocks on the chip by defining both their position and orientation, while leaving enough space between blocks for wires and interconnects. An initial floorplan is developed, sometimes as early as the initial architectural design of the system, to assess if the chip can meet its timing, performance, and cost goals. This is done by estimating the sizes of the blocks and the interconnect area. A preliminary floorplan is critical in accurately estimating the area budgets of each of the components, clock distribution requirements of the chip, the wire-load model of the design, and the interconnect resistances and capacitances. These estimates can be used to guide logic synthesis and the layout process. When there is no early floorplanning, an area-based wire-load model is adopted, based on the estimate of the die size of the final chip. However, in this method, the estimates of capacitances for global interconnects can be highly inaccurate.

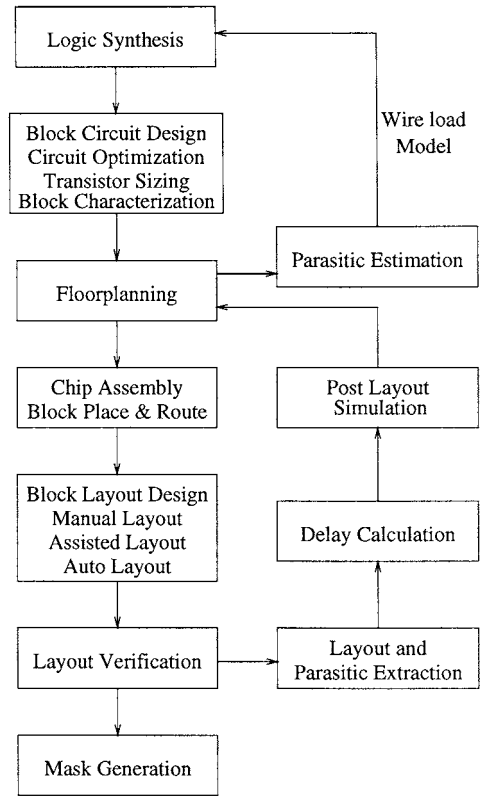


FIGURE 64.16 Physical design methodology.

Placement tools are used to optimally place the components or modules on the chip area. These tools take into account the size, aspect ratios, and pin positions of each component, so that the placement minimizes the area occupied by all the components. *Routing tools* then lay out or position the wires that connect the components so as to minimize the maximum, total, and average wire length. Routing on wafer can be done on multiple layers of metal, depending on the process technology being used. Usually, placement and routing tools make a lot of decisions that affect each other and are done iteratively or combined together in a single environment. Place-and-route tools are usually packaged with *layout tools*. These tools convert the logic-level design into the mask geometry of the targeted foundry using the techonology files of the foundry.

The *clock distribution* architecture of the chip is determined to a great extent by the area of the chip, placement of the blocks, target clock frequency, and the target library. As the size of chips increases, *clock skew* and other clock distribution delays become significant. A single clock can be distributed throughout the chip using a *balanced clock tree* with a low enough skew to prevent hold-time violations for flip-flops that directly drive other flip-flops. However, as the clock frequency and size of the chip increases, this approach leads to extremely large, high-power clock buffers, which are unacceptable. An alternative approach being used now, is to use a lower-speed bus to distribute the clock as a bused signal. Each major block in the chip synchronizes its local clock to the bus block, either by buffering the bus block or by using a phase-locked loop (PLL). The local bus can be at higher frequency which is a multiple of the bus clock.

Once the blocks have been placed and routed, the layout for each block is done either manually or with help of design automation tools. The layout is verified to check if the design works with the actual values of the parasitics of the interconnect on the chip and the clock distribution network. The parasitics are extracted, the delays along the interconnects are calculated, and the circuit is simulated. The results of the simulation are used to iterate over the entire physical design process as shown in Fig. 64.16.

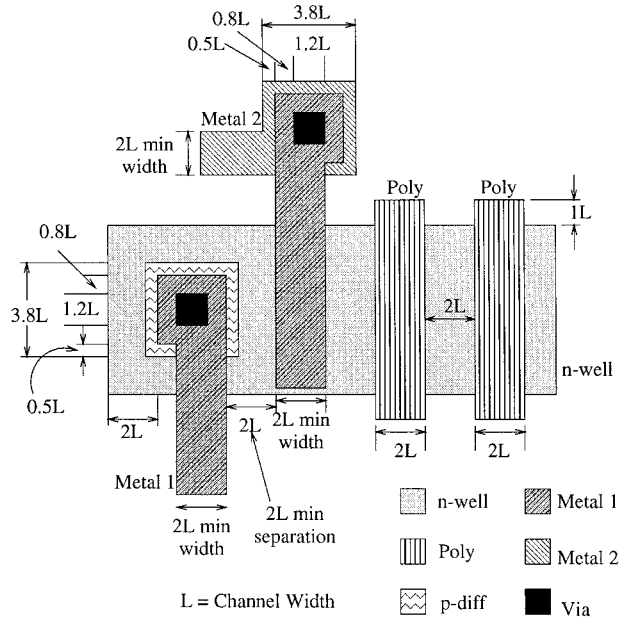


FIGURE 64.17 Illustrative example of layout design rules.

The final step in the physical design process is the *mask generation* phase. The masks are the geometric patterns that are used to etch the silicon by lithography. The output of design process is usually written out in *Caltech Intermediate Format* (CIF) or GDSII Stream. This is sent to the foundry, which manufactures the chip using the masks and runs its own design rule checks.

Layout Verification

The layout is verified using verification tools such as *design rule checkers* (DRC) and *extractors*. The DRC verifies that the geometric layout of the design does not violate the spacing and dimension rules of the foundry. In ensures that the mask layout has the minimum spacing and size required, and also verifies the spacings among the mask features. The extractor produces a netlist file, usually in SPICE format, after analyzing the connectivity of the design. The extracted SPICE file, which includes transistor sizes and parasitic capacitances, is used to run SPICE simulations on the circuit.[20]

Figure 64.17 demonstrates layout design rules. The numbers used in this figure are illustrative. The figure shows rules such as the minimum separation between two lines of metal-1 or polysilicon, the minimum overlap of polysilicon over the n-type (or p-type) subtrate, etc. These design rules are specified by the technology library provider (i.e., the foundry) and have to be obeyed while performing the layout. The DRC tools verify that the rules have been obeyed and flag errors if they have not. The design rules are necessary since violations can potentially lead to manufacturing faults in the chip. Layout and layout verification is discussed in detail in other chapters in this book.

64.11 I/O Architecture and Pad Design

Another important decision while developing the architecture of the chip is the *package* and *pin count* of the chip. The package type is determined by the area and heat generation of the chip. Packages are of various types such as plastic or ceramic, and each one has a different number of pins and different layout of pins in the chips.[36] Hence, the pin count is also determined at the same time as the package and is estimated during the initial architecture design.

Pads are the interface between the pins on the outside of the chip and the inputs and outputs in the digital circuits within the chip. Pads are usually distributed around the edge of the chip or, in recent packaging schemes, across the entire chip face. Each pad has an associated input or output circuitry which provides the necessary drive current required. Hence, each pad has V_{dd} and V_{ss} (i.e., positive and negative voltage) wires running through it. The number of pads and corresponding pins dedicated to V_{dd} and V_{ss} depends on how much current the chip draws and the power it consumes.

64.12 Tests after Manufacturing

There are several types of defects that can be introduced by the manufacturing process, such as stuck-at faults, delay faults, etc.[37] Hence, after the chip has been fabricated, it is tested extensively to find the faulty ones from the batch. By far one of the most expensive phases in the production of an integrated circuit, testing is done by applying test patterns to the unit being tested and comparing the unit's responses with the expected outputs for a working unit. *Automatic test pattern generation* (ATPG) tools use the description of the circuit to derive the sequence of the test vectors which exercise as many paths in the design as possible and test for the faults that may occur.[37]

Manufacturing tests aim at finding several different types of faults based on which they can be broadly classified into functional tests, diagnostic tests, and parametric tests.[43] *Functional tests* are simple tests which determine if a chip is functional or not and, hence, are also known as *go/no go* tests. *Diagnostic tests* are more involved since they aim at debugging the manufactured chip to determine which component in the chip has failed and possibly locate the fault within the component. This test is important to locate a manufacturing fault which is causing a large percentage of manufactured chips to fail. *Parameteric tests* check for clock skew, delay faults, noise margins, clock frequencies, etc. in the range of working conditions, such as supply voltage and temperature, for which the chip is supposed to function.

However, it is very difficult to create a set of test patterns that test for all the potential faults in the circuit. Recent developments have led to design methodologies which aim to improve the testability of the circuit while it is being designed. In this way, it is possible to design a circuit so that a set of test patterns can be generated which tests for all possible faults in the circuit. A detailed discussion on testing and testing methodologies is beyond the scope of this chapter.

64.13 High-Performance ASIC Design

The main optimization goal of ASIC chips is usually area. However, in a lot of mission-critical designs, speed is of foremost concern. Such high-performance designs require special design methodologies. A lot of design teams adopt a completely hand-crafted design methodology for these chips. However, it is recommended to use standard logic synthesis tools to make one pass over the design and the components in the chip, so as to at least get an estimate of the speed and area of the components. Since CAD tools are able to explore a much larger design space, they often can generate fairly optimal designs which come close to meeting the speed constraints of the design team. The design team can then take these components and hand-tune them to improve their speed. Common methods used are transistor resizing and transistor reodering.

Although most of the datapath blocks can be synthesized using standard cell libraries, there are always situations where a component is on the critical path. These critical blocks are typically completely hand-crafted. Alternatively, although most of the chip may be in CMOS technology, designers may choose faster technologies for the custom-crafted components and, hence, adopt a mixed technology methodology for the chip. *Dynamic* and dual-rail logic are popular as high-speed design styles, although their power consumption is much higher. In dynamic logic, all the nodes are precharged and typically require less number of transistors than static circuits and, hence, switch faster than CMOS circuits. However, these circuits are more power hungry since there is more switching activity and each node has to be precharged. Dual-rail logic has, as the name implies, two rails of signals, one being the complement of the other. The main disadvantage with this type of design is that it leads to reduced current drives,

especially at reduced voltages. However, recent technologies such as the *differential current switch logic* (DCSL) family have high-speed and low-power operations.[44]

Another factor often overlooked by designers is the fact that in most companies, technology libraries are designed so as to be optimum in terms of area (i.e., all the cells in the library have been hand-crafted so as to have the least area). However, there is always an area-speed tradeoff, and if a design is more speed critical and system architects are willing to throw some more area at the chip in order to improve speed, then the designers should request speed optimized technology libraries from the physical design team or foundry, as the case may be. This does not necessarily mean that all the cells in the library have to be redesigned to make them faster, but instead, only critical cells such as registers, full adders, or other components which are being used in components which are on the critical path, can be optimized.

64.14 Low Power Issues

The demand for portable semiconductor devices has fueled the need for more power-efficient semiconductor designs since the battery life on these portable devices is limited. This has led to the development of several power estimation and minimization design techniques. A considerable amount of this work is is focused on circuit-level power savings by modifying circuits and circuit design techniques to introduce low-power modes.[45-47] Several synthesis tools[23] also incorporate power estimation as part of their cost functions. In general, power management and savings have become a very important issue in IC design.

Power dissipation in CMOS circuits arises from switching or dynamic power due to the switching current, short-circuit current when both n-channel and p-channel transistors are momentarily on during switching, and leakage current during static operation. Of these, the main source of power consumption in CMOS gates is the switching current or dynamic power. The average power consumption of a CMOS gate due to the switching current is given by

$$P = \alpha C_L V_{dd}^2 f \tag{64.1}$$

where f is the system clock frequency, V_{dd} is the supply voltage, C_L is the load capacitance, and α is the switching activity (i.e., the probability of a $0 \rightarrow 1$ transition during a clock cycle). Some of the high-level strategies for reducing power consumption that can be deduced from this expression include:

- *Activity-based component shutdown:* Shut the component down during periods of inactivity by either shutting the clock ($f = 0$) or shutting the power supply ($V_{dd} = 0$). This can be done when it is known that a component will not be used in a clock cycle, by either gating the clock or gating the power supply or asserting a disable on the component's *enable* input (if any).
- *Supply voltage reduction:* Operate at the lowest possible supply voltage (since $P = \alpha V_{dd}^2$). Many chips which are embedded in portable devices adopt this methodology since the battery life of a portable device is limited. However, tradeoffs are made with other factors such as speed, noise margins, etc.
- *Switching activity reduction:* Architectural changes to restructure the computation, communication, or memory for example to reduce the switching activity, α. By far, this has been the area of most research which has led to methods for achieving fewer transitions, especially on interconnect and memory.

Recent work on system-level power shutdown and use of low-power modes, has shown that significant savings can be achieved by considering high-level system inactivity and usage information.[48-50]

64.15 Reuse of Semiconductor Blocks

In the past few years, the reuse of semiconductor functional blocks has become popular. High-level functional blocks such as signal-processing functions, input/output interface devices, audio/video compression and decompression functions etc., are being designed once and reused in several designs. These blocks are

also known *cores* and several companies specializing in developing these cores are selling them as intellectual property (IP).[51] These cores are designed with clear well-defined and well-documented interfaces so that they can be integrated into system designs easily. The resulting *system-on-a-chip* (SOC) uses several of these cores and sometimes a microprocessor core to implement a complex system targeted at, say, multimedia processing. This is akin to the use of software component libraries in software design.

This core reuse methodology has created a new set of challenges for ASIC design.[28,52] Frequently, while integrating the cores, a significant amount of "glue logic" is required to tie in the varied integration requirements of the cores. This glue logic effects system verification detrimentally, since the cores have to be tested and verified with the glue logic. Testing a chip with several cores is an open research problem. A methodology has to be developed that allows core access and isolation during scan-based testing.

The industry is moving towards defining modular design styles and standard interface templates for cores, so that they can easily be plugged-in to a system and parameterizable features can be included or deleted depending on the design requirements. Bus and interconnect standards are also being developed, which will allow minimal glue logic to incorporate cores. New core test strategies are being developed to facilitate test and verification of cores and their interaction with other cores in the system.

This system-on-a-chip technology is driving the next step in the evolution of semiconductor design and development of CAD tools. Design teams are re-learning the way designs are conceived and created, so as to allow reuse. The bus interface standardization efforts will eliminate glue logic and hence, the performance overheads due to glue logic. These standardizations will allow the development of CAD tools which will make the use of cores as easy as a standard cell library and core integration tools as interactive as circuit schematic tools of today.

64.16 Conclusion

As advances in semiconductor technology continue to provide the ability to put more on silicon with increasing circuit densities and performance, the ASIC design methodology is evolving to higher levels of system specification and an increasing use of CAD tools to automate the design process. Increasing complexity has also led to the proliferation of language-based approaches for digital design. More recently, programming languages are being used for system design due to their ability to quickly model and simulate digital system designs and the familiarity they enjoy with designers.[53] The use of high-level programming languages for hardware modeling also helps in the semiconductor block reuse methodology. At a lower level of abstraction, logic synthesis tools have matured to the extent that they are indispensible for large, complex designs. The linking of the physical design and logic synthesis is becoming important and popular since the effectiveness and accuracy of logic synthesis is impacted to a great extent by the feedback and parasitic information provided by floorplanning tools.

Behavioral synthesis methodologies are fast becoming available which allow the synthesis of high-level functional descriptions of systems in C-based languages. These tools attempt to raise the abstraction level and design entry level close to the conceptualization level. These high-level synthesis tools allow a more complete and efficient exploration of the design space which cannot be done effectively manually. They remove the onus from "experienced" system designers to tried and proven methodologies.

Additionally, the ever-increasing demands for semiconductor devices in all aspects of everyday life is fueling the development of better and faster design turn-around tools and methodologies. Logic design productivity is increasing due to the availability of new tools and methodologies such as emulators and prototyping environments, cycle simulators, hardware accelerators, formal verification tools, system-on-a-chip methodologies etc. The need for devices which are portable is prompting more power efficient design and power estimation methodologies. Increasingly complex interactions between physical aspects and higher levels of the design are causing a tighter integration of the various levels of design from high-level synthesis to logic design to physical design. Finally, better development styles are being adopted which allow fast prototyping of a system and involve more interaction between the various design teams working on different levels of the design.

References

1. G. De Micheli, *Synthesis and Optimization of Digital Circuits,* McGraw-Hill, New York, 1994.
2. Synopsys Module Compiler, http://www.synopsys.com/products/datapath/datapath.html.
3. A. Chowdhary, S. Kale, P. Saripella, N.K. Sehgal, and R.K. Gupta, A general approach for regularity extraction in datapath circuits, *International Conference on Computer-Aided Design,* 1998.
4. D. Gajski, S. Narayan, L. Ramachandran, F. Vahid, and P. Fung, System design methodologies: aiming at the 100 h design cycle, *IEEE Transactions on (VLSI) Systems,* vol. 4, no. 1, March 1996.
5. S. Devadas, A. Ghosh, and K. Keutzer, *Logic Synthesis,* McGraw-Hill, New York, 1994.
6. C.H. Roth Jr., *Digital Systems Design Using VHDL,* PWS Publishing, 1998.
7. R.H. Katz, *Contemporary Logic Design,* Benjamin/Cummings Publishing, 1994.
8. G.D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms,* Kluwer Academic Publishers, 1996.
9. E.J. McCluskey, *Logic Design Principles,* Prentice-Hall, Englewood Cliffs, NJ, 1986.
10. D.D. Gajski and R.H. Kuhn, Guest editor's Introduction: New VLSI tools, *IEEE Computer,* Dec. 1983.
11. A. Jantsch, A. Hemani, and S. Kumar, *The Rugby Model: A Conceptual Frame for the Study of Modeling, Analysis and Synthesis Concepts of Electronic Systems,* Design, Automation and Test in Europe, 1999.
12. IEEE Standard, VHDL Language Reference Manual, 1988.
13. D. Thomas and P. Moorby, *The Verilog Hardware Description Language,* Kluwer-Academic, 1991.
14. D. Ku and G. De Micheli, HardwareC — a language for hardware design, Stanford Univ. Tech. Rep. CSL-TR-90-419, 1988.
15. D. Harel, Statecharts: A visual formalism for complex systems, *Sci. Comput. Programming,* 8, 1987.
16. P. Hilfinger and J. Rabaey, *Anatomy of a Silicon Compiler,* Kluwer Academic Publishers, 1992.
17. N. Halbwachs, *Synchronous Programming of Reactive Systems,* Kluwer Academic Publishers, 1993.
18. F. Vahid, S. Narayan, and D.D. Gajski, SpecCharts: A VHDL frontend for embedded systems, *IEEE Trans. Computer-Aided Design,* vol. 14, pp. 694-706, 1995.
19. N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective,* Addison Wesley, 1994.
20. L.W. Nagel, SPICE2: a computer program to simulate semiconductor circuits, Memo ERL-M520, Dept. Electrical Engineering and Computer Science, University of California, Berkeley, 1975.
21. C. Terman, Timing simulation for large digital MOS circuits, *Advances in Computer-Aided Engineering Design,* vol. 1, JAI Press, 1984.
22. Z. Navabi, *VHDL: Analysis and Modeling of Digital Systems,* McGraw-Hill, New York, 1993.
23. Synopsys Design Compiler, http://www.synopsys.com/products/logic/logic.html.
24. D.D. Gajski and L. Ramachandran, Introduction to high-level synthesis, *IEEE Design Test Comput.,* winter 1994.
25. D.D. Gajski, *Principles of Digital Design,* Prentice-Hall, Englewood Cliffs, NJ, 1997.
26. S. Malik, *private communication.*
27. Synopsys Behavioral Compiler, http://www.synopsys.com/products/beh_syn/beh_syn.html.
28. M. Keating and P. Bricaud, *Reuse Methodology Manual for System-On-a-Chip Designs,* Kluwer Academic Publishers, 1998.
29. R. Camposano and W. Wolf, *High Level VLSI Synthesis,* Kluwer Academic, 1991.
30. C.P. Ravikumar, S. Gupta, and A. Jajoo, Synthesis of testable RTL designs using adaptive simulated annealing algorithm, *Eleventh International Conference on VLSI Design,* 1998, India.
31. D.D. Gajski, N.D. Dutt, C.-H. Wu Allen, and Steve Y.-L. Lin, *High-Level Synthesis: Introduction to Chip and System Design,* Kluwer Academic Publishers, 1992.
32. Quickturn Emulation Tools, http://www.quickturn.com/.
33. K. Keutzer, DAGON: Technology Binding and Local Optimization by DAG Matching, *Proceedings of the Design Automation Conference,* 1987.

34. B. Preas and M. Lorenzetti, *Physical Design Automation of VLSI Systems,* Benjamin Cummings Publishing, 1988.
35. S.M. Sait and H. Youssef, *VLSI Physical Design Automation,* IEEE Press, 1995.
36. W. Wolf, *Modern VLSI Design: Systems on Silicon,* Prentice-Hall, Englewood Cliffs, NJ, 1998.
37. M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design,* Computer Science Press, 1990.
38. S.-P. Lin, C. Njinda, and M. Breuer, Generating a family of testable designs using the BILBO methodology, *Journal of Electronic Testing: Theory and Applications,* pp. 71-89, 1993.
39. L. Avra, *Allocation and Assignment in High-Level Synthesis for Self-Testable Data Paths,* Proceedings of International Test Conference, pp. 463–472, 1991.
40. V.D. Agrawal, C.R. Kime, and K.K. Saluja, A tutorial on built-in self-test, Part 1. Principles, Part 2. Applications, *IEEE Design & Test of Computers,* 10, March/June 1993.
41. R.K. Brayton, C. McMullen, G.D. Hachtel, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis,* Kluwer Academic Publishers, 1984.
42. R.K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, MIS: a multiple-level logic optimization system, *IEEE Transactions on CAD/ICAS,* CAD-6, Nov. 1987.
43. J.M. Rabaey, *Digital Integrated Circuits: A Design Perspective,* Prentice-Hall, Englewood Cliffs, NJ, 1996.
44. D. Somasekhar and K. Roy, Differential current switch logic: a low power DCVS logic family, *European Solid-State Circuits Conference,* 1995.
45. F.N. Najm, A survey of power estimation techniques in VLSI circuits, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* Dec. 1994.
46. M. Pedram, Power Minimization in IC Design: Principles and Applications, *ACM Transactions on Design Automation of Electronic Systems,* Jan. 1996.
47. L. Benini and G. De Micheli, *Dynamic Power Management: Design Techniques and CAD Tools,* Kluwer Academic Publishers, 1997.
48. M.B. Srivastava, A.P. Chandrakasan, and R.W. Broderson, Predictive system shutdown and other architectural techniques for energy efficient programmable computation, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* Mar. 1996.
49. G.A. Paleologo, L. Benini, A. Bogliolo, and G. De Micheli, Policy optimization for dynamic power management, *Proc. of 35th Design Automation Conference,* June 1998.
50. D. Ramanathan, S. Irani, and R.K. Gupta, Online power management algorithms for embedded systems, submitted for publication.
51. Y. Zorian and R.K. Gupta, Introduction to core-based design, *IEEE Design and Test of Computers,* Oct. 1997.
52. J.J. Engel et al., Design methodology for IBM ASIC products, *IBM Journal of Research and Development,* 40, (no. 4), IBM, July 1996.
53. R.K. Gupta and S.Y. Liao, Using a programming language for digital system design, *IEEE Design and Test of Computers,* Apr. 1997.

Lockwood, J. "Logic Synthesis for Field Programmable Gate Array (FPGA) Technology"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

65

Logic Synthesis for Field Programmable Gate Array (FPGA) Technology

John W. Lockwood
Washington University

65.1 Introduction

Field Programmable Gate Arrays (FPGAs) enable rapid development and implementation of complex digital circuits. FPGA devices can be reprogrammed and reused, allowing the same hardware to be employed for entirely new designs or for new iterations of the same design. While much of traditional IC logic synthesis methods apply, FPGA circuits have special requirements that affect synthesis.

The FPGA device consists of a number of configurable logic blocks (CLBs), interconnected by a routing matrix. Pass transistors are used in the routing matrix to connect segments of metal lines. There are three major types of CLBs: those based on PLAs, those based on multiplexers, and those based on table look-up (TLU) functions.

Automated logic synthesis tools are used to optimize the mapping of the Boolean network to the FPGA device. FPGA synthesis is an extension to the general problem of multi-level logic synthesis. FPGA logic synthesis is usually solved in two phases. The *technology-independent* phase uses a general multi-level logic optimization tool (such as Berkeley's MIS) to reduce the complexity of the Boolean network. Next, a *technology-dependent* optimization phase is used to optimize the logic for the particular type of device. In the case of the TLU-based FPGA, each CLB can implement an arbitrary logic function of a limited

number of variables. FPGA optimization algorithms aim to minimize: the number of CLBs used, the logic depth, and the routing density.

The *Chortle* algorithm is a direct method that uses dynamic programming to map the logic into TLU-based CLBs. It converts the Boolean network into a forest of directed acyclic graphs (DAGs); then it evaluates and records the optimal subsolutions to the logic mapping problem as it traverses the DAG. The *two-step* algorithms operate by first *decomposing* the nodes, and then performing a *node elimination*. Later sections of this chapter discuss and detail the *Xmap, Hydra*, and *MIS-pga* algorithms.

FPGA devices are fabricated using the same sub-micron geometries as other silicon devices. As such, the devices benefit from the rapid advances in device-technology. The overhead of the programming bits, general function generators, and general routing structures, however, reduce the total amount of logic available to the end user.

65.2 FPGA Structures

An FPGA consists of reconfigurable logic elements, flip-flops, and a reprogrammable interconnect structure. The logic elements are typically arranged in a matrix. The interconnect is arranged as a mesh of variable-length metal wires and pass transistors to interconnect the logic elements. The logic elements are programmed by downloading binary control information from an external ROM, a build-in EPROM, or a host processor. After download, the control information is stored on the device and used to determine the function of the logic elements and the state of the pass transistors. Unlike a PLA, the FPGA can be used for multi-level logic functions.

The *granularity* of an FPGA refers to the complexity of the individual logic elements. A *fine-grain* logic block appear to the user to be much like a standard mask-programmable gate array. Each logic block consists of only a few transistors, and is limited to implementing only simple functions of a few variables. A *course-grain* logic block (such as those from Xilinx, Actel, Quicklogic, and Altera) provides more general functions of a larger number of variables. Each Xilinx 4000-series logic block, for example, can implement any Boolean function of five variables, or two Boolean functions of four variables.

It has been found that the course-grain logic blocks generally provide better performance than the fine-grain logic blocks, as the course-grained devices require less space for interconnect and routing by combining multiple logic functions into one logic block. In particular, it has been shown that a four-input logic block uses the minimal chip area for a large variety of benchmark circuits.[1] The expense of a few extra underutilized logic blocks outweighs the area required for the larger number of fine-grained logic blocks and their associated larger interconnect matrix and pass transistors. This paper will focus on the logic synthesis for course-grained logic elements.

A course-grained configurable logic block (CLB) can be implemented using: a PLA-based AND/OR elements, multiplexors, or SRAM-based table look-up (LUT) elements. These configurations are described below in detail.

Look-up Table (LUT)-Based CLB

The basic unit of look-up table (LUT)-based FPGAs is the *configurable logic block* (CLB), implemented as an SRAM of size $2^n \times 1$. Each CLB can implement any arbitrary logic function of n variables, for a total of 2^n functions.

An example of an LUT-based FPGA is the Xilinx 4000-series FPGA, as illustrated in Fig. 65.1. Each CLB has three LUT generators, and two flip-flops.[2] The first two LUTs implement any function of four variables, while the third LUT implements any function of three variables. Separately, each CLB can implement two functions of four variables. Combined, each CLB can implement any one function of five variables, or some restricted functions of nine variables (such as AND, OR, XOR).

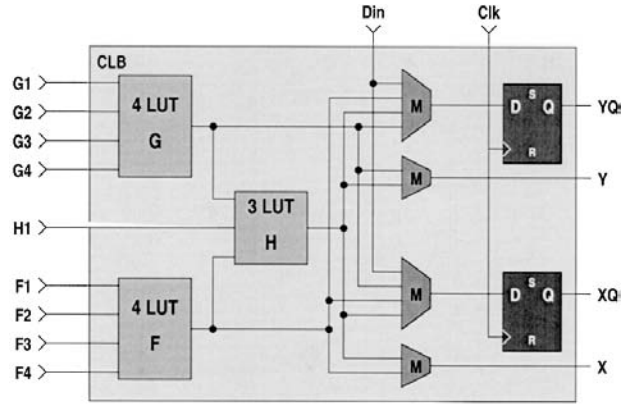


FIGURE 65.1 Xilinx 4000-series CLB.

PLA-Based CLB

PLA-based FPGA devices evolved from the traditional PLDs. Each basic logic block is an AND-OR block consisting of wide fan-in AND gates feeding a few-input OR gate. The advantage of this structure is that many logic functions can be implemented using only a few levels of logic, due of the large number of literals that can be used at each block. It is, however, difficult to make efficient use of all inputs to all gates. Even so, the amount of wasted area is minimized by the high packing density of the wired-AND gates.

To further improve the density, another type of logic block, called the *logic expander*, has been introduced. It is a wide-input NAND gate whose output could be connected to the input of the AND-OR block. While its delay is similar, the NAND block uses less area than the AND-OR block, and thus increases the effective number of product terms available to a logic block.

Multiplexer-Based CLB

Multiplexer-based FPGAs utilize a multiplexer to implement different logic function by connecting each input to a constant or a signal.[3] The ACT-1 logic block, for example, has three multiplexers and one logic gate. Each block has eight inputs and one output, implementing:

$$f = \left(\overline{s_3 + s_4}\right)\left(\overline{s}_1 w + s_1 x\right) + \left(s_3 + s_4\right)\left(\overline{s}_2 y + s_2 x\right)$$

Multiplexer-based FPGAs can provide a large degree of functionality for a relatively small number of transistors. Multiplexer-based CLBs, however, place high demands on routing resources due to the large number of inputs.

Interconnect

In all structures, a reprogrammable routing matrix interconnects the configurable logic blocks. A portion of the routing matrix for the Xilinx 4000-series FPGA, for example, is illustrated in Fig. 65.2. Local interconnects are used to join adjacent CLBs. Global routing modules are used to route signals across the chip.

The routing and placement issues for the FPGAs are somewhat different from those of custom logic. For a large fan-out node, for example, an optimal placement for the elements for the fan-out

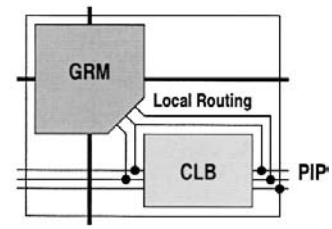


FIGURE 65.2 Xilinx routing matrix.

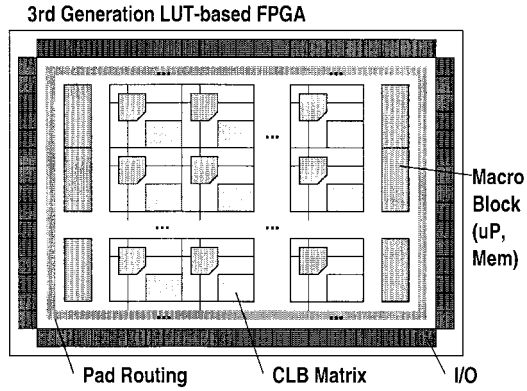


FIGURE 65.3 FPGA chip layout.

would be along a single row or column, where the routing could be done using a long line. For custom logic, the optimal placement would be as a cluster, where the optimization attempted to minimize the distance between nodes. For the FPGA, the routing delay is more influenced by the number of pass transistors for which the signal must cross rather than by the length of the signal line.

The power of the FPGA comes from the flexibility of the interconnect. A block diagram of a typical third-generation FPGA device is shown in Fig. 65.3. The CLB matrix and the mesh of the interconnect occupy most of the chip real area. Macro blocks, when present, implement functions such as high-density memory or microprocessing cores. The I/O blocks surround the chip and provide connectivity to external devices.

65.3 Logic Synthesis

Logic synthesis is typically implemented as a two-phase process: a technology-independent phase, followed by a technology mapping phase.[4] The first phase attempts to generate an optimized *abstract representation* of the target circuit, and the second phase determines the optimal mapping of the optimized abstract representation onto a particular type of device, such as an FPGA. The second phase optimization may drastically alter the circuit to optimize the logic for a particular technology. In most approaches published, the technology-dependent FPGA optimization is based on the area occupied by the logic as measured by the number of LUTs.

The abstract representation of a combination logic function f is not unique. For example, f may be expressed by a truth table, a *sum-of-products* (SOP) (such as $f = ab + cd + e'$), a *factored form* (such as $f = (a + b)(c + (e'(f + g')))$), a *binary decision diagram* (BDD) *directed acyclic graph* DAG), an *if-then-else DAG*, or any combination of the above forms.

The BDD is a DAG where the logic function is associated with each node, as shown in Fig. 65.4. It is canonical because, for a given function and a given order of the variables along all the paths, the BDD DAG is unique. A BDD may contain a great deal of redundant information, however, as the sub-functions may be replicated in the lower portions of the tree.

The *if-then-else DAG* consists of a set of nodes, each with three children. Each node is a two-to-one selector, where the first child is connected to the control input of the selector and the other two are connected to the signal inputs of the node.

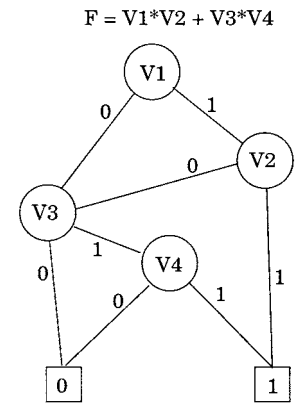


FIGURE 65.4 Binary decision diagram.

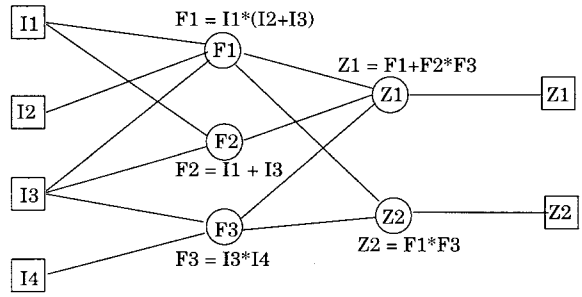


FIGURE 65.5 An example of Boolean network.

Technology-Independent Optimization

In the technology-independent synthesis phase, the combinational logic function is represented by the Boolean network, as illustrated in Fig. 65.5. The nodes of the network are initially *general nodes*, which can represent any arbitrary logic function. During optimization, these nodes are usually mapped from the *general form* to a *generic form*, which only consists of AND, OR, and NOT logic nodes.[4] At the end of first synthesis phase, the complexity and number of nodes of the Boolean network has been reduced.

Two classes of operations — *network restructuring* and *node minimization* — are used to optimize the network. Network restructuring operations modify the structure of the Boolean network by introducing new nodes, eliminating others, and adding and removing arcs. Node minimization simplifies the logic equations associated with nodes.[5]

Restructuring Operations

Decomposition reduces the *support* of the function, F, (denoted as $sup(F)$). The support of the function refers to the set of variables that F explicitly depends on. The *cardinality* of a function (denoted by $|sup(F)|$), represents the number of variables that F explicitly depends on.

Factoring is used to transform the SOP form of a logic function into a factored form. *Substitution* expresses one given logic function in terms of another. *Elimination* merges a subfunction, G, into the function, F, so that F is expressed only in terms of its fan-in nodes of F and G (not in terms of G itself).

The efficiency of the restructuring operations depends on finding a suitable *divisor*, P, to factor the function, that is, given functions F, choose a divisor P, and find the functions Q and R such that $F = PQ+R$. The number of possible divisors is hopelessly large; thus, an effective procedure is required to restrict the searching subspace for good divisors. The Brayton and McMullen kernel matching technique is used.

The *kernels* of a function F are the set of expressions: $K(F) = \{g \mid g \subset D(F)$, where g is cube-free, and $D(F)$ are the primary divisors.

A *cube* is a logic function given by the product of literals. A *cube of a function F* is a cube whose *on-set* does not have vertices in the *off-set* of F (e.g., if $F = ab(c + d)$, ab is a cube of F). An expression F is *cube-free* if no cube divides the expression evenly.[6] For example, $F = ab + c$ is cube-free, while $F = ab + ac$ is not cube-free. Finally, the *primary divisors* of F are the set of expression: $D(F) = F/C \mid C$ is a cube.[7]

Kernel functions can be computed effectively by several fast algorithms. Based on the kernel functions extracted, the restructuring operations can generate acceptable results usually within a reasonable amount of time.[4] Speed/quality tradeoffs are still needed, however, as is the case with MIS, which is a multi-level logic synthesis system.[8]

Node Minimization

Node minimization attempts to reduce the complexity of a given network by using Boolean minimization techniques on its nodes.

A two-level logic minimization with consideration of the *don't-care* inputs and outputs can be used to minimize the nodes in the circuit. Two types of *don't-care* sets — satisfiability don't care (SDC) and observability don't care (ODC) — are used in the two-level minimizer. The SCD set represents combinations of input variables that can never occur because of the structure of the network itself, while the ODC set represents combinations of variables that will never be observed at outputs. If the ODCs and SDCs are too large, a practical running time can only be achieved by using a limited subset of ODCs and SDCs.[8]

Another technique is to use a tautology checker to determine if two Boolean networks are equivalent, by taking XNOR of their corresponding primary outputs.[9] A node is first tentatively simplified by deleting either variables or cubes. If the result of tautology check is 1 (equivalent), then this deletion is performed. As with the first method, an exhaustive search is usually not possible because of the computational cost of the tautology check.

Technology Mapping

Taking the special characteristics of a particular FPGA device into account, the technology mapping phase attempts to realize the Boolean network using a minimal number of CLBs. Synthesis algorithms fall into two main categories: *algorithmic approaches* and *rule-based techniques.*

By expressing the optimized AND/OR/NOT network as a *subject graph* (a network of two-input NAND gates), and a library of potential mappings as a *pattern graphs,* the first approach converts the mapping problem to a covering problem with the goal of finding the minimum-cost cover of the subject graph by the pattern graphs. The problem is NP-hard; thus, heuristics must be used. If the network to be mapped is a tree, an optimal heuristic method has been found. It is inspired by Aho et al.'s work on optimizing compilers. If the Boolean network is not a tree, a step of decomposition into forest of trees is performed; then the mapping problem is solved as a tree-covering-by-tree problem, using the proven optimal heuristic.

The rule-based technique traverses the Boolean network and replaces subnetworks with patterns in the library when a match is found. It is slow compared to the first method, but can generate better results. Mixed approaches, which a perform tree-covering step followed by a rule-based clean-up step, are the current trend in industry.

65.4 Look-up Table (LUT) Synthesis

The existing approaches to synthesize FPGAs based on look-up tables (LUTs) are summarized in Fig. 65.6. Beginning with an optimized AND/OR/NOT Boolean network generated by a general-purpose multi-level logic minimizer, such as MIS-II, these algorithms attempt to minimize the number of LUTs needed to realize the logic network.

Library-Based Mapping

Library-based algorithms were originally developed for use in the synthesis of standard cell designs. It was assumed that there was a small number of pre-designed logic elements. The goal of the mapping function was to optimize the use of these blocks.

MIS is one such library-based approach that performs multi-level logic minimization. It existed long before the conception of FPGAs and has been used for TLU logic synthesis. Non-equivalent functions in MIS are explicitly described in terms of two-input NAND gates. Therefore, an optimal library needs to cover all functions that can be implemented by the TLU. Library-based algorithms are generally not appropriate for TLU-based FPGAs due to their large number of functions which each CLB can implement.

Direct Approaches

Direct approaches generate the optimized Boolean network directly, without the explicit construction of library components. Two classes of method are used currently: *modified tree covering algorithms* (i.e., *Chortle* and its improved versions) and *two-step methods.*

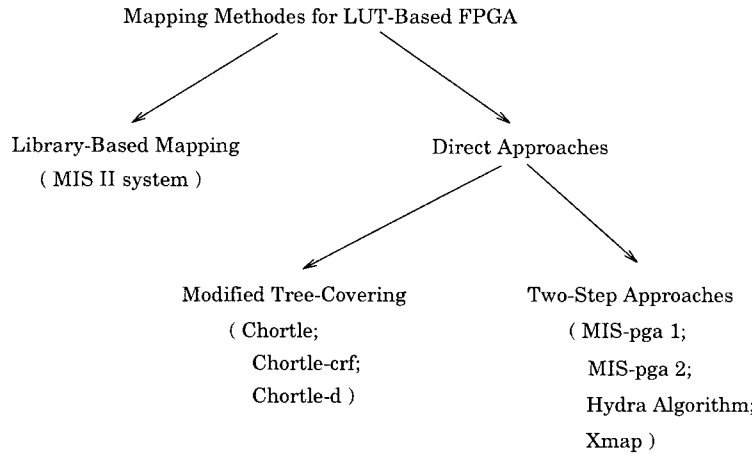


FIGURE 65.6 Approaches to synthesize FPGAs based on LUTs.

Modified Tree-Covering Approaches

The modified tree-covering approach begins with an AND/OR representation of the optimized Boolean network. *Chortle*, and its extensions (*Chortle-crf* and *Chortle-d*), first decompose the network into a forest of trees by clipping the multiple-fan-out nodes. An optimal mapping of each tree into LUTs is then performed using dynamic programming, and the results are assembled together according to the interconnection patterns of the forest. The details of the Chortle algorithms are given in the Section 65.5.

Two-step Approaches

Instead of processing the mapping in one direct step, the *two-step methods* handle the mapping by *node decompostion* followed by *node elimination*. The decomposition operation yields a network that is *feasible*. The node elimination step reduces the number of nodes by combining nodes based on the particular structure of a CLB.

A Boolean network is feasible if every intermediate node is realized by a feasible function. A *feasible function* is a function that satisfies $|sup(f)| \leq K$, or informally, can be realized by one CLB.

Different two-step approaches have been proposed and implemented, including MIS-pga1 and MIS-pga2 from U.C. Berkeley, Xmap from U.C. Santa Cruz, and Hydra from Stanford. Each algorithm has its own advantages and drawbacks. Details of these methods are given in Section 65.6. Comparisons among the direct and two-step methods are given in Section 65.7.

65.5 Chortle

The Chortle algorithm is specifically designed for TLU-based FPGAs. The input to the Chortle algorithm is an optimized AND/OR/NOT Boolean network. Internally, the circuit is represented as a forest of directed acyclic graphs (DAGs), with the leaves representing the inputs and the root representing the output, as shown in Fig. 65.7. The internal nodes represent the logic functions AND/OR. Edges represent inverting or non-inverting signal paths.

The goal of the algorithm is to implement the circuit using the fewest number of K-input CLBs in minimal running time. Efficient running time is a key advantage of Chortle, as FPGA mapping is a computationally intensive operation in the FPGA synthesis procedure.

The terminology of the Chortle algorithm defines the *mapping* of a node, n, in a tree as the circuit of look-up tables rooted at that node that extends to the leaf nodes. The *root look-up table* of node n is the mapping of the Boolean function that has the node n as its single output. The *utilization* of a look-up table refers to the number of inputs, U, out of the K inputs actually used in the mapping. Finally, the

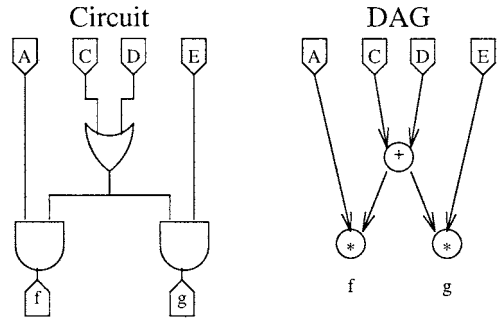


FIGURE 65.7 Boolean network and DAG representation.

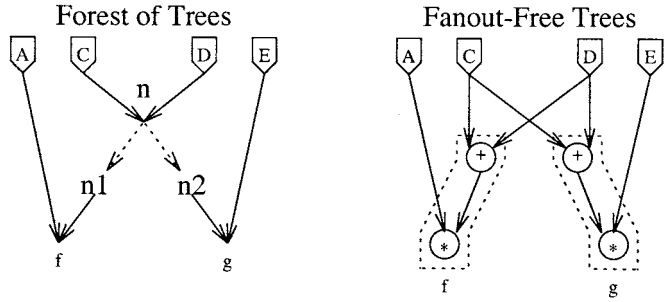


FIGURE 65.8 Forest of fan-out-free trees.

utilization division, μ, is a vector that denotes the distribution of the inputs to the root look-up table among subtrees. For example, a utilization vector of $\mu = \{2,1\}$ would refer to a table look-up function that has two of the K inputs from the left logic subtree, and one input from the right subtree.

Tree Mapping Algorithm

The first step of the Chortle algorithm is to convert the input graph to forest of fan-out-free trees, where each logic function has exactly one output. As illustrated in Fig. 65.8, node n has a fan-out degree of two; thus, two new nodes, n_1 and n_2, are created that implement the same Boolean equation of node n. Each subtree is then evaluated independently.

Chortle uses a *postorder traversal* of each DAG to determine the mapping of each node. The logic functions connecting the inputs (leaves) are processed first; the logic functions connecting those functions are processed next, and so on until reaching the output node (root).

Chortle's tree mapping algorithm is based on dynamic programming. Chortle *computes* and *records* the solution to all subproblems, proceeding from the smallest to the largest subproblem, avoiding recomputation of the smaller subproblems. The subproblem refers to computation of the minimum-cost mapping function of the node n in the tree. For each node n_i, the subproblem, $minMap(n_i,U)$, is solved for each value of U, ranging from 2 … K ($U = K$ refers to a look-up function that is fully utilized, while $U = 2$ refers to a TLU with only two inputs).

In general, for the same value of U, multiple utilization vectors, $\mu(u_1, u_2, \ldots, u_f)$, are possible, such that $\sum_{i=1}^{f} u_i = U$. The utilization vector determines how many inputs are to be used from each of the previous optimal subsolutions. Chortle examines each possible mapping function to determine this node's minimum-cost mapping function, $cost(minMap(n,U))$. For each value of $U \in \{2 \ldots K\}$, the utilization division of the minimum-cost mapping function is recorded.[10]

Example

The Chortle mapping function is best illustrated by an example, as illustrated in Fig. 65.9. For this example, we will assume that each CLB may have as many as four inputs (i.e., $K = 4$). The inputs, {A,B,C,D,E,F}, perform the logic function: A * B + (C * D) E + F.

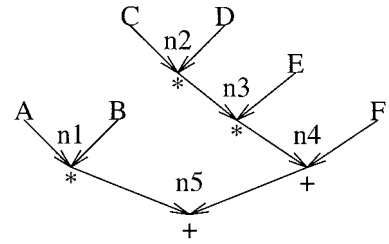


FIGURE 65.9 Chortle mapping example.

In the postorder traversal n_1 is visited first, followed by n_2 … n_5. For n_1, there is only one possible mapping function, namely, $U = 2$, $\mu = \{1,1\}$. The same is true for n_2.

When n_3 is evaluated, there are two possibilities, as illustrated in Fig. 65.10. First, the function could be implemented as a new CLB with two inputs ($U = 2$), driven from the outputs of n_2 and E. This sub-graph would use two TLBs; thus, it would have a cost function of 2. For $U = 3$, only one utilization vector is possible, namely, $\mu = \{2,1\}$. All three primary inputs C, D, and E are grouped into one CLB, thus producing a cost function of 1. We store only the utilization vectors and cost functions for $minMax(n_3, 2)$ and $minMax(n_3,3)$.

When n_4 is evaluated, there are many possibilities, as illustrated in Fig. 65.11. With $U = 2$ ($\mu = \{1,1\}$), a two-input CLB would combine the optimal result for n_3 with the primary input F, producing a function with a cost of 2. For $U = 3$ ($\mu = \{2,1\}$), a three-input CLB would combine the optimal result for n_3: $U = 2$ with both inputs E and F, also at a cost of two CLBs. Finally, for $U = 4$, a single CLB would implement the function (C * D) * E + F), at a cost of 1. We store the utilization vectors and cost functions for $minMax(n_4,2)$, $minMax(n_4,3)$, and $minMax(n_4,4)$.

Finally, we evaluate the output node, n_5, as illustrated in Fig. 65.12. We see that there are four possible mappings and, of those, two minimal mappings are possible. Chortle may return either of the mappings where two CLBs implement: $n_5 = (A * B) + n_3 + F$ and $n_3 = (C * D) * E$.

Chortle-crf

The Chortle-crf algorithm is an improvement of the original Chortle algorithm. The major innovation with Chortle-crf involves the method for choosing gate-level node decomposition. The other improvements involve the algorithm's response to reconvergent and replicated logic. The name, Chortle-crf, is based on the new command line options (`-crf`) that may be given when running the program (`-c` for constructive bin-packing for decomposition, `-r` for reconvergent optimization, and `-f` for replication optimization).[11] Each of the optimizations are detailed below.

Decomposition

Decomposition involves splitting a node and introducing intermediate nodes. Decomposition is *required* if the original circuit has a fan-in greater than K. In this case, no one CLB could implement the entire

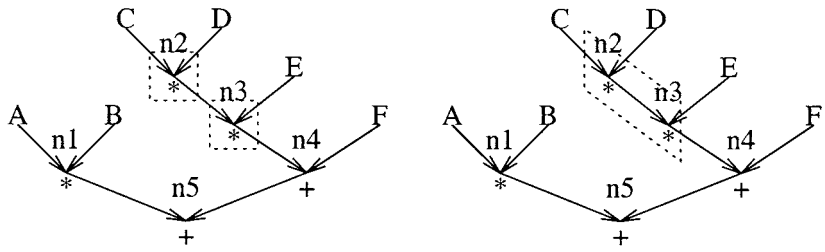


FIGURE 65.10 Mapping of node 3.

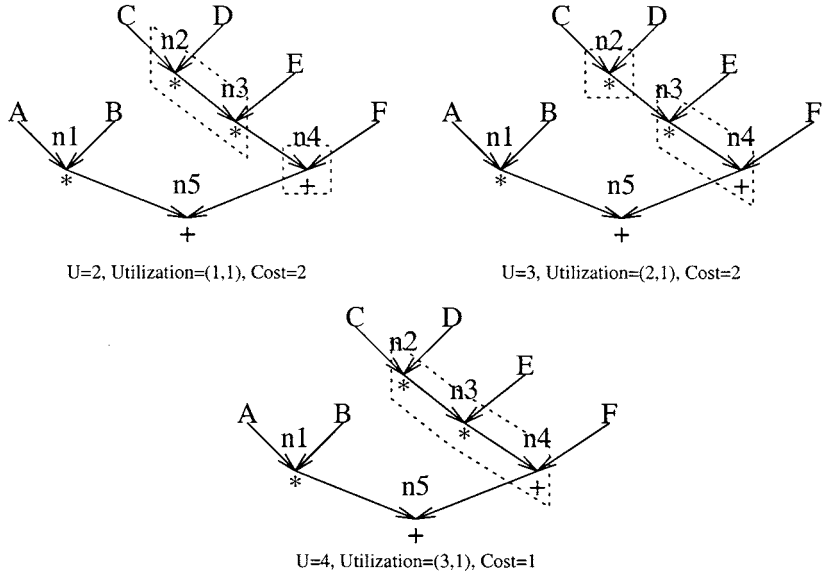


FIGURE 65.11 Mapping of node 4.

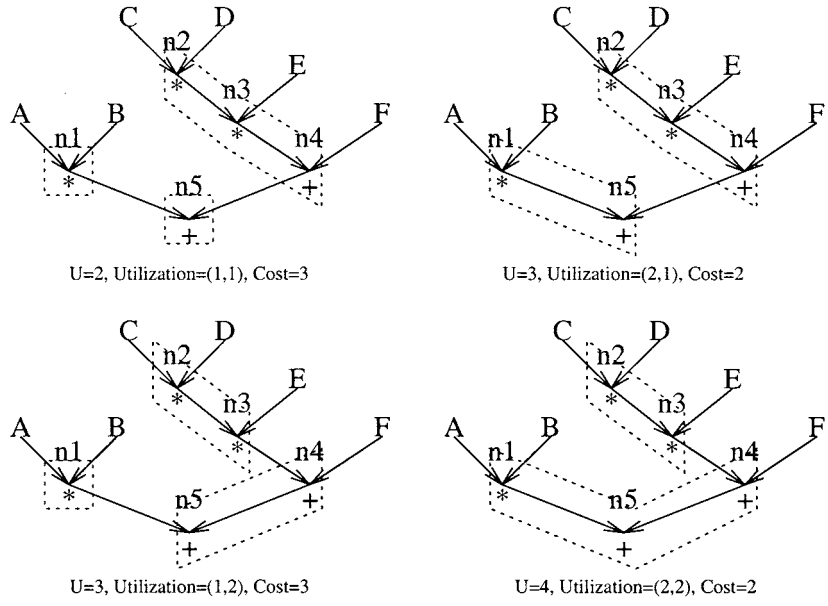


FIGURE 65.12 Mapping of node 5.

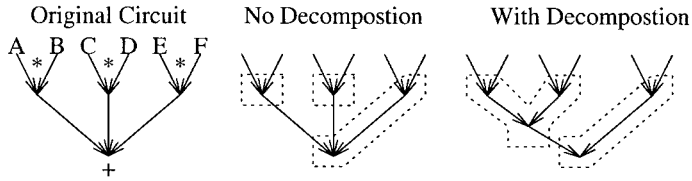


FIGURE 65.13 Decomposition example.

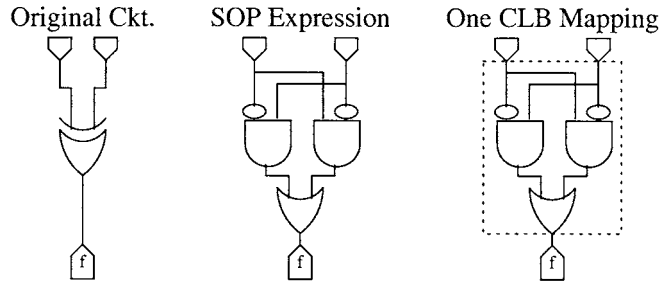


FIGURE 65.14 Reconvergent logic example.

function. In general, the decomposition of a node may yield a circuit that uses fewer CLBs. Consider, for example, implementations with four-input CLBs ($K = 4$) of the circuit shown in Fig. 65.13. Without decomposition, the output node forces the sub-optimal use of the first two function generators (i.e., A * B and C * D are implemented as individual CLBs). With decomposition, however, the output node OR gate is decomposed to form a new node, which implements the function: (A * B) + (C * D), which can be implemented in one CLB.

The original Chortle algorithm used an *exhaustive search* of all possible decompositions to find the optimal decomposition for the subcircuit, causing the running time at a node to increase exponentially as the fan-in increased. As a heuristic within the original Chortle algorithm, nodes would be arbitrarily *split* if the fan-in to a node exceeded 10, allowing each subfunction to be computed in a reasonable amount of time. If a node was split, however, the solution was no longer guaranteed to be optimal.

The improved Chortle-crf algorithm uses *first-fit-decreasing bin packing* algorithm to solve the decomposition problem. Large fan-in nodes are decomposed into smaller subnodes with smaller fan-in. Next, the look-up tables for the input functions are bin-packed into CLBs. A look-up table with k inputs is merged into the first CLB that has at least $K - k$ unused inputs remaining. A new CLB is generated, if needed, to accommodate the k inputs.

Reconvergent Logic

Reconvergent logic occurs when a signal is split into multiple function generators, and then those output signals merge at another generator. An example of reconvergent logic is shown in Fig. 65.14. When the XOR gate was converted to a SOP format by the technology-independent minimization phase, two AND gates and an OR gate were generated. Both AND gates share the same inputs. If the total number of *distinct* inputs is less than the size of the CLB, it is possible to map these functions into one CLB. The Chortle-crf algorithm finds all local reconvergent paths, and then examines the effect of merging those signals into one CLB.

Replicated Logic

For multi-output logic circuits, there are cases when logic duplication uses fewer CLBs than logic that uses subterms generated by a shared CLB. Figure 65.15 shows an example of a six-input circuit with two outputs. One product term is shared for both functions f and g. Without replication, the subfunction implemented by the middle AND gate would be implemented as one CLB, as well as the subfunctions for f and g. In this case, however, the middle AND gate can be replicated, and mapped into *both* function generators, thus allowing the entire circuit to be implemented using two CLBs, rather than three.

When a circuit has a fan-out greater than one, Chortle may implement the node *explicitly* or *implicitly*. For an explicit node, the subfunction is generated by a dedicated CLB, and this output signal is treated as an input to the rest of the logic. For an implicit node, the logic is replicated for each fan-out subcircuit. The algorithm computes the cost of the circuit, both with replication and without. Logic replication is chosen if this reduces the number of CLBs used to implement the circuit.

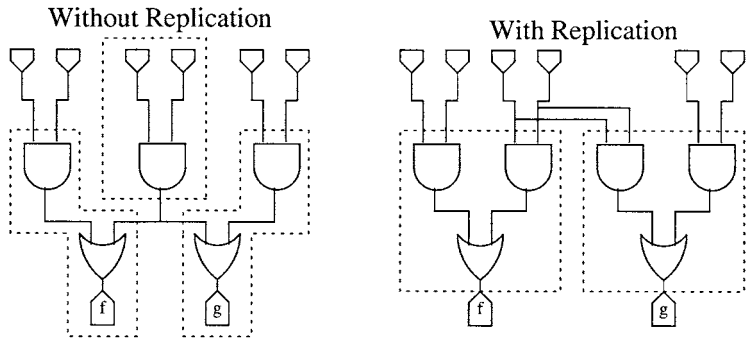


FIGURE 65.15 Replicated logic example.

Chortle-d

The primary goal of Chortle-d is to reduce the *depth* of the logic (i.e., the largest number of CLBs for any signal path through combinational logic).[12] By minimizing the longest paths, it is possible to increase the frequency at which the circuit can operate. Chortle-d is an enhancement of the Chortle-crf algorithm. Chortle-d, however, may use more look-up tables than Chortle-crf to implement a circuit with a shorter depth.

The Chortle-d algorithm separates logic into *strata*. Each *stratum* contains logic at the same depth. When nodes are decomposed, the outputs of the tables with the deepest stratum are connected to those at the next level. Chortle-d also employs logic replication, where possible. Replication often reduces the depth of the logic, as illustrated in Fig. 65.15.

The depth optimization is only applied to the critical paths in the circuit. The algorithm first minimizes depth for the entire circuit to determine the maximum *target depth*. Next, the Chortle-crf algorithm is employed to find a circuit that has minimum area. For paths in the area-optimized circuit that exceed the target depth, depth-minimization decomposition is performed. This has the effect of equalizing the delay throuth the circuit.

It was found that for the 20 circuits in the MCNC logic synthesis benchmark, the chortle-d algorithm constructed circuits with 35% fewer logic levels, but at the expense of 59% more look-up tables.

65.6 Two-Step Approaches

As with Chortle, the *two-step methods* start with an optimized network in which the number of literals is minimized. The network is decomposed to be feasible in the first step; then the number of nodes is reduced in the second step. If the given network is already feasible, the first step is skipped.

First Step: Decomposition

For a given FPGA device, with a k-input TLU, all nodes of the network with more than k inputs must be decomposed. Different methods decompose the network in different ways.

MIS-pga 1

MIS-pga 1 was developed at Berkeley for FPGA synthesis, as an extension of MIS-II. It uses two algorithms, *kernel decomposition* and *Roth-Karp decomposition*, to decompose the infeasible nodes separately; then it selects the better result.

Kernel decomposition decomposes an infeasible node n_i by extracting a kernel function, k_i, and splitting n_i based on k_i and its *residue*, r_i. The residue r_i, of a kernel k_i, of a function F, is the expression for F with a new variable substituted for all occurrences of k_i in F; for example, if $F = x_1x_2 + x_1x_3$, then

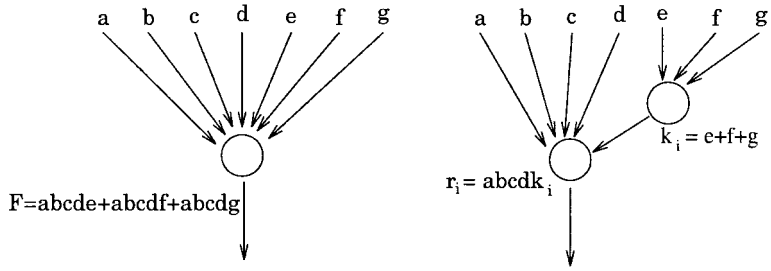


FIGURE 65.16 Example of kernel decomposition.

$k_i = x_2 + x_3$, and $r_i = x_1 k_i$. As there may be more than one kernel function that exists for a node, a cost function is associated with each kernel: $cost(k_i) = |sup(k_i) \bigcap sup(r_i)|$. The kernel with minimum cost is chosen. A kernel decomposition is illustrated in Fig. 65.16.

Splitting infeasible nodes by kernel functions minimizes the number of new edges generated. Therefore, the considerations of wiring resources and logic area are integrated together. This procedure is applied recursively until all nodes are feasible. If no kernels can be extracted for a node, an AND-OR decomposition is applied.

Roth-Karp decomposition is based on the classical decomposition of Ashenhurst and Curtis.[13] Instead of building a decomposition chart whose size grows exponentially, as it does with the original method, a compact cover representation of the on-set and the off-set of the function is used. The Roth-Karp algorithm avoids the expensive computation of the best solution by accepting the first bound set. As with kernel decomposition, the AND/OR decomposition is used as a last resort.

Hydra Decomposition

The Hydra algorithm, developed at Stanford University, is designed specifically for two-output TLU FPGAs.[14] Decomposition in Hydra is performed in three stages. The first and third stages are AND-OR decompositions, while the second stage is a *simple-disjoint decomposition*, which is defined as the following:

> Given a function, F, and its support, S, with $F = G(H(S^a), S^b)$, where S^a, $S^b \subseteq S$ and $S^a \bigcup S^b = S$; If $S^a \bigcap S^b = 0$, then G is a *disjoint decomposition* of F.

The first stage is executed only if the number of inputs to the nodes in the given network is larger than a given threshold. Without performing the first stage, the efficiency of the second stage would be reduced. The last stage is applied only if the resulting network is still infeasible.

In the second stage, the algorithm searches for all the function pairs that have common variables and then applies the simple-disjoint decomposition on them. As a result, two CLBs with the same fan-ins can be merged into one two-output CLB. The rationale is illustrated in Fig. 65.17.

A weighted graph $G(V,E,W)$ that represents the shared-variable relationship is constructed based on the given Boolean network. In the $G(V,E,W)$, V is the node set corresponding to that of the Boolean network; edge, $e_{ij} \subset E$, exists for any pair of nodes, $\{v_i, v_j\} \subset V$, if they share variables; and weight, $w_{ij} \subset W$, is the number of variables shared correspondingly. Edges are first sorted by weight and then traversed in decreasing order to check for simple-disjoint decomposition. A cost function, which is the linear combination of the number of the shared inputs and the total number of variables in the extracted functions, is computed to decide whether or not to accept a certain simple decomposition.

Xmap Decomposition

The Xmap decomposes the infeasible network by converting the SOP form from MIS--II to an if-then-else DAG representation.[15] The terms of the SOP network are collected in a set, T; then, variables are sorted in decreasing order of the frequency of their appearance in T; finally, the if-then-else DAG is formed by the following recursive function:

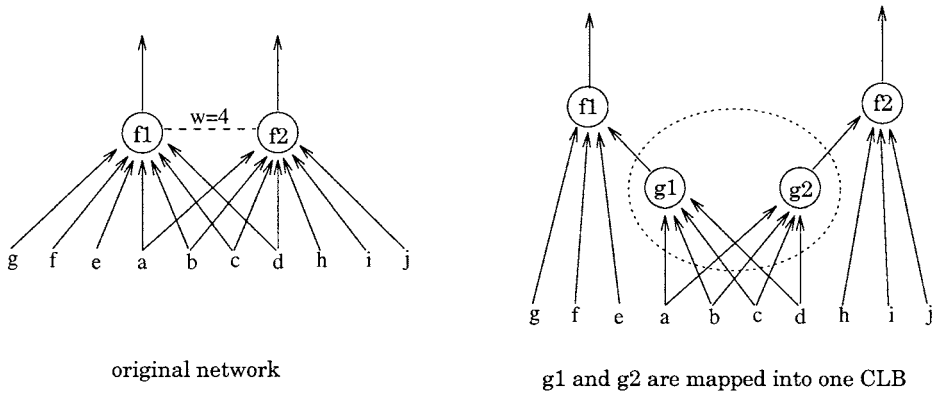


FIGURE 65.17 CLB mapping example.

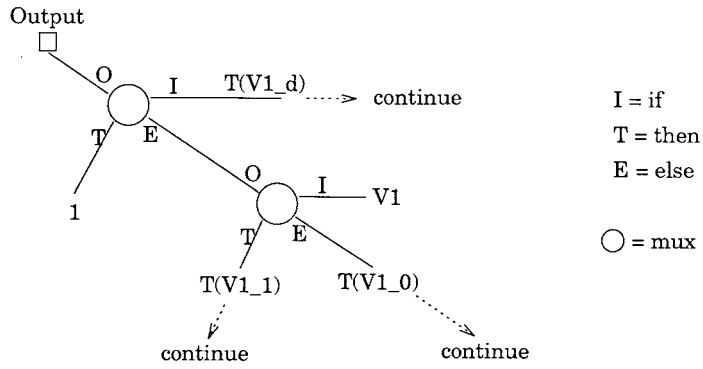


FIGURE 65.18 Result of first iteration.

- Let V be the most frequently used variable in the current set, T.
- Sort the terms in T into subsets $T(V_d)$, $T(V_1)$, according to V. $T(V_d)$ is the subset in which V does not appear, $T(V_1)$ is the onset of V, and $T(V_0)$ is the offset of V.
- Delete V from all terms in T; then apply the same procedure recursively to the three subsets until all variables are tested.

The resulting if-then-else DAG after first iteration is given in Fig. 65.18. A circuit that has been mapped to an if-then-else DAG is immediately suited for use with multiplexer-based CLBs.[16] Additional steps are used to optimize the DAG for use with TLU functions.

Second Step: Node Elimination

Three approaches have been proposed for node elimination: *local elimination*, *covering*, and *merging*.

Local Elimination

The operation used for *local elimination* is *collapsing*, which merges node n_i into node n_j whenever n_i is a fan-in node to n_j and the new node obtained is feasible. The **Hydra algorithm** accepts local eliminations as soon as they are found. **MIS-pga 1,** however, first orders all possible local eliminations as a function of the increase in the number of interconnections resulting from each elimination, and then greedily selects the best local eliminations.

The number of nodes can be reduced by local elimination, but its myopic view of the network causes local elimination to miss better solutions. Additionally, the new node created by merging multi-fan-out nodes may substantially increase the number of connections among TLUs and hence make the wiring problem more difficult. This problem is more severe in Hydra than in MIS-pga 1.

Covering

The *covering* operation takes a global view of the network by identifying clusters of nodes that could be combined into a single TLU. The operation is a procedure of finding and selecting supernodes. A *supernode*, S_i, of a node n_i, is a cluster of nodes consisting of n_i and some other nodes in the transitive fan-in of n_i such that the maximum number of inputs to S_i is k. Obviously, more than one supernode may exist for a node.

In **MIS-pga 1,** the covering operation is performed in two stages. In the first stage, the supernodes are found by repeatedly applying the *maxflow algorithm* at each node. In the second stage, an optimal subset of the supernodes that can cover the whole network using a minimum number of supernodes is selected by solving a *binate covering* problem whose constrains are: first, all intermediate nodes should be included in at least one supernode; second, if a supernode S_i is selected, some supernodes that supply the inputs of S_i must be selected [the ordinary (*unate*), covering problem just has the first constraint].

Hydra examines the nodes of the network in order of decreasing number of inputs. An unassigned node with the maximal number of inputs is chosen first. A second node is then chosen such that the two nodes can be merged into the same TLU and the cost function (same cost function as was used in decomposition step) is maximized. This greedy procedure stops when all unexamined nodes have been considered.

For **Xmap**, the logic blocks to be found are sub-DAGs of the if-then-else DAG for the entire circuit. The algorithm traverses the if-then-else DAG from inputs to outputs and keeps a log of inputs in the paths (called *signals set*) that can be used to compute the function of the node under consideration. Nodes in the signals set could be a *marked* node or a *clean* node. A marked node isolates its inputs to the current node, while a clean node exposes all its fan-ins. For an *overflow* node, whose signals set is larger than k (the number of inputs of the TLU), a *marking* procedure is executed to reduce the fan-ins of the overflow node. Xmap first marks the high-fan-out descendants of the node, and then marks the children of the node in decreasing order of the size of their signals set. The more inputs Xmap can isolate from the node under consideration, the better. The marking process cuts the if-then-else into pieces, each of which can be mapped into one CLB.

Merging

The purpose of the merging step is to combine nodes that share some inputs to exploit some of the particular features of FPGA architecture. For example, each CLB in the Xilinx XC4000 device has two four-input TLUs and a third TLU combining them with the ninth input (Section 65.3). In the three approaches discussed above, a post-processing step is performed to merge pairs of nodes after the covering operation. The problem is formulated as a maximum cardinality matching problem.

MIS-pga 2: A Framework for TLU-Logic Optimization

MIS-pga 2 is an improved version of MIS-pga 1. It combines the advantageous features of Chortle-crf, MIS-pga 1, Xmap, and Hydra. In each step, Mis-pga 2 tries different algorithms and chooses the best.[17]

Four decomposition algorithms are executed in the decomposition step:

1. *Bin-packing*: the algorithm is similar to that of Chortle-crf, except the heuristic of MIS-pga 2 is the *Best-Fit Decreasing.*
2. *Co-factoring decomposition*: it decomposes a node based on computing its *Shannon cofactor* ($f = f_1 f_2 + f_1' f_3$). The nodes in the resulting network have, at most, three inputs. This approach is particularly effective for functions in which cubes share many variables.

3. *AND/OR decomposition*: it can always find a feasible network, but is usually not a good network for the node elimination step. Therefore, it is used as the last resort.
4. *Disjoint decomposition*: unlike Hydra, this method is used on a node-by-node basis. When it is used as a preprocessing stage for the bin-packing approach, a locally optimal decomposition can be found.

MIS-pga 2 interweaves some operations of the two-step methods. For example, the local elimination operation is applied to the original infeasible network as well as to the decomposed, feasible network. This same operation is referred to as *partial collapse* when applied before decomposition. Unlike MIS-pga 1, which separates the covering and the merging operations, these two operations are combined together to solve a single, binate covering problem.

Because MIS-pga 2 does a more exhaustive decomposition phase, and because the combined covering/merging phase has a more global view of the circuit, MIS-pga 2 results are almost always superior to those of Chortle-crf, MIS-pga 2's results are almost always superior to those of Chortle-crf, MIS-pga 1, Hydra, and Xmap. For the same reason, MIS-pga 2 is relatively slow, as compared to the other algorithms.

65.7 Conclusion

By understanding how FPGA logic is synthesized, hardware designers can make the best use of their software development tools to implement complex, high-performance circuits. Synthesis of FPGA logic devices combines the algorithms of Chortle and its extensions, Xmap, Hydra, MIS-pga 1, and MIS-pga 2. Each of these methods starts with an optimized Boolean network and then maps the logic into the configurable logic blocks of a field-programmable gate array circuit. Because the optimal covering problem is NP-hard, heuristic approaches must balance between the optimality of the solution and the running time of the optimizer. Understanding this tradeoff is the key to rapidly prototyping logic using FPGA technology.

References

1. J. Rose, A.E. Gamal, and A. Sangiovanni-Vincentelli, Architecture of field-programmable gate arrays, *Proceedings of the IEEE*, vol. 81, pp. 1013-1029, July 1993.
2. Xilinx, Inc., *The Programmable Logic Data Book*, 1993.
3. ACTEL, *FPGA Data Book and Design Guide*, 1994.
4. A. Sangiovanni-Vincentelli, A.E. Gamal, and J. Rose, Synthesis methods for field programmable gate arrays, *Proceedings of the IEEE*, vol. 81, pp. 1057-1083, July 1993.
5. R.K. Brayton, G.D. Hachtel, and A. Sangiovanni-Vincentelli, Multilevel logic synthesis, *Proceedings of the IEEE*, vol. 78, pp. 264-300, Feb. 1990.
6. R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, Multi-level logic optimization and the rectangular covering problem, *IEEE International Conference on Computer-Aided Design*, (Santa Clara, CA), pp. 62-65, 1987.
7. R. Murgai, Y. Nishizaki, N. Shenoy, R.K. Brayton, and A. Sangiovanni-Vincentelli, Logic synthesis for programmable gate arrays, *ACM/IEEE Design Automation Conference*, (Orlando, FL), pp. 620-625, 1990.
8. R.K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A.R. Wang, MIS: A multiple-level logic optimization system, *IEEE Transactions on Computer-Aided Design*, vol. CAD-6, pp. 1062-1081, November 1987.
9. D. Bostick, G.D. Hachtel, R. Jacoby, M.R. Lightner, P. Moceyunas, C.R. Morrison, and D. Ravenscroft, The boulder optimal logic design system, *IEEE International Conference on Computer-Aided Design*, (Santa Clara, CA), pp. 62-69, 1987.

10. R.J. Francis, J. Rose, and K. Chung, Chortle: A technology mapping program for look-up table-based field programmable gate arrays, *ACM/IEEE Design Automation Conference*, (Orlando, FL), pp. 613-619, 1990.
11. R.J. Francis, J. Rose, and Z. Vranesic, Chortle-crf: Fast technology mapping for look-up table-based FPGAs, *ACM/IEEE Design Automation Conference,* (San Francisco, CA), pp. 227-233, 1991.
12. R.J. Francis, J. Rose, and Z. Vranesic, Technology mapping of look-up table-based FPGAs for performance, *IEEE International Conference on Computer-Aided Design*, (Santa Clara, CA), pp. 568-575, 1991.
13. T. Luba, M. Markowski, and B. Zbierzchowski, Logic decomposition for programmable gate arrays, *Euro ASIC '92*, pp. 19-24, 1992.
14. D. Filo, J.C.-Y. Yang, F. Mailhot, and G.D. Micheli, Technology mapping for a two-output RAM-based field programmable gate array, *European Design Automation Conference,* pp. 534-538, 1991.
15. K. Karplus, Xmap: a technology mapper for table-lookup field programmable gate arrays, *ACM/IEEE Design Automation Conference*, (San Francisco, CA), pp. 240-243, 1991.
16. R. Murgai, R.K. Brayton, and A. Sangiovanni-Vincentelli, An improved systhesis algorithm for multiplexor-based pga's *ACM/IEEE Design Automation Conference*, (Anaheim, CA), pp. 380-386, 1992.
17. R. Murgai, N. Shenoy, R.K. Brayton, and A. Sangiovanni-Vincentelli, Improved logic synthesis algorithms for table look up architectures, *IEEE International Conference on Computer-Aided Design*, (Santa Clara, CA), pp. 564-567, 1991.

Kanopoulos, N "Testability Concepts and DFT"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

66

Testability Concepts and DFT

Nick Kanopoulos
Atmel, Multimedia and Communications

66.1 Introduction: Basic Concepts

Physical faults or *design errors* may alter the behavior of a digital circuit. Design errors are tackled by redesigning the circuit, whereas physical errors can be reduced by determining appropriate operating conditions.[1,2]

There are many sources of physical faults: improper interconnections between parts, improper assembly, missing parts, and erroneous parts may occur while the circuit is being manufactured. After manufacturing, the circuit may fail due to excessive heat dissipation or for mechanical reasons associated with corrosions and, in general, bad maintenance. *Short-circuit faults* are those due to connections of signal lines that must be disconnected. In addition, disconnecting lines that must be connected may cause open-circuit faults.[1,3]

Failures in the operation of digital circuits are addressed in the testing process, which is abstracted in Fig. 66.1. Typically, the testing process determines the presence of faults. The circuit being tested is often called the *circuit under test* (CUT). Errors are detected by applying *test patterns* on the inputs of the CUT and analyzing the responses on its outputs. A test pattern is typically a vector of 0 and 1, and every bit corresponds to an input of the CUT. A test pattern is generated by a *test pattern generator* (TPG) tool. The responses are analyzed using an *output response verification* (ORV) tool. The ORV tool is a comparator circuit.

The testing process is done periodically during the circuit's life span. It is initially done after fabrication and while the CUT is still at the wafer. Testing is also done when it is removed from the wafer, and later it is tested as part of a *printed circuit board* (PCB).

Testing is done either at the *transistor level* or at the *logical level*. We are considering here logical-level testing for which TPG and ORV are concerned with binary values, that is, the signals are binary values. The components are gates and flip-flops (or latches). We do not consider *parametric testing*, which analyzes waveforms at the transistor level. A circuit $C = (V,E)$ is considered as a collection V of components and E lines. Figure 66.2 depicts a combinational circuit at the logic level. The components represent gates. The integer value on each circuit line indicates its label. the circuit inputs are lines 1, 2, 3, 6, 7, 23, and 24.

The test patterns may be precomputed by a pattern generator program, often referred to as an *automatic test pattern generator* (ATPG). The goal in an ATPG program is to quickly compute a small set of test patterns that detect all faults. The design of ATPG tools is a difficult task. Once the patterns are generated, they are stored in the memory of an *automatic test equipment* (ATE) mechanism that applies the test patterns and analyzes the responses using the ORV tool. In order for the ATE tools to test PCBs or complex digital systems, they must be controlled by computer programs.

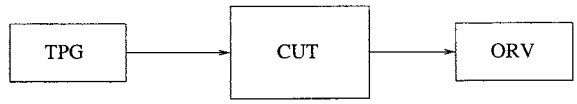


FIGURE 66.1 The testing process.

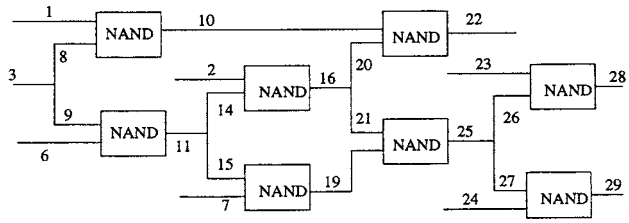


FIGURE 66.2 A circuit at the logic level.

ATE equipment is often very expensive. Thus, some circuits are designed so that they can test themselves. This concept is called *built-in self-testing* (BIST). In BIST, the TPG and ORV tools are on-chip and the concern is twofold: accuracy and hardware cost. Chapter 67 reviews popular ATPG tools and BIST mechanisms. Furthermore, the complexity of current *application-specific integrated circuits* (ASICs) has led to the development of sophisticated CAD tools that automate the design of BIST mechanisms. Such tools are presented in Chapter 68.

The testing process requires fault models that precisely define the behavior of the (logic-level) circuit. The standard model for logical-level testing is the *stuck-at fault model*. This model associates two types of faults for each line l of the circuit: the stuck-at 0 fault and the stuck-at 1 fault. The stuck-at 0 fault assumes that line l is permanently stuck at the logic value 0. Similarly, the stuck-at 1 assumes it is stuck at 1. The single stuck-at fault model assumes that only one such fault is present at a time. Under the single stuck-at fault model, a circuit with $|E|$ lines can have at most $2 \cdot |E|$ faults. Although the stuck-at fault model appears to be simplistic, it has been shown to be very effective, and a set of patterns that detect all single stuck-at faults covers most (physical) faults as well.

However, the stuck-at fault model is of limited use to faults associated with delays in the operation of the CUT. Such faults are called *delay faults*. Although it has been shown that testing for delay faults can be theoretically reduced to testing for stuck-at faults in an auxiliary circuit, the size of the latter circuit is prohibitively large. Instead, an alternative fault model, the path delay fault model, is applied successfully. The path delay fault model is postponed until Chapter 68.

In order for a test pattern to detect a stuck-at fault on line l, it must guarantee that the complementary logic value is applied on l. In addition, it must apply an appropriate logic value to each of the other lines in the circuit so that the erroneous behavior of the circuit at line l is propagated all the way to an output line. This way, the fault is observed and detected. The problem of generating a test pattern that detects a given stuck-at fault is an intractable problem, that is, it requires algorithms whose worst-case complexity it exponential to $O(|V| + |E|)$, the size of the input circuit. ATPG algorithms for the stuck-at fault model are described in Chapter 67. They are very efficient, and require seconds per stuck-at fault, even for very large circuits.

The stuck-at fault model is easy to use, involves only $2 \cdot |E|$ faults, and requires at most $2 \cdot |E|$ test patterns. Once a pattern is applied by the ATE equipment, a process called *fault simulation* is performed in order to determine how many faults are detected by the applied test pattern. A key measure of the effectiveness of a set of test patterns is its *fault coverage*. This is defined as the percentage of faults detected by the set of patterns.

Fault simulation is needed in order to determine the fault coverage of a set of test patterns. Fault simulation is important in testing with ATE as well as in the design of the on-chip test mechanisms. Fault

simulation is an inherently polynomial process for the stuck-at fault model. However, an overview of sophisticated fault simulation techniques is presented in Chapter 68.

Exhaustive TPG applies all possible test patterns at the circuit inputs, that is, $2^{|I|}$ test patterns for a circuit with $|I|$ inputs. Instead, *pseudo-exhaustive TPG* guarantees that all stuck-at faults are covered with less than $2^{|I|}$ patterns. BIST schemes are often designed so that pseudo-exhaustive TPG is guaranteed. (See also Chapter 67.)

However, sometimes we need to generate patterns only for a given set of stuck-at faults. This type of TPG is called a *deterministic TPG*, and the generated test patterns must detect the predefined set of test patterns. A good pseudo-exhaustive or deterministic TPG tool must guarantee that a compact test set is generated.

Consider a three-input NAND gate where lines *a*, *b*, and *c* are the three inputs and line *d* is the output. There exist three directly *controllable lines* and one *observable line*. Let us describe a test pattern as a binary vector of three values applied to lines *a*, *b*, and *c*, respectively. There are $2 \cdot 4$ stuck at faults. By applying 2^3 patterns, all the faults are covered. However, a compact test set contains at least four test patterns. Consider the following order of pattern application. Pattern (111) is applied first and covers four stuck-at faults. Pattern (110) covers two additional stuck-at faults. Finally, patterns (101) and (011) are needed to cover the last two faults. The number of applied patterns is also called the *test length*. The problem of minimizing the test length, which guarantees 100% fault coverage, is intractable.

Heuristic methods can be applied to reduce the test length. Two faults are called *indistinguishable* if they are detected by the same set of test patterns. Identification of indistinguishable faults is an important concept in test set compaction.

A stuck-at fault is called *undetectable* if it cannot be detected by any pattern. Any circuit that has at least one undetectable fault is called *redundant*. Any redundant circuit can be simplified by removing the line that contains the undetectable fault, and possibly other lines, without changing its functionality.

In the above, the CUT was assumed to be a combinational circuit. The TPG process is significantly more difficult in sequential logic. In order for a stuck-at fault to be detected, a sequence of test patterns rather than a single pattern must be applied. The process of generating sequences of pattern with ATPG or on-chip TPGs is a tedious job. These concepts are discussed in more details in Chapter 67.

66.2 Design for Testability

Design for testability (DFT) is applied to reduce difficulties associated with the TPG process on sequential circuits. DFT suggests that the digital circuit is designed with *built-in* features that assist the testing process. The goal in DFT is to maximize fault coverage, the test pattern generation process, the time required to apply the generated patterns, and the *built-in hardware overhead*. By definition, DFT is needed for BIST where TPG and ORV are on-chip. However, the majority of the proposed DFT methods are targeting the simplification of the ATPG process for sequential circuits, and assume that ATE is used.

There are some guidelines that have been developed by experienced engineers and lead the insertion of the built-in mechanisms so that the input sequential CUT becomes testable with ATPG tools.

1. Set the circuit at a known state before and during testing. This is achieved by a RESET control line that is connected to the asynchronous CLEAR of each flip-flop in the CUT.
2. Partition the CUT into subcircuits which are tested easier.
3. Simplify the circuit to avoid redundancies.
4. Control and observe lines on feedback paths, lines that are far from inputs and outputs, and lines with high fan-in and fan-out.

One way to implement the first guideline (1) is by inserting *test points* to control and observe at lines *x* that break all feedbacks. A test point on line $x = (x_{in}, x_{out})$ is a simple circuit that simulates the function $f(x, s, c) = s' \cdot (x + c)$. The output of this circuit feeds x_{out}. Input signals *s* and *c* are controlling. When $s = 0$ and $c = 0$, we have that $f = x$; that is, this combination can be used in operation mode. When $s = 0$ and $c = 1$, function *f* evaluates to 1. When $s = 1$ and $c = 0$, *f* evaluates to 0. The last two combinations

can be used in the testing mode, and they guarantee that the line is fully controllable. It can be made observable by simply allowing for a new primary output at signal *x*.

Another mechanism is to use *bypass latches*, also referred to as *bypass storage elements* (bses). These latches are bypassed during the *operation mode* and are fully controllable and observable points in the *testing mode*. This dual functionality is easily obtained with a simple multiplexing circuitry. (See also Fig. 66.3 below.)

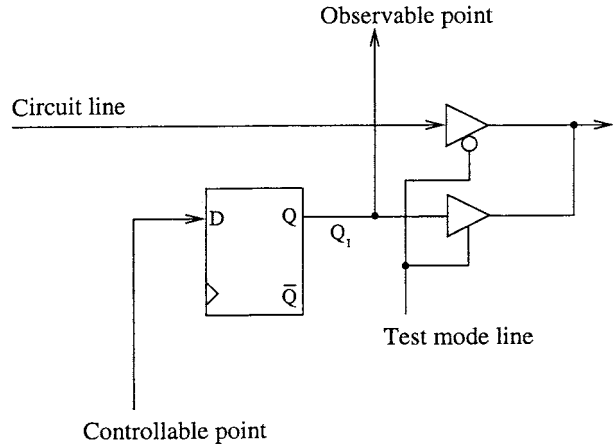


FIGURE 66.3 The structure of a bypass storage element.

In both cases, the total hardware must be minimized, subject to a lower bound on the enhancement of the circuit's testability. This optimization criterion requires sophisticated CAD tools, some of which are described in Chapter 68.

The most popular DFT approach is the *scan design*. The approach is a variation of the bypass latch approach discussed earlier. Instead of adding new latches, as the bypass latch approach suggests, the scan design approach enhances every flip-flop in the circuit with a multiplexing mechanism that allows for the following. In the operation mode, the flip-flop behaves as usual. In the *testing mode*, all the flip-flops are connected to a single shift chain. The input of this chain is a single controllable point and its output is a single observable point.

In the testing mode, each scanned flip-flop is a fully controllable and observable point. Observe that the testing phase amounts to testing combinational logic. Therefore, the ATPG (or the on-chip TPG) needs to generate single patterns instead of sequences of patterns. Each generated pattern is serially shifted in the scan chain. Typically, this process requires as many clock cycles as the number of flip-flops. Once every flip-flop obtains its controlling value, the circuit is turned to operation mode for a single cycle. Now the flip-flops are disconnected from the scan chain, and at the end of the clock cycle, the flip-flops are loaded with values that are to be observed and analyzed. Now the circuit is switched back into the testing mode (i.e., all flip-flops form again a scan chain). At this point, the states of the flip-flops are shifted out and are analyzed. This requires no more clock cycles than the number of flip-flops.

The described scan approach is also called *full scan* because all flip-flops in the circuit are scanned. The advantage of the full scan approach is that it requires only two additional I/O pins: the input and output of the scan chain, respectively. The disadvantage is that it is time-consuming due to the shift-in and shift-out processes for each applied pattern, especially for circuits with many flip-flops. For such circuits, it is also hardware intensive because every flip-flop must have dual operation mode capability. The hardware and the application time can be reduced by employing CAD tools. See also Chapter 68.

Another way to reduce application time and hardware cost is through *partial scan*. In partial scan, only a subset of flip-flops is scanned. The flip-flops and their ordering in the scan also require sophisticated CAD tools. The tradeoff in partial scan is that the ATPG tool may have to generate test sequences

rather than single patterns. A CAD tool is needed in order to select and scan a small number of flip-flops. This guarantees low hardware overhead and low application time. The flip-flop selection must also guarantee an upper bound on the length of any generated test sequence. This simplifies the task of the ATPG tool and has an impact on the test application time.

References

1. M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.
2. J.P. Hayes, *Introduction to Digital Logic Design*, Addison Wesley, Boston, 1993.
3. P.H. Bardell, W.H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques*, John Wiley & Sons, New York, 1987.

Kagaris, D. "ATPG and BIST"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

67

ATPG and BIST

Dimitri Kagaris
Southern Illinois University

67.1 Automatic Test Pattern Generation

Automatic test pattern generation (ATPG) refers in general to the set of algorithmic techniques for obtaining a set of test patters that detect possible faulty behavior of a circuit after its fabrication. Faults during fabrication can affect the functional correctness of the circuit (*functional faults*) and its timing performance (*delay faults*). In this chapter, we deal only with functional faults. The physical faults in a circuit (such as breaks, opens, technology-specific faults) have to be modeled as *logical faults* (like "stuck-at" and "bridging" faults) in order to reduce the required complexity of ATPG. The most common fault model used in practice is the *stuck-at model*, where lines in a gate-level or register-transfer-level description of a circuit are assumed to be set permanently to a "1" or "0" value in the presence of a fault. An additional restriction is that the modeled faults cause only one line in the circuit to have a stuck-at value (*single stuck-at fault model*). Patterns generated under this model have been shown in practice to cover many of the unmodeled faults as well.

Given a list of stuck-at faults of interest, the primary goal of ATPG is to generate a test pattern for each of these faults, and additionally to keep the overall number of test patterns generated as small as possible. The latter is required for reducing the time/cost of applying the test patterns to the circuit. In this section, we describe basic test pattern generation (TPG) algorithms for finding a test pattern given a stuck-at fault, and other aspects of the ATPG process for facilitating the task of TPG algorithms and reducing the number of generated test patterns.

TPG Algorithms

Given a target fault of line l being stuck at value v, denoted by l s–a–v, a TPG algorithm attempts to generate a pattern such that (1) the pattern brings l to have a value $\bar{v}$ (*fault activation*) and (2) the same pattern carries over the effect of the fault to a primary output (*fault propagation*). A path from line l to a primary output along each line of which the effect of the fault is carried over is called a sensitized path.

The case of a line having a value of "1" in the correct circuit and a value of "0" in the circuit under the fault l s–a–v is denoted by the symbol D and, similarly, the opposite case is denoted by $\overline{D}$. Given the symbols D and $\overline{D}$, the basic Boolean operations AND, OR, NOT can be extended in a straightforward manner. For example, AND $(1, D) = D$, AND$(1, \overline{D}) = \overline{D}$, AND$(0, D) = 0$, AND$(0, \overline{D}) = 0$, AND$(x, D) = x$, AND$(x, \overline{D}) = x$ (where x denotes the don't-care case), etc.

TPG Algorithms for Combinational Circuits

A basic TPG algorithm for combinational circuits is the *D*-algorithm.[1] This algorithm works as follows. All values are initially assigned a value of *x*, except line *l* which is assigned a value of *D* if the fault is *l* s–a–0, and a value of $\overline{D}$ if the fault is *l* s–a–1. Let *G* be the gate whose output line is *l*. The algorithm goes through the following steps:

1. Select an assignment for the inputs of *G* out of all possible assignments that produce the appropriate *D*-value (i.e., a *D* or $\overline{D}$) at the output of *G*. This step is known as *fault activation*. All possible assignments are fixed for each gate type and are referred to as the *primitive d-cubes for the faul* (*pdcf*s) of the gate. For example, the *pdcf*s of a two-input AND gate are $0x\overline{D}$, $x0\overline{D}$, and $11D$, and the *pdcf*s of a two-input OR gate are $1xD$, $x1D$, and $00\overline{D}$ (using the notation *abc* for a gate with input values *a* and *b* and output value *c*).
2. Repeatedly select a gate from the set of gates whose output is currently *x* but has at least one input with a *D*-value. This set of gates is known as the *D*-frontier. Then select an assignment for the inputs of that gate out of all possible assignments that set the output to a *D*-value. All possible assignments are fixed for each gate type and are referred to as the *propagation d-cubes* (*pdc*s) of the gate. For example, the *pdc*s of a two-input AND gate are $1DD$, $D1D$, $1\overline{D}\overline{D}$, $\overline{D}1\overline{D}$, DDD, and $\overline{D}\overline{D}\overline{D}$. By repeated application of this step, a *D*-value is eventually propagated to a primary output. This step is known as *fault propagation*.
3. Find an assignment of values for the primary inputs that establishes the candidate values required in steps (1) and (2). This step is known as *line justification*. For each value that is not currently accounted for, the line justification process tries to establish ("justify") the value by (a) assigning binary values (and no *D*-values) on the inputs of the corresponding gate, working its way back to the primary inputs (this process is referred to as *backtracing*; and (b) determining all values that are imposed by all candidate assignments thus far (*implication*) and checking for any inconsistencies (*consistency check*).
4. If during step (3), an inconsistency is found, then the computation is restored to its state at the last decision point. This process is known as *backtracking*. A decision point can be (a) the decision in step (1) of which *pdcf* to select; (b) the decisions in step (2) of which gate to select from the *D*-frontier and which *pdc* to select for that gate; (c) the decision in step (3) of which binary combination to select for each value that has to be justified.
5. If line justification is eventually successful after zero or more backtrackings, then the existing values on the primary inputs (some of which may well be *x*) constitute a test pattern for the fault. Otherwise, no pattern can be found to test the given fault and that fault is thus shown to be redundant.

The order of steps (2) and (3) may be interchanged, or even the two steps may be interspersed, in an attempt to reduce the running time, but the discovery or not of a pattern is not affected by such changes.

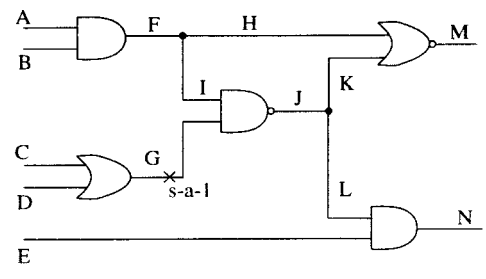


FIGURE 67.1 Example circuit.

As an example of the application of the *D*-algorithm, consider the circuit in Fig. 67.1 and the fault G s–a–1. In order to establish G ← $\overline{D}$, the *pdcf* CD ← 00 is chosen and the *D*-frontier becomes {J} (gates are named by their output line). Then, gate J is considered and the *pdc* setting I ← 1 is selected with result J ← *D* and new *D*-frontier {M, N}. Assume gate M is selected. Then, the *pdc* setting H ← 0 is selected with result M ← $\overline{D}$. However, the justification of current values H ← 0 and I ← 1 results in conflict, so the algorithm backtracks and tries the next *pdc* for gate M which sets H ← *D*. But again, this cannot be justified. Then the algorithm backtracks once

more and selects gate N from the *D*-frontier. Then the assignment E ← 1 is made, which results in N ← *D*. Since the values E ← 1 and I ← 1 can now be justified without conflict, the algorithm terminates successfully, returning test pattern ABCDE = 11001.

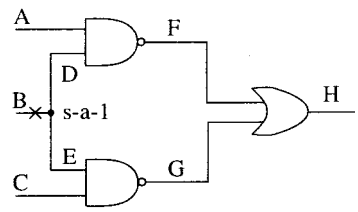


FIGURE 67.2 Multipath sensitization.

As another example, consider the circuit in Fig. 67.2 and the fault B s–a–1. In order to establish B ← $\overline{D}$, the assignment B ← 0 is made and the *D*-frontier becomes {F, G}. Assume that gate F is selected. In order to propagate the fault to line H, the *pdc* setting A ← 1 is selected and the *pdc* of gate H setting G ← 0 is tried. But this results in conflict, as B (and E) are required to be 0. Then the algorithm backtracks and tries the next available *pdc* of H which sets G ← *D*. This value can now be justified by setting C ← 1, with resulting test pattern ABC = 101. A similar thing happens if gate G is selected from the original *D*-frontier. That is, in this example, the algorithm had to sensitize two paths simultaneously from the fault site to a PO in order to detect the fault. This is referred to as *multipath sensitization*, but its need rarely arises in practice. To reduce computational time, examination of *pdc*s involving more than one input being set to *D* (or $\overline{D}$) is often omitted.

Another basic TPG algorithm is the PODEM.[2] The PODEM algorithm uses also the five-valued logic (0, 1, *x*, *D*, $\overline{D}$), and works as follows. Initially, all lines are assigned a value of *x* except line *l*, which is assigned a value of $\overline{D}$ if the fault is *l* s–a–0, and a value of *D* if the fault is *l* s–a–1. The algorithm at each step tries to satisfy an *objective* (*v*, *l*), defined as a desired value *v* at a line *l* by making assignments only to primary inputs (PIs), one PI at a time. The mapping of an objective to a single PI value is done heuristically, as explained below. The initial objective is ($\bar{v}$, *l*), assuming that the examined fault is *l* s–a–*v*. Then the algorithm computes all implications of the current pattern of values assigned to PIs. If the effect of the fault is propagated to a primary output (PO), the algorithm terminates with success. If a conflict occurs and the fault cannot be activated or cannot be propagated to a PO, then the algorithm backtracks to the previous decision point, which is the last assignment to a PI. If no conflict occurs but the fault has not been activated or not been propagated to a PO because the currently implied values on the lines involved are *x*, then the algorithm continues with the same objective (*v*, *l*) if the fault is still not activated, or with an objective ($\bar{c}$, *l*′) if the fault has been activated but not propagated, where *l*′ is an input line of a gate from the *D*-frontier that has currently assigned a value of *x* on it, and *c* is the controlling value of that gate.

The determination of which single PI to selected and which value to assign to it given an objective (*v*, *l*) is done heuristically (in the worst case, at random). A simple heuristic is to select a path from line *l* to a PI such that every line of the path except *l* has an *x* value on it, and assign to that PI the value *v* ($\bar{v}$) if the total number of inverting gates (i.e., NOT, NAND, NOR) along that path is even (odd). In addition, concerning the selection of a gate from the *D*-frontier, a simple heuristic is to select the gate that is closest to a PO. As an example of the application of PODEM, consider the circuit of Fig. 67.1 and the fault G s–a–1. The initial objective is (0, G). The chosen PI assignment is C ← 1, and this has no implications. The objective remains the same, with chosen PI assignment D ← 0 and implications G ← $\overline{D}$. The *D*-frontier becomes {J} and the next objective is (1, I). This results in PI assignments A ← 1 and B ← 1 with implications F ←1, H ← 1, I ← 1, M ← 0, J ← *D*, K ← *D*, L ← *D*, and new *D*-frontier {N}. The next objective is (1, E), which is immediately satisfied and has implication N ← *D*. So, the algorithm returns successfully with test pattern ABCDE = 11001.

In the example of Fig. 67.2, PODEM works as follows. The original objective is (0, B). With PI assignment B ← 0, the *D*-frontier becomes {F, G}. Assuming gate F is selected, the next objective is (1, A), which is immediately satisfied with resulting implication F ← *D* and new *D*-frontier {G, H}. Given that gate H is selected as closer to the output, the next objective is (0, G) which leads to the PI assignment C ← 1 with implications G ← *D* and H ← *D*. That is, the resulting test pattern is ABC = 101. Notice that although the implied value for G was *D* while the objective generated was (1, G), this is not considered a conflict, since the goal of any objective is only to lead to a PI assignment that activates and propagates the fault to a PO.

As an example involving backtracking in PODEM, consider the circuit of Fig. 67.3 and the fault J s–a–1. Starting with objective (0, J), the PI assignment A ← 0 is made (using path HFEA) with no implication, and then the PI assignment B ← 0 is made (using path HFEB) with implications E ← 0, F ← 0, G ← 0, H ← 0, I ← 1, J ← 1. But the latter constitutes a conflict, and so the algorithm backtracks trying PI assignment A ← 1. The implications of this assignment are E ← 1, F ← 1, G ← 1. Since the fault at J is still not activated, the objective (1, B) is generated next (using path HFEB), which is satisfied immediately but has no new implications, then the objective (0, C) is generated (using path HC), which is satisfied immediately and has implication H ← 0. Finally, the objective (1, D) is generated (using path ID), which is satisfied immediately and has implications I ← 0 and J ← 0. Since the fault is now activated and (trivially) propagated, the algorithm terminates successfully with test pattern ABCD = 1101.

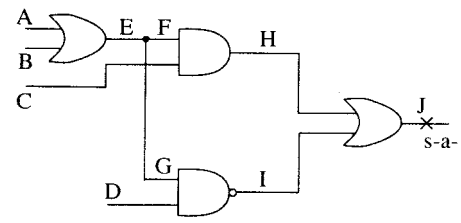


FIGURE 67.3 Backtracking in PODEM.

Both of these basic algorithms are complete in that given enough time, they will find a pattern for a fault if and only if the fault is not redundant. The *D*-algorithm performs an implicit state-space search by assigning values to the lines of the circuit, whereas PODEM performs an implicit state-space search by assigning values to the PIs only. For circuits with no fan-out or without reconvergent fan-out, the algorithms take linear time to the size of the circuit; but for general circuits (with reconvergent fan-out), the algorithms may take exponential time. In fact, the test pattern generation problem has been shown to be NP-complete.[3] The implicit state search in conjunction with a variety of heuristic measures can cut down the running time requirements. For instance, performing as many implications at each point as possible and checking for the existence of at least one path from a gate in the *D*-frontier to a PO such that every line on that path has an *x* value (otherwise, fault propagation is impossible) are very useful measures.

In general, PODEM is faster than the *D*-algorithm. Several extensions to PODEM have been proposed, such as working with more than one objective each time, and stopping backtracking before reaching PIs. For instance, the FAN algorithm[4] maintains a list of multiple objectives and stops backtracking at headlines rather than just PI lines. A *headline* is a line that is driven by a subcircuit containing no line that is reachable from some fan-out stem, and, therefore, can be justified at the end with no conflicts. As a short illustration, consider the example in Fig. 67.3. In order to activate the fault (i.e., J ← 0), both lines H and I must be driven to 0. The objectives (H, 0) and (I, 0) are now both taken into consideration. In order to achieve objective (H, 0), the assignment E ← 0 can be selected, as line E is a headline. But in order to achieve objective (I, 0), the assignment E ← 1 is required. Therefore, the algorithm selects the alternative assignment C ← 0 (as C is a PI) for objective (0, H), and then selects the assignment E ← 1 (as E is a headline) and D ← 1 (as D is a PI) for objective (0, I), which results in success. The justification of the value on E is left for a final pass with resulting test pattern ABCD = 1*x*00 or ABCD = *x*100.

There are a plethora of TPG algorithms based on various strategies (see, e.g., Ref. 5 for more information). There are also parallel TPG algorithms designed for particular devices such as ROMs and PLAs.

TPG Algorithms for Sequential Circuits

Detecting faults in sequential circuits is much more difficult than for combinational circuits. This is due to the fact that because of the memory elements present in the logic, a sequence of patterns is generally required for each fault, along with an appropriate initial state. In general, TPG techniques for combinational circuits can be applied to sequential circuits by considering the iterative logic array model of the sequential circuits. This model applies to both synchronous and asynchronous sequential circuits, although it is more complex for the latter.

Given a current state vector Q and a current input vector X, the function of a sequential circuit is specified as a mapping from (X, Q) to (Q^+, Z), where Q^+ is the next state vector and Z is the resulting

output. In the *iterative logic array* representation, the sequential circuit is modeled as a series of combinational circuits C_0, C_1, …, C_N, where N is the length of the current input pattern sequence applied to the sequential circuit. Each circuit C_i, referred to as a *time frame*, is an identical copy of the sequential circuit but with all feedback removed, and has inputs X_i and Q_i, and outputs Q_i^+ and Z_i. Inputs X_i are driven by the ith pattern applied to the sequential circuit and inputs Q_i are driven by the outputs Q_{i-1}^+ of the previous time frame for $i > 0$, with Q_0 being set to the original initial state of the sequential circuit. All outputs Z_i are ignored except for the outputs Z_N of the last time frame, which constitute the output of the sequential circuit resulting from the specific input sequence and initial state.

Given a stuck-at fault, the fundamental idea in sequential TPG is to create an iterative logic array of appropriate length N and justify all the values necessary for the fault to be activated and propagated to the outputs Z_N of the last time frame. If this can be achieved with the values of the Q_0 inputs of the first time frame being set to 'x's, then a *self-initializing* test sequence is produced. Otherwise, the specific values required for the Q_0 inputs (preferably, all "0"s) are assumed to be easily established through a reset capability. In principle, one can start from one time frame C_t (with the index t to be appropriately adjusted later) and try to propagate the effect of the fault to either some of the Z_t lines or some of the Q_t^+ lines. In case of propagation to the Z_t lines, C_t becomes tentatively the last frame in the iterative logic array and line justification by assignments to the X_t and Q_t lines is repeatedly done in additional time frames C_{t-1}, C_{t-2}, …, C_{t-N_b} (up to some number N_b), until all lines are justified with either Q_{t-N_b} being set to all 'x's or to a resetable initial state. In case of propagation to the Q_t lines, additional time frames C_{t+1}, C_{t+2}, …, C_{t+N_f} are considered (up to some number N_f), until the effect of the fault is propagated to the Z_{Nf} lines. Notice that because each time frame contains the same fault, the propagation can be done from any of the C_{t-1}, C_{t-2}, …, C_{t-N_b} time frames to the Z_{N_f} lines. Then, line justification is again attempted as above. In case of conflict during the justification process, backtracking is attempted to the last decision point, and this backtracking can reach as far as the C_{t-N_f} frame.

In order to reduce the storage required for the computation status as well as the time requirements of this process, algorithms that consider only backward justification and no forward fault propagation have been proposed. For example, the *Extended Backtrace* (EBT) algorithm[6] selects a path from the fault site to a primary output which may involve several time frames C_{t-1}, C_{t-i+1}, …, C_t, and then tries to justify all values for the sensitization of this path (along with the requirements for the initial state) by working with time frames C_t, C_{t-1}, …, C_{t-i}, …, C_{t-N_b}.

As an illustration of the application of the EBT algorithm, consider the sequential circuit in Fig. 67.4(a). The structure of each time frame in the iterative logic array representation of it is given in Fig. 67.4(b).

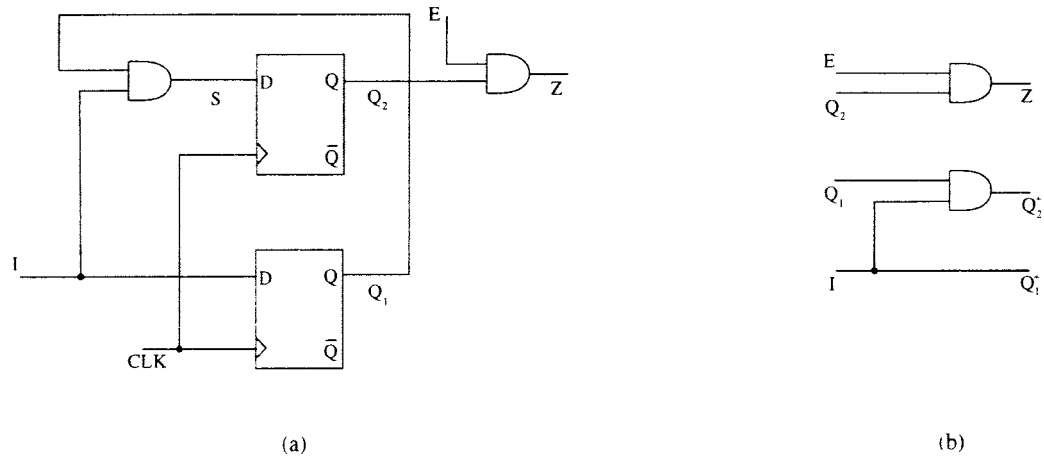


FIGURE 67.4 A sequential circuit and a time frame in the iterative logic array representation.

Consider the fault S s–a–0. The EBT algorithm selects the path SQ_2Z to propagate the fault. This path involves two time frames, as the value of line S is the value of line Q_2 before one clock cycle (by definition of the D-type flip-flop). Considering the index of the last frame to be t and following the structure of each time frame (Fig. 67.4(b)), the path actually comprises the lines $Z_{[t]}$, $Q_{2[t]}$, $Q^+_{2[t-1]}$. In order to sensitize this path, line $E_{[t]}$ must be set to 1. Now, in order to activate the fault at line S, which is identified with $Q^+_{2[t-1]}$, lines $I_{[t-1]}$ and $Q_{1[t-1]}$ must be set to 1. Assuming a self-initializing sequence is sought, further justification needs to be made for the value $Q_{1[t-1]}$, which is equal tot he value of line $Q^+_{1[t-2]}$ in an additional time frame indexed by $t-2$. Since $Q^+_{1[t-2]}$ is set directly by $I_{[t-2]}$, the search is over and the self-initializing sequence (first pattern first) is IE = ($1x$, $1x$, $x1$).

Other ATPG Aspects

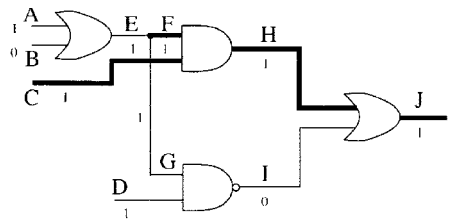
There are several components in the ATPG process that are centered around the TPG algorithm and can be viewed as preprocessing or postprocessing steps to it.

Given a list of target faults on which the TPG algorithm is to work on, some very useful preprocessing steps include the following:

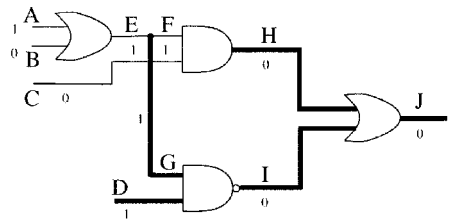
1. *Fault collapsing:* For a circuit with n lines in total, there are $2n$ possible stuck-at faults to consider. Fault collapsing reduces this initial number by taking advantage of equivalence and dominance relations among faults. Two faults are said to be *functionally equivalent* if all patterns that detect the one detect also the other. Given a set of functionally equivalent faults, only one fault from that set has to be considered for test generation. A fault f_1 is said to *dominate* a fault f_2 if all patterns that detect f_2 detect also f_1 and there is at least one pattern that detects f_1 but not f_2. Then only f_2 needs to be considered for test generation. It can be shown that the fault s–a–$(c \oplus i)$ on the output of a gate is functionally equivalent with the fault s–a–c on any of the gate inputs and that the fault s–a–$(\bar{c} \oplus i)$ on the output of a gate dominates the fault s–a–$\bar{c}$ on any of the gate inputs, where c is the controlling value of the gate and i is 1 (0) if the gate is inverting (non-inverting). As an example, using these relations on the circuit of Fig. 67.1, we obtain that (F–s–0, A–s–0, B–s–0), (G–s–1, C–s–1, D–s–1), (J–s–1, G–s–0, I–s–0), (M–s–0, H–s–1, K–s–1), (N–s–0, E–s–0, L–s–0) are functionally equivalent sets of faults, and that F–s–1 dominates A–s–1 and B–s–1, G–s–0 dominates C–s–0 and D–s–0, J–s–0 dominates G–s–1 and I–s–1, M–s–1 dominates H–s–0 and K–s–0, and N–s–1 dominates E–s–1 and L–s–1. Given tese relations, only the set of faults {A–s–1, B–s–1, C–s–0, D–s–0, G–s–1, I–s–1 H–s–0, K–s–0, E–s–1, L–s–1, F–s–0, M–s–0, N–s–0} need be considered; the number of target stuck-at faults is reduced from 28 to 13.
2. *Removal of randomly testable faults:* A very simple way of eliminating faults from a target fault list is to generate test patterns at random and verify, by *fault simulation*, which target faults (if any) each generated pattern detects. The generation of such patterns is done by a *pseudorandom* method, that is, an algorithmic method whose behavior under specific statistical criteria seems close to random. Eliminating all faults by pseudorandom test pattern generation generally requires a very large number of patterns. For instance, under the assumption of uniform input distribution and independent test pattern generation, the smallest number of patterns to detect with probability P_s a fault whose detection probability is d is $N = \left\lceil \frac{\ln(P_s)}{\ln(1-d)} \right\rceil$. In general, faults with small detection probability are referred to as *randomly untestable* or *hard-to-detect* faults, whereas faults with high detection probability are referred to as *randomly testable* or *easy-to-detect* faults. For example, in a circuit consisting of a single k-input AND gate with output line l, the fault l s–a–0 is a hard-to-detect fault as only one out of 2^k patterns can detect it, whereas the fault l s–a–1 is an easy-to-detect fault as $2^k - 1$ out of 2^k patterns can detect it; whereas the fault l s–a–1 is an easy-to-detect fault as $2^h - 1$ out of 2^k patterns can detect it. In practice, an acceptable number of pseudorandom test patterns are generated and simulated in order to drop many easy-to-detect faults from the target fault list, with all remaining faults given over to a *deterministic* (as opposed to pseudorandom) TPG tool, in case a complete test is desired.

3. *Removal of faults identified by critical path tracing:* A *critical path* under an input pattern t is a path from a primary input or internal line to a primary output such that if there is a change in the value under t of any line in the path, the PO also changes (in other words, input pattern t can serve as a test pattern for each fault l s–a–$\bar{v}$, where l is any line of the path and v is the value of that line under t). *Critical path tracing* is a technique for systematically identifying critical paths in a circuit. Starting from an assigned value to a PO (a PO line always constitutes a critical subpath), it works its way back to the PIs trying to extend current critical subpaths. The extension however cannot be done safely through stems of reconvergent fan-out. Given a gate whose output is the beginning of a current critical subpath, the method assigns only one input of the gate to a value c or all inputs of the gate to value $\bar{c}$ in order to justify the output value, where c is the critical value of the gate. In both cases, longer critical subpaths are created that can be developed further recursively. Once the PIs are reached and all non-critical values are justified, all corresponding faults on lines in critical paths are covered by the resulting input pattern, and so these faults can be dropped from the initial fault list. Some critical paths for the circuit of Fig. 67.3 are shown in Fig. 67.5. Notice that stem E in Fig. 67.5(a) is not critical (as found by separate fault simulation), whereas stem E in Fig. 67.5(b) turns out to be actually critical. Critical path tracing can also be viewed as a *fault-independent* (in contrast to *fault-driven*) deterministic TPG algorithm that is generally faster but may not cover all possible detectable faults or prove that a fault is undetectable.

A basic postprocessing step after test patterns have been generated by an ATPG technique is *compaction*. Compaction attempts to reduce the number of patterns by taking advantage of any x values in the patterns generated. The basic step is to merge two patterns which do not have conflicting values in any bit position. For example, in Fig. 67.6(a), we can compact patterns t_1, t_2 and t_3, t_4 to obtain the test set in Fig. 67.6(b), which cannot be compacted further. However, we can also compact patterns t_2, t_3, t_4 and t_1, t_5 to obtain the test set in Fig. 67.6(c), which is smaller than that of Fig. 67.6(b). In general, finding a compacted test set of minimum size is an NP-hard problem, but efficient heuristics exist to solve the problem satisfactorily. Compaction can also be done simultaneously with test pattern generation in order to better exploit



(a)



(b)

FIGURE 67.5 Some critical paths (shown in bold) found by critical path tracing.

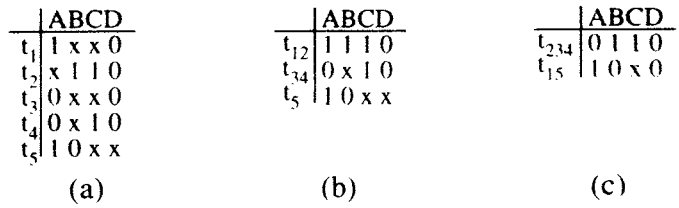


FIGURE 67.6 Compaction of test patterns.

the *x* values as soon as they are generated. This is referred to as *dynamic compaction* (in contrast to *static compaction*), and its basic idea is to assign appropriately any *x* values in the last generated pattern in order to obtain test patterns for additional faults.

67.2 Built-In Self-Test

In order to make the testing of a VLSI circuit easier, several design-for-testability criteria can be taken into account along with the other "traditional" design criteria of cost, delay, area, power, etc. For example, transforming a sequential circuit into combinational parts by linking in a "test mode" all its flip-flops into a shift register so that patterns to initialize the flip-flops can be easily loaded and responses can be observed is a common design-for-testability technique known as *full-scan. Built-in Self-Test* (BIST) is an ultimate design-for-testability technique in which extra circuitry is introduced on-chip in order to provide test patterns to the original circuit and verify its output responses. The aim is to provide a faster and more economic alternative to external testing. The difficulty in the BIST approach is the discovery of schemes which have very low hardware overhead and provide the required test quality in order to justify their inclusion on-chip.

Online BIST

A special form of BIST is the design of *self-checking* circuits in which no explicit test patterns are provided, but the operation of the circuit is tested *online* by identifying any *invalid* output responses (i.e., responses that can never occur under fault-free operation). If, however, there is a fault that can cause a valid response to be changed into another valid response, then that fault cannot be detected. The identification of faulty behavior is done by a special built-in circuit called *checker*. For example, in a k: 2^k decoder, a checker can check if exactly one of the 2^k output lines has a value 1 each time. If the number of 1s in the output pattern is zero or more than one, then an error is detected. If, however, a fault in the decoder causes an input pattern to assert only one output line but not the correct one, then the fault cannot be detected by such a checker. In general, the design of self-checking circuits is based on coding theory. The checker has to encode all output responses of the circuit under fault-free operation in order to distinguish between valid and invalid responses. For example, using the single-bit parity code, a checker can compute the parity of the actual response of the circuit for the current input, compute also the parity of the (known) correct output response corresponding to that input, and compare the two parities.

Faults in the checker can beat the purpose of fault detection in the original circuit. However, the assumption is that the logic of the checker is much simpler than the circuit it checks and therefore can be tested far more easily. Research on the design of *self-checking checkers* seeks to minimize the logic that is not self-testable.

Offline BIST

In a general offline BIST scheme, test pattern generation and application, as well as output response verification, are done by built-in mechanisms while the circuit operates in a test mode.

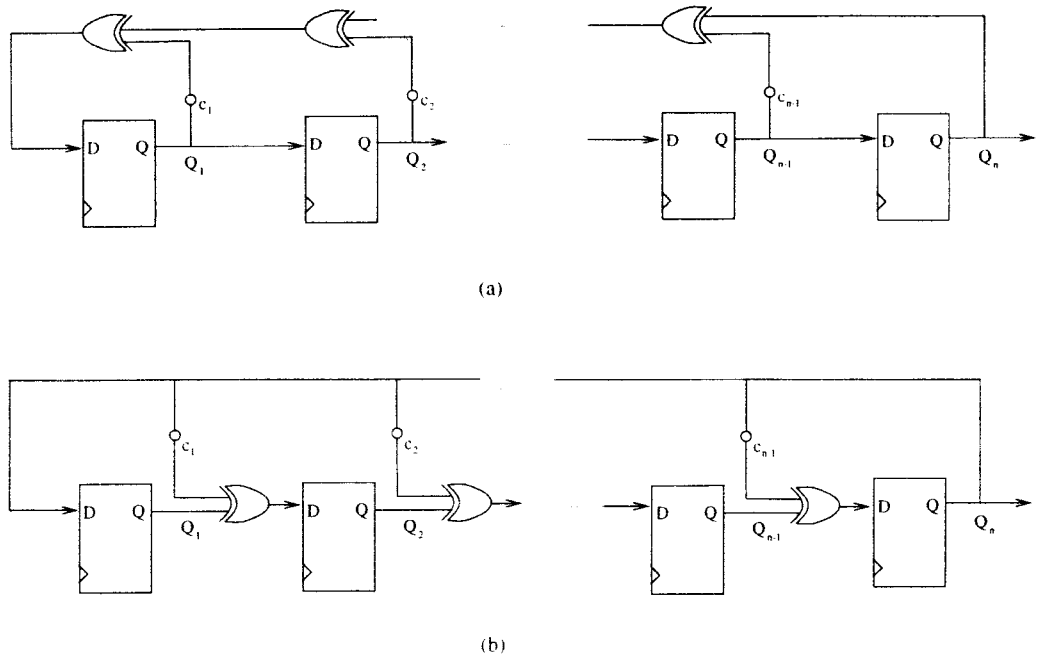


FIGURE 67.7 LFSR configurations.

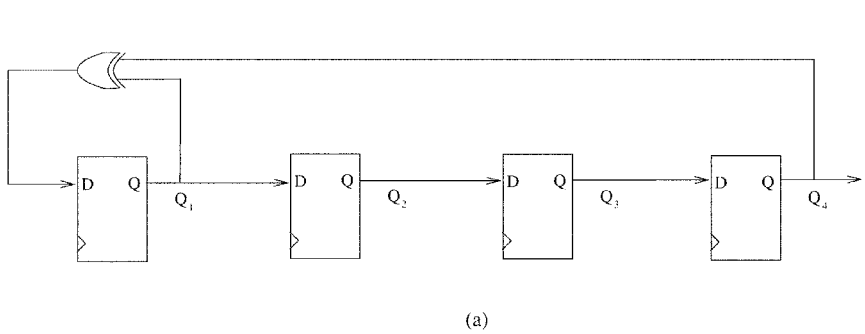
Built-in TPG Mechanisms

Mechanisms that have been considered for built-in test pattern generation and application include read-only memories, counters, cellular automata, and linear feedback shift registers (LFSRs). Of these mechanisms, LFSRs offer the most flexibility and have received the most attention. A *linear feedback shift register* (LFSR) consists of a series of flip-flops connected in a circular structure by means of exclusive-OR (XOR) gates. The two basic types of an LFSR are shown in Fig. 67.7(a) and Fig. 67.7(b).

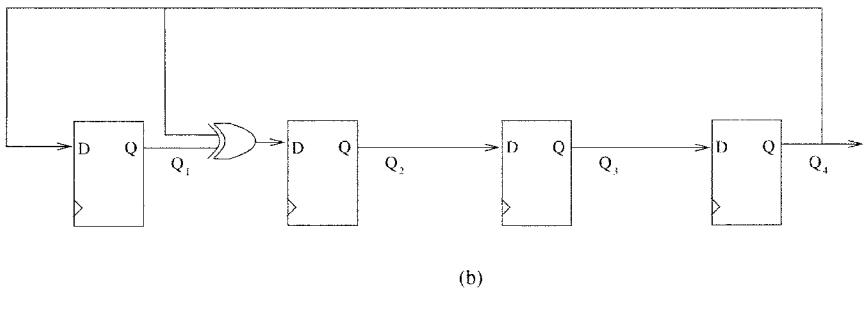
The structure in Fig. 67.7(a) uses the XOR gates externally, while the structure in Fig. 67.7(b) uses the XOR gates internally. The connections of the flip-flops to the XOR gates are fixed for a basic n-bit LFSR and are specified by the values c_i, $1 \le i \le n$, where $c_i = 1$ denotes a connection, and $c_i = 0$ denotes no connection. The specific pattern of c_i values is conveniently represented as a polynomial $P(x) = 1 + \Sigma_{i=1}^{n} c_i x^i$ over the field of elements mod 2 and is referred to as the *characteristic polynomial* of the LFSR. (The representation can also be done by the polynomial $P_r(x) = x^n + \Sigma_{i=1}^{n-1} c_{n-i} x^i$, which is referred to as the *reciprocal* polynomial of $P(x)$.) Given an initial state, an LFSR cycles through a sequence of states as determined by its characteristic polynomial. For particular characteristic polynomials known as *primitive polynomials*, the corresponding sequence of states has the maximum possible length (that is, $2^n - 1$, since the all-0 state will cause the LFSR to cycle through it continuously). A primitive polynomial of degree n has the property that the smallest value k such that $x^k \bmod P(x) = 1$ is $k = 2^n - 1$. Primitive polynomials exist for every degree and a list of them can be found in Ref. 7.

An example of a specific LFSR with characteristic polynomial $P(x) = x^4 + x + 1$, along with the sequence of the resulting states, is given in Fig. 67.8(a) for the external-XOR type and in Fig. 67.8(b) for the internal-XOR type. Although the properties of interest to most BIST applications are the same for the two LFSR types, an external-XOR type LFSR may be slower due to the multiple-level XOR logic. (Notice also that the stae of the external-XOR type LFSR at cycle i (starting from $i = 0$) is exactly the pattern $x^i \bmod P(x)$.)

There are three basic schemes for the design of a built-in test pattern generator: (1) deterministic, (2) pseudorandom, and (3) pseudo-exhaustive.



	Q Q Q Q
t_0	1 0 0 0
t_1	1 1 0 0
t_2	1 1 1 0
t_3	1 1 1 1
t_4	0 1 1 1
t_5	1 0 1 1
t_6	0 1 0 1
t_7	1 0 1 0
t_8	1 1 0 1
t_9	0 1 1 0
t_{10}	0 0 1 1
t_{11}	1 0 0 1
t_{12}	0 1 0 0
t_{13}	0 0 1 0
t_{14}	0 0 0 1



	Q Q Q Q
t_0	1 0 0 0
t_1	0 1 0 0
t_2	0 0 1 0
t_3	0 0 0 1
t_4	1 1 0 0
t_5	0 1 1 0
t_6	0 0 1 1
t_7	1 1 0 1
t_8	1 0 1 0
t_9	0 1 0 1
t_{10}	1 1 1 0
t_{11}	0 1 1 1
t_{12}	1 1 1 1
t_{13}	1 0 1 1
t_{14}	1 0 0 1

FIGURE 67.8 LFSRs with (a) characteristic polynomial $P(x) = x^4 + x + 1$ and (b) resulting sequences.

In *deterministic* TPG, a set of patterns for a list of target faults obtained by a TPG algorithm (after any postprocessing, like compaction) are "embedded" in a TPG mechanism. The obvious solution is to use a read-only memory (ROM) for this purpose, but this is applicable only for very small test sets. An alternative simple solution is to use a binary counter or an LFSR of length w (where w is the test pattern length) that starts from an initial state s_i and cycles through until it reaches another state s_j so that all the desired patterns appear somewhere between states s_i and s_j, with each intermediate state constituting a required or not required pattern. The problem here is to find (if at all) a pair of states s_i, s_j in the sequence produced by the underlying mechanism such that the absolute distance between s_i and s_j is acceptably smaller than 2^w, in order to keep the number of testing cycles acceptably low.

In *pseudorandom* built-in TPG, an LFSR is typically used as a pseudorandom generator, which cycles through a subsequence of l states, each state constituting a pseudorandom pattern, where l is again acceptably low. Such a sequence is analyzed by *fault simulation* in order to determine its *fault coverage* (defined as the ratio of the number of faults that the patterns in the sequence detected over the number of all detectable faults of interest). In general, very long subsequences are needed to achieve an acceptable level of fault coverage. An enhancement of this idea is to use *weighted random* LFSRs. These include extra logic in order to change the bit probabilities in the states that the LFSR generates. For example, by having bit i of each test pattern be the output of an AND gate driven by two LFSR bits, the probability of having a '1' in bit i is the product of the probabilities of having a '1' in those LFSR bits.

In *pseudo-exhaustive* built-in TPG, the goal is to reduce the testing of the circuit to the testing of appropriate subcircuits of it such that each subcircuit depends on a small number of primary inputs, then apply all possible patterns to each of these subcircuits. The benefits of an exhaustive test set is that no test pattern generation or fault simulation is needed and that the generated patterns guarantee that all detectable faults that do not induce sequential behavior are detected. In order for pseudo-exhaustive TPG to achieve the benefits of exhaustive testing without taking prohibitive time, particular relations must hold between the primary outputs (POs) and the primary inputs (PIs) on which they depend. If

such relations do not hold, they may be imposed upon the circuit through design-for-testability techniques.

In general, there are many pseudo-exhaustive test sets that can be obtained for a given circuit. The goal in pseudo-exhaustive built-in TPG is to find and embed a pseudo-exhaustive test set that offers the best tradeoff in hardware implementation cost and testing time.

As a simple example of how a pseudo-exhaustive test set can be obtained, consider a circuit with n inputs and one output fed by a two-input gate whose inputs are driven in turn by two disjoint subcircuits. Then, that output can be tested pseudo-exhaustively by $2^{n_1} + 2^{n_2} + 1$ patterns instead of 2^n, where n_1 and n_2 are the numbers of the (disjoint) primary inputs that drive the two subcircuits. The first 2^{n_1} of these patterns contain a constant subpattern (consisting of n_2 bits) required to sensitize the paths from the first subcircuit to the output; the next 2^{n_2} of these patterns contain a constant subpattern (consisting of n_1 bits) required to sensitize the paths from the second subcircuit to the output; and the last pattern is required to provide both inputs of the gate with the controlling value of the gate. This pseudo-exhaustive test set could be generated on-chip by using, for instance, a counter and some extra storage for the constant subpatterns, but such pseudo-exhaustive test sets can be impractical to implement in large circuits.

Obtaining suitable pseudo-exhaustive test sets for built-in implementation is based on the consideration of the subsets of PIs on which each PO depends. Let us call such a set a D-set. All D-sets must be smaller than the number n of PIs; otherwise, pseudo-exhaustive testing is not applicable. A general preprocessing step for pseudo-exhaustive TPG is to identify groups of PIs that never appear together in a D-set. All PIs in such a group can share the same test signal for the pseudo-exhaustive testing. In this way, the number of test signals is reduced from n to n', with an immediate reduction of the test time from 2^n to $2^{n'}$. Minimizing the value of n' is an NP-hard problem, but efficient heuristics exist to reduce it in practice.

Pseudo-exhaustive test sets can be obtained by considering only the size $k < n$ of the maximum D-set in a circuit and ignoring the structure of the D-sets as well as their number (i.e., such pseudo-exhaustive test sets are good for any n-input circuit with no output being dependent on more than k inputs). For example, it has been shown[8] that a test set that comprises all binary patterns containing w_1 '1's, all binary patterns containing w_2 '1's, etc., up to w_i '1's, where w_1, w_2, …, w_i are all the solutions of the equation $w = c \bmod (n - k + 1)$, for some constant $c \le n - k$, constitute a pseudo-exhaustive test set. For instance, if $n = 6$ and $k = 3$, the set of all patterns with 0 or 4 '1's (corresponding to $c = 0$), the set of all patterns with 1 or 5 '1's (corresponding to $c = 2$), the set of all patterns with 2 or 6 '1's (corresponding to $c = 2$), the set of all patterns with 3 '1's (corresponding to $c = 3$) constitute pseudo-exhaustive test sets that can be applied to any circuit with n inputs and maximum D-set size k. The structure of one of these sets (corresponding to $c = 2$) is given in Fig. 67.9. The generation of such a set of patterns can be done by using *constant-weight counters*, which produce a sequence of states with the same constant number of '1's in each. The disadvantages of this approach are the size of the test set which, although not 2^n, is still large $\left(\approx \frac{2^n}{n-k+1}\right)$, and the hardware overhead required for the implementation of a constant-weight counter.

Better solutions may be obtained by considering the particular structure of each D-set. A very important mechanism in this regard is the Extended LFSR. An Extended LFSR (also known as LFSR/SR) is a shift register (SR) of n cells whose initial k cells are configured into an LFSR with a characteristic polynomial of degree k. Let $P(x)$ be that characteristic polynomial. It has been shown (see, e.g., Ref. 9) that the successive states of such an LFSR/SR test exhaustively a D-set $D = \{d\_1, d\_2, \ldots, d\_s\}$, $s = |D|$ (the d_i elements denote the indices of the cells that drive the circuit inputs), if an only if the set of vectors $x^{d1} mod P(x)$, $x^{d2} mod P(x)$, …, $x^{ds} mod P(x)$ are linearly independent. If this relation holds for every D-set, then the corresponding test sequence tests the circuit pseudo-exhaustively in time 2^k (after the initialization of the LFSR and SR parts of the LFSR/SR). As an example, consider the D-sets $D_1 = \{1, 2, 3, 4\}$, $D_2 = \{2, 3, 5\}$, $D_3 = \{3, 5, 6\}$. All these D-sets satisfy the above relation under primitive polynomial $P(x) = x^4 + x + 1$ (see Fig. 67.10(a)). However, if a D-set $D_4 = \{1, 2, 5\}$ were also present, that D-set could no more be tested pseudo-exhaustively, as its corresponding vectors are linearly dependent (see Fig. 67.10(b)).

Obtaining an LFSR/SR under which the independency relation holds for every D-set of the circuit involves basically a search for an applicable polynomial of degree d, $k \le d \le n$, among all primitive polynomials of degree d, $k \le d \le n$. Primitive polynomials of any degree can be algorithmically generated. An applicable polynomial of degree n is, of course, bound to exist (this corresponds to exhaustive testing), but in order to keep the number of test cycles low, the degree should be minimized.

	ABCDEF
t_1	110000
t_2	101000
t_3	100100
t_4	100010
t_5	100001
t_6	011000
t_7	010100
t_8	010010
t_9	010001
t_{10}	001100
t_{11}	001010
t_{12}	001001
t_{13}	000110
t_{14}	000101
t_{15}	000011
t_{16}	111111

FIGURE 67.9 A pseudo-exhaustive test set for any circuit with six inputs and largest D-set size 3.

Built-in Output Response Verification Mechanisms

Verification of the output responses of a circuit under a set of test patterns consists, in principle, of comparing each resulting output value against the correct one, which has been precomputed and prestored for each test pattern. However, for built-in output response verification, such an approach cannot be used (at least for large test sets) because of the associated storage overhead. Rather, practical built-in output response verification mechanisms rely on some form of *compression* of the output responses so that only the final compressed form needs to be compared against the (precomputed and prestored) compressed form of the correct output response. Some representative built-in output response verification mechanisms based on compression are given below.

1. Ones count: In this scheme, the number of times that each output of the circuit is set to '1' by the applied test patterns is counted by a binary counter, and the final count is compared against the corresponding count in the fault-free circuit.
2. Transition count: In this scheme, the number of transitions (i.e., changes from both $0 \rightarrow 1$ and $1 \rightarrow 0$) that each output of the circuit goes through when the test set is applied is counted by a binary counter and the final count is compared against the corresponding count in the fault-free circuit. (These counts must be computed under the same ordering of the test patterns.)
3. Signature analysis: In this scheme, the specific bit sequence of responses of each output is represented as a polynomial $R(x) = r_0 + r_1x + r_2x^2 + \ldots + r_{s-1}x^{s-1}$, where r_i is the value that the output takes under pattern t_i, $0 \le i \le s$, and s is the total number of patterns. Then, this polynomial is divided by a selected polynomial $G(x) = g_0 + g_1x + g_2x^2 + \ldots + g_mx^m$ of degree m for some desired

	$x^1x^2x^3x^4$
	0 0 1 0
$D_1 = \{1,2,3,4\}$	0 1 0 0
	1 0 0 1
	0 0 0 1

	$x^2x^3x^5$
	0 1 0
$D_2 = \{2,3,5\}$	1 0 1
	0 0 1
	0 0 0

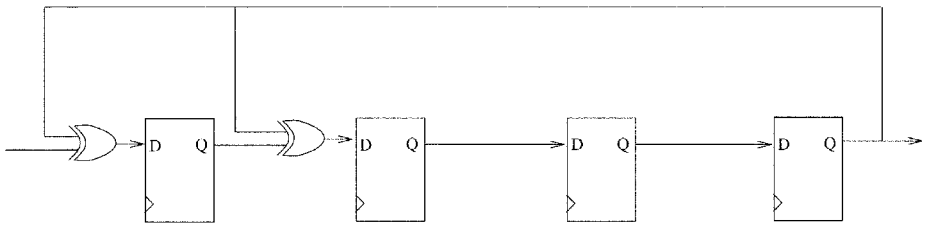
	$x^3x^5x^6$
	1 0 1
$D_3 = \{3,5,6\}$	0 1 1
	0 1 0
	0 0 0

(a)

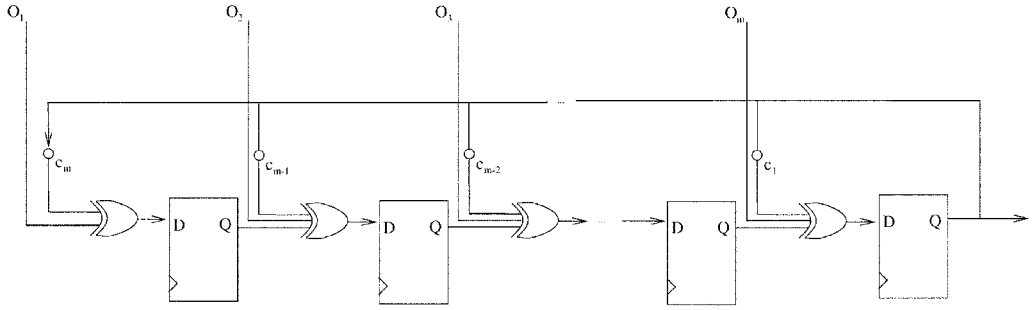
	$x^1x^2x^5$
	0 0 0
$D_4 = \{1,2,5\}$	0 1 1
	1 0 1
	0 0 0

(b)

FIGURE 67.10 Linear independence under $P(x) = x^4 + x + 1$: (a) D-sets that satisfy the condition; (b) A D-set that does not satisfy the condition.



(a)



(b)

FIGURE 67.11 (a) Structure for division by $x^4 + x + 1$; (b) general structure of an MISR.

value m, and the remainder of this division (referred to as *signature*) is compared against the remainder of the division by $G(x)$ of the corresponding fault-free response $C(x) = c_0 + c_1x + c_2x^2 + \ldots + c_{s-1}x^{s-1}$. Such a division is done efficiently in hardware by an LFSR structure such as that in Fig. 67.11(a). In practice, the responses of all outputs are handled together by an extension of the division circuit, known as *multiple-input signature register* (MISR). The general form of a MISR is shown in Fig. 67.11(b).

In all compression techniques, it is possible for the compressed forms of a faulty response and the correct one to be the same. This is known as *aliasing* or *fault masking*. For example, the effect of aliasing in ones count output response verification is that faults that cause the overall number of '1's in each output to be the same as in the fault-free circuit are not going to be detected after compression, although the appropriate test patterns for their detection have been applied. In general, signature analysis offers a very small probability of aliasing. This is due to the fact that an erroneous response $R(x) = C(x) = E(x)$, where $E(x)$ represents the error pattern (and addition is done mod 2), will produce the same signature as the correct response $C(x)$ and only if $E(x)$ is be a multiple of the selected polynomial $G(x)$.

BIST Architectures

BIST strategies for systems composed of combinational logic blocks and registers generally rely on partial modifications of the register structure of the system in order to economize on the cost of the required mechanisms for TPG and output response verification. For example, in the BILBO (Built-In Logic Block Observer) scheme,[10] each register that provides input to a combinational block and receives the output

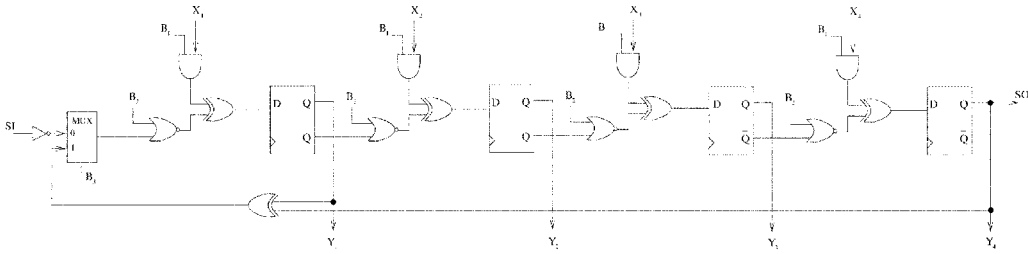


FIGURE 67.12 BILBO structure for a 4-bit register.

of another combinational block is transformed into a multipurpose structure that can act as an LFSR (for test pattern generation), as an MISR (for output response verification), as a shift register (for scan chain configurations), and also as a normal register. An implementation of the BILBO structure for a 4-bit register is shown in Fig. 67.12. In this example, the characteristic polynomial for the LFSR and MISR is $P(x) = x^4 + x + 1$.

By setting $B_1B_2B_3 = 001$, the structure acts like an LFSR. By setting $B_1B_2B_3 = 101$, the structure acts like an MISR. By setting $B_1B_2B_3 = 000$, the structure acts like a shift register (with serial input SI and serial output SO). By setting $B_1B_2B_3 = 11x$, the structure acts like a normal register, and by setting $B_1B_2B_3 = 01x$, the register can be cleared.

As two more representatives of system BIST architectures, we mention here the STUMPS scheme,[11] where each combinational block is interfaced to a scan path and each scan path is fed by one cell of the same LFSR and feeds one cell of the same MISR, and the LOCST scheme,[12] where there is a single boundary scan chain for inputs and a single boundary scan chain for outputs, with an initial portion of the input chain configured as an LFSR and a final portion of the output chain configured as an MISR.

References

1. J.P. Roth, W.G. Bouricious, and P.R. Schneider, Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits, *IEEE Trans. Electronic Computers*, 16, 567, 1967.
2. P. Goel, An implicit enumeration algorithm to generate tests for combinational logic circuits, *IEEE Trans. Computers*, 30, 215, 1981.
3. M.R. Garey and D.S. Johnson, *Computers and Intractability – A Guide to The Theory of NP-Completeness*, W.H. Freeman and Co., New York, 1979.
4. H. Fujiwara and T. Shimono, On the acceleration of test generation algorithms, *IEEE Trans. Computers*, 32, 1137, 1983.
5. M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, New York, 1990.
6. R.A. Marlett, EBT: A comprehensive test generation technique for highly sequential circuits, *Proc. 15th Design Automation Conf.*, 335, 1978.
7. W.W. Peterson and E.J. Weldon, Jr., *Error-Correcting Codes*, MIT Press, Cambridge, MA, 1972.
8. Tang, D.T. and Woo, L.S., Exhaustive test pattern generation with constant weight vectors, *IEEE Trans. Computers*, 32, 1145, 1983.
9. Z. Barzilai, Coppersmith, D., and Rosenberg, A.L., Exhaustive generation of bit patterns with applications to VLSI testing, *IEEE Trans. Computers*, 32, 190, 1983.
10. B. Koenemann, J. Mucha, and G. Zwiehoff, Built-in test for complex digital integrated circuits, *IEEE J. Solid State Circuits*, 15, 315, 1980.
11. P.H. Bardell and W.H. McAnney, Parallel pseudorandom sequences for built-in test, in *Proc. Intern'l. Test. Conf.*, 302, 1984.
12. J. LeBlanc, LOCST: A built-in self-test technique, *IEEE Design and Test of Computers*, 1, 42, 1984.

Tragoudas, S. "CAD Tools for BIST/DFT and Delay Faults"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

68

CAD Tools for BIST/DFT and Delay Faults

Spyros Tragoudas
Southern Illinois University

68.1 Introduction

This chapter describes computer-aided design (CAD) tools and methodologies for improved design for testability (DFT), built-in self-test (BIST) mechanisms, and fault simulation. Section 68.2 presents CAD tools for the traditional stuck-at fault model which was examined in Chapters 66 and 67. Section 68.3 describes a fault model suitable for delay faults — the path delay fault model. The number of path delay faults in a circuit may be a non-polynomial quantity. Thus, this fault model requires sophisticated CAD tools not only for BIST and DFT, but also for ATPG and fault simulation.

68.2 CAD for Stuck-at Faults

In the traditional stuck-at model, each line in the circuit is associated to at most two faults, a stuck-at 0 and a stuck-at 1 fault. We distinguish between combinational and sequential circuits. In the former case, *computer-aided design* (CAD) tools target efficient synthesis of BIST schemes. The testing of sequential circuits is by far a more difficult problem and must be assisted by DFT techniques. The most popular DFT approach is the scan design. The following subsections present CAD tools for combinational logic and sequential logic, and then a review of advances in fault simulation.

Synthesis of BIST Schemes for Combinational Logic

The Pseudo-exhaustive Approach

In the pseudo-exhaustive approach, patterns are generated pseudorandomly and target all possible faults. A common circuit preprocessing routine for CAD tools is called *circuit segmentation.*

The idea in circuit segmentation is to insert a small number of storage elements in the circuit. These elements are bypassed in operation mode — that is, they function as wires — but in testing mode, they are part of the BIST mechanism. Due to their dual functionality, they are called *bypass storage elements* (*bses*). The hardware overhead of a *bse* amounts to that of a flip-flop and a two-to-one multiplexer. Each

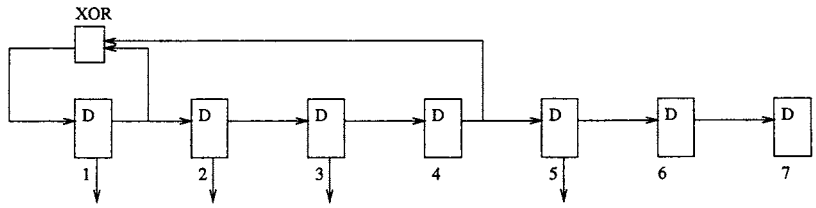


FIGURE 68.1 An observable point that depends on four controllable points.

bse is a controllable as well as an observable point, and must be inserted so that every observable point (primary output or *bse*) depends on at most k controllable points (primary inputs or *bses*), where k is an input parameter not larger than 25. This way, no more than 2^k patterns are needed to pseudo-exhaustively test the circuit.

The circuit segmentation problem is modeled as a combinational minimization problem. The objective function is to minimize the number of inserted *bses* so that each observable point depends on at most k controllable points. The problem is NP-hard in general.[1] However, efficient CAD tools have been proposed.[2-4] In Ref. 2, the *bse* insertion tool minimizes the hardware overhead using a *greedy methodology*. The CAD tool in Ref. 3 uses *iterative improvement*, and the one in Ref. 4 the concept of *articulation points*.

When the *test pattern generation* (TPG) is an LFSR/SR with a characteristic polynomial $P(x)$ with period $\mathcal{P}$, $\mathcal{P} \geq 2^k - 1$, *bse* insertion must be guided by a sophisticated CAD tools which guarantees that the $\mathcal{P}$ different patterns that are generated by the LFSR/SR suffice to test the circuit pseudo-exhaustively. This in turn implies that each observable point which depends on at most k controllable points must receive $2^k - 1$ patterns. (The all-zero input pattern is excluded because it cannot be generated by the LFSR/SR.) The example below illustrates the problem.

Example 1

Consider the LFSR/SR of Fig. 68.1, which has seven cells. In this case, the total number of primary inputs and inserted *bses* is seven. Consider a consecutive labeling of the LFSR/SR cells in the range [1…7], where the left-most element takes label 1. Assume that an observable point o in the circuit depends on elements 1, 2, 3, and 5 of the LFSR/SR. In this case, $k \geq 4$, and the input dependency of o is represented by the set $I_o = \{1, 2, 3, 5\}$.

Let the characteristic polynomial of the LFSR/SR be $P(x) = x^4 + x + 1$. This is a primitive polynomial and its period $\mathcal{P}$ is $\mathcal{P} = 2^4 - 1 = 15$. We list in Table 68.1 the patterns generated by $P(x)$ when the initial seed is 00010.

Any seed besides 00000 will return $2^4 - 1$ different patterns. Although 15 different patterns have been generated, the observable point o will received the set of subpatterns projected by columns 1, 2, 3, and 5 of the above matrix. In particular, o will receive patterns in Table 68.2.

TABLE 68.1

0	0	0	1	0
1	0	0	0	1
1	1	0	0	0
1	1	1	0	0
1	1	1	1	0
0	1	1	1	1
1	0	1	1	1
0	1	0	1	1
1	0	1	0	1
1	1	0	1	0
0	1	1	0	1
0	0	1	1	0
1	0	0	1	1
0	1	0	0	1
0	0	1	0	0

Although 15 different patterns have been generated by $P(x)$, point o receives only eight different patterns. This happens because there exists at least one linear combination in the set $\{x^1, x^2, x^3, x^5\}$, the set of monomials of o, which is divided by $P(x)$. In particular, the linear combination $x^5 + x^2 + 1$ is divisible by $P(x)$. If no linear combination is divisible by $P(x)$, then o will receive as many different patterns as the period of the characteristic polynomial $P(x)$.

For each linear combination in some set I_o which is divisible by the characteristic polynomial $P(x)$, we say that a *linear dependency* occurs. Avoiding linear dependencies in the set I_o sets is a fundamental problem in pseudo-exhaustive built-in TPG. The following describes CAD tools for avoiding linear dependencies.

The approach in Ref. 3 proposes that the elements of the LFSR/SR (inserted *bses* plus primary inputs) are assigned appropriate labels in the LFSR/SR. It has

been easily shown that no linear combination in some I_o is divisible by $P(x)$ if the largest label in I_o and the smallest label in I_o differ by less than k units.[3] We call this property the *k-distance property in set* I_o. Reference 3 presents a coordinated scheme that segments the circuit with *bse* insertion, and labels all the LFSR/SR cells so that the k-distance property is satisfied for each set I_o.

TABLE 68.2

0	0	0	0
1	0	0	1
1	1	0	0
1	1	1	0
1	1	1	0
0	1	1	1
1	0	1	1
0	1	0	1
1	0	1	1
1	1	0	0
0	1	1	1
0	0	1	0
1	0	0	1
0	1	0	1
0	0	1	0

It is an NP-hard problem to minimize the number of inserted *bses* subject to the above constraints. This problem contains a special case the traditional circuit segmentation problem. Furthermore, Ref. 3 shows that it is NP-complete to decide whether an appropriate LFSR/SR cell labeling exists so that k-distance property is satisfied for each set I_o without considering the circuit segmentation problem, that is, after *bse* elements have been inserted so that for each set I_o it holds that $|I_o| \leq k$. However, Ref. 3 presents an efficient heuristic for the k-distance property problem. It is reduced to the bandwidth minimization problem on graphs for which many efficient polynomial time heuristics have been proposed.

The outline of the CAD tool in Ref. 3 is as follows. Initially, *bse* elements are inserted so that for each set I_o, we have that $|I_o| \leq k$. Then, a bandwidth-based heuristic determines whether all sets I_o could satisfy the k-distance property. For each I_o that violates the k-distance property, a modification is proposed by recursively applying a greedy *bse* insertion scheme, which is illustrated in Fig. 68.2.

The primary inputs (or inserted *bses*) are labeled in the range [1…6], as shown in the Fig. 68.2. Assume that the characteristic polynomial is $P(x) = x^4 + x + 1$, i.e., $k = 4$. Under the given labeling, sets I_e and I_d satisfy the k-distance property but set I_g violates it. In this case, the tool finds the closest front of predecessors of *g* that violate the k-distance property. This is node *f*. New *bses* are inserted on the incoming edges if *f*. (The tool may attempt to insert *bses* on a subset of the incoming edges.) These *bses* are assigned labels 7, 8. In addition, 4 is relabeled to 6, and 6 to 4. This way, I_g satisfies the k-distance requirement.

The CAD tool can also be executed so that instead of examining the k-distance, it examines instead if each set I_o has at least one linear dependency. In this case, it finds the closest front of predecessors that contain some linear dependency, and inserts *bse* elements on their incoming edges. This approach increases the time performance without significant savings in the hardware overhead.

The reason that primitive polynomials are traditionally selected as characteristic polynomials of LFSR/SRs is that they have large period $\mathcal{P}$. However, any polynomial could serve as a characteristic polynomial of the LFSR/SR as long as its period $\mathcal{P}$ is no less than $2^k - 1$. If P is less than $2^k - 1$, then no set I_o with $|I_o| = k$ can be tested pseudo-exhaustively.

A desirable characteristic polynomial would be one that has large period $\mathcal{P}$ and whose multiples obey a given pattern which we could try to avoid when relabeling the cells of the LFSR/SR so that appropriate I_o sets are formed. This is the idea of the CAD tool in Ref. 5.

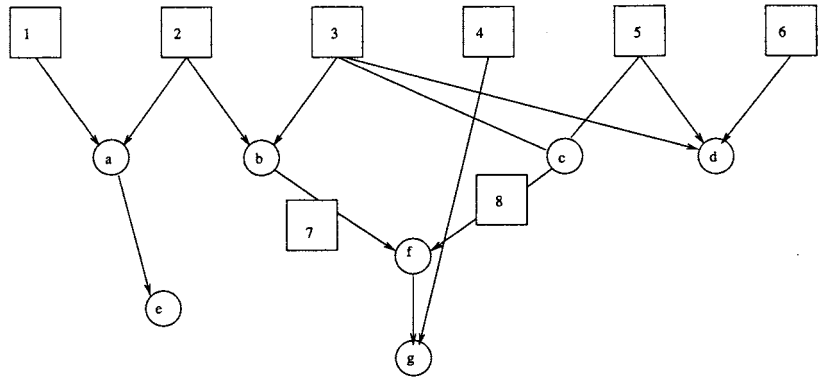


FIGURE 68.2 Enforcing the *k*-distance property with *bse* insertion.

In particular, Ref. 5 proposes that the characteristic polynomial is a product $P(x) = P_1(x) \cdot P_2(x)$ of two polynomials. $P_1(x)$ is a primitive polynomial of degree k which guarantees that the period of the characteristic polynomial $P(x)$ is at least $2^k - 1$. $P_2(x)$ is the polynomial $x^d + x^{d-1} + x^{d-2} + \ldots + x^1 + x^0$, whose degree d is determined by the CAD tool. $P_2(x)$ is called a *consecutive polynomial of degree d*. The CAD tool determines which primitive polynomial of degree d will be implemented in $P(x)$.

The multiples of consecutive polynomials have a given structure. Consider an $I_o = \{i_1, i_2, \ldots, i_k\}$ and $I'_o = \{i'_1, i'_2, \ldots, i'_{k'}\} \subseteq I_k$. Ref. 5 shows that there is no linear combination in set $I'o$ if the parity of all remainders of each $i'_j \in I'_o$ modulo d-1 is either even or odd. In more details, the algorithm groups all i'_j whose remainder modulo d-1 is x under list L_x, and then checks the parity of the list L_x. There are d lists labeled L_0 through L_{d-1}. If not all list parities agree, then there is no linear combination in I'_o. (If a list L_x is empty, it has even parity.) The example below illustrates the approach.

Example 2

Let $I_o = \{27, 16, 5, 3, 1\}$ and $P_2(x) = x^4 + x^3 + x^2 + x + 1$. Lists L_3, L_2, L_1 and L_0 are constructed, and their parities are examined. Set I_o contains linear dependencies because in subset $I'_o = \{27, 3\}$, there are even parities in all lists. In particular, list L_3 has two elements and all the remaining lists are empty.

However, there are no linear independencies in the subset $I'_o = \{16, 3, 1\}$. In this case, L_0, L_1, and L_3 have exactly one element each, and L_2 is empty. Therefore, there is no subset of I'_o where all L_i, $0 \le i \le 3$, have the same parity.

The performance of the approach in Ref. 5 is affected by the relative order of the LFSR/SR cells. Given a consecutive polynomial of degree d, one LFSR/SR cell labeling may give linear dependencies in some I_o whereas an appropriate relabeling may guarantee that no linear dependencies occur in any set I_o. Ref. 5 shows that it is an NP-complete problem to determine whether a relabeling exists so that no linear dependencies occur in any set I_o.

The idea of Ref. 5 is to label the LFSR/SR cells so that a small fraction of linear dependencies exist in each set I_o. In particular, for each set I_o, the approach returns a large subset I'_o with no linear dependencies with respect to polynomial $P_2(x)$. This is promise for pseudorandom built-in TPG. The objective is relaxed so that each set I_o receives many different test patterns. Experimentation in Ref. 5 shows that the smaller the fraction of linear dependencies in a set, the larger fraction of different patterns will receive. Also observe that many linear dependencies can be filtered out by the primitive polynomial $P_1(x)$.

A final approach for avoiding linear dependencies was proposed in Ref. 4. The idea is also to find a maximal subset I'_o of each I_o where no linear dependencies occur. The maximality of I'_o is defined with respect to linear independencies, that is, I'_o cannot be further expanded by adding another label a without introducing some linear dependencies. It is then proposed that cell a receives another label a' (as small as possible) which guarantees that there are no linear dependencies in $I'_o \cup \{a\}$. This may cause many "dummy" cells in the LFSR/SR (i.e., labels that do not belong to any I_o). Such dummy cells are subsequently removed by inserting XOR gates.

The Deterministic Approach

In this section we discuss BIST schemes for deterministic test pattern generation, where the generated patterns target a given list of faults. An initial set T of test patterns is traditionally part of the input instance. Set T has been generated by an ATPG tool and detects all the random resistant faults in the circuit.

The goal in deterministic BIST is to consult T and, within a short period of time, generate patterns on-chip which detect all random pattern resistant faults. The BIST scheme may be reproduced by a subset of the patterns in T as well as patterns not in T. If *all* the patterns of T are to be reproduced on-chip, then the mechanism is also called a *test set embedding scheme*. (In this case, only the patterns of T need to be reproduced on-chip.) The objective in test set embedding schemes is well defined, but the reproduction time or the hardware overhead may be less when we do not insist that all the patterns of T are reproduced on-chip.

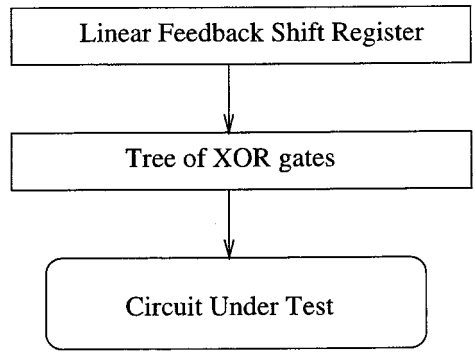


FIGURE 68.3 The schematic of a weighted random LFSR.

A very popular method for deterministic on-chip TPG is to use *weighted random LFSRs*. A weighted random LFSR consists of a simple LFSR/SR and a tree of XOR gates, which is inserted between the cells of the LFSR/SR and the inputs of the circuit under test, as Fig. 68.3 indicates. The tree of XOR gates guarantees that the test patterns applied to the circuit inputs are weighted with appropriate signal probabilities (probability of logic "1").

The idea is to weigh random test patterns with non-uniform probability distributions in order to improve detectability of random pattern resistant faults. The test patterns in T assist in assigning weights. The signal probability of an input is also referred to as the *weight* associated with that input. The collection of weights on all inputs of a circuit is called a *weight set*. Once a weight set has been calculated, the XOR tree of the weighted LFSR is constructed.

Many weighted random LFSR synthesis schemes have been proposed in the literature. Their syntheses mainly focuses on determining the weight set, thus the structure of the XOR tree. Recent approaches consider multiple weight sets. In Ref. 6, it has been shown that patterns with *small Hamming distance* are easier to be reproduced by the same weight set. This observation forms the basis of the approach which works in sessions.

A session starts by generating a weight set for a subset T' of patterns T with small *Hamming distance* from a given *centroid pattern in the subset*. Subsequently, the XOR tree is constructed and a characteristic polynomial is selected which guarantees high fault coverage. Next, fault simulation is applied and it is determined how many faults remain undetected. If there are still undetected faults, an *automatic test pattern generator* (ATPG) is activated, and a new set of patterns T is determined for the next session; otherwise, the CAD tool terminates.

For the test set embedding problem, weighted random LFSRs are not the only alternative. Binary counters may turn out to be a powerful BIST structure that requires very little hardware overhead. However, their design (synthesis) must be supported by sophisticated CAD tools that quickly and accurately determine the amount of time needed for the counter to reproduce a test matrix T on-chip. Such a CAD tool is described in Ref. 7, and recommends whether a counter may be suitable for the test embedding problem on a given circuit. The CAD tool in Ref. 7 designs a counter which reproduces T within a number of clock cycles that is within a constant factor from the smallest possible by a binary counter.

Consider a test matrix T of four patterns, consisting of eight columns, labeled 1 through 8. (The circuit under test has eight inputs.) A simple binary counter requires 125 clock cycles to reproduce these four patterns in a straightforward manner. The counter is seeded with the fourth pattern and incrementally will reach the second pattern, which is the largest, after 125 cycles. Instead, the

TABLE 68.3

1	0	1	0	1	1	0	1
1	0	1	1	1	1	0	1
1	0	1	0	1	1	1	1
0	1	0	0	0	0	0	0

CAD tool in Ref. 7 synthesizes the counter so that only 4 clock cycles are needed for reproducing on-chip these four patterns.

The idea is that matrix T can be manipulated appropriately. The following operations are allowed on T:

- Any constant columns (with all 0 or all 1) can be eliminated, since ground and power wires can be connected to the respective inputs.
- Merging of any two complimentary columns. This operation is allowed because the same counter cell (enhanced flip-flop) has two states Q and Q'. Thus, it can produce (over successive clock cycles) a column as well as its complement.
- Many identical columns (and respective complementary) can be merged into a single column since the output of a single counter cell can fan-out to many circuit inputs. However, due to delay considerations we do not allow more than a given number f of identical columns to be merged. Bound f is an input parameter in the CAD tool.
- Columns can be permuted. This corresponds to reordering of the counter cells.
- Any column can be replaced by its complementary column.

These five operations can be applied on T in order to reduce the number of clock cycles needed for reproducing it. The first three operations can be applied easily in a preprocessing step. In the presence of column permutation, the problem of minimizing the number of required clock cycles is NP-hard. In practice, the last two operations drastically reduce the reproduction time. The impact of column permutation is shown in the example in Table 68.4.

TABLE 68.4

1	0	1	0	1	1	0	1		0	1	1	1	1	1	0	0
1	0	1	1	1	1	0	1		0	1	1	1	1	1	0	0
1	0	1	0	1	1	1	1		0	1	1	1	1	1	0	1
0	1	0	0	0	0	0	0		1	0	0	0	0	0	0	0

The matrix on the left needs 125 cycles to be reproduced on-chip. The column permutation shown to the right reduces the reproduction time to only four cycles.

The idea of the counter synthesis CAD tool is to place as many identical columns as possible as the rightmost columns of the matrix. This set of columns can be preceded by a complementary column, if one exists. Otherwise, the first of the identical columns is complemented. The remaining columns are permuted so that a special condition is enforced, if possible.

The example in Table 68.5 illustrates the described algorithm. Consider matrix T given in Table 68.5.

TABLE 68.5

1	0	0	0	0	1	1	0	1	1	1	1
1	1	0	1	1	0	1	0	1	1	1	1
0	1	1	0	0	0	0	1	0	0	0	0
1	1	0	1	1	0	1	1	0	0	0	1
1	1	0	0	0	0	1	1	0	0	0	1
0	0	1	0	1	1	0	1	0	0	0	1

Assume that $f = 1$, that is, no fan-out stems are required. The columns are permuted as given in Table 68.6.

The leading (right-most) four columns are three identical columns and a complementary column to them. These four leading columns partition the vectors into two parts. Part 1 consists of the first two vectors with prefix 0111. Part 2 contains the remaining vectors. Consider the subvectors of both parts in the partition, induced when removing the leading columns. This set of subvectors (each has eight bits) will determine the relative order of the remaining columns of T.

TABLE 68.6

0	1	1	1	1	1	1	0	1	0	0	0
0	1	1	1	1	1	1	0	0	1	1	1
1	0	0	0	0	0	0	1	0	1	0	0
1	0	0	0	1	1	1	0	0	1	1	1
1	0	0	0	1	1	1	0	0	1	0	0
1	0	0	0	1	0	0	1	1	0	0	1

The unassigned eight columns are permuted and complemented (if necessary) so that the smallest subvector in part 1 is not smaller than the largest subvector in part 2. We call this conduction, the *low order condition*. The column permutation in Table 68.6 satisfies the low order condition. In this example, no column needs to be complemented in order for the low order condition to be satisfied.

The CAD tool in Ref. 7 determines in polynomial time whether the columns can be permuted or complemented so that the low order condition is satisfied. If it is satisfied, it is shown that the amount of required clock cycles for reproducing T is within a factor of two from the minimum possible. This also holds when the low order condition cannot be satisfied.

A test matrix T may contain don't-cares. Don't-cares are assigned so that we maximize the number of identical columns in T. This problem is shown to be NP-hard.[7] However, an assignment that maximizes the number of identical columns is guided by efficient heuristics for the maximum independent set problem on a graph $G = (V, E)$, which is constructed in the following way.

For each column c of T, there exists a node $v_c \in V$. In addition, there exists an edge between a pair of nodes if and only if there exists at least one column where one of the two columns has 1 and the other has 0. In other words, there exists an edge if and only if there is no don't-care assignment that makes the respective columns identical. Clearly, $G = (V, E)$ has an independent set of size k if and only if there exists a don't-care assignment that makes the respective columns of T identical. The operation of this CAD tool is illustrated in the example below.

Example 3

Consider matrix T with don't-cares and columns labeled c_1 through c_6 in Table 68.7. In graph $G = (V, E)$ of Fig. 68.4, node i corresponds to column c_i, $1 \le i \le 6$. Nodes 3, 4, 5, and 6 are independent. The matrix to the left below shows the don't-care assignment on columns c_3, c_4, c_5, and c_6. The don't-care assignment on the remaining columns (c_1 and c_2) is done as follows. First, it is attempted to find a don't-care assignment that makes either c_1 or c_2 complementary to the set of identical columns $\{c_3, c_4, c_5, c_6\}$. Column c_2 satisfies this condition. Then, columns c_2, c_3, c_4, c_5 and c_6 are assigned to the left-most positions of T. As described earlier, the test patterns of T are now assigned in two parts. Part 1 has patterns 1 and 3, and part 2 has patterns 2 and 4. The don't-cares of column c_1 are assigned so that the low order condition is satisfied. The resulting don't-care assignment and column permutation is shown in the matrix to the right in Table 68.8.

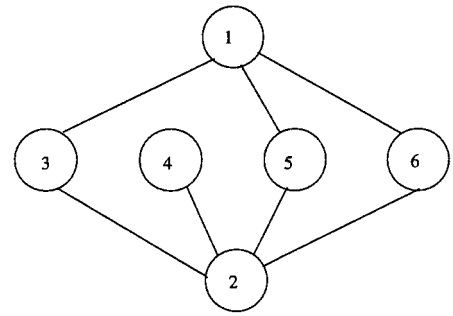


FIGURE 68.4 Graph construction with the don't-care assignment.

TABLE 68.7

c_1	c_2	c_3	c_4	c_5	c_6
0	0	1	x	1	1
x	1	0	0	x	0
1	x	x	1	x	x
0	x	x	x	0	x

TABLE 68.8

0	0	1	1	1	1	0	1	1	1	1	0
x	1	0	0	0	0	1	0	0	0	0	0
1	x	1	1	1	1	0	1	1	1	1	1
0	x	0	0	0	0	1	0	0	0	0	0

Extensions of the CAD tool involve partitioning of the patterns into submatrices where some or all of the above-mentioned operations are applied independently. For example, the columns of one submatrix can be permuted in a completely different way from the columns of another submatirx. Tradeoffs between hardware overhead and reproduction time have been analyzed among different variations (extensions) of the CAD tools. The tradeoffs are determined by the subset of operations that can be applied independently in each submatrix. The larger the set, the higher the hardware overhead is.

DFT and BIST for Sequential Logic

CAD Tools for Scan Designs

In the full scan design, all the flip-flops in the circuit must be scanned and inserted in the scan chain. The hardware overhead is large and the test application time is lengthy for circuits with a large number of flip-flops. Test application time can be drastically reduced by an appropriate reordering of the cells in the scan chain. This cell reordering problem has been formulated as a combinatorial optimization problem which is shown to be NP-hard. However, an efficient CAD tool for determining an efficient cell reordering is presented in Ref. 8.

One useful approach for reducing both of the above costs is to resynthesize the circuit by repositioning its flip-flops so that their number is minimized while the functionality of the design is preserved. We describe such a circuit resynthesis scheme.

Let us consider the circuit graph $G = (V, E)$ of the circuit, where each node $v \in V$ is either an input/output port or a combinational module. Each edge $(u, v) \in E$ is assigned a weight $ff(u, v)$ equal to the number of flip-flops on it. Ref. 9 has shown that flip-flops can be repositioned without changing the functionality of the circuit as follows.

Let IO denote the set of input/output ports. The flip-flop repositioning problem amounts to assigning $r()$ values to each node in V so that

$$\begin{aligned} &r(v) = 0, \ \forall v \in IO \\ &r(u) = r(v) \le ff(u,v), \ \forall (u,v) \in E \end{aligned} \tag{68.1}$$

Once an $r()$ value is assigned to each node at I/O port, the new number of flip-flops on each edge (u, v) is computed using the formula

$$ff_{new}(u,v) = ff(u,v) + r(u) - r(v) \tag{68.2}$$

The set of constraints in Eq. 68.1 is a set of difference constraints and forms a special case of linear programming which can be solved in polynomial time using Bellman–Ford shortest path calculations. The described resynthesis scenario is also referred to as *retiming* because flip-flop repositionings may affect the clock period.

The above set of difference constraints has an infinite number of solutions. Thus, there exists an infinite number of circuit designs with an equivalent functionality. One can benefit from these alternative designs, and resynthesis can be done in order to optimize certain objective functions. In full scan, the objective is to minimize the total number of flip-flops. The latter quantity is precisely

$$\sum_{(u,v)} ff_{new}(u,v)$$

which can be rewritten (using Eq. 68.2) as

$$\sum_{(u,v)}\left(ff(u,v)+r(u)-r(v)\right)=\sum_{(u,v)}ff(u,v)+\sum_{(u,v)}\left(r(u)-r(v)\right) \tag{68.3}$$

Since the first term in Eq. 68.3 is an invariant, the goal is to find $r()$ values that minimize $\sum_{(u,v)}(r(u)-r(v))$ subject to the constraints in Eq. 68.1. This special case of integer linear programming is polynomially solvable using min-cost flow techniques.[9] Once the $r()$ values are computed, Eq. 68.2 is applied to determine where the flip-flops will be repositioned. The resulting circuit has minimum number of flip-flops.[9]

Although full scan is widely used by the industry, its hardware overhead is often prohibitive. An alternative approach for scan designs is the *structural partial scan approach* where a minimum cardinality subset of the flip-flops must be scanned so that every cycle contains at least one scanned flip-flop. This is an NP-hard problem. Reference 10 has shown that minimizing the number of flip-flops subject to some constraints additional to Eq. 68.1 turns out to be a beneficial approach for structural partial scan. The idea here is that minimizing the number of flip-flops amounts to maximizing the average number of cycles per flip-flop. This leads to efficient heuristics for selecting a small number of flip-flops for breaking all cycles.

Other resynthesis schemes that reposition the flip-flops in order to reduce the partial scan overhead have been proposed in Refs. 11 and 12. Both schemes initially identify a set of lines L that forms a low cardinality solution for partial scan. L may have lines without flip-flops. Thus, the flip-flops must be repositioned so each line of L has a flip-flop which is then scanned.

Another important goal in partial scan is to minimize the *sequential depth of the scanned circuit*. This is defined as the maximum number of flip-flops along any path in the scanned circuit whose endpoints are either controllable or observable. The sequential depth of a scanned circuit is a very important quantity because it affects the upper bound on the length of the test sequences which need to be applied in order to detect the stuck-at faults. Since the scanned circuit is acyclic, the sequential depth can be determined in polynomial time by a simple topological graph traversal.

Figure 68.5 below illustrates the concept of the sequential depth. Cycles denote I/O ports, oval nodes represent combinational modules, solid square nodes indicate unscanned flip-flops, and empty square nodes are scanned flip-flops. The sequential depth of the circuit graph to the left is 2. The figure to the right shows an equivalent circuit where the sequential depth has been reduced to 1. In this figure, the unscanned (solid flip-flops) have been repositioned, while the scanned flip-flops remain at the original positions so that the scanned circuit is guaranteed to be acyclic. Flip-flop repositioning is done subject to the constraints in Eq. 68.1 so that the functionality of the design is preserved.

Let F be the set of observable/controllable points in the scanned circuit. Let $F(u, v)$ denote the maximum number of unscanned flip-flops between u and v, $u, v \in F$, and E' denote the set of edges in the scanned sequential graph that have a scanned flip-flop. Ref. 10 proves that the sequential depth is at most k if and only if there exists a set of $r()$ values that satisfy the following set of inequalities:

$$\begin{aligned} &r(u)-r(v)=0, \ \forall(u,v)\in E' \\ &r(v)-r(u)\le k-F(u,v), \ \forall u,v\in F \end{aligned} \tag{68.4}$$

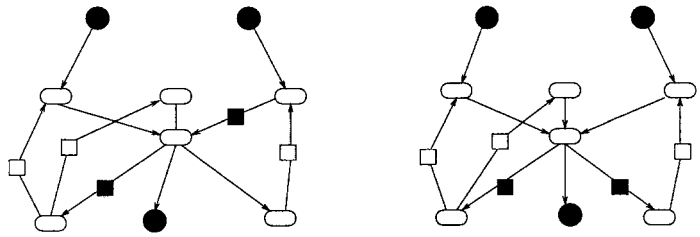


FIGURE 68.5 The impact of flip-flop repositioning on the sequential depth.

A simple hierarchy search can then be applied in order to find the smallest sequential depth that can be obtained with flip-flop repositioning.

A final objective in partial scan is to be able to *balance the scanned circuit.* In a balanced circuit, all paths between any pair of combinational modules have the same number of flip-flops. It has been shown that the TPG process for a balanced circuit reduces to TPG for combinational logic.[13] It has been proposed to balance a circuit by enhancing already existing flip-flops in the circuit and then bypassing them during testing mode.[13] A multiplexing circuitry needs to be associates with each selected flip-flop. Minimizing the multiplexer-related hardware overhead amounts to minimizing the number of selected flip-flops, which is an NP-hard problem.[13]

The natural question is whether flip-flop repositioning may help in balancing a circuit with less hardware overhead. Unfortunately, it has been shown that it cannot. It can however assist in inserting the minimum possible *bse* elements in order for the circuit to be balanced. Each inserted *bse* element is bypassed during operation mode but acts as a delay element in testing mode.

The algorithm consists of two steps. In the first step, *bses* are greedily inserted so that the scanned circuit becomes balanced. Subsequently, the number of the inserted *bse* elements is minimized by repositioning the inserted elements.

This is a variation of the approach that was described earlier for minimizing the number of flip-flops in a circuit. *Bses* are treated as flip-flops, but for every edge (u, v) with original circuit flip-flops, the set of constraints in Eq. 68.1 is enhanced with the additional constraint $r(u) - r(v) = 0$. This ensures that the flip-flops of the circuit will not be repositioned.

The correctness of the approach relies on the property that any flip-flop repositioning on a balanced circuit always maintains the balancing property. This can be easily shown as follows.

In an already balanced circuit, the number of flip-flops on any path $p_i(u, v)$ between any combinational nodes u, v has a number of flip-flops $c(u, v)$. When u and v are not adjacent nodes but the endpoints of a path p with two or more lines, a telescoping summation using Eq. 68.2 can be applied on the edges of the path to show that $ff_{new}p(u, v)$, the number of flip-flops on p after retiming, is

$$ff_{new}p(u,v) = c(u,v) + r(u) - r(v)$$

Observe now that quantity $ff_{new}p(u, v)$ is independent of the actual path $p(u, v)$, and remains invariant as long as we have a path between nodes u and v. This argument holds for all pairs of combinational nodes u, v. Thus, the circuit remains balanced after repositioning the flip-flops.

Test application time is a complex issue for designs that have been resynthesized for improved partial scan. Test sequences that have been precomputed for the circuit prior to its resynthesis cannot any more be applied to the resynthesized circuit. However, Ref. 14 shows that one can apply such recomputed test sequences after an initializing sequence of patterns brings the circuit to a given state s. State s guarantees that the precomputed patterns can be applied.

On-chip Schemes for Sequential Logic

Many CAD tools have been proposed in the literature for automating the design of BIST on-chip schemes for sequential logic. The first CAD tool of this section considers LFSR-based pseudo-exhaustive BIST. Then, a deterministic scheme that uses Cellular Automata is presented.

A popular LFSR-based approach for pseudorandom built-in self-test (BIST) of sequential proposes to enhance the scanned flip-flops of the circuit into either *Built-In Logic-Block Observation* (BILBO) cells or *Concurrent Built-In Logic-Block Observation* (CBILBO) cells. Additional BILBO cells and CBILBO cells that are transparent in normal mode can also be inserted into arbitrary lines in sequential circuits. The approach uses pseudorandom pattern generators (PRPGs) and *multiple-input signature registers* (MISRs).

There are two important differences between BILBO and CBILBO cells. (For the detailed structure of BILBO and CBILBO cells, see Ref. 15, Chapter 11.) First, in testing mode, a CBILBO cell operates both in the PRPG mode and the MISR mode, while a BILBO cell only can operate in one of the two modes.

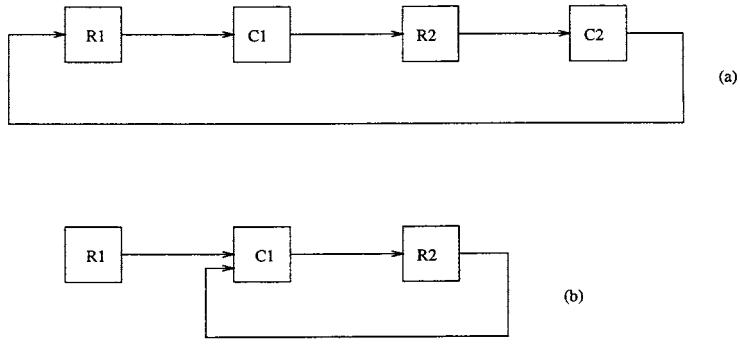


FIGURE 68.6 Illustration of the different hardware overheads.

The second difference is that CBILBO cells are more expensive than BILBO cells. Clearly, inserting a whole transparent test cell into a line is more expensive than enhancing an existing flip-flop regarding hardware costs.

The basic BILBO BIST architecture partitions a sequential circuit into a set of registers and blocks of combinational circuits with normal registers replaced by BILBO cells. The choice between enhancing existing flip-flops to BILBO cells or to insert transparent BILBO cells generates many alternative scenarios with different hardware overheads.

Consider the circuit in Fig. 68.6(a) with two BILBO registers R1 and R2 in a cycle. In order to test C1, register R1 is set in PRPG mode and R2 in MISR mode. Assuming that the inputs of register R1 are held at the value zero, the circuit is run in this mode for as many clock cycles as needed, and can be tested exhaustively for most cases — except for the all-zero pattern. At the end of this test process, the contents of R2 can be scanned out and the signature is checked. In the same way, C2 can be tested by configuring register R1 into MISR mode and R2 into PRPG mode.

However, the circuit in Fig. 68.6(b) does not conform to a normal BILBO architecture. This circuit has only one BILBO register R2 in a self-loop. In order to test C1, register R1 must be in PRPG mode, and register R2 must be in both MISR mode and PRPG mode, which is impossible due to the BILBO cell structure. This situation can be handled by either adding a transparent BILBO register in the cycle or by using a CBILBO that can operate simultaneously in both MISR and PRPG modes.

In order to make a sequential circuit self-testable, each cycle of the circuit must contain at least one CBILBO cell or two BILBO cells. This combinatorial optimization problem is stated as follows. The input is a sequential circuit, and a list of hardware overhead costs.

cB: The cost of enhancing a flip-flop to a BILBO cell.
cCB: The cost of enhancing a flip-flop to a CBILBO cell.
cBt: The cost of inserting a transparent BILBO cell.
cCBt: The cost of inserting a transparent CBILBO cell.

The goal is to find a minimum cost solution of this scan register placement problem in order to make every cycle in the circuit have at least one CBILBO cell or at least two BILBO cells.

The optimal solution for a circuit may vary, depending upon different cost parameter sets. For example, we can have three different solutions for the circuit in Fig. 68.7. The first is that both flip-flops FF1 and FF2 can be enhanced to CBILBO cells. The second is that one transparent CBILBO cell can be inserted at the output of gate G3 to break the two cycles. The third is that both flip-flops FF1 and FF2 can be enhanced to BILBO cells, together with one transparent BILBO cell inserted at the output of gate G3. Under the cost parameter set $cB = 20$, $cBt = 30$, $cCB = 40$, $cCBt = 60$, the hardware overhead of the three solutions are 80, 60, and 70, in that order. The second solution, using a transparent CBILBO cell, has the least hardware overhead.

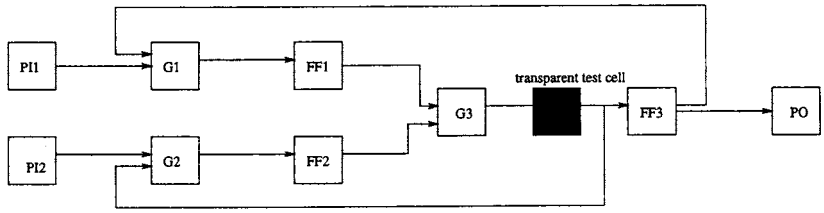


FIGURE 68.7 The solution depends on the cost parameter set.

However, under the cost parameter set $cB = 10$, $cBt = 30$, $cCB = 40$, $cCBt = 60$, the first solution, using both transparent and enhanced BILBO cells, yields the optimal solution with total hardware overhead of 50. Although a CBILBO cell is more expensive than a BILBO cell, and a transparent cell is more expensive than an enhanced one, in some situations using CBILBO cells and transparent test cells may be beneficial to the hardware overhead.

For this difficult combinatorial problem, Ref. 16 presents a CAD tool that finds the optimal hardware overhead using a branch and bound approach. The worst-case time complexity of the CAD tool is exponential and, in many instances, its time response is prohibitive. For this reason, Ref. 16 proposes an alternative branch and bound CAD tool that terminates the search whenever solutions close to the optimal are found. Although time complexity still remains exponential, the results reported in Ref. 16 show that branch and bound techniques are promising.

The remainder of this section presents a CAD tool for embedding test sequences on-chip. Checking for stuck-at faults in sequential logic requires the application of a sequence of test patterns to set the values of some flip-flops along with those values required for fault justification/propagation. Therefore, it is imperative that *all test patterns in each test sequence are applied in the specified order. Cellular automata* (CA) have been proposed as a TPG mechanism to achieve this goal, the advantage being mainly that they are a finite state machine (FSM) with a very regular structure.

References 17 and 18 propose that *hybrid CAs* are used for embedding test sequences on-chip. Hybrid CAs consist of a series of flip-flops $f_i 1 \le n$. The next state f_i^+ of flip-flop i is a function F_i of the present states of f_{i-1}, f_i, and f_{i+1}. (We call them the *3-neighborhood CAs.*) For the computation of f_1^+ and f_n^+, the missing neighbors are considered to be constant 0. A straightforward implementation of function F_i is by an 8-to-1 multiplexer.

Consider a $p \times w$ test matrix T comprising p ordered test vectors. The CAD tool in Ref. 18 presents a systematic methodology for this embedding problem. First, we give some definitions.[18]

Given a sequence of three columns (X_L, X, X_R), each row i, $1 \le i \le p-1$, is associated to a template $\tau_i = \begin{bmatrix} x_L^i & x^i & x_R^i \\ & x^{i+1} & \end{bmatrix}$. (No template is associated with the last row p). Let $H(\tau_i)$ denote the upper part $[x_L^i\, x^i\, x_R^i]$ of τ_i and let $L(\tau_i)$ denote the lower part, $[x^{i+1}]$.

Given a sequence of columns (X_L, X, X_R), two templates τ_i and τ_j, $1 \le i, j \le p-1$, are conflicting if and only if it happens that $H(\tau_i) = H(\tau_j)$ and $L(\tau_i) \ne L(\tau_j)$. A sequence of three columns (X_L, X, X_R) is a *valid triplet* if and only if there are no conflicting templates. This is imperative in order to have a properly defined F_i function for the corresponding CA cell that will generate column X of the test matrix, if column X is assigned between columns X_L and X_R in the CA cell ordering. If a valid triple cannot be formed from test matrix columns, a so-called "link column" must be introduced (corresponding to an extra CA cell) so as to make a valid triplet.

The goal in the studied on-chip embedding problem by a hybrid CA is to introduce the minimum number of link columns (extra CA cells) so as to generate the whole sequence. The CAD tool in Ref. 18 tackles this problem by a systematic procedure that uses *shift-up columns*. Given a column $X = (x^1, x^2, \ldots, x^p)^{tr}$, the shift-up column of X is the column $\hat{X} = (x^1, x^2, \ldots, x^p, d)^{tr}$, where d is a don't-care. Given a column X, the sequence of columns $(X_L, X, \hat{X})$ is a valid triplet for *any* column X_L.

Moreover, given two columns A and B of the test matrix, a *shifting sequence* from A to B to be a sequence of columns $(A, L_0, L_1, L_2, \ldots, L_j, B)$ such that $L_0 = \hat{A}$, $L_i = \hat{L}_{i-1}$, $1 \le i \le j$, and (L_{j-1}, L_j, B), is a valid triplet. A shifting sequence is always a valid sequence.

The important property of a shifting sequence $(A, L_0, L_1, L_2, \ldots, L_j, B)$ is that column A can be preceded by any other column X in a CA ordering, with the resulting sequence $(X, A, L_0, L_1, L_2, \ldots, L_j, B)$ being still valid. That is, for any two columns A and B of the test matrix, column B can always be placed after column A with some intervening link columns *without regard to what column is placed before A*. Given any two columns A and B of the test matrix, the goal of the CAD tool in Ref. 18 is to find a shifting sequence $(A, L_0, L_1, \ldots, L_{jAB}, B)$ of minimum length. This minimum number (denoted by m_{AB}) can be found by successive shift-ups of $L_0 = \hat{A}$ until a valid triplet ending with column B is formed.

Given an ordered test matrix T, the CAD tool in Ref. 18 reduces the problem of finding short length test shifting sequences to that of computing a Traveling Salesman (TS) solution on an auxiliary graph. Experimental results reported in Ref. 18 show that this hybrid CA-based approach is promising.

Fault Simulation

Explicit fault simulation is needed whenever the test patterns are generated using an ATPG tool. Fault simulation is needed in scan designs when an ATPG tool is used for TPG. Fault simulation procedures may also be used in the design of deterministic on-chip TPG schemes. On the other hand, pseudo-exhaustive/pseudorandom BIST schemes mainly use compression techniques for detecting whether the circuit is faulty. Compression techniques were covered in Chapter 67.[15] (Chapter 10 provides a more detailed discussion.)

This section reviews CAD tools proposed for fault simulation of stuck-at faults in single-output combinational logic. For a more extensive discussion on the subject, we refer the reader to Ref. 15 (Chapter 5).

The simplest form of simulation is called *single-fault propagation*. After a test pattern is simulated, the stuck-at faults are inserted one after the other. The values of every faulty circuitry are compared with the error-free values. A faulty value needs to be propagated from the line where the fault occurs. The propagation process continues line-by-line, in a topological search manner, until there is no faulty value that differs from the respective good one. If the latter condition is not satisfied, the fault is detected.

In an alternative approach, called *parallel-fault propagation*, the goal is to simulate n test patterns in parallel using n–bit memory. Gates are evaluated using boolean instructions operating on n–bit operands. The problem with this type of simulation is that events may occur only in a subset of the n patterns while at a gate. If one average α fraction of gates have events on their inputs in one test pattern, the parallel simulator will simulate $1/\alpha$ more gates than an event-driven simulator. Since n patterns are simulated in parallel, the approach is more efficient when $n \ge 1/\alpha$, and the speed-up is $n \cdot \alpha$. Single and parallel fault propagation are combined efficiently in a CAD tool proposed in Ref. 19.

Another approach for fault simulation is the *critical path tracing approach*.[20] For every test pattern, the approach first simulates the fault-free circuit and then determines the detected faults by determining which lines have *critical values*. A line has critical value 0 (1) in pattern t if and only if test pattern t detects the fault stuck-at 0 (1) at the line. Therefore, finding the lines that are critical in pattern t amounts to finding the stuck-at faults that are detected by t.

Critical lines are found by backtracking from the primary outputs. Such a backtracking process determines paths of critical lines that are called *critical paths*. The process of generating critical paths uses the concept of *sensitive inputs* of a gate with two or more inputs (for a test pattern t). This is determined easily: If only input l has the controlling value of a gate, then it is sensitive. On the other hand, if all the inputs of a gate have noncontrolling value, then they are all sensitive. There is no other condition for labeling some input line of a gate as sensitive. Thus, the sensitive inputs of a gate can be identified during the fault-free simulation of the circuit.

The operation of the critical path tracing algorithm is based on the observation that when a gate output is critical, then all its sensitive inputs are critical. On fan-out free circuits, critical path tracing is

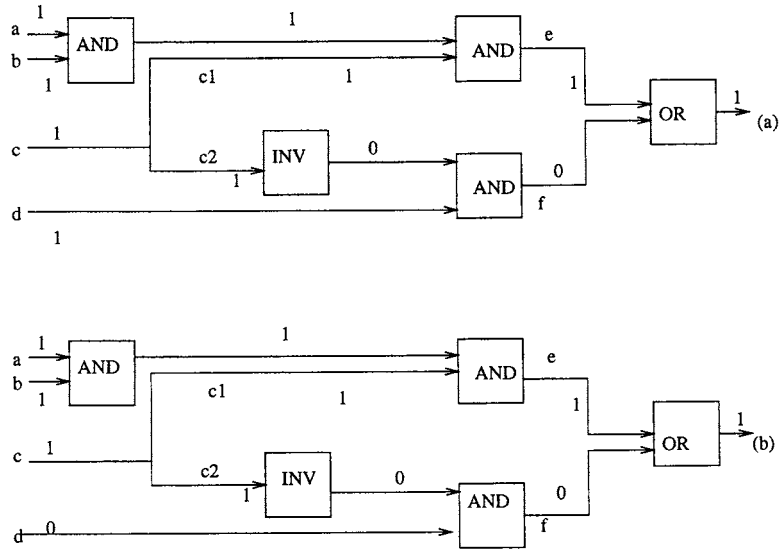


FIGURE 68.8 The solution depends on the cost parameter set.

a simple traversal that applies recursively to the above observation. The situation is more complicated when there exist reconvergent fan-outs. This is illustrated in Fig. 68.8.

In Fig. 68.8(a), starting from *g*, we determine critical lines *g*, *e*, *b*, and *c*1 as critical, in that order. In order to determine whether *c* is critical, we need additional analysis. The effects of the fault stuck-at 0 on line *c* propagate on reconvergent paths with different parities which cancel each other when they reconverge at gate *g*. This is called *self-masking*. Self-masking does not occur at Fig. 68.8(b) because the fault propagation from *c*2 does not reach the reconvergent point. In Fig. 68.8(b), *c* is critical.

Therefore, the problem is to determine whether self-masking occurs or not at the stem of the circuit. Let 0 (1) be the value of a stem *l* under test *t*. A solution is to explicitly simulate the fault stuck-at 1 (0) on *l*, and if *t* detects this fault, then *l* is marked as critical.

Instead, the CAD tool uses bottlenecks in the propagation of faults that are called *capture lines*. Let *a* be a line with topological level tl_a, sensitized to stuck-at fault *f* with a pattern *t*. If every path sensitized to *f* either goes through *a* or does not reach any other line with greater topological level greater than tl_a, then *a* is a capture line of *f* under pattern *t*. Such a line is common to all paths on which the effects of *f* can propagate to the primary output under pattern *t*.

The capture lines of a fault form a transitive chain. Therefore, a test *t* detects fault *f* if and only if all the capture lines of *f* under test pattern *t* are critical in *t*. Thus, in order to determine whether a stem is critical, the CAD tool does not propagate the effects of the fault step up to the primary output; it only propagates the fault effects up to the capture line that is closest to the stem.

68.3 CAD for Path Delays

CAD Tools for TPG

Fault Models and Non-enumerative ATPG

In the path delay fault problem, defects cause the propagation time along paths in the circuit under test to exceed the clock period. We assume here a fully scanned circuit where path delays are examined in combinational logic. A *path delay fault* is any path where either a rising ($0 \rightarrow 1$) or falling ($1 \rightarrow 0$) transition occurs on every line in the path. Therefore, for every physical path in the circuit, there exist two path delay faults. The first path delay fault is associated with a rising transition on the first line on

the path. The second path delay fault is associated with a falling transition on the first line on the path. In order to detect path delay faults, pairs of patterns must be applied rather than single test patterns.

One of the conditions that can be imposed on the tests for path delay faults is the *robust* condition. Robust tests guarantee the detection of the targeted path delay faults independent of any delays in the rest of the circuit. Table 68.9 lists the conditions for robust propagation of path delay faults in a circuit containing AND, OR, NAND, and NOR gates.

TABLE 68.9 Requirements for Robust Propagation

	Output Transition	
gate	$0 \rightarrow 1$	$1 \rightarrow 0$
AND	Any number of inputs	Single input
OR	Single input	Any number of inputs
NAND	Single input	Any number of inputs
NOR	Any number of inputs	Single input

Thus, when the output of a AND gate has been assigned, a rising transition multiple inputs are allowed to have rising transitions because rising transitions for an AND gate are transitions from a *controlling value* (*cv*) to a *noncontrolling value* (*ncv*). If, on the other hand, the output of an AND gate has a falling transition ($ncv \rightarrow cv$), then only one input is allowed to have a $ncv \rightarrow cv$ transition in order to satisfy the robustness.

Some definitions are necessary before we describe additional path delay fault families. Given a path delay fault p and a gate g on the p, the *on-input* of g with respect to path p is the input of g that is also on p. All other inputs of g are called *off-inputs* of g with respect to path p.

Robust path delay faults are a subset of the *non-robust path delay faults*. A non-robust test vector satisfies the conditions: (1) a transition is launched at the primary input of the target path, and (2) all off-inputs of the target path settle to non-controlling values under the second pattern in the vector. A robust test vector must satisfy the conditions of the non-robust tests, and whenever the transition at an on-input line a is $cv \rightarrow ncv$, each off-input of a is steady at ncv. The target faults detected by robust test vectors are called *robustly testable*, and are a subset of the target faults that are detected by non-robust test vectors. The target faults that are not robust testable and are detected by non-robust test vectors are called *non-robustly testable*. Non-robust test vectors cannot guarantee the detection of the target fault in the presence of other delay faults.

Functionally sensitizable test vectors allow for faults to be detected in the presence of multiple path delays. They detect a set of faults that is a superset of those detected by non-robust test vectors. A target fault is *functionally testable* (*FT*) if there is at least one gate with one or more off-inputs with $ncv \rightarrow ncv$ transition, where all of its off-inputs with $ncv \rightarrow cv$ transition are also delayed while its remaining off-inputs satisfy the conditions for non-robust test vectors. We say that each such gate satisfies the *functionally testable* (*FT*) *condition*. It has been shown that FT faults have better probability to be detected when the maximum off-input slack (or, simply, slack) is a small integer. (The slack of an off-input is defined as the difference between the stable time of the on-input signal and the stable time of the off-input signal.) Faults that are not detected by functionally sensitizable test vectors are called *functionally unsensitizable*. Table 68.10 summarizes the above-mentioned off-input conditions.[21]

TABLE 68.10 Off-input Signals for Two Input Gates and Fault Classification

	Off-input Transition	On-input Transition
$cv \rightarrow ncv$	Robust	Non-robustly testable
$ncv \rightarrow cv$	Funct. unsensitizable	Functionally testable
Stable *ncv*	Robust	Robust
Stable *cv*	Funct. unsensitizable	Funct. unsensitizable

Other classifications of path delay faults have been recently proposed in the literature, but they are not presented here.[22,23] Systematic path delay fault classification is very important when considering test pattern generation. For example, test pattern generation for robust path delay faults does not need to consider actual delays on the gates. However, delays have to be considered when generating pairs of patterns for non-robust and functionally testable faults. For the latter fault family, the generator must take into consideration that they are multiple faults, and that the slack is an important parameter for their detection.

The conventional approach for generating test patterns for path delay faults is a modification of the test pattern generation for stuck-at faults. It consists of a two-phase loop, each loop iteration resulting in a generated pair of patterns. Initially, transitions are assigned on the lines of path *P*. This is called the *path sensitization phase*. Then, a modified ATPG for stuck-at fault is executed twice. The first time, a test pattern must be generated so that every line of the selected path delay fault receives its initial transition value. The second execution of the modified ATPG generates another pattern, which assigns the final transition value on every line on the path. This is called the *line justification phase*.

The problem with this conventional approach is that the repeat loop will be executed as many times as the number of path delay faults, which is an exponential quantity to the size of the circuit. More explicitly, the difficulty of the path delay fault model is that the number of targeted faults is exponential, therefore we cannot afford to generate pairs of test patterns that detect one fault at a time.

Any practical ATPG tool must be able to generate a polynomial number of test patterns. Thus, in the case of path delay faults, the two-phase loop must be modified as follows. The first phase must be able to sensitize multiple paths. The second phase must be able to justify the assigned line transitions of as many sensitized paths as possible.

The goal in a *non-enumerative ATPG* is to generate a pair of patterns that sensitizes and justifies the transitions on all the lines of a subcircuit. Clearly, the average number of paths in each examined subcircuit must be an exponential quantity when the number of paths in the circuit is exponential. Thus, a necessary condition for the path sensitization phase is to generate, on average, subgraphs with large size.

The ATPG tools described in this section generate pairs of test patterns for robust path delay faults.[24,25] Both tools target an efficient path sensitization phase. A necessary condition for the paths of a subcircuit to be simultaneously sensitized is to be *structurally compatible* with respect to the parity (on the number of inverters) between any two reconvergent nodes in the subcircuit. This concept is illustrated in Fig. 68.9.

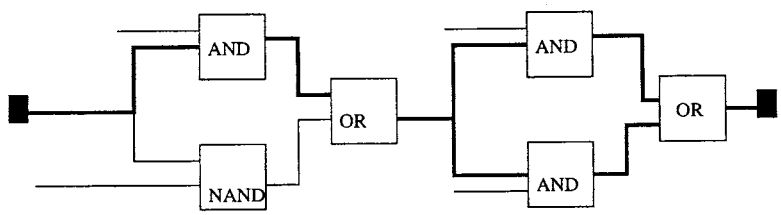


FIGURE 68.9 A graph consisting of structurally compatible paths.

Consider the circuit on the top portion of Fig. 68.9. The subgraph induced by the thick edges consists of two structurally compatible paths. These two paths share two OR gates. The two subpaths that share the same OR gate endpoints have even parity.

Any graph that constraints structurally compatible graphs is called a *structurally compatible* (SG) graph. The tools in Refs. 24 and 25 consider a special case of SG graphs with a single primary input and a single primary output. We call such an SG graph a primary compatible SG graph (PCG graph).

For the same pair of primary input and output nodes in the circuit, there may be many different PCG graphs, which are called sibling PCG graphs. Sibling PCG graphs contain mutually incompatible paths. The subgraph induced by the thick edges on the bottom portion of Fig. 68.9 shows a PCG that is sibling to the one on the top portion. This graph also contains two paths (the ones induced by the thick edges).

The ATPG tool in Ref. 25 generates large sibling PCGs for every pair of primary input and output nodes in the circuit. The size of each returned PCG is measured in terms of the number of structurally compatible paths that satisfy the requirements for robust propagation described earlier. Experimentation in Ref. 25 shows that the line justification phase satisfies the constraints along paths in a manner proportional to the size of the graph returned by the multiple path sensitization phase.

Given a pair of primary input and primary output nodes, Ref. 25 constructs large sibling PCGs as follows. Initially, a small number of lines in the circuit are removed so that the subcircuit between the selected primary inputs and outputs is a series-parallel graph. A polynomial time algorithm is applied on the series-parallel graph which finds the maximum number of structurally compatible paths that satisfy the conditions for robust propagation. An intermediate tree structure is maintained, which helps extract many such large sibling PCGs for the same pair of primary input and output nodes. Finally, many previously deleted edges are inserted so that the size of the sibling PCGs is increased further by considering paths that do not necessarily belong on the previously constructed series-parallel graph.

Once a pair of patterns is generated by the ATPG tool in Ref. 25, fault-simulation must be done so that the number of robust paths detected by the generated pair of patterns can be determined. The fault simulation problem for the path delay fault model is not as easy as for the stuck-at model. The difficulty relies on the fact that the number of path delay faults is not necessarily a polynomial quantity.

Each generated pair of patterns by the CAD tool in Ref. 25 targets robust path delay faults in a particular sibling PCG. It may, however, detect robust path delay faults in the portion of the circuit outside the targeted PCG. This complicates the fault simulation process. Thus, Ref. 25 suggests that faults are simulated only within the current PCG in which case a simple topological graph traversal suffices to detect them.

On-chip TPG Aspects

Many recent on-chip TPG schemes have been recently proposed for generating pairs of patterns. They are classified as either pseudo-exhaustive/pseudorandom or deterministic.

A pseudo-exhaustive scheme for generating pairs of patterns on-chip is proposed in Ref. 26. The method is based on a simple LFSR that has $2 \cdot w$ cells for a circuit with w inputs. Every other LFSR cell is connected to a circuit input. In particular, all the LFSR cells at even positions are connected to circuit inputs, and the remaining LFSR cells are used for "destroying" the shift dependency of the contents in the LFSR cells at even positions. The cells at odd positions are also called *separation cells*. Since the contents of the latter cells are independent, the scheme can generate all the possible two-input patterns. The schematic of the approach is given in Fig. 68.10.

Such an LFSR scheme is called a *full-input separation LFSR*.[26] It requires a significant hardware overhead and long wire feedback connections. A CAD tool is presented in Ref. 26 that reduces the size of the hardware overhead and the wire lengths by simply observing that separation cells must exist between any two LFSR cells that are connected to inputs that affect at least one circuit output. For each circuit output o, the I_o set which contains the labels of all the input cells of the *full separation LFSR* which affect o is constructed. Then, an LFSR cell relabeling CAD tool is proposed which minimizes the total number of separation cells so that the labels of all I_os are even numbers.[26]

Weighted random LFSRs can be used for on-chip deterministic TPG of pairs of patterns. Let us, for simplicity, consider the embedding problem. Here, the goal is to reproduce on-chip a matrix T consisting

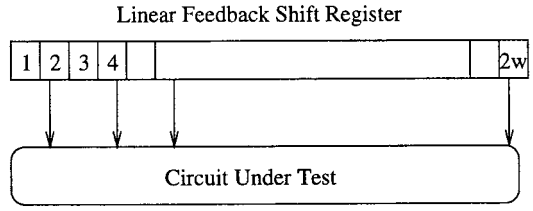


FIGURE 68.10 The schematic of an LFSR-based scheme for pseudo-exhaustive on-chip TPG.

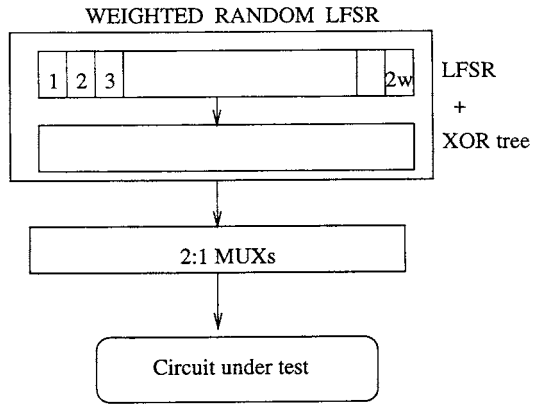


FIGURE 68.11 The schematic of a weighted random LFSR-based approach for deterministic on-chip TPG.

of n pairs of patterns (p_i^1, p_i^2), $1 \le i \le n$, each of size w, that have been generated by an ATPG tool such as the one described in the previous section.

A simple approach is to use a weighted random LFSR that n generates patterns p_i of size $2w$. Every pattern p_i is simply the concatenation of patterns p_i^1 and p_i^2. Once pattern p_i is generated, a simple circuit consisting of two-to-one multiplexers "splits" pattern p_i into its two pattern p_i^1 and p_i^2 and, in addition, guarantees that patterns p_i^1 are applied at even clock pulses and pattern p_i^2 are applied at odd clock pulses. The schematic of the approach is given in Fig. 68.11.

Fault Simulation and Estimation

Exact fault simulation for path delay faults is not a trivial aspect independent of the model used to propagate the delays (robust, non-robust, functionally testable path delay faults). The number of path delay faults remains, in the worst case, exponential, independent of propagation restrictions. Ref. 27 presents an exact simulation CAD tool for any type of path delay fault. The drawback of the approach in Ref. 27 is that it may require exponential time (and space) complexity, although experimentation has shown that in practice it is very efficient.

The following describes CAD tools for obtaining lower bounds on the number of detected path delay faults by a given set of n pairs of patterns. These approaches apply to any type of path delay fault and are referred to as fault estimation schemes.

In Ref. 28, every time a pair of patterns is applied, the CAD tool examines whether there exists at least one line where either a rising or falling transition has not been encountered by the previously applied pairs of test patterns. Let E_i, $1 \le i \le n$, denote the set of lines for which either a rising or a falling transition occurs for the first time when the pair of patterns P_i is applied.

When $|E_i| > 0$, a new set of path delay faults is detected by pattern P_i. These are the paths that contain lines in E_i. A simple topological search of the combinational circuit suffices to detect their number. If for some P_i, we have $|E_i| = 0$, the approach does not detect any path delay faults.

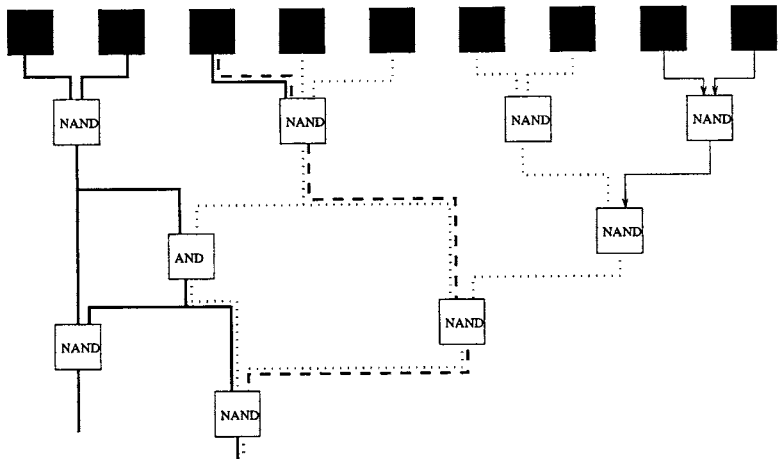


FIGURE 68.12 An undetected path delay fault.

The approach in Ref. 28 is non-enumerative but returns a conservative lower bound to the number of detected paths. Figure 68.12 illustrates a case where a path delay fault may not be counted.

Assume that the path delay faults in all three patterns start with a rising transition. Furthermore, assume that the first pair of patterns detects path delay faults along all the paths of the subgraph which is covered by thick edges. Let the second pair of patterns detect path delay faults on all the paths of the subgraph covered by dotted edges, and let the dashed path indicate a path delay fault detected by the third pair of patterns. Clearly, the latter path delay fault cannot be detected by the approach in Ref. 28.

For this reason, Ref. 28 suggests that fault simulation is done by virtually partitioning the circuit into subcircuits. The subcircuits should contain disjoint paths. One implementation for such a partitioning scheme is to consider lines that are independent in the sense that there is no physical path in the circuit that contains any two selected lines. Once a line is selected, we form a subcircuit that consists of all lines that depend on the selected line. In addition, the selected lines must form a cut separating the inputs from the outputs so that every physical path. This way, every path delay fault belongs to exactly one subcircuit. Figure 68.13 below shows three selected lines (the thick lines) of the circuit in Fig. 68.12 that are independent and also separate the inputs from the outputs.

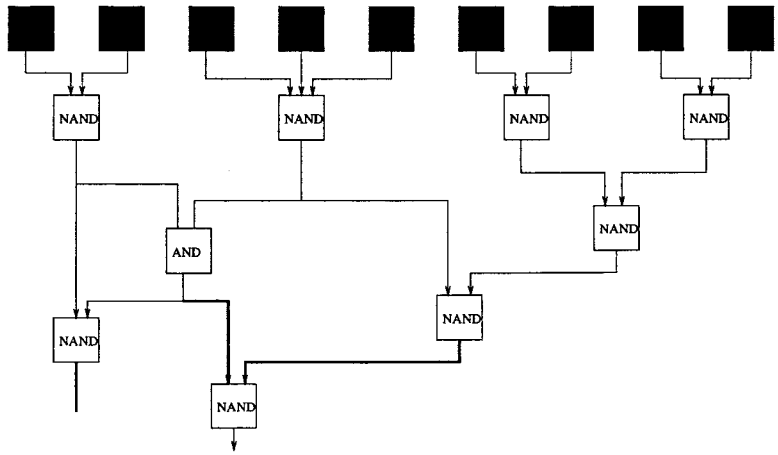


FIGURE 68.13 Three independent lines that form a cut.

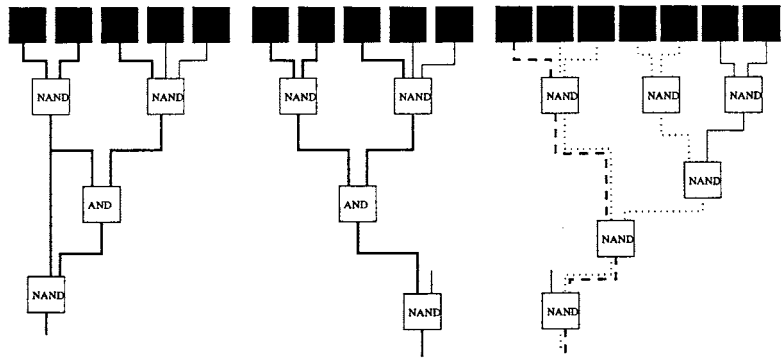


FIGURE 68.14 All paths are detected using three subcircuits.

Figure 68.14 contains the subcircuits corresponding to these lines. The first pattern detects path delay faults in the first two subcircuits, and the second pattern detects path delay faults in the third subcircuit. The missed path delay fault by the third pattern of Fig. 68.2 is detected on the third subcircuit because, in that subcircuit, its first line does not have a marked rising transition when the third pair of patterns is applied.

Reference 29 gives a new dimension to the latter problem. Such a cut of lines is called a *strong cut*. The idea is to find a maximum strong cut that allows for a maximum collection of subcircuits where fault coverage estimation can take place. A CAD tool is presented in Ref. 29 that returns such a maximum cardinality strong cut. The problem reduces to that of finding a maximum weighted independent set in a comparability graph, which is solvable in polynomial time using a minimum flow technique. There is no formal proof that the more the subcircuits, the better the fault coverage estimation is. However, experimentation verifies this assertion.[29]

Another CAD tool is given in Ref. 30. Every time a new pair of patterns is applied, the approach searches for sequences of rising and falling transitions on segments that terminate (or originate) at a given line. Therefore, if the CAD tool is implemented using segments of size two, every line can have up to four associated transitions. This enhances fault coverage estimation because new paths can be identified when a new sequence of transitions occurs through a line instead of a single transition.

References

1. S.N. Bhatt, F.R.K. Chung, and A.L. Rosenberg, Partitioning Circuits for Improved Testability, *Proc. MIT Conference on Advanced Research in VLSI*, 91, 1986.
2. W.B. Jone and C.A. Papachristou, A Coordinated Approach to Partitioning and Test Pattern Generation for Pseudoexhaustive Testing, *Proc. 26th ACM/IEEE Design Automation Conference*, 525, 1989.
3. D. Kagaris and S. Tragoudas, Cost-Effective LFSR Synthesis for Optimal Pseudoexhaustive BIST Test Sets, *IEEE Transactions on VLSI Systems*, 1, 526, 1993.
4. R. Srinivasan, S.K. Gupta, and M.A. Breuer, An Efficient Partitioning Strategy for Pseudo-Exhaustive Testing, *Proc. 30th ACM/IEEE Design Automation Conference*, 242, 1993.
5. D. Kagaris and S. Tragoudas, Avoiding Linear Dependencies for LFSR Test Pattern Generators, *Journal of Electronic Testing: Theory and Applications*, 6, 229, 1995.
6. B. Reeb and H.J. Wunderlich, Deterministic Pattern Generation for Weighted Random Pattern Testing, *Proc. European Design and Test Conference*, 30, 1996.
7. D. Kagaris, S. Tragoudas, and A. Majumdar, On the use of Counters for Reproducing Deterministic Test Sets, *IEEE Transactions on Computers*, 45, 1405, 1996.
8. S. Narayanan and M.A. Breuer, Asynchronous Multiple Scan Chains, *Proc. IEEE VLSI Test Symposium*, 270, 1995.

9. C.E. Leiserson and J.B. Saxe, Retiming Synchronous Circuitry, *Algorithmica*, 6, 5, 1991.
10. D. Kagaris and S. Tragoudas, Retiming-based Partial Scan, *IEEE Transactions on Computers*, 45, 74, 1996.
11. S.T. Chakradhar and S. Dey, Resynthesis and Retiming for Optimum Partial Scan, *Proc. 31st Design Automation Conference*, 87, 1994.
12. P. Pan and C.L. Liu, Partial Scan with Preselected Scan Signals, *Proc. 32nd Design Automation Conference*, 189, 1995.
13. R. Gupta, R. Gupta, and M.A. Breuer, The BALLAST Methodology for Structured Partial Scan Design, *IEEE Transactions on Computers*, 39, 538, 1990.
14. A. El-Maleh, T. Marchok, J. Rajski, and W. Maly, On Test Set Preservation of Retimed Circuits, *Proc. 32nd ACM/IEEE Design Automation Conference*, 341, 1995.
15. M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.
16. A.P. Stroele and H.-J. Wunderlich, Test Register Insertion with Minimum Hardware Cost, *Proc. International Conference on Computer-Aided Design*, 95, 1995.
17. S. Boubezari and B. Kaminska, A Deterministic Built-In Self-Test Generator Based on Cellular Automata Structures, *IEEE Transactions on Computers*, 44, 805, 1995.
18. D. Kagaris and S. Tragoudas, Cellular Automata for Generating Deterministic Test Sequences, *Proc. European Design and Test Conference*, 77, 1997.
19. J.A. Waicukauski, E.B. Eichelberger, D.O. Florlenza, E. Lindbloom, and T. McCarthy, Fault Simulation for Structured VLSI, *VLSI Systems Design*, 6, 20, 1985.
20. M. Abramovici, P.R. Menon, and D.T. Miller, Critical Path Tracing: An Alternative to Fault Simulation, *IEEE Design and Test of Computers*, 1, 83, 1984.
21. K.-T. Cheng and H.-C. Chen, Delay Testing for Robust Untestable Faults, *Proc. International Test Conference*, 954, 1993.
22. W.K. Lam, A Saldhana, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, Delay Fault Coverage and Performance Tradeoffs, *Proc. Design Automation Conference*, 446, 1993.
23. M.A. Gharaybeh, M.L. Bushnell, and V.D. Agrawal, Classification and Test Generation for Path-Delay Faults Using Stuck-Fault Tests, *Proc. International Test Conference*, 139, 1995.
24. I. Pomeranz, S.M. Reddy, and P. Uppalui, NEST: An Nonenumerative Test Generation Method for Path Delay Faults in Combinational Circuits, *IEEE Transactions on CAD*, 14, 1505, 1995.
25. D. Karayiannis and S. Tragoudas, ATPD: An Automatic Test Pattern Generator for Path Delay Faults, *Proc. International Test Conference*, 443, 1996.
26. J. Savir, Delay Test Generation: A Hardware Perspective, *Journal of Electronic Testing: Theory and Applications*, 10, 245, 1997.
27. M.A. Gharaybeh, M.L. Bushnell, and V.D. Agrawal, An Exact Non-Enumerative Fault Simulator for Path-Delay Faults, *Proc. International Test Conference*, 276, 1996.
28. I. Pomeranz and S.M. Reddy, An Efficient Nonenumerative Method to Estimate the Path Delay Fault Coverage in Combinational Circuits, *IEEE Transactions on Computer-Aided Design*, 13, 240, 1994.
29. D. Kagaris, S. Tragoudas, and D. Karayiannis, Improved Nonenumerative Path Delay Fault Coverage Estimation Based on Optimal Polynomial Time Algorithms, *IEEE Transactions on Computer-Aided Design*, 3, 309, 1997.
30. K. Heragu, V.D. Agrawal, M.L. Bushnell, and J.H. Patel, Improving a Nonenumerative Method to Estimate Path Delay Fault Coverage, *IEEE Transactions on Computer-Aided Design*, 7, 759, 1997.

Long, S.I. "Materials"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

69 Materials

Stephen I. Long
University of California at Santa Barbara

69.1 Introduction

Very-high-speed digital integrated circuit design is a multidisciplinary challenge. First, there are several IC technologies available for very-high-speed applications. Each of these claims to offer unique benefits to the user. In order to choose the most appropriate or cost-effective technology for a particular application or system, the designer must understand the materials, the devices, the limitations imposed by process on yields, and the thermal limitations due to power dissipation.

Second, very-high-speed digital ICs present design challenges if the inherent performance of the devices is to be retained. At the upper limits of speed, there are no digital circuits, only analog. Circuit design techniques formerly thought to be exclusively in the domain of analog IC design are effective in optimizing digital IC designs for highest performance.

Finally, system integration when using the highest-speed technologies presents an additional challenge. Interconnections, clock and power distribution both on-chip and off-chip require much care and often restrict the achievable performance of an IC in a system.

The entire scope of very-high-speed digital design is much too vast to present in a single tutorial chapter. Therefore, we must focus the coverage in order to provide some useful tools for the designer. We will focus primarily on compound semiconductor technologies in order to restrict the scope. Silicon IC design tutorials can be found in other chapters in this handbook. This chapter gives a brief introduction to compound semiconductor materials in order to justify the use of non-silicon materials for the highest-speed applications. The transport properties of several materials are compared. Second, a technology-independent description of device operation for high-speed or high-frequency applications will be given in Chapter 70. The charge control methodology provides insight and connects the basic material properties and device geometry with performance. Chapter 71 describes the design basics of very-high-speed ICs. Static design methods are illustrated with compound semiconductor circuit examples, but are based on generic principles such as noise margin. The transient design methods emphasize analog circuit techniques and can be applied to any technology.

Finally, Chapter 72 describes typical circuit design approaches using FET and bipolar device technologies and presents applications of current interest.

69.2 Compound Semiconductor Materials

The compound semiconductor family is composed of the group III and group V elements shown in Table 69.1. Each semiconductor is formed from at least one group III and one group V element. Group IV

elements such as C, Si, and Ge are used as dopants, as are several group II and VI elements such as Be or Mg for p-type and Te and Se for n-type. Binary semiconductors such as GaAs and InP can be grown in large single-crystal ingot form using the liquid-encapsulated Czochralski method[1] and are the materials of choice for substrates. At the present time, GaAs wafers with a diameter of 100 and 150 mm are most widely used. InP is still limited to 75 mm diameter.

TABLE 69.1 Column III, IV, and V Elements Associated with Compound Semiconductors

B	C	N
Al	Si	P
Ga	Ge	As
In	Sn	Sb

Three or four elements are often mixed together when grown as thin *epitaxial* films on top of the binary substrates. The alloys thus formed allow electronic and structural properties such as bandgap and lattice constant to be varied as needed for device purposes. Junctions between different semiconductors can be used to further control charge transport as discussed in Section 69.4.

69.3 Why III-V Semiconductors?

The main motivation for using the III-V compound semiconductors for device applications is found in their electronic properties when compared with those of the dominant semiconductor material, silicon. Figure 69.1 is a plot of steady-state *electron velocity* of several n-type semiconductors versus electric field. From this graph, we see that at low electric fields the slope of the curves (*mobility*) is higher than that of silicon. High mobility means that the resistivity will be less for III-V n-type materials, and it may be easier to achieve lower access resistance. *Access resistance* is the series resistance between the device contacts and the internal active region. An example would be the base resistance of a bipolar transistor. Lower resistance will reduce some of the fundamental device time constants to be described in Chapter 70 that often dominate device high-frequency performance. Figure 69.1 also shows that the peak electron velocity is higher for III-V materials, and the peak velocity can be achieved at much lower electric fields. High velocity reduces *transit time*, the time required for a charge carrier to travel from its source to its destination, and improves device high-frequency performance, also discussed in Chapter 70. Achieving this high velocity at lower electric fields means that the devices will reach their peak performance at

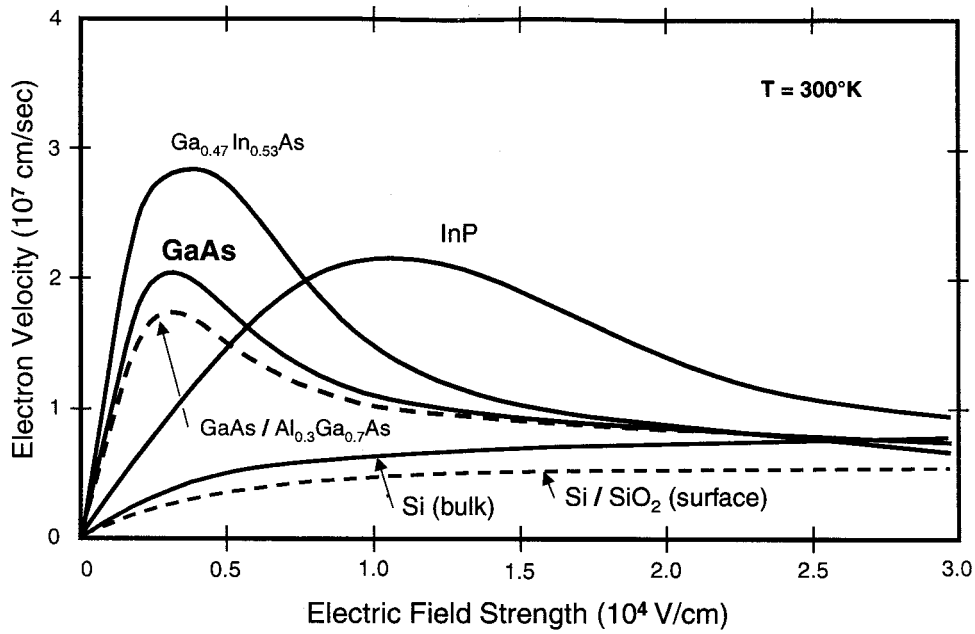


FIGURE 69.1 Electron velocity versus electric field for several n-type semiconductors.

TABLE 69.2 Comparison of Mobilities and Peak Velocities of Several n- and p-type Semiconductors

Semiconductor	E_G (eV)	ε_r	Electron Mobility (cm^2/V-s)	Hole Mobility (cm^2/V-s)	Peak Electron Velocity (cm/s)
Si (bulk)	1.12	11.7	1450	450	N.A.
Ge	0.66	15.8	3900	1900	N.A.
InP	1.35 D	12.4	4600	150	2.1×10^7
GaAs	1.42 D	13.1	8500	400	2×10^7
$Ga_{0.47}In_{0.53}As$	0.78 D	13.9	11,000	200	2.7×10^7
InAs	0.35 D	14.6	22,600	460	4×10^7
$Al_{0.3}Ga_{0.7}As$	1.80 D	12.2	1000	100	—
AlAs	2.17	10.1	280	—	—
$Al_{0.48}In_{0.52}As$	1.92 D	12.3	800	100	—

Note: In bandgap energy column, the symbol "D" indicates direct bandgap; otherwise, it is indirect bandgap. T = 300 K and "weak doping" limit.

lower voltages, which is useful for low-power, high-speed applications. Mobilities and peak velocities of several semiconductors are compared in Table 69.2.

On the other hand, as also shown in Table 69.2, p-type III-V semiconductors have rather poor hole mobility when compared with elemental semiconductor materials such as silicon or germanium. Holes also reach their peak velocities at much higher electric fields than electrons. Therefore, p-type III-V materials needed for the base of a bipolar transistor, for example, are used, but their thickness must be extremely small to avoid degradation in transit time. Lateral distances must also be small to avoid excessive series resistance. CMOS-like complementary FET technologies have also been developed,[2] but their performance has been limited by the poorer speed of the p-channel devices.

69.4 Heterojunctions

In the past, most semiconductor devices were composed of a single semiconductor element, such as silicon or gallium arsenide, and employed n- and p-type doping to control charge transport. Figure 69.2(a) illustrates an energy band diagram of a semiconductor with uniform composition that is in an applied electric field. Electrons will drift downhill and holes will drift uphill in the applied electric field. The electrons and/or holes could be produced by doping or by ionization due to light. In a *heterogeneous* semiconductor as shown in Fig. 69.2(b), the bandgap can be graded from wide bandgap on the left to narrow on the right by varying the composition. In this case, even without an applied electric field, a built-in quasi-electric field is produced by the bandgap variation that will transport both holes and electrons in the *same* direction.

The abrupt *heterojunction* formed by an atomically abrupt transition between AlGaAs and GaAs, shown in the energy band diagram of Fig. 69.3, creates discontinuities in the valence and conduction bands. The conduction band energy discontinuity is labeled ΔE_C and the valence band discontinuity, ΔE_V. Their sum equals the energy bandgap difference between the two materials. The potential energy steps caused by these discontinuities are used as barriers to electrons or holes. The relative sizes of these potential barriers depend on the composition of the semiconductor materials on each side of the heterojunction. In this example, an electron barrier in the conduction

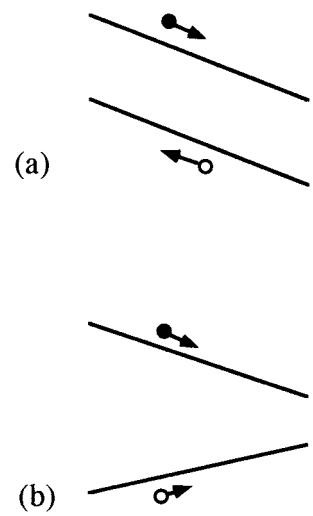


FIGURE 69.2 (a) Homogeneous semiconductor in uniform electric field, and (b) Heterogeneous semiconductor with graded energy gap. No applied electric field.

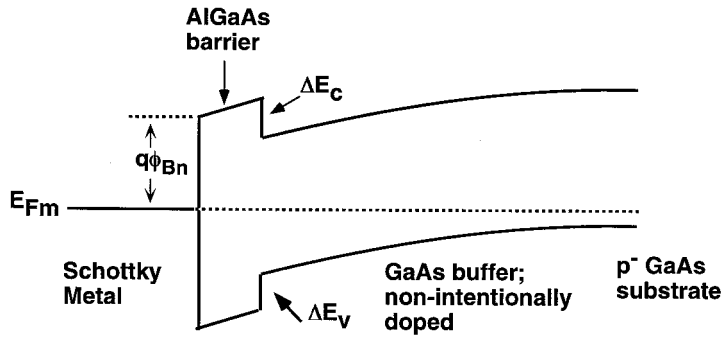


FIGURE 69.3 Energy band diagram of an abrupt heterojunction.

band is used to confine carriers into a narrow potential energy well with triangular shape. Quantum well structures such as these are used to improve device performance through two-dimensional charge transport channels, similar to the role played by the inversion layer in MOS devices. The structure and operation of heterojunctions in FETs and BJTs will be described in Chapter 70.

The overall principle of the use of heterojunctions is summarized in a *Central Design Principle*:

> "Heterostructures use energy gap variations in addition to electric fields as forces acting on holes and electrons to control their distribution and flow."[3,4]

The energy barriers can control motion of charge both across the heterojunction and in the plane of the heterojunction. In addition, heterojunctions are most widely used in light-emitting devices, since the compositional differences also lead to either stepped or graded index of refraction, which can be used to confine, refract, and reflect light. The barriers also control the transport of holes and electrons in the light-generating regions.

Figure 69.4 shows a plot of bandgap versus lattice constant for many of the III-V semiconductors.[3] Consider GaAs as an example. GaAs and AlAs have the same lattice constant (approximately 0.56 nm) but different bandgaps (1.4 and 2.2 eV, respectively). An alloy semiconductor, AlGaAs, can be grown

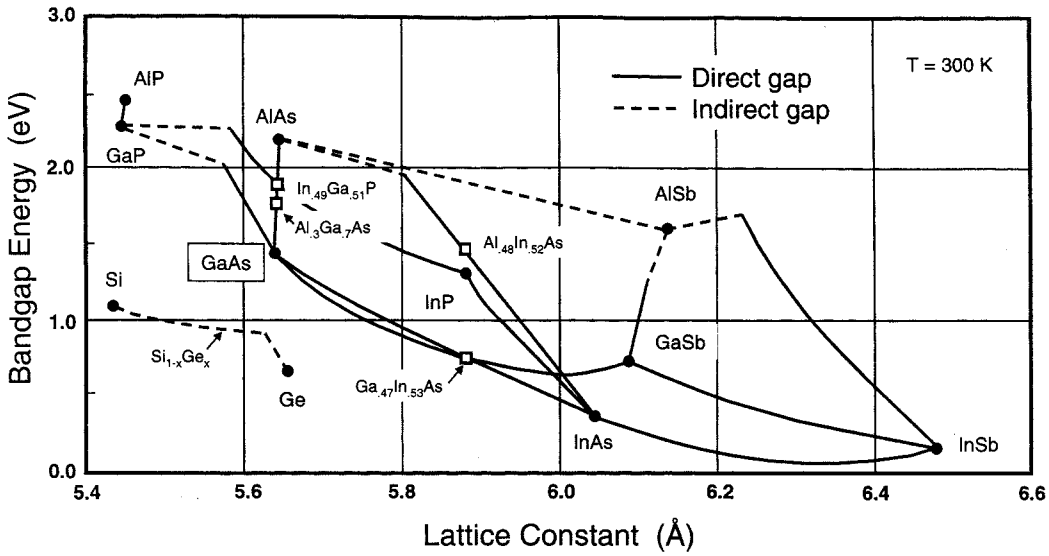


FIGURE 69.4 Energy bandgap versus lattice constant for compound semiconductor materials.

epitaxially on a GaAs substrate wafer using standard growth techniques. The composition can be selected by the Al-to-Ga ratio, giving a bandgap that can be chosen across the entire range from GaAs to AlAs. Since both lattice constants are essentially the same, very low lattice mismatch can be achieved for any composition of $Al_xGa_{1-x}As$. Lattice matching permits low defect density, high-quality materials to be grown that have good electronic and optical properties. It quickly becomes apparent from Fig. 69.4, however, that a requirement for lattice matching to the substrate greatly restricts the combinations of materials available to the device designer. For electron devices, the low mismatch GaAs/AlAs alloys, GaSb/AlSb alloys, $Al_{.48}In_{.52}As/InP/Ga_{.47}In_{.53}As$ and $GaAs/In_{.49}Ga_{.51}As$ combinations alone are available. Efforts to utilize combinations such as GaP on Si or GaAs on Ge that lattice match have been generally unsuccessful because of problems with interface structure, polarization, and autodoping.

For several years, lattice matching was considered to be a necessary condition if mobility-damaging defects were to be avoided. This barrier was later broken when it was discovered that high-quality semiconductor materials could still be obtained although lattice-mismatched if the thickness of the mismatched layer is sufficiently small.[5,6] This technique, called *pseudomorphic* growth, opened another dimension in III-V device technology, and allowed device structures to be optimized over a wider range of bandgap for better electron or hole dynamics and optical properties.

Two of the pseudomorphic systems that have been very successful in high-performance millimeter-wave FETs are the InAlAs/InGaAs/GaAs and InAlAs/InGaAs/InP systems. The $In_xGa_{1-x}As$ layer is responsible for the high electron mobility and velocity which both improve as the In concentration x is increased. Up to $x = 0.25$ for GaAs substrates and $x = 0.80$ for InP substrates have been demonstrated and result in great performance enhancements when compared with lattice-matched combinations.

Estriech, D.B. "Compound Semiconductor Devices for Digital Circuits"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

70

Compound Semiconductor Devices for Digital Circuits

Donald B. Estreich
Hewlett-Packard Company

70.1 Introduction

An *active device* is an electron device, such as a transistor, capable of delivering power amplification by converting dc bias power into time-varying signal power. It delivers a greater energy to its load than if the device were absent. The *charge control* framework[7-9] discussed below presents a unified understanding of the operation of all electron devices and simplifies the comparison of the several active devices used in digital integrated circuits.

70.2 Unifying Principle for Active Devices: Charge Control Principle

Consider a generic electron device as represented in Fig. 70.1. It consists of three electrodes encompassing a charge *transport region*. The transport region is capable of supporting charge flow (electrons as shown in the figure) between an *emitting electrode* and a *collecting electrode*. A third electrode, called the *control electrode*, is used to establish the electron concentration within the transport region. Placing a *control charge*, Q_C, on the control electrode establishes a *controlled charge*, denoted as $-Q$, in the transport region. The operation of active devices depends on the *charge control principle*[7]:

> Each charge placed upon the control electrode can at most introduce an equal and opposite charge in the transport region between the emitting and collecting electrode.

At most, we have the relationship, $|-Q| = |Q_C|$. Any parasitic coupling of the control charge to charge on the other electrodes, or remote parts of the device, will decrease the controlled charge in the transport region, that is $|-Q| < |Q_C|$ more generally. For example, charge coupling between the control

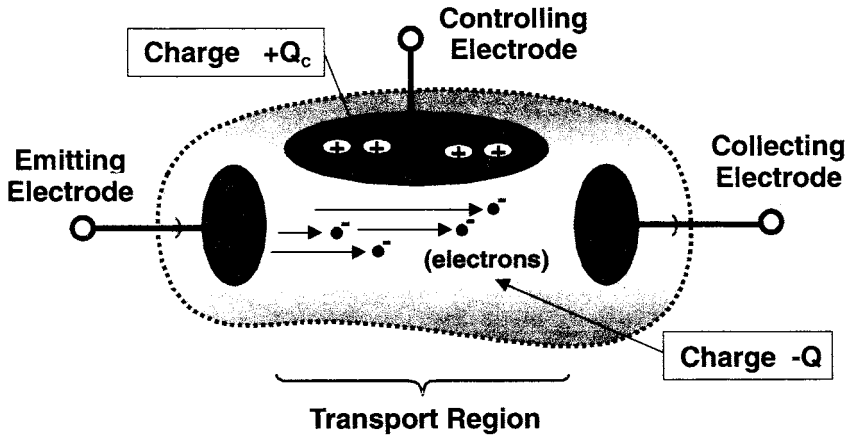


FIGURE 70.1 Generic charge control device consisting of three electrodes embedded around a charge transport region.

electrode and the collecting electrode forms a feedback or output capacitance, say $\mathbf{C_o}$. Time variation of Q_C leads to the modulation of the current flow between emitting and collecting electrodes.

The generic structure in Fig. 70.1 could represent any one of a number of active devices (e.g., vacuum tubes, unipolar transistors, bipolar transistors, photoconductors, etc.). Hence, charge control analysis is very broad in scope, since it applies to all electronic transistors.

Starting from the charge control principle, we associate two characteristic time constants with an active device, thereby leading to a first-order description of its behavior. Application of a potential difference between the emitting and collecting electrodes, say V_{CC}, establishes an electric field in the transport region. Electrons in the transport region respond to the electric field and move across this region with a *transit time* τ_r. The transit time[1] is the first of the two important characteristic times used in charge control modeling. With charge $-Q$ in the transit region, the static (dc) current I_o between emitting and collecting electrodes is

$$I_o = -Q/\tau_r = Q_c/\tau_r \tag{70.1}$$

A simple interpretation of τ_r is as follows: τ_r is equal to the length l of the transport region, divided by the average velocity of transit (i.e., $\tau_r = l/\langle v \rangle$). From this perspective, a charge of $-Q$ (coulombs) is swept out the collecting electrode every τ_r seconds.

Now consider Fig. 70.2, showing the common-emitting electrode connection of the active device of Fig. 70.1 connected to input and output (i.e., load) resistances, say R_{in} and R_L, respectively. The second characteristic time of importance can now be defined: It is the *lifetime time constant*, and we denote it by τ. It is a measure of how long a charge placed on the control electrode will remain on the control terminal. The lifetime time constant is established in one of several ways, depending on the physics of the active device and/or its connection. The controlling charge may "leak away" by (1) discharging through the external resistor R_{in} as typically happens with FET devices, (2) recombining with intermixed oppositely charged carriers within the device (e.g., base recombination in a bipolar transistor), or (3) discharging through an internal shunt leakage path within the device. The dc current flowing to replenish the lost control charge is given by

$$I_{in} = -Q/\tau = Q_c/\tau \tag{70.2}$$

The *static* (*dc*) *current gain* G_I of a device is defined as the current delivered to the output, divided by the current replenishing the control charge during the same time period. Where in τ seconds charge $-Q$

[1]The transit time τ_r is best interpreted as an average transit time per carrier (electron). We note that $1/\tau_r$ is common to all devices — it is related to a device's ultimate capability to process information.

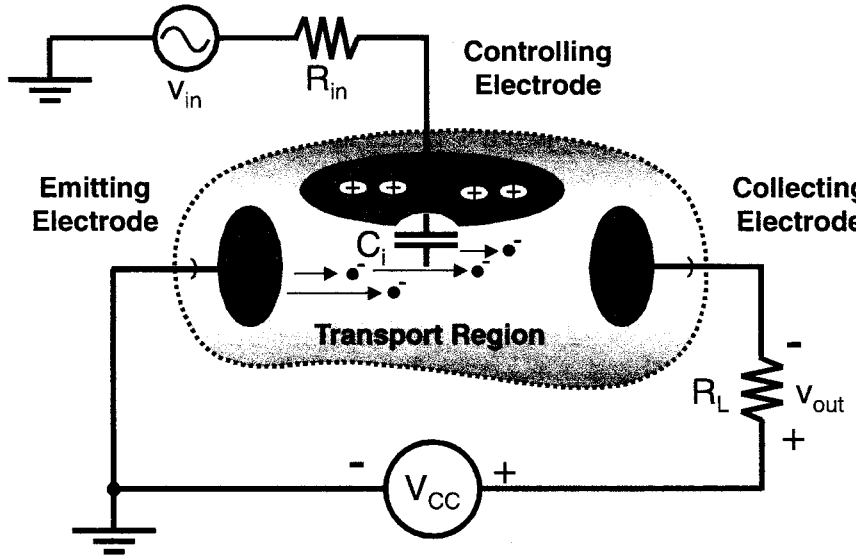


FIGURE 70.2 Generic charge control device of Fig. 70.1 connected to input and output resistors, R_{in} and R_L, respectively, with bias voltage and input signal applied.

is both lost and replenished, charge Q_c times the ratio τ/τ_r has been supplied to the output resistor R_L. In symbols, the static current gain is

$$G_I = I_o/I_{in} = \tau/\tau_r \tag{70.3}$$

provided $|-Q| = |Q_C|$ holds.

In the dynamic case, the process of *small-signal amplification* consists of an incremental variation of the control charge Q_c directly resulting in an incremental change in the controlled charge, $-Q$. The resulting variation in output current flowing in the load resistor translates into a time-varying voltage v_o. The charge control formalism holds just as well for large-signal situations. In the large-signal case, the changes in control charge are no longer small incremental changes. Charge control analysis under large charge variations is less accurate due to the simplicity of the model, but still very useful for approximate switching calculations in digital circuits.

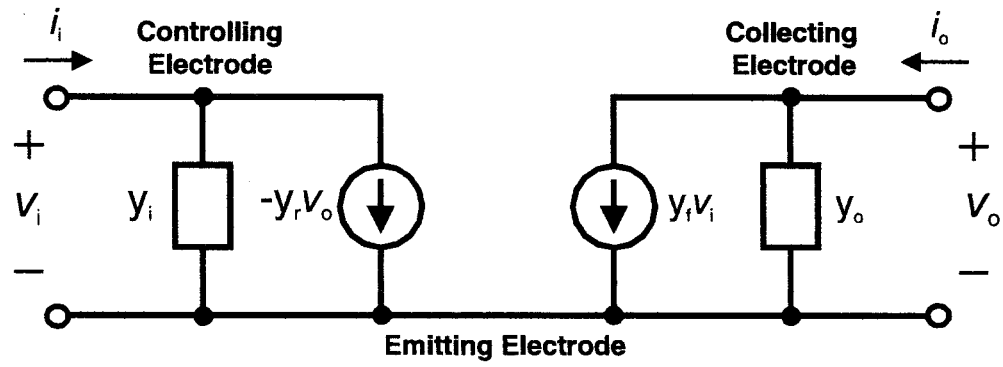
An important dynamic parameter is the *input capacitance* C_i of the active device. Capacitance C_i is a measure of the work required to introduce a charge carrier in the transport region. Capacitance C_i is given by the change in charge Q for a corresponding change in input voltage v_{in}. It is desirable to maximize C_i in an active device. The *transconductance* g_m is calculated from

$$g_m = \left(\frac{\partial I_o}{\partial v_{in}}\right)_{v_o} = \left(\frac{\partial I_o}{\partial Q}\right) \cdot \left(\frac{\partial Q}{\partial v_{in}}\right) \tag{70.4}$$

The first partial derivative on the right-hand side of Eq. 70.4 is simply $(1/\tau_r)$, and the second partial derivative is C_i. Hence, the transconductance g_m is the ratio

$$g_m = \frac{C_i}{\tau_i} \tag{70.5}$$

A physical interpretation of g_m is the ratio of the work required to introduce a charge carrier to the average transit time of a charge carrier in the transport region. The transconductance is one of the most commonly used device parameters in circuit design and analysis.



$$y_i = \frac{C_i}{\tau} \qquad y_f \cong g_m = \frac{C_i}{\tau_r}$$

$$y_r = -\frac{C_o}{\tau} \qquad y_o = \frac{C_o}{\tau_r}$$

FIGURE 70.3 Two-port, small-signal, admittance charge control model with the emitting electrode selected as the common terminal to both input and output.

In addition to C_i, another capacitance, say C_o, is introduced and associated with the collecting electrode. Capacitance C_o accounts for charge on the collecting electrode coupled to either static charge in the transport region or charge on the control electrode. A non-zero C_o indicates that the coupling between the controlling electrode and the charge in transit is less than unity (i.e., $|-Q| < |Q_C|$).

For small-signal analysis the capacitance parameters are usually taken at fixed numbers evaluated about the device's bias state. When using charge control in the large-signal case, the capacitance parameters must include the voltage dependencies. For example, the input capacitance C_i can be strongly dependent upon the control electrode to emitting electrode and collecting electrode potentials. Hence, during the change in bias state within a device, the magnitude of the capacitance C_i is time varying. This variation can dramatically affect the switching speed of the active device. Parametric dependencies on the instantaneous bias state of the device are at the heart of accurate modeling of large-signal or switching behavior of active devices.

We introduce the *small-signal admittance charge control model* shown in Fig. 70.3. This model uses the emitting electrode as the common terminal in a two-port connection. The transconductance g_m is the magnitude of the real part of the *forward admittance* y_f and is represented as a voltage-controlled current source positioned from collecting-to-emitting electrode. The *input admittance*, denoted by y_i, is equivalent to (C_i/τ), where τ is the control charge lifetime time constant. Parameter y_i can be expressed in the form $(g_i + sC_i)$ where $s = j\omega$. An *output admittance*, similarly denoted by y_o, is given by (C_o/τ_r) where τ_r is the transit time and, in general $y_o = (g_o + sC_o)$. Finally, the *output-to-input feedback admittance* y_r is included using a voltage-controlled current source at the input. Often, y_r is small enough to approximate as zero (the model is then said to be *unilateral*).

Consider the frequency dependence of the *dynamic* (*ac*) *current gain* G_i. The low-frequency current gain is interpreted as follows: an incremental charge q_c is introduced on the control electrode with lifetime τ. This produces a corresponding incremental charge –q in the transport region. Charge –q is swept across the transport region every transit time τ_r seconds. In time τ, charge –q crosses the transit region τ/τ_r times, which is identically equal to the low-frequency current gain.

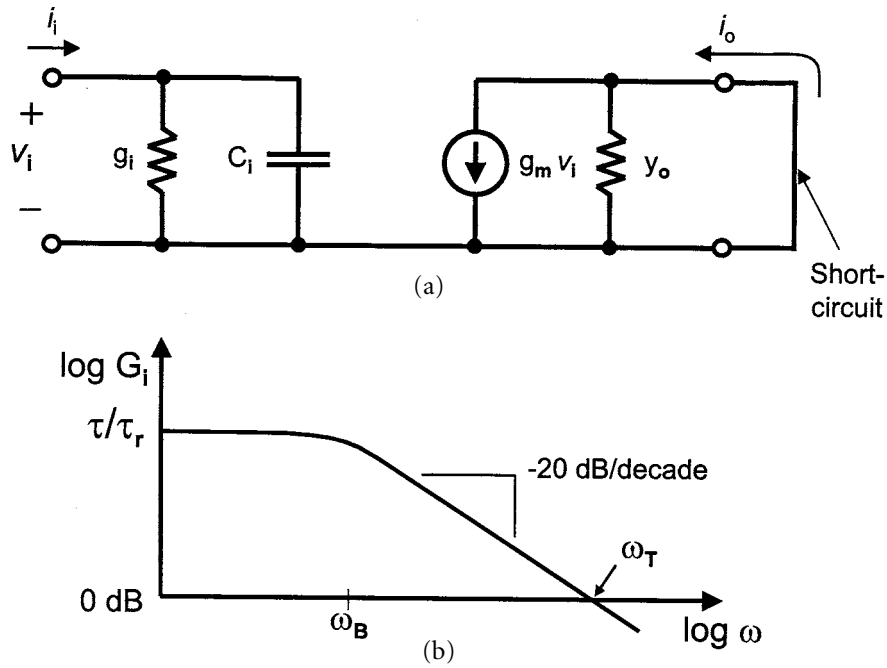


FIGURE 70.4 (a) Small-signal admittance model with output short-circuited, and (b) magnitude of the small-signal current gain G_i plotted as a function of frequency. The unity current gain crossover (i.e., $G_i = 1$) defines the parameter f_T (or ω_T).

The lifetime τ associated with the control electrode arises from charge "leaking off" the controlling electrode. This is modeled as an RC time constant at the input of the equivalent circuit shown in Fig. 70.4(a) with τ equal to $R_{eq}C_i$. R_{eq} is the equivalent resistance presented to capacitor C_i. That is, R_{eq} is determined by the parallel combination of $1/g_i$ and any external resistance at the input. The *break frequency* ω_B associated with the control electrode is

$$\omega_B = \frac{1}{\tau} = \frac{1}{R_{eq}C_i} \tag{70.6}$$

When the charge on the control electrode varies at a rate ω less than ω_B, G_i is given by τ/τ_r because charge "leaks off" the controlling electrode faster than $1/\omega$. Alternatively, when ω is greater than ω_B, G_i decreases with increasing ω because the applied signal charge varies upon the control electrode more rapidly than $1/\tau$. In this case, G_i is inversely proportional to ω, that is,

$$G_i = \frac{1}{\omega\tau_r} = \frac{\omega_T}{\omega} \tag{70.7}$$

where ω_T is the common-emitter *unity current gain frequency.* At $\omega = \omega_T$ ($= 2\pi f_T$), the ac current gain equals unity, as illustrated in Fig. 70.4(b).

Consider the *current gain-bandwidth product* $G_i\Delta f$. A purely capacitive input impedance cannot define a bandwidth. However, a finite real impedance always appears at the input terminal in any practical application. Let R_i be the *effective input resistance* of the device (i.e., R_i will be equal to $(1/g_i)$ in parallel with the external resistance R_{in}). Since the input current is equal to q_c/τ and the output current is equal to q/τ_r, the current gain-bandwidth product becomes

$$G_i \cdot \Delta f = \frac{q/\tau_r}{q_c/\tau} \frac{\omega}{2\pi} \tag{70.8}$$

For $\omega \gg \omega_B$, at $\tau = 1/\omega$, and assuming $|q_c| = |-q|$,

$$G_i \cdot \Delta f = \frac{1}{2\pi\tau_r} = \frac{\omega_T}{2\pi} = f_T \tag{70.9}$$

f_T (or ω_T) is a widely quoted parameter used to compare or "benchmark" active devices. Sometimes, f_T (or ω_T) is interpreted as a measure of the maximum speed a device can drive a replica of itself. It is easy to compute and historically has been easy to measure with bridges and later using S-parameters. However, f_T does have interpretative limitations because it is defined as current into a short-circuit output. Hence, it ignores input resistance and output capacitance effects upon actual circuit performance.

Likewise, voltage and power gain expressions can be derived. It is necessary to define the output impedance before either can be quantified. Let R_o be the *effective output resistance* at the output terminal of the active device. Assuming both the input and output RC time constants to be identical (i.e., $R_iC_i = R_oC_o$), the *voltage gain* G_v can be expressed in terms of G_i as

$$G_v = G_i \frac{R_o}{R_i} = G_i \frac{C_i}{C_o} \tag{70.10}$$

where R_o is the parallel equivalent output resistance from all resistances at the output node.

The *power gain* G_p is computed from the product of $G_i \cdot G_v$ along with the *power gain-bandwidth product*. These results are listed in Table 70.1 as summarized from Johnson and Rose.[7] These simple expressions are valid for all devices as interpreted from the charge control perspective. They provide for a first-order comparison, in terms of a few simple parameters, among the active devices commonly

TABLE 70.1 Charge Control Relations for All Active Devices

Parameter	Symbol	Expression
Transconductance	g_m	$\frac{C_i}{\tau_r} \Leftrightarrow \omega_T C_i$
Current amplification	G_i	$\frac{1}{\omega\tau_r} \Leftrightarrow \frac{\omega_T}{\omega}$
Voltage amplification	G_v	$\frac{1}{\omega\tau_r}\frac{C_i}{C_o} \Leftrightarrow \frac{\omega_T}{\omega}\frac{C_i}{C_o}$
Power amplification	$G_p = G_iG_v$	$\frac{1}{\omega^2\tau_r^2}\frac{C_i}{C_o} \Leftrightarrow \frac{\omega_T^2}{\omega^2}\frac{C_i}{C_o}$
Current gain-bandwidth product	$G_i \cdot \Delta f$	$\frac{1}{\tau_r} \Leftrightarrow \omega_T$
Voltage gain-bandwidth product	$G_v \cdot \Delta f$	$\frac{1}{\tau_r}\frac{C_i}{C_o} \Leftrightarrow \omega_T\frac{C_i}{C_o}$
Power gain-bandwidth product	$G_p \cdot \Delta f^2$	$\frac{1}{\tau_r^2}\frac{C_i}{C_o} \Leftrightarrow \omega_T^2\frac{C_i}{C_o}$

Note: Table assumes $R_iC_i = R_oC_o$. (After Johnson and Rose (Ref. 7), March 1959. © 1959 IEEE, reproduced with permission of IEEE.)

available. From an examination of Table 70.1, it is evident that maximizing C_i and minimizing τ_r leads to higher transconductance, higher parametric gains, and greater frequency response. This is an important observation in understanding how to improve upon the performance of any active device.

Whereas f_T has limitations, the frequency at which the maximum power gain extrapolates to unity, denoted by ω_{max}, is often a more useful indicator of device performance. The primary limitation of ω_{max} is that it is very difficult to calculate and is usually extrapolated from S-parameter measurements in which the extrapolation is approximate at best.

70.3 Comparing Unipolar and Bipolar Transistors

Unipolar transistors are active devices that operate using only a single charge carrier type, usually electrons, in their transport region. *Field-effect transistors* fall into the unipolar classification. In contrast, *bipolar transistors* depend on positive and negative charged carriers (i.e., both majority and minority carriers) within the transport region. A fundamental difference arises from the relative locations of the control electrode and transport region — in unipolar devices, they are physically separated, whereas in bipolar devices, they are merged into the same physical region (i.e., base region). Before reviewing the physical operation of each, transport in semiconductors is briefly reviewed.

Charge Transport in Semiconductors[10-12]

Bulk semiconducting materials are useful because their conductivity can be controlled over many orders of magnitude by changing the doping level. Both electrons and holes[10] can conduct current in semiconductors. In integrated circuits metal, semiconductor, and insulator layers are used together in precisely positioned shapes and thicknesses to form useful device and circuit functions.

Fig. 69.1 illustrates the behavior of electron velocity as a function of local electric field strength for several important semiconducting materials. Two characteristic regions of behavior can be identified: a *linear* or *ohmic* region at low electric fields, and a *velocity-saturated* region at high fields. At low fields, current transport is proportional to the carrier's mobility. Mobility is a measure of how easily carriers move through a material.[10] At high fields, carriers saturate in velocity; hence, current levels will correspondingly saturate in active devices. The data in Fig. 69.1 assumes low doping levels (i.e., $N_x < 10^{15}$ cm^{-3}). The dashed curve represents transport in a GaAs quantum well formed adjacent an $Al_{0.3}Ga_{0.7}As$ layer — in this case, *interface scattering* lowers the mobility. A similar situation is found for transport in silicon at a semiconductor–oxide interface such as found in metal-oxide-semiconductor (MOS) devices.

Several general conclusions can be extracted from this data:

1. Compound semiconductors generally have higher electron mobilities than silicon.
2. At high fields (say E > 20,000 V/cm), saturated electron velocities tend to converge to values close to 1×10^7 cm/s.
3. Many compound semiconductors show a transition region between low and high electric field strengths with a *negative differential mobility* due to electron transfer from the Γ ($\mathbf{k} = 0$) valley to conduction band valleys with higher effective masses (this gives rise to the *Gunn Effect*[13]).

Hole mobilities are much lower than electron mobilities in all semiconductors. Saturated velocities of holes are also lower at higher electric fields. This is why n-channel field-effect transistors have higher performance than p-channel field-effect transistors, and why npn bipolar transistors have higher performance than pnp bipolar transistors. Table 69.2 compares electron and hole mobilities for several semiconducting materials.

Field-Effect (Unipolar) Transistor[14-16]

Fig. 70.5(a) shows a conceptual view of an n-channel field-effect transistor (FET). As shown, the n-type channel is a homogeneous semiconducting material of thickness b, with electrons supporting the drain-to-source current. A p-type channel would rely on mobile holes for current transport and all voltage polarities would be exactly reversed from those shown in Fig. 70.5(a). The control charge on the gate

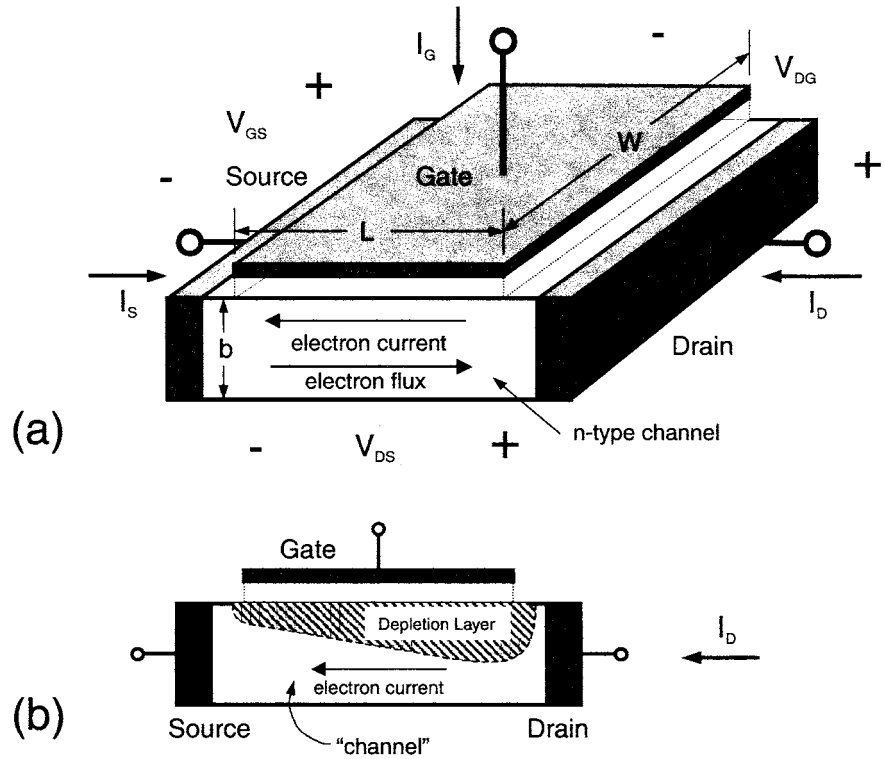


FIGURE 70.5 (a) Conceptual view of a field-effect transistor with the channel sandwiched between source and drain ohmic contacts and a gate control electrode in close proximity; and (b) cross-sectional view of the FET with a depletion layer shown such as would be present in a compound semiconductor MESFET.

region (of length L and width W) establishes the number of conduction electrons per unit area in the channel by electrostatic attraction or exclusion. The cross-section on the FET channel in Fig. 70.5(b) shows a *depletion layer*, a region void of electrons, as an intermediary agent between the control charge and the controlled charge. This depletion region is present in *junction* FET and *Schottky barrier junction* (i.e., *metal-semiconductor junction*) MESFET structures.

In all FET structures, the gate is physically separated from the channel. By physically separating the control charge from the controlled charge, the gate-to-channel impedance can be very large at low frequencies. The gate impedance is predominantly capacitive and, typically, very low gate leakage currents are observed in high-quality FETs. This is a distinguishing feature of the FET — its high input impedance is desirable for many circuit applications.

The channel, positioned between the source and drain ohmic contacts, forms a resistor whose resistance is modulated by the applied gate-to-channel voltage. We know the gate potential controls the channel charge by the charge control relation. Time variation of the gate potential translates into a corresponding time variation of the drain current (and the source current also). Therefore, transconductance g_m is the natural parameter to describe the FET from this viewpoint.

Fig. 70.6(a) shows the I_D-V_{DS} characteristic of the n-channel FET in the common-source connection with constant electron mobility and a long channel assumed. Two distinct operating regions appear in Fig. 70.6(a) — the *linear* (i.e., *non-saturated*) region, and the *saturated* region, separated by the dashed parabola. The origin of current saturation corresponds to the onset of *channel pinch-off* due to carrier exclusion at the drain end of the channel. Pinch-off occurs when the drain voltage is positive enough to deplete the channel completely of electrons at the drain end; this corresponds to a gate-to-source voltage equal to the *pinch-off voltage*, denoted as $-V_p$ in Figs. 70.6(b) and (c). For constant V_{DS} in the saturated region, the I_D vs. V_{GS} *transfer curve* approximates "square law" behavior; that is,

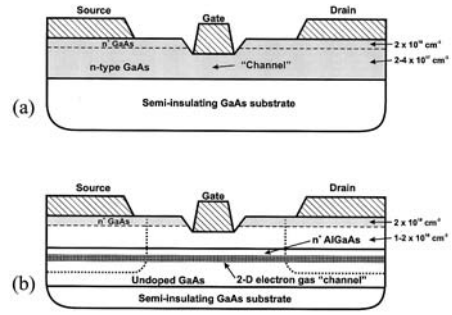


FIGURE 70.6 (a) Field-effect transistor drain current (I_D) versus drain-to-source voltage (V_{DS}) characteristic with the gate-to-source voltage (V_{GS}) as a stepped parameter; (b) I_D versus V_{GS} "transfer curve" for a constant V_{DS} in the saturated region of operation, revealing its "square-law" behavior; (c) transconductance g_m versus V_{GS} for a constant V_{DS} in saturated region of operation corresponding to the transfer curve in (b). These curves assume constant mobility, no velocity saturation, and the "long-channel FET approximation."

$$I_D = I_{D,sat} = I_{DSS}\left[1 - \frac{V_{GS}}{\left(-V_P\right)}\right]^2 \quad \text{for } -V_P \le V_{GS} \le \varphi \tag{70.11}$$

where I_{DSS} is the drain current when $V_{GS} = 0$, and φ is a built-in potential associated with the gate-to-channel junction or interface (e.g., a metal-semiconductor Schottky barrier as in the MESFET). The symbol $I_{D,sat}$ denotes the drain current in the saturated region of operation. Transconductance g_m is linear with V_{GS} for the saturation transfer characteristic of Eq. (70.11) and is approximated by

$$g_m = \frac{\partial I_D}{\partial V_{GS}} \cong 2\frac{D_{DSS}}{V_P}\left[1 - \frac{V_{GS}}{\left(-V_P\right)}\right] \quad \text{for } -V_P \le V_{GS} \le \varphi \tag{70.12}$$

Equations 70.11 and 70.12 are plotted in Figs. 70.6(b) and (c), respectively.

Bipolar Junction Transistors (Homojunction and Heterojunction)[13-17]

In the *bipolar junction transistor* (BJT), both the control charge and the controlled charge occupy the same region (i.e., the *base region*). A control charge is injected into the base region (i.e., this is the base current flowing in the base terminal), causing the emitter-to-base junction's potential barrier to be lowered. Barrier lowering results in majority carrier diffusion across the emitter-to-base junction. Electrons diffuse into the base and holes into the emitter in the npn BJT shown in Fig. 70.7. By controlling the emitter-to-base junction's physical structure, the dominant carrier diffusion across this n-p junction should be injection into the base region. For our npn transistor, the dominant carrier transport is electron diffusion into the base region where the electrons are minority carriers. They transit the base region, of base width W_b, by both diffusion and drift. When collected at the collector-to-base junction, they establish the collector current I_C. The base width must be short to minimize recombination in the base region (this is reflected in the current gain parameter commonly used with BJT and HBT devices).

In *homojunction* BJT devices, the emitter and base regions have the same bandgap energy. The respective carrier injection levels are set by the ratio of the emitter-to-base doping levels. For high emitter efficiency, that is, the number of carriers diffusing into the base being much greater than the number of carriers simultaneously diffusing into the emitter, the emitter must be much more heavily doped than the base region. This places a limit on the maximum doping level allowed in the base of the homojunction

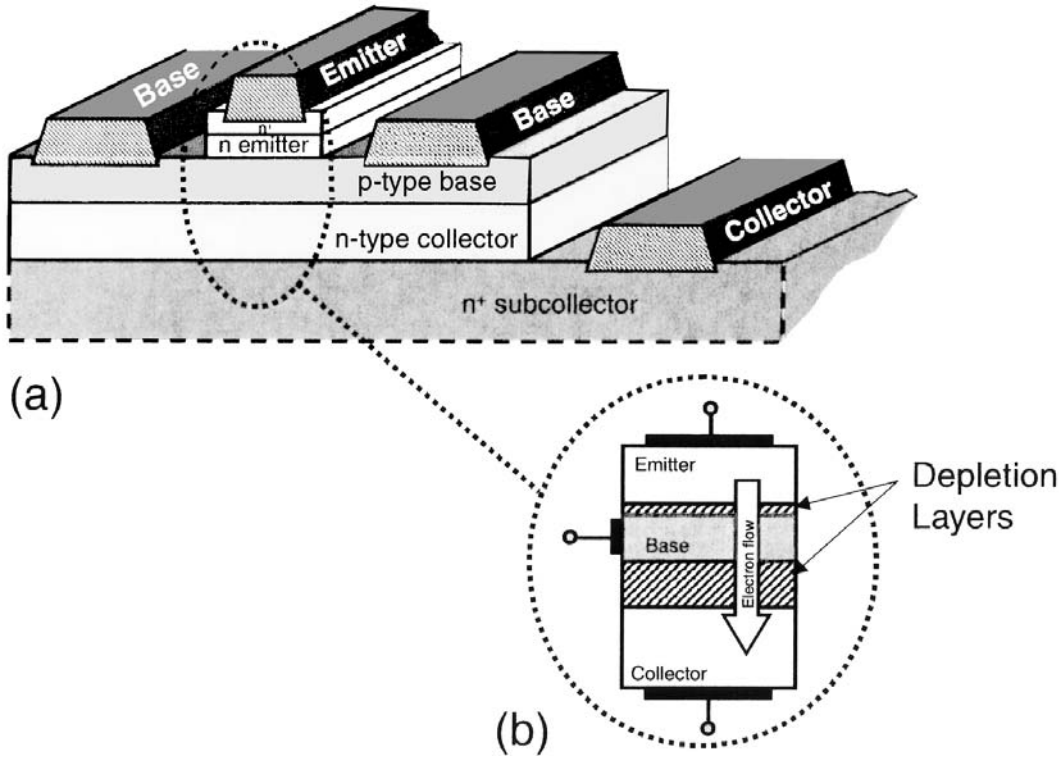


FIGURE 70.7 (a) Conceptual view of a bipolar junction transistor with the base region sandwiched between emitter and collector regions. Structure is representative of a compound semiconductor heterojunction bipolar transistor. (b) Simplified cross-sectional view of a vertically structured BJT device with primary electron flow represented by large arrow.

BJT, thereby leading to higher base resistance than the device designer would normally desire.[16] In contrast, the *heterojunction bipolar transistor* (HBT) uses different semiconducting materials in the base and emitter regions to achieve high emitter efficiency. A wider bandgap emitter material allows for high emitter efficiency while allowing for higher base doping levels which in turn lowers the parasitic base resistance. An example of a wider bandgap emitter transistor is shown in Fig. 70.8. In this example, the emitter is AlGaAs whereas the base and collector are formed with GaAs. Figure 70.8 shows the band diagram under normal operation with the emitter–base junction forward-biased and the collector–base junction reverse-biased. The discontinuity in the valence band edge at the emitter–base heterojunction is the origin of the reduced diffusion into the emitter region. The injection ratio determining the emitter efficiency depends exponentially on this discontinuity. If ΔE_g is the valence band discontinuity, the injection ratio is proportional to the exponential of ΔE_g normalized to the thermal energy kT[4]:

$$\frac{J_n}{J_p} \propto \exp\left(-\Delta E_g / kT\right) \tag{70.13}$$

For example, ΔE_g equal to 8kT gives an exponential factor of approximately 8000, thereby leading to an emitter efficiency of nearly unity, as desired. The use of the emitter-base band discontinuity is a very efficient way to hold high emitter efficiencies.

In bipolar devices, the collector current I_C is given by the exponential of the *base-emitter forward voltage* V_{BE} normalized to the *thermal voltage* kT/q

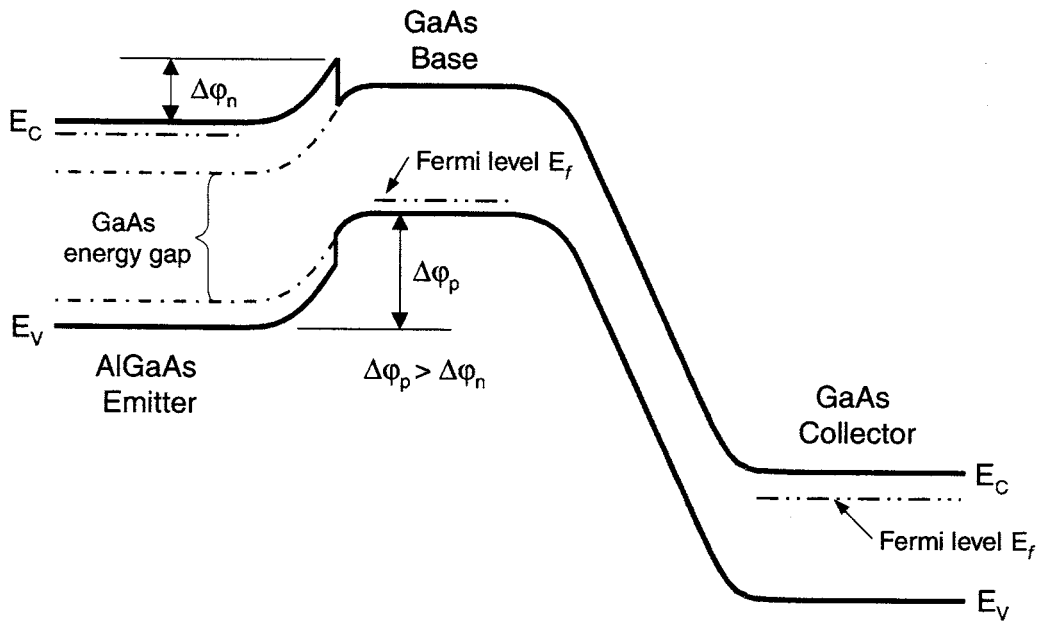


FIGURE 70.8 The bandgap diagram for an HBT AlGaAs/GaAs device with the wider bandgap for the AlGaAs emitter (solid line) compared with a homojunction GaAs BJT emitter (dot-dash line). The double dot-dashed line represents the Fermi level in each region.

$$I_C = I_S \exp\left(qV_{BE}/kT\right) \tag{70.14}$$

The saturation current I_S is given by a quantity that depends on the structure of the device; it is inversely proportional to the base doping charge Q_{BASE} and proportional to the device's area A, namely

$$I_S = \frac{qADn_i^2}{Q_{BASE}} \tag{70.15}$$

where the other symbols have their usual meanings (D is the minority carrier *diffusion constant* in the base, n_i is the *intrinsic carrier concentration* of the semiconductor, and q is the *electron's charge*).

A typical collector current versus collector-emitter voltage characteristic, for several increasing values of (forward-biased) emitter-base voltages, is shown in Fig. 70.9(a). Note the similarity to Fig. 70.6(a), with the BJT having a quicker turn-on for low V_{CE} values compared with the softer knee for the FET. The transconductance of the BJT and HBT is found by taking the derivative of Eq. 70.14, thus

$$g_m = \frac{\partial I_C}{\partial V_{BE}} = \frac{qI_S}{kT}\exp\left(qV_{BE}/kT\right) \tag{70.16}$$

Both I_C and g_m are of exponential form, as observed in Fig. 70.9; Eqs. 70.14 and 70.16 are plotted in Figs. 70.9(b) and (c), respectively. The transconductance of the BJT/HBT is generally much larger than that of the best FET devices (this can be verified by comparing Eq. (70.12) with Eq. (70.16) with typical parameter values inserted). This has significant circuit design advantages for the BJT/HBT devices over the FET devices because high transconductance is needed for high current drive to charge load capacitance

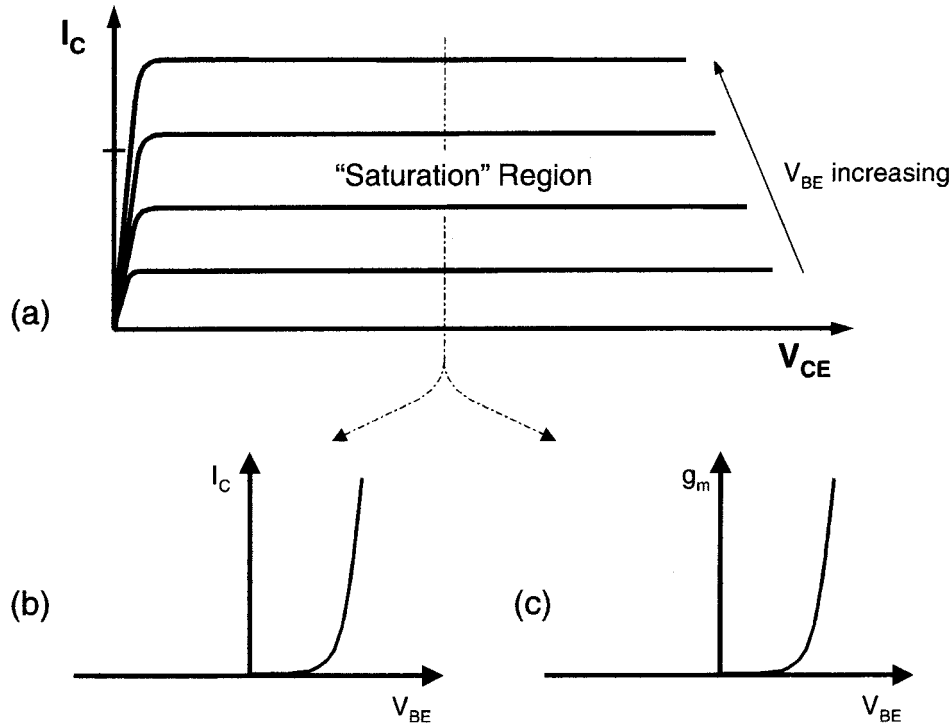


FIGURE 70.9 (a) Collector current (I_C) versus collector-to-emitter voltage (V_{CE}) characteristic curves with the base-to-emitter voltage (V_{BE}) as stepped parameter; (b) I_C versus V_{BE} "transfer curve" for a constant V_{CE} in saturated region of operation shows exponential behavior; and (c) transconductance g_m versus V_{BE} for a constant V_{CE} in the saturated region of operation corresponding to the transfer curve in (b).

in digital circuits. In general, higher g_m values allow a designer to use feedback to a greater extent in design and this provides for greater tolerance to process variations.

Comparing Parameters

Table 70.2 compares some of the more important features and parameters of the BJT/HBT device with the FET device. For reference, a common-source FET configuration is compared with a common-emitter BJT/HBT configuration. One of the most striking differences is the input impedance parameter. A FET has a high input impedance at low to mid-range frequencies because it essentially is a capacitor. As the frequency increases, the magnitude of the input impedance decreases as $1/\omega$ because a capacitive reactance varies as $|C_{gs}/\omega|$. The BJT/HBT emitter-base is a forward-biased pn junction, which is inherently a low impedance structure because of the lowered potential barrier to carriers. The BJT/HBT input is also capacitive (i.e., a large diffusion capacitance due to stored charge), but a large conductance (or small resistance) appears in parallel assuring a low input impedance even at low frequencies.

BJT/HBT devices are known for their higher transconductance g_m, which is proportional to collector current. An FET's g_m is proportional to the saturated velocity v_{sat} and its input capacitance C_{gs}. Thus, device structure and material parameters set the performance of the FET whereas thermodynamics play the key role in establishing the magnitude of g_m in a BJT/HBT.

Thermodynamics also establishes the magnitude of the *turn-on voltage* (this follows simply from Eq. 70.14) in the BJT/HBT device. For digital circuits, turn-on voltage (or *threshold voltage*) is important in terms of repeatability and consistency for circuit robustness. The BJT/HBT is clearly superior to the FET in this regard because doping concentration and physical structure establish an FET's turn-on voltage. In general, these variables are less controllable. However, the forward turn-on voltage in the AlGaAs/GaAs

TABLE 70.2 Comparing Electrical Parameters for BJT/HBT vs. FET

Parameter	BJT/HBT	FET
Input impedance Z	Low Z due to forward-biased junction; large diffusion capacitance C_{be}	High Z due to reverse biased junction or insulator; small depletion layer capacitance C_{gs}
Turn-on Voltage	Forward voltage V_{BE} highly repeatable; set by thermodynamics	Pinch-off voltage V_P not very repeatable; set by device design
Transconductance	High g_m [$= I_C/(kT/q)$]	Low g_m [$\cong v_{sat}C_{gs}$]
Current gain	β (or h_{FE}) = 50 to 150; β is important due to low input impedance	Not meaningful at low frequencies and falls as $1/\omega$ at high frequencies
Unity current gain cutoff frequency f_T	$f_T = g_m/2\pi C_{BE}$ is usually lower than for FETs	$f_T = g_m/2\pi C_{gs}$ ($= v_{sat}/2\pi L_g$) higher for FETs
Maximum frequency of oscillation f_{max}	$f_{max} = [f_T/(8\pi r_b C_{bc}]^{1/2}$	$f_{max} = f_T [r_{ds}/R_{in}]^{1/2}$
Feedback capacitance	C_{bc} large because of large collector junction	Usually C_{gd} is much smaller than C_{bc}
1/*f* Noise	Low in BJT/HBT	Very high 1/*f* noise corner frequency
Thermal behavior	Thermal runaway and second breakdown	No thermal runaway
Other		Backgating is problem in semi-insulating substrates

HBT is higher (~1.4 V) because of the band discontinuity at the emitter–base heterojunction. For InP-based HBTs, the forward turn-on voltage is lower (~0.8 V) than that of the AlGaAs/GaAs HBT and comparable to the approximate 0.7 V found in silicon BJTs. This is important in digital circuits because reducing the signal swing allows for faster circuit speed and lowers power dissipation by allowing for reduced power supply voltages.

For BJT/HBT devices, *current gain* (often given the symbol of β or h_{FE}) is a meaningful and important parameter. Good BJT devices inject little current into the emitter and, hence, operate with low base current levels. The current gain is defined as the collector current divided by the base current and is therefore a measure of the quality of the device (i.e., traps and defects, both surface and bulk, degrade the current gain due to higher recombination currents). At low to mid-range frequencies, current gain is not especially meaningful for the high input impedance FET device because of the capacitive input.

The intrinsic gain of an HBT is higher because of its higher *Early voltage* V_A. The Early voltage is a measure of the intrinsic output conductance of a device. In the HBT, the change in the collector voltage has very little effect on the modulation of the collector current. This is true because the band discontinuity dominates the establishment of the current collected at the collector–base junction. A figure of merit is the *intrinsic voltage gain* of an active device, given by the product $g_m V_A$, and the HBT has the highest values compared to silicon BJTs and compound semiconductor FETs.

It is important to have a dynamic figure of merit or parameter to assess the usefulness of an active device for high-speed operation. Both the unity current gain cutoff frequency f_T and maximum frequency of oscillation f_{max} have been discussed in the charge control section above. Both of these figures of merit are used because they are simple and can generally be correlated to circuit speed. The higher the value of both parameters, the better the high-speed circuit performance. This is not the whole story because in digital circuits other factors such as output node-to-substrate capacitance, external load capacitances, and interconnect resistance also play an important role in determining the speed of a circuit.

Generally, 1/*f* noise is much higher in FET devices than in the BJT/HBT devices. This is usually of more importance in analog applications and oscillators however. Thermal behavior in high-speed devices is important as designers push circuit performance. Bipolar devices are more susceptible to thermal runaway than FETs because of the positive feedback associated with a forward-biased junction (i.e., a smaller forward voltage is required to maintain the same current at higher temperatures). This is not true in the FET; in fact, FETs generally have negative feedback under common biases used in digital circuits. Both GaAs and InP have poorer thermal conductivity than silicon, with GaAs being about one-third of silicon and InP being about one-half of silicon.

Finally, circuits built on GaAs or InP semi-insulating substrates are susceptible to *backgating*. Backgating is similar to the backgate-bias effects in MOS transistors, only it is not as predictable or repeatable as the well-known *backgate-bias effect* is in silicon MOSFETs on silicon lightly doped substrates. Interconnect traces with negatively applied voltages and located adjacent to devices can change their threshold voltage (or turn-on voltage). It turns out that HBT devices do not suffer from backgating, and this is one of their advantages. Of course, semi-insulating substrates are nearly ideal for microstrip transmission lines on top of the substrates because of their very low loss. Silicon substrates are much more lossy in comparison and this is a decided advantage in GaAs and InP substrates.

70.4 Typical Device Structures

In this section, a few typical device structures are described. We begin with FET structures and then follow with HBT structures. There are many variants on these devices and the reader is referred to the literature for more information.[15,16,19-22]

FET Structures

In the silicon VLSI world, the MOSFET (*metal-oxide-semiconductor field-effect transistor*) dominates. This device forms a channel at the oxide–semiconductor interface upon applying a voltage to the gate to attract carriers to this interface.[23] The thin layer of mobile carriers forms a two-dimensional sheet of carriers. One of the limitations with the MOSFET is that the oxide–semiconductor interface scatters the carriers in the channel and degrades the performance of the MOSFET. This is evident in Fig. 69.1 where the lower electron velocity at the Si-SiO_2 interface is compared with electron velocities in compound semiconductors. For many years, device physicists have looked for device structures and materials which increase electron velocity. FET structures using compound semiconductors have led to much faster devices such as the MESFET and the HEMT.

The MESFET (*metal-semiconductor FET*) uses a thin doped channel (almost always n-type because electrons are much more mobile in semiconductors) with a reverse-biased Schottky barrier for the gate control.[15] The cross-section of a typical MESFET is shown in Fig. 70.10(a). A recessed gate is used along with a highly doped n^+ layer at the surface to reduce the series resistance at both the source and drain connections. The gate length and electron velocity in the channel dominate in determining the high-speed performance of a MESFET. Much work has gone into developing processes that form shorter gate structures. For digital devices, lower breakdown voltages are permissible, and therefore shorter gate lengths and higher channel doping is more compatible with such devices. For a given semiconductor material, a device's breakdown voltage BV_{GD} times its unity current gain cutoff frequency f_T is a constant. Therefore, it is possible to tradeoff BV_{GD} for f_T in device design. A high f_T is required in high-speed digital circuits because devices with a high f_T over their logic swing will have a high g_m/C ratio for large-signal operation. A high g_m/C ratio translates into a device's ability to drive load capacitances.

It is also desirable to maximize the charge in the channel per unit gate area. This allows for higher currents per unit gate width and greater ability to drive large capacitive loads. The higher current per unit gate width also favors greater IC layout density. In the MESFET, the doping level of the channel sets this limit. MESFET channels are usually ion-implanted and the added lattice damage further reduces the electron mobility.

To achieve still higher currents per gate width and even higher figures of merit (such as f_T and f_{max}), the HEMT (*high electron mobility transistor*) structure has evolved.[16,22] The HEMT is similar to the MESFET except that the doped channel is replaced with a *two-dimensional quantum well* containing electrons (sometimes referred to as a *2-D electron gas*). The quantum well is formed by a discontinuity in conduction band edges between two different semiconductors (such as AlGaAs and GaAs in Fig. 69.3). From Fig. 69.4 we see that GaAs and $Al_{0.3}Ga_{0.7}As$ have nearly identical lattice constants but with somewhat different bandgaps. One compound semiconductor can be grown (i.e., using *molecular beam epitaxy* or *metalo-organic chemical vapor deposition* techniques) on a different compound semiconductor if the

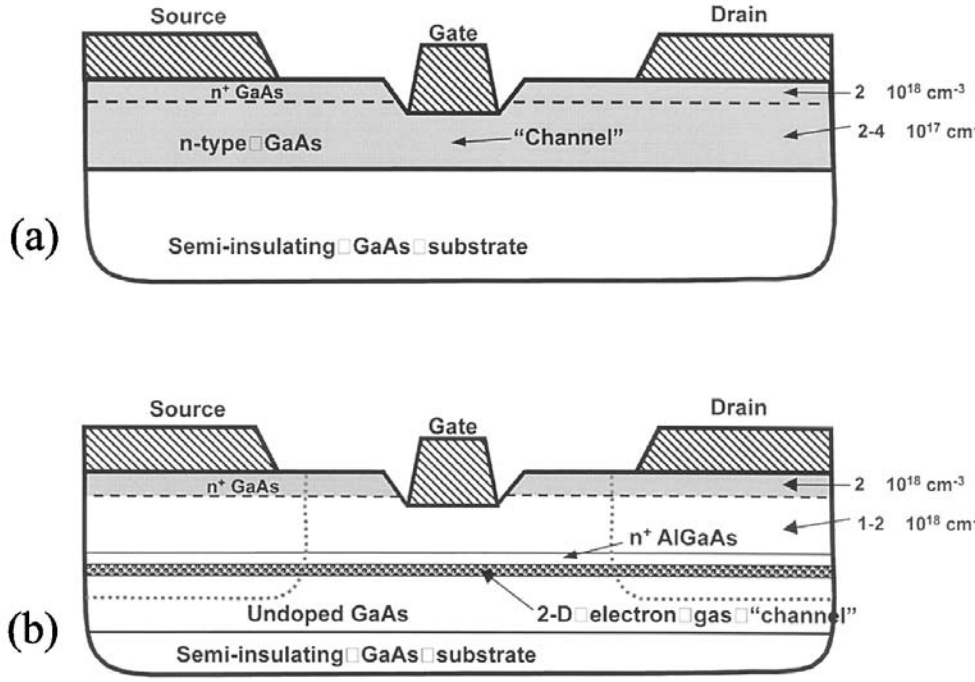


FIGURE 70.10 Typical FET cross-sections for (a) GaAs MESFET device with doped channel, and (b) AlGaAs/GaAs HEMT device with single quantum well containing and two-dimensional electron gas.

lattice constants are identical. Another example is $Ga_{0.47}In_{0.53}As$ and InP, where they are lattice matched. The difference in conduction band edge alignment leads to the formation of a quantum well. The greater the edge misalignment, the deeper the quantum well can be, and generally the greater the number of carriers the quantum well can hold. The charge per unit area that a quantum well can hold directly translates into greater current per unit gate width. Thus, the information in Fig. 69.4 can be used to *bandgap engineer* different materials that can be combined in lattice matched layers.

A major advantage of the quantum well comes from being able to use semiconductors that have higher electron velocity and mobility than the substrate material (e.g., GaAs) and also avoid charge impurity scattering in the quantum well by locating the donor atoms outside the quantum well itself. Figure 70.10(b) shows a HEMT cross-section where the dopant atoms are positioned in the wider bandgap AlGaAs layer. When these donors ionize, electrons spill into the quantum well because of its lower energy. Higher electron mobility is possible because the ionized donors are not located in the quantum well layer. A recessed gate is placed over the quantum well, usually on a semiconductor layer such as the AlGaAs layer in Fig. 70.10(b), allowing modulation of the charge in the quantum well.

There are only a few lattice-matched structures possible. However, semiconductor layers for which the lattice constants are not matched are possible if the layers are thin enough (of the order of a few nanometers). Molecular beam epitaxy and MOCVD make it possible to grow layers of a few atomic layers. Such structures are called *pseudomorphic HEMT* (PHEMT) devices.[16,22] This gives more flexibility in selecting quantum well layers which hold greater charge and have higher electron velocities and mobilities. The highest performance levels are achieved with pseudomorphic HEMT devices.

FET Performance

All currently used FET structures are n-channel because hole velocities are very low compared with electron velocities. Typical gate lengths range from 0.5 microns down to about 0.1 microns for the fastest devices. The most critical fabrication step in producing these structures is the gate recess width and depth.

The GaAs MESFET (ca. 1968) was the first compound semiconductor FET structure and is still used today because of its simplicity and low cost of manufacture. GaAs MESFET devices have f_T values in the 20 GHz to 50 GHz range corresponding to gate lengths of 0.5 microns down to 0.2 microns, and g_m values of the order of 200 to 400 mS/mm, respectively. These devices will typically have I_{DSS} values of 200 to 400 mA/mm, where parameter I_{DSS} is the common-source, drain current with zero gate voltage applied in a saturated state of operation.

In comparison, the first HEMT used an AlGaAs/GaAs material structure. These devices are higher performance than the GaAs MESFET (e.g., given an identical gate length, the AlGaAs/GaAs HEMT has an f_T about 50% to 100% higher, depending on the details of the device structure and quality of material). Correspondingly higher currents are achieved in the AlGaAs/GaAs HEMT devices.

Higher performance still is achieved using InP based HEMTs. For example, the $In_{0.53}Ga_{0.47}As/In_{0.52}Al_{0.48}As$ on InP lattice-matched HEMT have reported f_T numbers greater than 250 GHz with gate lengths of the order of 0.1 microns. Furthermore, such devices have I_{DSS} values approaching 1000 mA/mm and very high transconductances of greater than 1500 mS/mm.[22,24] These devices do have low breakdown voltages of the order of 1 or 2 V because of the small bandgap of InGaAs. Changing the stoichiometric ratios to $In_{0.15}Ga_{0.85}As/In_{0.70}Al_{0.30}As$ on a GaAs substrate produces a pseudomorphic HEMT structure. The $In_{0.15}Ga_{0.85}As$ is a strained layer when grown on GaAs. The use of strained layers gives the device designer more flexibility in accessing a wider variety of quantum wells depths and electronic properties.

Heterojunction Bipolar Structures

Practical heterojunction bipolar transistors (HBT) devices[19,21] are still evolving. Molecular beam epitaxy (MBE) is used to grow the doped layers making up the vertical semiconductor structure in the HBT. In fact, HBT structures were not really practical until the advent of MBE, although the idea behind the HBT goes back to around 1950 (Shockley). The vastly superior compositional control and layer thickness control with MBE is what made HEMTs and HBTs possible. The first HBT devices used an AlGaAs/GaAs junction with the wider bandgap AlGaAs layer forming the emitter region. Compound semiconductor HBT devices are typically mesa structures, as opposed to the more nearly planar structures used in silicon bipolar technology, because top surface contacts must be made to the collector, base, and emitter regions. Molecular beam epitaxy grows the stack of layers over the entire wafer, whereas, in silicon VLSI processes, selective implantations and oxide masking localize the doped regions. Hence, etching down to the respective layers allows for contact to the base and collector regions. An example of such a mesa HBT structure[20] is shown in Fig. 70.11. The HBT shown uses an InGaP emitter primarily for improved reliability over the AlGaAs emitter and a carbon-doped p^+ base GaAs layer.

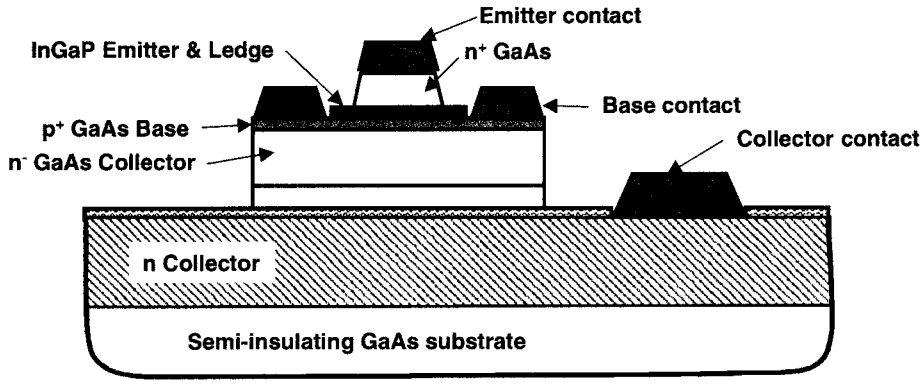


FIGURE 70.11 Cross-section of an HBT device with carbon-doped p^+ base and an InGaP emitter.[20] Note the commonly used mesa structure, where selective layer etching is required to form contacts to the base and collector regions.

Recently, InP-based HBTs[21] have emerged as candidates for use in high-speed circuits. The two dominant heterojunctions are InP/InGaAs and AlInAs/InGaAs in InP devices. The small but significant bandgap difference between AlInAs directly on InP greatly limits its usefulness. InP-based HBT device structures are similar to those of GaAs-based devices and the reader is referred to Chapter 5 of Jalali and Pearton[16] for specific InP HBT devices. Generally, InP has advantages of lower surface recombination (higher current gain results), better electron transport, lower forward turn-on voltage, and higher substrate thermal conductivity.

HBT Performance

Typical current gain values in production-worthy HBT devices range from 50 at the low range to 150 at the high range. Cutoff frequency f_T values are usually quoted under the best (i.e., peak) bias conditions. For this reason f_T values must be carefully interpreted because in digital circuits, the bias state varies widely over the entire switching swing. For this reason, probably an averaged f_T value would be better, but it is difficult to determine. Typical f_T values for HBT processes in manufacturing (say 1998) are in the 50 to 150 GHz range. For example, for the HBT example in Fig. 70.11 with a 2 μm × 2 μm emitter f_T is approximately 65 GHz at a current density of 0.6 mA/μm^2 and its dc current gain is around 50. Of course, higher values for f_T have been reported for R&D or laboratory devices. In HBT devices, the parameter f_{max} is often lower than its f_T value (e.g., for the device in Fig. 70.11, f_{max} is about 75 GHz). Base resistance (refer to Table 70.2 for equation) is the dominant limiting factor in setting f_{max}. The best HBT devices have f_{max} values only slightly higher than their f_T values. In comparison, MESFET and HEMT devices typically have higher f_{max}/f_T ratios, although in digital circuits this may be of little importance.

Where the HBT really excels is in being able to generate much higher values of transconductance. This is a clear advantage in driving larger loading capacitances found in large integrated circuits. Biasing the HBT in the current range corresponding to the highest transconductance is essential to take advantage of the intrinsically higher transconductance.

Long, S.I. "Logic Design Principles and Examples"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

71

Logic Design Principles and Examples

Stephen I. Long
University of California at Santa Barbara

71.1 Introduction

The logic circuits used in high-speed compound semiconductor digital ICs must satisfy the same essential conditions for design robustness and performance as digital ICs fabricated in other technologies. The static or dc design of a logic cell must guarantee adequate voltage and/or current gain to restore the signal levels in a chain of similar cells. A minimum *noise margin* must be provided for tolerance against process variation, temperature, and induced noise from ground bounce, crosstalk, and EMI so that functional circuits and systems are produced with good electrical yield. Propagation delays must be determined as a function of loading and power dissipation.

Compound semiconductor designs emphasize speed, so logic voltage swings are generally low, τ_r low so that transconductances and f_T are high, and device access resistances are made as low as possible in order to minimize the lifetime time constant τ. This combination makes circuit performance very sensitive to parasitic R, L, and C, especially when the highest operation frequency is desired. The following sections will describe the techniques that can be used for static and dynamic design of high-speed logic.

71.2 Static Logic Design

A basic requirement for any logic family is that it must be capable of *restoring* the logic voltage or current swing. This means that the voltage or current gain with loading must exceed 1 over part of the transfer characteristic. Figure 71.1 shows a typical V_{out} vs. V_{in} *dc transfer characteristic* for a static ratioed logic inverter as is shown in the schematic diagram of Fig. 71.2. It can be seen that a chain of such inverters will restore steady-state logic voltage levels to V_{OL} and V_{OH} because the high-gain transition region around $V_{in} = V_{TH}$ will result in voltage amplification. Even if the voltage swing is very small, if centered on the *inverter threshold* voltage V_{TH}, defined as the intersection between the transfer characteristic and the $V_{in} = V_{out}$ line, the voltage will be amplified to the full swing again by each successive stage.

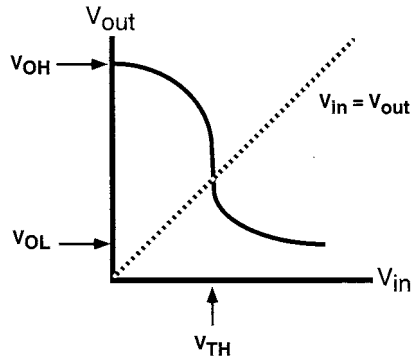


FIGURE 71.1 Typical voltage transfer characteristic for the logic inverter shown in Fig. 71.2.

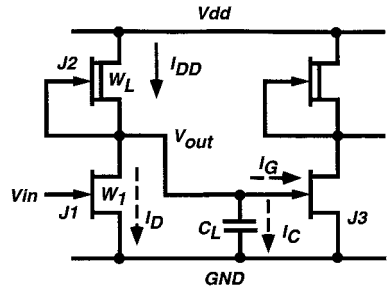


FIGURE 71.2 Schematic diagram of a direct-coupled FET logic (DCFL) inverter.

Ratioed logic implies that the *logic high and low voltages* V_{OH} and V_{OL} shown in Fig. 71.1 are a function of the widths W_1 and W_L of the FETs in the circuit shown in Fig. 71.2. In III-V technologies, this circuit is implemented with either MESFETs or HEMTs. The circuit in Fig. 71.2 is called *Direct Coupled FET Logic* or DCFL.

The logic levels of *non-ratioed* logic are independent of device widths. Non-ratioed logic typically occurs when the switching transistors do not conduct any static current. This is typical of logic families such as static CMOS or its GaAs equivalent *CGaAs*[2] which make use of complementary devices. Dynamic logic circuits such as precharged logic[25] and pass transistor logic[26,27] also do not require static current in pull-down chains. Such circuits have been used with GaAs FETs in order to reduce static power dissipation. They have not been used, however, for the highest speed applications.

Direct-Coupled FET Logic

DCFL is the most widely used logic family for the high-complexity, low-power applications that will be discussed in Chapter 72. The operation of DCFL shown in Fig. 71.2 is easily explained using a load line analysis. Currents are indicated by arrows in this figure. Solid arrows correspond to currents that are nearly constant. Dashed arrows represent currents that depend on the state of the output of the inverter. Figure 71.3 presents an I_D–V_{DS} characteristic of the enhancement mode (normally-off with threshold voltage $V_T > 0$) transistor J1. A family of characteristic curves is drawn representing several V_{GS} values. In this circuit, $V_{GS} = V_{in}$ and $V_{DS} = V_{out}$.

A load line representing the I–V characteristic of the active load ($V_{GS} = 0$) depletion-mode (normally-on with $V_T < 0$) transistor J2 is also superimposed on this drawing. Note that the logic low level V_{OL} is determined by the intersection of the two curves when $I_D = I_{DD}$. Load line 1 corresponds to a load device with narrow width; load line 2 for a wider device. It is evident that the narrow, weaker device will provide a lower V_{OL} value and thus will increase the logic swing. However, the weaker device will also have less

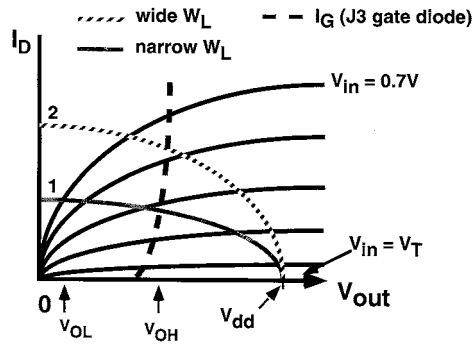


FIGURE 71.3 Drain current versus drain-source voltage characteristic of J1. The active load, J2, is also shown superimposed over the J1 characteristics as a load line. Two load lines corresponding to wide and narrow J2 widths are shown. In addition, the gate current I_G of J3 versus V_{out} limits the logic high voltage.

current available to drive any load capacitance, so the inverter with load line 1 will therefore be slower than the one with load line 2. There is therefore a tradeoff between speed and logic swing. So far, the analysis of this circuit is the same as that of an analogous nMOS E/D inverter.

In the case of DCFL logic inverters implemented with GaAs-based FETs, the Schottky barrier gate electrode of the next stage will limit the maximum value of V_{OH} to the forward voltage drop across the gate-source diode. This is shown by the gate diode I_G–V_{GS} characteristic also superimposed on Fig. 71.3. V_{OH} is given by the point of intersection between the load current I_{DD} and the gate current I_G, because a logic high output requires that the switch transistor J1 is off. V_{OH} will therefore also depend on the load transistor current. Effort must be made not to overdrive the gate since excess gate current will flow through the internal parasitic source resistance of the driven device J3, degrading V_{OL} of this next stage.

Source-Coupled FET Logic

A second widely used type of logic circuit — source-coupled FET Logic or SCFL is shown in Fig. 71.4. SCFL, or its bipolar counterpart, ECL, is widely used for very high-speed applications, which will be discussed in Chapter 72. The core of the circuit consists of a differential amplifier, J_1 and J_2, a current source J_3, and pull-up resistors R_L on the drains. The differential topology is beneficial for rejection of common-mode noise. The static design procedure can be illustrated again by a load-line analysis. A maximum current I_{CS} can flow through either J_1 or J_2.

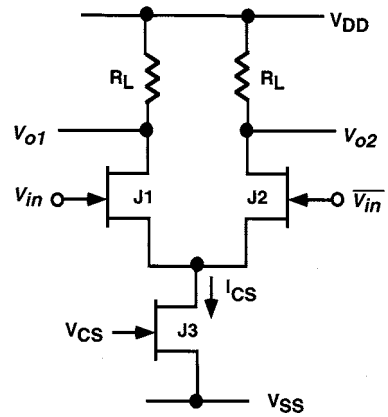


FIGURE 71.4 Schematic of differential pair J1,J2 used as a source-coupled FET logic (SCFL) cell.

Figure 71.5 shows the I_D–V_{DS} characteristic of J_1 for example. The maximum current I_{CS} is shown by a dotted line. The output voltage V_{O1} is either V_{DD} or $V_{DD} - I_{CS}R_L$; therefore, the maximum differential voltage swing, $\Delta V = 2\ I_{CS}R_L$, is determined by the choice of R_L. Next, the width of J_1 should be selected so that the change in V_{GS} needed to produce the voltage drop $I_{CS}R_L$ at the drain is less than $I_{CS}R_L$. This will ensure that the voltage gain is greater than 1 (needed to compensate for the source followers described below) and that the device is biased in its saturation region or cutoff at all times. The latter requirement is necessary if the maximum speed is to be obtained from the SCFL stage, since device capacitances are minimized in saturation and cutoff.

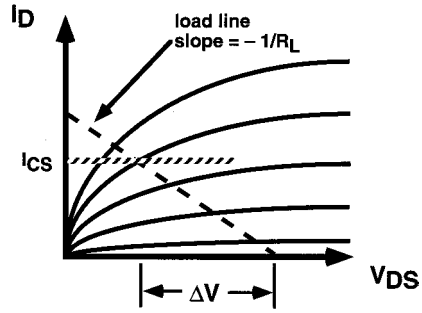


FIGURE 71.5 Load-line analysis of the SCFL inverter cell.

Source followers are frequently used on the output of an SCFL stage or at the inputs of the next stage. Figure 71.6(a) shows the schematic diagram of the follower circuit. The follower can serve two functions: level shifting and buffering of capacitive loads. When used as a level shifter, a negative or positive voltage offset can be obtained between input and output. The only requirement is that the V_{GS} of the source follower must be larger than the FET threshold voltage. If the source follower is at the output of an SCFL cell, it can be used as a buffer to reduce the sensitivity of delay to load capacitance or fanout.

The voltage gain of a source follower is always less than 1. This can be illustrated by another load-line analysis. Figure 71.6(b) presents the I_{D1}–V_{DS1} characteristic of the source follower FET, J1. A constant V_{GS1} is applied for every curve plotted in the figure. The load line (dashed line) of a depletion-mode, active current source J2 is also superimposed. In this circuit, the output voltage is $V_{out} = V_{DD} - V_{DS1}$. The V_{out} is determined by the intersection of the load line with the I_{D1} characteristic curves. The current of the pull-down current source is selected according to the amount of level shifting needed. A high current will result in a greater amount of level shift than a small load current. If the devices have high output resistance, and are accordingly very flat, very little change in V_{GS1} will be required to change V_{out} over the

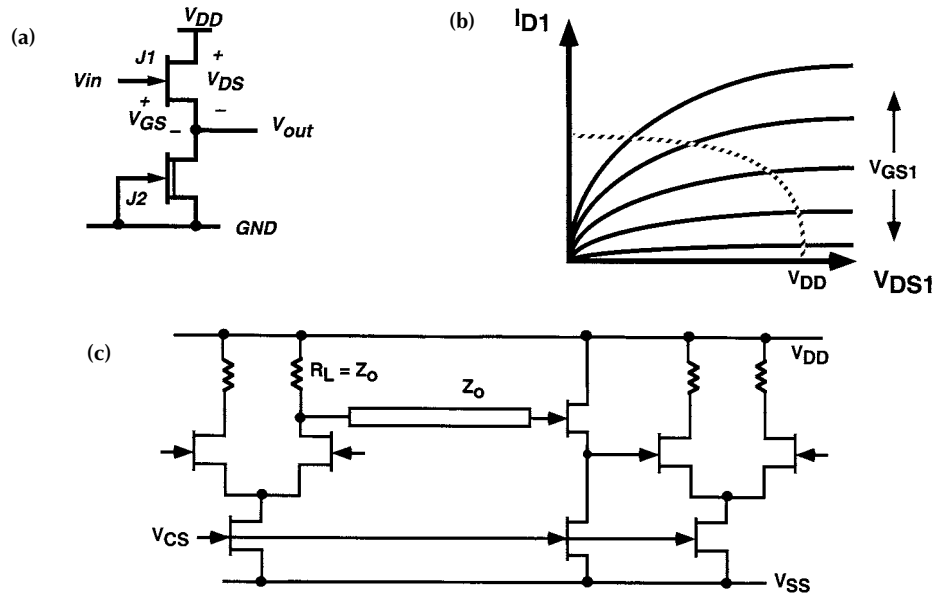


FIGURE 71.6 (a) Schematic of source follower, (b) load-line analysis of source follower, and (c) source follower buffer between SCFL stages.

full range from V_{OL} to V_{DD}. If V_{GS1} remains nearly constant, then V_{out} follows V_{in}, hence the name of the circuit. Since the input voltage to the source follower stage is $V_{in} = V_{GS1} + V_{out}$, a small change in V_{GS1} would produce an incremental voltage gain close to unity. If the output resistance is low, then the characteristic curves will slope upward and a larger range of V_{GS1} will be necessary to traverse the output voltage range. This condition would produce a low voltage gain. Small signal analysis shows that

$$A_v = \frac{1}{1 + 1 \Big/ \left[g_{m1} \left(r_{ds1} \| r_{ds2} \right) \right]} \tag{71.1}$$

The buffering effect of the source follower is accomplished by reducing the capacitive loading on the drain nodes of the differential amplifier because the input impedance of a source follower is high. Since the output tries to follow the input, the change in V_{GS} will be less than that required by a common source stage. Therefore, the input capacitance is dominated by C_{GD}, typically quite small for compound semiconductor FETs biased in saturation. The effective small-signal input capacitance is

$$C_G = C_{GD} + C_{GS}\left(1 - A_v\right) \tag{71.2}$$

where $A_v = dV_{out}/dV_{in}$ is the incremental voltage gain.

The source follower also provides a low output impedance, whose real part is approximately $1/g_m$ at low frequency. The current available to charge and discharge the load capacitance can be adjusted by the width ratio of J1 and J2. If the load is capacitive, V_{out} will be delayed from V_{in}. This will cause V_{GS1} to temporarily increase, providing excess current I_{D1} to charge the load capacitance. Ideally, for equal rise and fall times, the peak current available from J1 should equal the steady-state current of J2.

Source followers can also be used at the input of an SCFL stage to provide level shifting as shown in Fig. 71.6(c). In this case, the drain resistors, R_L, should be chosen to provide the proper termination resistance for the on-chip interconnect transmission line. These resistors provide a reverse termination to absorb signals that reflect from the high input impedance of the source follower. Alternatively, the drain resistors can be located at the gate of the source follower, thereby providing a shunt termination at the destination end of the interconnect. This practice results in good signal integrity, but because the practical values of characteristic impedance are less than 100-Ω, the current swing in the differential amplifier core must be large. This will increase power dissipation per stage.

SCFL logic structures generally employ more than one level of differential pairs so that complex logic functions (XOR, latch, and flip-flop) that require multiple gates to implement in logic families such as DCFL can be implemented in one stage. More details on SCFL gate structures and examples of their usage will be given in Chapter 72.

Static and Dynamic Noise Margin and Noise Sources

Noise margin is a measure of the ability of a logic circuit to provide proper functionality in the presence of noise.[28] There are many different definitions of noise margin, but a simple and intuitive one for the static or dc noise margin is illustrated by Fig. 71.8. Here, the transfer characteristic from Fig. 71.1 is plotted again. In order to evaluate the ability of a chain of such inverters to reject noise, a loop consisting of two identical inverters is considered. This might be representative of the positive feedback core of a bistable latch. Because the inverters are connected in a loop, an infinite chain of inverters is represented. The transfer characteristic of inverter 2 in Fig. 71.7 is plotted in gray lines. For inverter 2, $V_{out\,1} = V_{in2}$ and $V_{out2} = V_{in1}$. Therefore, the axes are reversed for the characteristic plotted for inverter 2. If a series noise source V_N were placed within the loop as shown, the maximum static noise voltage allowed will be represented by the *maximum width* of the loops formed by the transfer characteristic.[28] These widths, labeled V_{NL} and V_{NH}, respectively, for the low and high noise margins, are shown in the figure. If the voltage V_N exceeds V_{NL} or V_{NH}, the latch will be set into the opposite state and will remain there until

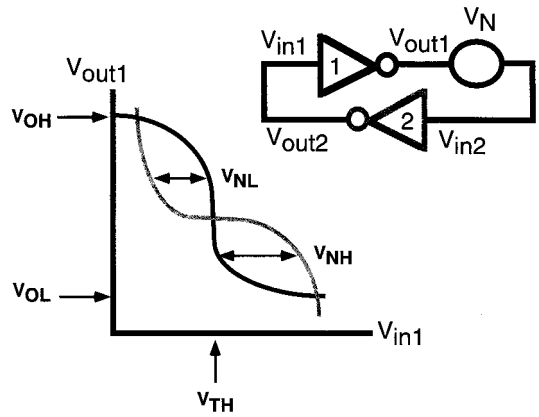


FIGURE 71.7 Voltage transfer characteristics of an inverter pair connected in a loop. Noise margins are shown as V_{NH} and V_{NL}.

reset. This would constitute a logic failure. Therefore, we must insist that any viable logic circuit provide noise margins well in excess of ambient noise levels in the circuit.

The static noise margin defined above utilized a dc voltage source V_N in series with logic inverters to represent static noise. This source might represent a static offset voltage caused by IR drop along IC power and ground distribution networks. The DCFL inverter, for example, would experience a shift in V_{TH} that is directly proportional to a ground voltage offset. This shift would skew the noise margins. The smallest noise margin would determine the circuit electrical yield. The layout of the power and ground distribution networks must consider this problem. The width of power and ground buses on-chip must be sufficient to guarantee a maximum IR drop that does not compromise circuit operation. It is important to note that this width is frequently much greater than what might be required by electromigration limits. It is essential that the designer consider IR drop in the layout. Some digital IC processes allow the topmost metal layer to form a continuous sheet, thereby minimizing voltage drops.

The static noise voltage source V_N might also represent static threshold voltage shifts on the active devices due to statistical process variation or backgating effects. Therefore, the noise margin must be several times greater than the variance in device threshold voltages provided by the fabrication process so that electrical yields will not be compromised.[29]

The above definition of maximum width noise margin has assumed a steady-state condition. It does not account for transient noise sources and the delayed response of the logic circuit to noise pulses. Unfortunately, pulses of noise are quite common in digital systems. For example, the ground potential can often be modified dynamically by simultaneous switching events on the IC chip.[30] Any ground distribution bus can be modeled as a transmission line with impedance Z_0 where

$$Z_0 = \sqrt{\frac{L_o}{C_o}} \tag{71.3}$$

Here, L_o is the equivalent series inductance per unit length and C_o the equivalent shunt capacitance per unit length. Since the interconnect exhibits a series inductance, there will be transient voltage noise ΔV induced on the line by current transients as predicted by

$$\Delta V = L\frac{dI}{dt} \tag{71.4}$$

This form of noise is often called *ground bounce*. The ground bounce ΔV is particularly severe when many devices are being switched synchronously, as would be the case in many applications involving

flip-flops in shift registers or pipelined architectures. The high peak currents that result in such situations can generate large voltage spikes. For example, output drivers are well-known sources of noise pulses on power and ground buses unless they are carefully balanced with fully differential interconnections and are powered by power and ground pins separate from the central logic core of the IC.

Designing to minimize ground bounce requires minimization of inductance. Bakoglu[30] provides a good discussion of power distribution noise in high-speed circuits. There are several steps often used to reduce switching noise. First, it is standard practice to make extensive use of multiple ground pins on the chip to reduce bond-wire inductance and package trace inductance when conventional packaging is used. Bypass capacitance off-chip can be useful if it can be located inside the package and can have a high series resonant frequency. On-chip bypass capacitance is also helpful, especially if enough charge can be supplied from the capacitance to provide the current during clock transitions. The objective is to provide a low impedance between power and ground on-chip at the clock frequency and at odd multiples of the clock frequency. Finally, as mentioned above, high-current circuits such as clock drivers and output drivers should not share power and ground pins with other logic on-chip.

Crosstalk is another common source of noise pulses caused by electromagnetic coupling between adjacent interconnect lines. A signal propagating on a driven interconnect line can induce a crosstalk voltage and current in a coupled line. The duration of the pulse depends on the length of interconnect; the amplitude depends on the mutual inductance and capacitance between the coupled lines.[31]

In order to determine how much noise is acceptable in a logic circuit, the noise margin definition can be modified to accommodate the transient noise pulse situation. The logic circuit does not respond instantaneously to the noise pulse at the input. This delay in the response is attributed to the device and interconnect capacitances and the device current limitations which will be discussed extensively in Section 71.3. Consider the device input capacitance. Sufficient charge must be transferred during the noise pulse to the input capacitance to shift the control voltage either above or below threshold. In addition, this voltage must be maintained long enough for the output to respond if a logic upset is to occur. Therefore, a logic circuit can withstand a larger noise pulse amplitude than would be predicted from the static noise margin if the pulse width is much less than the propagation delay of the circuit. This increased noise margin for short pulses is called the dynamic noise margin (DNM). The DNM approaches the static NM if the pulse width is wide compared with the propagation delay because the circuit can charge up to the full voltage if not constrained by time.

The DNM can be predicted by simulation. Figure 71.8(a) shows the loop connection of the set-reset NOR latch similar to that which was used for the static NM definition in Fig.10.7. The inverter has been modified to become a NOR gate in this case. An input pulse train $V_1(t)$ of fixed duration but with

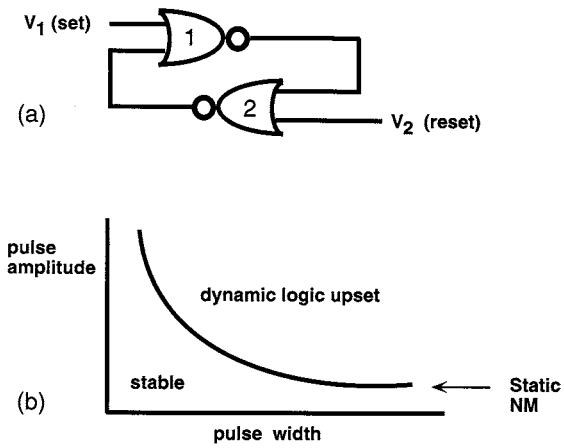


FIGURE 71.8 (a) Set-reset latch used to describe dynamic noise margin simulation, and (b) plot of the pulse amplitude applied to the set input in (a) that results in a logic upset.

gradually increasing amplitude can be applied to the set input. The latch was initialized by applying an initial condition to the reset input $V_2(t)$. The output response is observed for the input pulse train. At some input amplitude level, the output will be set into the opposite state. The latch will hold this state until it is reset again. The cross-coupled NOR latch thus becomes a logic upset detector, dynamically determining the maximum noise margin for a particular pulse width. The simulation can be repeated for other pulse widths, and a plot of the pulse amplitude that causes the latch to set for each pulse duration can be constructed, as shown in Fig. 71.8(b). Here, any amplitude or duration that falls on or above the curve will lead to a logic upset condition.

Power Dissipation

Power dissipation of a static logic circuit consists of a static and a dynamic component as shown below.

$$P_D = V_{DD}\bar{I}_{DD} + C_L \Delta V^2 \; f\eta \tag{71.5}$$

In the case of DCFL, the current I_{DD} from the pull-up transistor J2 is relatively constant, flowing either in the pull-down (switch) device J1 or in the gate(s) of the subsequent stage(s). Taking its average value, the static power is $V_{DD}\overline{I_{DD}}$. The dynamic power $C_L\Delta V^2 f\eta$ depends on the frequency of operation f, the load capacitance C_L, and the duty factor η. η is application dependent. Since the voltage swing is rather small for the DCFL inverter under consideration (about 0.6 V for MESFETs), the dynamic power will not be significant unless the load capacitance is very large, such as in the case of clock distribution networks. The V_{DD} power supply voltage is traditionally 2 V because of compatibility with the bipolar ECL V_{TT} supply, but a V_{DD} as low as 1 V can be used for special low-power applications. Typical power dissipation per logic cell (inverter, NOR) depends on the choice of supply voltage and on I_{DD}. Power is typically determined based on speed and is usually in the range of 0.1 to 0.5 mW/gate. DCFL logic circuits are often used when the application requires high circuit density and very low power.

71.3 Transient Analysis and Design for Very-High-Speed Logic

Adequate attention must be given to static or dc design, as described in the previous section, in order to guarantee functionality under the worst-case situations. In addition, since the only reason to use the compound semiconductor devices for digital electronics at all is their speed, attention must be given to the dynamic performance as well. In this section, we will describe three methods for estimating the performance of high-speed digital logic circuit functional blocks. Each of these methods has its strengths and weaknesses.

The most effective methods for guiding the design are those that provide insight that helps to identify the dominant time constants that determine circuit performance. These are not necessarily the most accurate methods, but are highly useful because they allow the designer to determine what part of the circuit or device is limiting the speed. Circuit simulators are far more accurate (at least to the extent that the device models are valid), but do not provide much insight into performance limitations. Without simple analytical techniques to guide the design, performance optimization becomes a trial-and-error exercise.

Zero-Order Delay Estimate

The first technique, which uses the simple relationship between voltage and current in a capacitor,

$$I = C_L \frac{dV}{dt} \tag{71.6}$$

is relevant when circuit performance is dominated by wiring or fan-out capacitance. This will be the case if the delay predicted by Eq. 71.6 due to the total loading capacitance, C_L, significantly exceeds the intrinsic delay of a basic inverter or logic gate. To apply this approach, determine the average current available from the driving logic circuit for charging (I_{LH}) and discharging (I_{HL}) the load capacitance. The logic swing ΔV is known, so low-to-high (t_{PLH}) and high-to-low (t_{PHL}) propagation delays can be determined from Eq. 71.6. These delays represent the time required to charge or discharge the circuit output to 50% of its final value. Thus, t_{PLH} is given by

$$t_{PLH} = \frac{C_L \Delta V}{2 I_{LH}} \tag{71.7}$$

where I_{LH} is the average charging current during the output transition from V_{OL} to $V_{OL} + \Delta V/2$. The net propagation delay is given by

$$t_P = \frac{t_{PLH} + t_{PHL}}{2} \tag{71.8}$$

At this limit, where speed is dominated by the ability to drive load capacitance, we see that increasing the currents will reduce t_p. In fact, the product of power (proportional to current) and delay (inversely proportional to current) is nearly constant under this situation. Increases in power lead to reduction of delay until the interconnect distributed RC delays or electromagnetic propagation delays become comparable to t_p.

The equation also shows that small voltage swing ΔV is good for speed if the noise margin and drive current are not compromised. This means that the devices must provide high transconductance.

For example, the DCFL inverter of Fig. 71.2 can be analyzed. Figure 71.9 shows equivalent circuits that represent the low-to-high and high-to-low transitions. The current available for the low-to-high transition, I_{PLH}, shown in Fig. 71.9(a), is equal to the average pullup current, I_{DD}. If we assume that $V_{OL} = 0.1$ V and $V_{OH} = 0.7$ V, then the ΔV of interest is 0.6 V. This brings the output up to 0.4 V at $V_{50\%}$. In this range of V_{out}, the active load transistor J2 is in saturation at all times for $V_{DD} > 1$ V, so I_{DD} will be relatively constant, and all of the current will be available to charge the capacitor.

The high-to-low transition is more difficult to model in this case. V_{out} will begin at 0.7 V and discharge to 0.4 V. The discharge current through the drain of J1 is going to vary with time because the device is below saturation over this range of V_{out}. Looking at the Vin = 0.7 V characteristic curve in Fig. 71.3, we see that its I_D–V_{DS} characteristic is resistive. Let's approximate the slope by $1/R_{on}$. Also, the discharge current I_{PHL} is the difference between I_{DD} and $I_{D1} = V_{out}/R_{on}$, as shown in Fig. 71.9(b). The average current available to discharge the capacitor can be estimated by

$$I_{HL} = \frac{V_{OH} + V_{50\%}}{2 R_{on}} - I_{DD} \tag{71.9}$$

Then, t_{PHL} is estimated by

$$t_{PHL} = \frac{C_L \Delta V}{2 I_{HL}} \tag{71.10}$$

Time Constant Delay Methods: Elmore Delay and Risetime

Time constant delay estimation methods are very useful when the wiring capacitance is quite small or the charging current is quite high. In this situation, typical of very-high-speed SSI and MSI circuits that

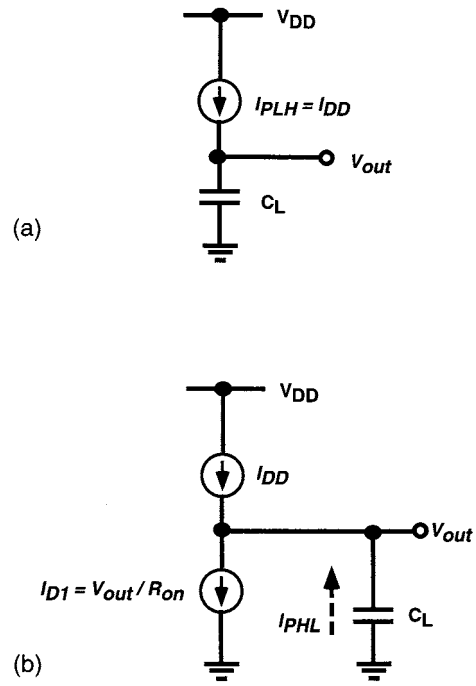


FIGURE 71.9 (a) Equivalent circuit for low-to-high transition; and (b) Equivalent circuit for high-to-low transition.

push the limits of the device and process technology, the circuit delays are dominated by the devices themselves. Both methods to be described rely on a large-signal equivalent circuit model of the transistors, an approximation dubious at best. But, the objective of these techniques is not absolute accuracy. That is much less important than being able to identify the dominant contributors to the delay and risetime, since more accurate but less intuitive solutions are easily available through circuit simulation. The construction of the large signal equivalent circuit requires averaging of non-linear model elements such as transconductance and certain device capacitances over the appropriate voltage swing.

The propagation delay definition described above, the delay required to reach 50% of the logic swing, must be relaxed slightly to apply methods based on linear system analysis. It was first shown by Elmore in 1948[32] and apparently rediscovered by Ashar in 1964[33] that the delay time t_D between an impulse function $\delta(0)$ applied at $t = 0$ to the input of a network and the centroid or "center-of-mass" of the impulse response (output) is quite close to the 50% delay. This definition of delay t_D is illustrated in Fig. 71.10. Two conditions must be satisfied in order to use this approach. First, the step response of the network is monotonic. This implies that the impulse response is purely a positive function. Monotonic step response is valid only when the circuit poles are all negative and real, or the circuit is heavily damped. Due to feedback through device capacitances, this condition is seldom completely correct. Complex poles often exist. However, strongly underdamped circuits are seldom useful for reliable logic circuits because their transient response will exhibit ringing, so efforts to compensate or damp such oscillations are needed in these cases anyway. Then, the circuit becomes heavily damped or at least dominated by a single pole and fits the above requirement more precisely.

Second, the correspondence between t_D and t_{PLH} is improved if the impulse response is symmetric in shape, as in Fig. 71.10(b). It is shown in Ref. 33 that cascaded stages with similar time constants have a tendency to approach a Gaussian-shaped distribution as the number of stages becomes large. Most logic systems require several cascaded stages, so this condition is often true as well.

Assuming that these conditions are approximately satisfied, we can make use of the fact that the impulse response of a circuit in the frequency domain is given by its transfer (or network) function F(s) in the complex frequency $s = \sigma + j\omega$. Then, the propagation delay, t_D, can be determined by

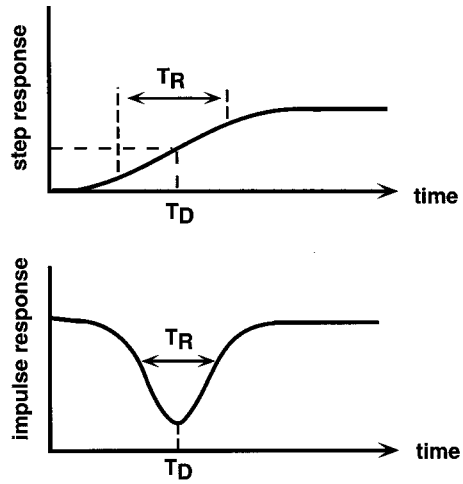


FIGURE 71.10 (a) Monotonic step response of a network; (b) corresponding impulse response. The delay t_D is defined as the centroid of the impulse response.

$$t_D = \frac{\int_0^\infty tf(t)dt}{\int_0^\infty f(t)dt} = \lim_{s\to 0}\frac{\int_0^\infty tf(t)e^{-st}dt}{\int_0^\infty f(t)e^{-st}dt} = \left[\frac{-\frac{d}{ds}F(s)}{F(s)}\right]_{s=0} \tag{71.11}$$

Fortunately, the integration never needs to be performed. t_D can be obtained directly from the network function F(s) as shown. But, the network function must be calculated from the large-signal equivalent circuit of the device, including all important parasitics, driving impedances, and load impedances. This is notoriously difficult if the circuit includes a large number of capacitances or inductances.

Fortunately, in most cases, circuits of interest can be subdivided into smaller networks, cascaded, and the presumed linearity of the circuits can be employed to simplify the task. In addition, the evaluation of the function at s = 0 eliminates many terms in the equations that result. In particular, Tien[34] shows that two corollaries are particularly useful in cascading circuit blocks:

1. If the network function F(s) = A(s)/B(s), then

$$t_D = \left[\frac{-\frac{d}{ds}A(s)}{A(s)}\right]_{s=0} + \left[\frac{\frac{d}{ds}B(s)}{B(s)}\right]_{s=0} \tag{71.12}$$

2. If F(s) = A(s)B(s)C(s), then

$$t_D = \left[\frac{-\frac{d}{ds}A(s)}{A(s)}\right]_{s=0} + \left[\frac{-\frac{d}{ds}B(s)}{B(s)}\right]_{s=0} + \left[\frac{-\frac{d}{ds}C(s)}{C(s)}\right]_{s=0} \tag{10.13}$$

This shows that the total delay is just the sum of the individual delays of each circuit block. When computing the network functions, care must be taken to include the driving point impedance of the

previous stage and to represent the previous stage as a Thevenin-equivalent open-circuit voltage source. A good description and illustration of the use of this approach in the analysis of bipolar ECL and CML circuits can be found in Ref. 34.

Risetime: the standard definition of risetime is the 10 to 90% time delay of the step response of a network. While convenient for measurement, this definition is analytically unpleasant to derive for anything except simple, first-order circuits. Elmore demonstrated that the standard deviation of the impulse response could be used to estimate the risetime of a network.[32] This definition provides estimates that are close to the standard definition. The standard deviation of the impulse response can be calculated using

$$T_R^2 = 2\pi\left[\int_0^\infty t^2 f(t)dt - t_D^2\right] \tag{71.14}$$

Since the impulse response frequently resembles the Gaussian function, the integral is easily evaluated.

Once again, the integration need not be performed. Lee[35] has pointed out that the transform techniques can also be used to obtain the Elmore risetime directly from the network function F(s).

$$T_R^2 = 2\pi\left[\frac{\frac{d^2}{ds^2}F(s)}{F(s)}\right]_{s=0} - 2\pi\left[\frac{\frac{d}{ds}F(s)}{F(s)}\right]^2_{s=0} \tag{71.15}$$

This result can also be used to show that the risetimes of cascaded networks add as the square of the individual risetimes. If two networks are characterized by risetimes T_{R1} and T_{R2}, the total risetime $T_{R,total}$ is given by the RMS sum of the individual risetimes

$$T_{R,total}^2 = T_{R1}^2 + T_{R2}^2 \tag{71.16}$$

Time Constant Methods: Open Circuit Time Constants

The frequency domain/transform methods for finding delay and risetime are particularly valuable for design optimization because they identify dominant time constants. Once the time constants are found, the designer can make efforts to change biases, component values, or optimize the design of the transistors themselves to improve the performance through addressing the relevant bottleneck in performance. The drawback in the above technique is that a network function must be derived. This becomes tedious and time-consuming if the network is of even modest complexity. An alternate technique was developed[36,37] that also can provide reasonable estimates for delay, but with much less computational difficulty. The open-circuit time constant (OCTC) method is widely used for the analysis of the bandwidth of analog electronic circuits just for this reason. It is just as applicable for estimating the delay of very-high-speed digital circuits.

The basis for this technique again comes from the transfer or network function F(s) = Vo(s)/Vi(s). Considering transfer functions containing only poles, the function can be written as

$$F(s) = \frac{a_0}{b_n s^n + b_{n-1}s^{n-1} + \cdots + b_1 s + 1} \tag{71.17}$$

The denominator comes from the product of n factors of the form $(\tau_j s + 1)$, where τ_j is the time constant associated with the *j*-th pole in the transfer function. The b_1 coefficient can be shown to be equal to the sum

$$b_1 = \sum_{j=1}^{n} \tau_j \tag{71.18}$$

of the time constants and b_2 the product of all the time constants. Often, the first-order term dominates the frequency response. In this case, the 3-dB bandwidth is then estimated by $\omega_{3dB} = 1/b_1$. The higher-order terms are neglected. The accuracy of this approach is good, especially when the circuit has a dominant pole. The worst error would occur when all poles have the same frequency. The error in this case is about 25%. Much worse errors can occur however if the poles are complex or if there are zeros in the transfer function as well. We will discuss this later.

Elmore has once again provided the connection we need to obtain delay and risetime estimates from the transfer function. The Elmore delay is given by

$$D = b_1 - a_1 \tag{71.19}$$

where a_1 is the corresponding coefficient of the first-order zero (if any) in the numerator. The risetime is given by

$$T_R^2 = b_1^2 - a_1^2 + 2\left(a_2 - b_2\right) \tag{71.20}$$

In Eq. 71.20, a_2 and b_2 correspond to the coefficients of the second-order zero and pole, respectively.

At this point, it would appear that we have gained nothing since finding that the time constants associated with the poles and zeros is well known to be difficult. Fortunately, it is possible to obtain the b_1 and b_2 coefficients directly by a much simpler method: open-circuit time constants. It has been shown that[35,36]

$$b_1 = \sum_{j=1}^{n} R_{jo} C_j = \sum_{j=1}^{n} \tau_{jo} \tag{71.21}$$

that is, the sum of the time constants τ_{jo}, defined as the product of the effective open-circuit resistance R_{jo} across each capacitor C_j when all other capacitors are open-circuited, equals b_1. These time constants are very easy to calculate since open-circuiting all other capacitors greatly simplifies the network by decoupling many other components. Dependent sources must be considered in the calculation of the R_{jo} open-circuit resistances. Note that these open-circuit time constants are not equal to the pole time constants, but their sum gives the same result for b_1. It should also be noted that the individual OCTCs give the time constant of the network if the j-th capacitor were the only capacitor. Thus, each time constant provides information about the relative contribution of that part of the circuit to the bandwidth or the delay.[35] If one of these is much larger than the rest, this is the place to begin working on the circuit to improve its speed.

The b_2 coefficient can also be found by a similar process,[38] taking the sum of the product of time constants of all possible pairs of capacitors. For example, in a three-capacitor circuit, b_2 is given by

$$b_2 = R_{1o} C_1 R_{2s}^1 C_2 + R_{1o} C_1 R_{3s}^1 C_3 + R_{2o} C_2 R_{3s}^2 C_3 \tag{71.22}$$

where the R_{js}^i resistance is the resistance across capacitor Cj calculated when capacitor C_i is *short*-circuited and all other capacitors are open-circuited. The superscript indicates which capacitor is to be shorted. So, R_{3s}^2 is the resistance across C_3 when C_2 is short-circuited and C_1 is open-circuited. Note that the first time constant in each product is an open-circuit time constant that has already been calculated. In

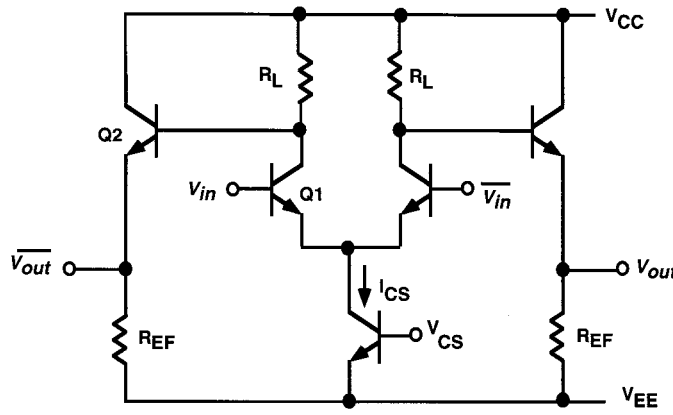


FIGURE 71.11 Schematic of basic ECL inverter.

addition, for any pair of capacitors in the network, we can find an OCTC for one and a SCTC for the other. The order of choice does not matter because

$$R_{io}C_iR_{js}^iC_j = R_{jo}C_jR_{is}^jC_i \tag{71.23}$$

so we are free to choose whichever combination minimizes the computational effort.[38]

At this stage, it would be helpful to illustrate the techniques described above with an example. An ECL inverter whose schematic is shown in Fig. 71.11 is selected for this purpose. The analysis is based on work described in more detail in Ref. 39.

The first step is to construct the large-signal equivalent circuit. We will discuss how to evaluate the large-signal component values later. Figure 71.12 shows such a model applied to the ECL inverter, where

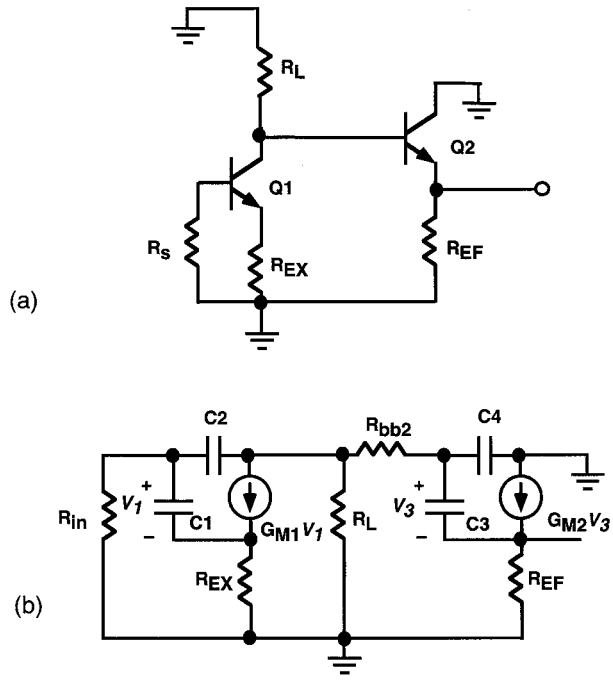


FIGURE 71.12 (a) Large-signal half-circuit model of ECL inverter; and (b) large-signal equivalent circuit of (a).

the half-circuit approximation has been used in Fig. 71.12(a) due to the inherent symmetry of differential circuits.[40] The hybrid-pi BJT model shown in Fig. 71.12(b) has been used with several simplifications. The dynamic input resistance, r_π, has been neglected because other circuit resistances are typically much smaller. The output resistance, r_o, has also been neglected for the same reason. The collector-to-substrate capacitance, C_{CS}, has been neglected because in III-V technologies, semi-insulating substrates are typically used. The capacitance to substrate is quite small compared to other device capacitances. Retained in the model are resistances R_{bb}, the extrinsic and intrinsic base resistance, and R_{EX}, the parasitic emitter resistance. Both of these are very critical for optimizing high-speed performance.

In the circuit itself, R_{IN} is the sum of the driving point resistance from the previous stage, probably an emitter follower output, and R_{bb1} of Q_1. R_L is the collector load resistor, whose value is determined by half of the output voltage swing and the dc emitter current, I_{CS}. $R_L = \Delta V/2I_{CS}$. The R_{EX} of the emitter follower is included in R_{EF}.

We must calculate open-circuit time constants for each of the four capacitors in the circuit. First consider C_1, the base-emitter diffusion and depletion capacitance of Q_1. C_2 is the collector-base depletion capacitance of Q_1. C_3 and C_4 are the corresponding base-emitter and base-collector capacitances of Q_2. Figure 71.13 represents the equivalent circuit schematic when $C_2 = C_3 = C_4 = 0$. A test source, V_1, is placed at the C_1 location. $R_{1o} = V_1/I_1$ is determined by circuit analysis to be

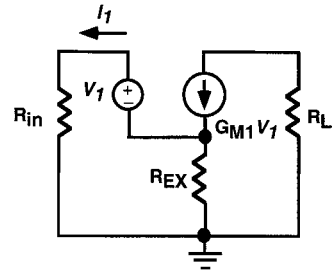


FIGURE 71.13 Equivalent large-signal half-circuit model for calculation of R_{1o}.

$$R_{1o} = \frac{R_{IN} + R_{EX}}{1 + G_{M1}R_{EX}} \tag{71.24}$$

Table 71.1 shows the result of similar calculations for R_{2o}, R_{3o}, and R_{4o}. The b_1 coefficient (first-order estimate of t_D) can now be found from the sum of the OCTCs:

$$b_1 = R_{1o}C_1 + R_{2o}C_2 + R_{3o}C_3 + R_{4o}C_4 \tag{71.25}$$

Considering the results in Table 71.1, one can see that there are many contributors to the time constants and that it will be possible to determine the dominant terms after evaluating the model and circuit parameters.

Next, estimates must be made of the non-linear device parameters, G_{Mi} and C_i. The large signal transconductances can be estimated from

$$G_M = \frac{\Delta I_C}{\Delta V_{BE}} \tag{71.26}$$

For the half-circuit model of the differential pair, the current ΔI_C is the full value of I_{CS} since the device switches between cutoff and I_{CS}. The ΔV_{BE} corresponds to the input voltage swing needed to switch the device between cutoff and I_{CS}. This is on the order of $3V_T$ (or 75 mV) for half of a differential input. So, $G_{M1} = I_{CS}/0.075$ is the large-signal estimate for transconductance of Q_1.

The emitter follower Q_2 is biased at I_{EF} when the output is at V_{OL}. Let us assume that an identical increase in current, I_{EF}, will provide the logic swing needed on the output of the inverter to reach V_{OH}. Thus, $\Delta I_{C2} = I_{EF}$ and $R_{EF} = (V_{OH} - V_{OL})/I_{EF}$. The difference in V_{BE} at the input required to double the collector current can be calculated from

TABLE 71.1 Effective Zero-Frequency Resistances for Open-Circuit Time-Constant Calculation for the Circuit of Fig. 71.12 ($G'_{MI} = G_{MI}/(1 + G_{MI}\,R_{EX})$)

R_{1o}	$\dfrac{R_{IN} + R_{EX}}{1 + G_{M1}R_{EX}}$
R_{2o}	$R_{IN} + R_L + G'_{M1}R_{IN}R_L$
R_{3o}	$\dfrac{R_{bb} + R_L + R_{EF}}{1 + G_{M2}R_{EF}}$
R_{4o}	$R_{bb} + R_L$

$$\Delta V_{BE} = V_T \ln(2) = 0.7 V_T = 17.5\,\text{mV} \tag{71.27}$$

Thus, $G_{M2} = I_{EF}/0.0175$.

C_1 and C_3 consist of the parallel combination of the *depletion* (*space charge*) *layer capacitance*, C_{be}, and the *diffusion capacitance*, C_D. C_2 and C_4 are the base-collector depletion capacitances. Depletion capacitances are voltage varying according to

$$C(V) = C(0)\left(1 - \frac{V}{\phi}\right)^{-m} \tag{71.28}$$

where C(0) is the capacitance at zero bias, ϕ is the built-in voltage, and m the grading coefficient. An equivalent large-signal capacitance can be calculated by

$$C = \frac{Q_2 - Q_1}{V_2 - V_1} \tag{71.29}$$

Q_i is the charge at the initial (1) or final (2) state corresponding to the voltages V_i. $Q_2 - Q_1 = \Delta Q$ and

$$\Delta Q = \int_{V_1}^{V_2} C(V)\,dV \tag{71.30}$$

The large-signal diffusion capacitance can be found from

$$C_D = G_M \tau_f \tag{71.31}$$

where τ_f is the forward transit delay (τ_r) as defined in Section 70.2.

Finally, the Elmore risetime estimate requires the calculation of b_2. Since there are four capacitors in the large-signal equivalent circuit, six terms will be necessary:

$$b_2 = R_{1o}C_1R_{2s}^1C_2 + R_{1o}C_1R_{3s}^1C_3 + R_{1o}C_1R_{4s}^1C_4 + R_{2o}C_2R_{3s}^2C_3 + R_{2o}C_2R_{4s}^2C_4 + R_{3o}C_3R_{4s}^3C_4 \tag{71.32}$$

R_{2s}^1 will be calculated to illustrate the procedure. The remaining short-circuit equivalent resistances are shown in Table 71.2. Referring to Fig. 71.14, the equivalent circuit for calculation of R_{2s}^1 is shown. This is the resistance seen across C_2 when C_1 is shorted. If C_1 is shorted, $V_1 = 0$ and the dependent current source is dead. It can be seen from inspection that

$$R_{2s}^1 = R_{IN} \| R_{EX} + R_L \tag{71.33}$$

Time Constant Methods: Complications

As attractive as the time constant delay and risetime estimates are computationally, the user must beware of complications that will degrade the accuracy by a large margin. First, consider that both methods have depended on a restrictive assumption regarding monotonic risetime. In many cases, however, it is not unusual to experience complex poles. This can occur due to feedback which leads to inductive input or output impedances and emitter or source followers which also have inductive output impedance. When

TABLE 71.2 Effective Resistances for Short Circuit Time Constant Calculation for the Circuit of Fig. 71.12

R_{2s}^{1}	$R_{IN} \Vert R_{EX} + R_L$
R_{3s}^{1}	R_{3o}
R_{4s}^{1}	R_{4o}
R_{3s}^{2}	$\dfrac{\left(\dfrac{1}{G'_{M1}} \Vert R_{IN} \Vert R_L\right) + R_{bb} + R_{EF}}{1 + G_{M2} R_{EF}}$
R_{4s}^{2}	$\left(\dfrac{1}{G'_{M1}} \Vert R_{IN} \Vert R_L\right) + R_{bb}$
R_{4s}^{3}	$\left(R_L + R_{bb}\right) \Vert R_{EF}$

combined with a predominantly capacitive input impedance, complex poles will generally result unless the circuit is well damped. The time constant methods ignore the complex pole effects which can be quite significant if the poles are split and $\sigma \ll j\omega$. In this case, the circuit transient response will exhibit ringing, and time constant estimates of bandwidth, delay, and risetime will be in serious error. Of course, the ringing will show up in the circuit simulation, and if present, must be dealt with by adding damping resistances at appropriate locations.

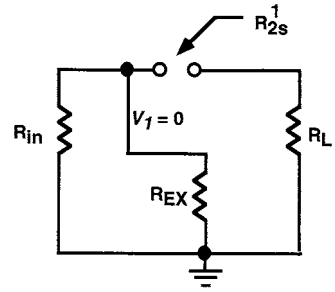


FIGURE 71.14 Equivalent circuit model for calculation of R_{2s}^{1}.

An additional caution must be given for circuits that include zeros. Although Elmore's equations can modify the estimates for t_D and T_R when there are zeros, the OCTC method provides no help in finding the time constants of these zeros. Zeros often occur in wideband amplifier circuits that have been modified through the addition of inductance for shunt peaking, for example. The addition of inductance, either intentionally or accidentally, can also produce complex pole pairs. Zeros are intentionally added for the optimization of speed in very-high-speed digital ICs as well; however, the large area required for the spiral inductors when compared with the area consumed by active devices tends to discourage the use of this method in all but the simplest (and fastest) designs.[35]

Chang, C.E., et al "Logic Design Examples"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

72

Logic Design Examples

Charles E. Chang
Conexant Systems, Inc.

Meera Venkataraman
Troika Networks, Inc.

Stephen I. Long
University of California at Santa Barbara

72.1 Design of MESFET and HEMT Logic Circuits

The basis of dc design, definition of logic levels, noise margin, and transfer characteristics were discussed in Chapter 71 using a DCFL and SCFL inverter as examples. In addition, methods for analysis of high-speed performance of logic circuits were presented. These techniques can be further applied to the design of GaAs MESFET, HEMT, or P-HEMT logic circuits with depletion-mode, enhancement-mode, or mixed E/D FETs. Several circuit topologies have been used for GaAs MESFETs, like direct-coupled FET logic (DCFL), source-coupled FET logic (SCFL), as well as dynamic logic families,[41] and have been extended for use with heterostructure FETs. Depending on the design requirements, whether it be high speed or low power, the designer can adjust the power-delay product by choosing the appropriate device technology and circuit topology, and making the correct design tradeoffs.

Direct-Coupled FET Logic (DCFL)

Among the numerous GaAs logic families, DCFL has emerged as the most popular logic family for high-complexity, low-power LSI/VLSI circuit applications. DCFL is a simple enhancement/depletion-mode GaAs logic family, and the circuit diagram of a DCFL inverter was shown in Fig. 71.2. DCFL is the only static ratioed GaAs logic family capable of VLSI densities due to its compactness and low power dissipation. An example demonstrating DCFL's density is Vitesse Semiconductor's 350K sea-of-gates array. The array uses a two-input DCFL NOR as the basic logic structure. The number of usable gates in the array is 175,000. A typical gate delay is specified at 95 ps with a power dissipation of 0.59 mW for a buffered two-input NOR gate with a fan-out of three, driving a wire load of 0.51 mm.[42] However, a drawback of DCFL is its low noise margin, the logic swing being approximately 600 mV. This makes the logic sensitive to changes in threshold voltage and ground bus voltage shifts.

DCFL NOR and NAND Gate

The DCFL inverter can easily be modified to perform the NOR function by placing additional enhancement-mode MESFETs in parallel as switch devices. A DCFL two-input NOR gate is shown in Fig. 72.1. If any input rises to V_{OH}, the output will drop to V_{OL}. If n inputs are high simultaneously, then V_{OL} will be decreased because the width ratio W_I/W_L in Fig. 71.2 has effectively increased by a factor of n. There is a limit to the number of devices that can be placed in parallel to form very wide NOR functions. The

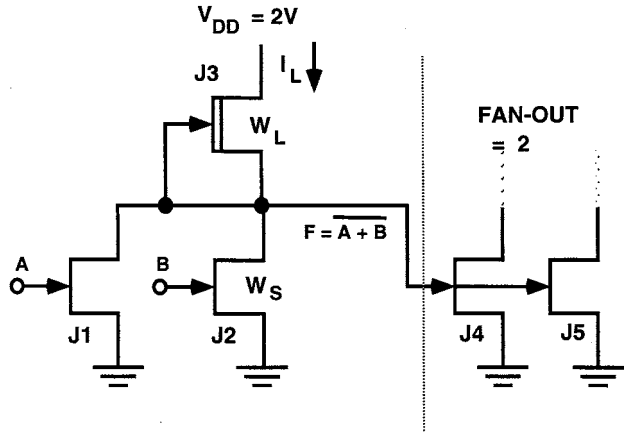


FIGURE 72.1 DCFL two-input NOR gate schematic.

drain capacitance will increase in proportion to the number of inputs, slowing down the risetime of the gate output. Also, the subthreshold current contribution from n parallel devices could become large enough to degrade V_{OH}, and therefore the noise margin. This must be evaluated at the highest operating temperature anticipated because the subthreshold current will increase exponentially with temperature according to[43,44]:

$$I_D = I_S\left[1-\exp\left(\frac{cV_{DS}}{V_T}\right)\right]\left[\exp\left(\frac{bV_{DS}}{V_T}\right)\right]\left[\exp\left(\frac{aV_{GS}}{V_T}\right)\right] \tag{72.1}$$

The parameters a, b, and c are empirical fitting parameters. The first term arises from the diffusion component of the drain current which can be fit from the subthreshold I_D–V_{DS} characteristic at low drain voltage. The second and third terms represent thermionic emission of electrons over the channel barrier from source to drain. The parameters can be obtained by fitting the subthreshold I_D–V_{DS} and I_D–V_{GS} characteristics, respectively, measured in saturation.[45] For the reasons described above, the fan-in of the DCFL NOR is seldom greater than 4.

In addition to the subthreshold current loading, the forward voltage of the Schottky gate diode of the next stage drops with temperature at the rate of approximately –2mV/degree. Higher temperature operation will therefore reduce V_{OH} as well, due to this thermodynamic effect.

A NAND function can also be generated by placing enhancement-mode MESFETs in series rather than in parallel for the switch function. However, the low voltage swing inherent in DCFL greatly limits the application of the NAND function because V_{OL} will be increased by the second series transistor unless the widths of the series devices are increased substantially from the inverter prototype. Also, the switching threshold V_{TH} shown in Fig. 71.1 will be slightly different for each input even if width ratios are made different for the two inputs. The combination of these effects reduces the noise margin even further, making the DCFL NAND implementation generally unsuitable for VLSI applications.

Buffering DCFL Outputs

The output (drain) node of a DCFL gate sources and sinks the current required to charge and discharge the load capacitance due to wiring and fan-out. Excess propagation delay of the order of 5 ps per fan-out is typically observed for small DCFL gates. Sensitivity to wiring capacitance is even higher, such that unbuffered DCFL gates are never used to drive long interconnections unless speed is unimportant. Therefore, an output buffer is frequently used in such cases or when fan-out loading is unusually high.

The superbuffer shown in Fig. 72.2(a) is often used to improve the drive capability of DCFL. It consists of a source follower J_3 and pull-down J_4. The low-to-high transition begins when $V_{IN} = V_{OL}$. J_4 is cut off

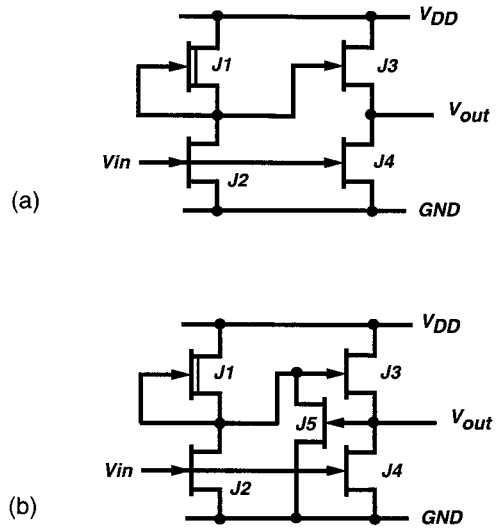


FIGURE 72.2 (a). Superbuffer schematic, and (b) modified superbuffer with clamp transistor. J5 will limit the output current when $V_{out} > 0.7$ V.

and J_3 becomes active, driving the output to V_{OH}. V_{OUT} follows the DCFL inverter output. For the output high-to-low transition, J_4 is driven into its linear region, and the output can be pulled to $V_{OL} = 0$ V in steady state. J_3 is cut off when the DCFL output (drain of J_1) switches from high to low. Since this occurs one propagation delay after the input switched from low-to-high, it is during this transition that the superbuffer can produce a current spike between V_{DD} and ground. J_4 attempts to discharge the load capacitance before the DCFL gate output has cut off J_3. Thus, superbuffers can become an on-chip noise source, so ground bus resistance and inductance must be controlled.

There is also a risk that the next stage might be overdriven with too much input current when driven by a superbuffer. This could happen because the source follower output is capable of delivering high currents when its V_{GS} is maximum. This occurs when $V_{out} = V_{OH} = 0.7$ V, limited by forward conduction of the gate diodes being driven. For a supply voltage of 2 V, a maximum $V_{GS} = 0.7$ V is easily obtained on J_3, leading to the possibility of excess static current flowing into the gates. This would degrade V_{OL} of the subsequent stage due to voltage drop across the internal source resistance. Figure 72.2(b) shows a modified superbuffer design that prevents this problem through the addition of a clamp transistor, J_5. J_5 limits the gate potential of J_3 when the output reaches V_{OH}, thus preventing the overdriving problem.

Source-Coupled FET Logic (SCFL)

SCFL is the preferred choice for very-high-speed applications. An SCFL inverter, a buffered version of the basic differential amplifier cell shown in Fig. 71.4, is shown in Fig. 72.3. The high-speed capability of SCFL stems from four properties of this logic family: small input capacitance, fast discharging time of the differential stage output nodes, good drive capability, and high F_t.

In addition to higher speed, SCFL is characterized by high functional equivalence and reduced sensitivity to threshold voltage variations.[29] The current-mode approach used in SCFL ensures an almost constant current consumption from the power supplies and, therefore, the power supply noise is greatly reduced as compared to other logic families. The differential input signaling also improves the dc, ac, and transient characteristics of SCFL circuits.[46]

SCFL, however, has two drawbacks. First, SCFL is a low-density logic family due to the complex gate topology. Second, SCFL dissipates more power than DCFL, even with the high functional equivalence taken into account.

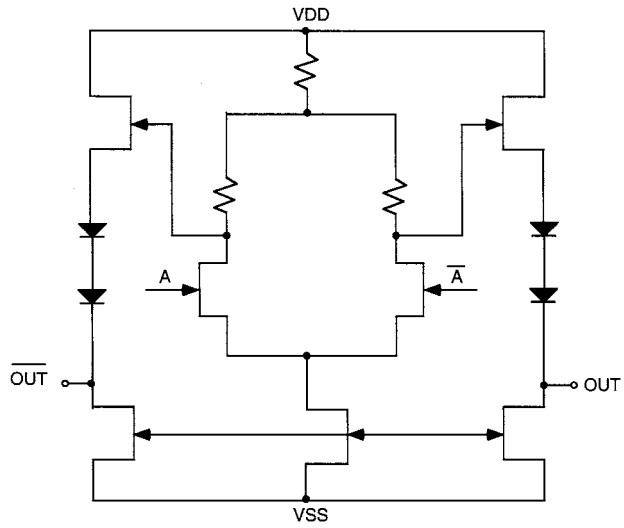


FIGURE 72.3 Schematic diagram of SCFL inverter with source follower output buffering.

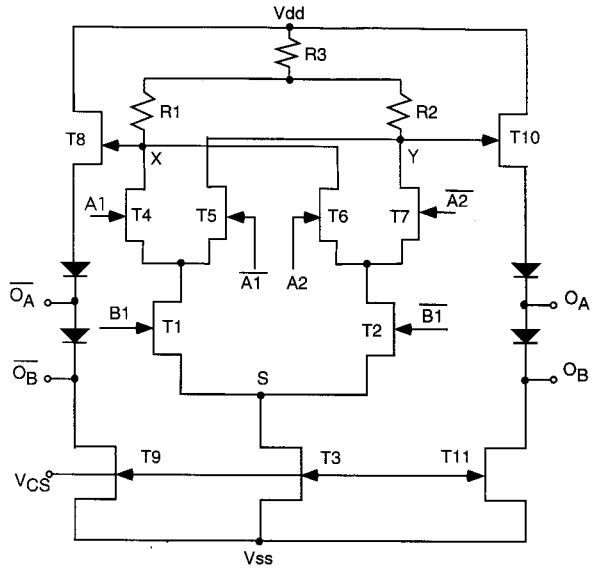


FIGURE 72.4 SCFL two-level series-gated circuit.

SCFL Two-Level Series-Gated Circuit

A circuit diagram of a two-level series-gated SCFL structure is shown in Fig. 72.4 2-to-1 MUXs, XOR gates, and D-latches and flip-flops can be configured using this basic structure. If the A inputs are tied to the data signals and the B inputs are tied to the select signal, the resulting circuit is a 2-to-1 MUX. If the data is fed to the A1 input, the clock is connected to B and the A outputs ($O_A, \overline{O_A}$) are fed back to the A2 inputs, the resulting circuit is a D-latch as seen in Fig. 72.5. Finally, an XOR gate is created by connecting $A_1 = \bar{A}_2$, forming a new input A_{IN} and $\bar{A}_1 = A_2$ to complementary new input $\bar{A}_{IN}$.

The inputs to the two levels require different dc offsets in order for the circuit to function correctly, thus level-shifting networks using diodes or source followers are required. Series logic such as this also requires higher supply voltages in order to keep the devices in their saturation region. This will increase the power dissipation of SCFL.

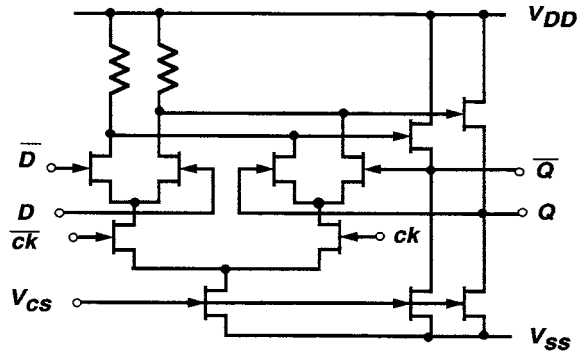


FIGURE 72.5 SCFL D latch schematic. Two cascaded latch cells with opposite clock phasing constitute a master-slave flip-flop.

The logic swing of the circuit shown in Fig. 72.4 is determined by the size of the current source T3 and the load resistors R1 and R2 (R1 = R2). Assuming T3 is in saturation, the logic swing on nodes X and Y is

$$\Delta V_{X,Y} = I_{ds3}\ R1 = I_{dss3}\ R1 \tag{72.2}$$

where I_{dss3} is the saturation current of T3 at Vgs = 0 V. The logic high and low levels on node X ($V_{X,H}$, $V_{X,L}$) are determined from the voltage drop across R3 and Eq. 72.2.

$$V_{X,H} = V_{dd} - (I_{dss3}\ R3) \tag{72.3}$$

$$V_{X,L} = V_{dd} - [I_{dss3}\ (R1 + R3)] \tag{72.4}$$

The noise margin is the difference between the minimum voltage swing required on the inputs to switch the current from one branch to the other (V_{SW}) and the logic swing $\Delta V_{X,Y}$. V_{SW} is set by the ratio between the sizes of the switch transistors (T1,T2, T4-T7) and T3. For symmetry reasons, the sizes of all the switch transistors are kept the same size. Assuming the saturation drain-source current of an FET can be described by the simplified square-law equation:

$$I_{ds} = \beta W(V_{gs} - V_T)^2 \tag{72.5}$$

where V_T is the threshold voltage, W is the FET width, and β is a process-dependent parameter. V_{SW} is calculated assuming all the current from T3 flows only through T2.

$$V_{SW} = \left|V_T\right|\sqrt{\left(W3/W2\right)} \tag{72.6}$$

For a fixed current source size (W3), the larger the size of the switch transistors, the smaller the voltage swing required to switch the current and, hence, a larger noise margin. Although a better noise margin is desirable, it needs to be noted that the larger switch transistors means increased input capacitance and decreased speed. Depending on the design specifications, noise margin and speed need to be traded off.

Since all FETs need to be kept in the saturation region for the correct operation of an SCFL gate, level-shifting is needed between nodes A and B and the input to the next gate, in order to keep T1, T2, and T4-T7 saturated. T3 is kept in saturation if the potential at node S is higher than $V_{SS} + V_{ds,sat}$. The potential at node S is determined by the input voltages to T1 and T2. V_S settles at a potential such that the drain-source current of the conducting transistor is exactly equal to the bias current, I_{dss3}, since no current flows through the other transistor. The minimum logic high level at the output node B ($V_{OB,H}$) is

$$V_{OB,H} \geq V_{ss} + V_{ds,sat} + V_{gs} = V_{ss} + V_{ds,sat} + V_{SW} + V_{th} \tag{72.7}$$

To keep T9 and T11 in saturation, however, requires that

$$V_{OB,H} \geq V_{ss} + V_{ds,sat} + V_{SW} \tag{72.8}$$

As with the voltage on node S, the drain voltages of T1 and T2 are determined by the voltage applied to the A inputs. The saturation condition for T1 and T2 is

$$V_{OA,H} - V_{SW} - V_{th} - V_S = V_{OA,H} - V_{OB,H} \geq V_{ds,sat} \tag{72.9}$$

Equation 72.9 shows that the lower switch transistors are kept in saturation if the level-shifting difference between the A and B outputs is larger than the FET saturation voltage. Since diodes are used for level-shifting, the minimum difference between the two outputs is one diode voltage drop, V_D. If $V_{ds,sat} > V_D$, more diodes are required between the A and B outputs.

The saturation condition for the upper switch transistors, T4 to T7, is determined by the minimum voltage at nodes A and B and the drain voltage of T1 and T2.

$$V_{A,min} - (V_{OA,H} - V_{SW} - V_{th}) \geq V_{ds,sat} \tag{72.10}$$

Substituting Eq. 72.4 into Eq. 72.10 yields

$$\left(V_{dd} - I_{dss3} * \left(R1 + R3\right)\right) - \left(V_{OA,H} - V_{SW} - V_{th}\right) \geq V_{ds,sat} \tag{72.11}$$

Rewriting Eq. 72.11 using Eq. 72.8 gives the minimum power supply range

$$V_{dd} - V_{ss} \geq I_{dss3} * (R1+R3) + 3V_{ds,sat} \tag{72.12}$$

Equation 72.11 allows the determination of the minimum amount of level-shifting required between nodes A and B to the outputs

$$V_{A,H} - V_{OA,H} \geq V_{ds,sat} + I_{dss3} * R1 - V_{th} - V_{SW} \tag{72.13}$$

Equations 72.8 to 72.13 can be used for designing the level shifters. The design parameters available in the level-shifters are the widths of the source followers (W8, W10), the current sources (W9, W11), and the diode (W_D). Assuming the current source width (W9) is fixed, the voltage drop across the diodes is partially determined by the ratio (W_D/W9). This ratio should not be made too small. Operating Schottky diodes at high current density will result in higher voltage drop, but this voltage will be partially due to the $I_D R_S$ drop across the parasitic series resistance. Since this resistance is often process dependent and difficult to reproduce, poor reproducibility of V_D will result in this case.

The ratio between the widths of the source follower and the current source (W8/W9) determines the gate-source voltage of the source follower (V_{gs8}). V_{gs8} should be kept below 0.5 V to prevent gate-source conduction.

The dc design of the two-level series-gated SCFL gate in Fig. 72.4 can be accomplished by applying Eqs. 72.2 to 72.13. Ratios between most device sizes can be determined by choosing the required noise margin and logic swing. Only W3 in the differential stage and W9 among the level-shifters are unknown at this stage. All other device sizes can be expressed in terms of these two transistor widths.

The relation between W3 and W9 can be determined only by considering transient behavior. For a given total power dissipation, the ratio between the power dissipated in the differential stage and the output buffers determines how fast the outputs are switched. If fast switching at the outputs is desired, more power needs to be allocated to the output buffers and, consequently, less power to the differential

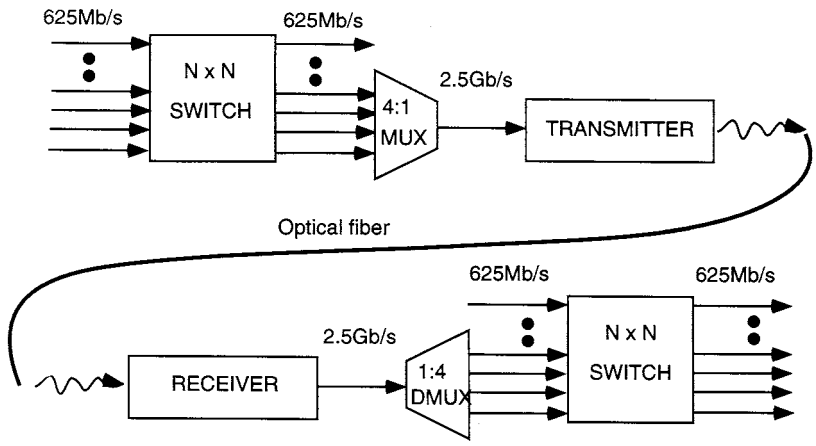


FIGURE 72.6 2.5-Gb/s optical communication system.

stage. While this allocation will ensure faster switching at the output, the switching speed of the differential stage is reduced because of the reduced current available to charge and discharge the large input capacitance of the output buffers.

Finally, it is useful to note that scaling devices to make a speed/power tradeoff is simple in SCFL. If twice as much power is allocated to a gate, all transistors and diodes are made twice as wide while all resistors are reduced by half.[46]

Advanced MESFET/HEMT Design Examples

High-Speed TDM Applications

The need for high bandwidth transmission systems continues to increase as the number of bandwidth-intensive applications in the areas of video imaging, multimedia and data communication (such as database sharing and database warehousing) continues to grow. This has led to the development of optical communication systems with transmission bit rates, for example, of 2.5 Gb/s and 10 Gb/s. A simplified schematic of a 2.5 Gb/s communication system is shown in Fig. 72.6.

As seen in Fig. 72.6, MUXs, DMUXs, and switches capable of operating in the Gb/s range are crucial for the operation of these systems. GaAs MESFET technology has been employed extensively in the design of these high-speed circuits because of the excellent intrinsic speed performance of GaAs. SCFL is especially well suited for these circuits where high speed is of utmost importance and power dissipation is not a critical factor.

The design strategies employed in the previous subsection can now be further applied to a high-speed 4:1 MUX, as shown in Fig. 72.7. It was shown that the two-level series gated SCFL structure could be easily configured into a D-latch. The MSFF in the figure is simply a master-slave flip-flop containing two D-latches. The PSFF is a phase-shifting flip-flop that contains three D-latches and has a phase shift of 180° compared with an MSFF.

The 4:1 MUX is constructed using a tree-architecture in which two 2:1 MUXs merge two input lines each into one output operating at twice the input bit rate. The 2:1 MUX at the second stage takes the two outputs of the first stage and merges it into a single output at four times the primary input bit rate. The architecture is highly pipelined, ensuring good timing at all points in the circuit. The inherent propagation delay of the flip-flops ensures that the signals are passed through the selector only when they are stable.[46]

The interface between the two stages of 2:1 MUXs is timing-critical, and care needs to be taken to obtain the best possible phase margin at the input of the last flip-flop. To accomplish this, a delay is added between the CLK signal and the clock input to this flip-flop. The delay is usually implemented

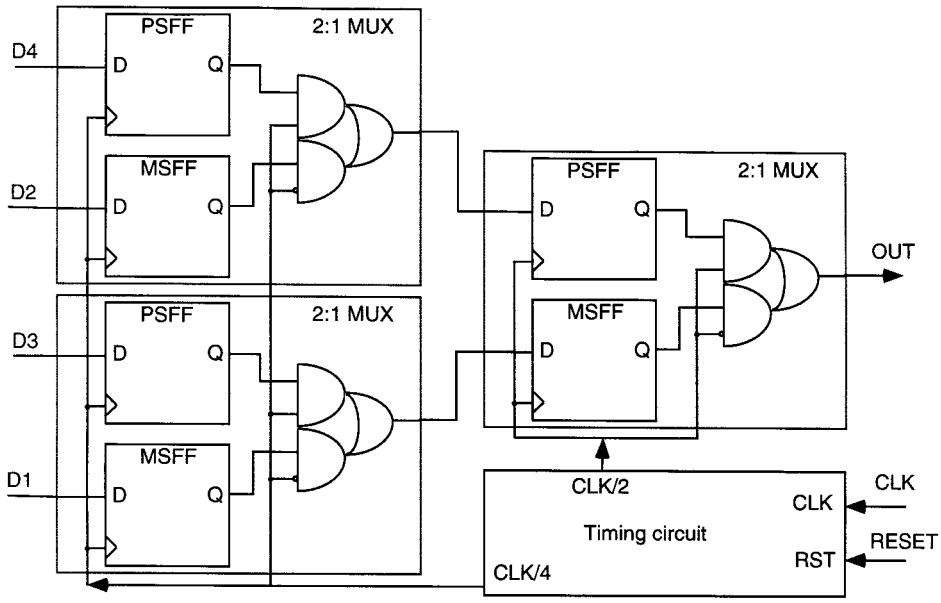


FIGURE 72.7 High-speed 4:1 multiplexer (MUX).

using logic gates because their delays are well-characterized in a given process. Output jitter can be minimized if 50% duty-cycle clock signals are used. Otherwise, a retiming MSFF will be needed at the output of the 4:1 MUX.

The 4:1 MUX is a good example of an application of GaAs MESFETs with very-high-speed operation and low levels of integration. Vitesse Semiconductor has several standard products operating at the Gb/s range fabricated in GaAs using their own proprietary E/D MESFET process. For example, the 16×16 crosspoint switch, VSC880, has serial data rates of 2.0 Gb/s. The VS8004 4-bit MUX is a high-speed, parallel-to-serial data converter. The parallel inputs accept data at rates up to 625 Mb/s and the differential serial data output presents the data sequentially at 2.5 Gb/s, synchronous with the differential high-speed clock input.[42]

While the MESFET technologies have proven capable at 2.5 and 10 Gb/s data rates for optical fiber communication applications, higher speeds appear to require heterojunction technologies. The 40-Gb/s TDM application is the next step, but it is challenging for all present semiconductor device IC technologies. A complete 40-Gb/s system has been implemented in the laboratory with 0.1-μm InAlAs/InGaAs/InP HEMT ICs as reported in Refs. 47 and 48. Chips were fabricated that implemented multiplexers, photodiode preamplifiers, wideband dc 47-GHz amplifiers, decision circuits, demultiplexers, frequency dividers, and limiting amplifiers. The high-speed static dividers used the super-dynamic FF approach.[49]

A 0.2-μm AlGaAs/GaAs/AlGaAs HEMT quantum well device technology has also demonstrated 40-Gb/s TDM system components. A single chip has been reported that included clock recovery, data decision, and a 2:4 demultiplexer circuit.[50] The SCFL circuit approach was employed.

Very-High-Speed Dynamic Circuits

Conventional logic circuits using static DCFL or SCFL NOR gates such as those described above are limited in their maximum speed by loaded gate delays and serial propagation delays. For example, a typical DCFL NOR-implemented edge-triggered DFF has a maximum clock frequency of approximately $1/5\tau_D$ and the SCFL MSFF is faster, but it is still limited to $1/2\tau_D$ at best. Frequency divider applications that require clock frequencies above 40 GHz have occasionally employed alternative circuit approaches which are not limited in the same sense by gate delays and often use dynamic charge storage on gate

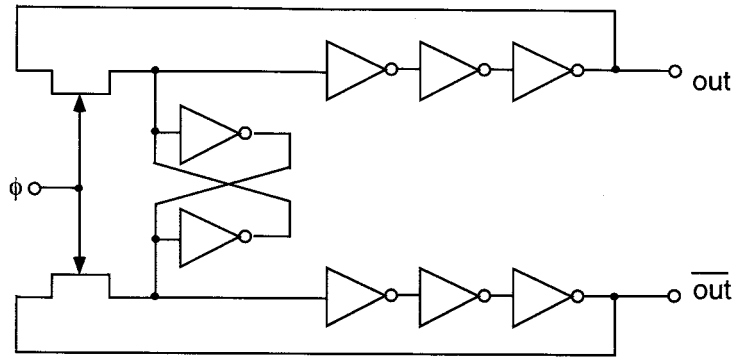


FIGURE 72.8 Dynamic frequency divider (DFD) divide-by-2 circuit. (Ref. 51, ©1989 IEEE, with permission.)

nodes for temporarily holding a logic state. These approaches have been limited to relatively simple circuit functions such as divide-by-2 or -4.

The dynamic frequency divider (DFD) technique is one of the well-known methods for increasing clock frequency closer to the limits of a device technology. For an example, Fig. 72.8 shows a DFD circuit using a single-phase clock, a cross-coupled inverter pair as a latch to reduce the minimum clock frequency, and pass transistors to gate a short chain of inverters.[51,52] These have generally used DCFL or DCFL superbuffers for the inverters. The cross-coupled inverter pair can be made small in width, since its serial delay is not in the datapath. But the series inverter chain must be designed to be very fast, generally requiring high power per inverter in order to push the power-delay product to its extreme high-speed end. Since fan-out is low, the intrinsic delays of an inverter in a given technology can be approached.

The maximum and minimum clock frequencies of this circuit can be calculated from the gate delays of the n series inverters as shown in Eqs. 72.14 and 72.15. An odd number n is required to force an inversion of the data so that the circuit will divide-by-2. Here, t_1 is the propagation delay of the pass transistor switches, J1 and J2, and t_D is the propagation delay of the DCFL inverters. The parameter "a" is the duty cycle of the clock. For a 50% clock duty cycle, the range of minimum to maximum clock frequency is about 2 to 1.

$$f_{\phi\max} = \frac{1}{t_1 + nt_D} \tag{72.14}$$

$$f_{\phi\min} = \frac{a}{t_1 + nt_D} \tag{72.15}$$

Clock frequencies as high as 51 GHz have been reported using this approach with a GaAs/AlGaAs P-HEMT technology.[52] The power dissipation was relatively high, 440 mW. Other DFD circuit approaches can also be found in the literature.[53-55]

A completely different approach, as shown in Fig. 72.9, utilizes an injection-locked push-pull oscillator (J1 and J2) whose free running frequency is a subharmonic of the desired input frequency.[56] FETs J3 and J4 are operating in their ohmic regions and act as variable resistors. The variation in V_{GS1} and V_{GS2} cause the oscillator to subharmonically injection-lock to the input source. Here, a divide-by-4 ratio was demonstrated with an input frequency of 75 GHz and a power dissipation of 170 mW using a 0.1-μm InP-based HEMT technology with f_T = 140 GHz and f_{max} = 240 GHz. This divider also operated in the 59–64 GHz range with only –10 dBm RF input power. The frequency range is limited by the tuning range of the oscillator. In this example, about 2 octaves of frequency variation was demonstrated.

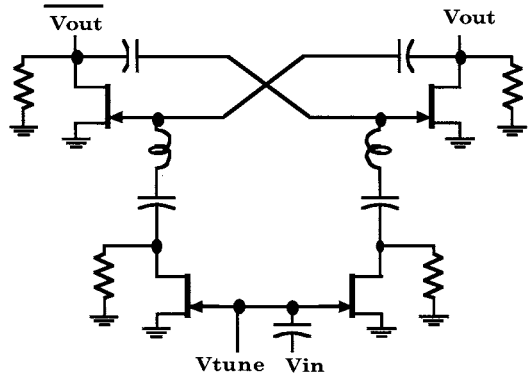


FIGURE 72.9 Injection-locked oscillator divide-by-4 (Ref. 56, ©1996 IEEE. With permission.)

Finally, efforts have also been made to beat the speed limitations of a technology by dynamic design methods while still maintaining minimum power dissipation. The quasi-dynamic FF[48,49] and quasi-differential FF[57] are examples of circuit designs emphasizing this objective. The latter has achieved 16-GHz clock frequency with approximately 2 mW of power per FF.

72.2 HBT Logic Design Examples

From a circuit topology perspective, both III-V HBTs and silicon BJTs are interchangeable, with myriad similarities and a few essential differences. The traditional logic families developed for the Si BJTs serve as the starting point for high-speed logic with III-V HBTs. During the period of intense HBT development in the 1980s and early 1990s, HBTs have implemented ECL, CML, DTL, and, I^2L logic topologies as well as novel logic families with advanced quantum devices (such as resonant tunneling diodes[1]) in the hopes of achieving any combination of high-speed, low-power, and high-integration level. During that time, III-V HBTs demonstrated their potential integration limits with an I^2L 32-bit microprocessor[58] and benchmarked its high-speed ability with an ECL 30-GHz static master/slave flip-flop based frequency divider. During the same time, advances in Si based technology, especially CMOS, have demonstrated that parallel circuit algorithms implemented in a technology with slower low-power devices capable of massive integration will dominate most applications. Consequently, III-V-based technologies such as HBTs and MESFET/HEMT have been relegated to smaller but lucrative niche markets.

As HBT technology evolved into a mature production technology in the mid-1990s, it was clear that III-V HBT technology had a clear advantage in high-speed digital circuits, microwave integrated circuits, and power amplifier markets. Today, in the high-speed digital arena, III-V HBTs have found success in telecom and datacom lightwave communication circuits for SONET/ATM-based links that operate from 2.5 to 40 Gb/s. HBTs also dominate the high-speed data conversion area with Nyquist-rate ADCs capable of gigabit/gigahertz sampling rates/bandwidths, sigma-delta ADCs with very high oversampling rates, and direct digital synthesizers with gigahertz clock frequencies and ultra-low spurious outputs. In these applications, the primary requirement is ultra-high-speed performance with LSI (10 K transistors) levels of integration. Today, the dominant logic type used in HBT designs is based on non-saturating emitter coupled pairs such as ECL and current-mode logic (CML), which is the focus of this chapter.

III-V HBT for Circuit Designers

III-V HBTs and Si BJTs are inherently bipolar in nature. Thus, from a circuit point of view, both share many striking similarities and some important differences. The key differences between III-V HBT

technology and Si BJT technology, as discussed below, can be traced to three essential aspects: (1) heterojunction vs. homojunction, (2) III-V material properties, and (3) substrate properties.

First, the primary advantage of a base–emitter heterojunction is that the wide bandgap emitter allows the base to be doped higher than the emitter (typically 10 to 50X in GaAs/AlGaAs HBTs) without a reduction in current gain. This translates to lower base resistance for improved f_{max} and reduces base width modulation with V_{ce} for low output conductance. Alternatively, the base can be made thinner for lower base-transit time (τ_b) and higher f_t without having R_b too high. If the base composition is also graded from high bandgap to low, an electric field can be established to sweep electrons across the base for reduced τ_b and higher f_t. With a heterojunction B-E and a homojunction B-C, the junction turn-on voltage is higher in the B-E than it is in the B-C. This results in a common-emitter I–V curve offset from the off to saturation transition. This offset is approximately 200 mV in GaAs/AlGaAs HBTs. With a highly doped base, base punch-through is not typically observed in HBTs and does not limit the f_t-breakdown voltage product as in high-performance Si BJTs and SiGe HBT with thin bases. Furthermore, if a heterojunction is placed in the base–collector junction, a larger bandgap material in the collector can increase the breakdown voltage of the device and reduce the I–V offset.

Second, III-V semiconductors typically offer higher electron mobility than Si for overall lower τ_b and collector space charge layer transit times (τ_{cscl}). Furthermore, many III-V materials exhibit velocity overshoot in the carrier drift velocity. When HBTs are designed to exploit this effect, significant reductions in τ_{cscl} can result. With short collectors, the higher electron mobility can result in ultra-high f_t; however, this can also be used to form longer collectors with still acceptable τ_{cscl}, but significantly reduced C_{bc} for high f_{max}. The higher mobility in the collector can also lead to HBTs with lower turn on resistance in the common emitter I–V curves.

Since GaAs/AlGaAs and GaAs/InGaP have wider bandgaps than Si, the turn-on voltage of the B–E ($V_{be,on}$) junction is typically on the order of 1.4 V vs. 0.9 V for advanced high-speed Si BJT. InP-based HBTs can have $V_{be,on}$ on the order of 0.7 V; however, most mature production technologies capable of LSI integration levels are based on AlGaAs/GaAs or InGaP/GaAs. The base–collector turn-on voltage is typically on the order of 1 V in GaAs-based HBTs. This allows V_{ce} to be about 600 mV lower than V_{be} without placing the device in saturation. The wide bandgap material typically results in higher breakdown voltages, so III-V HBTs typically have a high Johnson figure of merit (f_t * breakdown voltage) compared with Si- and SiGe-based bipolar transistors.

The other key material differences between III-V vs. silicon materials are the lack of a native stable oxide in III-V, the extensive use of poly-Si in silicon-based processes, and the heavy use of implants and diffusion for doping silicon devices. III-V HBTs typically use epitaxial growth techniques, and interconnect step height coverage issues limit the practical structure to one device type, so PNP transistors are not typically included in an HBT process. These key factors contribute to the differences between HBTs and BJTs in terms of fabrication.

Third, the GaAs substrate used in III-V HBTs is semi-insulating, which minimizes parasitic capacitance to ground through the substrate, unlike the resistive silicon substrate. Therefore, the substrate contact as in Si BJTs is unnecessary with III-V HBTs. In fact, the RF performance of small III-V HBT devices can be measured directly on-wafer without significant de-embedding of the probe pads below 26 GHz. For interconnects, the line capacitance is typically dominated by parallel wire-to-wire capacitance, and the loss is not limited by the resistive substrate. This allows for the formation of high-Q inductors, low-loss transmission lines, and longer interconnects that can be operated in the 10's of GHz. Although BESOI and SIMOX Si wafers are insulating, the SiO_2 layer is typically thin resulting in reduced but still significant capacitive coupling across this thin layer.[59]

Most III-V substrates have a lower thermal conductivity than bulk Si, resulting in observed self-heating effects. For a GaAs/AlGaAs HBT, this results in observed negative output conductance in the common-emitter I–V curve measured with constant I_b. The thermal time constant for GaAs/AlGaAs HBTs is on the order of microseconds. Since thermal effects cannot track above this frequency, the output conductance

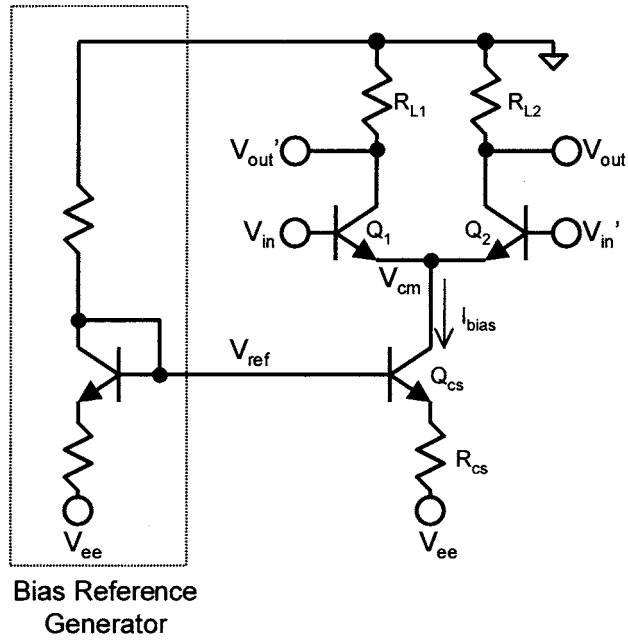


FIGURE 72.10 Standard differential CML buffer with a simple reference generator.

of HBTs at RF (> 10 MHz) is low but positive. This effect does result in a small complication for HBT models based on the standard Gummel Poon BJT model.

Current-Mode Logic

The basic current-mode logic (CML) buffer/inverter cell is shown in Fig. 72.10. The CML buffer is a differential amplifier that is operated with its outputs clipped or in saturation. The differential inputs (V_{in} and V_{in}') are applied to the bases of Q_1 and Q_2. The difference in potential between V_{in} and V_{in}' determines which transistor I_{bias} is steered through, resulting in a voltage drop across either load resistance R_{L1} or R_{L2}. If $V_{in} = V_{OH}$ and $V_{in}' = V_{OL}$ ($V_{in,High} > V_{in,Low}$), Q_1 is on and Q_2 is off. Consequently, I_{bias} completely flows through R_{L1}, causing V_{out}' to drop for a logic low. With Q_2 off, V_{out} floats to ground for a logic high. If the terminal assignment of V_{out} and V_{out}' were reversed, this CML stage would be an inverter instead of a buffer

The logic high V_{OH} of a CML gate is 0 V. The logic low output is determined by $V_{OL} = -R_{L1}I_{bias}$. With R_{L1}/R_{L2} = 200-Ωs, and I_{bias} = 2 mA, the traditional logic low of a CML gate is –400 mV. As CML gates are cascaded together, the outputs of one stage directly feed the inputs of another CML gate. As a result, the base-collector of the "on" transistor is slightly forward-biased (by 400 mV in this example). For high-speed operation, it is necessary to keep the switching transistors out of saturation. With a GaAs base-collector turn-on voltage near 1V, 500 to 600 mV forward-bias is typically tolerated without any saturation effects. In fact, this bias shortens the base-collector depletion region, resulting in the highest f_t vs. V_{ce} (f_{max} suffers due to increase in C_{bc}). As a result, maximum logic swing of a CML gate is constrained by the need to keep the transistors out of saturation. As the transistor is turned on, the logic high is actively pulled to a logic low; however, as the transistor is turned off, the logic low is pulled up by a RC time constant. With a large capacitive loading, it is possible that the risetime is slower than the falltime, and that may result in some complications with high-speed data.

A current mirror (Q_{cs} and R_{cs}) sets the bias (I_{bias}) of the differential pair. This is an essential parameter in determining the performance of CML logic. In HBTs, the f_t dependence on I_c is as follows:

$$1/2\pi f_t = \tau_{ec} = nkT/\left(qI_c\left(C_{bej} + C_{bc} + C_{bed}\right)\right) + R_c\left(C_{bej} + C_{bc}\right) + \tau_b + \tau_{cscl} \tag{72.16}$$

where τ_{ec} is the total emitter-to-collector transit time, qI_c/nkT is the transconductance (g_m), C_{bc} is the base-collector capacitance, C_{bej} is the base–emitter junction capacitance, C_{bed} is the B–E diffusion capacitance, R_c is the collector resistance, τ_b is the base transit time, and τ_{cscl} is the collector space charge layer transit time. At low currents, the transit time is dominated by the device g_m and device capacitance. As the bias increases, τ_{ec} is eventually limited by τ_b and τ_{cscl}. As this limit approaches, Kirk effect typically starts to increase τ_b/τ_{cscl}, which decreases f_t. In some HBTs, the peak f_{max} occurs a bit after the peak f_t. With this in mind, optimal performance is typically achieved when I_{bias} is near $I_{c,maxft}$ or $I_{c,maxfmax}$. In some HBT technologies, the maximum bias may be constrained by thermal or reliability concerns. As a rule of thumb, the maximum current density of HBTs is typically on the order of 5×10^4 A/cm^2.

The bias of CML and ECL logic is typically set with a bias reference generator, where the simplest generator is shown in Fig. 72.10. Much effort has been invested in the design of the reference generator to maintain constant bias with power supply and temperature variation. In HBT, secondary effects of heterojunction design typically result in slightly varying ideality factor with bias. This makes the design of bandgap reference circuits quite difficult in most HBT technologies, which complicates the design of the reference generator. Nevertheless, the reference generators used today typically result in a 2% variation in bias current from –40 to 100C with a 10% variation in power supply. In most applications, the voltage drop across R_{cs} is set to around 400 mV. With V_{ee} set at –5.2 V, V_{ref} is typically near –3.4 V. With constant bias, as V_{ee} moves by ±10%, then the voltage drop across R_{cs} remains constant, so V_{ref} moves by the change in power supply (about ±0.5 V). Since the logic levels are referenced to ground, the average value of V_{cm} (around –1.4 V) remains constant. This implies that changes in the power supply are absorbed by the base–collector junction of Q_{cs}, and it is important that this transistor is not deeply saturated.

Since the device goes from the cutoff mode to the forward active mode as it switches, the gate delay is difficult to predict analytically with small-signal analysis. Thus, large-signal models are typically used to numerically compute the delay in order to optimize the performance of a CML gate. Nevertheless, the small-signal model, frequency-domain approach described in Chapter 71.3 (Elmore) leads to the following approximation of a CML delay gate with unity fan-out[33]:

$$\tau_{d,cml} = \left(1 + g_m R_L\right)R_b C_{bc} + R_b\left(C_{be} + C_d\right) + \left(2C_{bc} + 1/2C_{be} + 1/2C_d\right)/g_m \tag{72.17}$$

where C_d is the diffusion capacitance of $g_m(\tau_b + \tau_{cscl})$. Furthermore, by considering the difference in charge storage at logic high and logic low, divided by the logic swing, the effective CML gate capacitance can be expressed[19] as

$$C_{cml} = C_{be}/2 + 2C_{bc} + C_s + \left(\tau_b + \tau_{cscl}\right)/R_L \tag{72.18}$$

where C_s is collector-substrate and interconnect capacitances. Both equations show that the load resistor and bias (which affects g_m and device capacitors) have a strong effect on performance.

For a rough estimate of the CML maximum speed without loading, one can assume that f_t is the gain bandwidth product. With the voltage gain set at $g_m R_L$, the maximum speed is $f_t/(g_m R_L)$. In the above example, at 1 mA average bias, $g_{m,int}$ = 1/26 S at room temperature. Assuming the internal parasitic emitter resistance R_E is 10 ohms and using the fact that $g_{m,ext} = g_{m,int}/(1 + R_E g_{m,int})$, the effective extrinsic g_m is 1/36 mhos. With a 200-ohm load resistor, the voltage gain is approximately 5.5. With a 70-GHz f_t HBT process, the maximum switching rate is about 13 GHz. Although this estimate is quite rough, it does show that high-speed CML logic desires high device bias and low logic swing. In most differential circuits, only 3 to 4 kT/q is needed to switch the transistors and overcome the noise floor. With such low levels

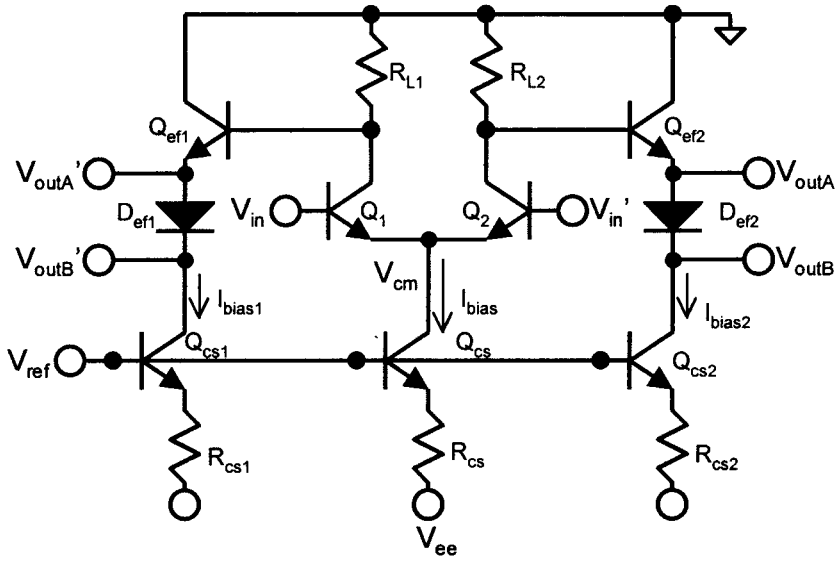


FIGURE 72.11 Standard differential ECL buffer with outputs taken at two different voltage levels.

and limited gain, the output may not saturate to the logic extremes, resulting in decreasing noise margin. In practice, the differential logic level should not be allowed to drop below 225 mV. With a 225-mV swing vs. 400 mV, the maximum gate bandwidth improves to 23 GHz from 13 GHz.

Emitter-Coupled Logic

By adding emitter followers to the basic HBT CML buffer, the HBT ECL buffer is formed in Fig. 72.11. From a dc perspective, the emitter followers (Q_{ef1} and Q_{ef2}) shift the CML logic level down by V_{be}. With the outputs at V_{outA}/V_{outA}' and 400 mV swing from the differential pair, the first level ECL logic high is –1.4 V and the ECL logic low is –1.8 V. For some ECL logic gates, a second level is created through a Schottky diode voltage shift (D_{ef1}/D_{ef2}). The typical Schottky diode turn-on voltage for GaAs is 0.7 V, so the output at V_{outB}/V_{outB}' is –2.1 V for a logic high and –2.5 V for a logic low. In general, the HBT ECL levels differ quite a bit from the standard Si ECL levels. Although resistors can be used to bias Q_{ef1}/Q_{ef2}, current mirrors (Q_{cs1}/Q_{cs2} and R_{cs1}/R_{cs2}) are typically used. Current mirrors offer stable bias with logic level at the expense of higher capacitance, while resistors offer lower capacitance but the bias varies more and may be physically quite large.

From an ac point of view, emitter followers have high input impedance (approximately β times larger than an unbuffered input) and low output impedance (approx. $1/g_m$), which makes it an ideal buffer. In comparison with CML gates, since the differential pair now drives a higher load impedance, the effect of loading is reduced, yielding increased bandwidth, faster edge rates, and higher fan-out. The cost of this improvement is the increase in power due to the bias current of the emitter followers. For example, in a 50 GHz HBT process, a CML buffer (fan-out = 1, I_{bias} = 2mA, R_{L1}/R_{L2} = 150 Ω), the propagation delay (t_D) is 14.8 ps with a risetime [20 to 80%] (t_r) of 31 ps and a falltime [20 to 80%] (t_f) of 21 ps. In comparison, an ECL buffer with level 1 outputs (fan-out = 1, I_{bias} = 2 mA, I_{bias1}/I_{bias2} = 2 mA, R_{L1}/R_{L2} = 150 Ω) has t_d = 14 ps, t_f = 9 ps, and t_r = 17 ps. With a threefold increase in P_{diss}, the impedance transformation of the EF stage results in slightly reduced gate delays and significant improvements in the rise/falltimes. With the above ECL buffer modified for level 2 (level shifted) outputs, the performance is only slightly lower with t_D = 14.2 ps, t_f = 11 ps, and t_r = 22 ps.

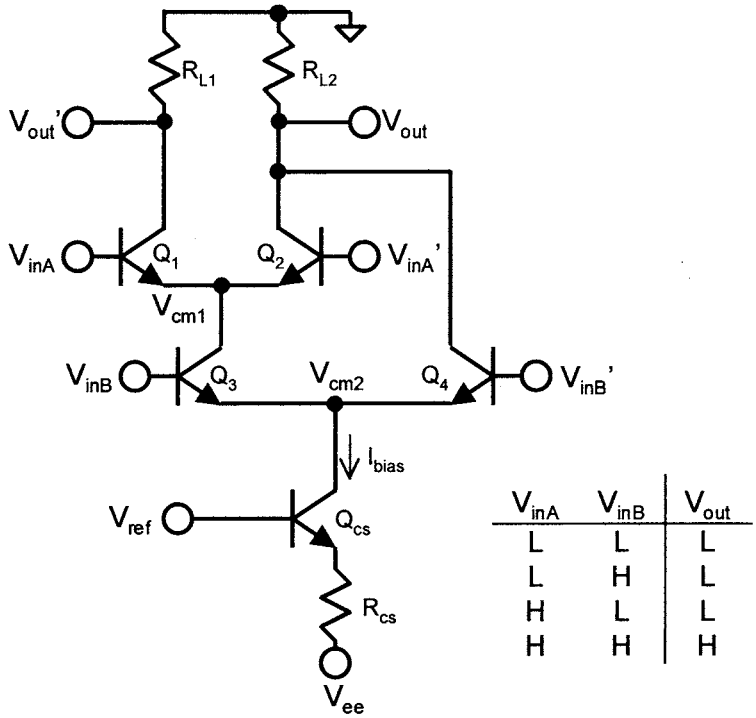
In general, emitter followers tend to have bandwidths approaching the f_t of the device, which is significantly higher than the differential pair. Consequently, it is possible to obtain high-speed operation with the EF biased lower than would be necessary to obtain the maximum device f_t. With the ECL level 1

buffer, if the I_{bias1}/I_{bias2} is lowered to 1 mA from 2mA, the performance is still quite high, with t_d = 15 ps, t_f = 13 ps, and t_r = 18 ps. Although t_D approaches the CML case, the t_f and t_r are still significantly better.

As the EF bias is increased, its driving ability is also increased; however, at some point with high bias, the output impedance of the EF becomes increasingly inductive. When combined with large load capacitance (as in the case of high fan-out or long interconnect), it may result in severe ringing in the output that can result in excessive jitter on data edges. The addition of a series resistor between the EF output and the next stage can help to dampen the ringing by increasing the real part of the load. This change, however, increases the RC time constant, which usually results in a significant reduction in performance. In practice, changing the impedance of the EF bias source (high impedance current source or resistor bias) does not have a significant effect on the ringing. As a result, the primary method to control the ringing is through the EF bias, which places a very real constraint on bandwidth, fan-out, and jitter that needs to be considered in the topology of real designs. In some FET DCFL designs, several source followers are cascaded together to increase the input impedance and lower the output resistance between two differential pairs for high-bandwidth drive. Due to voltage headroom limits, it is very difficult to cascade two HBT emitter followers without causing the current source to enter deep saturation. In general, ECL gates are typically used for the high-speed sections due to significant improvement in rise/falltimes (bandwidth) and drive ability, although the power dissipation is higher.

ECL/CML Logic Examples

Typically, ECL and CML logic is mixed throughout high-speed GaAs/AlGaAs HBT designs. As a result, there are three available logic levels that can be used to interconnect various gates. The levels are CML (0/–400 mV), ECL1 (–1.4/–1/8 V), and ECL2 (–2.1/–2.5 V). To form more complex logic functions, typically two levels of transistors are used to steer the bias current. Figure 72.12 shows an example of an CML AND/NAND gate. For the ECL counterpart, it is only necessary to add the emitter followers. The top input is V_{inA}/V_{inA}'. The bottom input is V_{inB}/V_{inB}'. In general, the top can be driven with either the



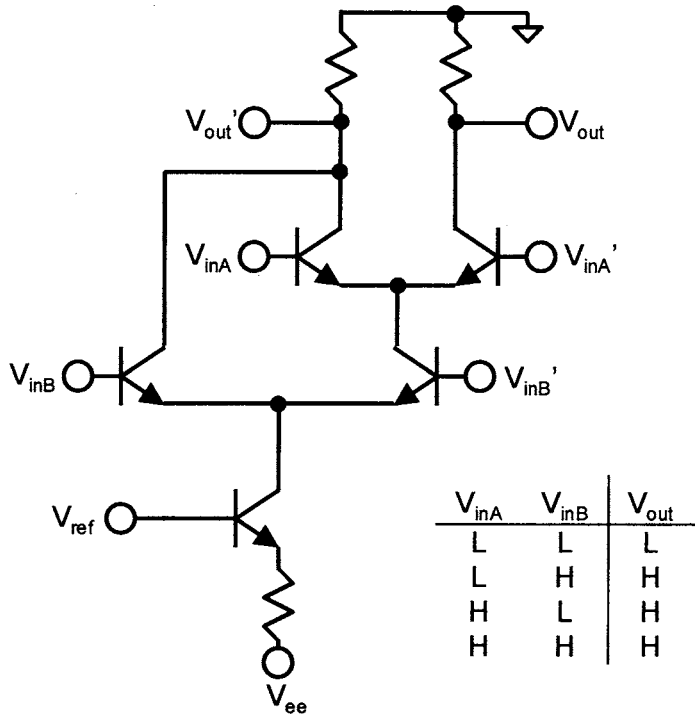
V_{inA}	V_{inB}	V_{out}
L	L	L
L	H	L
H	L	L
H	H	H

FIGURE 72.12 Two-level differential CML AND gate.

CML or the ECL1 inputs, and the bottom level can be driven by ECL1 and ECL2 levels. The choice of logic input levels are typically dictated by the design tradeoff between bandwidth, power dissipation, and fan-out. As seen in Fig. 72.12, only when V_{inA} and V_{inB} are high, will I_{bias} current be steered into the load resistor that makes $V_{out} = V_{OH}$. All other combinations will make $V_{out} = V_{OL}$, as required by the AND function. Due to the differential nature, if the output terminal labels were reversed, this would be a NAND gate.

For the worst-case voltage headroom, V_{inA}/V_{inA}' is driven with ECL1 levels, resulting in V_{cm1} of –2.8 V. With an ECL2 high on V_{inB} (–2.1 V), the lower stage (Q_3/Q_4) has a B-C forward-bias of 0.7 V, which may result in a slight saturation. This also implies that V_{cm2} is around –3.5 V, which results in an acceptable nominal 100 mV forward-bias on the current source transistor (Q_{cs}). As V_{ee} becomes less negative, the change in V_{ee} is absorbed across Q_{cs}, which places Q_{cs} closer into saturation. In saturation, the current source holds I_e in Q_{cs} constant; so if I_b increases (due to saturation), then I_c decreases. For some current source reference designs that cannot source the increased I_b, the increased loading due to saturated I_b may lower V_{ref}, which would have a global effect on the circuit bias. If the current source reference can support the increase in I_b, then the bias of only the local saturated differential pair starts to decrease leading to the potential of lower speed and lower logic swing. For HBTs, the worst-case Q_{cs} saturation occurs at low temperature, and the worst-case saturation for Q_3/Q_4 occurs at high temperature since V_{be} changes by –1.4 mV/C and V_{diode} = –1.1 mV/C (for constant-current bias). It is possible to decrease the forward-bias of the lower stage by using the base-emitter diode as the level shift to generate the second ECL levels; however, the power supply voltage needs to increase from –5.2 to possibly –6 V.

With the two-level issues in mind, Fig. 72.13 illustrates the topology for a two-level OR/NOR gate. This design is similar to an AND gate except that $V_{out} = V_{OL}$ if both V_{inA} and V_{inB} are low. Otherwise, Vout = V_{OH}. By using the bottom differential pair to select one of the two top differential pairs, many other prime logic functions can be implemented. In Fig. 72.14, the top pairs are wired such that $V_{out} = V_{OL}$ if $V_{inA} = V_{inB}$, forming the XOR/XNOR block. If the top differential pairs are thought of as selectable



V_{inA}	V_{inB}	V_{out}
L	L	L
L	H	H
H	L	H
H	H	H

FIGURE 72.13 Two-level differential CML OR/NOR gate.

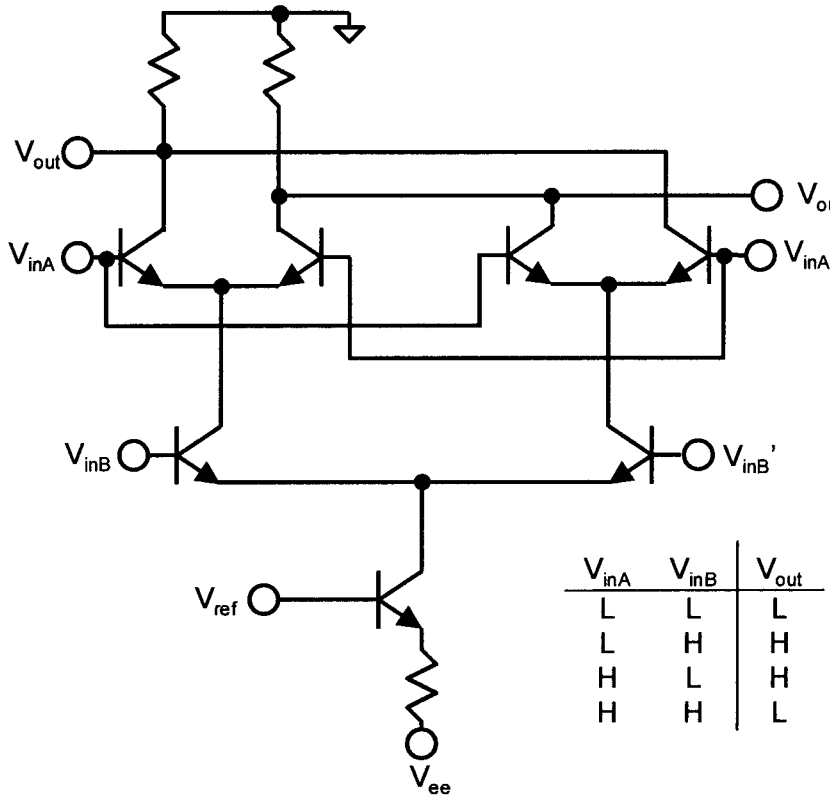


FIGURE 72.14 Two-level differential CML XOR gate.

buffers with a common output as shown in Fig. 72.15, then a basic 2:1 MUX cell is formed. Here, V_{inB}/V_{inB}' determines which input (V_{inA1}/V_{inA1}' or V_{inA2}/V_{inA2}') is selected to the output. This concept can be further extended to a 4:1 MUX if the top signals are CML and the control signals are ECL1 and ECL2, as shown in Fig. 72.16. Here, the MSB (ECL1) and LSB (ECL2) determine which of the four inputs are selected. With the 2:1 MUX in mind, if each top differential pair had separate output resistors with a common input, a 1:2 DEMUX is formed as shown in Fig. 72.17.

The last primary cell of importance is the latch. This is shown in Fig. 72.18. Here, the first differential pair is configured as a buffer. The second pair is configured as a buffer with positive feedback. The positive feedback causes any voltage difference between the input transistors to be amplified to full logic swing and that state is held as long as the bias is applied. With this in mind, as the first buffer is selected ($V_{inB} = V_{OH}$), the output is transparent to the input. As $V_{inB} = V_{OL}$, the last value stored in the buffer is held, forming a latch, which in this case, is triggered on the falling edge of the ECL2 level. When two of these blocks are connected together in series, it forms the basic master-slave flip-flop.

Advanced ECL/CML Logic Examples

With small signal amplifiers, the cascode configuration (common base on top of a common emitter stage) typically reduces the Miller capacitance for higher bandwidth. With the top-level transistor on, the top transistor forms a cascode stage with the lower differential amplifier. This can lead to higher bandwidths and reduced rise/falltimes. However, in large-signal logic, the bottom transistor must first turn on the top cascode stage before the output can change. This added delay results in a larger propagation delay from for the bottom pair vs. the top switching pair. In the case of the OR/NOR, as in Fig. 72.12, if Q_1 or Q_2 switches with Q_3 on or if Q_4 switches, the propagation delay is short. If Q_3 switches, it must first turn

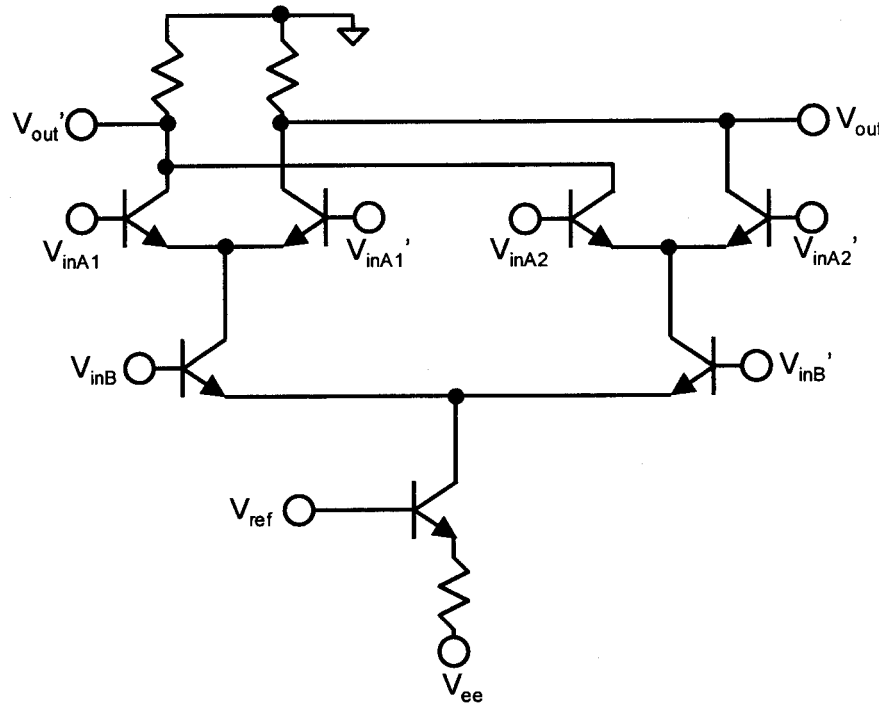


FIGURE 72.15 Two-level 2:1 differential CML MUX gate.

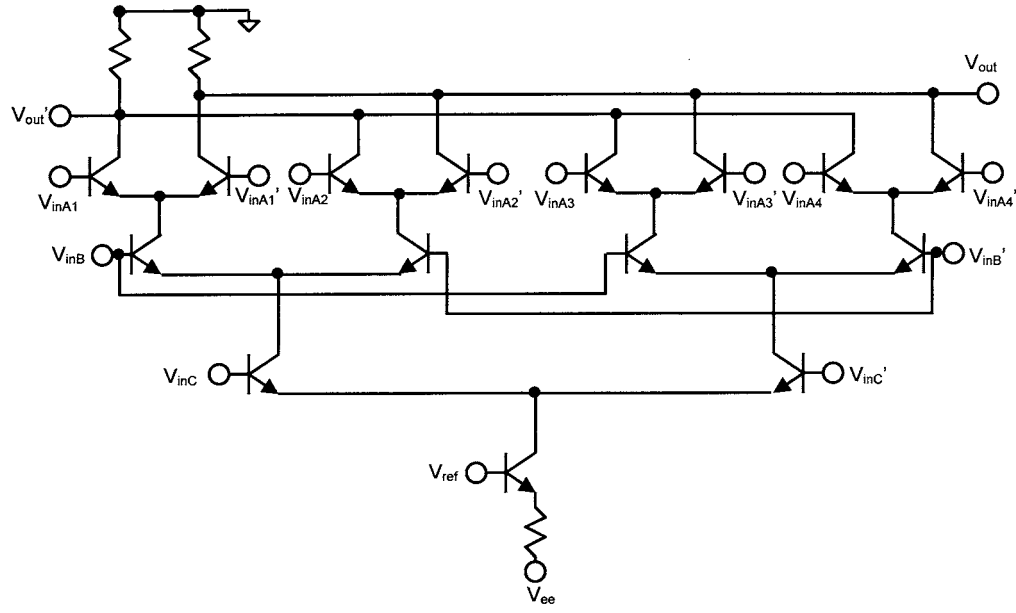


FIGURE 72.16 Three-level 4:1 differential CML MUX gate.

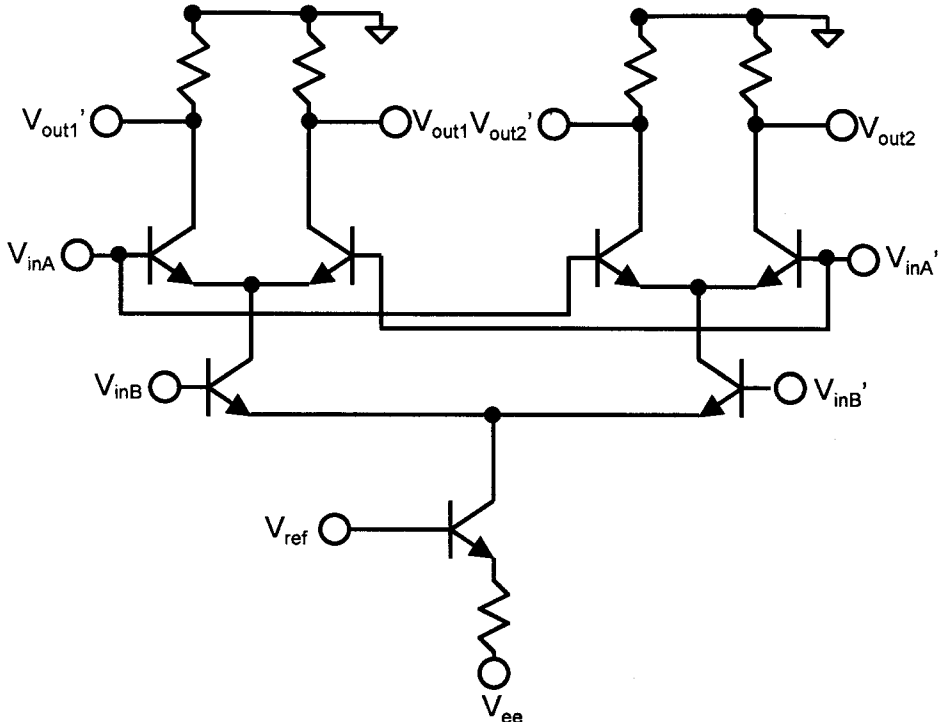


FIGURE 72.17 Two-level 1:2 differential CML DEMUX gate.

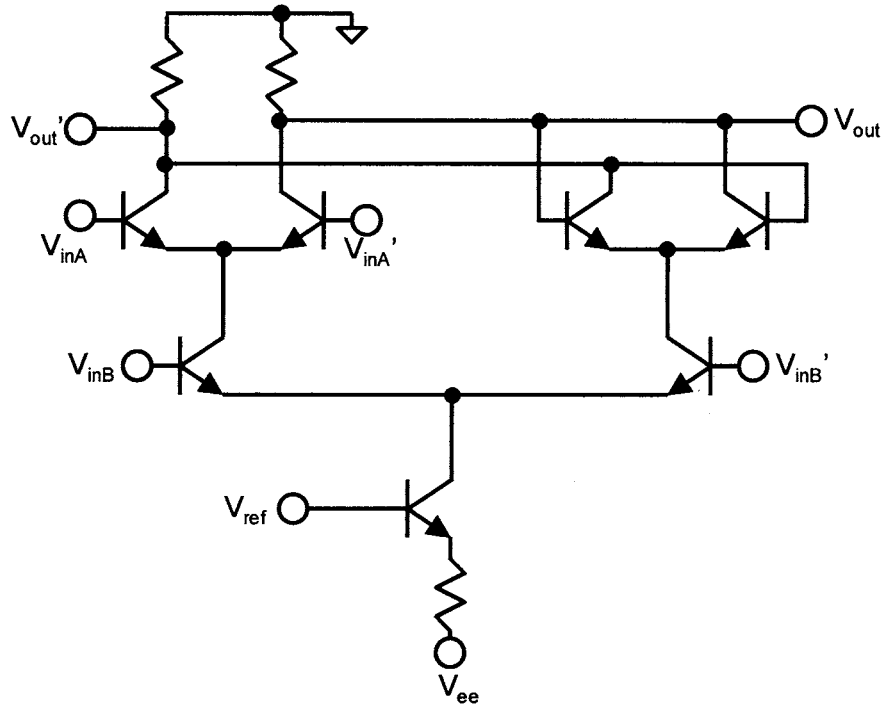
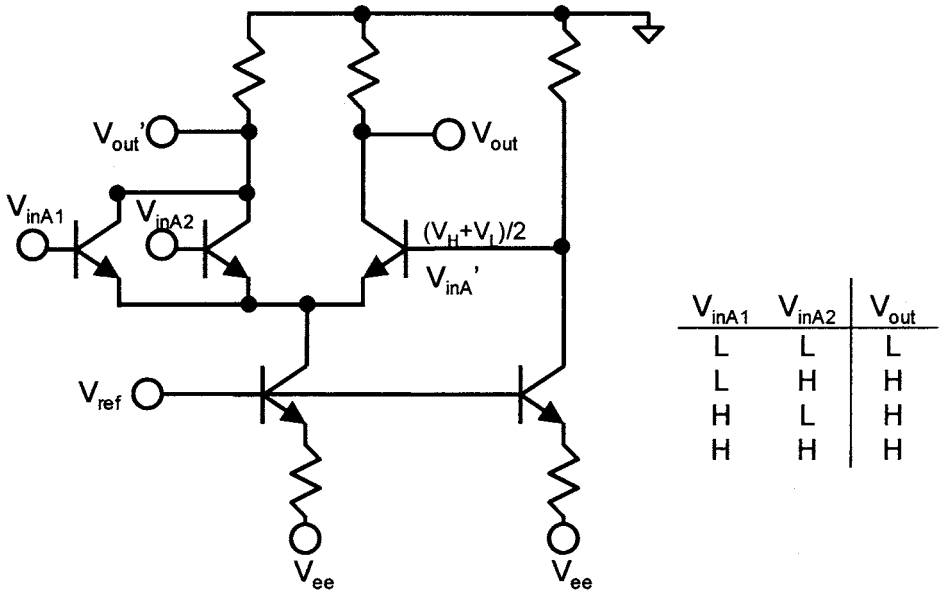


FIGURE 72.18 CML latch.



V_{inA1}	V_{inA2}	V_{out}
L	L	L
L	H	H
H	L	H
H	H	H

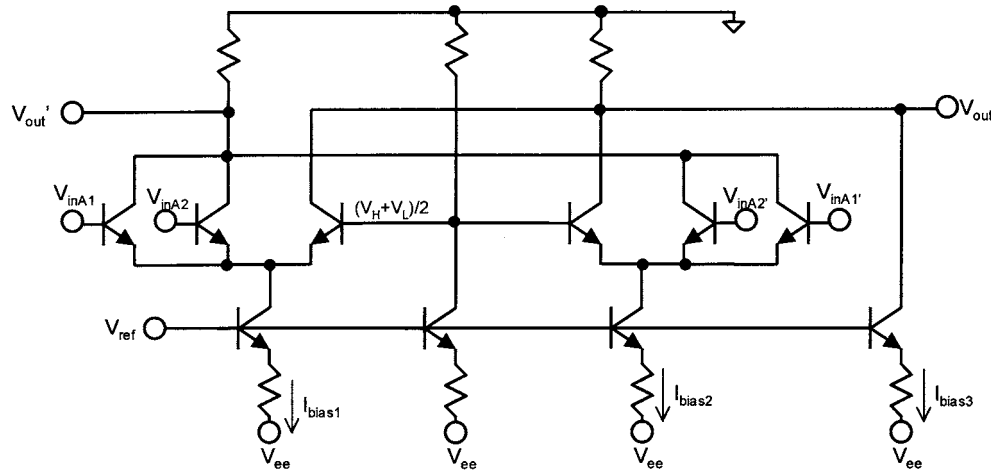
FIGURE 72.19 Single-level CML quasi-differential OR gate.

on either Q_1/Q_2, leading to the longest propagation delay. In this case, the longest delay limits the usable bandwidth of the AND gate. Likewise, in the XOR case, the delay of the top input is shorter than the lower input, which results in asymmetric behavior and reduced bandwidth. For a 10-GHz flip-flop, this can result in as much as a 10-ps delay from rising edge of the clock to the sample point of the data. This issue must be taken into account in determining the optimal input data phase for lowest bit errors when dealing with digital data.

One solution to the delay issue is to use a quasi-differential signal. In Fig. 72.19, a single-ended single-level OR/NOR gate is shown. Here, a reference generator of $(V_H + V_L)/2$ is applied to V_{inA}'. If either of the V_{inA1} or V_{inA2} is high, then $V_{out} = V_{OH}$. This design has more bandwidth than the two-level topology shown in Fig. 72.12, but some of the noise margin may be sacrificed.

Figure 72.20 shows an example of a single-level XOR gate with a similar input level reference. In this case, $I_{bias1} = I_{bias2} = I_{bias3}$. The additional I_{bias3} is used to make the output symmetric. Ignoring I_{bias3}, when V_{inA} is not equal to V_{inB}, I_{bias1} and I_{bias2} are used to force $V_{out}' = V_{OL}$. When $V_{inA} = V_{inB}$, $V_{out} = V_{out}'$ since both are lowered by I_{bias}, resulting in an indeterminate state. To remedy this, I_{bias3} is added to V_{out} to make the outputs symmetric. This design results in higher speed due to the single-level design; however, the noise margin is somewhat reduced due to the quasi-differential approach and the outputs have a common-mode voltage offset of $R_L I_{bias}$.

In a standard differential pair, the output load capacitance can be broken into three parts. The base-collector capacitance of the driving pair, the interconnect capacitance, and the input capacitance of the next stage. The interconnect capacitance is on the order of 5 to 25 fF for adjacent to nearby gates. The base-collector depletion capacitance is on the order of 25 fF. Assuming that the voltage gain is 5.5, the effective C_{bc} or Miller capacitance is about 140 fF. $C_{be,j}$, when the transistor is off, is typically less than 6 fF. The $C_{be,d}$ capacitance when the transistor is on is of the order of 50 to 200 fF. These rough numbers show that the Miller effect has a significant effect on the effective load capacitance. For the switching transistor, the Miller effect increases both the effective internal C_{bc} as well as the external load. In these situations, a cascode stage may result in higher bandwidth and sharper rise/falltimes with a slight increase in propagation delay. Figure 72.21 shows a CML gate with an added cascode stage. Due to the 400-mV logic swing, the cascode bases are connected to ground. For higher swings, the cascode bases can be biased to a more negative voltage to avoid saturation. The cascode requires that the input level be either



V_{inA1}	V_{inA2}	I_{out}	$I_{out'}$	V_{out}	$V_{out'}$
L	L	$2I_{bias}$	I_{bias}	L	H
L	H	I_{bias}	$2I_{bias}$	H	L
H	L	I_{bias}	$2I_{bias}$	H	L
H	H	$2I_{bias}$	I_{bias}	L	H

FIGURE 72.20 Single-level CML quasi-differential XOR gate.

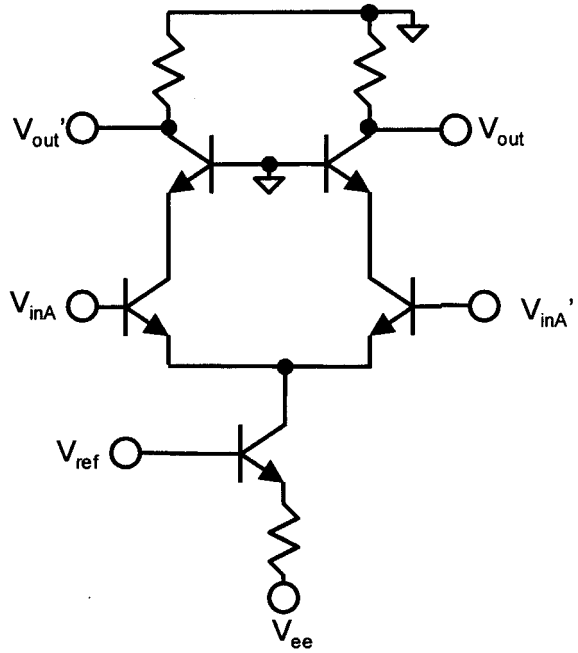


FIGURE 72.21 CML buffer with a cascode output stage.

ECL1 or ECL2 to account for the V_{be} drop of the cascode. Since the base of the cascode is held at ac ground, the Miller effect is not seen at the input of the common-base stage as the output voltage swings. From the common-emitter point of view, the collector swings only about 60 mV per decade change in I_c, thus the Miller effect is greatly reduced. The reduction of the Miller effect through cascoding reduces

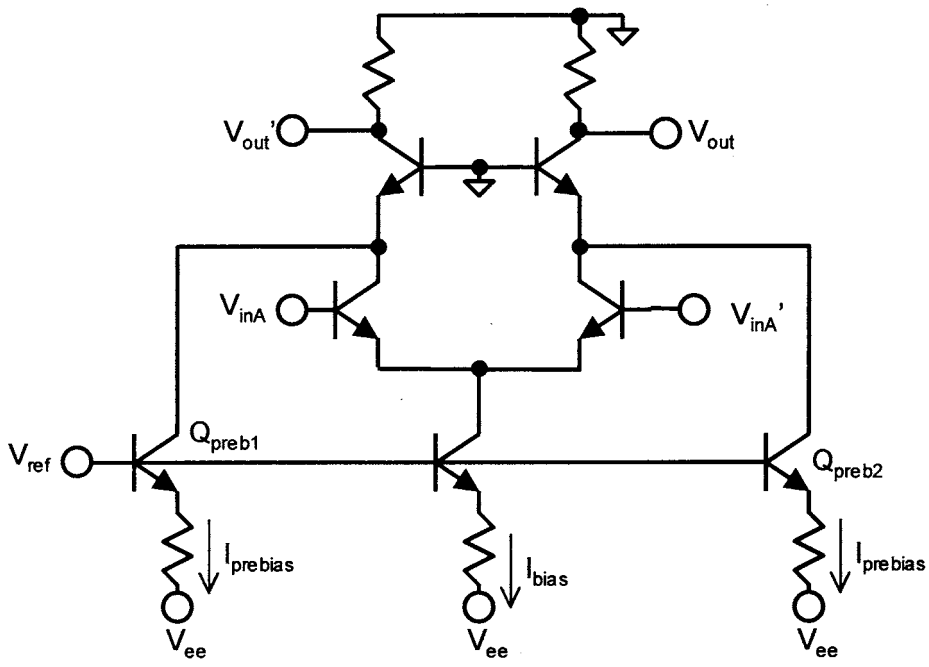


FIGURE 72.22 CML buffer with a cascode output stage and bleed current to keep the cascode "on".

the effect of both the internal transistor C_{bc} and load capacitance due to C_{bc} of the other stages, which results in the reduced rise/falltimes, especially at the logic transition region.

As both of the transistors in a cascode turn off, the increased charge stored in both transistors that has to discharge through an RC time constant may result in a slower edge rate near the logic high of a rising edge. In poorly designed cascode stages, the corner point between the fast-rising edge to the slower-rising edge may occur near the 20/80% point, canceling out some of the desired gains. Furthermore, with the emitter node of the off common-base stage floating in a high-impedance state, its actual voltage varies with time (large RC discharge compared to the switching time). This can result in some "memory" effects where the actual node voltage depends on the previous bit patterns. In this case, as the transistor is turns on, the initial voltage may vary, which can result in increased jitter with digital data. With these effects in mind, the cascoded CML design can be employed with performance advantages in carefully considered situations.

One way to remedy the off cascode issues is to use prebias circuits as shown in Fig. 72.22. Here, the current sources formed with Q_{preb1} and Q_{preb2} ($I_{prebias} \ll I_{bias}$) ensures that the cascode is always slightly on by bleeding a small bias current. This results in improvements in the overall rise- and falltimes, since the cascode does not completely turn off. This circuit does, however, introduce a common-mode offset in the output that may reduce the headroom in a two-level ECL gate that it must drive. Furthermore, a series resistor can be introduced between the bleed point and the current source to decouple the current source capacitance into the high-speed node. This design requires careful consideration to the design tradeoffs involving the ratio of $I_{bias}/I_{prebias}$ as well as the potential size of the cascode transistor vs. the switch transistors for optimal performance. When properly designed, the bleed cascode can lead to significant performance advantages.

In general, high-speed HBT circuits require careful consideration and design of each high-speed node with respect to the required level of performance, allowable power dissipation, and fan-out. The primary tools the designer has to work with are device bias, device size, ECL/CML gate topology, and logic level to optimize the design. Once the tradeoff is understood, CML/ECL HBT-based circuits have formed

some of the faster circuits to date. The performance and capability of HBT technology in circuit applications are summarized below.

HBT Circuit Design Examples

A traditional method to benchmark the high-speed capability of a technology is to determine the maximum switching rate of a static frequency divider. This basic building block is employed in a variety of high-speed circuits, which include frequency synthesizers, demultiplexers, and ADCs. The basic static frequency divider consists of a master/slave flip-flop where the output data of the slave flip-flop is fed back to the input data of the master flip-flop. The clock of the master and the clock of the slave flip-flop are connected together, as shown in Fig. 72.23. Due to the low transistor count and importance in many larger high-speed circuits, the frequency divider has emerged as the primary circuit used to demonstrate the high-speed potential of new technologies. As HBT started to achieve SSI capability in 1984, a frequency divider with a toggle rate of 8.5 GHz[6] was demonstrated. During the transition from research to pilot production in 1992, a research-based AlInAs/GaInAs HBT (f_t of 130 GHz and f_{max} of 90 GHz) was able to demonstrate a 39.5 GHz divide-by 4.[60] Recently, an advanced AlInAs/GaInAs HBT technology (f_t of 164 GHz and f_{max} of 800 GHz) demonstrated a static frequency divider operating at 60 GHz.[61] This HBT ECL-based design, to date, reports the fastest results for any semiconductor technology, which illustrates the potential of HBTs and ECL/CML circuit topology for high-speed circuits.

Besides high-speed operation, production GaAs HBTs have also achieved a high degree of integration for LSI circuits. For ADCs and DACs, the turn-on voltage of the transistor (V_{be}) is determined by material constants, thus there is significantly less threshold variation when compared to FET-based technologies. This enables the design of high-speed and accurate comparators. Furthermore, the high linearity characteristics of HBTs enable the design of wide dynamic range and high linearity sample-and-hold circuits. These paramount characteristics result in the dominance of GaAs HBTs in the super-high performance/high-speed ADCs. An 8-bit 2 gigasamples/s ADC has been fabricated with 2500 transistors. The input bandwidth is from dc to 1.5 GHz, with a spur-free dynamic range of about 48 dB.[62]

Another lucrative area for digital HBTs is in the area of high-speed circuits that are employed in fiber-optic based telecommunications systems. The essential circuit blocks (such as a 40-Gb/s 4:1 multiplexers[63]

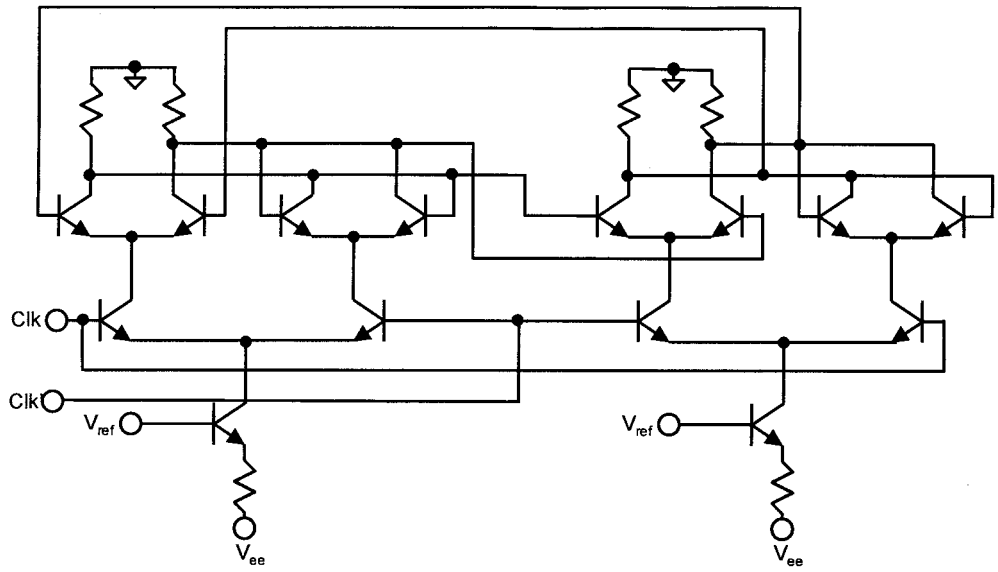


FIGURE 72.23 2:1 Frequency divider based on two CML latches (master/slave flip-flop).

and 26-GHz variable gain-limiting amplifiers[64]) have been demonstrated with HBTs in the research lab. In general, the system-level specifications (SONET) for telecommunication systems are typically very stringent compared with data communication applications at the same bit rate. The tighter specifications in telecom applications are due to the long-haul nature and the need to regenerate the data several times before the destination is reached. Today, there are many ICs that claim to be SONET-compliant at OC-48 (2.5 Gb/s) and some at OC-192 (10 Gb/s) bit rates. Since the SONET specifications apply on a system level, in truth, there are very few ICs having the performance margin over the SONET specification for use in real systems. Due the integration level, high-speed performance, and reliability of HBTs, some of the first OC-48 (2.5 Gb/s) and OC-192 (10 Gb/s) chip sets (e.g., preamplifiers, limiting amplifiers, clock and data recovery circuits, multiplexers, demultiplexers, and laser/modulator drivers) deployed are based on GaAs HBTs. A 16 × 16 OC-192 crosspoint switch has been fabricated with a production 50 GHz f_t and f_{max} process.[65] The LSI capability of HBT technology is showcased with this 9000 transistor switch on a 6730 × 6130 μm^2 chip. The high-speed performance is illustrated with a 10 Gb/s eye diagram shown in Fig. 72.24. With less than 3.1 ps of RMS jitter (with four channels running), this is the lowest jitter 10-Gb/s switch to date. At this time, only two 16 × 16 OC-192 switches have been demonstrated[65,66] and both were achieved with HBTs. With a throughput of 160,000 Mb/s, these HBT parts have the largest amount of aggregate data running through it of any IC technology.

In summary, III-V HBT technology is a viable high-speed circuit technology with mature levels of integration and reliability for real-world applications. Repeatedly, research labs have demonstrated the world's fastest benchmark circuits with HBTs with ECL/CML-based circuit topology. The production line has shown that current HBTs can achieve the both the integration and performance level required for high-performance analog, digital circuits, and hybrid circuits that operate in the high gigahertz range. Today, the commercial success of HBTs can be exemplified by that fact that HBT production lines ship several million HBT ICs every month and that several new HBT production lines are in the works. In the future, it is expected that advances in Si based technology will start to compete in the markets currently held by III-V technology; however, it is also expected that III-V technology will move on to address ever higher speed and performance issues to satisfy our insatiable demand for bandwidth.

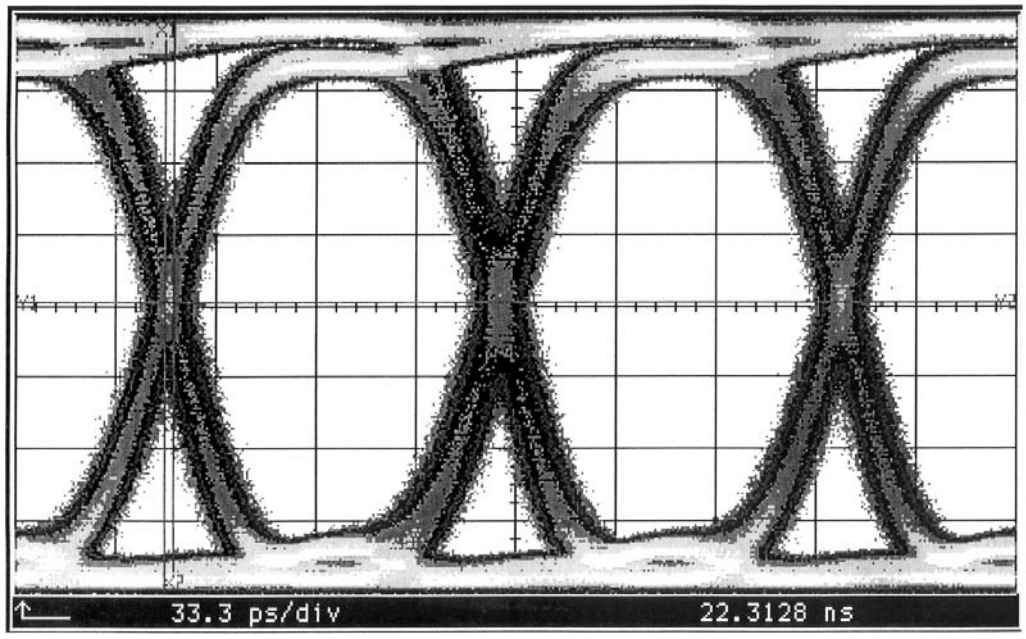


FIGURE 72.24 Typical 10 Gbps eye diagram for OC-192 crosspoint switch.

Section References

1. Ware, R., Higgins, W., O'Hearn, K., and Tiernan, M., Growth and Properties of Very Large Crystals of Semi-Insulating Gallium Arsenide, presented at *18th IEEE GaAs IC Symp.,* Orlando, FL, 54, 1996.
2. Abrokwah, J. K., Huang, J. H., Ooms, W., Shurboff, C., Hallmark, J. A., et al., A Manufacturable Complementary GaAs Process, presented at *IEEE GaAs IC Symposium,* San Jose, CA, 127, 1993.
3. Kroemer, H., Heterostructures for Everything: Device Principles of the 1980s?, *Japanese J. Appl. Phys.*, 20, 9, 1981.
4. Kroemer, H., Heterostructure Bipolar Transistors and Integrated Circuits, *Proc. IEEE*, 70, 13, 1982.
5. Matthews, J. W. and Blakeslee, A. E., Defects in Epitaxial Multilayers. III. Preparation of Almost Perfect Layers, *J. Crystal Growth*, 32, 265, 1976.
6. Matthews, J. W. and Blakeslee, A. E., Coherent Strain in Epitaxially Grown Films, *J. Crystal Growth*, 27, 118, 1974.
7. Johnson, E. O. and Rose, A., Simple General Analysis of Amplifier Devices with Emitter, Control, and Collector Functions, *Proceedings of the IRE*, 47, 407, 1959.
8. Cherry, E. M. and Hooper, D. E., *Amplifying Devices and Low-Pass Amplifier Design.*, John Wiley & Sons, New York, 1968, Chap. 2 and 5.
9. Beaufoy, R. and Sparkes, J. J., The Junction Transistor as a Charge-Controlled Device, *ATE Journal*, 13, 310, 1957.
10. Shockley, W., *Electrons and Holes in Semiconductors*, Van Nostrand, New York, 1950.
11. Ferry, D. K., *Semiconductors*, Macmillan, New York, 1991.
12. Lundstrom, M., *Fundamentals of Carrier Transport*, Addison-Wesley, Reading, MA, 1990.
13. Sze, S. M., *Physics of Semiconductor Devices*, second ed., John Wiley & Sons, New York, 1981.
14. Yang, E. S., *Fundamentals of Semiconductor Devices*, McGraw-Hill, New York, 1978, Chap. 7.
15. Hollis, M. A. and Murphy, R. A., Homogeneous Field-Effect Transistors, *High-Speed Semiconductor Devices*, Sze, S. M., Ed., Wiley-Interscience, New York, 1990.
16. Pearton, S. J. and Shah, N. J., Heterostructure Field-Effect Transistors, *High-Speed Semiconductor Devices*, S. M. Sze, Ed.: Wiley-Interscience, 1990, Chap. 5.
17. Muller, R. S. and Kamins, T. I., *Device Electronics for Integrated Circuits*, second ed., John Wiley & Sons, New York, 1986, Chap. 6 and 7.
18. Gray, P. E., Dewitt, D., Boothroyd, A. R., and Gibbons, J. F., *Physical Electronics and Circuit Models of Transistors*, John Wiley & Sons, New York, 1964, Chap. 7.
19. Asbeck, P. M., Bipolar Transistors, *High-Speed Semiconductor Devices*, S. M. Sze, Ed., John Wiley & Sons, New York, 1990, Chap. 6.
20. Low, T. S. et al., Migration from an AlGaAs to an InGaP Emitter HBT IC Process for Improved Reliability, presented at *IEEE GaAs IC Symposium Technical Digest*, Atlanta, GA, 153, 1998.
21. Jalali, B. and Pearson, S. J., *InP HBTs Growth, Processing and Applications*, Artech House, Boston, 1995.
22. Nguyen, L. G., Larson, L. E., and Mishra, U. K., Ultra-High-Speed Modulation-Doped Field-Effect Transistors: A Tutorial Review, *Proc. of IEEE*, 80, 494, 1992.
23. Brews, J. R., The Submicron MOSFET, *High-Speed Semiconductor Devices*, Sze, S. M., Ed., Wiley-Interscience, New York, 1990, Chap. 3.
24. Nguyen, L. D., Brown, A. S., Thompson, M. A., and Jelloian, L. M., 50-nm Self-Aligned-Gate Pseudomorphic AlInAs/GaInAs High Electron Mobility Transistors, *IEEE Trans. on Elect. Dev.*, 39, 2007, 1992.
25. Yuan, J.-R. and Svensson, C., High-Speed CMOS Circuit Technique, *IEEE Journal of Solid-State Circuits*, 24, 62, 1989.
26. Weste, N. H. E. and Eshraghian, K., *Principles of CMOS VLSI Design – A Systems Perspective*, second ed., Addison-Wesley, Reading, MA, 1993.
27. Rabaey, J. M., *Digital Integrated Circuits: A Design Perspective*, Prentice-Hall, New York, 1996.

28. Hill, C. F., Noise Margin and Noise Immunity in Logic Circuits, *Microelectronics*, 1, 16, 1968.
29. Long, S. and Butner, S., *Gallium Arsenide Digital Integrated Circuit Design,* McGraw-Hill, New York, 1990, Chap. 3.
30. Bakoglu, H. B., *Circuits, Interconnections, and Packaging,* Addison-Wesley, Reading, MA, 1990, Chap. 7.
31. Long, S. and Butner, S., *Gallium Arsenide Digital Integrated Circuit Design,* McGraw-Hill, New York, 1990, Chap. 5.
32. Elmore, W. C., The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers, *J. Appl Phys.*, 19, 55, 1948.
33. Ashar, K. G., The Method of Estimating Delay in Switching Circuits and the Fig. of Merit of a Switching Transistor, *IEEE Trans. on Elect. Dev.*, ED-11, 497, 1964.
34. Tien, P. K., Propagation Delay in High Speed Silicon Bipolar and GaAs HBT Digital Circuits, *Int. J. of High Speed Elect.*, 1, 101, 1990.
35. Lee, T. H., *The Design of CMOS Radio-Frequency Integrated Circuits,* Cambridge Univ. Press, Cambridge, U.K., 1998, Chap. 7.
36. Gray, P. E. and Searle, C. L., *Electronic Principles: Physics, Models, and Circuits,* John Wiley & Sons, New York, 1969, 531.
37. Gray, P. and Meyer, R., *Analysis and Design of Analog Integrated Circuits,* 3rd ed., John Wiley & Sons, New York, 1993, Chap. 7.
38. Millman, J. and Grabel, A., *Microelectronics,* second ed., ed. McGraw-Hill, New York, 1987, 482.
39. Hurtz, G. M., Applications and Technology of the Transferred-Substrate Schottky-Collector Heterojunction Bipolar Transistor, M.S. Thesis, University of California, Santa Barbara, 1995.
40. Gray, P. and Meyer, R., *Analysis and Design of Analog Integrated Circuits,* 3rd ed., John Wiley & Sons, New York, 1993, Chap. 3.
41. Long, S. and Butner, S., *Gallium Arsenide Digital Integrated Circuit Design,* McGraw-Hill, New York, 1990, 210.
42. Vitesse Semiconductor, *1998 Product Selection Guide,* 164, 1998.
43. Troutman, R. R., Subthreshold Design Considerations for Insulated Gate Field-Effect Transistors, *IEEE J. Solid-State Cir.*, SC-9, 55, 1974.
44. Lee, S. J. et al., Ultra-low Power, High-Speed GaAs 256 bit Static RAM, presented at *IEEE GaAs IC Symp.,* Phoenix, AZ, 1983, 74.
45. Long, S. and Butner, S., *Gallium Arsenide Digital Integrated Circuit Design,* McGraw-Hill, New York, 1990, Chap. 2.
46. Lassen, P. S., High-Speed GaAs Digital Integrated Circuits for Optical Communication Systems, Ph.D Dissertation, Tech. U. Denmark, Lyngby, Denmark, 1993.
47. Miyamoto, Y., Yoneyama, M., and Otsuji, T., 40-Gbit/s TDM Transmission Technologies Based on High-Speed ICs, presented at *IEEE GaAs IC Symp.,* Atlanta, GA, 51, 1998.
48. Otsuji, T. et al., 40 Gb/s IC's for Future Lightwave Communications Systems, *IEEE J. Solid State Cir.*, 32, 1363, 1997.
49. Otsuji, T. et al., A Super-Dynamic Flip-Flop Circuit for Broadband Applications up to 24 Gb/s Utilizing Production-Level 0.2 μm GaAs MESFETs, *IEEE J. Solid State Cir.*, 32, 1357, 1997.
50. Lang, M., Wang, Z. G., Thiede, A., Lienhart, H., Jakobus, T., et al., A Complete GaAs HEMT Single Chip Data Receiver for 40 Gbit/s Data Rates, presented at *IEEE GaAs IC Symposium,* Atlanta, GA, 55, 1998.
51. Ichioka, T., Tanaka, K., Saito, T., Nishi, S., and Akiyama, M., An Ultra-High Speed DCFL Dynamic Frequency Divider, presented at *IEEE 1989 Microwave and Millimeter-Wave Monolithic Circuits Symposium,* 61, 1989.
52. Thiede, A. et al., Digital Dynamic Frequency Dividers for Broad Band Application up to 60 GHz, presented at *IEEE GaAs IC Symposium,* San Jose, CA, 91, 1993.
53. Rocchi, M. and Gabillard, B., GaAs Digital Dynamic IC's for Applications up to 10 GHz, *IEEE J. Solid-State Cir.*, SC-18, 369, 1983.

54. Shikata, M., Tanaka, K., Inokuchi, K., Sano, Y., and Akiyama, M., An Ultra-High Speed GaAs DCFL Flip Flop – MCFF (Memory Cell type Flip Flop), presented at *IEEE GaAs IC Symp.,* Nashville, TN, 27, 1988.
55. Maeda, T., Numata, K., et al., A Novel High-Speed Low-Power Tri-state Driver Flip Flop (TD-FF) for Ultra-low Supply Voltage GaAs Heterojunction FET LSIs, presented at *IEEE GaAs IC Symp.,* San Jose, CA, 75, 1993.
56. Madden, C. J., Snook, D. R., Van Tuyl, R. L., Le, M. V., and Nguyen, L. D., A Novel 75 GHz InP HEMT Dynamic Divider, presented at *IEEE GaAs IC Symposium,* Orlando, FL, 137, 1996.
57. Maeda, T. et al., An Ultra-Low-Power Consumption High-Speed GaAs Quasi-Differential Switch Flip-Flop (QD-FF), *IEEE J. Solid-State Cir.*, 31, 1361, 1996.
58. Yuan, H. T., Shih, H. D., Delaney, J., and Fuller, C., The Development of Heterojunction Integrated Injection Logic, *IEEE Trans. on Elect. Dev.*, 36, 2083, 1989.
59. Johnson, R. A. et al., Comparison of Microwave Inductors Fabricated on Silicon-on-Sapphire and Bulk Silicon, *IEEE Microwave and Guided Wave Letters,* 6, 323, 1996.
60. Jensen, J., Hafizi, M., Stanchina, W., Metzger, R., and Rensch, D., 39.5 GHz Static Frequency Divider Implemented in AlInAs/GaInAs HBT Technology, presented at *IEEE GaAs IC Symposium,* Miami, FL, 103, 1992.
61. Lee, Q., Mensa, D., Guthrie, J., Jaganathan, S., Mathew, T. et al., 60 GHz Static Frequency Divider in Transferred-substrate HBT Technology, presented at *IEEE International Microwave Symposium,* Anaheim, CA, 1999.
62. Nary, K. R., Nubling, R., Beccue, S., Colleran, W. T. et al., An 8-bit, 2 Gigasample per Second Analog to Digital Converter, presented at *17th Annual IEEE GaAs IC Symposium,* San Diego, CA, 303, 1995.
63. Runge, K., Pierson, R. L., Zampardi, P. J., Thomas, P. B., Yu, J. et al., 40 Gbit/s AlGaAs/GaAs HBT 4:1 Multiplexer IC, *Electronics Letters,* 31, 876, 1995.
64. Yu, R., Beccue, S., Zampardi, P., Pierson, R., Petersen, A. et al., A Packaged Broadband Monolithic Variable Gain Amplifer Implmented in AlGaAs/GaAs HBT Technology, presented at *17th Annual IEEE GaAs IC Symposium,* San Diego, CA, 197, 1995.
65. Metzger, A. G., Chang, C. E., Campana, A. D., Pedrotti, K. D., Price, A. et al., A 10 Gb/s High Isolation 16×16 Crosspoint Switch, Implemented with AlGaAs/GaAs HBT's, to be published, 1999.
66. Lowe, K., A GaAs HBT 16×16 10 Gb/s/channel Cross-Point Switch, *IEEE J. Solid State Cir.*, 32, 1263, 1997.

Chung, M.J., Kim, H.
"Internet Based Micro-Electronic Design Automation (IMEDA) Framework"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

73

Internet-Based Micro-Electronic Design Automation (IMEDA) Framework

Moon Jung Chung
Michigan State University

Heechul Kim
Hankuk University of Foreign Studies

73.1 Introduction

As the complexity of VLSI systems continues to increase, the micro-electronic industry must possess an ability to reconfigure design and manufacturing resources and integrate design activities so that it can quickly adapt to the market changes and new technology. Gaining this ability imposes a two-fold challenge: (1) to coordinate design activities that are geographically separated and (2) to represent an immense amount of knowledge from various disciplines in a unified format. The Internet can provide the catalyst by abridging many design activities with the resources around the world not only to exchange information but also to communicate ideas and methodologies.

In this chapter, we present a collaborative engineering framework that coordinates distributed design activities through the Internet. Engineers can represent, exchange, and access the design knowledge and carry out design activities. The crux of the framework is the formal representation of process flow using the process grammar, which provides the theoretical foundation for representation, abstraction, manipulation, and execution of design processes. The abstraction of process representation provides mechanisms to represent hierarchical decomposition and alternative methods, which enable designers to manipulat e the process fl ow diag ram and se lect the best method. In the fram e work, the process inf orm at ion

is layered into separate specification and execution levels so that designers can capture processes and execute them dynamically. As the framework is being executed, a designer can be informed of the current status of design such as updating and tracing design changes and be able to handling exception. The framework can improve design productivity by accessing, reusing, and revising the previous process for a similar. The cockpit of our framework interfaces with engineers to perform design tasks and to negotiate design tradeoff. The framework has the capability to launch whiteboards that enable the engineers in a distributed environment to view the common process flows and data and to concurrently execute dynamic activities such as process refinement, selection of alternative process, and design reviews. The proposed framework has a provision for various browsers where the tasks and data used in one activity can be organized and retrieved later for other activities.

One of the predominant challenges for micro-electronic design is to handle the increased complexity of VLSI systems. At the turn of the century, it is expected that there will be 100 million transistors in a single chip with 0.1 micron features, which will require an even shorter design time (Spiller, 1997). This increase of chip complexity has given impetus to trends such as system on a chip, embedded system, and hardware/software co-design. To cope with this challenge, industry uses custom-off- the-shelf (COTS) components, relies on design reuse, and practices outsourcing design. In addition, design is highly modularized and carried out by many specialized teams in a geographically distributed environment. Multi-facets of design and manufacturing, such as manufacturability and low power, should be considered at the early stage of design. It is a major challenge to coordinate these design activities (Fair, 94). The difficulties are caused by due to the interdependencies among the activities, the delay in obtaining distant information, the inability to respond to errors and changes quickly, and general lack of communications. At the same time, the industry must contend with decreased expenditures on manufacturing facilities while maintaining rapid responses to market and technology changes.

To meet this challenge, the U.S. government has launched several programs. The Rapid Prototyping of Application Specific Signal Processor (RASSP) Program was initiated by the Department of Defense to bring about the timely design and manufacturing of signal processors. One of the main goals of the RASSP program was to provide an effective design environment to achieve a four-time improvement in the development cycle of digital systems (Chung, 1996). DARPA also initiated a program to develop and demonstrate key software elements for Integrated Product and Process Development (IPPD) and agile manufacturing applications. One of the foci of the earlier program was the development of infrastructure for distributed design and manufacturing. Recently, the program is continued to Rapid Design Exploration & Optimization (RaDEO) to support research, development, and demonstration of enabling technologies, tools, and infrastructure for the next generation of design environments for complex electro-mechanical systems. The design environment of RaDEO is planned to provide cognitive support to engineers by vastly improving their ability to explore, generate, track, store, and analyze design alternatives (Lyons, 1997).

The new information technologies, such as the Internet and mobile computing, are changing the way we communicate and conduct business. More and more design centers use PCs, and link them on the Internet/intranet. The web-based communication allows people to collaborate across space and time, between humans, humans and computers, and computers in a shared virtual world (Berners-Lee, 1994). This emerging technology holds the key to enhance design and manufacturing activities. The Internet can be used as the medium of a virtual environment where concepts and methodologies can be discussed, accessed, and improved by the participating engineers. Through the medium, resources and activities can be reorganized, reconfigured, and integrated by the participating organizations. This new paradigm certainly impacts the traditional means for designing and manufacturing a complex product. Using Java, programs can be implemented in a platform-independent way so that they can be executed in any machine with a Web browser. Common Object Request Broker Architecture (CORBA) (Yang, 1996) provides distributed services for tools to communicate through the Internet (Vogel). Designers may be able to execute remote tools through the Internet and see the visualization of design data (Erkes, 1996; Chan, 1998; Chung, 1998).

Even though the potential impact of this technology will be great on computer aided design, Electronic Design Automation (EDA) industry has been slow in adapting this new technology (Spiller, 1997). Until

recently, EDA frameworks used to be a collection of point tools. These complete suites of tools are integrated tightly by the framework using their proprietary technology. These frameworks have been suitable enough to carry out a routine task where the process of design is fixed. However, new tools appear constantly. To mix and match various tools outside of a particular framework is very difficult. Moreover, tools, expertise, and materials for design and manufacturing of a single system are dispersed geographically. Now we have reached the stage where a single tool or framework is not sufficient enough to handle the increasing complexity of a chip and emerging new technology. A new framework is necessary which is open and scalable. It must support collaborative design activities so that designers can add new tools to the framework, and interface them with other CAD systems. There are two key functions of the framework: (1) managing the process and (2) maintaining the relationship among many design representations. For design data management, refer to (Katz, 1987). In this chapter, we will focus on the process management aspect.

To cope with the complex process of VLSI system design, we need a higher level of viewing of a complete process, i.e., the abstraction of process by hiding all details that need not to be considered for the purpose at hand. As pointed out in National Institute of Standards and Technology reports (Schlenoff, 1996; Knutilla, 1998), a "unified process specification language" should have the following major requirements: abstraction, alternative task, complex groups of tasks, and complex sequences.

In this chapter we first review the functional requirements of the process management in VLSI system design. We then present the Internet-based Micro-Electronic Design Automation (IMEDA) System. IMEDA is a web-based collaborative engineering framework where engineers can represent, exchange, and access design knowledge and perform the design activities through the Internet. The crux of the framework is a formal representation of process flow using process grammar. Similar to the language grammar, production rules of the process grammar map tasks into admissible process flows (Baldwin, 1995a). The production rules allow a complex activity to be represented more concisely with a small number of high-level tasks. The process grammar provides the theoretical foundation for representation, abstraction, manipulation, and execution of design and manufacturing processes. It facilitates the communication at an appropriate level of complexity. The abstraction mechanism provides a natural way of browsing the process repository and facilitates process reuse and improvement. The strong theoretical foundation of our approach allows users to analyze and predict the behavior of a particular process. The cockpit of our framework interfaces with engineers to perform design tasks and to negotiate design tradeoff. The framework guides the designer in selecting tools and design methodologies, and it generates process configurations that provide optimal solutions with a given set of constraints. The just-in-time binding and the location transparency of tools maximize the utilization of company resources. The framework is equipped with whiteboards so that engineers in a distributed environment can view the common process flows and data and concurrently execute dynamic activities such as process refinement, selection of alternative processes, and design reviews. With the grammar, the framework gracefully handles exceptions and alternative productions. A layered approach is used to separate the specification of design process and execution parameters. One of the main advantages of this separation is freeing designers from the overspecification and graceful exception handling. The framework, implemented using Java, is open and extensible. New process, tools, and user-defined process knowledge and constraints can be added easily.

73.2 Functional Requirements of Framework

Design methodology is defined as a collection of principles and procedures employed in the design of engineering systems. Baldwin and Chung (Baldwin, 1995a) define design methodology management as selecting and executing methodologies so that the input specifications are transformed into desired output specifications. Kleinfeldt (1994), states that "design methodology management provides for the definition, presentation, execution, and control of design methodology in a flexible, configured way." Given a methodology, we can select a process or processes for that particular methodology.

Each design activity, whether big or small, can be treated as a task. A complex design task is hierarchically decomposed into simpler subtasks, and each subtask in turn may be further decomposed. Each

task can be considered as a transformation from input specification to output specification. The term *workflow* is used to represent the details of a process including its *structure in terms of all the required tasks and their interdependencies*. Some process may be ill-structured, and capturing it as a workflow may not be easy. Exceptions, conditional executions, and human involvement during the process make it difficult to model the process as a workflow.

There can be many different tools or alternative processes to accomplish a task. Thus, a design process requires design decisions such as selecting tools and processes as well as selecting appropriate design parameters. At a very high level of design, the input specifications and constraints are very general and may even be ill-structured. As we continue to decompose and perform the tasks based on design decisions, the output specifications are refined and the constraints on each task become more restrictive. When the output of a task does not meet certain requirements or constraints, a new process, tools, or parameters must be selected. Therefore, the design process is typically iterative and based on previous design experience. Design process is also a collaborative process, involving many different engineering activities and requiring the coordination among engineers, their activities, and the design results.

Until recently, it was the designer's responsibility to determine which tools to use and in what order to use them. However, managing the design process itself has become difficult, since each tool has its own capabilities and limitations. Moreover, new tools are developed and new processes are introduced continually. The situation is further aggravated because of incompatible assumptions and data formats between tools. To manage the process, we need a framework to monitor the process, carry out design tasks, support cooperative teamwork, and maintain the relationship among many design representations (Chiueh, 1990; Katz, 1987). The framework must support concurrent engineering activities by integrating various CAD tools and process and component libraries into a seamless environment. Figure 73.1 shows the RASSP enterprise system architecture (Welsh, 1995). It integrates tools, tool frameworks, and data management functions into an enterprise environment. The key functionality of the RASSP system is managing the RASSP design methodology by "process automation", that is, controlling CAD program execution through workflow.

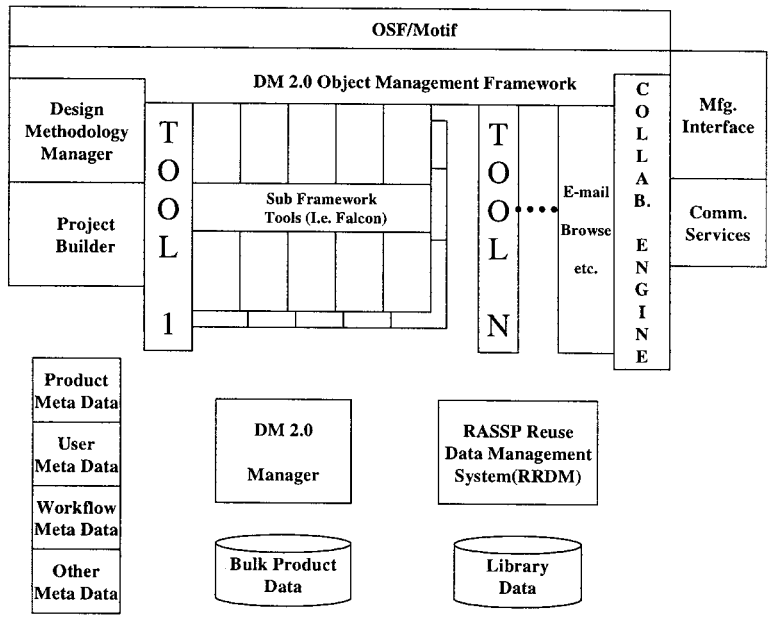


FIGURE 73.1 RASSP enterprise system architecture.

The Building Blocks of Process

The lowest level of a building block of a design process is a tool. A *tool* is an unbreakable unit of a CAD program. It usually performs a specific task by transforming given input specifications into output specifications. A *task* is defined as design activities that include information about what tools to use and how to use them. It can be decomposed into smaller subtasks. The simplest form of the task, called an *atomic task*, is the one that cannot be decomposed into subtasks. In essence, an atomic task is defined as an encapsulated tool. A task is called *logical* if it is not atomic. A workflow of a logical task describes the details of how the task is decomposed into subtasks, and the data and control dependencies such as the relationship between design data used in the subtasks. For a given task, there can be several workflows, each of which denotes a possible way of accomplishing the task. A *methodology* is a collection of workflow supported together with information on which workflow should be selected in a particular instance.

Functional Requirements of Workflow Management

To be effective, a framework must integrate many design automation tools and allow the designer to specify acceptable methodologies and tools together with information such as when and how they may be used. Such a framework must not only guide the designer in selecting tools and design methodologies, but also aid the designer in constructing a workflow that is suitable to complete the design under given constraints. The constructed workflow should guarantee that required steps are not skipped; built-in design checks are incorporated into the workflow. The framework must also keep the relationships between various design representations, maintain the consistency between designs and support cooperative teamwork, and allow the designer to interact with the system to adjust design parameters or to modify the previous design process. The framework must be extendible to accommodate rapidly changing technologies and emerging new tools. Such a framework can facilitate developing new hardware systems as well as redesigning a system from a previous design.

During a design process, a particular methodology or workflow selected by a designer must be based on available tools, resources (computing and human), and design data. For example, a company may impose a rule that if input is a VHDL behavioral description, then designers should use Model Technology's VHDL simulator, but if the input is Verlig, they must use ViewLogic simulator. Or, if a component uses Xilinx, then all other components must also use Xilinx. Methodology must be driven by local expertise and individual preference, which in turn, are based on the designer's experience.

The process management should not constrain the designer. Instead, it must free designers from routine tasks, and guide the execution of workflow. User interaction and a designer's freedom are especially important when exceptions are encountered during the execution of flows, or when designers are going to modify the workflow locally. The system must support such activities through "controlled interactions" with designers.

Process management can be divided into two parts:

- A formal specification of supported methodologies and tools that must show the tasks and data involved in a workflow and their relationships.
- An execution environment that helps designers to construct workflow and execute them.

Process Specification

Methodology management must provide facilities to specify design processes. Specification of processes involves tasks and their structures (i.e., workflow). The task involved and the flow of process, that is the way the process can be accomplished in terms of its subtasks, must be defined. Processes must be encapsulated and presented to designers in a usable way. Designers want an environment to guide them in building a workflow and to help them execute it during the design process. Designers must be able to browse related processes, and compare, analyze, and modify them.

Tasks

Designers should be able to define the tasks that can be logical or atomic, organize the defined tasks, and retrieve them. Task abstraction refers to using and viewing a task for specific purposes and ignoring the irrelevant aspects of the task. In general, object-oriented approaches are used for this purpose. Abstraction of the task may be accomplished by defining tasks in terms of "the operations the task is performing" without detailing the operations themselves. Abstraction of tasks allows users to clearly see the behavior of them and use them without knowing the details of their internal implementations. Using the generalization–specialization hierarchy (Chung, 1990), similar tasks can be grouped together. In the hierarchy, a node in the lower level inherits its attributes from its predecessors. By inheriting the behavior of a task, the program can be shared, and by inheriting the representation of a task (in terms of its flow), the structure (workflow) can be shared. The Process Handbook (Malone, in press) embodies concepts of specialization and decomposition to represent processes.

There are various approaches associated with binding a specific tool to an atomic task. A tool can be bound to a task statically at the compile time, or dynamically at the run time based on available resources and constraints. When a new tool is installed, designers should be able to modify the existing bindings. The simplest approach is to modify the source code or write a script file and recompile the system. The ideal case is plug and play, meaning that CAD vendors address the need of tool interoperability, e.g., the Tool Encapsulation Specification (TES) proposed by CFI (CFI, 1995).

Workflow

To define a workflow, we must specify the tasks involved in the workflow, data, and their relationship. A set of workflows defined by methodology developers enforces the user to follow the flows imposed by the company or group. Flows may also serve to guide users in developing their own flows. Designers would retrieve the cataloged flows, modify them, and use them for their own purposes based on the guidelines imposed by the developer. It is necessary to generate legal flows. A blackboard approach was used in (Lander, 1995) to generate a particular flow suitable for a given task. In Nelsis (Bosch, 1991), branches of a flow are explicitly represented using "or" nodes and "merge" nodes. A task can be accomplished in various ways. It is necessary to represent alternative methodologies for the task succinctly so that designers can access alternative methodologies and select the best one based on what-if analysis. IDEF3.X (IDEF) is used to graphically model workflow in RASSP environment. Figure 73.2 shows an example of workflow using IDEF3.X. A node denotes a task. It has inputs, outputs, mechanism, and conditions. IDEF definition that has been around for 20 years mainly to capture flat modeling such as a shop floor process. IDEF specification, however, requires complete information such as control mechanisms and scheduling at the specification time, making the captured process difficult to understand. In IDEF, "or" nodes are used to represent the alternative paths. It does not have an explicit mechanism to represent alternative workflow. IDEF is ideal only for documenting the current practice and not suitable for executing iterative process which is determined during the execution of the process. Perhaps, the most important aspect missing from most process management systems is the abstraction mechanism (Schlenoff, 1996).

Execution Environment

The execution environment provides dynamic execution of tasks and tools and binds data to tools, either manually or automatically. Few frameworks separate the execution environment from the specification of design process. There are several modes in which a task can be executed (Kleinfeldth, 1994): manual mode, manual execution of flow, automatic flow execution, and automatic flow generation. In manual flow execution, the environment executes a task in the context of a flow. In an automatic flow execution environment, tasks are executed based on the order specified on the flow graph. In automatic flow generation, the framework generates workflow dynamically and executes them without the guidance of designers. Many frameworks use blackboard- or knowledge-based approaches to generate workflow. However, it is important for designers to be able to analyze the workflow created and share it with others.

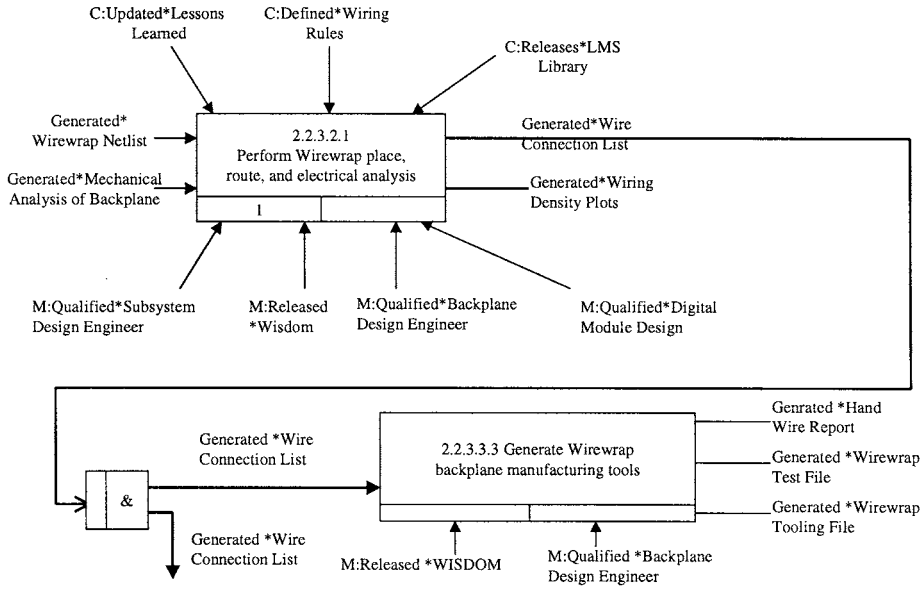


FIGURE 73.2 Workflow example using IDEF definition.

That is, repeatability and predictability are important factors if frameworks support dynamic creation of workflow.

Each task may be associated with pre- and post-conditions. Before a task is executed, the pre-condition of the task is evaluated. If the condition is not satisfied, the framework either waits until the condition is met, or aborts the task and selects another alternative. After the task is executed, its post-condition is evaluated to determine if the result meets the exit criteria. If the evaluation is unsatisfactory, another alternative should be tried.

When a task is complex involving many subtasks and each subtask in turn has many alternatives, generating a workflow for the task that would successfully accomplish the task is not easy. If the first try of an alternative is not successful, another alternative should be tried. In some cases, backtrack occurs which nullifies all the executions of previous workflow.

Literature Surveys

Many systems have been proposed to generate design process (Knapp, 1991) and manage workflow (Dellen, 1997; Lavana, 1997; Schurmann, 1997; Sutton, 1998). Many of them use the web technology to coordinate various activities in business (Andreoli, 1997), manufacturing (Berners-Lee, 1994; Cutkosy, 1996; Erkes, 1996), and micro-electronic design (Rastogi, 1993, Chan, 1998). WELD (Chan, 1998) is a network infrastructure for a distributed design system that offers users the ability to create a customizable and adaptable virtual design system that can couple tools, libraries, design, and validation services. It provides support not only for designing but also for manufacturing, consulting, component acquisition, and product distribution, encompassing the developments of companies, universities, and individuals throughout the world. Lavana et al. (1997) proposed an Internet-based collaborative design. They use Petri nets as a modeling tool for describing and executing workflow. User teams, at different sites, control the workflow execution by selection of its path. Minerva II (Sutton, 1998) is a software tool that provides design process management capabilities serving multiple designers working with multiple CAD frameworks. The proposed system generates design plan and realizes unified design process management across multiple CAD frameworks and potentially across multiple design disciplines. ExPro (Rastogi, 1993) is an expert-system-based process management system for the semiconductor design process.

There are several systems that automatically determine what tools to execute. OASIS (OASIS, 1992) uses Unix make file style to describe a set of rules for controlling individual design steps. The Design Planning Engine of the ADAM system (Knapp, 1986; Knapp, 1991) produces a plan graph using a forward chaining approach. Acceptable methodologies are specified by listing pre-conditions and post-conditions for each tool in a lisp-like language. Estimation programs are used to guide the chaining. Ulysses (Bushnell, 1986) and Cadweld (Daniel, 1991) are blackboard systems used to control design processes. A knowledge source, which encapsulates each tool, views the information on the blackboard and determines when the tool would be appropriate. The task management is integrated into the CAD framework and Task Model is interpreted by a blackboard architecture instead of a fixed inference mechanism. Minerva (Jacome, 1992) and the OCT task manager (Chiu, 1990) use hierarchical strategies for planning the design process. Hierarchical planning strategies take advantage of knowledge about how to perform abstract tasks which involve several subtasks.

To represent design process and workflow, many languages and schema have been proposed. NELSIS (Bosch, 1991) framework is based on a central, object-oriented database and on a flow management. It uses a dataflow graph as Flow Model and provides the hierarchical definition and execution of design flow. PLAYOUT (Schurmann, 1997) framework is based on separate Task and Flow Models which are highly interrelated among themselves and the Product Model. In (Barthelmann, 1996), graph grammar is proposed in defining the task of software process management. Westfechtel (1996) proposed "process-net" to generate the process flow dynamically. However, in many of these systems, the relationship between task and data is not explicitly represented. Therefore, representing the case in which a task generates more than one datum and each of them goes to a different task is not easy. In (Schurmann, 1997), Task Model (describing the I/O behavior of design tools) is used as a link between the Product Model and the Flow Model. The proposed system integrates data and process management to provide traceability. Many systems use IDEF to represent a process (Chung, 1996; IDEF; Stavas). IDEF specification, however, requires complete information such as control mechanisms and scheduling at the specification time, making the captured process difficult to understand.

Although there are many other systems that address the problem of managing process, most proposed system use either a rule-based approach or a hard-coded process flow. They frequently require source code modification for any change in process. Moreover, they do not have mathematical formalism. Without the formalism, it is difficult to handle the iterative nature of the engineering process and to simulate the causal effects of any changes in parameters and resources. Consequently, coordinating the dynamic nature of processes is not well supported in most systems. It is difficult to analyze the rationale how an output is generated and where a failure has occurred. They also lack a systematic way of generating all permissible process flows at any level of abstraction while providing means to hide the details of the flow when they are not needed. Most systems have the tendency to over-specify the flow information, requiring complete details of a process flow before executing the process. In most real situations, the complete flow information may not be known after the process has been executed: they are limited in their ability to address the underlying problem of process flexibility. They are rather rigid and not centered on users, and do not handle exceptions gracefully. Thus, the major functions for the collaborative framework such as adding new tools and sharing and improving the process flow cannot be realized. Most of them are weak in at least one of the following criteria suggested by NIST (Schlenoff et al., 1996): process abstraction, alternative tasks, complex groups of tasks, and complex sequences.

73.3 IMEDA System

The Internet-based Micro-Electronic Design Automation (IMEDA) System is a general management framework for performing various tasks in design and manufacturing of complex micro-electronic systems. It provides a means to integrate many specialized tools such as CAD and analysis packages, and allows the designer to specify acceptable methodologies and tools together with information such as when and how they may be used. IMEDA is a collaborative engineering framework that coordinates

design activities distributed geographically. The framework facilitates the flow of multimedia data sets representing design process, production, and management information among the organizational units of a virtual enterprise. IMEDA uses process grammar (Baldwin, 1995) to represent the dynamic behavior of the design and manufacturing process. In a sense, IMEDA is similar to agent-based approach such as Redux (Petrie, 1996). Redux, however, does not provide process abstraction mechanism or facility to display the process flow explicitly.

The major functionality of the framework includes

- Formal representation of the design process using the process grammar that captures a complex sequence of activities of micro-electronic design.
- Execution environment that selects a process, elaborates the process, invokes tools, pre- and post-evaluates the productions if the results meet the criterion, and notifies designers.
- User interface that allows designers to interact with the framework, guides the design process, and edits the process and productions.
- Tool integration and communication mechanism using Internet Socket and HTTP.
- Access control that provides a mechanism to secure the activity and notification and approval that provide the mechanisms to disperse design changes to, and responses from, subscribers

IMEDA is a distributed framework. design knowledge, including process information, manager programs, etc., are maintained in a distributed fashion by local servers. The following Fig. 73.3 illustrates how IMEDA links tools and sites for distributed design activities. The main components of IMEDA are

- **System Cockpit**: It controls all interactions between the user and the system and between the system components. The cockpit will be implemented as a Java applet and may be executable on any platform for which a Java enabled browser is available. It keeps track of the current design status and informs the user of possible actions. It allows users to collaboratively create and edit process flows, production libraries, and design data.
- **Manager Programs:** These encapsulate design knowledge. Using pre-evaluation functions, managers estimate the possibility of success for each alternative. They invoke tools and call post-evaluation functions to determine if a tool's output meets the specified requirements. The interface servers allow cockpits and other Java-coded programs to view and manipulate production, task

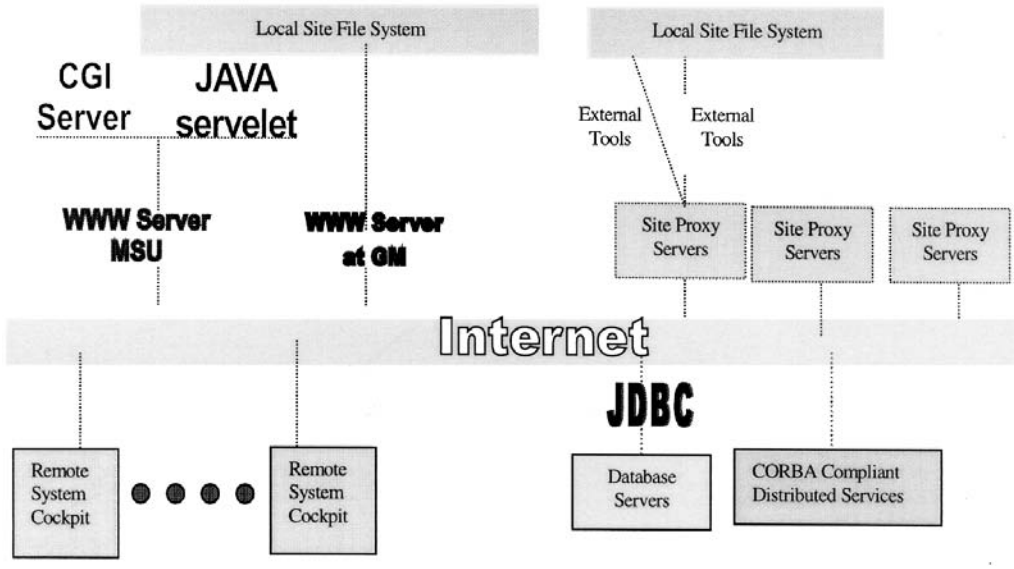


FIGURE 73.3 The architecture of IMEDA.

and design data libraries. Manager programs must be maintained by tool integrators to reflect site-specific information such as company design practices and different ways of installing tools.

- **Browsers:** The task browser organizes the tasks in a generalization-specialization (GS) hierarchy and contains all the productions available for each task. The data-specification browser organizes the data-specifications in a GS hierarchy and contains all the children.
- **External Tools**. These programs are the objects invoked by the framework during DM activities. Each atomic task in a process flow is bound to an external tool. External tools are written typically by the domain experts.
- **Site Proxy Server:** Any physical site that will host external tools must have a site proxy server running. These servers provide an interface between the cockpit and the external tools. The site server receives requests from system cockpits, and invokes the appropriate tool. Following the tool completion, the site server notifies the requesting cockpit, returning results, etc.
- **CGI Servers and Java Servlets:** The system cockpit may also access modules and services provided by CGI servers or the more recently introduced Java servlets. Currently, the system integrates modules of this type as direct components of the system (as opposed to external tools that may vary with the flow).
- **Database Servers:** Access to component data is a very important function. Using an API called JBDC, the framework can directly access virtually any commercially available database server remotely.
- **Whiteboard:** The shared cockpit, or "whiteboard" is a communication medium to share information among users in a distributed environment. It allows designers to interact with the system and guides the design process collaboratively. Designers will be able to examine design results and current process flows, post messages, and carry out design activities both concurrently and collaboratively. Three types of whiteboards are the process board, the chat board, and the freeform drawing board. Their functionality includes: (i) process board to the common process flow graph indicating the current task being executed and the intermediate results arrived at before the current task; (ii) drawing board to load visual design data, and to design and simulate process; and (iii) chat board to allow participants to communicate with each other via text-based dialog box.

IMEDA uses a methodology specification based on a process flow graphs and process grammars (Baldwin, 1995). Process grammars are the means for transforming high-level process flow graphs into progressively more detailed graphs by applying a set of substitution rules, called productions, to nodes that represent logical tasks. It provides not only the process aspect of design activities but also a mechanism to coordinate them. The formalism in process grammar facilitates abstraction mechanisms to represent hierarchical decomposition and alternative methods, which enable designers to manipulate the process flow diagram and select the best method. The formalism provides the theoretical foundations for the development of IMEDA.

IMEDA contains the database of admissible flows, called process specifications. With the initial task, constraints, and execution environment parameters, including personal profile, IMEDA guides designers in constructing process flow graphs in a top-down manner by applying productions. It also provides designers with the ability to discover process configurations that provide optimal solutions. It maintains consistency among designs and allows the designer to interact with the system and adjust design parameters, or modify the previous design process. As the framework is being executed, a designer can be informed of the current status of design such as updating and tracing design changes and be able to handling exception.

Real-world processes are typically very complex by their very nature; IMEDA provides designers the ability to analyze, organize, and optimize processes in a way never before possible. More importantly, the framework can improve design productivity by accessing, reusing, and revising the previous process for a similar design.

The unique features of our framework include

Process Abstraction/Modeling — Process grammars provide abstraction mechanism for modeling admissible process flows. The abstraction mechanism allows a complex activity to be represented

more concisely with a small number of higher-level tasks, providing a natural way of browsing the process repository. The strong theoretical foundation of our approach allows users to analyze and predict the behavior of a particular process. With the grammar, the process flow gracefully handles exceptions and alternative productions. When a task has alternative productions, backtracking occurs to select other productions.

Separation of Process Specification and Execution Environment — Execution environment information such as complex control parameters and constraints is hidden from the process specification. The information of these two layers is merely linked together to show the current task being executed on a process flow. The represented process flow can be executed in both automatic and manual modes. In the automatic mode, the framework executes all possible combinations to find a solution. In the manual mode, users can explore design space.

Communication and Collaboration — To promote real-time collaboration among participants, the framework is equipped with the whiteboard, a communication medium to share information. Users can browse related processes, compare them with other processes, analyze, and simulate them. Locally managed process flows and productions can be integrated by the framework in the central server. The framework manages the production rules governing the higher level tasks, while lower level tasks and their productions are managed by local servers. This permits the framework to be effective in orchestrating a large-scale activity.

Efficient Search of Design Process and Solution — IMEDA is able to select the best process and generate a process plan, or select a production dynamically and create a process flow. The process grammar easily captures design alternatives. The execution environment selects and executes the best one. If the selected process does not meet the requirement, then the framework backtracks and selects another alternative. This backtrack occurs recursively until a solution is found. If you allow a designer to select the best solution among many feasible ones, the framework may generate many multiple versions of the solution.

Process Simulation — The quality of a product depends on the tools (maturity, speed, and special strength of the tool), process (or workflow selected), and design data (selected from the reuse library). Our framework predicts the quality of results (product) and assesses the risk and reliability. This information can be used to select the best process/work flow suitable for a project.

Parallel Execution of Several Processes and Multiple Versions — To reduce the design time and risk, it is necessary to execute independent tasks in parallel whenever they are available. Sometimes, it is necessary to investigate several alternatives simultaneously to reduce the design time and risk. Or the designer may want to execute multiple versions with different design parameters. The key issue in this case is scheduling the tasks to optimize the resource requirements.

Life Cycle Support of Process Management — The process can be regarded as a product. A process (such as airplane designing or shipbuilding) may last many years. During this time, it may be necessary for the process itself to be modified because of new tools and technologies. Life cycle support includes updating the process dynamically, and testing/validating the design process, version history and configuration management of the design process. Tests and validations of the design processes, the simulation of processes, and impact analysis are necessary tools.

73.4 Formal Representation of Design Process[1]

IMEDA uses a methodology specification based on a process flow graphs and process grammars (Baldwin, 1995). The grammar is an extension of graph grammar originally proposed by Ehrig (1979) and has been applied to interconnection network (Derk, 1998) and software engineering (Heiman, 1997).

[1]Materials in this section are excerpted from R. Baldwin and M.J. Chung, *IEEE Computer,* Feb. 1995. With permission.

Process Flow Graph

A process flow graph depicts tasks, data, and the relationships among them, describing the sequence of tasks for an activity. Three basic symbols are used to represent a process flow graph. Oval nodes represent Logical Tasks, two-concentric oval nodes represent Atomic Tasks, rectangular nodes represent Data Specifications and diamond nodes represent Selectors. A task that can be decomposed into subtasks is called *logical*. Logical task nodes represent abstract tasks that could be done with several different tools or tool combinations. A task that cannot be decomposed is *atomic*. An atomic task node, commonly called a tool invocation, represents a run of an application program.

A *selector* is a task node that selects data or parameter. Data specifications are design data, where the output specification produced by a task can be consumed by another task as an input specification. Each data specification node, identified by a rectangle, is labeled with a data specification type. Using the graphical elements of the flow graph, engineers can create a process flow in a top-down fashion. These elements can be combined into a process flow graph using directed arcs. The result is a bipartite acyclic directed graph that identifies clearly the task and data flow relationships among the tasks in a design activity. The set of edges indicates those data specifications used and produced by each task. Each specification must have at most one incoming edge. Data specifications with no incoming edges are inputs of the design exercise. $T(G)$, $S(G)$, and $E(G)$ are the sets of task nodes, specification nodes, and edges of graph G, respectively. Figure 73.4 shows a process flow graph that describes a possible rapid

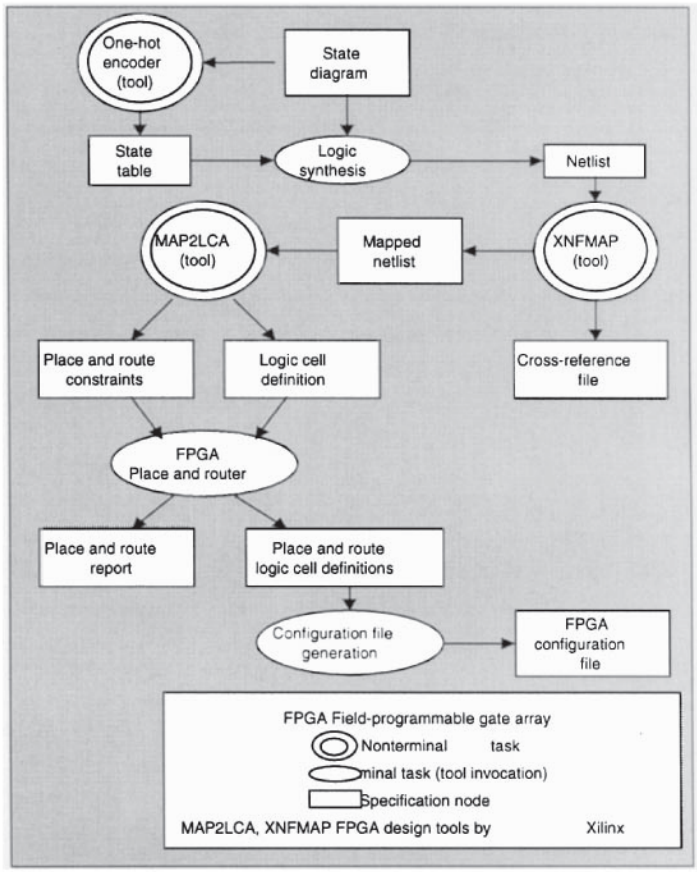


FIGURE 73.4 A sample process flow graph in which a state diagram is transformed into a field-programmable gate array configuration file.

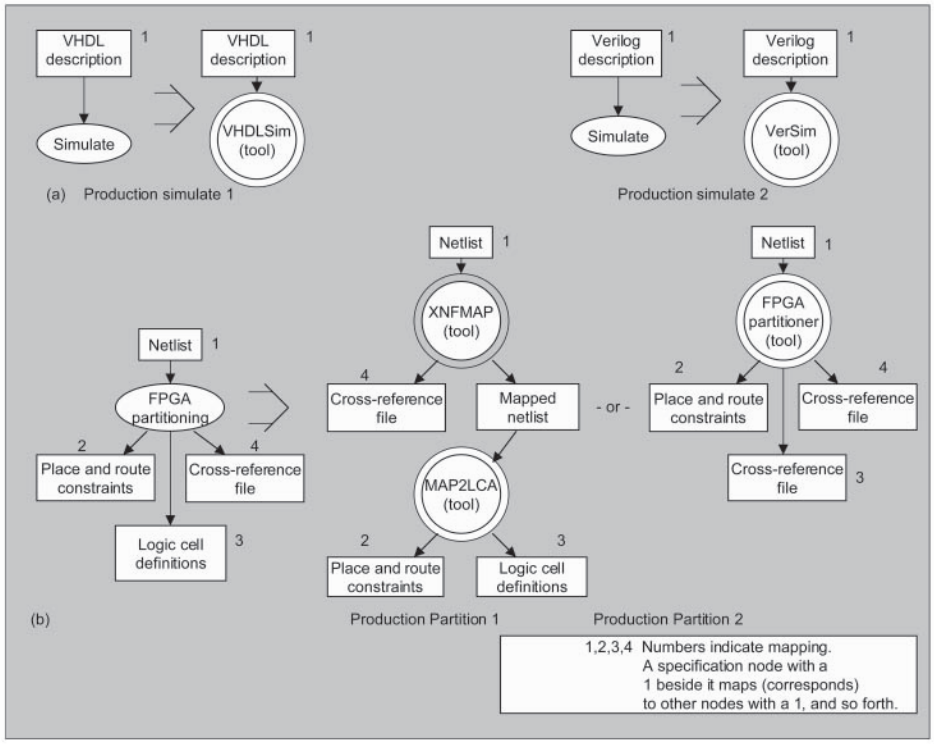


FIGURE 73.5 Graph production from a design process grammar. Two simulation alternatives based on input format are portrayed in (a); two partition alternatives representing different processes for an abstract task are protrayed in (b).

prototyping design process, in which a state diagram is transformed into a field-programmable gate array (FPGA) configuration file.

The various specification types form a class hierarchy where each child is a specialization of the parent. There may be several incompatible children. For example, VHDL and Verilog descriptions are both children of simulation models. We utilize these specification types to avoid data format incompatibilities between tools (see Fig. 73.5a). Process flow graphs can describe design processes to varying levels of detail. A graph containing many logical nodes abstractly describes what should be done without describing how it should be done (i.e., specifying which tools to use). Conversely, a graph in which all task nodes are atomic completely describes a methodology.

In our prototype, we use the following definitions: $In(N)$ is the set of input nodes of node N: $In(N) = \{ M \mid (M,N) \in E\}$. Out(N) is the set of output nodes of node N: $OUT(N) = \{ M \mid (N,M) \in E\}$. $I(G)$ is the set of input specifications of graph G: $\{ N \in S(G) \mid In(N) = \varnothing \}$.

Process Grammars

The designer specifies the overall objectives with the initial graph that lists available input specifications, desired output specifications, and the logical tasks to be performed. By means of process grammars, logical task nodes are replaced by the flows of detailed subtasks and intermediate specifications. The output specification nodes are also replaced by nodes that may have a child specification type.

The productions in a graph grammar permit the replacement of one subgraph by another. A production in a design process grammar can be expressed formally as a tuple $P = (G_{LHS}, G_{RHS}, \sigma_{in}, \sigma_{out})$, where G_{LHS} and G_{RHS} are process flow graphs for the left side and the right side of the production, respectively, such that (i) G_{LHS} has one logical task node representing the task to be replaced, (ii) σ_{in} is a mapping

from the input specifications $I(G_{LHS})$ to $I(G_{RHS})$, indicating the relationship between two input specifications (each input specification of $I(G_{RHS})$ *is* a subtype of $I(G_{LHS})$), and (iii) σ_{out} is a mapping from the output specifications of G_{LHS} to output specifications of G_{RHS} indicating the correspondence between them. (each output specification must be mapped to a specification with the same type or a subtype). Figure 73.5 illustrates productions for two tasks, `simulate` and `FPGA partitioning`. The mappings are indicated by the numbers beside the specification nodes. Alternative productions may be necessary to handle different input specification types (as in Fig. 73.5a), or because they represent different processes- separated by the word "or" — for performing the abstract task (as in Fig. 73.5b).

Let A be the logical task node in $G_{LHS,}$ and A' be a logical task node in the original process flow graph G such that A has the same task label as A'. The production rule P can be applied to A', which means that A' can be replaced with G_{RHS} only if each input and output specifications of A' matches to input and output specifications of G_{LHS}, respectively. If there are several production rules with the same left side flow graph, it implies that there are alternative production rules for the logical task. Formally, the production matches A' if

(i) A' has the same task label as A.
(ii) There is a mapping ρ_{in}, from $In(A)$ to $In(A')$, indicating how the inputs should be mapped. For all nodes $N \in In(A)$, $\rho_{in}(N)$ should have the same type as N or a subtype.
(iii) There is a mapping, ρ_{out}, from $Out(A')$ to $Out(A)$, indicating how the outputs should be mapped. For all nodes $N \in Out(A')$, $\rho_{out}(N)$ should have the same type as N or a subtype.

The mappings are used to determine how edges that connected the replaced subgraph to the remainder should be redirected to nodes in the new subgraph. Once a match is found in graph G, the production is applied as follows:

(i) Insert G_{RHS}–$I(G_{RHS})$ into G. The inputs of the replaced tasks are not replaced.
(ii) For every N in $I(G_{RHS})$ and edge (N,M) in G_{RHS}, add edge $(\rho_{in}(\sigma_{in}(N)),M)$ to G. That is to connect the inputs of A' to the new task nodes that will use them.
(iii) For every N in Out(A') and edge (N,M) in G, replace edge (N,M) with edge $(\sigma_{out}(\rho_{out}(N)),M)$. That is to connect the new output nodes to the tasks that will use them.
(iv) Remove A' and Out(A') from G, along with all edges incident on them.

Figure 73.6 illustrates a derivation in which the FPGA partitioning task is planned, using a production from Fig. 73.5b.

The process grammar provides mechanism of specifying alternative methods for a logical task. A high-level flow graph can then be decomposed into detailed flow graphs by applying *production rules* to a logical task. A *production rule* is a substitution that permits the replacement of a logical task node with a flow graph that represents a possible way of performing the task. The concept of applying productions to logical tasks is somewhat analogous to the idea of productions in traditional (i.e., non-graph) grammars. In this sense, logical tasks correspond to *logical symbols* in grammar, and atomic tasks correspond to *terminal symbols*.

73.5 Execution Environment of the Framework

Figure 73.7 illustrates the architecture of our proposed system, which applies the theory developed in the previous section. Decisions to select or invoke tools are split between the designers and a set of manager programs, where manager programs are making the routine decisions and the designers make decisions that requires higher-level thinking. A program called Cockpit coordinates the interaction among manager programs and the designers. Tool sets and methodology preferences will differ among sites and over time. Therefore, our assumption is that each unit designates a person (or group) to act as system integrator, who writes and maintains the tool-dependent code in the system. We provide the tool-independent code and template to simplify the task of writing tool-dependent code.

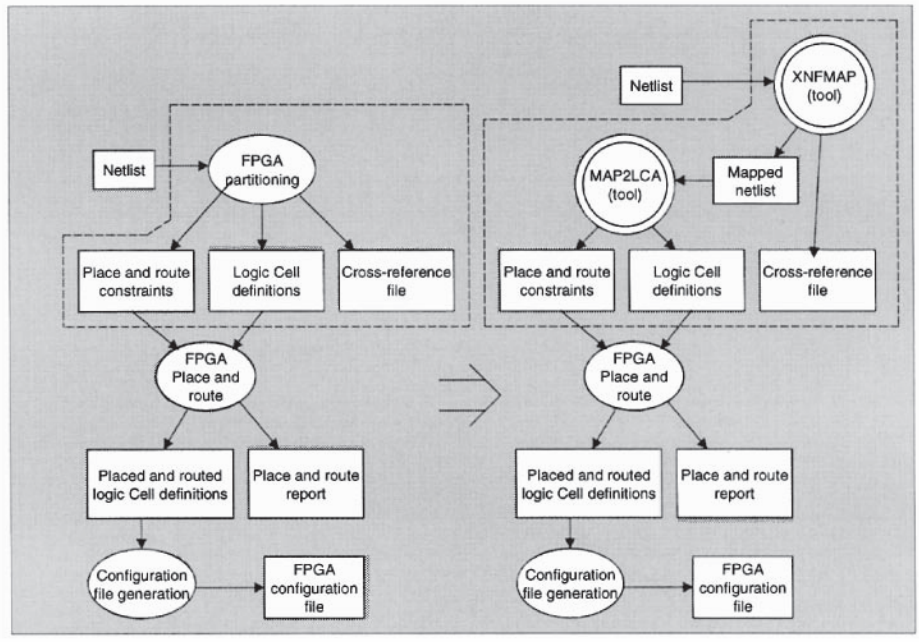


FIGURE 73.6 A sample graph dertivation. Nodes in the outlined region, left, are replaced with nodes in the outlined region, right, according to production partition 1 in Fig. 73.5.

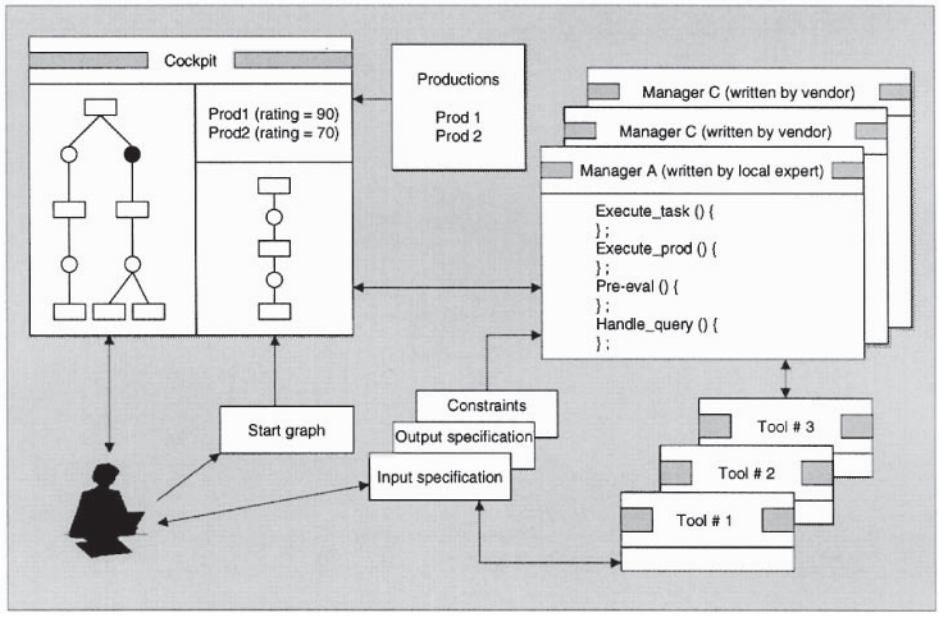


FIGURE 73.7 The proposed system based on Cockpit.

The Cockpit Program

The designer interacts with Cockpit, a program which keeps track of the current process flow graph and informs the designer of possible actions such as productions that could be applied or tasks that could be executed. Cockpit contains no task-specific knowledge; its information about the design process comes

entirely from a file of graph productions. When new tools are acquired or new design processes are developed, the system integrator modifies this file by adding, deleting, and editing productions.

To assist the designer in choosing an appropriate action, Cockpit interacts with several manager programs which encapsulate design knowledge. There are two types of manager programs: task managers and production managers. Task managers invoke tools and determine which productions to execute for logical task nodes. Production managers provide ratings for the productions and schedule the execution of tasks on the right-had side of the production. Managers communicate with each other using messages issued by Cockpit.

Our prototype system operates as follows. Cockpit reads the initial process flow graph from an input file generated by using a text editor. Cockpit then iteratively identifies when productions can be applied to logical task nodes and requests that the production managers assign the ratings to indicate how appropriate the productions are for those tasks. The process flow graph and the ratings of possible production applications are displayed for the designer, who directs Cockpit through a graphical user interface to apply a production or execute a task at any time. When asked to execute a task, Cockpit sends a message to a task manager. For an atomic task node, the task manager simply invokes the corresponding tool. For a logical task, the task manager must choose one or more productions, as identified by a Cockpit. The Cockpit applies the production and requests that the production manager executes it.

Manager Programs

Manager programs must be maintained by system integrators to reflect site-specific information, such as company design practices and tool installation methods. Typically, a manager program has its own thread. A Cockpit may have several manager programs, and therefore multi-threads. We define a communication protocol between Cockpit and manager programs and provide templates for manager programs. The manager programs provide five operations: pre-evaluation, tool invocation, logical task execution, production execution, and query handling. Each operation described below corresponds to a C++ or Java function in the templates, which system integrators can customize as needed.

Pre-evaluation: Production managers assign ratings to help designers and task managers select the most appropriate productions. The rating indicates the likelihood of success from applying this production. The strategies used by the system integrator provide most of the code to handle the rating. In some cases, it may be sufficient to assign ratings statically, based on the success of past productions. These static ratings can be adjusted downward when the production has already been tried unsuccessfully on this task node (which could be determined using the query mechanism). Alternatively, the ratings may be an arbitrarily complex function of parameters obtained through the query mechanism or by examining the input files. Sophisticated manager programs may continuously gather and analyze process metrics that indicate those conditions leading to success, adjust adjusting ratings accordingly.

Tool Invocation: Atomic task mangers must invoke the corresponding software tool when requested by Cockpit, then determine whether the tool completed successfully. In many cases, information may be predetermined and entered in a standard template, which uses the tool's result status to determine success. In other cases, the manager must determine tool parameters using task-specific knowledge or determine success by checking task-specific constraints. Either situation would require further customization of the manager program.

Logical Task Execution: Logical task managers for logical tasks must select productions to execute the logical task. Cockpit informs the task manager of available productions and their ratings. The task manager can either direct Cockpit to apply and execute one or more productions, or it can decide that none of the productions is worthwhile and report failure. The task manager can also request that the productions be reevaluated when new information has been generated that might influence the ratings, such as a production's failure. If a production succeeds, the task manager checks any constraints; if they are satisfied, it reports success.

Production Execution: Production managers execute each task on the right-hand side of the production at the appropriate time and possibly check constraints. If one of the tasks fails or a constraint is violated, backtracking can occur. The production manager can use task-specific knowledge to determine which tasks to repeat. If the production manager cannot handle the failure itself, it reports the failure to Cockpit, and the managers of higher level tasks and productions attempt to handle it.

Query Handling: Both production and task managers participate in the query mechanism. A production manager can send queries to its parent (the task manager for the logical task being performed) or to one of its children (a task manager of a subtask). Similarly, a task manager can send a query to its parent production manager or to one of its children (a production manager of the production it executed). The manager templates define C functions, which take string arguments, for sending these queries. System integrators call these functions but do not need to modify them. The manager templates also contain functions which are modified by system integrators for responding to queries. Common queries can be handled by template code; for example, a production manager can frequently ask its parent whether the production has already been attempted for that task and whether it succeeded. The manager template handles any unrecognized query from a child manager by forwarding it to the parent manager. Code must be added to handle queries for task-specific information such as the estimated circuit area or latency.

Execution Example

Now we describe a synthesis scenario that illustrates our prototype architecture in use. In this scenario, the objective is to design a controller from a state diagram, which will ultimately be done following the process flow graph in Fig. 73.4. There are performance and cost constraints on the design, and the requirement to produce a prototype quickly. The productions used are intended to be representative but not unique. For simplicity, we assume that a single designer is performing the design with, therefore, only one Cockpit.

The start graph for this scenario contains only the primary task, chip synthesis, and specification nodes for its inputs and outputs (like the graph in the left in Fig. 73.8). Cockpit tells us that the production of Fig. 73.8 can be applied. We ask Cockpit to apply it. The chip synthesis node is then replaced by nodes for state encoding, logic synthesis, and physical synthesis, along with intermediate specification nodes. Next, we want to plan the physical synthesis task. Tasks can be planned in an order other than they are to be performed. Cockpit determines that any of the productions shown in Fig. 73.9 may be applied, then queries each production's task manager program asking it to rate the production's appropriateness in the current situation. Based on the need to implement the design quickly, the productions for standard cell synthesis and full custom synthesis are rated low while the production for FPGA synthesis is rated high. Ratings are displayed to help us decide.

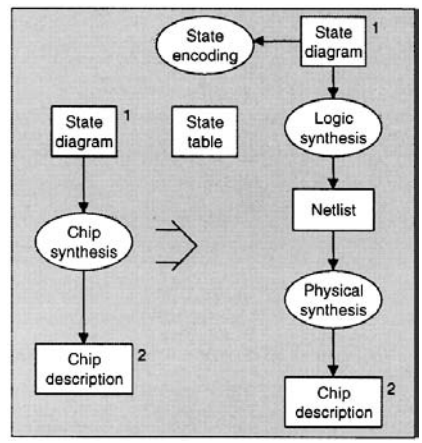


FIGURE 73.8 Productions for chip synthesis.

When we plan the state encoding task, Cockpit finds two productions: one to use the tool Min-bits Encoder and the other to use the tool One-hot Encoder. One-hot Encoder works well for FPGAs, while Minbits Encoder works better for other technologies. To assign proper ratings to these productions, their production managers must find out which implementation technology will be used. First, they send a query to their parent manager, the state encoding task manager. This manager forwards the message to

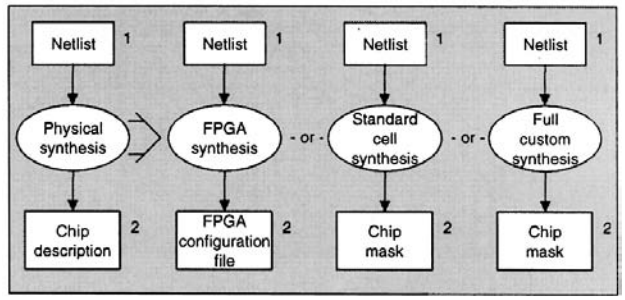


FIGURE 73.9 Productions for physical synthesis.

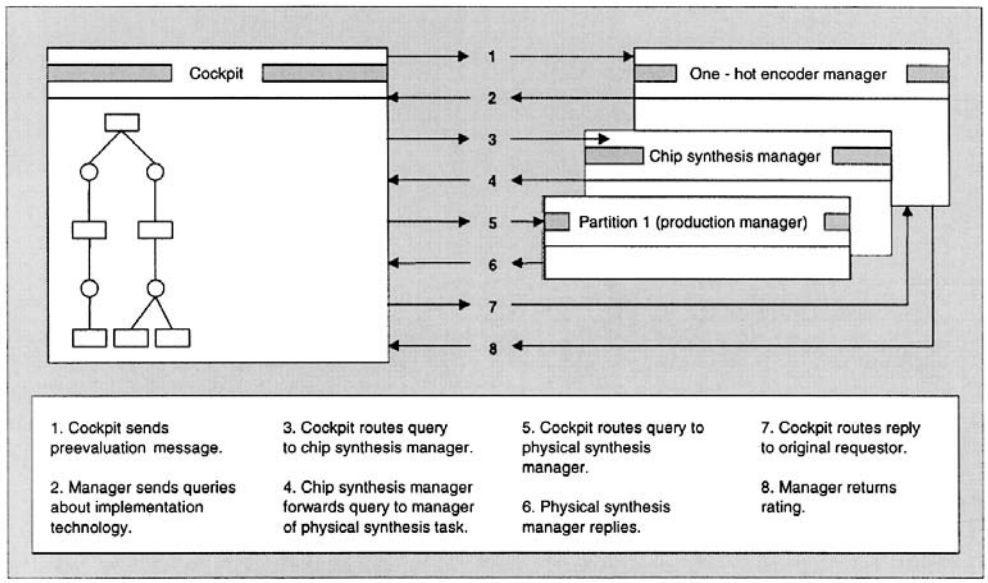


FIGURE 73.10 Sequence of actions for query handling.

its parent, the chip synthesis production manager. In turn, this manager forwards the query to the physical synthesis task manager for an answer. All messages are routed by Cockpit, which is aware of the entire task hierarchy. This sequence of actions is illustrated in Fig. 73.10.

After further planning and tool invocations, a netlist is produced for our controller. The next step is the FPGA synthesis task. We apply the production in Fig. 73.11 and proceed to the FPGA partitioning task. The knowledge to automate this task has already been encoded into the requisite manager programs, so we direct Cockpit to execute the FPGA partitioning task. It finds the two productions illustrated in Fig. 73.5b and requests their ratings, Next, Cockpit sends an execute message, along with the ratings, to the FPGA partitioning task manager. This manager's strategy is to always execute the highest-rated production, which in this case is production Partition 1. (Other task managers might have asked that both productions be executed or, if neither were promising, immediately reported failure.) This sequence of actions is shown in Fig. 73.12.

Because the Partition 1 manager used an as-soon-as-possible task scheduling strategy, it asks Cockpit to execute XNFMAP immediately. The other subtask, MAP2LCA, is executed when XNFMAP complete successfully. After both tasks complete successfully, Cockpit reports success to the FPGA partitioning task manager. This action sequence is illustrated in Fig. 73.13.

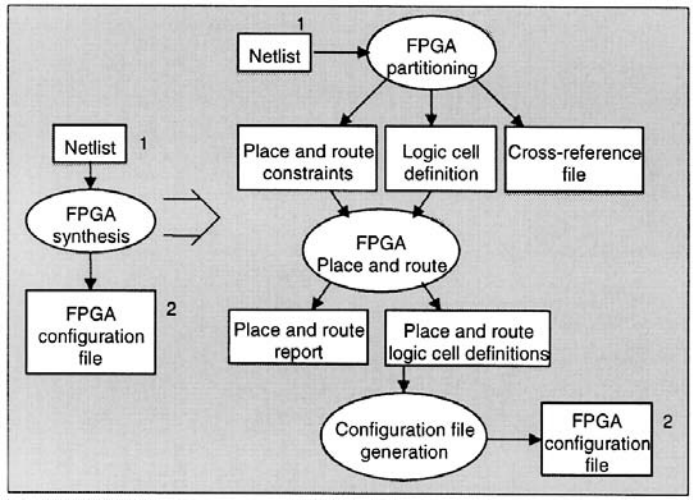


FIGURE 73.11 Production for field-programmable gate array synthesis.

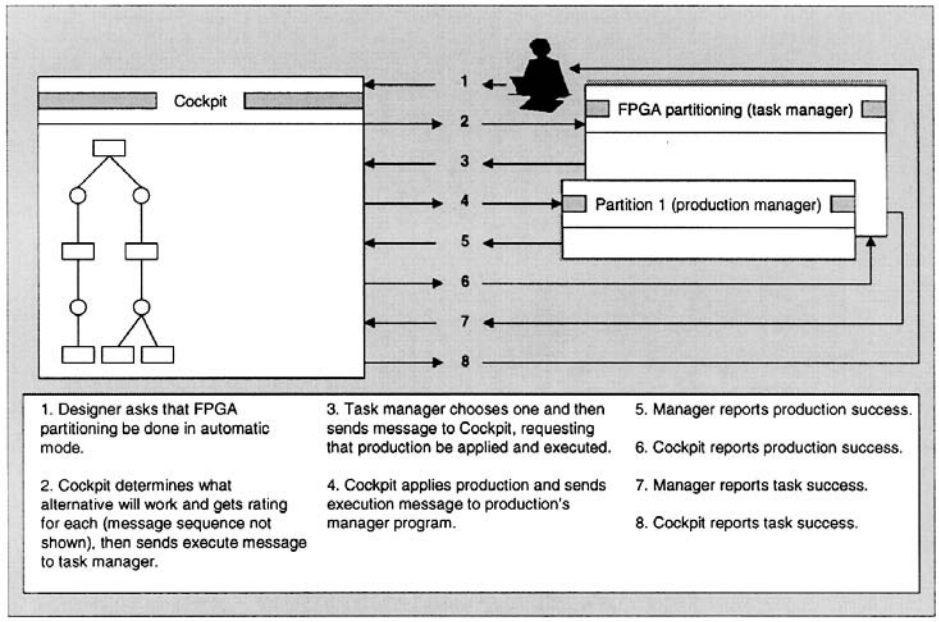


FIGURE 73.12 Sequence of action during automatic task execution.

Scheduling

In this subsection, we describe a detailed description and discussion of auto-mode scheduling, including the implementation of the *linear scheduler*. The ability to search through the configuration space of a design process for a design configuration that meets user-specified constraints is important. For example, assume that a user has defined a process for designing a digital filter with several different alternative ways of performing logical tasks such as "FPGA Partitioning" and "Select the Filter Architecture." One constraint that an engineer may wish to place on the design might be: "Find a process configuration that produces a filter that has maximum delay at most 10 nanoseconds." Given such a constraint, the framework must search through the configuration space of the filter design process, looking for a sequence of valid atomic

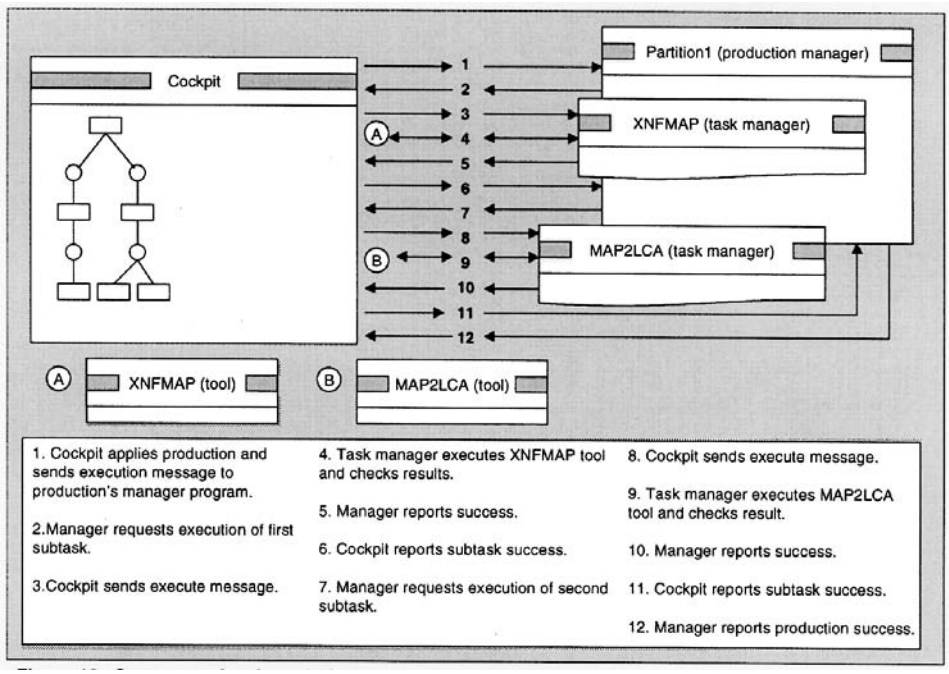


FIGURE 73.13 Sequence of actions during automatic production execution.

tasks that produces a filter with "maximum delay at most 10 nanoseconds." We call the framework component that performs this search a *scheduler*. There are, of course, many different ways of searching through the design process configuration space. In general, a successful scheduler will provide the following functionality:

- **Completeness (Identification of Successful Configurations):** Given a particular configuration of a process, the correct scheduler will be able to conclusively determine whether the configuration meets user-specified constraints. The scheduler must guarantee that before reporting failure, all possible process configurations have been considered, and if there is a successful configuration, the algorithm must find it.
- **Reasonable Performance:** The configuration space of a process grows exponentially (in the number of tasks). Ideally, a scheduler will be able to search the configuration space using an algorithm that requires less than exponential time.

The Linear Scheduling Algorithm is very simple yet complete and meets most of the above criteria. In this algorithm, for each process flow graph (corresponding to an initial process flow graph or a production), it has a scheduler. Each scheduler is a separate thread with a *Task Schedule List* (TSL) representing the order in which tasks are to be executed. The tasks in a scheduler's TSL are called its children tasks. A scheduler also has a *task pointer* to indicate the child task being executed in the TSL. The algorithm is recursive such that with each new instantiation of a production of a given task, a new scheduler is created to manage the flow graph representing the production selected. A liner scheduler creates a TSL by performing a topological sort of the initial process flow graph and executes its children tasks in order. If a child task is atomic, the scheduler executes the task without creating a new scheduler; otherwise, it selects a new alternative, creates a new child scheduler to manage the selected alternative, and waits for a signal from the child scheduler indicating success or failure. When a child task execution is successful, the scheduler increments the *task pointer* in its TSL and proceeds to execute the next task. If a scheduler reaches the end of its TSL, it signals success to its own parent, and awaits signals from its parent if it should terminate itself (all successful) or rollback (try another to find new configurations).

If a child task fails, the scheduler tries another alternative for the task. If there is no alternatives left, it rolls back (by decrementing the task pointer) until it finds a logical task that has another alternative to try. If a scheduler rolls back to the beginning of the TSL and cannot find an alternative, then its flow has failed. In this case, it signals a failure to its parent and terminates itself.

In the linear scheduling algorithm, each scheduler can send or receive any of five signals: `PROCEED`, `ROLLBACK`, `CHILD-SUCCESS`, `CHILD-FAILURE`, and `DIE`. These signals comprise scheduler-to-scheduler communication, including self-signaling. Each of the five signals is discussed below.

- **PROCEED:** This signal tells the scheduler to execute the next task in the `TSL`. It can be self-sent or received from a parent scheduler. For example, a scheduler increments its task pointer and sends itself a `PROCEED` signal when a child task succeeds, whereas it sends a `PROCEED` signal to its children to start its execution
- **ROLLBACK:** This is signaled when a task execution has failed. This signal may be self-sent or received from a parent scheduler. Scheduler self-signals `ROLLBACK` whenever a child task fails. A Rollback can result in either trying the next alternative of a logical task, or decrementing the task pointer and trying the previous task in the TSL. If rollback results in decrementing the task pointer to point to a child task node which has received a success-signal, the parent scheduler will send a rollback signal to that child task scheduler.
- **CHILD-SUCCESS:** A child scheduler sends a `CHILD-SUCCESS` to its parent scheduler if it has successfully completed the execution of all of the tasks in its `TSL`. After sending the child-success signal, the scheduler remains active, listening for possible rollback signals from the parent. After receiving a child-success signal, parent schedulers self-send a proceed signal.
- **CHILD-FAILURE:** A child-failure signal is sent from a child scheduler to its parent in the event that the child's managed flow fails. After sending a child-failure signal, children schedulers terminate. Upon receiving child-failure signals, parent scheduler self-send a rollback signal.
- **DIE:** This signal may be either self-sent, or sent from parent schedulers to their children schedulers.

73.6 Implementation

In this section, a high level description of the major components of IMEDA and their organization and functionality will be presented. Detailed explanations of the key concepts involved in the architecture of the Process Management Framework will also be discussed, including external tool integration, the tool invocation process, the Java File System, and state properties.

The System Cockpit

The System Cockpit, as its name suggests, is where nearly all user interaction with the framework takes place. It is here that users create, modify, save, load, and simulate process flow graphs representing design processes. This system component has been implemented as a Java applet. As such, it is possible to run the cockpit in any Java-enabled Web browser such as Netscape's Navigator or Microsoft's Internet Explorer. It is also possible to run the cockpit in some Java-enabled operating systems such as IBM's OS/2.

Each cockpit component also has the following components:

- **Root Flow.** Every cockpit has a Root Flow. The Root Flow is the flow currently being edited in the Cockpit's Flow Edit Panel. Notice that the Root Flow may change as a result of applying a production to a flow graph, in which case the Root Flow becomes a derivation of itself.
- **Flow Edit Panel.** The Flow Edit Panel is the interactive Graphical User Interface for creating and editing process flow graphs. This component also acts as a display for animating process simulations performed by various schedulers such as the manual or automode linear scheduler.
- **Class Directory.** The Cockpit has two Class Directories: Task Directory and the Specification Directory. These directories provide the "browser" capabilities of the framework, allowing users to create reusable general-to-specific hierarchies of task classes. Class Directories are implemented using a *tree* structure.

- **Production Database.** The Production Database acts as a warehouse for logical task productions. These productions document the alternative methods available for completing a logical task. Each Production Database has a list of Productions. The Production Database is implemented as a tree-like structure, with Productions being on the root trunk, and Alternatives being leaves.
- **Browser**. Browsers provide the tree-like graphical user interface for users to edit both Class Directories and Databases. There are three Browsers: Database Browser for accessing the Production Database, Directory Browser for accessing the Task Directory, and Directory Browser for accessing the Spec Browser. Both Database Browsers and Directory Browsers inherit properties from object Browser, and offer the user nearly identical editing environments and visual representations. This deliberate consolidation of Browser interfaces allowed us to provide designers with an interface that was consistent and easier to learn.
- **Menu.** A user typically performs and accesses most of the system's key function from the cockpit's Menu.
- **Scheduler.** The cockpit has one or more schedulers. Schedulers are responsible for searching the configuration space of a design process for configurations that meet user specified design constraints. The Scheduler animates its process simulations by displaying them in the Flow Edit Panel of the Cockpit.

External Tools

External Tools are the concrete entities to which atomic tasks from a production flow are bound. When a flow task object is expanded in the Cockpit Applet (during process simulation), the corresponding external tool is invoked. The external tool uses a series of inputs and produces a series of outputs (contained in files). These inputs and outputs are similarly bound to specifications in a production flow. Outputs from one tool are typically used as inputs for another. IMEDA can handle the transfer of input and output files between remote sites. The site proxy servers, in conjunction with a remote file server (also running at each site) automatically handle the transfer of files from one system to another. External tools may be implemented using any language, and on any platform that has the capability of running a site server. While performing benchmark tests of IMEDA, we used external tools written in C, Fortran, Perl, csh (a Unix shell script), Java applications, and Mathematica scripts.

External Tool Integration

One of the primary functionality of IMEDA is the integration of user-defined external tools into an abstract process flow. IMEDA then uses these tools both in simulating the process flow to find a flow configuration that meets specific constraints, and in managing selected flow configurations during actual design execution.

There are two steps to integrating tools with a process flow defined in IMEDA: *association* and *execution.* Association involves "linking" or "binding" an abstract flow item (e.g., an atomic task) to an external tool. Execution describes the various steps that IMEDA takes to actually invoke the external tool and process the results.

Binding Tools

External tools may be bound to three types of flow objects: Atomic Tasks, Selectors, and Multiple Version Selectors. Binding an external tool to a flow object is a simple and straightforward job, involving simply defining certain properties in the flow object.

The following properties must be defined in an object that is to be bound to an external tool:

- **SITE.** Due to the fact that IMEDA can execute tools on remote systems, it is necessary to specify the site where the tool is located on. Typically, a default SITE will be specified in the system defaults, and making it unnecessary to define the site property unless the default is to be overridden. Note that the actual site ID specified by the SITE property must refer to a site that is running a Site Proxy Server listening on that ID. See the "Executing External Tools" section below for more details.

- **CMDLINE.** The CMDLINE property specifies the command to be executed at the specified remote site. The CMDLINE property should include any switches or arguments that will always be sent to the external tool. Basically, the CMDLINE argument should be in the same format that would be used if the command were executed from a shell/DOS prompt.
- **WORKDIR.** The working directory of the tool is specified by the WORKDIR property. This is the directory in which IMEDA will actually execute the external tool, create temporary files, etc. This property is also quite often defined in the global system defaults, and thus may not necessarily have to be defined for every tool.
- **WRAPPERPATH.** The JDK 1.0.2 does not allow Java Applications to execute a tool in an arbitrary directory. To handle remote tool execution, a wrapper is provided. It is a "go-between" program that would simply change directories and then execute the external tool. This program can be as simple as a DOS/NT batch file, a shell script, or a perl program. The external tool is wrapped in this simple script, and executed. Since IMEDA can execute tools at remote and heterogeneous sites, it was very difficult to create a single wrapper that would work on all platforms (WIN32, Unix, etc.). Therefor, the wrapper program may be specified for each tool, defined as global default, or a combination of the two.

Once the properties above have been defined for a flow object, the object is said to be "bound" to an external tool. If no site, directory, or filename is specified for the outputs of the flow object, IMEDA automatically creates unique file names, and stores the files in the working directory of the tool on the site that the tool was run. If a tool uses as inputs data items that are not specified by any other task, then the data items must be bound to static files on some site.

Executing External Tools

Once flow objects have been bound to the appropriate external tools, IMEDA can be used to perform process simulation or process management. IMEDA actually has several "layers" that lie between the Cockpit (a Java applet) and the external tool that is bound to a flow being viewed by a user in the Cockpit. A description of each of IMEDA components for tool invocations is listed below.

- **Tool Proxy.** The tool proxy component acts as a liaison between flow objects defined in Cockpits and the Site Proxy Server. All communication is done transparently through the communication server utilizing TCP/IP sockets. The tool proxy "packages" information from Cockpit objects (atomic tasks, selectors, etc.) into string messages that the Proxy Server will recognize. It also listens for and processes messages from the Proxy Server (through the communications server) and relays the information back to the Cockpit object that instantiated the tool proxy originally.
- **Communications Server.** Due to various security restrictions in the 1.0.2 version of Sun Microsystem's Java Development Kit (JDK), it is impossible to create TCP/IP socket connections between a Java applet and any IP address other than the address from which the applet was loaded. Therefore, it was necessary to create a "relay server" in order to allow cockpit applets to communicate with remote site proxy servers. The sole purpose of the communications server is to receive messages from one source and then to rebroadcast them to all parties that are connected and listening on the same channel.
- **Site Proxy Server.** Site Proxy Servers are responsible for receiving and processing invocation requests from tool proxies. When an invocation request is received, the site proxy server checks to see that the request is formatted correctly, starts a tool monitor to manage the external tool invocation, and returns the exit status of the external tool after it has completed.
- **Tool Monitors.** When the site proxy server receives an invocation request and invokes an external tool, it may take a significant amount of time for the tool to complete. If the proxy server had to delay the handling of other requests while waiting for each external tool to complete, IMEDA would become very inefficient. For this reason, the proxy server spawns a tool monitor for each external tool that is to be executed. The tool monitor runs as a separate thread, waiting on the tool, storing its stdout and stderr, and moving any input or output files that need moving to their appropriate

site locations, and notifying the calling site proxy server when the tool has completed. This allows the site proxy server to continue receiving and processing invocation requests in a timely manner.

- **Tool Wrapper.** Tool wrapper changes directories into the specified `WORKDIR`, and then executes the `CMDLINE`.
- **External Tool.** External tools are the actual executable programs that run during a tool invocation. There is very little restriction on the nature of the external tools.

Communications Model

The Communications Model of IMEDA is perhaps the most complex portion of the system in some respects. This is where truly *distributed* communications come into play. One system component is communicating with another via network messages rather than function calls.

The heart of the communications model is the Communications Server. This server is implemented as a broadcast server. All incoming messages to the server are simply broadcast to all other connected parties. FlowObjects communicate with the Communications Server via ToolProxys. A ToolProxy allows a FlowObject to abstract all network communications and focus on the functionality of invoking tasks. A ToolProxy takes care of constructing a network message to invoke an external tool. That message is then sent to the Communications Server via a Communications Client. The Communication Client takes care of the low-level socket based communication complexities. Finally, the Communications Client sends the message to the Communications Server, which broadcasts the message to all connected clients. The client for which the message was intended (typically a Site Proxy Server) decodes the message and, depending on its type, creates either a ToolMonitor (for an Invocation Message) or an External Redraw Monitor (for a Redraw Request).

The Site Proxy Server creates these monitors to track the execution of external programs, rather than monitoring them itself. In this way, the Proxy Server can focus on its primary job – receiving and decoding network messages. When the Monitors invoke an external tool, they must do so within a Wrapper. Once the Monitors have observed the termination of an external program, they gather any output on *stdout* or *stderr* and return these along with the exit code of the program to the Site Proxy Server. The Proxy Server returns the results to the Communications Server, then the Communications Client, then the ToolProxy, and finally to the original calling FlowObject.

User Interface

The Cockpit provides both the user interface and core functionality of IMEDA. While multiple users may use different instances of the Cockpit simultaneously, there is currently no provision for direct collaboration between multiple users. Developing efficient means of real-time interaction between IMEDA users is one of the major thrusts of the next development cycle.

Currently the GUI of the cockpit provides the following functionality:

- **Flow editing.** Users may create and edit process flows using the flow editor module of the Cockpit. The flow editor provides the user with a simple graphical interface that allows the use of a template of tools for "drawing" a flow. Flows can be optimally organized via services provided by a remote Layout Server written in Perl.
- **Production Library Maintenance.** The Cockpit provides functionality for user maintenance of collections of logical task productions, called libraries. Users may organize productions, modify input/output sets, or create/edit individual productions using flow editors.
- **Class Library Maintenance.** Users are provided with libraries of task and specification classes that are organized into a generalization-specialization hierarchy. Users can instantiate a class into an actual task, specification, selector, or database when creating a flow by simply dragging the appropriate class from a class browser and dropping it onto a flow editor's canvas. The Cockpit

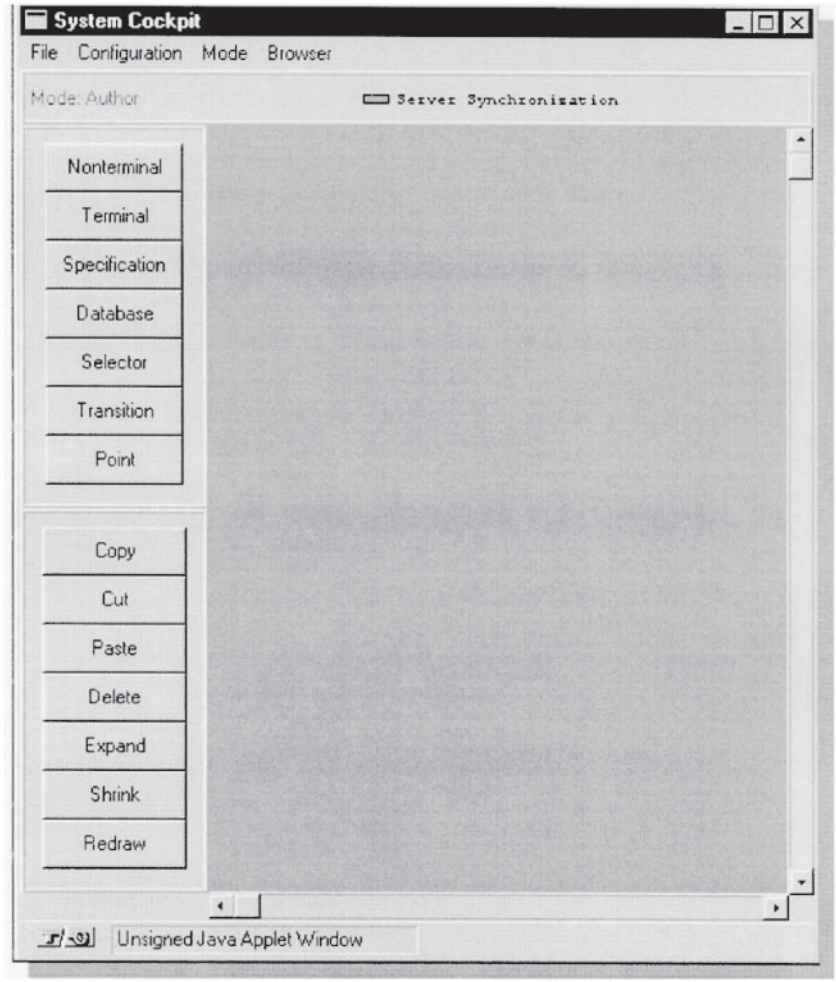


FIGURE 73.14 A system cockpit window.

provides the user with a simple tree structure interface to facilitate the creation and maintenance of class libraries.

- **Process Simulation.** Processes may be simulated using the Cockpit. The Cockpit provides the user with several scheduler modules that determine how the process configuration space will be explored. The schedulers control the execution of external tools (through the appropriate site proxy servers) and simulation display (flow animation for user monitoring of simulation progress). There are multiple schedulers for the user to choose from when simulating a process, including the manual scheduler, comprehensive linear scheduler, etc.
- **Process Archival.** The Cockpit allows processes to be archived on a remote server using the Java File System (JFS). The Cockpit is enabled by a JFS client interface to connect to a remote JFS server where process files are saved and loaded. While the JFS system has its clear advantages, it is also awkward to not allow users to save process files, libraries, etc. on their local systems. Until version 1.1 of the Java Development Kit, local storage by a Java applet was simply not an option — the browser JVM definition did not allow access to most local resources. With version 1.1 of the JDK, however, comes the ability to electronically sign an applet. Once this has been done, users can grant privileged resource access to specific applets after a signature has been verified.

Design Flow Graph Properties

Initially, a flow graph created by a user using GUI is not associated with any system-specific information. For example, when a designer creates an atomic task node in a flow graph, there is initially no association with any external tool. The framework must provide a mechanism for users to bind flow graph entities to the external tools or activities that they represent.

We have used the concept of *properties* to allow users to bind flow graph objects to external entities. In an attempt to maintain flexibility, properties have been implemented in a very generic fashion. Users can define any number of properties for flow object. There are a number of key properties that the framework recognizes for each type of flow object. The user defines these properties to communicate needed configuration data to the framework.

A property consists of a *property label* and *property contents.* The label identifies the property, and consists of an alpha-numeric string with no white space. The contents of a property is any string. Currently users define properties using a freeform text input dialog, with each line defining a property. The first word on a line represents the property label, and the remainder of the line constitutes the property contents.

Property Inheritance

To further extend the flexibility of flow object properties, the framework requires that each flow object be associated with a *flow object class.* Classes allow designers to define properties that are common to all flow objects that inherit from that flow object class. Furthermore, classes are organized into a general-to-specific hierarchy, with children classes inheriting properties from parent classes.

Therefore, the properties of a particular class consist of any properties defined locally for that object, in addition to properties defined in the object's inherited class hierarchy. If a property is defined in both the flow object and one of its parent classes, the property definition in the flow object takes precedence. If a property is defined in more than one class in a class hierarchy, the "youngest" class (e.g., the child in a parent-child relationship) takes precedence.

Classes are defined in the Class Browsers of IMEDA. Designers that have identified a clear general-to-specific hierarchy of flow object classes can quickly create design flow graphs by dragging and dropping from class browsers onto flow design canvases. The user would then need only to overload those properties in the flow objects that are different from their respective parent classes.

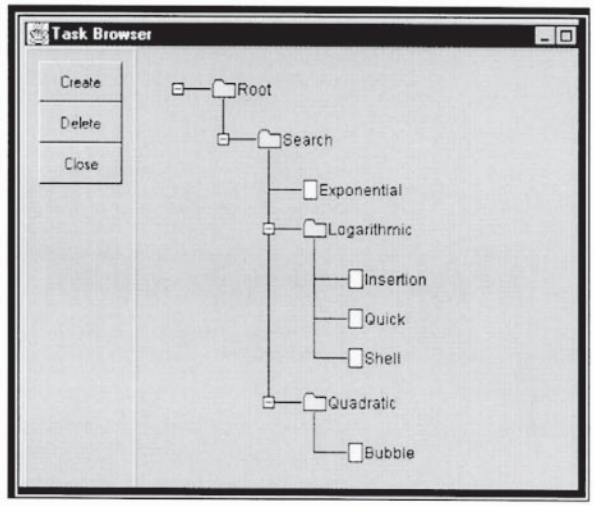


FIGURE 73.15 A task browser.

For example, consider a class hierarchy of classes that all invoke the same external sort tool, but pass different flags to the tool, based on the context. It is likely that all of these tools will have properties in common, such as a common working directory and tool site. By defining these common properties in a common ancestor of all of the classes, such as *Search*, it is unnecessary to redefine the properties in the children classes.

Of course, children classes can define new properties that are not contained in the parent classes, and may also overload property definitions provided by ancestors. Following these rules, class *Insertion* would have the following properties defined: `WORKDIR`, `SITE`, `WRAPPERPATH`, and `CMDLINE`.

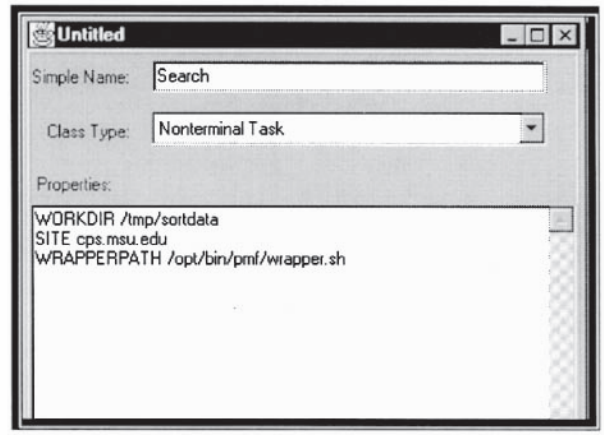


FIGURE 73.16 A property window and property inheritance.

Macro Substitution

While performing benchmarks on IMEDA, one cumbersome aspect of the framework that users often pointed out was the need to re-enter properties for tasks or specifications if, for example, a tool name or working directory changed. Finding every property that needed to be changed was a tedious job, and prone to errors.

In an attempt to deal with this problem, we came up with the idea of *property macros*. That is, a property macro is any macro that is not a key system macro. A macro is a textual substitution rule that can be created by users. By using macros in the property databases of flow objects, design flows can be made more flexible and more amiable to future changes.

As an example, consider a design flow that contains many atomic tasks bound to an external tool. Our previous example using searches is one possible scenario. On one system, the path to the external tool may be "/opt/bin/sort," while on another system the path is "`/user/keyesdav/public/bin/sort`." Making the flow object properties flexible is easy if a property macro named `SORTPATH` is defined in an ancestor of all affected flow objects. Children flow objects can then use that macro in place of a static path when specifying the flow object properties. As a further example, consider a modification to the previous "Search task hierarchy" where we define a macro `SORTPATH` in the Search class, and then use that macro in subsequent children classes, such as the Insertion class.

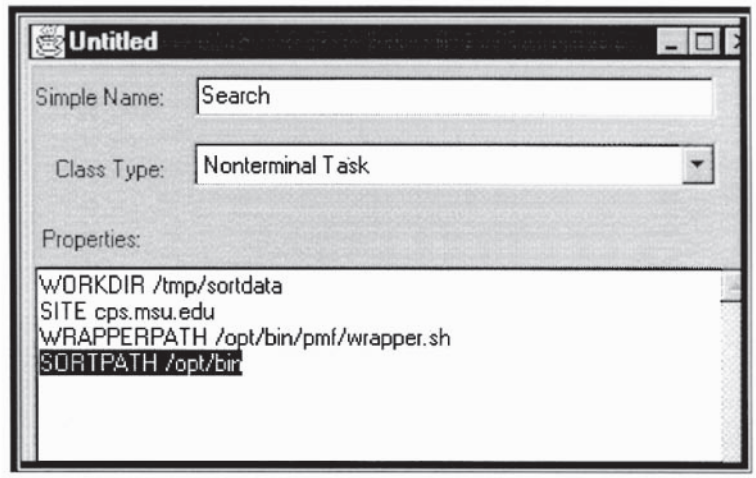


FIGURE 73.17
Macro definition.

In the highlighted portion of the Property Database text area, a macro called "SORTPATH" is defined. In subsequent class' Property Databases, this macro can be used in place of a static path. This makes it easy to change the path for all tools that use the SORTPATH property macro — just the property database dialog where SORTPATH is originally defined needs to be modified.

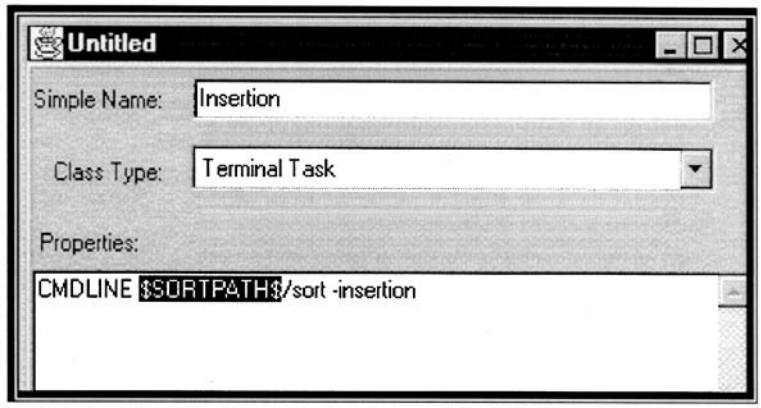


FIGURE 73.18 Macro substitution.

Key Framework Properties

In our current implementation of IMEDA, there are a number of key properties defined. These properties allow users to communicate needed information to the framework in a flexible fashion. Most importantly, it allows system architects to define or modify system properties quickly. This is an important benefit when working with evolving software such as IMEDA.

73.7 Conclusion

Managing the design process is the key factor to improve the productivity in the micro-electronic industry. We have presented an *I*nternet-based *M*icro-*E*lectronic *D*esign *A*utomation (IMEDA) framework to manage the design process. IMEDA uses a powerful formalism, called design process grammars, for representing design processes. We have also proposed an execution environment that utilizes this formalism to assist designers in selecting and executing appropriate design processes. The proposed approach is applicable not only in rapid prototyping but also in any environment where a design is carried out hierarchically and many alternative processes are possible.

The primary advantages of our system are

- *Formalism*: A strong theoretical foundation enables us to analyze how our system will operate with different methodologies.
- *Parallelism*: In addition to performing independent tasks within a methodology in parallel, our system also allows multiple methodologies to be executed in parallel.
- *Extensibility*: New tools can be integrated easily by adding productions and manager programs.
- *Flexibility*: Many different control strategies can be used. They can even be mixed within the same design exercise.

The prototype of IMEDA is implemented using Java. We are currently integrating more tools into our prototype system and developing manager program templates that implement more sophisticated algorithms for pre-evaluation, logical task execution, and query handling. Our system will become more useful as CAD vendors to adapt open software systems and allow greater tool interoperability.

References

Andreoli, J.-M., Pacull, F., and Pareschi, R., XPECT: A framework for electronic commerce, *IEEE Internet Comput.,* vol. 1, no. 4, pp. 40-48, 1998.

Baldwin, R. and Chung, M.J., A formal approach to mangaging design processes, *IEEE Comput.*, pp. 54-63, Feb. 1995a.

Baldwin, R. and Chung, M.J., Managing engineering data for complex products, *Res. Eng. Design*, 7, pp. 215-231, 1995b.

Barthelmann, K., Process specification and verification, *Lect. Notes Comput. Sci.,* 1073 pp. 225-239, 1996.

Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., and Secret, A., The World-Wide Web, *Commun. ACM*, 37, 8, pp. 76-82, 1994.

ten Bosch, K. O., Bingley, P., and Van der Wolf, P., Design flow management in the NELSIS CAD framework, *Proc. 28th Design Automation Conf.*, pp. 711-716, 1991.

Bushnell, M. L. and Director, S. W., VLSI CAD tool integration using the Ulysses environment, *23rd ACM/IEEE Design Automation Conf.*, pp. 55-61, 1986.

Casotto, A., Newton, A. R., and Snagiovanni-Vincentelli, A., Design management based on design traces, *27th ACM/IEEE Design Automation Conf.*, pp. 136-141, 1990.

Tool Encapsulation Specification, Draft Standard, Version 2.0, released by the CFI TES Working Group, 1995.

Chan, F. L., Spiller, M. D., and Newton, A. R., WELD — An environment for web-based electronic design, *35th ACM/IEEE Design Automation Conf.*, June 1998.

Chiueh, T. F. and Katz, R. H., A history model for managing the VLSI design process, *Int. Conf. Comput. Aided Design*, pp. 358-361, 1990.

Chung, M. J., Charmichael, L., and Dukes, M., Managing a RASSP design process, *Comp. Ind.,* 30, pp. 49-61, 1996.

Chung, M. J. and Kim, S., An object-oriented VHDL environment, *27th Design Automation Conf.*, pp. 431-436, 1990.

Chung, M. J. and Kim, S., Configuration management and version control in an object-oriented VHDL environment, *ICCAD 91,* pp. 258-261, 1991.

Chung, M. J. and Kwon, P., A web-based framework for design and manufacturing a mechanical system, *1998 DETC*, Atlanta, GA, Sept. 1998.

Cutkosy, M. R., Tenenbaum, J. M., and Glicksman, J., Madefast: collaborative engineering over the Internet, *Commun. ACM*, vol. 39, no. 9, pp. 78-87, 1996.

Daniell, J. and Director, S. W., An object oriented approach to CAD tool control, *IEEE Trans. Comput.-Aided Design*, pp. 698-713, June 1991.

Dellen, B., Maurer, F., and Pews, G., Knowledge-based techniques to increase the flexibility of workflow management, in *Data and Knowledge Engineering,* North-Holland, 1997.

Derk, M. D. and DeBrunner, L. S., Reconfiguartion for fault tolerance using graph grammar, *ACM Trans. Comput. Syst.,* vol. 16, no. 1, pp. 41-54, Feb. 1998.

Ehrig, H., Introduction to the algebraic theory of graph grammars, *1st Workshop on Graph Grammars and Their Applications to Computer Science and Biology*, pp. 1-69, Springer, LNCS, 1979.

Erkes, J. W., Kenny, K. B., Lewis, J. W., Sarachan, B. D., Sobololewski, M. W., and Sum, R. N., Implementing shared manufacturing services on the World-Wide Web, *Commun. ACM*, vol. 39, no. 2, pp. 34-45, 1996.

Fairbairn, D. G., 1994 Keynote Address, *31st Design Automation Conference*, pp. xvi-xvii, 1994.

Hardwick, M., Spooner, D. L., Rando, T., and Morris, K. C., Sharing manufacturing information in virtual enterprises, *Commun. ACM*, vol. 39, no. 2, pp. 46-54, 1996.

Hawker, S., SEMATECH Computer Integrated Manufacturing(CIM) framework Architecture Concepts, Principles, and Guidelines, version 0.7.

Heiman, P. et al., Graph-based software process management, *Int. J. Software Eng. Knowledge Eng.*, vol.7, no. 4, pp. 1-24, Dec. 1997.

Hines, K. and Borriello, G., A geographically distributed framework for embedded system design and validation, *35th Annual Design Automation Conf.,* 1998.

Hsu, M. and Kleissner, C., Objectflow: towards a process management infrastructure, *Distributed and Parallel Databases*, 4, pp. 169-194, 1996.

IDEF *http://www.idef.com.*

Jacome, M. F. and Director, S. W., A formal basis for design process planning and management, *IEEE Trans. Comput.-Aided Design of Integr. Circuits Syst.*, vol. 15, no. 10, pp. 1197-1211, October 1996.

Jacome, M. F. and Director, S. W., Design process management for CAD frameworks, *29th Design Automation Conf.*, pp. 500-505, 1992.

Di Janni, A., A monitor for complex CAD systems, *23rd Design Automation Conference*, pp. 145-151, 1986.

Katz, R. H., Bhateja, R., E-Li Chang, E., Gedye, D., and Trijanto, V., Design version management, *IEEE Design and Test*, 4(1) pp. 12-22, Feb. 1987.

Kleinfeldth, S., Guiney, M., Miller, J. K., and Barnes, M., Design methodology management, *Proc. IEEE*, vol. 82, no.2, pp. 231-250, Feb. 1994.

Knapp, D. and Parker, A., The ADAM design planning engine, *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.,* vol. 10, no. 7, July 1991.

Knapp, D. W. and Parker, A. C., A design utility manager: the ADAM planning engine, *23rd ACM/IEEE Design Automation Conf.*, pp. 48-54, 1986.

Kocourek, C., An architecture for process modeling and execution support, *Comput. Aided Syst. Theor.* — EUROCAST, 1995.

Kocourek, C., Planning and execution support for design process, *IEEE Interantional symposium and workshop on systems engineering of computer based system proceedings*, 1995.

Knutilla, A., Schlenoff, C., Ray, S., Polyak, S. T., Tate, A., Chiun Cheah, S., and Anderson, R. C., Process specification language: an analysis of existing representations, NISTIR 6160, National Institute of Standards and Technology, Gaithersburg, MD, 1998.

Lavana, H., Khetawat, A., Brglez, F., and Kozminski, K., Executable workflows: a paradigm for collaborative design on the Internet, *34th ACM/IEEE Design Automation Conf.*, June 1997.

Lander, S. E., Staley, S. M., and Corkill, D. D., Designing integrated engineering environments: blackboard-based integration of design and analysis tools, *Proc. IJCAI-95 Workshop Intelligent Manuf. Syst.*, AAAI, 1995

Lyons, K., RaDEO Project Overview, http://www.cs.utah.edu/projects/alpha1/arpa/mind/index.html.

Malone, T. W., Crowston, K., Lee, J., Pentland, B. T., Dellarocas, C., Wyner, G., Quimby, J., Osborne, C., Bernstein, A., Herman, G., Klein, M., and O'Donnell, E., in press.

OASIS Users Guide and Reference Manual, MCNC, Research Triangle Park, North Carolina, 1992.

Petrie, C. J., Agent Based Engineering, the Web, and Intelligence, *IEEE Expert,* Dec. 1996.

Rastogi, P., Koziki, M., and Golshani, F., ExPro-an expert system based process management system, *IEEE Trans. Semiconductor Manuf.,* vol. 6, no. 3, pp. 207-218.

Schlenoff, C., Knutilla, A., and Ray, S., Unified process specification language: requirements for modeling process, *NISTIR 5910,* National Institute of Standards and Technology, Gaithersburg, Maryland, 1996.

Schurmann, B. and Altmeyer, J., Modeling design tasks and tools — the link between product and flow model, *Proc. 34th ACM/IEEE Design Automation Conf.,* June 1997.

Sutton, P. R. and Director, S. W., Framework encapsulations: a new approach to CAD tool interoperability, *35th ACM/IEEE Design Automation Conf.*, June 1998.

Sutton, P. R. and Director, S. W., A description language for design process management, *33rd ACM/IEEE Design Automation Conf.*, pp. 175-180, June 1996.

Spiller, M. D. and Newton, A. R., EDA and Network, *ICCAD,* pp. 470-475, 1997.

Stavas, J. et al., Workflow modeling for implementing complex, CAD-based, design methodologies.

Toye, G., Cutkosky, M. R., Leifer, L. J., Tenenbaum, J. M., and Glicksman, J., SHARE: a methodology and environment for collaborative product development, *Proc. Second Workshop Enabling Technol.: Infrastruct. Collaborative Enterprises*, Los Alamitos, California, IEEE Computer Society Press, pp.33-47, 1993.

Vogel, A. and Duddy, K., *Java Programming with CORBA*, Wiley Computer Publishing, New York.
Welsh, J., Kalathil, B., Chanda, B., Tuck, M. C., Selvidge, W., Finnie, E., and Bard, A., Integrated process control and data management in RASSP enterprise system, *Proc. of 1995 RASSP Conf.,* 1995.
Westfechtel, B., Integrated product and process management for engineering design applications, *Integr. Comput.-Aided Eng.*, vol. 3, no. 1, pp. 20-35, 1996.
Yang, Z. and Duddy, K., CORBA: a platform for distributed object computing *ACM Operating Syst. Rev.,* vol. 30, no. 2, pp. 4-31, 1996.

Parker, A.C., et al. "System-Level Design"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

74

System-Level Design

Alice C. Parker
University of Southern California

Yosef Gavriel
Virginia Polytechnic Institute and State University

Suhrid A. Wadekar
IBM Corp.

74.1 Introduction

The term **system**, when used in the digital design domain, implies many different entities. A system can consist of a processor, memory, and input/output, all on a single integrated circuit, or it can consist of a network of processors, geographically distributed, each performing a specific application. There can be a single clock, with modules communicating synchronously, multiple clocks with asynchronous communication, or an entirely asynchronous operation. The design can be general, or specific to a given application, i.e., application-specific. Together, the previously mentioned variations constitute the system style. To a great extent, system style selection is determined by the physical technologies used, the environment in which the system operates, designer experience, and corporate culture, and is not automated to any great extent.

System-level design covers a wide range of design activities and design situations. It includes the more specific activity system engineering, that involves the requirements development, test planning, subsystem interfacing, and end-to-end analysis of systems. System-level design is sometimes called system architecting, a term used widely in the aerospace industry.

General-purpose system-level design involves the design of programmable digital systems, including the basic modules containing storage, processors, input/output, and system controllers. At the system level, the design activities include determining the following:

- the power budget (the amount of power allocated to each module in the system);
- the cost and performance budget allocated to each module in the system;
- the interconnection strategy;
- the selection of commercial off-the-shelf modules (COTS);
- the packaging of each module;
- the overall packaging strategy;
- the number of processors, storage units, and input/output interfaces required; and
- the overall characteristics of each processor, storage unit, and input/output interface.

For example, memory system design focuses on the number of memory modules required, how they are organized, and the capacity of each module. A specific system-level issue in this domain can be the question of how to partition the memory between the processor chip and the off-chip memory. At a higher level, a similar issue might involve configuration of the complete storage hierarchy, including memory, disk drives, and archival storage.

For each general-purpose system designed, many systems are designed to perform specific applications. Application-specific system design involves the same activities as described previously, but can involve many more issues, since there are usually more custom logic modules involved. Specifications for application-specific systems contain not only requirements on general capabilities but also the functionality required in terms of specific tasks to be executed. Major application-specific, system-level design activities include not only the general-purpose system design activities, but the following activities as well:

- partitioning an application into multiple functional modules;
- scheduling the application tasks on shared functional modules;
- allocating functional modules to perform the application tasks;
- allocating and scheduling storage modules to contain blocks of data as it is processed;
- determining the implementation styles of functional modules;
- determining the word lengths of data necessary to achieve a given accuracy of computation; and
- predicting resulting system characteristics once the system design is complete.

Each of the system design tasks given in the previous two lists will be described in subsequent detail. Since the majority of system design activities are application-specific, this section will focus on system-level design of application-specific systems. Related activities, hardware-software co-design, verification, and simulation are covered in other sections.

Design Philosophies and System-Level Design

Many design tools have been constructed with a top-down design philosophy. Top-down design represents a design process whereby the design becomes increasingly detailed until final implementation is complete. Considerable prediction of resulting system characteristics is required in order to make the higher-level decisions with some degree of success.

Bottom-up design, on the other hand, relies on designing a set of primitive elements and forming more complex modules from those elements. Ultimately, the modules are assembled into a system. At each stage of the design process there is complete knowledge of the parameters of the lower-level elements. However, the lower-level elements may be inappropriate for the tasks at hand.

Industry system designers describe the design process as being much less organized and considerably more complex than the top-down and bottom-up philosophies suggest. There is a mixture of top-down and bottom-up activities, with major bottlenecks of the system receiving detailed design consideration while other parts of the system still exist only as abstract specifications. For this reason, the system-level design activities presented in detail here support such a complex design situation. Modules, elements, and components used to design at the system level might exist or might only exist as abstract estimates along with requirements. The system can be designed after all modules have been designed and manufactured, prior to any detailed design, or with a mixture of existing and new modules.

The System Design Space

System design, like data path design, is quite straightforward as long as the constraints are not too severe. However, most modern system designs must solve harder problems than problems solved by existing systems; moreover, designers must race to produce working systems faster than competitors. More variations in design are possible than ever before, and such variations require that a large design space be explored. The dimensions of the design space (its axes) are system properties such as cost, power, design time, and performance. The design space contains a population of designs, each of which possesses different values of these system properties. There are literally millions of system designs for a given specification, each of which exhibits different cost, performance, power consumption, and design time. Straightforward solutions that do not attempt to optimize system properties are easy to obtain but may be inferior to designs that require use of system-level design tools and perhaps many iterations of design. The complexity of system design is not due to the fact that system design is an inherently difficult activity, but that so many variations in design are possible and time does not permit exploration of all of them.

74.2 System Specification

Complete system specifications contain a wide range of information including

- constraints on the system power, performance, cost, weight, size, and delivery time;
- required functionality of the system components;
- any required information about the system structure;
- required communication between system components;
- the flow of data between components;
- the flow of control in the system; and
- the specification of input precision and desired output precision.

Most systems specifications that are reasonably complete exist first in a natural language. Such natural language interfaces are not currently available with commercial, system-level design tools.

More conventional system-specification methods, used to drive system-level design tools, include formal languages, graphs, or a mixture of the two. Each of the formal system-specification methods described here contains some of the information found in a complete specification, i.e., most specification methods are incomplete. The remaining information necessary for full system design can be provided interactively by the designer, can be entered later in the design process, or can be provided in other forms at the same time the specification is processed. The required design activities determine the specification method used for a given system design task.

There are no widely adopted formal languages for system-level hardware design although SLDL (System-Level Design Language) is currently being developed by an industry group. Hardware descriptive languages such as VHDL[1] and Verilog[2] are used to describe the functionality of modules in an application-specific system. High-level synthesis tools can then synthesize such descriptions to produce register-transfer designs. Extensions of VHDL have been proposed to encompass more system-level design properties. Apart from system constraints, VHDL specifications can form complete system descriptions. However, the level of detail required in VHDL, and to some extent in Verilog, requires the designer to make some implementation decisions. In addition, some information explicit in more abstract specifications, such as the flow of control between tasks, is implicit in HDLs.

Graphical tools have been used for a number of years to describe system behavior and structure. Block diagrams are often used to describe system structure. Block diagrams assume that tasks have already been assigned to basic blocks and that their configuration in the system has been specified. They generally cannot represent the flow of data or control, or design constraints. The PMS (Processor Memory Switch) notation invented by Bell and Newell was an early attempt to formalize the use of block diagrams for system specification.[3]

Petri nets have been used for many years to describe system behavior using a token-flow model. A token-flow model represents the flow of control with tokens, which flow from one activity of the system to another. Many tokens can be active in a given model concurrently, representing asynchronous activity and parallelism, important in many system designs. Timed Petri nets have been used to model system performance, but Petri nets cannot easily be used to model other system constraints, system behavior, or any structural information.

State diagrams and graphical tools such as State Charts[4] provide alternative methods for describing systems. Such tools provide mechanisms to describe the flow of control, but they do not describe system constraints, system structure, data flow, or functionality.

Task-flow graphs, an outgrowth from the Control/Data-Flow Graphs (CDFG) used in high-level synthesis, are often used for system specification. These graphs describe the flow of control and data between tasks. When used in a hierarchical fashion, task nodes in the task-flow graph can contain detailed functional information about each task, often in the form of a CDFG. Task flow graphs contain no mechanisms for describing system constraints or system structure.

Spec Charts[5] incorporate VHDL descriptions into State-Chart-like notation, overcoming the lack of functional information found in State Charts.

Figure 74.1 illustrates the use of block diagrams, Petri nets, task flow graphs, and spec charts.

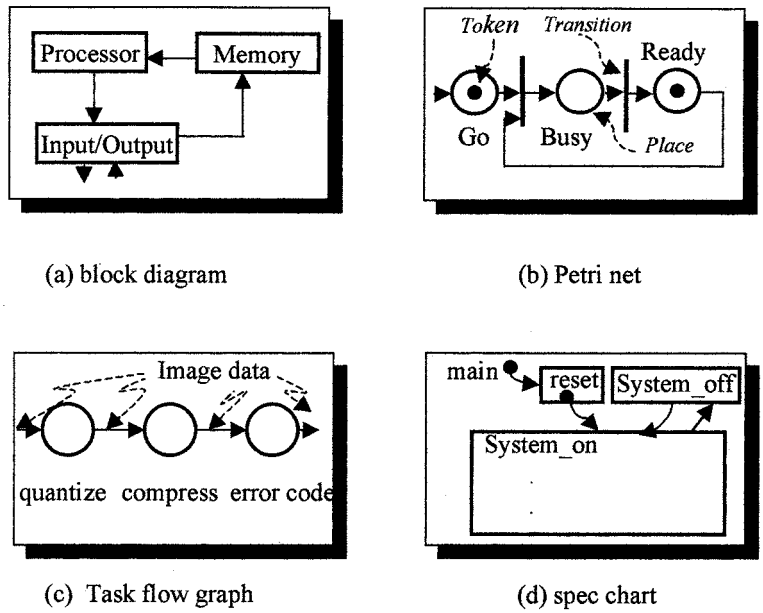


FIGURE 74.1 The use of block diagrams, Petri nets, task flow graphs, and spec charts, shown in simplified form.

74.3 System Partitioning

Most systems are too large to fit on a single substrate. If the complexity of the system tasks and the capacity of the system modules are of the same order, then partitioning is not required. All other systems must be partitioned so that they fit into the allowed substrates, packages, boards, multi-chip modules, and cases. Partitioning determines the functions, tasks, or operations in each partition of a system. Each partition can represent a substrate, package, multi-chip module, or larger component. Partitioning is performed with respect to a number of goals, including minimizing cost, design time, or power, or maximizing performance. Any of these goals can be reformulated as specific constraints, like meeting given power requirements.

When systems are partitioned, resulting communication delays must be taken into account, affecting performance. Limitations on interconnection size must be taken into account, affecting performance as well. Pin and interconnection limitations force the multiplexing of inputs and outputs, reducing performance, and sometimes affecting cost. Power consumption must also be taken into account. Power balancing between partitions and total power consumption might both be considerations. In order to meet market windows, system partitions can facilitate the use of COTS, programmable components, or easily fabricated components such as gate arrays. In order to meet cost constraints, functions that are found in the same partition might share partition resources. Such functions or tasks cannot execute concurrently, affecting performance.

Partitioning is widely used at the logic level, as well as on physical designs. In these cases, much more information is known about the design properties, and the interconnection structure has been determined. System partitioning is performed when information about the specific components' properties might be uncertain, and the interconnection structure undetermined. For these reasons, techniques used at lower levels must be modified to include predictions of design properties not yet known and prediction of the possible interconnection structure as a result of the partitioning.

The exact partitioning method used depends on the type of specification available. If detailed CDFG or HDL specifications are used, the partitioning method might be concerned with which register-transfer functions (e.g., add, multiply, shift) are found in each partition. If the specification primitives are tasks, as in a task-flow graph specification, then the tasks must be assigned to partitions. Generally, the more detailed the specification, the larger the size of the partitioning problem. Powerful partitioning methods can be applied to problems of small size ($n < 100$). Weaker methods such as incremental improvement must be used when the problem size is larger.

Partitioning methods can be based on constructive partitioning or iterative improvement. Constructive partitioning involves taking an unpartitioned design and assigning operations or tasks to partitions. Basic constructive partitioning methods include bin packing using a first-fit decreasing heuristic, clustering operations into partitions by assigning nearest neighbors to the same partition until the partition is full, random placement into partitions, and integer programming approaches.

Constructive Partitioning Techniques

Bin packing involves creating a number of bins equal in number to the number of partitions desired and equal in size to the size of partitions desired. Then, the tasks or operations are sorted by size. The largest task in the list is placed in the first bin, and then the next largest is placed in the first bin, if it will fit, or if it does not fit, into the second bin. Each task is placed into the first bin in which it will fit, until all tasks have been placed in bins. More bins are added if necessary. This simple heuristic is useful to create an initial set of partitions to be improved iteratively later.

Clustering is a more powerful method to create partitions. Here is a simple clustering heuristic. Each task is ranked by the extent of "connections" to other tasks either due to control flow, data flow, or physical position limitations. The most connected task is placed in the first partition, and then the tasks connected to it are placed in the same partition, in order of the strength of their connections to the first task. Once the partition is full, the task with the most total connections remaining outside a partition is placed in a new partition, and other tasks are placed there in order of their connections to the first task. This heuristic continues until all tasks are placed.

Random partitioning places tasks into partitions in a greedy fashion until the partitions are full. Some randomization of the choice of tasks is useful in producing a family of systems, of which each member is partitioned randomly. This family of systems can be used successfully in iterative improvement techniques for partitioning, as described later in this section.

The most powerful technique for constructive partitioning is mathematical programming. Integer and mixed integer-linear programming techniques have been used frequently in the past for partitioning. Such powerful techniques are computationally very expensive, and they are successful only when the number of objects to be partitioned is small. The basic idea behind integer programming used for

partitioning is the following: An integer, $TP(i,j)$, is used to represent the assignment of tasks to partitions. When $TP = 1$, task i is assigned to partition j. For each task in this problem, there would be an equation

$$\sum_{j=1}^{partition\ total} TP(i,j) = 1 \tag{74.1}$$

This equation more or less states that each task must be assigned to one and only one partition. There would be many constraints of this type in the integer program, some of which were inequalities. There would be one function representing cost, performance, or other design property, to be optimized. The simultaneous solution of all constraints, given some minimization or maximization goal, would yield the optimal partitioning.

Apart from the computational complexity of this technique, the formulation of the mathematical programming constraints is tedious and error prone if performed manually. The most important advantage of mathematical programming formulations is the discipline it imposes on the CAD programmer in formulating an exact definition of the CAD problem to be solved. Such problem formulations can prove useful when applied in a more practical environment, as described below in the next section, "Iterative Partitioning Techniques."

Iterative Partitioning Techniques

Of the many iterative partitioning techniques available, two have been applied most successfully at the system level. These techniques are min-cut partitioning, first proposed by Kernigan and Lin, and genetic algorithms.

Min-cut partitioning involves exchanging tasks or operations between partitions in order to minimize the total amount of "interconnections" cut. The interconnections can be computed as the sum of data flowing between partitions, or as the sum of an estimate of the actual interconnections that will be required in the system. The advantage of summing the data flowing is that it provides a quick computation, since the numbers are contained in the task flow graph. Better partitions can be obtained if the required physical interconnections are taken into account, since they are related more directly to cost and performance than to the amount of data flowing. If a partial structure exists for the design, predicting the unknown interconnections allows partitioning to be performed on a mixed design, one that contains existing parts as well as parts under design.

Genetic algorithms, highly popular for many engineering optimization problems, are especially suited to the partitioning problem. The problem formulation is similar in some ways to mathematical programming formulations. A simple genetic algorithm for partitioning is described here. In this example, a chromosome represents each partitioned system design, and each chromosome contains genes, representing information about the system. A particular gene, $TP(i,j)$, might represent the fact that task i is contained in partition j when it is equal to 1, and is set to 0 otherwise. A family of designs created by some constructive partitioning technique then undergoes mutation and crossover as new designs evolve. A fitness function is used to check the quality of the design, and the evolution is halted when the design is considered fit, or when no improvement has occurred after some time. In the case of partitioning, the fitness function might include the estimated volume of interconnections, the predicted cost or performance of the system, or other system properties.

The reader might note some similarity between the mathematical programming formulation of the partitioning problem presented here and the genetic algorithm formulation. This similarity allows the CAD developer to create a mathematical programming model of the problem to be solved, find optimal solutions to small problems, and create a genetic algorithm version. The genetic algorithm version can be checked against the optimal solutions found by the mathematical program. However, genetic

algorithms can take into account many more details than can mathematical program formulations, can handle non-linear relationships better, and can even handle stochastic parameters.[1]

Partitioning is most valuable when there is a mismatch between the sizes of system tasks and the capacities of system modules. When the system tasks and system modules are more closely matched, then the system design can proceed directly to scheduling and allocating tasks to processing modules.

74.4 Scheduling and Allocating Tasks to Processing Modules

Scheduling and allocating tasks to processing modules involve the determination of how many processing modules are required, which modules execute which tasks, and the order in which tasks are processed by the system. In the special case where only a single task is processed by each module, the scheduling becomes trivial. Otherwise, if the tasks share modules, the order in which the tasks are processed by the modules can affect system performance or cost. If the tasks are ordered inappropriately, some tasks might wait too long for input data, and performance might be affected. Or, in order to meet performance constraints, additional modules must be added to perform more tasks in parallel, increasing system cost.

A variety of modules might be available to carry out each task, with differing cost and performance parameters. As each task is allocated to a module, that module is selected from a set of modules available to execute the task. This is analogous to the task module selection, which occurs as part of high-level synthesis. For the system design problem considered here, the modules can be either general-purpose processors, special-purpose processors (e.g., signal processing processors), or special-purpose hardware. If all (or most) modules used are general-purpose, the systems synthesized are known as heterogeneous application-specific multiprocessors.

A variety of techniques can be used for the scheduling and allocation of system tasks to modules. Just as with partitioning, these techniques can be constructive or iterative. Constructive scheduling techniques for system tasks include greedy techniques such as ASAP (as soon as possible) and ALAP (as late as possible). In ASAP scheduling, the tasks are scheduled as early as possible on a free processing module. The tasks scheduled first are the ones with the longest paths from their outputs to final system outputs or system completion. Such techniques, with variations, can be used to provide starting populations of system designs to be further improved iteratively. The use of such greedy techniques for system synthesis differs from the conventional use in high-level synthesis, where the system is assumed to be synchronous, with tasks scheduled into time steps. System task scheduling assumes no central clock, and tasks take a wide range of times to complete. Some tasks could even complete stochastically, with completion time a random variable. Other tasks could complete basic calculations in a set time, but could perform a finer grain (more accurate) of computations if more time were available. A simple task-flow graph is shown in Fig. 74.2, along with a Gantt chart illustrating the ASAP scheduling of tasks onto two processors. Note that two lengthy tasks are performed in parallel with three shorter tasks, and that no two tasks take the same amount of time.

Similar to partitioning, scheduling and allocation, along with module selection, can be performed using mathematical programming. In this case, since the scheduling is asynchronous, time becomes a linear rather than integer quantity. Therefore, mixed integer-linear programming (MILP) is employed to model system-level scheduling and allocation. A typical MILP timing constraint is the following:

$$T_{OA}\left(i\right) + Cdelay \le T_{IR}\left(j\right) \tag{74.2}$$

[1]Stochastic parameters represent values that are uncertain. There is a finite probability of a parameter taking a specific value that varies with time, but that in general, probability is less than one.

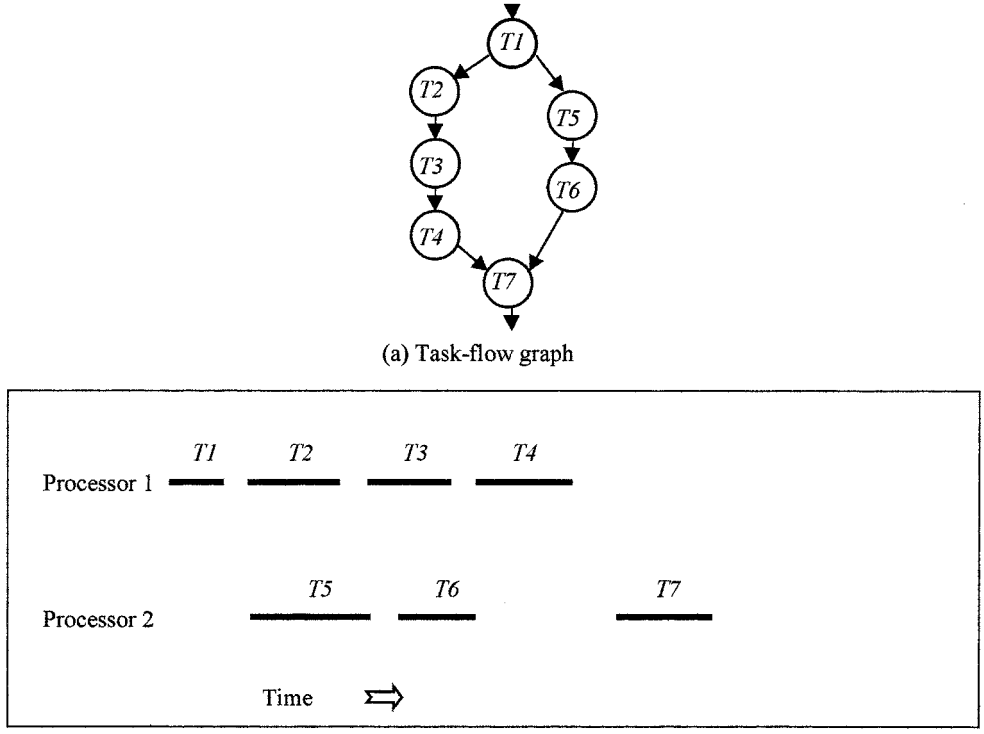


FIGURE 74.2 An example task-flow graph and schedule.

where $T_{OA}(i)$ is the time the output is available from task *i*, *Cdelay* is the communication delay, and $T_{IR}(j)$ is the time the input is required by task *j*. Unfortunately, the actual constraints used in scheduling and allocation are mostly more complex than this, because the design choices have yet to be made. Here is another example:

$$T_{OA}(i) \geq T_{IR}(i) + \sum_{k}\left[Pdelay(k) * M(i,k)\right] \tag{74.3}$$

This constraint states that the time an output from task *i* is available is greater than or equal to the time necessary inputs are received by task *i*, and a processing delay *Pdelay* has occurred. $M(i,k)$ indicates that task *i* is allocated to module *k*. *Pdelay* can take on a range of values, depending on which of k modules is being used to implement task *i*. The summation is actually a linearized select function that picks the value of *Pdelay* to use depending on which value of $M(i,k)$ is set to 1.

As with partitioning, mathematical programming for scheduling and allocation is computationally intensive and impractical for all but the smallest designs, but it does provide a baseline model of design that can be incorporated in other tools.

The most frequent technique used for iterative improvement in scheduling and allocation at the system level is a genetic algorithm. The genes can be used to represent task allocation and scheduling. In order to represent asynchronous scheduling accurately, time is generally represented as a linear quantity in such genes, rather than an integer quantity.

74.5 Allocating and Scheduling Storage Modules

In digital systems, all data requires some form of temporary or permanent storage. If the storage is shared by several data sets, the use of the storage by each data set must be scheduled. The importance of this task in system design has been overlooked in the past, but it has now become an important system-level task.

Modern digital systems usually contain some multimedia tasks and data. The storage requirements for multimedia tasks sometimes result in systems where processing costs are dwarfed by storage costs, particularly caching costs. For such systems, storage must be scheduled and allocated either during or after task scheduling and allocation. If storage is scheduled and allocated concurrently with task scheduling and allocation, the total system costs are easier to determine, and functional module sharing can be increased if necessary in order to control total costs. On the other hand, if storage allocation and scheduling are performed after task scheduling and allocation, then both programs are simpler, but the result may not be as close to optimal.

Techniques similar to those used for task scheduling and allocation can be used for storage scheduling and allocation.

74.6 Selecting Implementation and Packaging Styles for System Modules

Packaging styles can range from single-chip dual-in-line packages (DIPs) to multi-chip modules (MCMs), boards, racks, and cases. Implementation styles include general-purpose processor, special-purpose programmable processor (e.g., signal processor), COTS modules, Field Programmable Gate Arrays (FPGAs), gate array, standard cell, and custom integrated circuits. For many system designs, system cost, performance, power, and design time constraints determine selection of implementation and packaging styles. Tight performance constraints favor custom integrated circuits, packaged in multi-chip modules. Tight cost constraints favor off-the-shelf processors and gate array implementations, with small substrates and inexpensive packaging. Tight power constraints favor custom circuits. Tight design time constraints favor COTS modules and FPGAs. If a single design property has high priority, the designer can select the appropriate implementation style and packaging technology. If, however, design time is crucial, but the system to be designed must process video signals in real time, then tradeoffs in packaging and implementation style must be made. The optimality of system cost and power consumption might be sacrificed: The entire design might be built with FPGAs, with much parallel processing and at great cost and large size. Because time-to-market is so important, early market entry systems may sacrifice the optimality of many system parameters initially and then improve them in the next version of the product.

Selection of implementation styles and packaging can be accomplished by adding some design parameters to the scheduling and allocation program, if that program is not already computationally intensive. The parameters added would include

- a variable indicating that a particular functional module was assigned a certain implementation style;
- a variable indicating that a particular storage module was assigned a certain implementation style;
- a variable indicating that a particular functional module was assigned a certain packaging style; and
- a variable indicating that a particular storage module was assigned a certain packaging style.

Some economy of processing could be obtained if certain implementation styles precluded certain packaging styles.

74.7 The Interconnection Strategy

Modules in a digital system are usually interconnected in some carefully architected, consistent manner. If point-to-point interconnections are used, they are used throughout the system, or in a subsystem. In the

same manner, buses are not broken arbitrarily to insert point-to-point connections or rings. For this reason, digital system design programs usually assume an interconnection style and determine the system performance relative to that style. The most common interconnection styles are bus, point-to-point, and ring.

74.8 Word Length Determination

Functional specifications for system tasks are frequently detailed enough to contain the algorithm to be implemented. In order to determine the implementation costs of each system task, knowledge of the word widths to be used is important, as system cost varies almost quadratically with word width.

Tools to automatically select task word width are currently experimental, but the potential for future commercial tools exists.

In typical hardware implementations of an arithmetic-intensive algorithm, designers must determine the word lengths of resources such as adders, multipliers, and registers. In a recent publication,[6] Wadekar and Parker presented algorithm-level optimization techniques to select distinct word lengths for each computation which meet the desired accuracy and minimize the design cost for the given performance constraints. The cost reduction is possible by avoiding unnecessary bit-level computations that do not contribute significantly to the accuracy of the final results. At the algorithm level, determining the necessary and sufficient precision of an individual computation is a difficult task, since the precision of various predecessor/successor operations can be traded off to achieve the same desired precision in the final result. This is achieved using a mathematical model[7] and a genetic selection mechanism.[6] There is a distinct advantage to word-length optimization at the algorithmic level. The optimized operation word lengths can be used to guide high-level synthesis or designers to achieve an efficient utilization of resources of distinct word lengths and costs. Specifically, only a few resources of larger word lengths and high cost may be needed for operations requiring high precision to meet the final accuracy requirement. Other relatively low-precision operations may be executed by resources of smaller word lengths. If there is no timing conflict, a large word length resource can also execute a small word length operation, thus improving the overall resource utilization further. These high-level design decisions cannot be made without the knowledge of word lengths prior to synthesis.

74.9 Predicting System Characteristics

In system-level design, early prediction gives designers the freedom to make numerous high-level choices (such as die size, package type, and latency of the pipeline) with confidence that the final implementation will meet power and energy as well as cost and performance constraints. These predictions can guide power budgeting and subsequent synthesis of various system components, which is critical in synthesizing systems that have low power dissipation, or long battery life. The use by synthesis programs of performance and cost lower bounds allows smaller solution spaces to be searched, which leads to faster computation of the optimal solution.

System cost, performance, power consumption, and design time can be computed if the properties of each system module are known. System design using existing modules requires little prediction. However, if system design is performed prior to design of any of the contained system modules, their properties must be predicted or estimated. Due to the complexities of prediction techniques, describing these techniques is a subject worthy of an entire chapter. A brief survey of related readings is found in the next section.

The register-transfer and subsequent lower level power prediction techniques such as gate- and transistor-level techniques are essential for validation before fabricating the circuit. However, these techniques are less efficient for system-level design process, as a design must be generated before prediction can be done.

74.10 A Survey of Research in System Design

Many researchers have investigated the problem of system design, dating back to the early 1970s. This section highlights work that is distinctive, along with tutorial articles covering relevant topics. Much

good research is not referenced here, and the reader is reminded that the field is dynamic, with new techniques and tools appearing almost daily.

Issues in top-down vs. bottom-up design approaches were highlighted in the design experiment reported by Gupta et al.[8]

System Specification

System specification has received little attention historically except in the specific area of software specifications. Several researchers have proposed natural language interfaces capable of processing system specifications and creating internal representations of the systems that are considerably more structured. Of note is the work by Granacki[9] and Cyre.[10] One noteworthy approach is the Design Specification Language (D)SL, found in the Design Analysis and Synthesis Environment.[11] One of the few books on the subject concerns the design of embedded systems — systems with hardware and software designed for a particular application set.[12] In one particular effort, Petri nets were used to specify the interface requirements in a system of communicating modules, which were then synthesized.[13] The SIERA system designed by Srivastava, Richards, and Broderson[14] supports specification, simulation, and interactive design of systems.

Partitioning

Partitioning research covers a wide range of system design situations. Many early partitioning techniques dealt with assigning register-level operations to partitions. APARTY, a partitioner designed by Lagnese and Thomas, partitions CDFG designs for single-chip implementation in order to obtain efficient layouts.[15] Vahid[16] performed a detailed survey of techniques for assigning operations to partitions. CHOP assigns CDFG operations to partitions for multi-chip design of synchronous, common clocked systems.[17] Vahid and Gajski developed an early partitioner, SpecPart, which assigns processes to partitions.[18] Chen and Parker reported on a process-to-partition technique called ProPart.[19]

Non-pipelined Design

Although research on **system design** spans more than two decades, most of the earlier works focus on single aspects of design like task assignment, and not on the entire design problem. We cite some representative works here. These include graph theoretical approaches to task assignment,[20,21] analytical modeling approaches for task assignment,[22] and probabilistic modeling approaches for task partitioning,[23,24] scheduling,[25] and synthesis.[26] Two publications of note cover application of heuristics to system design.[27,28]

Other noteworthy publications include mathematical programming formulations for task partitioning[29] and communication channel assignment.[30] Early efforts include those done by soviet researchers since the beginning of the 1970s such as Linsky and Kornev[31] and others, where each model only included a subset of the entire synthesis problem. Chu et al.[32] published one of the first mixed integer-linear programming (MILP) models for a sub-problem of system-level design, scheduling. Recently the program SOS (Synthesis of Systems), including a compiler for MILP models[33,34] was developed, based on a comprehensive MILP model for system synthesis. SOS takes a description of a system described using a task-flow graph, a processor library and some cost and performance constraints, and generates an MILP model to be optimized by an MILP solver. The SOS tool generates MILP models for the design of non-periodic (non-pipelined) heterogeneous multiprocessors. The models share a common structure, which is an extension of the previous work by Hafer and Parker for high-level synthesis of digital systems.[35]

Performance bounds of solutions found by algorithms or heuristics for system-level design are proposed in many papers, including the landmark papers by Fernandez and Bussel[36] and Garey and Graham[37] and more recent publications.[38]

The recent work of Gupta et al.[8] reported the successful use of system-level design tools in the development of an application-specific heterogeneous multiprocessor for image processing. Gupta and Zorian[39] describe the design of systems using cores, silicon cells with at least 5000 gates. The same issue of Design and Test contains a number of useful articles on design of embedded core-based systems. Li and Wolf[40] report on a model of hierarchical memory and a multiprocessor synthesis algorithm which takes into account the hierarchical memory structure.

A major project, RASSP, is a rapid-prototyping approach whose development is funded by the U.S. Department of Defense.[41] RASSP addresses the integrated design of hardware and software for signal processing applications.

An early work on board-level design, MICON, is of particular interest.[42] Newer research results solving similar problems with more degrees of design freedom include the research by C-T Chen[43] and D-H Heo.[44] GARDEN, written by Heo, finds the design with the shortest estimated time to market, which meets cost and performance constraints.

All the MILP synthesis works cited to this point address only the nonperiodic case.

Synthesis of application-specific heterogeneous multiprocessors is a major activity in the general area of system synthesis. One of the most significant system-level design efforts is Lee's Ptolemy project at the University of California, Berkeley. Representative publications include papers by Lee and Bier describing a simulation environment for signal processing[45] and the paper by Kalavade et al.[46] Another prominant effort is the SpecSyn project[47] which is a system-level design methodology and framework.

Macro-pipelined Design

Macro-pipelined (periodic) multiprocessors execute tasks in a pipelined fashion, with tasks executing concurrently on different sets of data. Most research work on design of macro-pipelined multiprocessors has been restricted to homogeneous multiprocessors having negligible communication costs. This survey divides the past contributions according to the execution mode: preemptive or nonpreemptive.

Nonpreemptive Mode

The nonpreemptive mode of execution assumes that each task is executed without interruption. It is used quite often in low-cost implementations. Much research has been performed on system scheduling for the nonpreemptive mode. A method to compute the minimum possible value for the initiation interval for a task-flow graph given an unlimited number of processors and no communication costs was found by Renfors and Neuvo.[48]

Wang and Hu[49] use heuristics for the allocation and full static scheduling (meaning that each task is executed on the same processor for all iterations) of generalized perfect-rate task-flow graphs on homogeneous multiprocessors. Wang and Hu apply planning, an artificial intelligence method, to the task scheduling problem. The processor allocation problem is solved using a conflict-graph approach.

Gelabert and Barnwell[50] developed an optimal method to design macro-pipelined homogeneous multiprocessors using cyclic-static scheduling, where the task-to-processor mapping is not time-invariant as in the full static case, but is periodic, i.e., the tasks are successively executed by all processors. Gelabert and Barnwell assume that the delays for intra-processor and inter-processor communications are the same, which is an idealistic scenario. Their approach is able to find an optimal implementation (minimal iteration interval) in exponential time in the worst case.

In his doctoral thesis, Tirat-Gefen[51] extended the SOS MILP model to solve for optimal macro-pipelined, application-specific heterogeneous multiprocessors. He also proposed an integer-linear programming (ILP) model allowing simultaneous optimal retiming and processor/module selection in high and system-level synthesis.[52]

Verhaegh[53] addresses the problem of periodic multidimensional scheduling. His thesis uses an ILP model to handle the design of homogeneous multiprocessors without communication costs implementing data-flow programs with nested loops. His work evaluates the complexity of the scheduling and

allocation problems for the multidimensional case, which were both found to be NP-complete. Verhaegh proposes a set of heuristics to handle both problems.

Passos and Sha[54] evaluate the use of multi-dimensional retiming for synchronous data-flow graphs. However, their formalism can only be applied to homogeneous multiprocessors without communication costs.

The Preemptive Mode of Execution

Feng and Shin[55] address the optimal static allocation of periodic tasks with precedence constraints and preemption on a homogeneous multiprocessor. Their approach has an exponential time complexity. Ramamrithan[56] developed a heuristic method that has a more reasonable computational cost. Rate-monotonic scheduling (RMS) is a commonly used method for allocating periodic real-time tasks in distributed systems.[57] The same method can be used in homogeneous multiprocessors.

Genetic Algorithms

Genetic algorithms are becoming an important tool for solving the highly non-linear problems related to system-level synthesis. The use of genetic algorithms in optimization is well discussed by Michalewicz[58] where formulations for problems such as bin packing, processor scheduling, traveling salesman, and system partitioning are outlined.

Research works involving the use of genetic algorithms to system-level synthesis problems are starting to be published, for example, as are the results of the following:

- Hou et al.[59] — scheduling of tasks in a homogeneous multiprocessor without communication costs;
- Wang et al.[60] — scheduling of tasks in heterogeneous multiprocessors with communication costs but not allowing cost vs. performance tradeoff, i.e., all processors have the same cost;
- Ravikumar and Gupta[61] — mapping of tasks into a reconfigurable homogeneous array processor without communication costs;
- Tirat-Gefen and Parker[62] — a genetic algorithm for design of application-specific heterogeneous multiprocessors (ASHM) with nonnegligible communications costs specified by a nonperiodic task-flow graph representing both control and data flow; and
- Tirat-Gefen[51] — introduced a full-set of genetic algorithms for system-level design of ASHMs incorporating new design features such as imprecise computation and probabilistic design.

Imprecise Computation

The main results in imprecise computation theory are due to Liu et al.[63] who developed polynomial time algorithms for optimal scheduling of preemptive tasks on homogeneous multiprocessors without communications costs. Ho et al.[64] proposed an approach to minimize the total error, where the error of a task being imprecisely executed is proportional to the amount of time that its optional part was not allowed to execute, i.e., the time still needed for its full completion. Polynomial time-optimal algorithms were derived for some instances of the problem.[63]

Tirat-Gefen et al.[65] presented a new approach for application-specific, heterogeneous multiprocessor design that allows tradeoffs between cost, performance, and data-quality through incorporation of imprecise computation into the system-level design cycle.

Probabilistic Models and Stochastic Simulation

Many probabilistic models for solving different subproblems in digital design have been proposed recently. The problem of task and data-transfer scheduling on a multiprocessor when some tasks (data transfers) have nondeterministic execution times, (communication-times) can be modeled by PERT

networks, which were introduced by Malcolm et al.[66] along with the critical path method (CPM) analysis methodology.

A survey on PERT networks and their generalization to conditional PERT networks is done by Elmaghraby.[67] In system-level design, the completion time of a PERT network corresponds to the system latency, whose cumulative distribution is a nonlinear function of the probability density distributions of the computation times of the tasks and the communication times of the data transfers in the task-flow graph.

The exact computation of the cumulative probability distribution function (c.d.f.) of the completion time is computationally expensive for large PERT networks, therefore it is important to find approaches that approximate the value of the expected time of the completion time and its c.d.f. One of the first of these approaches was due to Fulkerson,[68] who derived an algorithm to find a tight estimate (lower bound) of the expected value of the completion time. Robillrad and Trahan[69] proposed a different method using the characteristic function of the completion time in approximating the c.d.f. of the completion time.

Mehrotra et al.[70] proposed a heuristic for estimating the moments of the probabilistic distribution of the system latency t_c. Kulkarni and Adlakha[71] developed an approach based on Markov processes for the same problem. Hagstrom[72] introduced an exact solution for the problem when the random variables modeling the computation and communication times are finite discrete random variables. Kamburowski[73] developed a tight upper bound on the expected completion time of a PERT network.

An approach using random graphs to model distributed computations was introduced by Indurka et al.,[23] whose theoretical results were improved by Nicol.[24] Purushotaman and Subrahmanyam[74] proposed formal methods applied to concurrent systems with a probabilistic behavior. An example of modeling using queueing networks instead of PERT networks is given by Thomasian and Bay.[75] Estimating errors due to the use of PERT assumptions in scheduling problems is discussed by Lukaszewicz.[76]

Tirat-Gefen developed a set of genetic algorithms using stratified stochastic sampling allowing simultaneous probabilistic optimization of the scheduling and allocation of tasks and communications on application-specific heterogeneous multiprocessor with nonnegligible communication costs.[51]

Performance Bounds Theory and Prediction

Sastry[77] developed a stochastic approach for estimation of wireability (routability) for gate arrays. Kurdahi[78] created a discrete probabilistic model for area estimation of VLSI chips designed according to a standard cell methodology. Küçükçakar[79] introduced a method for partitioning of behavioral specifications onto multiple VLSI chips using probabilistic area/performance predictors integrated into a package called BEST (Behavioral ESTimation). BEST provides a range of prediction techniques that can be applied at the algorithm level and includes references to prior research. These predictors provide information required by Tirat-Gefen's system-level probabilistic optimization methods.[51]

Lower bounds on the performance and execution time of task-flow graphs mapped to a set of available processors and communication links were developed by Liu and Liu[80] for the case of heterogeneous processors but no communication costs and by Hwang et al.[81] for homogeneous processors with communication costs. Tight lower bounds on the number of processors and execution time for the case of homogeneous processors in the presence of communication costs were developed by Al-Mouhamed.[82] Yen and Wolf[83] provide a technique for performance estimation for real-time distributed systems.

At the system and register-transfer level, estimating power consumption by the interconnect is important.[84] Wadekar et al.[85] reported "Freedom," a tool to estimate system energy and power that accounts for functional-resource, register, multiplexer, memory, input/output pads, and interconnect power. This tool employees a statistical estimation technique to associate low-level, technology-dependent, physical and electrical parameters with expected circuit resources and interconnection. At the system level, "Freedom" generates predictions with high accuracy by deriving an accurate model of the load capacitance for the given target technology — a task reported as critical in high level power prediction by Brand and Visweswariah.[86] Methods to estimate power consumption prior to high-level synthesis were also investigated by Mehra and Rabaey.[87] Liu and Svensson[88] reported a technique to estimate power consumption

in CMOS VLSI chips. The reader is referred to an example publication that reports power prediction and optimization techniques at the register transfer level.[89]

Word Length Selection

Many researchers studied word-length optimization techniques at the register-transfer level. A few example publications are cited here. These technique can be classified as statistical techniques applied to digital filters,[90] simulated annealing-based optimization of filters,[91] and simulation-based optimization of filters, digital communication, and signal processing systems.[92] Sung and Kum reported a simulation-based word-length optimization technique for fixed-point digital signal processing systems.[93] The objective of these particular architecture-level techniques is to minimize the number of bits in the design which is related to, but not the same as the overall hardware cost.

References

1. *IEEE Standard VHDL Language Reference Manual,* IEEE Std. 1076, IEEE Press, New York, 1987.
2. Bhasker, J., *A Verilog HDL primer,* Star Galaxy Press, 1997.
3. Bell, G. and Newell, A., *Computer Structures: Readings and Examples,* McGraw Hill, New York, 1971.
4. Harel, D., Statecharts: a visual formalism for complex systems, *Sci. Comput. Progr.*, 8, 231, 1987.
5. Vahid, F., Narayan, S., and Gajski, D. D., SpecCharts: a VHDL front-end for embedded systems, *IEEE Trans. CAD*, 14, 694, 1995.
6. Wadekar, S. A. and Parker, A. C., Accuracy sensitive word-length selection for algorithm optimization, *Proc. Int. Conf. Circuit Design* [*ICCD*], 54, 1998.
7. Wadekar, S. A. and Parker, A. C., Algorithm-level verification of arithmetic-intensive application-specific hardware designs for computation accuracy, in *Digest Third International High Level Design Validation and Test Workshop*, 1998.
8. Gupta, P., Chen, C. T., DeSouza-Batista, J. C., and Parker, A. C., Experience with image compression chip design using unified system construction tools, *Proc. 31st Design Automation Conf.*, 1994.
9. Granacki, J. and Parker, A.C., PHRAN – Span: a natural language interface for system specifications, *Proc. 24th Design Automation Conf.*, 416, 1987.
10. Cyre, W. R. Armstrong, J. R., and Honcharik, A. J., Generating simulation models from natural language specifications, *Simulation*, 65, 239, 1995.
11. Tanir, O. and Agarwal, V. K., A specification-driven architectural design environment, *Computer*, 6, 26, 1995.
12. Gajski, D. D., Vahid, F., Narayan, S., and Gong, J., *Specification And Design of Embedded Systems*, Prentice Hall, Englewood Cliffs, NJ, 1994.
13. de Jong, G. and Lin, B., A communicating Petri net model for the design of concurrent asynchronous modules, *ACM/IEEE Design Automation Conf.*, June 1994.
14. Srivastava, M. B., Richards, B. C., and Broderson, R. W., System level hardware module generation, *IEEE Trans. Very Large Scale Integration [VLSI] Syst.*, 3, 20, 1995.
15. Lagnese, E. and Thomas, D., Architectural partitioning for system level synthesis of integrated circuits, *IEEE Trans. Comput.-Aided Design,* 1991.
16. Vahid, F., *A Survey of Behavioral-Level Partitioning Systems*, Technical Report TR ICS 91-71, University of California, Irvine, CA, 1991.
17. Kucukcakar, K. and Parker, A.C., Chop: a constraint-driven system-level partitioner, *Proc. 28th Design Automation Conf.*, 514, 1991.
18. Vahid, F. and Gajski, D. D., Specification partitioning for system design, *Proc. 29th Design Automation Conf.*, 1992.
19. Parker, A. C., Chen, C.-T., and Gupta, P., Unified system construction, *Proc. SASIMI Conf.*, 1993.
20. Bokhari, S. H., *Assignment problems in parallel and distributed computing,* Kluwer Academic Publishers, 1987.

21. Stone, H. S. and Bokhari, S. H., Control of distributed processes, *Computer*, 11, 97, 1978.
22. Haddad, E. K., *Optimal Load Allocation for Parallel and Distributed Processing*, Technical Report TR 89-12, Department of Computer Science, Virginia Polytechnic Institute and State University, April 1989.
23. Indurkhya, B., Stone, H. S., and Cheng, L. X., Optimal partitioning of randomly generated distributed programs, *IEEE Trans. Software Eng.,* SE-12, 483, 1986.
24. Nicol, D. M., Optimal partitioning of random programs across two processors, *IEEE Trans. Software Eng.,* 15, 134, 1989.
25. Lee, C. Y., Hwang, J. J., Chow, Y. C., and Anger, F. D., Multiprocessor scheduling with interprocessor communication delays, *Operations Res. Lett.,* 7, 141, 1988.
26. Tirat-Gefen, Y.G., Silva, D. C., and Parker, A. C., Incorporating imprecise computation into system-level design of application-specific heterogeneous multiprocessors, in *Proc. 34th. Design Automation Conf.*, 1997.
27. DeSouza-Batista, J. C. and Parker, A. C., Optimal synthesis of application specific heterogeneous pipelined multiprocessors, *Proc. Int. Conf. Appl.-Specific Array Process.*, 1994.
28. Mehrotra, R. and Talukdar, S. N., *Scheduling of Tasks for Distributed Processors*, Technical Report DRC-18-68-84, Design Research Center, Carnegie-Mellon University, December 1984.
29. Agrawal, R. and Jagadish, H. V., Partitioning techniques for large-grained parallelism, *IEEE Trans. Comput.,* 37, 1627, 1988.
30. Barthou, D., Gasperoni, F., and Schwiegelshon, U., Allocating communication channels to parallel tasks, in *Environments and Tools for Parallel Scientific Computing*, Elsevier Science Publishers B.V., 275, 1993.
31. Linsky, V. S. and Kornev, M. D., Construction of optimum schedules for parallel processors, *Eng. Cybernet.,* 10, 506, 1972.
32. Chu, W. W., Hollaway, L.J., and Efe, K., Task allocation in distributed data processing, *Computer*, 13, 57, 1980.
33. Prakash, S. and Parker, A. C., SOS: synthesis of application specific heterogeneous multiprocessor systems, *J. Parallel Distrib. Comput.,* 16, 338, 1992.
34. Prakash, S., Synthesis of Application-Specific Multiprocessor Systems, Ph.D. thesis, Department of Electrical Engineering and Systems, University of Southern California, Los Angeles, January 1994.
35. Hafer, L. and Parker, A., Automated synthesis of digital hardware, *IEEE Trans. Comput.,* C-31, 93, 1981.
36. Fernandez, E. B. and Bussel, B., Bounds on the number of processors and time for multiprocessor optimal schedules, *IEEE Trans. Comput.,* C-22, 745, 1975.
37. Garey, M. R. and Graham, R. L., Bounds for multiprocessor scheduling with resource constraints, *SIAM J. Comput.,* 4, 187, 1975.
38. Jaffe, J. M., Bounds on the scheduling of typed task systems, *SIAM J. Comput.,* 9, 541, 1991.
39. Gupta, R. and Zorian, Y., Introducing core-based system design, *IEEE Design Test Comput.,* Oct.-Dec., 15, 1997.
40. Li, Y. and Wolf, W., A task-level hierarchical memory model for system synthesis of multiprocessors, *Proc. Design Automation Conference*, 1997, 153.
41. *Design and Test*, special issue on rapid prototyping, 13,3, 1996.
42. Birmingham, W. and Siewiorek, D., MICON: a single board computer synthesis tool, *Proc. 21st Design Automation Conf.*, 1984.
43. Chen, C-T, System-Level Design Techniques and Tools for Synthesis of Application-Specific Digital Systems, Ph.D. thesis, Department of Electrical Engineering and Systems, University of Southern California, Los Angeles, January 1994.
44. Heo, D. H., Ravikumar, C. P., and Parker, A., Rapid synthesis of multi-chip systems, *Proc. 10th Int. Conf. VLSI Design*, 62, 1997.
45. Lee, E. A. and Bier, J. C., Architectures for statically scheduled dataflow, *J. Parallel Distrib. Comput.,* 10, 333, 1990.

46. Kalavede, A., Pino, J. L., and Lee, E. A., Managing complexity in heterogeneous system specification, simulation and synthesis, *Proc. Int. Conf. Acoustics, Speech, Signal Process. (ICASSP),* May, 1995.
47. Gajski, D. D., Vahid, F., and Narayan, S., A design methodology for system-specification refinement, *Proc. European Design Automation Conf.*, 458, 1994.
48. Renfors, M. and Neuvo, Y., The maximum sampling rate of digital filters under hardware speed constraints, *IEEE Trans. Circuits Syst.,* CAS-28, 196, 1981.
49. Wang, D. J. and Hu, Y. H., Multiprocessor implementation of real-time DSP algorithms, *IEEE Trans. Very Large Scale Integration (VLSI) Syst.,* 3, 393, 1995.
50. Gelabert, P. R. and Barnwell, T. P., Optimal automatic periodic multiprocessor scheduler for fully specified flow graphs, *IEEE Trans. Signal Process.,* 41, 858, 1993.
51. Tirat-Gefen, Y.G., Theory and Practice in System-Level of Application Specific Heterogeneous Multiprocessors, Ph.D. dissertation, Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, 1997.
52. CasteloVide-e-Souza, Y.G., Potkonjak, M., and Parker, A.C., Optimal ILP-based approach for throughput optimization using algorithm/architecture matching and retiming, *Proc. 32nd Design Automation Conf.*, June 1995.
53. Verhauger, W.F., Multidimensional Periodic Scheduling, Ph.D. thesis, Eindhoven University of Technology, Holland, 1995.
54. Passos, N. L., Sha, E. H., and Bass S. C., Optimizing DSP flow-graphs via schedule-based multi-dimensional retiming, *IEEE Trans. Signal Process.,* 44, 150, 1996.
55. Feng, D. T. and Shin, K. G., Static allocation of periodic tasks with precedence constraints in distributed real-time systems, *Proc. 9th Int. Conf. Distrib. Comput.*, 190, 1989.
56. Ramamritham, K., Allocation and scheduling of precedence-related periodic tasks, *IEEE Trans. Parallel Distrib. Syst.,* 6, 1995.
57. Ramamritham, K., Stankovic, J. A., and Shiah, P.F., Efficient scheduling algorithms for real-time multiprocessors systems, *IEEE Trans. Parallel Distrib. Syst.,* 1, 184, 1990.
58. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin, 1994.
59. Hou, E.S.H, Ansari, N., and Ren, H., A Genetic algorithm for multiprocessor scheduling, *IEEE Trans. Parallel Distrib. Syst.,* 5, 113, 1994.
60. Wang, L., Siegel, H. J., and Roychowdhury, V. P., A genetic-algorithm-based approach for task matching and scheduling in heterogeneous computing environments, *Proc. Heterogeneous Comput. Workshop, Int. Parallel Process. Symp.*, 72, 1996.
61. Ravikumar, C. P. and Gupta, A., Genetic algorithm for mapping tasks onto a reconfigurable parallel processor, *IEE Proc. Comput. Digital Tech.*, 142, 81, 1995.
62. Tirat-Gefen, Y. G. and Parker, A. C., MEGA: an approach to system-level design of application-specific heterogeneous multiprocessors, *Proc. Heterogeneous Comput. Workshop, Int. Parallel Process. Symp.*, 105, 1996.
63. Liu, J. W. S., Lin, K.-J., Shih, W.-K., Yu, A. C.-S., Chung, J.-Y., and Zhao, W., Algorithms for scheduling imprecise computations, *IEEE Comput.*, 24, 58, 1991.
64. Ho, K., Leung, J. Y-T. and Wei, W-D., *Minimizing Maximum Weighted Error for Imprecise Computation Tasks*, Technical Report UNL-CSE-92-017, Department of Computer Science and Engineering, University of Nebraska, Lincoln, 1992.
65. Tirat-Gefen, Y. G., Silva, D. C., and Parker, A. C., Incorporating imprecise computation into system-level design of application-specific heterogeneous multiprocessors, *Proc. 34th. Design Automation Conf.*, 1997.
66. Malcolm, D. G., Roseboom, J. H., Clark, C. E., and Fazar, W., Application of a technique for research and development program evaluation, *Oper. Res.*, 7, 646, 1959.
67. Elmaghraby, S. E., The theory of networks and management science: part II, *Manage. Sci.*, 17, B.54, 1970.
68. Fulkerson, D. R., Expected critical path lengths in pert networks, *Oper. Res.*, 10, 808, 1962.

69. Robillard, P. and Trahan, M., The completion time of PERT networks, *Oper. Res.,* 25, 15, 1977.
70. Mehrotra, K., Chai, J., and Pillutla, S., *A Study of Approximating the Moments of the Job Completion Time in PERT Networks*, Technical Report, School of Computer and Information Science, Syracuse University, New York, 1991.
71. Kulkarni, V. G. and Adlakha, V. G., Markov and Markov-regenerative pert networks, *Oper. Res.,* 34, 769, 1986.
72. Hagstrom, J. N., Computing the probability distribution of project duration in a PERT network, *Networks*, 20, John Wiley & Sons, New York, 1990, 231.
73. Kamburowski, J., An upper bound on the expected completion time of PERT networks, *Eur. J. Oper. Res.,* 21, 206, 1985.
74. Purushothaman, S. and Subrahmanyam, P. A., Reasoning about probabilistic behavior in concurrent systems, *IEEE Trans. Software Eng.,* SE-13, 740, 1987.
75. Thomasian, A., Analytic queueing network models for parallel processing of task systems, *IEEE Trans. Comput.,* C-35, 1045, 1986
76. Lukaszewicz, J., On the estimation of errors introduced by standard assumptions concerning the distribution of activity duration in pert calculations, *Oper. Res.,* 13, 326, 1965.
77. Sastry, S. and Parker, A. C., Stochastic models for wireability analysis of gate arrays, *IEEE Trans. Comput.-Aided Design,* CAD-5, 1986.
78. Kurdahi, F. J., Techniques for area estimation of VLSI layouts, *IEEE Trans. Comput.-Aided Design,* 8, 81, 1989.
79. Küçükçakar, K. and Parker, A. C., A methodology and design tools to support system-level VLSI design, *IEEE Trans. Very Large Scale Integration [VLSI] Syst.,* 3, 355, 1995.
80. Liu, J. W. S. and Liu, C. L., Performance analysis of multiprocessor systems containing functionally dedicated processors, *Acta Informatica,* 10, 95, 1978
81. Hwang, J. J., Chow, Y. C., Ahnger, F. D., and Lee, C. Y., Scheduling precedence graphs in systems with interprocessor communication times, *SIAM J. Comput.,* 18, 244, 1989.
82. Mouhamed, M., Lower bound on the number of processors and time for scheduling precedence graphs with communication costs, *IEEE Trans. Software Eng.,* 16, 1990.
83. Yen, T.-Y. and Wolf, W., Performance estimation for real-time embedded systems, *Proc. Int. Conf. Comput. Design*, 64, 1995.
84. Landman, P. E. and Rabaey, J. M., Activity-sensitive architectural power analysis, *IEEE Trans. on CAD*, 15, 571, 1996.
85. Wadekar, S. A., Parker, A. C., and Ravikumar, C. P., FREEDOM: statistical behavioral estimation of system energy and power, *Proc. Eleventh Int. Conf. on VLSI Design*, 30, 1998.
86. Brand, D. and Visweswariah, C., Inaccuracies in power estimation during logic synthesis, *Proc. Eur. Design Automation Conf. (EURO-DAC),* 388, 1996.
87. Mehra, R. and Rabaey, J., Behavioral level power estimation and exploration, *Proc. First Int. Workshop Low Power Design*, 197, 1994.
88. Liu, D. and Svensson, C., Power consumption estimation in CMOS VLSI chips, *IEEE J. Solid-State Circuits*, 29, 663, 1994.
89. Landman, P. E. and Rabaey, J. M., Activity-sensitive architectural power analysis, *IEEE Trans. Comput.-Aided Design,* 15, 571, 1996.
90. Zeng, B. and Neuvo, Y., Analysis of floating point roundoff errors using dummy multiplier coefficient sensitivities, *IEEE Trans. Circuits Syst.,* 38, 590, 1991.
91. Catthoor, F., Vandewalle, J., and De Mann, H., Simulated annealing based optimization of coefficient and data word lengths in digital filters, *Int. J. Circuit Theor. Appl.,* 16, 371, 1988.
92. Grzeszczak, A., Mandal, M. K., Panchanathan, S. and Yeap, T., VLSI implementation of discrete wavelet transform, *IEEE Trans. VLSI Syst.,* 4, 421, 1996.
93. Sung, W. and Kum, Ki-II., Simulation-based word-length optimization method for fixed-point digital signal processing systems, *IEEE Trans. Signal Process.,* 43, 3087, 1995.

Bhasker, J. "Synthesis at the Register Transfer Level and the Behavioral Level"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

75

Synthesis at the Register Transfer Level and the Behavioral Level

J. Bhasker
Cadence Design Systems

75.1 Introduction

This chapter provides an overview of register transfer level synthesis and behavioral synthesis, contrasting the two with examples. Examples are written using VHDL and Verilog HDL, the two dominant hardware description languages (HDL) in the industry today.

The chapter intends to be more of a tutorial. It first describes the distinguishing characteristics of register transfer level (RTL) modeling as opposed to behavioral level modeling. It then uses both HDLs to illustrate how RTL models are mapped to hardware. Both combinational logic synthesis and sequential logic synthesis are presented. This includes how flip-flops, latches, and three-state gates are inferred from the RTL model. A finite state machine modeling example is also described. The later part of the chapter shows the behavioral synthesis methodology with examples to illustrate the flow of transformations that occur during the synthesis process. Many scheduling and resource allocation algorithms exist today. In this chapter, we illustrate the basic ideas behind the algorithms. Examples are used to show the architectural exploration that can be performed with behavioral synthesis, something that is not possible with register transfer level synthesis.

Synthesis is here! It has become an integral part of every design process. Once upon a time, all circuits were designed by hand and logic gates and their interconnections were entered into a system using a schematic capture tool. This is no longer the norm. More and more designers are resorting to synthesis because of the tremendous advantages that it provides, for example, describing the design at a higher level of abstraction. To this end, a language is needed to describe the design. This is where a hardware description language comes in. A hardware programming language is a formal language designed with the intent of describing hardware. Additionally, each language construct has a functional semantic

associated with it that can be used to verify a design described in HDL (a design described in a HDL is often called a "model"). The model also serves as a means of documenting the design.

75.2 The Two HDL's

The two dominant hardware description languages in use today are

i VHDL
ii Verilog HDL

VHDL originated from the Department of Defense through its VHSIC program and became a public domain standard in 1987, whereas Verilog HDL originated from a private company and became a public domain standard in 1995. Both languages are targeted at describing digital hardware. A design can be expressed structurally, in a dataflow style or in a sequential behavior style. The key difference between the two languages is that VHDL extends the modeling to higher levels of data abstraction, provides for strong type checking, and supports the delta delay mechanism.

An excellent introduction to both languages can be found in (Bhasker, 1995) and (Bhasker, 1997). The complete descriptions of languages can be found in their respective language reference manuals (LRMs), (IEEE, 1993) and (IEEE, 1995).

Here is an example of a simple arithmetic logic unit described using both languages. The design is described using a mixed style — it contains structural components, dataflow, and sequential behavior.

```
-- VHDL:
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity ALU is
  port (A, B: in UNSIGNED(3 downto 0);
        SEL: in STD_LOGIC_VECTOR(0 to 1);
        Z: out UNSIGNED(7 downto 0);
        ZComp: out BOOLEAN);
end;
architecture MIXED_STYLE of ALU is
  component MULTIPLIER
    port (PortA, PortB: in UNSIGNED (3 downto 0);
          PortC: out UNSIGNED (7 downto 0));
  end component;
  signal MulZ: UNSIGNED (7 downto 0);
begin
  ZComp <= A < B when SEL = "11" else FALSE;
  M1: MULTIPLIER port map (PortA => A, PortB => B,
                           PortC => MulZ);
  process (A, B, SEL, MulZ)
  begin
    Z <= (others => '0');
    case SEL is
      when "00" => Z(3 downto 0) <= A + B;
      when "01" => Z(3 downto 0) <= A – B;
      when "10" => Z < = MulZ;
      when others => Z <= (others => 'Z');
    end case;
  end process;
end;
```

```
//Verilog:
module ALU (A, B, SEL, Z, ZComp);
  input [3:0] A, B;
  input [0:1] SEL;
  output [7:0] Z;
  reg [7:0] Z;
  output ZComp;
  assign ZComp = (SEL == 2'b11) ? A < B: 'b0;
  MULTIPLIER M1 (.PortA(A), .PortB(B), .PortC(MulZ));
  always @(A or B or SEL)
  begin
    case (SEL)
      2'b00: Z = A + B;
      2'b01: Z = A – B;
      2'b10: Z = MulZ;
      default: Z = 'bz;
    endcase
  end
endmodule
```

In VHDL, the interface of the design (entity declaration) is separate from the description of the design (architecture body). Note that each signal is declared as a specific type (UNSIGNED). This type is declared in the package NUMERIC_STD, which in turn is included in the design using the context clauses (library and use clause). The structural part is described using a component instantiation statement — a component declaration is required to specify the interface for the component. The dataflow part is specified using a concurrent signal assignment. The sequential part is specified using a process statement; this contains a case statement that switches to an appropriate branch based on the value of the case expression.

In the Verilog model, each variable can have at most four values: 0, 1, x, and z. The model shows the two main data types in Verilog: net and register (a wire is a net data type, while a reg is a register data type). The structural part is described using a module instantiation statement. Notice that named association is used to specify the connection between the ports of the module and its external nets to which they are connected. Dataflow part is modeled using the continuous assignment statement, while the sequential part is represented using the always statement.

In this chapter, we shall use both of the languages to illustrate the examples when describing synthesis.

75.3 The Three Different Domains of Synthesis

There are three distinct domains in synthesis:

i logic synthesis
ii RTL synthesis
iii behavioral synthesis

But first, the definition (at least the author's) of synthesis: Synthesis is the process of transforming an HDL description of a design into logic gates. The synthesis process itself, starting from HDL, involves a number of tasks that need to be performed. These tasks may or may not be distinct in synthesis tools (Fig. 75.1).

Starting from an HDL description, synthesis generates a technology-independent RTL level netlist (RTL blocks interconnected by nets). Based on the target technology and design constraints, such as area and delay, the module builder generates a technology-specific gate level netlist. A logic optimizer further optimizes the logic to match the design constraints and goals such as area and delay. The synthesis process may bypass the module build phase and directly generate a gate level netlist if there are no RTL blocks in the design.

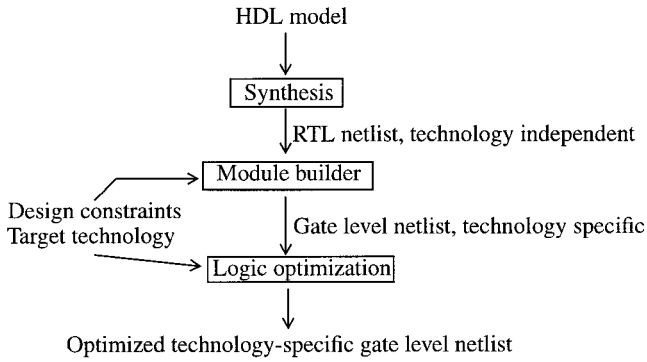


FIGURE 75.1 The tasks involved in synthesis process.

In this chapter, we will not discuss logic optimization and module building. One source that describes the algorithms behind logic optimization is (DeMicheli, 1994).

Coming back to the three different synthesis domains, let us briefly explore them. The level of abstraction increases as we go from the logic level to the behavioral level. Inversely, the structural inference reduces as we go from the logic level to the behavioral level (Fig. 75.2).

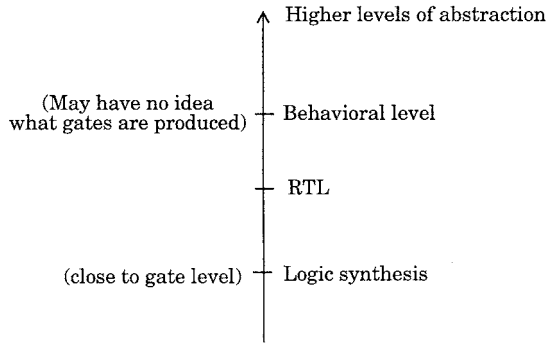


FIGURE 75.2 Varying levels of abstraction.

In the logic synthesis domain, a design is described in terms of Boolean equations. Components may be instantiated to describe hierarchy or may lower level primitives such as flip-flops. Here is a logic synthesis model for an incrementor whose output is latched.

```
-- VHDL:
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
entity INCREMENT is
  port (A: in UNSIGNED(0 to 2); CLOCK: in STD_LOGIC;
        Z: out UNSIGNED(0 to 2));
end;

architecture LOGIC_LEVEL of INCREMENT is
  component FD1S3AX
    port (DATA, CLK: in STD_LOGIC; Q: out STD_LOGIC);
  end component;
  signal DZ1, DZ2, DZ0, S429, A1BAR: STD_LOGIC;
```

```
begin
  DZ1 <= not ((A(1) or DZ2) and (A(2) or A1BAR));
  DZ2 <= not A(2);
  DZ0 <= not ((A(0) or DZ2) and (A1BAR or S429));
  A1BAR <= not A(1);
  S429 <= not ((DZ2 or A1BAR) and A(0));

  S0: FD1S3AX port map (DZ2, CLOCK, Z(2));
  S1: FD1S3AX port map (DZ1, CLOCK, Z(1));
  S2: FD1S3AX port map (DZ0, CLOCK, Z(0));
end;

//Verilog:
module INCREMENT (A, CLOCK, Z);
  input [0:2] A;
  input CLOCK;
  output [0:2] Z;

  wire A1BAR, S429, DZ0, DZ1, DZ2;

  assign DZ1 = ! ((A[1] || DZ2) && (A[2] || A1BAR));
  assign DZ2 = ! A[2];
  assign DZ0 = ! ((A[0] || DZ2) && (A1BAR || S429));
  assign A1BAR = ! A[1];
  assign S429 = ! ((DZ2 || A1BAR) && A[0]);

  FD1S3AX     S0 (DZ2, CLOCK, Z[2]),
              S1 (DZ1, CLOCK, Z[1]),
              S2 (DZ0, CLOCK, Z[0]);
endmodule
```

Notice that all operators are logical operators; these implicitly specify the structure of the synthesized netlist (Fig. 75.3). For example, the `and` operator produces an `and` gate, the `or` operator produces an `or` gate, and so on. The component instantiations (module instantiations in Verilog) instantiate three flip-flops.

In the register transfer level synthesis domain, the arithmetic operations are modeled using language operators and the behavior is described using the language behavioral constructs. The design is described in terms of data transfers between arithmetic logic units and registers, possibly at explicitly defined clock edges. Moreover, registers may be inferred implicitly from the design. In addition, hierarchy can be described by instantiating other RTL descriptions. RTL blocks, such as a custom-built multiplier, may be directly instantiated.

The key difference from logic synthesis is that in the register transfer level case, the behavior is described using higher level operators such as + and –, and constructs such as if and case, that are provided by the hardware description language. In addition, for sequential circuits, the behavior of the design at each clock edge is explicitly described. Here is the same example of the incrementor, this time at the register transfer level:

```
-- VHDL:
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
entity INCREMENT is
  port (A: in UNSIGNED(0 to 2); CLOCK: in STD_LOGIC;
        Z: out UNSIGNED(0 to 2));
end INCREMENT;
```

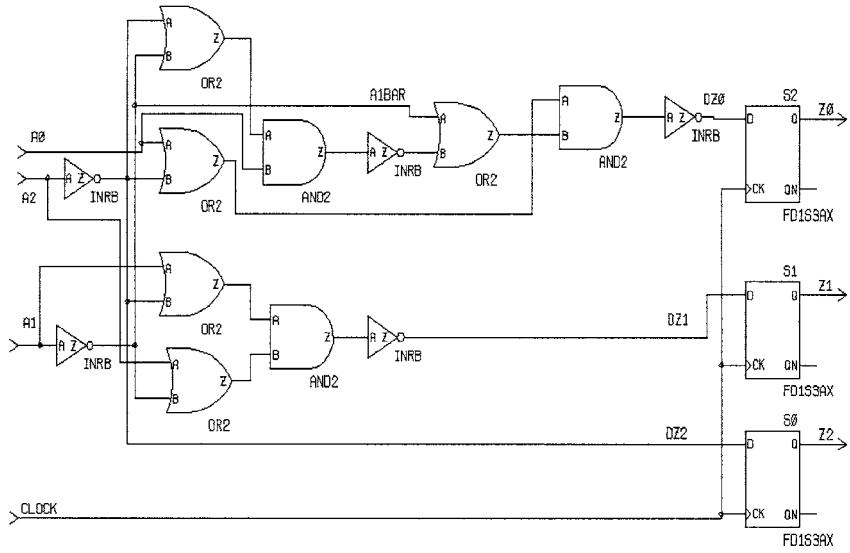


FIGURE 75.3 The synthesized netlist.

```
//The above entity declaration is identical to the
//logic level case.

architecture REGISTER_TRANSFER_LEVEL of INCREMENT is
  PZ: process
  begin
    wait until CLOCK = '1';
    Z <= A + 1;
  end process PZ;
end REGISTER_TRANSFER_LEVEL;

//Verilog:
module INCREMENT (A, CLOCK, Z);
  input [0:2] A;
  input CLOCK;
  output [0:2] Z;
  reg [0:2] Z;

    always @(posedge CLOCK)
    Z <= A + 1;
endmodule
```

Notice that flip-flops are not explicitly instantiated but are inferred. Inference rules are used to infer flip-flops. One such rule is that if any variable or signal (a reg in Verilog HDL) is assigned a value under the control of a clock edge (as shown using a wait on a clock edge in a process statement for VHDL and using a posedge event in an always statement for Verilog), then the variable or signal is inferred as a flip-flop. Also note that in the RTL description, the behavior of the design at every clock edge is explicitly specified. In the behavioral synthesis domain, this is one level above register transfer level synthesis, no clocking information is specified in the model (this is not completely true as we will see later when we talk about partially scheduled and fully scheduled models). Inference rules are not used to infer registers. The behavior of the design is described as a sequential program, and the synthesis tool's job is to figure

out which operations are to be done in which clock cycle based on the design constraints that are specified. Any variable whose value needs to be saved between clock cycles will have to be stored in a register.

Here is a simple example used to illustrate behavioral synthesis:

```
-- VHDL:
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity SIMPLE_TREE is
  port (InData: in UNSIGNED(0 to 3);
        OutData: out UNSIGNED(0 to 3));
end SIMPLE_TREE;

architecture ONLY_TO_EXPLAIN of SIMPLE_TREE is
begin
  P1: process (InData)
    variable Tree: UNSIGNED (0 to 3);
  begin
    Tree: = InData + 1;
    Tree: = Tree + 1;
    OutData <= Tree + 1;
  end process P1;
end ONLY_TO_EXPLAIN;

//Verilog:
module SIMPLE_TREE (InData, OutData);
  input [0:3] InData;
  output [0:3] OutData;
  reg [0:3] OutData;
  reg [0:3] Tree;

  always @(InData)
  begin
    Tree = InData + 1;
    Tree = Tree + 1;
    OutData <= Tree + 1;
  end
endmodule
```

Notice that no clocking information, such as clock edges, is explicitly present in the model. Now if we assume that an adder (the addition operator) takes 5 ns to compute and assume a clock period of 6 ns, we can see that it will take three clock cycles to completely execute the behavior. In addition, all the three "+" operators can be shared using only one adder. Also, only one register *Tree* is required to save the intermediate values (the value between clock cycles). Figure 75.4 shows one architecture (RTL netlist) generated under the given assumptions. The controller sequencing is shown in Fig. 75.5.

If a clock period of 11 ns is specified, then only two clock cycles are required to complete execution. See Fig. 75.6 for the controller sequencing. This time two adders are required. Figure 75.7 shows the new architecture.

By changing the constraints, different architectures for the same behavior are produced — this is called "architectural exploration." Such generation of different architectures is not possible with RTL synthesis, since the architecture of the design is explicitly specified in the RTL model.

We describe register transfer level synthesis and behavioral synthesis in further detail in the next two sections.

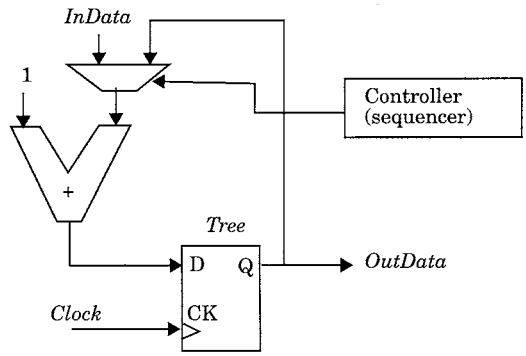


FIGURE 75.4 One architecture with a clock cycle of 6 ns.

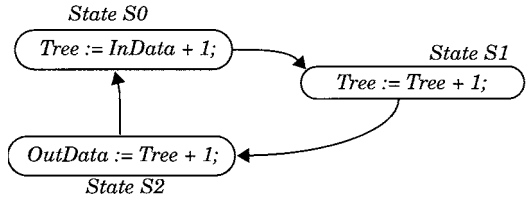


FIGURE 75.5 Controller sequencing.

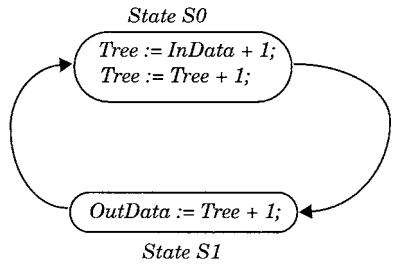


FIGURE 75.6 Controller sequencing with new specification.

75.4 RTL Synthesis

In describing an RTL model, different design styles are used for modeling combinational logic and for modeling sequential logic.

Combinational Logic

Combinational logic can be described using

i concurrent signal assignment (Continuous assignment in Verilog)
ii process statement with no clock edges (always statement with no edge event in event list for Verilog).

Language operators such as + and –, and behavioral constructs such as if, case, and loop statements, may be used to model the design. Here is an example of a BCD to seven-segment decoder:

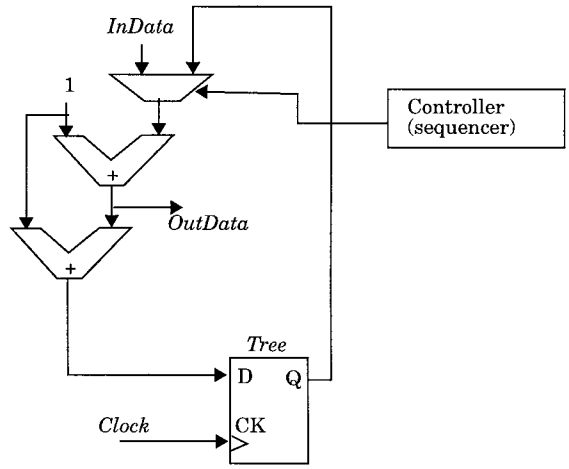


FIGURE 75.7 Another architecture with a clock period of 11 ns.

```
-- VHDL:
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity BCD_TO_7 is
  port (D: in STD_LOGIC_VECTOR(0 to 3);
        Q: out STD_LOGIC_VECTOR(0 to 6));
end BCD_TO_7;

architecture RTL_MODEL of BCD_TO_7 is
begin
  P1: process (D)
  begin
    case D is
      when "0000" => Q <= "1111110";
      when "0001" => Q <= "00011000";
      when "0010" => Q <= "10110111";
      when "0011" => Q <= "00111111";
      when "0100" => Q <= "01011101";
      when "0101" => Q <= "01101111";
      when "0110" => Q <= "11001111";
      when "0111" => Q <= "00111100";
      when "1000" => Q <= "11111111";
      when "1001" => Q <= "01111101";
      when others => Q <= "--------";
    end case;
  end process P1;
end RTL_MODEL;

//Verilog:
module BCD_TO_7 (D, Q);
  input [0:3] D;
  output [0:6] Q;
  reg [0:6] Q;
```

```
  always @(D)
    case (D)
      4'd0: Q = 7'b1111110;
      4'd1: Q = 7'b00011000;
      4'd2: Q = 7'b10110111;
      4'd3: Q = 7'b00111111;
      4'd4: Q = 7'b01011101;
      4'd5: Q = 7'b01101111;
      4'd6: Q = 7'b11001111;
      4'd7: Q = 7'b00111100;
      4'd8: Q = 7'b11111111;
      4'd9: Q = 7'b01111101;
      default: Q = 7'bx;
    endcase
endmodule
```

It is required that all signals (variables in Verilog) read within the process statement (always statement) must appear in the sensitivity list (event list for Verilog), or else there may be a possibility for a functional mismatch between the RTL model and its synthesized netlist. Some synthesis tools issue a warning about such missing signals.

Sequential Logic

Edge-sensitive storage elements (flip-flops) are modeled differently from level-sensitive storage elements (latches). First, let us take a look at how level-sensitive storage devices are modeled.

Modeling a Level-Sensitive Storage Element

When a variable or a signal is not explicitly assigned a value in every iteration of a process statement (or an always statement in Verilog), a level-sensitive storage element is inferred. For example, this can happen if a variable is not assigned a value in all branches of an if statement.

```
-- VHDL:
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity LSSE is
  port (D, Clock: in STD_LOGIC; Q: out STD_LOGIC);
end LSSE;

architecture LATCH_INFERENCE of LSSE is
begin
  process (Clock, D)
  begin
    if (Clock = '1') then
      Q <= D;
    end if;
  end process;
end LATCH_INFERENCE;

//Verilog:
module LSSE (D, Clock, Q);
  input D, Clock;
  output Q;
  reg Q;
```

```
  always @(D or Clock)
    if (Clock)
      Q = D;
endmodule
```

Based on the language semantics, *Q* needs to retain its value during the period when *Clock* is 0. This is achieved by implementing a level-sensitive storage element for *Q*. This is shown in Fig. 75.8, which shows the synthesized netlist (FD1S1A is a latch).

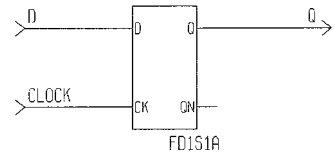


FIGURE 75.8 Synthesizing a level-sensitive storage element.

Modeling an Edge-Sensitive Storage Element

A signal or a variable assigned a value under the control of a clock edge is inferred as a flip-flop.

In VHDL: Here is a template of a process statement that has to be used to infer edge-sensitive storage elements.

```
process
  declarations, if any
begin
  wait until <clock_edge>
  <sequential_statements>
end process;
-- <clock_edge> can be either of:
--  Signal = '1' for a rising edge
--  Signal = '0' for a falling edge.
```

In Verilog: Here is a template of the always statement.

```
always @(<edge_event>)
  <statement>
//<edge_event> is:
//  "posedge VariableName" for positive edge,
//  "negedge VariableName" for negative edge.
```

Here is an example.

```
-- VHDL:
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;
entity ESSE is
  port (Clk: in STD_LOGIC; D: in UNSIGNED (0 to 2);
        Q: out UNSIGNED(0 to 2));
end ESSE;

architecture FF_INFERENCE of ESSE is
begin
  process
  begin
    wait until Clk = '0';
    Q <= D + 1;
  end process;
end FF_INFERENCE;

//Verilog:
module ESSE (Clk, D, Q);
```

```
  input Clk;
  input [0:2] D;
  output [0:2] Q;
  reg [0:2] Q;

  always @(negedge Clk)
    Q <= D + 1;
endmodule
```

Since *Q* is assigned a value under the control of a clock edge, an edge-sensitive storage element is inferred for *Q*. This is shown in the synthesized netlist (Fig. 75.9).

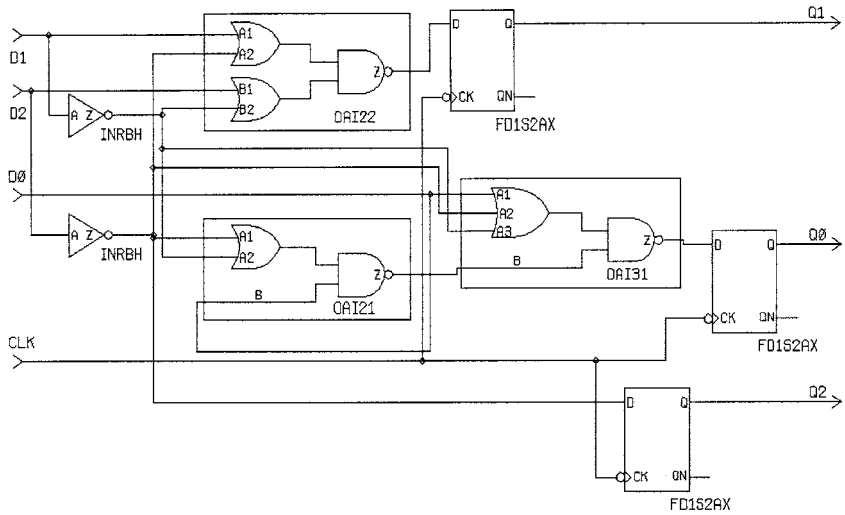


FIGURE 75.9 Inferring a flip-flop.

75.5 Modeling a Three-State Gate

A three-state gate is modeled at the register transfer level by assigning the high-impedance value to a signal or variable under the control of a condition.

```
-- VHDL:
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity THREE_STATE is
  generic (NBITS: POSITIVE:= 3);
  port (Enable: in STD_LOGIC;
        MemDataReg: in STD_LOGIC_VECTOR(1 to NBITS);
        ReadBus: out STD_LOGIC_VECTOR (1 to NBITS));
end THREE_STATE;

architecture INFERENCE of THREE_STATE is
begin
  process (Enable, MemDataReg)
  begin
    if (Enable = '0') then
      ReadBus <= (others => 'Z');
    else
      ReadBus <= MemDataReg;
```

```
    end if;
  end process;
end INFERENCE;

//Verilog:
module THREE_STATE (Enable, MemDataReg, ReadBus);
  parameter NBITS = 3;
  input Enable;
  input [1:NBITS] MemDataReg;
  output [1:NBITS] ReadBus;
  reg [1:NBITS] ReadBus;

  always @(Enable or MemDataReg)
    if (! Enable)
      ReadBus = 'bz;
    else
      ReadBus = MemDataReg;
endmodule
```

The target *ReadBus* is assigned the high impedance value under the control of the condition "*Enable* = 0". The synthesized netlist contains three three-state gates with the condition "*Enable* = 0" controlling the three-state behavior of the gate and *MemDataReg* as the input to the three-state gate (see Fig. 75.10 for the synthesized netlist).

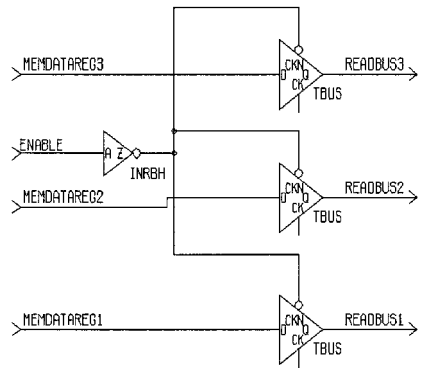


FIGURE 75.10 Synthesizing a three-state gate.

75.6 An Example

Here is an example of a synthesis model. There exists a mango juice dispenser that dispenses mango juice cartons for 25¢ each. The machine accepts only coins (quarters, dimes, and nickels), and appropriate change is to be returned. All state transitions are synchronized to the positive edge of a clock. The state transition diagram is shown as shown in Fig. 75.11.

```
-- VHDL:
entity MANGO_JUICE is
  port (CoinIn: in POSITIVE; Clock, Reset: in BIT;
        DispenseJuice: out BOOLEAN;
        NickelOut, DimeOut: out NATURAL);
end MANGO_JUICE;
```

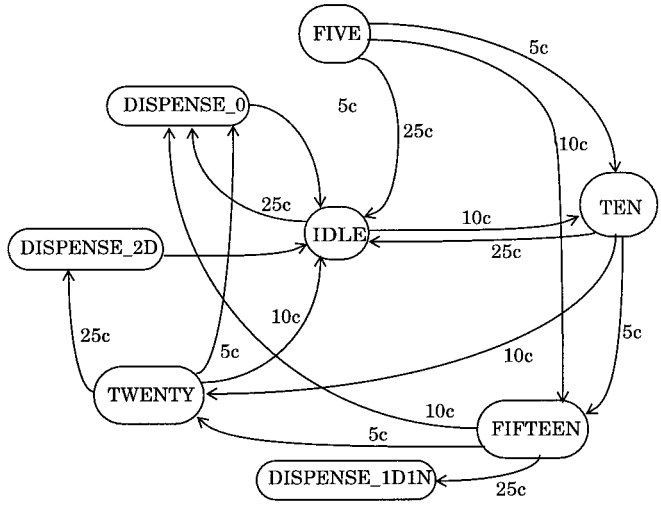


FIGURE 75.11 State transition for mango juice dispensing machine.

```
architecture DISPENSER of MANGO_JUICE is
  type STATES is (IDLE, FIVE, TEN, FIFTEEN, TWENTY);
  signal NextState, CurrentState: STATES;
begin
  process (Clock, Reset, NextState)
  begin
    if Reset = '0' then
      CurrentState <= IDLE;
    elsif (Clock = '0' and Clock'EVENT) then
      CurrentState <= NextState;
    end if;
  end process;

  process (CoinIn, CurrentState)
  begin
    NextState <= CurrentState;
    DispenseJuice <= FALSE;
    NickelOut <= 0;
    DimeOut <= 0;

    case CurrentState is
      when IDLE =>
        case CoinIn is
          when 5 => NextState <= FIVE;
          when 10 => NextState <= TEN;
          when 25 =>
            NextState <= IDLE;
            DispenseJuice <= TRUE;
          when others => NextState  <= IDLE;
        end case;
      when FIVE =>
        case CoinIn is
          when 5 => NextState <= TEN;
```

```
            when 10 => NextState <= FIFTEEN;
            when 25 =>
              NextState <= IDLE;
              DispenseJuice <= TRUE;
              NickelOut <= 1;
            when others => NextState <= IDLE;
          end case;
        when TEN =>
          case CoinIn is
            when 5 => NextState <= FIFTEEN;
            when 10 => NextState <= TWENTY;
            when 25 =>
              NextState <= IDLE;
              DispenseJuice <= TRUE;
              DimeOut <= 2;
            when others => NextState <= IDLE;
          end case;
        when FIFTEEN =>
          case CoinIn is
            when 5 => NextState <= TWENTY;
            when 10 =>
              NextState <= IDLE;
              DispenseJuice <= TRUE;
            when others => NextState <= IDLE;
          end case;
        when TWENTY =>
          NextState <= IDLE;
          case CoinIn is
            when 5 => DispenseJuice <= TRUE;
            when 10 =>
              DispenseJuice <= TRUE;
              NickelOut <= 1;
            when 25 =>
              DispenseJuice < = TRUE;
              DimeOut <= 2;
            when others => null;
          end case;
        when others => NextState <= IDLE;
      end case;
    end process;
  end DISPENSER;

  //Verilog:
  module MANGO_GUICE (CoinIn, Clock, Reset,
                      DispenseJuice, NickelOut, DimeOut);
    input [0:4] CoinIn;
    input Clock, Reset;
    output DispenseJuice;
    reg DispenseJuice;
    output [0:1] NickelOut, DimeOut;
    reg [0:1] NickelOut, DimeOut;
```

```
parameter IDLE = 0, FIVE = 1, TEN = 2, FIFTEEN = 3,
          TWENTY = 4;
parameter FALSE = 0, TRUE = 1;
reg [0:2] NextState, CurrentState;

always @(negedge Reset or negedge Clock)
  if (! Reset)
    CurrentState <= IDLE;
  else
    CurrentState <= NextState;

always @(CoinIn or CurrentState)
begin
  NextState = CurrentState;
  DispenseJuice = FALSE;
  NickelOut = 0;
  DimeOut = 0;

  case (CurrentState)
    IDLE:
      case (CoinIn)
        5: NextState = FIVE;
        10: NextState = TEN;
        25:
          begin
            NextState = IDLE;
            DispenseJuice = TRUE;
          end
        default: NextState = IDLE;
      endcase
    FIVE:
      case (CoinIn)
        5: NextState = TEN;
        10: NextState = FIFTEEN;
        25:
          begin
            NextState = IDLE;
            DispenseJuice = TRUE;
            NickelOut = 1;
          end
        default: NextState = IDLE;
      endcase
    TEN:
      case (CoinIn)
        5: NextState = FIFTEEN;
        10: NextState = TWENTY;
        25:
          begin
            NextState = IDLE;
            DispenseJuice = TRUE;
            DimeOut = 2;
          end
```

```
            default: NextState = IDLE;
          endcase
        FIFTEEN:
          case (CoinIn)
            5: NextState = TWENTY;
            10:
              begin
                NextState = IDLE;
                DispenseJuice = TRUE;
              end
            default: NextState = IDLE;
          endcase
        TWENTY:
          begin
            NextState = IDLE;

            case (CoinIn)
              5: DispenseJuice = TRUE;
              10:
                begin
                  DispenseJuice = TRUE;
                  NickelOut = 1;
                end
              25:
                begin
                  DispenseJuice = TRUE;
                  DimeOut = 2;
                end
              default:;
            endcase
          end
        default: NextState = IDLE;
      endcase
    end
  endmodule
```

The finite state machine is modeled using two processes (two always statements in Verilog). The first process is used to model the sequential logic, while the second process models the combinational logic. A case statement switches between the different states based on *CurrentState* and goes to the next state, depending on the value of the coin read in. Three asynchronous, clear flip-flops are synthesized from this model: these flip-flops hold the state of the mango juice dispenser.

For further reading on these topics of register transfer level synthesis, see (Bhasker, 1998). Since the modeling style enforces certain restrictions on what features of the language can be used, an IEEE standard for such a register transfer level synthesis subset is just evolving. The one for VHDL synthesis, IEEE PAR 1076.6, will be an IEEE standard by the end of this year, while the one for Verilog synthesis, IEEE PAR 1364.1, is expected to be an IEEE standard by the end of 1999. The VHDL synthesis subset standard is being developed by the VHDL synthesis interoperability working group, whereas the Verilog standard is being developed by the Verilog synthesis interoperability working group. The author is the chair for both working groups.

75.7 Behavioral Synthesis

As mentioned in an earlier section, behavioral synthesis differs from register transfer level synthesis due to the fact that clocking information may not exist in the model, that is, the behavior of the design at every clock edge is not explicitly described in the model. This is the task of the behavioral synthesis tool to determine.

Behavioral synthesis is often referred to as "high-level synthesis." In its basic form, an input to such a program is a sequential program, that is, a program that contains a sequence of statements that describes the behavior. This program is analyzed by a behavioral synthesis tool that then performs three major tasks:

- Scheduling — this is the task of determining the clock boundaries or assigning operations to clock cycles.
- ALU allocation — this is the task of determining which types of ALUs and how many ALUs are required.
- Register allocation — this task is to determine the required registers.

These tasks may be performed in any order by a behavioral synthesis tool. The fact is that these three problems need to be addressed. We shall explore each of these tasks in greater detail using the algorithm of a differential equation solver (see (Paulin, 1986)).

```
//Solve the differential equation:
//y'' + 5 x y' + 3y = 0

-- VHDL:
entity DIFF_EQ is
  port (A, DX: in INTEGER; X, U, Y: inout INTEGER);
end DIFF_EQ;

architecture HIGH_LEVEL of DIFF_EQ is
begin
  process
  begin
    while X < A loop
      X < = X + DX;
      U < = U – 5 * U * X * DX – 3 * Y * DX;
      Y < = Y + U * DX;
      wait for 0 ns;//Just so signal values settle.
    end loop;
  end process;
end HIGH_LEVEL;

//Verilog:
module DIFF_EQ (A, DX, X, U, Y);
  input [0:31] A, DX;
  output [0:31] X, U, Y;
  reg [0:31] X, U, Y;

  always @ (A or DX or X or U or )
    while (X < A)
    begin
      X < = X + DX;
      U < = U – 5 * U * X * DX – 3 * Y * DX;
      Y < = Y + U * DX;
```

```
        #0;
    end
endmodule
```

As you can see from the model, there is no mention of a clock, just the behavior of the differential equation solver is described. To ease the explanation of the following tasks that occur in a behavioral synthesis tool, we will use just one of the assignments (the second assignment) to explain the techniques and issues behind these tasks. The line is:

```
U <= U - 5 * U * X * DX - 3 * Y * DX;
```

Quite often, it is useful to construct a data flow graph (DeMicheli, 1994) of such a behavior. This is shown in Fig. 75.12. There are five multipliers and two subtractors. The arrows show the flow of data from one operation to another.

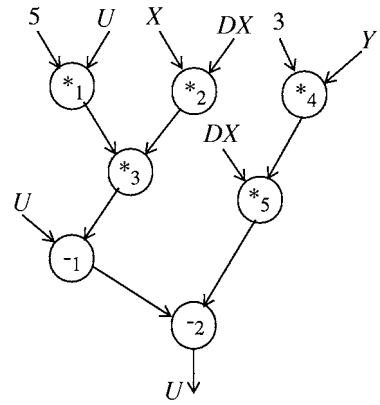


FIGURE 75.12 Data flow graph of assignment statement.

Scheduling

Given the previous model, we can see that there is no information about what operations are to be done in which clock cycle. The behavior would be performed in a number of clock cycles. Note that each clock cycle represents one state of the behavior. More or less, the behavior is broken up into a number of clock cycles based on the clock period. To determine the scheduling, additional data needs to be provided. In the previous case, the path delay of the subtractor and the multiplication operator needs to be known. This information can be derived from a technology library. In this instance, let us assume that a subtractor has a delay of 5 ns and a multiplier has a delay of 12 ns.

The number of operations that can be done in one control step is dependent on the clock period — this also needs to be known. Let us say, a clock period of 14 ns is specified. In such a case, Fig. 75.13 and Fig. 75.14 shows two possible ways of scheduling the operations. In the first schedule, operations $*_1$, $*_2$, $*_3$ are performed in the first clock cycle, operations $*_3$, $*_5$ are performed in the second clock cycle, operation $-_1$ is performed in the third clock cycle, and the operation $-_2$ is performed in the fourth clock cycle. With this schedule, the behavior takes four clock cycles to complete execution. In the second schedule, operators $*_1$ is performed in the first clock cycle, $*_2$ and $*_4$ are performed in the second clock cycle, $*_3$ is done in the third clock cycle, $-_1$ and $*_5$ in the fourth, and $-_2$ in the fifth. Which schedule is better depends on how the other tasks (ALU allocation and register allocation) perform in achieving the user constraints (typically area and speed).

If the clock period is changed to 25 ns, a different set of schedules can occur. One such example is shown in Fig. 75.15. Notice that the clock period is large enough to support two or more operations that are performed in sequence to be done in a single clock cycle. With this schedule, it takes only three clock cycles to complete execution.

Another constraint that could have been provided is the number of clock cycles.

Thus, by controlling the clock period and the number of clock cycles, different schedules (and different architectures) for the same behavior can be produced. This is called "exploration of architectural design space." The output of a scheduling task is typically an equivalent RTL model. An RTL model for the last schedule is shown next.

```
-- VHDL:
architecture RTL of EXAMPLE is
  type STATES is (CLOCK_CYCLE_1, CLOCK_CYCLE_2,
                  CLOCK_CYCLE_3);
```

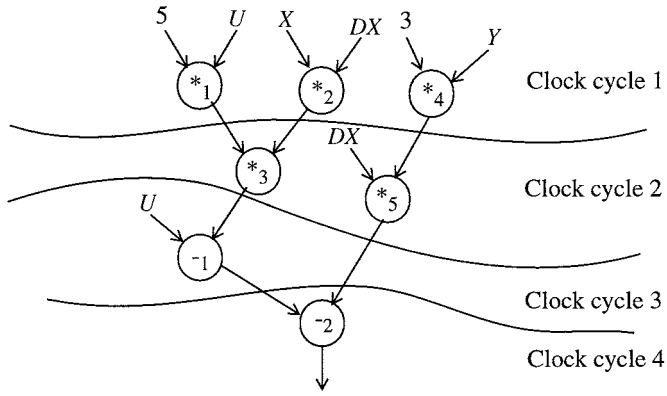


FIGURE 75.13 One possible schedule.

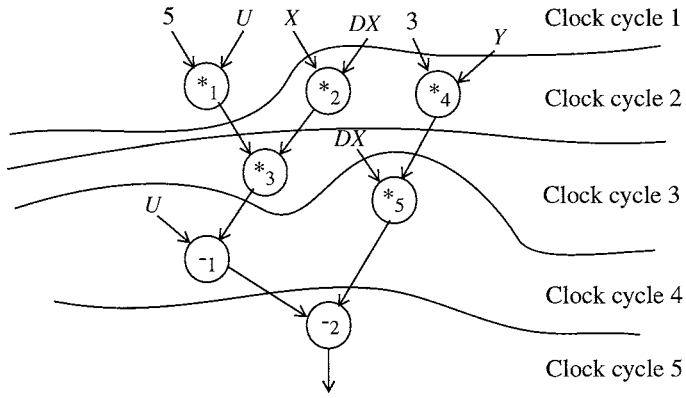


FIGURE 75.14 Another possible schedule.

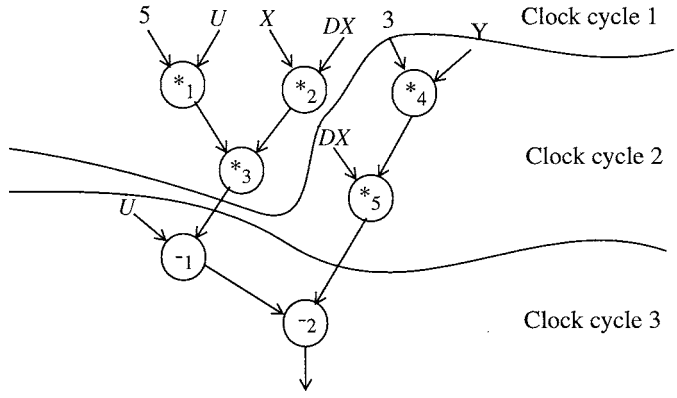


FIGURE 75.15 A schedule when the clock period is 25 ns.

```
  signal STATE: STATES;
  signal SAVE_1, SAVE_2: INTEGER;
begin
  process
    variable T1, T2: INTEGER;
  begin
    wait until CLOCK = '1';
    case STATE is
      when CLOCK_CYCLE_1 =>
        T1:= 5 * U;                          -- *1
        T2:= X * DX;                         -- *2
        SAVE_1 <= T1 * T2;                   -- *3
        STATE <= CLOCK_CYCLE_2;
      when CLOCK_CYCLE_2 =>
        T1:= 3 * Y;                          -- *4
        SAVE_2 <= T1 * DX;                   -- *5
        STATE <= CLOCK_CYCLE_3;
      when CLOCK_CYCLE_3 =>
        T1:= U - SAVE_1;                     -- -1
        U <= T1 - SAVE_2;                    -- -2
        STATE <= CLOCK_CYCLE_1; - Go back and repeat.
      when others:
        STATE <= CLOCK_CYCLE_1;
    end case;
  end process;
end RTL;

//Verilog:
parameter CLOCK_CYCLE_1 = 1, CLOCK_CYCLE_2 = 2,
          CLOCK_CYCLE_3 = 3;
reg [0:1] STATE;
integer T1, T2, SAVE_1, SAVE_2;

always @(posedge CLOCK)
  case (STATE)
    CLOCK_CYCLE_1:
      T1 = 5 * U;                        //*1
      T2 = X * DX;                       //*2
      SAVE_1 <= T1 * T2;                 //*3
      STATE <= CLOCK_CYCLE_2;
    CLOCK_CYCLE_2:
      T1 = 3 * Y;                        //*4
      SAVE_2 <= T1 * DX;                 //*5
      STATE <= CLOCK_CYCLE_3;
    CLOCK_CYCLE_3:
      T1 = U - SAVE_1;                   //-1
      U <= T1 - SAVE_2;                  //-2
      STATE <= CLOCK_CYCLE_1;
    default:
      STATE <= CLOCK_CYCLE_1;
  endcase
```

ALU Allocation

Once a schedule is known, the knowledge about which operations are mutually exclusive is also known. Two operations that are mutually exclusive, that is, are not used in the same clock cycle, may be shared using just one ALU. For example, in the schedule of Fig. 75.15, operations $*_2$ and $*_4$ are mutually exclusive; therefore, these two operations can be performed using a single multiplier (ALU that just does multiplication): in clock cycle 1, the multiplier performs the $*_2$ operation while in clock cycle 2, the multiplier performs the $*_4$ operation. However, sharing has a cost. Extra multiplexers may be needed at the input of the multiplier to multiplex the inputs in. This introduces extra delay and could be in the critical path of the circuit. So, a behavioral synthesis tool has to make a judicious choice when sharing.

Here are the various possibilities that can occur when sharing an ALU:

i Same operator, same operands: definitely must share. Example: $A + B$, $A + B$
ii Same operator, one different operand: tradeoff, since one multiplexer is introduced. Example: $A + B$, $A + C$
iii Same operator, different operands: tradeoff, since two multiplexers are introduced. Example: $A + B$, $C + D$
iv Different operators, same operands: useful to share. Example: $A + B$, $A - B$
v Different operators, one different operand: tradeoff, since one multiplexer is introduced. Example: $A + B$, $A - C$
vi Different operators, different operands: tradeoff, since two multiplexers are introduced. Example: $A + B$, $C - D$

Possibility i is the best case to share followed by iv, ii, v, iii, and vi.

Consider the schedule in Fig. 75.13. Sharing to its maximum, to reduce area, yields us just one subtractor and three multipliers. One possible way to share is:

- Operation $*_1$, $*_3$: ALU1 (multiplier)
- Operation $*_4$, $*_5$: ALU2 (multiplier)
- Operation $*_2$: ALU3 (multiplier)
- Operation $-_1$, $-_2$: ALU4 (subtractor)

The block diagram of the architecture is shown in Fig. 75.16. Notice that there are values that need to be saved in registers; these are shown as circles in the figure; more on this is in the next section.

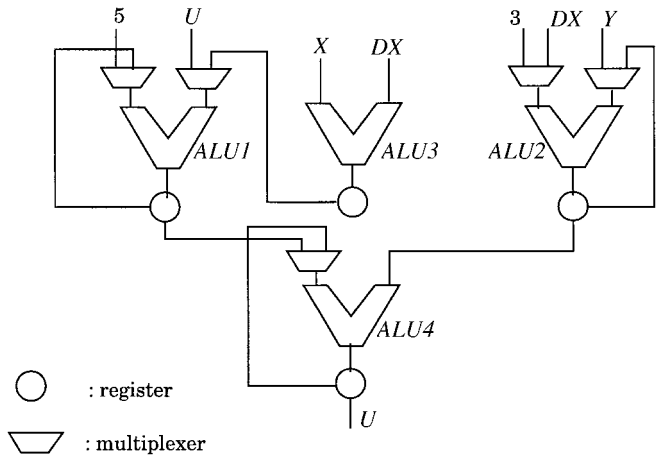


FIGURE 75.16 The ALU architecture of the first schedule.

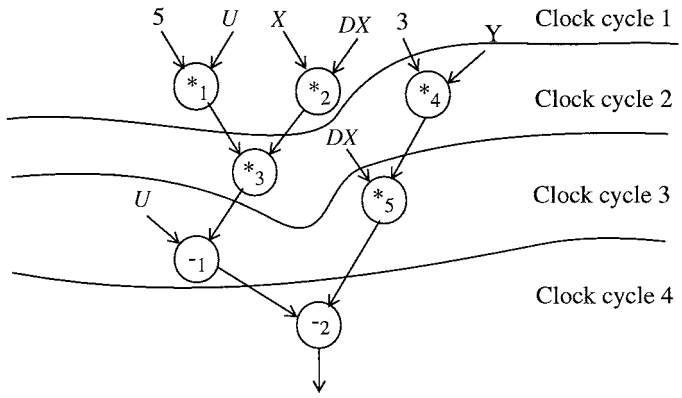


FIGURE 75.17 A schedule subject to resource constraints.

If a different kind of constraint, called resource constraints, is placed on the behavioral synthesis tool, for example, only two multipliers and one subtractor are available, then a different schedule and different sharing will occur. Figure 75.17 shows a possible schedule. Notice that in each clock cycle, two multiplications at most and one subtraction at most are performed. A possible sharing strategy is

- $*_1$, $*_3$: ALU1 (multiplier)
- $*_2$, $*_4$, $*_5$: ALU2 (multiplier)
- $-_1$, $-_2$: ALU3 (subtractor)

The architecture produced with this sharing is shown in Fig. 75.18. Another possible sharing strategy is

- $*_1$, $*_4$, $*_5$: ALU1 (multiplier)
- $*_2$, $*_3$: ALU2 (multiplier)
- $-_1$, $-_2$: ALU3 (subtractor)

Notice that even within the same schedule, different kinds of sharing can produce architectures that have different total area and speed.

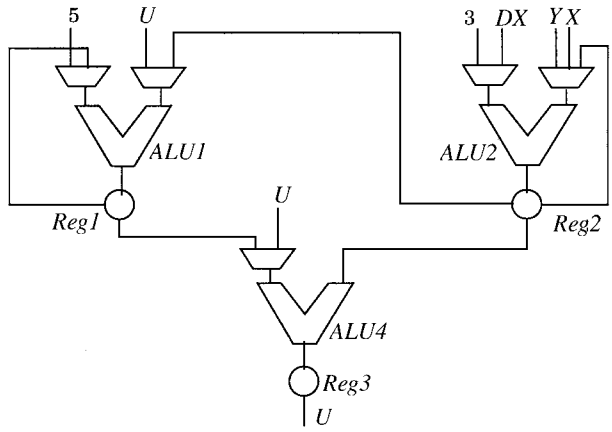


FIGURE 75.18 Architecture with resource constraints specified.

Register Allocation

The input to a behavioral synthesis tool is a behavioral model in which no registers are explicitly specified. But once a schedule has been determined, registers can be identified. A register is needed for any variable whose value has to be saved from one clock cycle to the next. For example, in Fig. 75.17, the output of $*_1$ has to be saved, since its value is defined in clock cycle 1 and used by $*_3$ in clock cycle 2. Similarly, the output of $*_4$ has to be saved as a register, since it is defined in clock cycle 2 and the value used in clock cycle 3.

The problem of register allocation is minimizing the number of such registers. If we do not apply any minimization, from the schedule shown in Fig. 75.17, we see that we will need six registers:

- R1: output of $*_1$
- R2: output of $*_2$
- R3: output of $*_3$
- R4: output of $*_4$
- R5: output of $*_5$
- R6: output of $-_1$

By optimizing the number of registers, this number can be reduced significantly. Such an analysis is performed by looking at the lifetimes of the potential registers. A lifetime is defined as the period from the clock cycle;it is defined to the clock cycle in which its value ceases to be not required. For example, here are the lifetimes of the above identified registers:

- R1: clock cycle 1 to 2
- R2: clock cycle 1 to 2
- R3: clock cycle 2 to 3
- R4: clock cycle 2 to 3
- R5: clock cycle 3 to 4
- R6: clock cycle 3 to 4

Two or more registers whose lifetimes are nonoverlapping can share the same register — basically once a register use is over, another register can reuse the same register for its lifetime, after which another register can use the register and so on. A possible register allocation may produce the following:

- Reg1: R1, R3
- Reg2: R2, R4, R5
- Reg3: R6

This allocation is shown in Fig. 75.18.

A different register allocation scheme may produce the following:

- Reg1: R1, R3, R6
- Reg2: R2, R4, R5

Notice that sharing of registers may also introduce multiplexers at the inputs of registers, as in the previous specified register allocation scheme. This is shown in Fig. 75.19. Selecting an appropriate register allocation is once again based on user-specified constraints such as area and speed.

There are other variations that are possible with the input HDL model:

i Zero scheduled
ii Partially scheduled
iii Fully scheduled

The example we have seen so far follows the “zero scheduled” model, since no explicit clock boundaries were defined. In a “fully scheduled” model, all the scheduling information is explicitly specified by the user. In a “partially scheduled” model, some points in the model are explicitly demarcated as clock

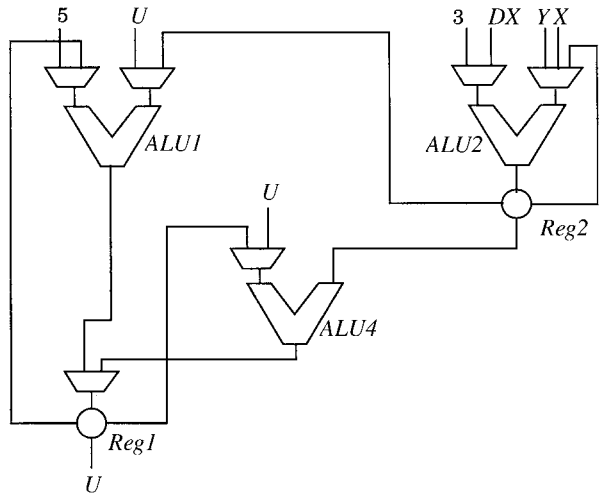


FIGURE 75.19 A register allocation may introduce extra multiplexers.

boundaries — for example, when a user knows exactly where a clock boundary must be but the behavioral synthesis tool is free to schedule the remaining parts of the behavior. We will break up our assignment statement that we have been using as an example to illustrate the various scheduling options.

```
-- Zero scheduled:
T1:= 5 * U;                -- *1
T2:= X̄ * DX;               -- *2
T3:= T1 * T2;              -- *3
T4:= 3 * Y;                -- *4
T5:= DX * T4;              -- *5
T6:= U – T3;               -- -1
U <= T6 – T5;              -- -2

-- Partially scheduled:
T1:= 5 * U;                -- *1
T2:= X * DX;               -- *2
T3 = T1 * T2;              -- *3
T4:= 3 * Y;                -- *4
wait until CLOCK = '1';
T5:= DX * T4;              -- *5
T6:= U – T3;               -- -1
U <= T6 – T5;              -- -2

-- Fully scheduled:
T1 = 5 * U;                -- *1
T2:= X * DX;               -- *2
wait until CLOCK = '1';
T3:= T1 * T2;              -- *3
T4:= 3 * Y;                -- *4
wait until CLOCK = '1';
T5:= DX * T4;              -- *5
T6:= U – T3;               -- -1
wait until CLOCK = '1';
U <= T6 – T5;              -- -2
```

Scheduling information is explicitly specified in the model by placing the clock edges — the demarcation points are "**wait until** *CLOCK* = '1';". In Verilog HDL, "@(**posedge** *CLOCK*);" can be used. In the partially scheduled and fully scheduled cases, architectural exploration is limited but still can be achieved by controlling the ALU sharing and register sharing.

An additional factor that may be considered in behavioral synthesis is scheduling with a clock period less than the maximum delay of an operator, for example, if a multiplier has a delay of 12 ns, then having a clock period of 5 ns.

Due to lack of space, we have skipped talking about controller synthesis which is an important aspect of behavioral synthesis. A good starting point on this topic is the reference (Nagle, 1982).

Verification of a behavioral synthesis model is a little more tricky than verification of an RTL model. This is due to lack of clocking information in the behavioral model. (See (Bhasker, 1991) for further details of an example of how verification can be performed.)

Two references for reading on the topic of logic synthesis are (Devadas, 1994) and (DeMicheli, 1994). Detailed algorithms are also described in these references. Further readings on behavioral synthesis can be found in (DeMicheli, 1994), (Camposano, 1991), and (Gajski, 1992).

For behavioral synthesis, it is usually not possible to support the entire set of language constructs supported by Verilog HDL and VHDL. However, at present, there is no publicly available standard subset.

75.8 Conclusion

The purpose of this chapter was to give an overview of synthesis in the register transfer level domain and the behavioral synthesis domain and to elaborate on the differences between the two. VHDL and Verilog HDL examples were used to illustrate these differences.

Behavioral synthesis is still a well-studied research topic. If you are interested in learning more about the latest state-of-the-art advances in this technology, a good starting point would be the Design Automation Conference proceedings, which usually contain a good one or two full sessions devoted to the topic of behavioral (high-level) synthesis.

References

Bhasker, J. and Ernst, R., SATYA: a simulation-based test system for high-level synthesis verification, *IEEE Design and Test,* 8(1), 14, 1991.

Bhasker, J., *A VHDL Primer: Revised Edition*, Prentice-Hall, Englewood Cliffs, NJ, 1995.

Bhasker, J., *A Verilog HDL Primer*, Star Galaxy Press, PA, 1997.

Bhasker, J., *A VHDL Synthesis Primer: Second Edition*, Star Galaxy Publishing, PA, 1998.

Camposano, R. and Wolf, W., *High-level VLSI Synthesis*, Kluwer Academic Publishers, MA, 1991.

DeMicheli, G., *Synthesis and Optimizations of Digital Circuits*, McGraw-Hill, New York, 1994.

Devadas S., Keutzer, K., and Ghosh, A., *Logic Synthesis*, McGraw-Hill, New York, 1994.

Gajski, D., *High-level Synthesis: Introduction to Chip and System Design*, Kluwer Academic Publishers, MA, 1992.

IEEE, ANSI/IEEE Std 1076-1993 *IEEE Standard VHDL Language Reference Manual,* 1993.

IEEE, IEEE Std 1364, *Hardware Description Language Based on the Verilog Hardware Description Language*, 1995.

Nagle, A., Cloutier, R., and Parker, A. C., Synthesis of hardware for the control of digital systems, *IEEE Trans. CAD IC Syst.,* vol CAD-1, no. 4, 201, 1982.

Paulin, P. et al., HAL: a multi-paradigm approach to automatic data path synthesis, *Design Automation Conf.,* 263, 1985.

Aylor, J.H., et al "Performance Modeling and Analysis in VHDL"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

76

Performance Modeling and Analysis in VHDL

James H. Aylor
University of Virginia

Robert H. Klenke
Virginia Commonwealth University

76.1 Introduction

It has been noted by the digital design community that the greatest potential for additional cost and iteration cycle time savings is through improvements in tools and techniques that support the early stages of the design process.[1] As shown in Fig. 76.1, decisions made during the initial phases of a product's development cycle determine up to 80% of its total cost. The result is that accurate, fast analysis tools must be available to the designer at the early stages of the design process to help make these decisions. Design alternatives must be effectively evaluated at this level with respect to multiple metrics, such as performance, dependability, and testability. This analysis capability will allow a larger portion of the design space to be explored yielding higher quality as well as lower cost designs.

There are a number of current tools and techniques that support analysis of these metrics at the system level to varying degrees. A major problem with these tools is that they are not integrated into the engineering design environment in which the system will ultimately be implemented. This problem leads to a major disconnection in the design process. Once the system-level model is developed and analyzed, the resulting high-level design is specified on paper and thrown "over the wall" for implementation by the engineering design team, as illustrated in Fig. 76.2. As a result, the engineering design team has to often interpret this specification in order to implement the system, which often leads to design errors. It also has to develop their own initial "high-level" model from which to begin the design process in a top-down manner. Additionally, there is no automated mechanism by which feedback on design assumptions and estimations can be provided to the system design team by the engineering design team.

For systems that contain significant portions of both application-specific hardware and software executing on embedded processors, design alternatives for competing system architectures and hardware/software (HW/SW) partitioning strategies must be effectively and efficiently evaluated using high-level performance models. Additionally, the selected hardware and software system architecture must be

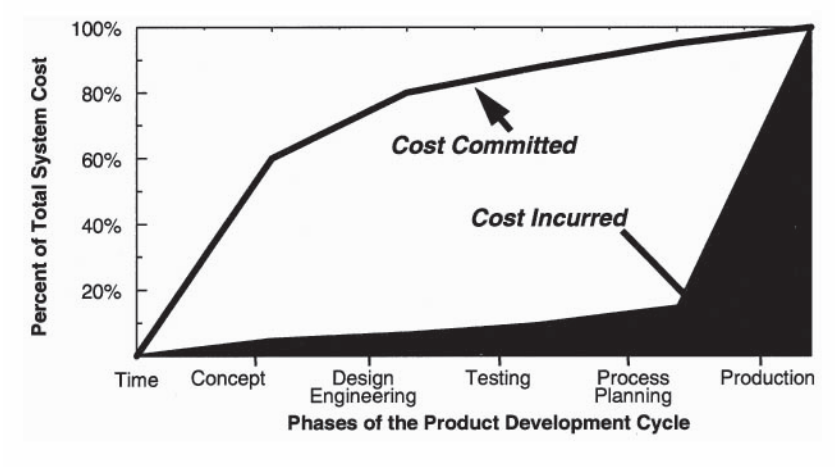


FIGURE 76.1 Product costs over the development cycle.

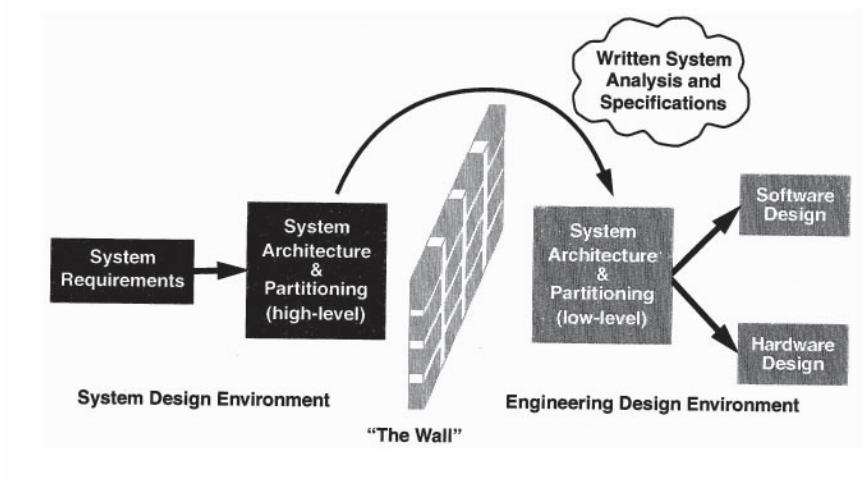


FIGURE 76.2 The disconnection between system-level design environments and engineering design environments.

refined in an integrated manner from the high-level models to an actual implementation in order to avoid implementation mistakes and the associated high redesign costs. Unfortunately, most existing design environments lack the ability to model and design a system's hardware and software in the same environment. A similar wall to that between the system design environment and the engineering design environment exists between the hardware design environment and the software design environment. This results in a design path as shown Fig. 76.3, where the hardware and software design process begins

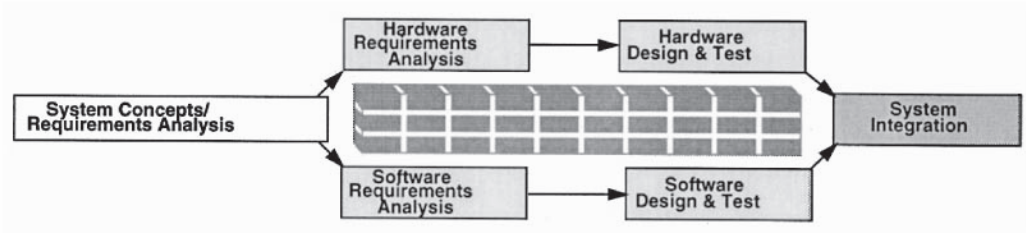


FIGURE 76.3 Current hardware/software system development methodology.

with a common system requirement and specification but proceeds through a separate and isolated design process until final system integration. At this point, assumptions on both sides may prove to be drastically wrong, resulting in incorrect system function and poor system performance.

A unified, cooperative approach in which the hardware and software options can be considered together is required to increase the quality and decrease the design time for complex hardware/software (HW/SW) systems. This approach is called hardware/software codesign, or simply codesign.[2-4] Codesign leads to more efficient implementations and improves overall system performance, reliability, and cost-effectiveness.[4] Also, because decisions regarding the implementation of functionality in software can impact hardware design (and vice versa), problems can be detected and changes made earlier in the development process.[5]

Codesign can especially benefit the design of embedded systems,[6] systems which contain hardware and software tailored for a particular application. As the complexity of these systems increases, the issue of providing design approaches that scale up to more complicated systems becomes of greater concern. A detailed description of a system can approach the complexity of the system itself,[7] and the amount of detail present can make analysis intractable. Therefore, decomposition techniques and abstractions are necessary to manage this complexity.

What is needed is a design environment in which the capability for performance modeling of HW/SW systems at a high-level of abstraction is fully integrated into the engineering design environment. In order to completely eliminate the "over the wall" problem and the resulting model discontinuity, this environment must support the incremental refinement of the abstract system level performance model into an implementation level model. Using this environment, a design methodology based on incremental refinement can be developed.

The design methodology illustrated in Fig. 76.4 was proposed by Lockheed Martin Advanced Technology Laboratory as a new way to design systems.[8] This methodology uses the level of the risk of not meeting the design specifications as the metric for driving the design process. In this spiral-based design methodology, there are two iteration cycles. The major cycles (or spirals), denoted as CYCLE 1, CYCLE 2, …, CYCLE N, on the figure correspond to the design iterations where major architectural changes are made in response to some specification metric(s) and/or the system as a whole is refined and more design detail is added to the model. Consistent with the new paradigm of system design, these iterations will actually produce virtual or simulation-based prototypes. A virtual prototype is simply a simulatable model of the system with stimuli described at a given level of design detail or design abstraction that describes the system's operation. Novel to this approach are the mini spirals. The mini spiral cycles, denoted by the levels on the figure labled SYSTEMS, ARCHITECTURE, and DETAILED DESIGN, correspond to the refinement of only those portion(s) of the design that are deemed to be "high risk." High risk is obviously defined by the designer but is most often the situation where if one or more of

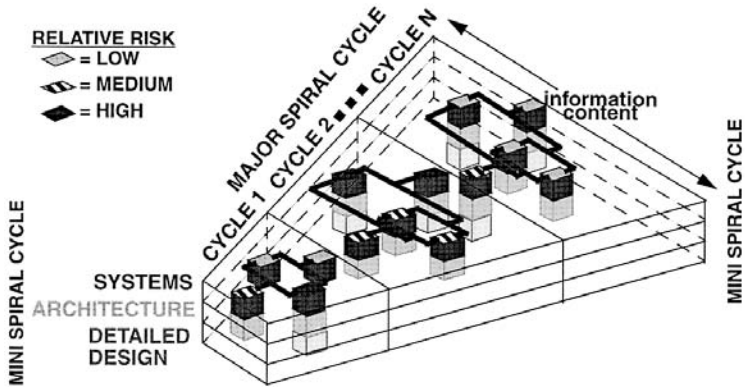


FIGURE 76.4 Risk-driven expanding information model (RDEIM).

these components fail to meet their individual specifications, the system will fail to meet its specifications. The way to minimize the risk is to refine these components to possibly an implementation so that the actual performance is known. Unlike the major cycles where the entire design is refined, the key to the mini spirals is the fact that only the critical portions of design are refined. For obvious reasons, the resulting models have been denoted as risk-driven expanding information models (RDEIMs).

The key to being able to implement this design approach is to be able to evaluate the overall design with portions of the system having been refined to a detailed level while the rest of the system model remains at the abstract level. For example, in the first major cycle of Fig. 76.4, the element with the highest relative risk is fully implemented (detailed design level) while the other elements are described at more abstract levels (system level or architectural level). If the simulation of the model shown in the first major cycle detects that the overall system will not meet its performance requirements, then the "high risk" processing element could be replaced by two similar elements operating in parallel. This result is shown in the second major cycle, and at this point another element of the system may become the new "bottleneck", i.e., the highest relative risk, and it will be refined in a similar manner.

Implied in the RDEIM approach is a solution to the "over the wall" problem, including HW/SW codesign. The proposed solution is to fully integrate performance modeling into the design process.

Obviously, one of the major capabilities necessary to implement a top-down design methodology like the RDEIM is the ability to co-simulate system models which contain some components that are modeled at an abstract performance level (uninterpreted models) and some that are modeled at a detailed behavioral level (interpreted models). This ability to model and co-simulate uninterpreted models and interpreted models is called mixed-level modeling (sometimes referred to as hybrid modeling). Mixed-level modeling requires the development of interfaces that can resolve the differences between uninterpreted models that, by design, do not contain a representation of all of the data or timing information of the final implementation, and interpreted models which require most, or possibly all, data values and timing relationships to be specified. Techniques for systematic development of these mixed-level modeling interfaces and resolution of these differences in abstraction is the focus of the latest work in mixed-level modeling.

A unified end-to-end design environment has been developed at the University of Virginia that attempts to achieve this goal. This environment supports the development of system-level models of digital systems that can be analyzed for multiple metrics, like performance and dependability, and can then be used as a starting point for the actual implementation. A tool called ADEPT (Advanced Design Environment Prototype Tool) has been developed to implement this environment. ADEPT actually supports both system-level performance and dependability analysis in a common design environment using a collection of predefined library elements. ADEPT also includes the ability to simulate both system level and implementation-level (behavioral) models in a common simulation environment. This capability allows the stepwise refinement of system-level models into implementation-level models.

Multi-Level Modeling

The need for multi-level modeling was recognized almost three decades ago. Multi-level modeling implies that representations at different levels of detail coexist within a model.[7,9,10] Until the early 1990s, the term multi-level modeling was used for integrating behavioral or functional models with lower level models. The objective was to provide a continuous design path from functional models down to implementation. This objective was achieved, and the VLSI industry utilizes it today. An example is the tool called Droid, developed by Texas Instruments.[11]

The mixed-level modeling approach, as described in this chapter, is a specific type of multi-level modeling which integrates performance and behavioral models. Thus, only related research on multi-level modeling systems that spans both the performance and the functional/behavioral domains will be described.

Although behavioral or functional modeling is typically well-understood by the design community, performance modeling is a foreign topic to most designers. Performance modeling, also called uninterpreted modeling, is utilized in the very early stages of the design process in evaluating such metrics as

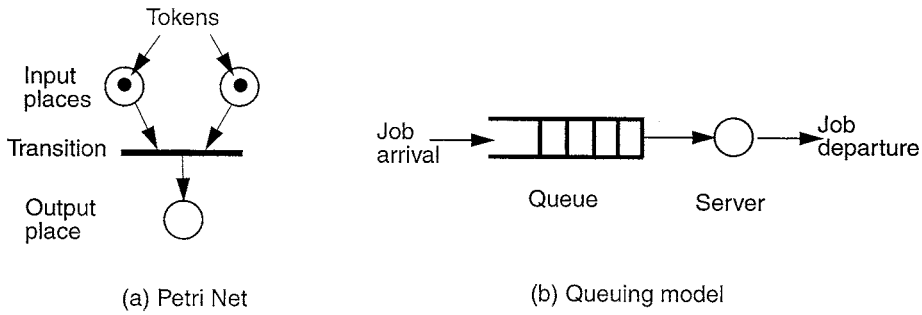


FIGURE 76.5 Petri Net and queuing model.

throughput and utilization. Performance models are also used to identify bottlenecks within a system and are often associated with the job of a system engineer. The term "uninterpreted modeling" reflects the view that performance models lack value-oriented data and functional (input/output) transformations. However, in some instances, this information is necessary to allow adequate analysis to be performed.

A variety of techniques have been employed for performance modeling. The most common techniques are Petri Nets[12-14] and queuing models.[15,16] A combination of these techniques, such as a mixture of Petri Nets and queuing models,[17] has been utilized to provide more powerful modeling capabilities. All of these models have mathematical foundations. However, models of complex systems constructed using these approaches can quickly become unwieldy and difficult to analyze.

Examples of a Petri Net and a queuing model are shown in Fig. 76.5. A queuing model consists of queues and servers. Jobs (or customers) arrive at a specific arrival rate and are placed in a queue for service. These jobs are removed from the queue to be processed by a server at a particular service rate. Typically, the arrival and service rates are expressed using probability distributions. There is a queuing discipline, such as first-come-first-serve, which determines the order in which jobs are to be serviced. Once they are serviced, the jobs depart and arrive at another queue or simply leave the system. The number of jobs in the queues represents the model's state. Queueing models have been used successfully for modeling many complex systems. However, one of the major disadvantages of queuing models is their inability to model synchronization between processes.

As a system modeling paradigm, Petri Nets overcome this disadvantage of queuing models. Petri Nets consist of places, transitions, arcs, and a marking. The places are equivalent to conditions and hold tokens which represent information. Thus, the presence of a token in the place of a Petri Net corresponds to a particular condition being true. Transitions are associated with events, and the "firing" of a transition indicates that some event has occurred. A marking consists of a particular placement of tokens within the places of a Petri Net and represents the state of the net. When a transition fires, tokens are removed from the input places and are added to the output places, changing the marking (the state) of the net and allowing the dynamic behavior of a Petri Net to be modeled.

Petri Nets can be used for performance analysis by associating a time with the transitions. Timed and stochastic Petri Nets contain deterministic and probabilistic delays, respectively. Normally, these Petri Nets are uninterpreted, since no interpretation (values or value transformations) are associated with the tokens or transitions. However, values or value tranformations can be associated with the various elements of Petri Net models as described below.

Petri Nets that have values associated with tokens are known as colored Petri Nets (CPNs). In the colored Petri Nets, each token has an attached color, indicating the identity of the token. The net is similar to the basic definition of the Petri Net except that a functional dependency is specified between the color of the token and the transition firing action. In addition, the color of the token produced by a transition may be different from the color of the token(s) on the input place(s). CPNs have an increased

ability to efficiently model real systems with small nets which are equivalent to much larger plain Petri Nets due to their increased descriptive powers.

Numerous multi-level modeling systems exist based on these two performance modeling techniques. ADAS (Architecture Design and Assessment System) is a set of tools specifically targeted for high-level design.[18] ADAS models both hardware and software using directed graphs based on timed Petri Nets. The flow of information is quantified by identifying discrete units of information called tokens. ADAS supports two levels of modeling. The more abstract level is a dataflow description and is used for performance estimation. The less abstract level is defined as a behavioral level but still uses tokens which carry data structures with them. The functionality is embedded into the models using C or Ada programs. The capability of generating high-level VHDL models from the C or Ada models is provided. These high-level VHDL models can be further refined and developed in a VHDL environment, but the refined models cannot be integrated back into the ADAS performance model. The flow of information in these high-level VHDL models is still represented by tokens. Therefore, implementation-level components cannot be integrated into an ADAS performance model. Another limitation is that all input values to the behavioral node must be contained within the token data structure.

SES/Workbench (Scientific and Engineering Software) is a design specification, modeling and simulation tool.[19] It is used to construct and evaluate proposed system designs and to analyze their performance. A graphical interface is used to create a structural model which is then converted into a specific simulatable description (SES/sim). SES/Workbench enables the transition across domains of interpretation by using a user node, in which C-language and SES/sim statements can be executed. Therefore, SES/Workbench has similar limitations to ADAS; the inability to simulate a multi-level model when input values of behavioral nodes are not fully specified and the inadequacy of simulating components described as implementation-level HDLs (the capability of integrating VHDL models will be introduced later in the next paragraph). In addition, multiple simulation languages (both SES/sim and C) are required for multi-level models.

The Reveal Interactor is a tool developed by Redwood Design Automation.[20] A model constructed in Reveal is aimed at the functional verification of RTL-level VHDL and Verilog descriptions and, therefore, does not include a separate transaction-based performance modeling capability. However, Reveal can work in conjunction with SES/Workbench. By mixing models created in Reveal and SES/Workbench, a multi-level modeling capability exists. Again, these multi-level models are very limited due to the fact that all the required information at the lower level part of the model must be available within the higher level model.

IDAS (Integrated Design Automation System) is a multi-level design environment which allows for rapid prototyping of systems.[21] Although the behavioral specifications need to be expressed as Ada, C, or Fortran programs, IDAS provides the capability of automatically translating VHDL description to Ada. However, the user cannot create abstract models in which certain behavior is unspecified. Also, it does not support classical performance models (such as queuing models and Petri Nets) and forces the user to specify a behavioral description very early in the design process.

Transcend claims to integrate multi-level descriptions into a single environment.[22,23] In the more abstract level, T-flow models are used, in which tokens are used to represent flow of data. The capability of integrating VHDL sub-models into a T-flow model is provided. However, interfacing between the two models requires a "C++ like" language, which maps variables to/from VHDL signals, resulting in a heterogeneous simulation environment. Although its approach is geared toward the same objective as mixed-level modeling, the T-flow model must also include all the necessary data to activate the VHDL sub-models. Therefore, the upper level model cannot be "too abstract" and must include lower level details.

MIDAS (Methodology for Integrated Design and Simulation) supports the design of distributed systems via iterative refinement of PIPS (Partially Implemented Performance Specification) models.[24] A PIPS model is a partially implemented design where some components exist as simulation models and others as operational subsystems (i.e., implemented components). Although the term "hybrid model" is used in this context, it refers to a different type of modeling. MIDAS is an "integrated approach to

software design."[24] It supports the performance evaluation of software being executed on a given machine. It does not allow the integration of components expressed in an HDL into the model.

The Ptolemy project is an academic research effort being conducted at the University of California at Berkeley.[25,26] Ptolemy, a comprehensive system prototyping tool, is actually constructed of multiple domains. Most domains are geared toward functional verification and have no notion of time. Each domain is used for modeling a different type of system. The domains also vary in the modeling level (level of abstraction). Ptolemy provides limited capability of mixing domains within one design. The execution of a transition across domains is accomplished with a "wormhole." A wormhole is the mechanism for supporting the simulation of heterogeneous models. Thus, a multi-level modeling and analysis capability is provided. There are two major limitations to this approach compared with the mixed-level modeling approach being described. The first one is the heterogeneity — several description languages. Therefore, translation between simulators is required. The second limitation is that the interface between domains only translates data. Therefore, all the information required by the receiving domain must be generated by the transmitting domain.

Honeywell Technology Center (HTC) currently has a research effort that is specifically addressing the mixed-level modeling problem.[27] This research has its basis in the UVa mixed-level modeling effort. The investigators at HTC have developed a Performance Modeling Library (PML)[28,29] and are currently working on adding a mixed-level modeling capability to this environment. The PML is used for performance models at a relatively low level of abstraction. Therefore, it assumes that all the information required by the interpreted element is provided by the performance model. In addition, their interface between uninterpreted and interpreted domains allows for bidirectional data flow.

To summarize, numerous multi-level modeling efforts exist. However, they are being developed not addressing the issue of lack of information at the transition between levels of abstraction. A solution to this problem is essential for true stepwise refinement of the performance models to behavioral models. In addition, integration of performance modeling level and behavioral modeling level was mostly performed by mixing different simulation environments, which results in heterogeneous modeling approach.

76.2 The ADEPT Design Environment

Two approaches to creating a unified design environment are possible. An evolutionary solution is to provide an environment that translates data from different models at various points in the design process and creates interfaces for the noncommunicating software tools used to develop these models. With this approach, users must be familiar with several modeling languages and tools. Also, analysis of design alternatives is difficult and is likely to be limited by design time constraints.

A revolutionary approach, the one being developed in ADEPT, is to use a single modeling language and mathematical foundation. This approach uses a common modeling language and simulation environment, which decreases the need for translators and multiple models, reducing inconsistencies and the probability of errors in translation. Finally, the existence of a mathematical foundation provides an environment for complex system analysis using analytical approaches.

Simulators for hardware description languages accurately and conveniently represent the physical implementation of digital systems at the circuit, logic, register-transfer, and algorithmic levels. By adding a system-level modeling capability based on extended Petri Nets and queuing models to the hardware description language, a single design environment can be used from concept to implementation. The environment would also allow for the mixed simulation of both uninterpreted (performance) models and interpreted (behavioral) models due to the use of a common modeling language. Although it would be possible to develop the high-level performance model and the detailed behavioral model in two different modeling languages and then develop some sort of translator or foreign language interface to hook them together, a better approach is to use a single modeling language for both models. A single modeling language that spans numerous design phases is much easier to use, encouraging more design analysis and consequently better designs.

ADEPT implements an end-to-end unified design environment based on the use of the VHSIC Hardware Description Language (VHDL).[30] VHDL is a natural choice for this single modeling language in that it has high-level language constructs, but unlike other programming languages, it has a built-in timing and concurrency model. VHDL does have some disadvantages in terms of simulation execution time, but techniques have been developed to help address this problem.[31]

ADEPT supports the integrated performance and dependability analysis of system-level models and includes the ability to simulate both uninterpreted and interpreted models through mixed-level modeling. ADEPT also has a mathematical basis in Petri Nets, thus providing the capability for analysis through simulation or analytical approaches.[32]

In the ADEPT environment, a system model is constructed by interconnecting a collection of predefined elements called ADEPT modules. The modules model the information flow, both data and control, through a system. Each ADEPT module has a VHDL behavioral description and a corresponding mathematical description in the form of a CPN based on Jensen's CPN model.[33] The modules communicate by exchanging tokens, which represent the presence of information, using a fully interlocked, four-state handshaking protocol.[34] The basic ADEPT modules are intended to be building blocks from which useful modeling functionality can be constructed. In addition, custom modules can be developed by the user if required and incorporated into a system model as long as the handshaking protocol is adhered to. Finally, some libraries of application-specific, high-level modeling modules such a Multiprocessor Communications Network Modeling Library[35] have been developed and included in ADEPT.

The following sections discuss the VHDL implementation of the token data type and transfer mechanism used in ADEPT and the modules provided in the standard ADEPT modeling library.

Token Implementation

The modules defined in this chapter have been implemented in VHDL and use a modified version of the token passing mechanism defined by Hady.[34] Signals used to transport tokens must be of the type token, which has been defined as a record with two fields. The first field, labeled STATUS, is used to implement the token passing mechanism. The second field, labeled COLOR, is an array or integers that is used to hold user-defined color information in the model. The ADEPT tools allow the user to select from a predefined number of color field options. The tools then automatically link in the proper VHDL design library so that the VHDL descriptions of the primitive modules operate on the defined color field. The default structure of the data-type token used in the examples discussed in this document is shown in Fig. 76.6.

There are two types of outputs and two types of inputs in the basic ADEPT modules. Independent outputs are connected to control inputs, and the resulting connection is referred to as a control-type signal. Dependent outputs are connected to data inputs, and the resulting connection is referred to as a data-type signal. To make the descriptions more intuitive, outputs are often referred to as the "source side" of a signal, and inputs are referred to as the "sink side" of a signal.

Tokens on independent outputs may be written over by the next token, so the "writing" process is independent of the previous value on the signal. On the other hand, new tokens may not be placed on dependent outputs until the previous token has been removed, so the "writing" process is dependent on the previous value on the signal. Data-type signals use the four-step handshaking process to ensure that tokens do not get overwritten, but no handshaking occurs on control-type signals.

The STATUS field of a token signal can take on four values. Signals connecting independent outputs to control inputs (control-type signals) make use of only two of the four values. Independent outputs place a value PRESENT on the status field to indicate that a token is present. Since control inputs only copy the token but do not remove it, the independent output only needs to change the value of the status field to RELEASED to indicate that the token is no longer present on the signal, and the control input can no longer consider the signal to contain valid information. The signals connecting dependent outputs to data inputs (data-type signals) need all four values (representing a fully interlocked handshaking scheme) to ensure that a dependent output does not overwrite a token before the data input to which it

```
type handshake is (removed, acked, released, present);
type token_fields is (status,
                      tag1, tag2, tag3, tag4, tag5,
                      tag6, tag7, tag8, tag9, tag10,
                      tag11, tag12, tag13, tag14, tag15,
                      boole1, boole2, boole3,
                      fault, module_info,
                      index,act_time,color);

type color_type is array (token_fields range tag1 to act_time) of integer;

type token is
      record
         status    : handshake;
         color     : color_type;
     end record;

type op_fields is (add, sub, mul, div);
type cmp_fields is (lt, le, gt, ge, eq, ne);
type tkf_array is array (1 to 25) of token_fields;-- used in file_read


type token_vector is array (integer range <>) of token;

--  resolution function for token
function protocol (input : token_vector) return token;
subtype token_res is protocol token;
```

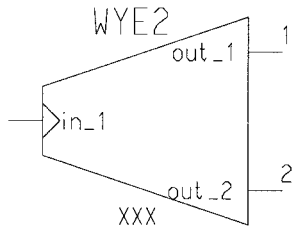
FIGURE 76.6 Token type definition.

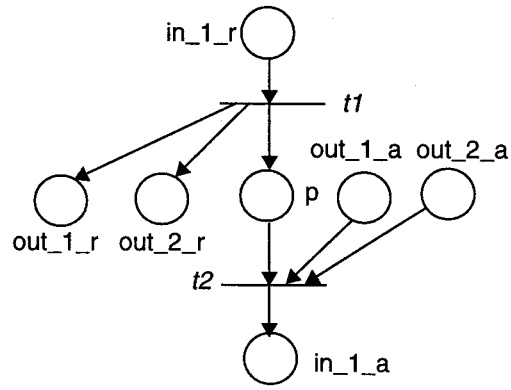
is connected has read and removed it. This distinction is important, since a token on control-type signals represents the presence or absence of a condition in the network while tokens on data-type signals, on the other hand, represent information or data that cannot be lost or overwritten. In addition, fan-out is not permitted on dependent outputs and is permitted on independent outputs.

The signals used in the implementation are of type token. The token passing mechanism for data-type signals is implemented by defining a VHDL bus resolution function. This function is called each time a signal associated with a dependent output and a data input changes value. The function essentially looks at the value of the STATUS field at the ports associated with the signal and decides the final value of the signal. At the beginning of simulation, the STATUS field is initialized to REMOVED. This value on a signal corresponds to an idle link in the nets defined by Dennis.[36] This state of a signal indicates that a signal is free and a request token may be placed on it. Upon seeing the REMOVED value, the requesting module may set the value of the signal to PRESENT. This operation corresponds to the ready signal in Dennis' definition. The requested module acknowledges a ready signal (PRESENT) by setting the signal value to ACKED, after the requested operation has been completed. Upon sensing the value of ACKED on the signal, the requesting module sets the value of the signal to RELEASED, which in turn causes the requested module to place the value of REMOVED on the signal. This action concludes one request/acknowledge cycle, and the next request/acknowledge cycle may begin.

In terms of Petri Nets, the models described here can be described as one-safe Petri Nets, since at any given time, no place in the net can contain more than one token. The correspondence between the signal values and transitions in a Petri Net may be defined in general in the following manner:

1. PRESENT: A token arrives at the input place of a transition.
2. ACKED: The output place of a transition is empty.
3. RELEASED: The transition has fired.
4. REMOVED: The token has been transferred from the input place to the output place.






```
library uvalib;
   use uvalib.uva.all;
   use uvalib.rng.all;

entity wye2 is
   port (in_1 : inout token;
         out_1: inout token;
         out_2: inout token);
end wye2;

architecture ar_wye2 of wye2 is
begin
   pr_wye2 : process (in_1, out_1, out_2)
   begin
      if token_present (in_1)
         and token_removed (out_1)
         and token_removed (out_2) then
         out_1 <= in_1;
         out_2 <= in_1;
      end if;

      if token_acked (out_1) and token_acked (out_2) then
         ack_token (in_1);
         release_token (out_1);
         release_token (out_2);
      end if;

      if token_released (in_1) then
         remove_token (in_1);
      end if;
   end process pr_wye2;
end ar_wye2;
```

FIGURE 76.7 Wye module ADEPT symbol, its behavioral VHDL description, and its CPN representation.

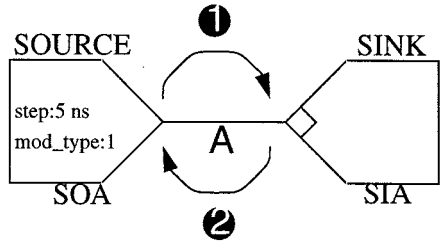
The modules to be described in this chapter may be defined in terms of Petri Nets. As an example, a Petri Net description of the Wye module is shown in Fig. 76.7.

The function of the Wye module is to copy the input token to both outputs. The input token is not acknowledged until both output tokens have been acknowledged. In the Petri Net of Fig. 76.7, the "R" and "A" labels correspond to "ready" and "acknowledge", respectively. When a token arrives at the place labeled "0r", the top transition is enabled and a token is placed in the "1r", "2r", and center places. The first two places correspond to a token being placed on the module outputs. Once the output tokens are acknowledged (corresponding to tokens arriving at the "1a" and "2a" places, the lower transition is enabled and a token is placed in "0a", corresponding to the input token being acknowledged. The module is then ready for the next input token. Note that since this module does not manipulate the color fields, no color notation appears on the net.

The specific CPN description used is based on the work of Jensen.[33] The complete CPN descriptions of each of the ADEPT building blocks can be found in the work of Swaminathan et al.[37]

ADEPT Handshaking and Token Passing Mechanism

Recall that the ADEPT standard token has a status field which can take on four values: PRESENT, ACKED, RELEASED, and REMOVED. These values reflect which stage of the token passing protocol is currently in progress. Several examples will now be presented to show how the handshaking and token passing



Time between new tokens on A = step + path delay = 5 ns

Table A: Simplified Event Sequence

Event	Time	Description
1	0 ns	Source module places token on signal A.
2	0 ns	Sink module acknowledges token on A.
1	5 ns	Source module places next token on signal A.

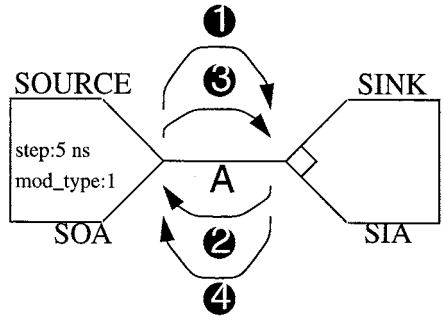


Table B: Detailed Event Sequence

Event	Time	Delta	Description	Resolved Signal A
1	0 ns	1	Source module places token on A	present
2	0 ns	2	Sink module acknowledges token on A	acked
3	0 ns	3	Source module releases token on A	released
4	0 ns	4	Sink module removes token on A	removed
1	5 ns	1	Source module places token on A	present

FIGURE 76.8 Two-module handshaking example.

occurs between ADEPT modules. Figure 76.8 illustrates the handshaking process between a source module connected to a Sink module.

Figure 76.8A describes the simplified event sequence. Here, we can think of the four-step handshaking as consisting of simply a forward propagation of the token and a backward propagation of the token acknowledgment. Thinking of the handshaking in this simplified manner, Table A in Fig. 76.8 shows the simplified event sequence and times. At time 0 ns, the source module places a token on A, corresponding to Event 1. The token is immediately acknowledged by the sink module, corresponding to Event 2. Since the source module has a step generic of 5 ns and there is no other delay along the path to the sink, the next token will be output by the source module at time 5 ns.

Figure 76.8B details the entire handshaking process. Table B in Fig. 76.8 lists the detailed event sequence for this example. Since handshaking occurs in zero simulation time, the events are listed by delta cycles within each simulation time. The function of delta cycles in VHDL is to provide a means for ordering and synchronizing events (such as handshaking) which occur in zero time. The actual event sequence for this example consists of four steps, as shown in Fig. 76.8B. Event 1 is repeated at time 5 ns, which starts the handshaking process over again.

A Three-Module Example

To illustrate how tokens are passed through intermediate modules, a three-module example will now be examined. Consider the case where a fixed delay module is placed between a source and sink. This situation is illustrated in Fig. 76.9.

Figure 76.9A shows the simplified event sequence, where we can think of the four-step handshaking as consisting of simply a forward propagation of the token and a backward propagation of the token acknowledgment. Table A in Fig. 76.9A lists this simplified event sequence. Notice that since now there is a path delay from the source to the sink, the time between new tokens from the source is *step* + *path_delay* = 5 ns + 5 ns = 10 ns. At time 0 ns, the source module places the first token on Signal A (Event 1). This token is read by the delay module and placed on signal B after a delay of 5 ns (Event 2). The sink module then immediately acknowledges the token on signal B (Event 3). The delay module then passes this acknowledgment back through to signal A (Event 4). At time 10 ns, the source module places the next token on Signal A, starting the process over again.

Figure 76.9B shows the detailed event sequence for this example. Since there are two signals, there are a total of eight detailed events. Table B in Fig. 76.9 lists the detailed event sequence for this example. Again, since the handshaking occurs in zero simulation time, the events are listed by delta cycles within each simulation time. Table B lists the resolved values on both signal A and signal B, where a value of "----" indicates no change in value. Notice that the sequence of events on each signal proceeds in this order: place the token (present) by the source side, acknowledge the token (acked) by the sink side, release the token (released) by the source side, and finally remove the token (removed) by the sink side. The important concept to note in this example is the ordering of events. Notice that the delay module releases the token on its output (Event 4) before it acknowledges the token on its input (Event 5), even though the two events occur in the same simulation time. These actions trigger the sink module to then remove the token on signal B (Event 6) and the source module to release the token on signal A (Event 7), both in the next delta cycle. Thus signal B is ready for another token (removed) a full delta cycle before signal A is ready.

Token Passing Example

To illustrate how tokens propagate in a larger system, consider the model shown in Fig. 76.10. This figure shows the simplified event sequence, where again we can think of the four-step handshaking as consisting of simply a forward propagation of the token and a backward propagation of the token acknowledgment. In this system, the source module provides tokens to the sequence A (SA) module (Event 1), who first passes them to the sequence B (SB) module (Event 2). The SB module first passes the token to the fixed delay A (FDA) module (Event 3), who passes it to the sink A (SIA) module after a delay of 5 ns (Event 4). The SIA module then immediately acknowledges the token (Event 5), which causes the FDA module to pass the acknowledgment back to the SB module (Event 6). At this point, the SB module places a copy of the token on its second output (Event 7), where it is passed through the Union A and onto its output (Event 8). The read color A module then places the token on its out_1 output (Event 9) and simultaneously places the token on its independent output. The sink B module then acknowledges the token on signal H (Event 10). Upon seeing this acknowledgment, the read color A module releases the token on its independent output, since the read color module has no "memory" (the *release?* generic equals true). The acknowledgment is then propagated back to the SB module (Events 11 and 12). At this point, the SB module acknowledges its input token (Event 13), freeing the SA module to place a token on its second output (Event 14). This token is then passed through the union A, read color A, and sink module (Events

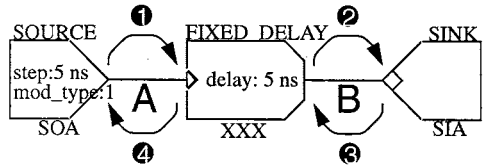


Table A: Simplified Event Sequence

Event	Time	Description
1	0 ns	Source module places token on signal A.
2	5 ns	Delay module places token on signal B.
3	5 ns	Sink module acknowledges token on signal B.
4	5 ns	Delay module acknowledges token on signal A.
1	10 ns	Source module places next token on signal A.

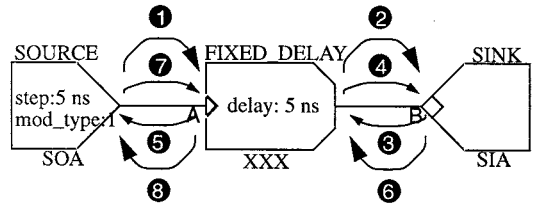


Table B: Detailed Event Sequence

Event	Time	Delta	Description	Resolved Signal A	Resolved Signal B
1	0 ns	1	Source module places token on A	present	removed
2	5 ns	1	Delay module places token on B	--	present
3	5 ns	2	Sink module acknowledges token on B	--	acked
4	5 ns	3	Delay module releases token on B	--	released
5			Delay module acknowledges token on A	acked	--
6	5 ns	4	Sink module removes token on B	--	removed
7			Source module releases token on A	released	--
8	5 ns	5	Delay module removes token on A	removed	--
1	10 ns	1	Source module places token on A	present	--

FIGURE 76.9 Three module handshaking example.

15, 16) and the acknowledgment is returned (Events 17, 18, and 19). The SA module can then acknowledge the token on its input (Event 20), allowing the source module to generate the next token 5 ns (step) later.

If we examine the activity on Signal I (the independent output of the read color A module), we see that it goes present after Event 8, released after Event 10, present after Event 15, and released again after Event 17. Consider the operation of the terminator module attached to this signal. Since the Terminator module halts simulation after the number of active events specified by the *stop_after* generic, if we were to simulate this example, the simulation would halt after Event 15 (the second active event).

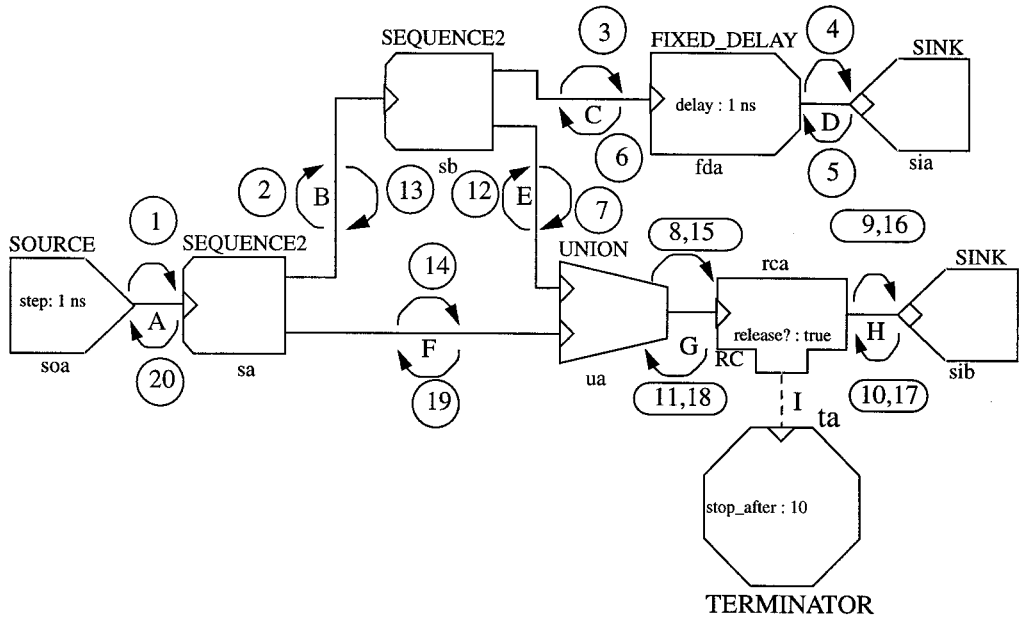


FIGURE 76.10 Token passing example showing simplified event sequence.

Module Categories

The ADEPT primitive modules may be divided into six categories: control modules, color modules, delay modules, fault modules, miscellaneous parts modules, and mixed-level modules. The control modules, except the Switch, Queue and logical modules, have been adapted from Dennis.[36] The color and delay categories of primitive modules have been defined to enable the manipulation of the coloring of tokens and to introduce the element of time into model descriptions, respectively. The fault modules are used to model the presence of faults and errors in a system model. The miscellaneous parts category contains some modules that are used for statistical data-collection with the ADEPT system. The mixed-level modules were created to aid in mixed-level modeling. The functionality of each primitive module is defined and the generics associated with each of the modules is described in the *ADEPT Library Reference Manual.*[38] The standard ADEPT module symbol convention is also explained in more detail in the *ADEPT Library Reference Manual.*

In addition to the primitive modules, there are libraries of more complex modules included in ADEPT. In general, these libraries contain modules for modeling systems in a specific applications area. These libraries are also discussed in more detail in the *ADEPT Library Reference Manual.*

Control Modules

The ADEPT representations of the 20 control modules are shown in Fig. 76.11. For each module in the figure, the XXX label represents a user-defined unique name that is assigned to each module in the system model. Further, some of the modules have generic parameters that need to be set when the module is instantiated.

The control modules operate only on the STATUS field of a signal and do not alter the color of a token on a signal. Further, no elapsed simulation time results from the activation of the control modules, since they do not have any delay associated with them.

The modules whose names end in a numeral, such as "union2", are part of a family of modules that have the same function, but differ in the number of inputs or outputs that they possess. For example, there are eight "union" modules, "union2," "union3," "union4,"… "union8."

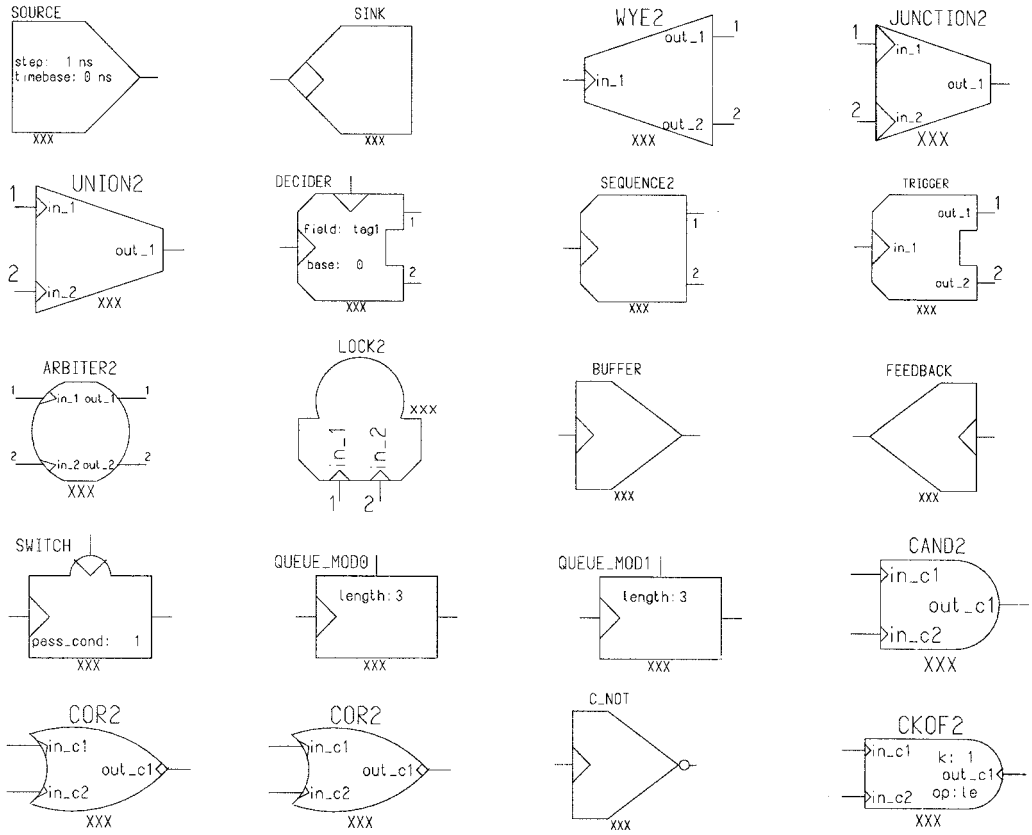


FIGURE 76.11 ADEPT control modules.

Color Modules

As noted when describing Petri Nets, using tokens that can be distinguished significantly increases the modeling capability and understanding of a given net. Therefore, ADEPT supports the ability to encapsulate pertinent system state (values) in a portion of the token called the color field. Such state information might include a memory address or an instruction opcode for a performance model of a computer system. The Control modules described in the previous section process colored tokens but do not alter the color fields of the tokens passing through them. Manipulation of the color field of a token is reserved to color modules. These color modules permit various operations on the color fields such as allowing the user to read and write the color fields of the tokens. The color modules also permit the user to compare the color information carried by the tokens and to control the flow of the tokens based on the result of the comparisons.

The ADEPT representations of the ten color modules are shown in Fig. 76.12. The use of these modules enables high-level modeling of systems at different levels of detail. These modules permit the designer to add more detail or information to the model by placing and manipulating information placed on the color fields of tokens flowing through the model. The color fields of these tokens can be set and read by these modules to represent such things as destination or source node addresses, data length, data type ("digital" or "analog," for example), or any type of such information that the designer feels is important to the design.

Delay Modules

The delay modules facilitate the description of data path delays at the conceptual or block level of a design. The ADEPT representation of the six delay modules is shown in Fig. 76.13. The fixed delay (FD) module, may be used in conjunction with the control modules to model the delay in the control structure

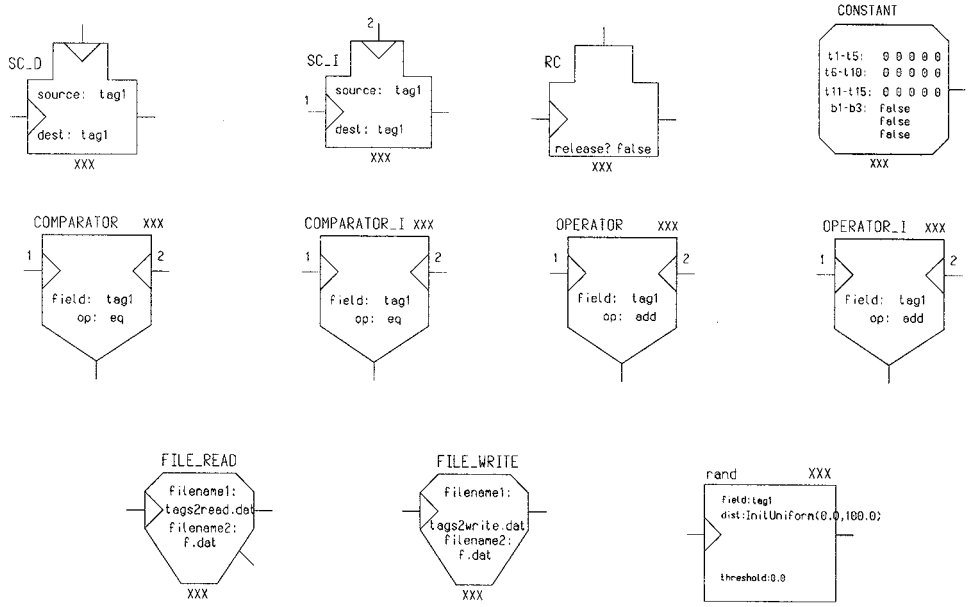


FIGURE 76.12 ADEPT color modules.

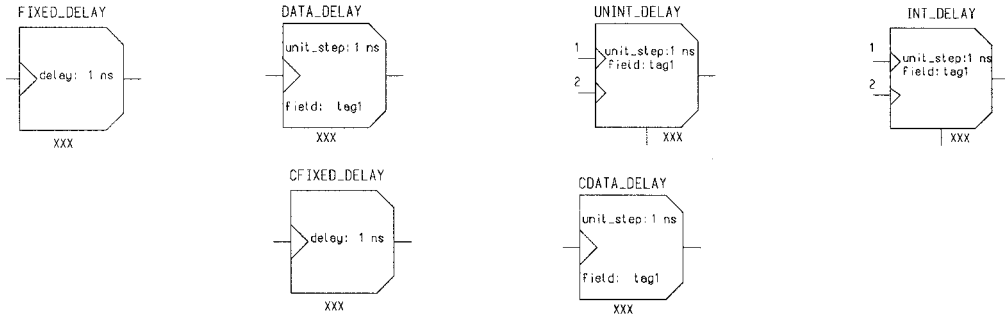


FIGURE 76.13 ADEPT delay modules.

of a design. The data-dependent delay module may be used to model processes in which the time to complete a particular operation is dependent on the data being processed. The uninterruptible data-dependent delay (UD) module and the interruptible data-dependent delay module (ID) may be used to model both hardware and software processes at the uninterpreted level. The control fixed delay (CFD) and control data-dependent delay (CDD) modules are used to add delay to independent paths in an ADEPT model that models real control processes.

Fault Modules

The fault modules are used to represent the presence of faults and errors in a system model. The modules allow the user to model fault injection, fault/error detection, and error correction processes. The ADEPT representation of the 13 Fault modules is shown in Fig. 76.14.

Miscellaneous Parts Modules

The Miscellaneous Parts category contains six modules that perform convenience functions in ADEPT. The ADEPT representation of three these modules is shown in Fig. 76.15. The collector module is used to write input activation times to a file, and the terminator module is used to halt simulation after a

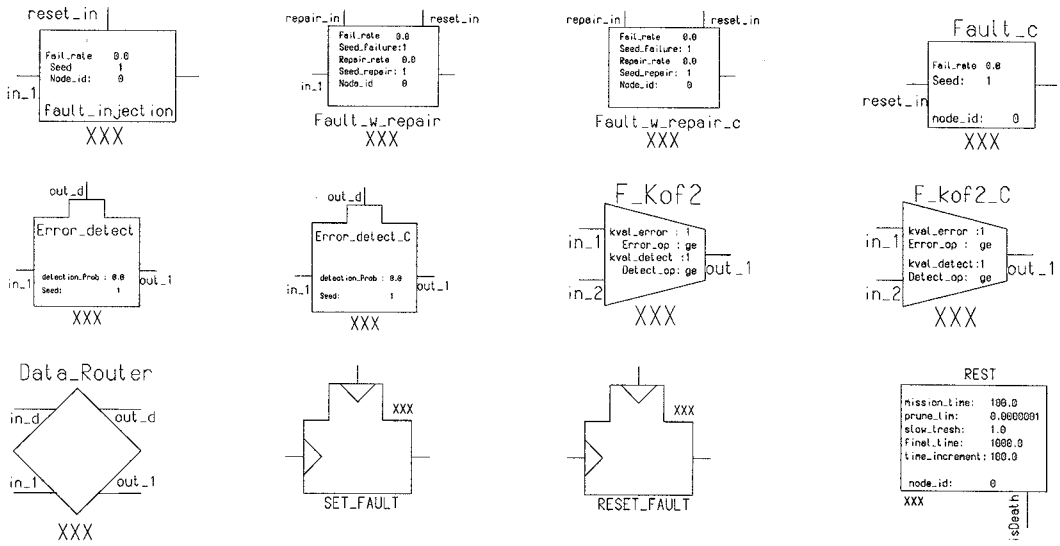


FIGURE 76.14 ADEPT fault modules.

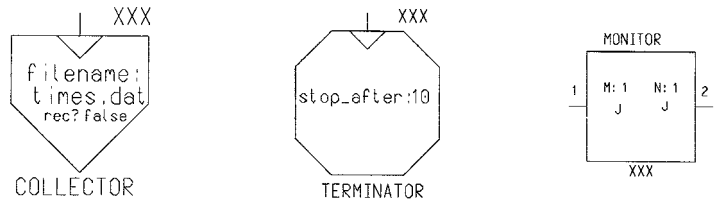


FIGURE 76.15 ADEPT miscellaneous parts modules.

specified number of events has occurred. The monitor module is a data-collection device that can be connected across other modules to gather statistical information during a VHDL simulation. This category also contains three additional modules (not shown in the figure) which are used to collect statistical information during simulation. These modules have no symbolic representation and are inserted automatically into the VHDL representation by the ADEPT tools.

Mixed-Level Modeling Modules

In order to support mixed-level modeling in the ADEPT environment, a category of building blocks called mixed-level modules was created which are used to define the interfaces around the interpreted and uninterpreted components in a system model. The functions of these modules and their use in creating mixed-level models is described in more detail in Section 76.4.

ADEPT Tools

The ADEPT system is currently available on Sun platforms using Mentor Graphics' *Design Architect* as the front-end schematic capture system, or on Windows PCs using OrCAD's *Capture* as the front-end schematic capture system. The overall architecture of the ADEPT system is shown in Fig. 76.16.

The schematic front-end is used to graphically construct the system model from a library of ADEPT module symbols. Once the schematic of the model has been constructed, the schematic capture system's netlist generation capability is used to generate an EDIF (Electronic Design Interchange Format) 2.0.0 netlist of the model. Once the EDIF netlist of the model is generated, the ADEPT software is used to

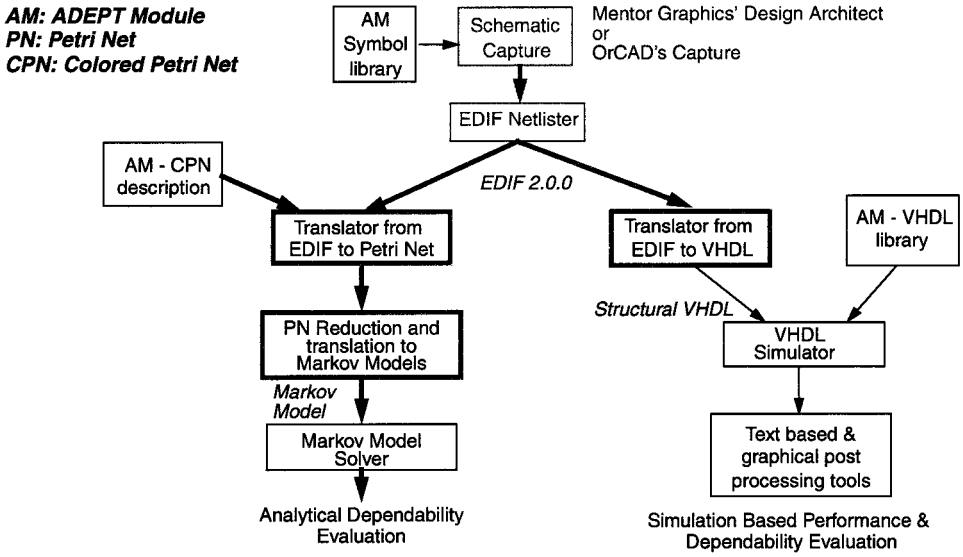


FIGURE 76.16 ADEPT design flow.

translate the model into a structural VHDL description consisting of interconnections of ADEPT modules. The user can then simulate the structural VHDL that is generated using the compiled VHDL behavioral descriptions of the ADEPT modules to obtain performance and dependability measures.

In addition to VHDL simulation, a path exists that allows the CPN description of the system model to be constructed from the CPN descriptions of the ADEPT modules. This CPN description can then be translated into a Markov model using well-known techniques and then solved using commercial tools to obtain reliability, availability, and safety information.

Figure 76.17 is an illustration of the construction of a schematic of an ADEPT model using *Design Architect*. The schematic shown is that of an ADEPT model of a simple three-computer system used in the ADEPT tutorial. Most of the elements in this top-level schematic are hierarchical, with separate schematics describing each component. The most primitive elements of the hierarchy are the ADEPT modules.

76.3 A Simple Example of an ADEPT Performance Model

This section presents a simple example of the usage of the primitive building blocks for performance modeling. The example is a three-computer system wherein the three computers share a common bus. The example also presents simulation results and system performance evaluations.

A Three-Computer System

This section discusses a simple example to illustrate how the modules discussed previously may be interconnected to model and evaluate the performance of a complete system. The system to be modeled consists of three computers communicating over a common bus, as shown in Fig. 76.18. Each block representing a computer can be thought to contain its own processor, memory and peripheral devices. The actual ADEPT schematic for the three-computer system is shown in Fig. 76.19.

Computer C1 contains some sensors and preprocessing capabilities. It collects data from the environment, converts it into a more compact form, and sends it to computer C2 via the bus. The computer C2 further processes the data and passes it to computer C3 where the data is appropriately utilized. It is assumed that data is transferred in packets and each packet of data is of varying length. In the example described here, computers C1 and C2 receive packets whose sizes are uniformly distributed between 0

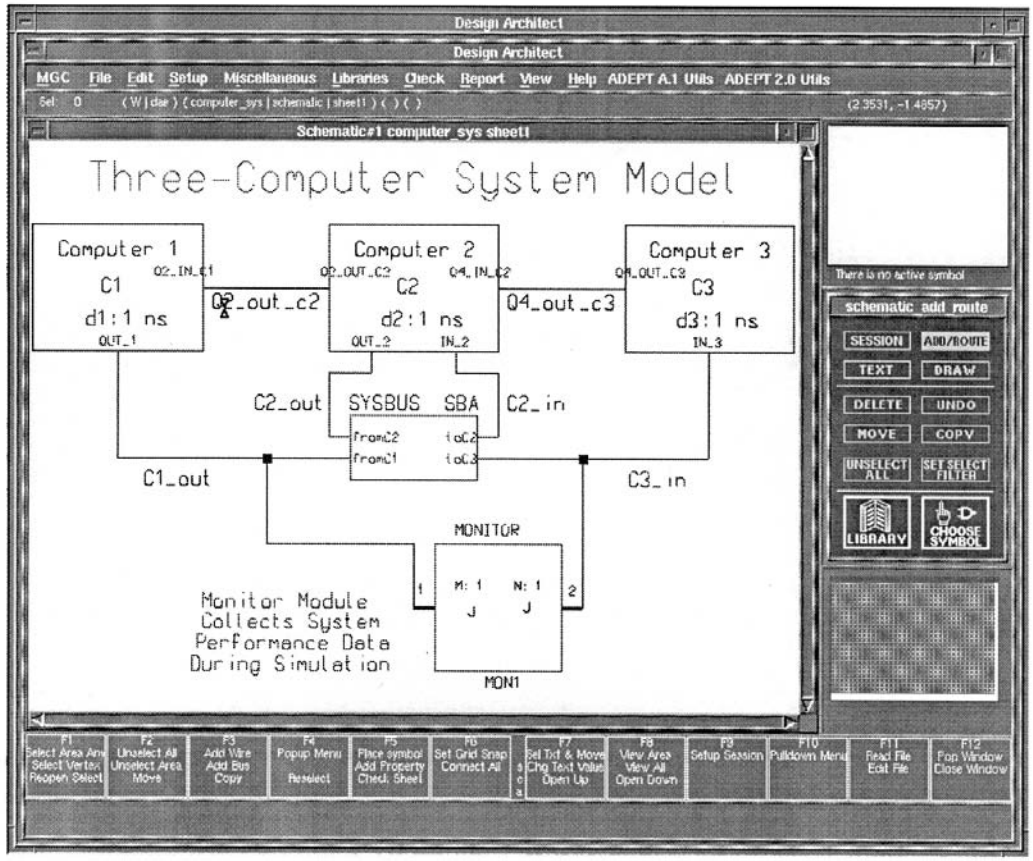


FIGURE 76.17 Sample ADEPT schematic (*Design Architect*).

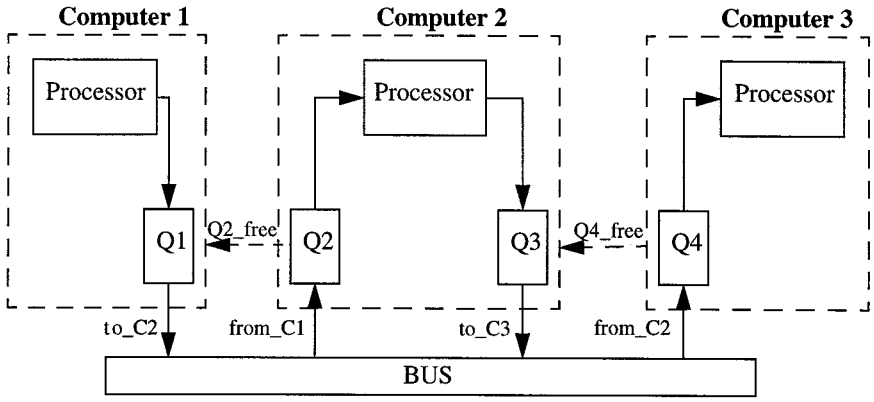


FIGURE 76.18 A three-computer system.

and 100. The packet size of computer C3 is uniformly distributed between 0 and 500. The external environment in this example is modeled by a Source module in C1 and a sink module in C3.

C1 has an output queue, C2 has both an input queue and an output queue, while C3 has one input queue. All queues in this example are assumed to be of length 8. If the input queues of C2 or C3 are full, the corresponding Q_free signal is released (value = RELEASED). This interconnection prevents the

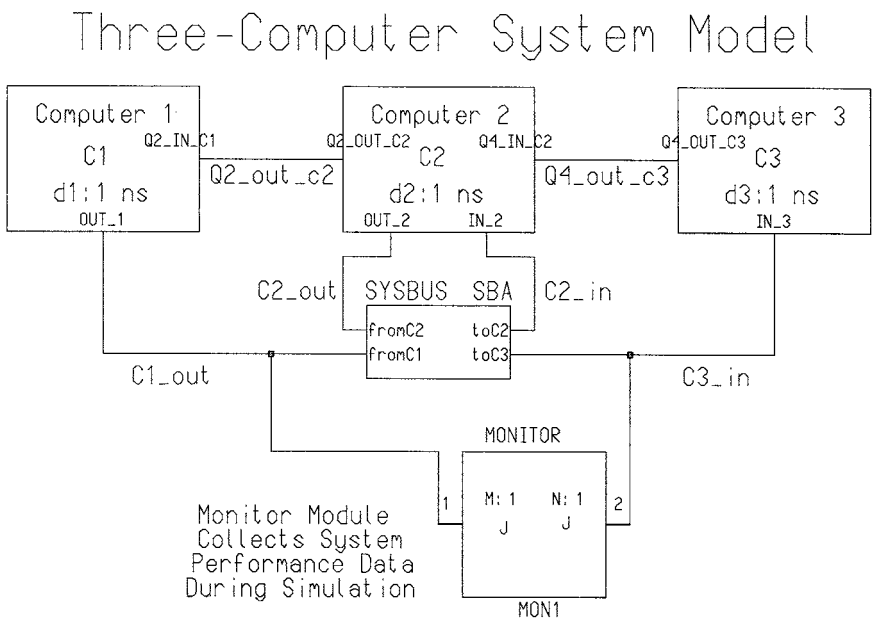


FIGURE 76.19 A three-computer system ADEPT schematic.

computer writing into the corresponding queue from placing data on the bus when the queue is full. This technique not only prevents the bus from being unnecessarily held up but also eliminates the possibility of a deadlock.

The ADEPT model of Computer 1 (C1) is shown in Fig. 76.20. The Source A (SOA) module along with the SA, Set_Color A (SCA), and random A (RA) modules generate tokens whose *tag1* field is set according to a distribution which is representative of the varying data sizes that C1 receives. The data-dependent delay A (DDA) models the processing time of C1 which is directly proportional to the packet size. The *unit_step* delay for the DDA module is passed down as a generic *d1**1ns. As soon as a token, representing one packet of data, appears at the output of the DDA module, the SB, Set_Color B (SCB), and random B (RB) modules together set the *tag1* field of the token to represent the packet size after processing. The constant A (COA) and Set_Color C (SCC) modules set the *tag2* field of the token to 2. This coloring indicates to the bus arbitration unit that the token is to be transferred to C2. The token is then placed in the output queue (Q1) of C1 and the token is acknowledged. This acknowledge signal is passed back to the source A module, which then produces the next token. The fixed delay (FDA & FDB) modules represent the read and write times associated with the queue. The switch A (SWA) element is controlled by the Q_free signal from C2 and prevents a token from the output queue of C1 from being placed on the bus if the incoming Q_Free signal is inactive.

Figure 76.21 shows the ADEPT model of Computer 2 (C2). When a token arrives at the *data* input of C2, the token is placed at the input of the queue (Q2). The *control* output of the queue becomes the Q2_Free output of C2. The remaining modules perform the same function as in C1 except that the tag2 field of the tokens output by C2 is set to 3, which indicates to the bus arbitration unit that the token is to be sent to C3. The DDA module represents the relative processing speed of Computer C2. The *unit_step* delay for the DDA module is passed down as a generic *d2**1ns.

The modules defining Computer 3 (C3) are shown in Fig. 76.22. A token arriving at the input is placed in the queue. The DDA element reads one token at a time and provides the delay associated with the processing time of C3 before the Sink A (SIA) module removes the token. The *unit_step* delay for the DDA module is passed down as a generic *d3**1ns.

The bus is shown in Fig. 76.23. Arbitration is provided to ensure that both C1 and C2 do not write onto the bus at the same time. The arbiter A (AA) element provides the required arbitration. Since the

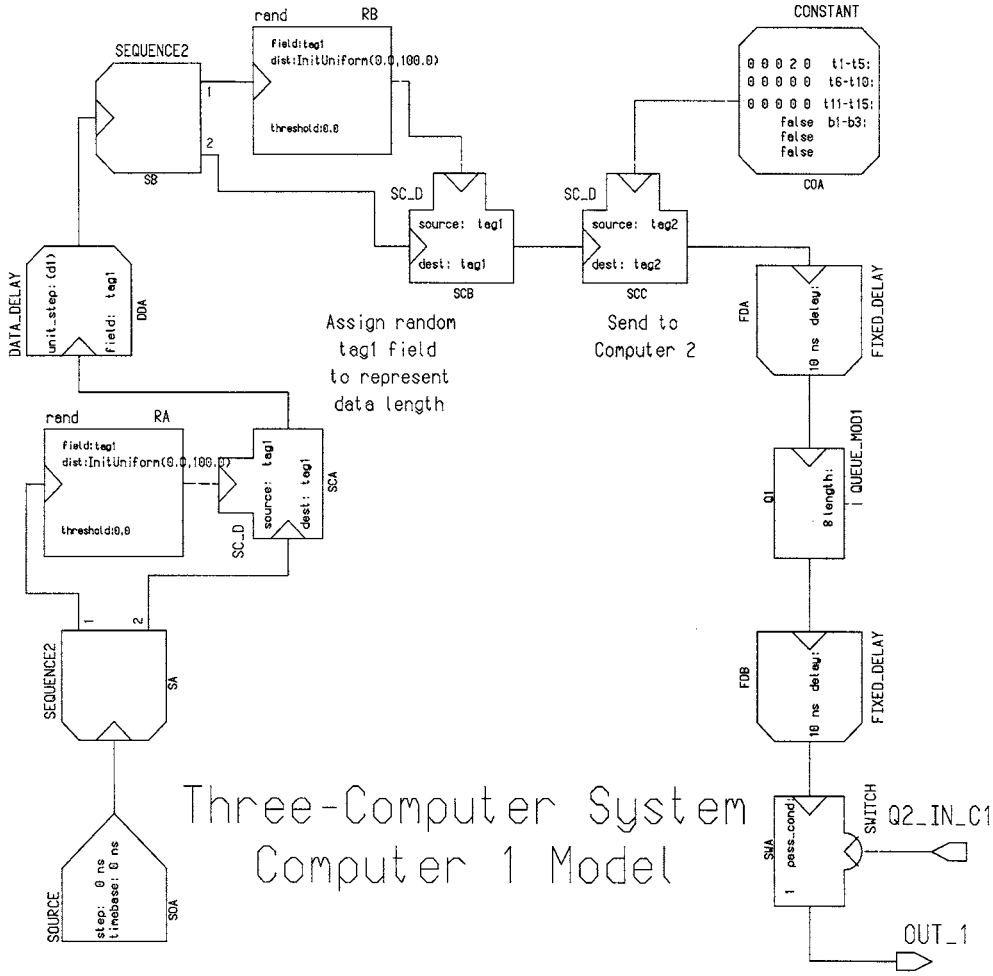


FIGURE 76.20 ADEPT model of computer 1.

output of C2 is connected to the IN_1(1) input of the AA element, it has a higher priority over C1. The union A (UA) element passes a token present at either of its inputs to its output. The output of the UA element is connected to the DDA element, which models the packet transfer delay associated with moving packets over the bus. The delay through the bus is dependent on the size of the packet of information being transferred (size stored on the *tag1* field). Note that in this analysis, the bus delay was set to zero. The independent output of the Read_Color A(RCA) element is connected to the control input of the Decider A (DA) module. The base of the DA element is set to 2. Since the *tag2* field of the token is set to 2 or 3 depending on whether it originated from C1 or C2, the first output of the DA module is connected to C2 and the second output is connected to C3. This technique ensures that the tokens are passed on to the correct destination, C2 or C3.

Simulation Results

This section presents simulation results that illustrate the queuing model capabilities of the ADEPT system. The model enables the study of the overall speed of the system in terms of the number of packets of information transferred in a given time. It also allows the study of the average number of tokens present in each queue during simulation, and the effect of varying system parameters on the number of items in the queues and overall throughput of the system. The generic unit_step delay of the DD elements

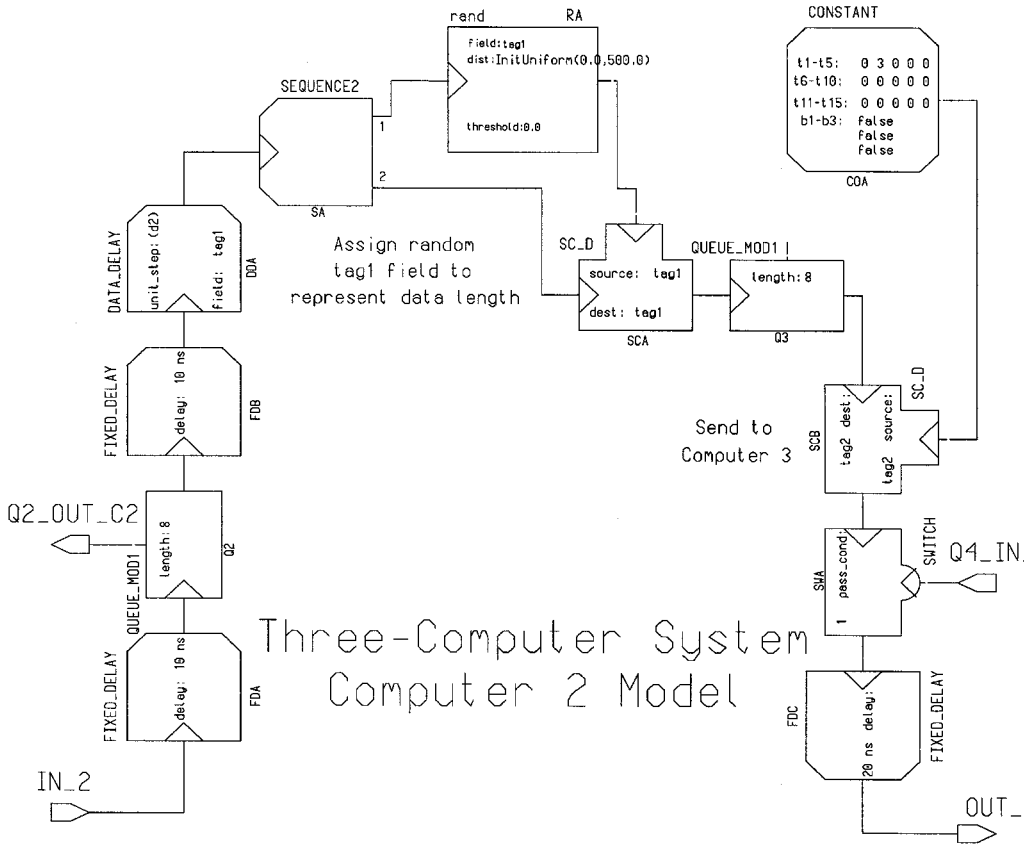


FIGURE 76.21 ADEPT model of computer 2.

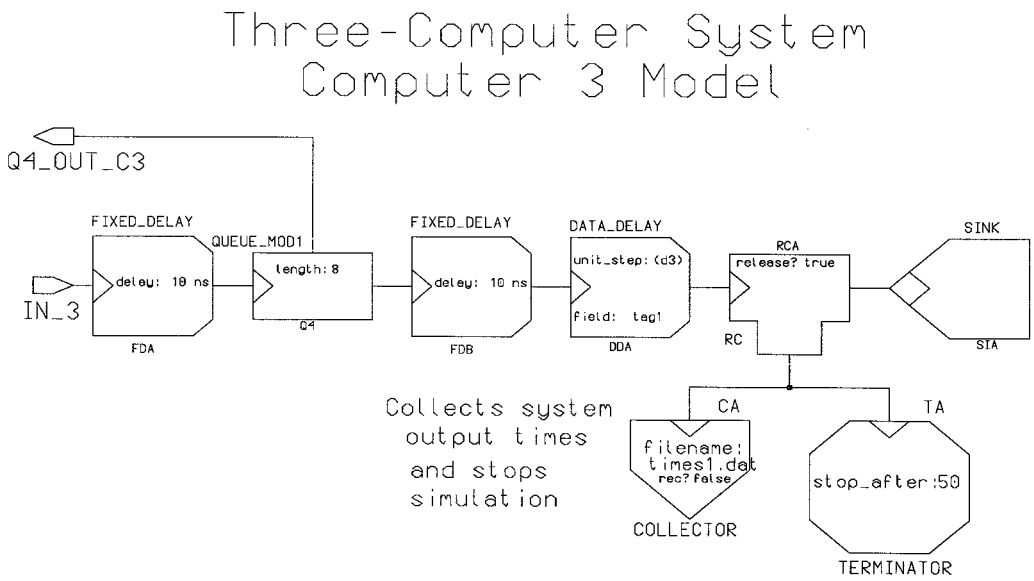


FIGURE 76.22 ADEPT model of computer 3.

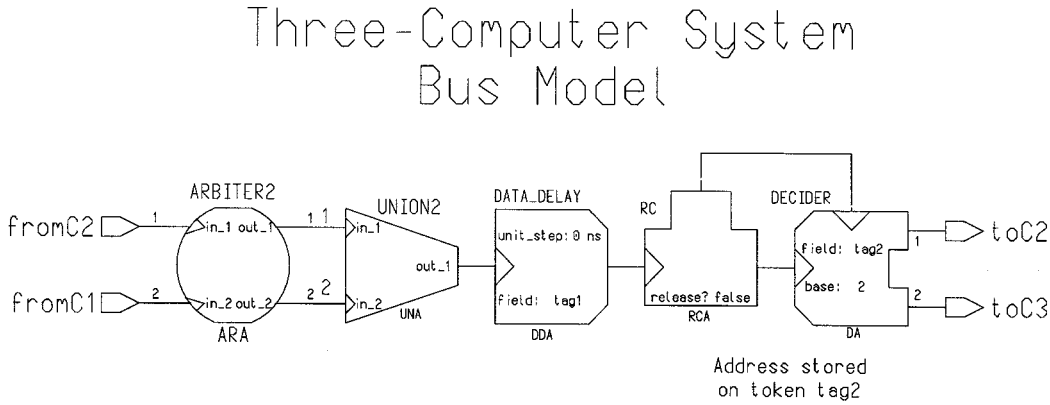


FIGURE 76.23 ADEPT model of system bus.

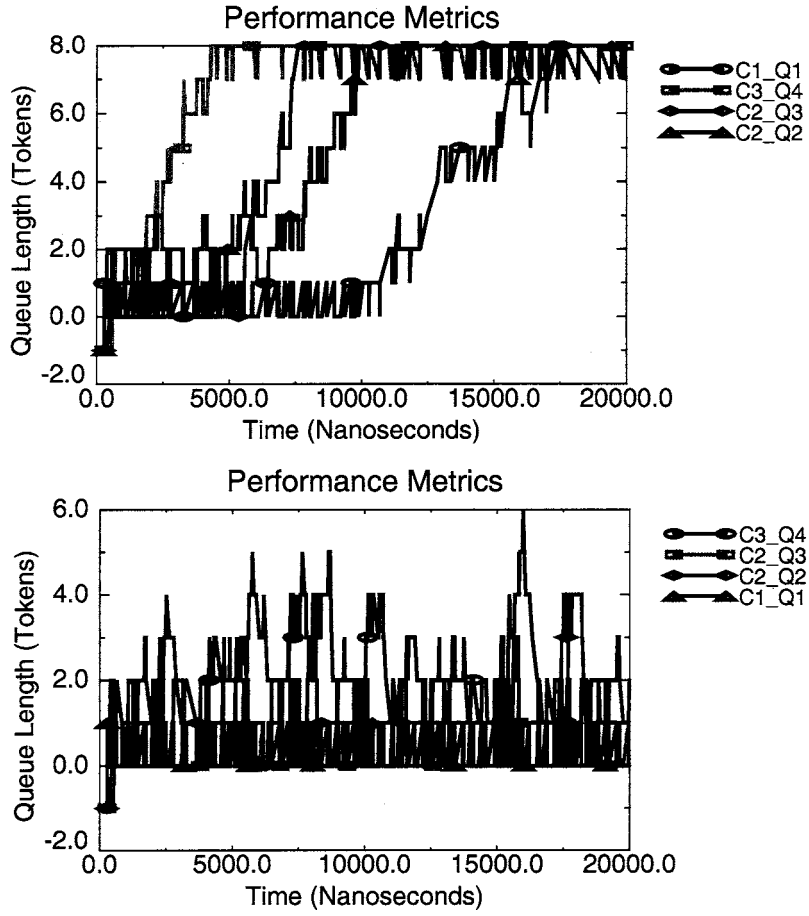


FIGURE 76.24 Queue lengths vs. simulation time for various processing delay times.

(*d1*, *d2*, and *d3*) associated with the three computers is representative of the processing speed of the computers. The optimal relative speeds of the computers may also be deduced by simulation of this model. Figure 76.24 shows graphs of the number of items in each queue versus simulation time. These graphs were generated using the BAARS postsimulation analysis tool. The upper graph shows the queue

lengths when *d1*, *d2*, *d3* was set to 5 ns, 5 ns, and 2 ns. The lower graph shows the queue lengths when *d1*, *d2*, *d3* was set to 5 ns, 4 ns, and 1 ns. In the first case, because the processing time of Computer 3 was so much longer than Computers 1 or 2, the queue in Computer 3 became full at approximately 4000 ns of simulation time. The filling of the queue in Computer 3 delayed items coming out of Computers 1 and 2, thus causing their queues to also become full. In the second case, the processing time ratios were such that Computer 3 could keep up with the incoming tokens and the queues never got completely full.

Table 76.1 summarizes the effect of relative speeds of the computers on the number of packets transferred. Since the size of the packets received by C3 is uniformly distributed between 0 and 500 while the size of the packets received by C1 and C2 is uniformly distributed between 0 and 100, it is intuitively obvious that the overall throughput of the system is largely determined by the speed of computer C3. The results do indicate this behavior. For example, when the relative instruction execution times for C1, C2, and C3 are 5, 5, and 2, respectively, a total of 197 packets are transferred. By increasing the instruction execution time of C2 by one time unit and decreasing the instruction execution time of C3 by one time unit, it is seen that a total of 321 packets are transferred, an increase of slightly over 60%.

TABLE 76.1 System Throughput vs. Computer Delay Times

C1 delay (ns)	C2 delay (ns)	C3 delay (ns)	Packets per 100000 time units (ns)
8	8	3	131
9	10	2	194
5	5	2	197
3	5	2	197
2	2	2	197
5	6	1	321
5	4	1	322
4	3	1	384
3	2	1	385
2	2	1	386

This example has illustrated the use of the various ADEPT modules to model a complete system. Note how the interconnections between modules describing a component of the system are similar to a flow chart describing the behavior of the component. This example also demonstrated that complex systems at varying levels of abstraction and interpretation can easily be modeled using the VHDL-based ADEPT tools.

76.4 Mixed-Level Modeling

As described earlier, performance (uninterpreted) modeling has been previously used primarily by systems engineers when performing design analysis at the early stages of the design process. Although most of the design detail is not included in these models, since this detail does not yet exist, techniques such as token coloring can be used to include that design detail that is necessary for an accurate model. However, most of the detail, especially very low-level information such as word widths and bit encoding, are not present. In fact, many additional design decisions must be made before an actual implementation could ever be constructed. In performance models constructed in ADEPT, abstract tokens are used to represent this information (data and control) and its flow in the system. An illustration of such a performance model with its analysis was given the three-computer system described in Section 76.3. On the other hand, behavioral (interpreted) models can be thought of as including much more design detail often to the point that an implementation could be constructed. Therefore, data values and system timing are available and variables typically take on integer, real, or bit values. For example, a synthesizable model of a carry-lookahead adder would be one extreme of a behavioral model.

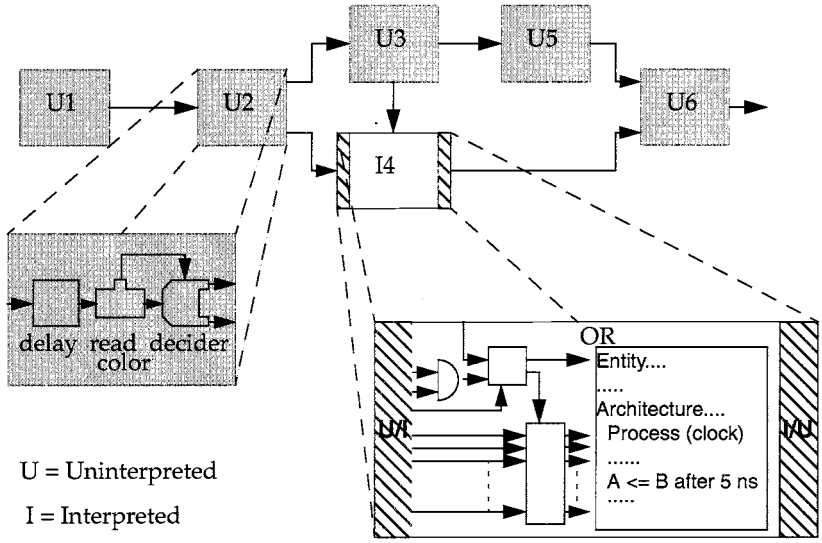


FIGURE 76.25 General structure of a mixed-level model.

It should be obvious that the two examples described above are extremes of the two types of models. Extensive design detail can be housed in the tokens of a performance model and information in a behavioral model can be encapsulated in a very abstract type. However, there is always a difference in the abstraction level of the two modeling types. Therefore, in order to develop a model that can include both performance and behavioral models, interfaces between the different levels of design abstraction, called mixed-level interfaces, must be included in the overall model to resolve differences between these two modeling domains.

The mixed-level interface is divided into two primary parts which function together: the part that handles the transition from the uninterpreted domain to the interpreted domain (U/I) and the part that handles the transition from the interpreted domain to the uninterpreted domain (I/U). The general structure of a mixed-level model is shown in Fig. 76.25.

In addition to the tokens-to-values (U/I) and values-to-tokens (I/U) conversion processes, the mixed-level interface must resolve the differences in design detail that naturally exist at the interface between uninterpreted and interpreted elements. These differences in detail appear as differences in data and timing abstraction across the interface. The differences in timing abstraction across the interface arise because the components at different levels model timing events at different granularities. For example, the passing of a token that represents a packet of data being transferred across a network in a performance model may correspond to hundreds or thousands of bus cycles for a model at the behavioral level.

The differences in data abstraction across the interface are due to the fact that typically performance models do not include full functional details, whereas behavioral models require full functional data (in terms of values on their inputs) to be present before they will execute correctly. For example, a performance level modeling token arriving at the inputs to the mixed-level interface for a behavioral floating point coprocessor model may contain information about the operation the token represents, but it may not contain the data on which the operation is to take place. In this case, the mixed-level interface must generate the data required by the behavioral model and do it in a way that a meaningful performance metric, such as best or worst case delays, is obtained.

Mixed-Level Modeling Taxonomy

The functions that the mixed-level interface must perform and the most efficient structure of the interface is affected by several attributes of the system being modeled and the model itself. In order to partition

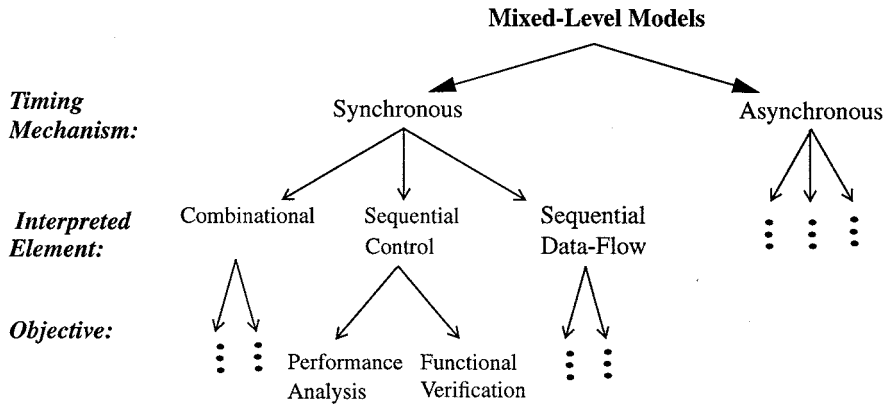


FIGURE 76.26 Mixed-level modeling categories.

the mixed-level modeling space and better define the specific solutions, a taxonomy of mixed-level model classes has been developed.[39] The classes of mixed-level modeling are defined by those model attributes which fundamentally alter the development and the implementation of the mixed-level interface. The mixed-level modeling space is partitioned according to three major characteristics:

1. The evaluation objective of the mixed-level model,
2. The timing mechanism of the uninterpreted model, and
3. The nature of the interpreted element.

For a given mixed-level model, these three characteristics can be viewed as attributes of the mixed-level model and the analysis effort. Figure 76.26 summarizes the taxonomy of mixed-level models.

Mixed-Level Modeling Objectives

The structure and the functionality of the mixed-level interface are strongly influenced by the objective that the analysis of the mixed-level model will be targeted toward. For the purposes of this work, these objectives were broken down into two major categories:

1. *Performance analysis* and *timing verification*: To analyze the performance of the system (as defined previously) and verify that the specific component(s) under consideration meet system timing constraints. Note that other metrics, such as power consumption, could be analyzed using mixed-level models, but these were outside the scope of this classification.
2. *Functional verification*: To verify that the function (input-to-output value transformation) of the interpreted component is correct within the context of the system model.

Timing Mechanisms

Typically, performance (uninterpreted) models are asynchronous in nature, and the flow of tokens depends on the handshaking protocol. However, in modeling a system that is globally synchronous (all elements are synchronized to a global clock) a mechanism to synchronize the flow of tokens across the model can be introduced. This synchronization of the performance model will require different mixed-level modeling approaches. Thus, the two types of system models that affect the mixed-level interface are

1. *Asynchronous models*: Tokens on independent signal paths within the system model move asynchronously with respect to each other and arrive at the interface at different times.
2. *Synchronous models*: The flow of tokens in the system model is synchronized by some global mechanism, and they arrive at the interface at the same time according to that mechanism.

Interpreted Component

The mixed-level modeling technique strongly depends on the type of the interpreted component that is introduced into the performance model. It is natural to partition interpreted models into those that model combinational elements and those that model sequential elements. Techniques for constructing mixed-level interfaces for models of combinational interpreted elements have been developed previously.[40] In the combinational element case, the techniques for resolving the timing across the interface were more straightforward because of the asynchronous nature of the combinational elements, and the data abstraction problem was solved using a methodology similar to the one presented here. However, using sequential interpreted elements in mixed-level models requires more complex interfaces to solve the synchronization problem and different specific techniques for solving the data abstraction problem. Further research into the problem of mixed-level modeling with sequential elements suggested that interpreted components be broken down into three classifications as described below:

1. *Combinational Elements*: Unclocked (with no states) elements.
2. *Sequential Control Elements* (SCE): Clocked elements (with states) that are used for controlling data flow, e.g., a control unit or a controller
3. *Sequential Dataflow Elements* (SDE): Elements that include datapath elements and clocked elements that control the data flow, e.g., control unit and datapath.

The major reason for partitioning the sequential elements into sequential dataflow and sequential control elements is based on the timing attributes of these elements. In a SCE, control input values are read every cycle and control output values (that control a datapath) are generated every cycle. On the other hand, SDEs have data inputs and may have some control inputs, but the output data is usually generated several clock cycles later. This difference in the timing attributes will dictate a different technique for mixed-level modeling. Because the solution for the timing and data abstraction problem for SDE elements is more complex and the solution for SCE elements can be derived from the solution for SDE elements, developing a solution for SDE elements was the focus of this work.

An Interface for Mixed-Level Modeling with FSMD Components

This section describes the interface structure and operation for sequential interpreted elements that can be described as Finite State Machines with a Datapath (FSMDs).[41] They consist of a Finite State Machine (FSM) used as a control unit, and a datapath, as shown in Fig. 76.27. System models are quite often

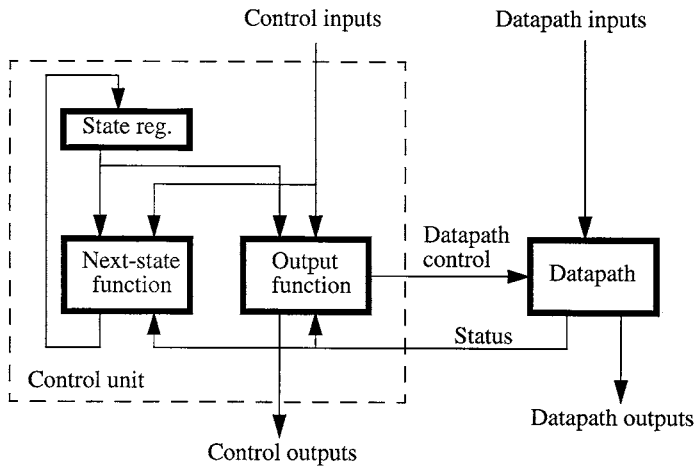


FIGURE 76.27 Generic FSMD block diagram.

naturally partitioned to blocks which adhere to the FSMD structure. Each FSMD processes the data and has some processing delays associated with it. These FSMDs indicate the completion of a processing task to the rest of the system either by asserting a set of control outputs or by the appearance of valid data on their outputs. This characteristic of being able to determine when the data processing task of the FSMD is completed is a key property in the methodology of constructing mixed-level models with FSMD interpreted components.

The functions performed by the U/I operator include placing the proper values on the inputs to the FSMD interpreted model and generating a clock signal for the FSMD interpreted model. The required values to be placed on the inputs to the FSMD interpreted model are contained on the incoming token's information, or "color" fields, are supplied by the modeler via some outside source, or are derived using the techniques described in the next section. The clock signal is either generated locally if the system-level model is globally asynchronous or converted from the global clock into the proper format if the system-level model is globally synchronous. The functions performed by the I/U operator include releasing the tokens back into the performance model at the appropriate time and, if required, coloring them with new values according to the output signals of the interpreted element. The structure of these two parts of the mixed-level interface is shown in Fig. 76.28. The U/I operator is composed of the following blocks: a driver, an activator, and a clock_generator. The I/U operator is composed of an Output_Condition_Detector, a Colorer, and a Sequential_Releaser.

In the U/I operator, the activator is used to detect the arrival of a new token (packet of data) to the interpreted element, inform the driver of a new token arrival, and drive control inputs of the interpreted element. The activator's output is also connected to the I/U operator for use in gathering information on the delay through the interpreted element. The driver reads information from the token's color fields and drives the proper input signals to the interpreted element according to predefined assignment properties. The Clock_Generator generates the clock signal according to the overall type of system-level model, either synchronous or asynchronous.

In the I/U operator, the Output_Condition_Detector detects the completion of the interpreted element data processing operation, as discussed earlier, by comparing the element's outputs with predefined properties. The colorer samples the datapath outputs and maps them to color fields according to predefined binding properties. The Sequential_Releaser, which "holds" the original token, releases it back to the uninterpreted model on receiving the signal from the Output_Condition_Detector. The information carried by the token is then updated by the colorer, and the token flows back to the uninterpreted part of the model.

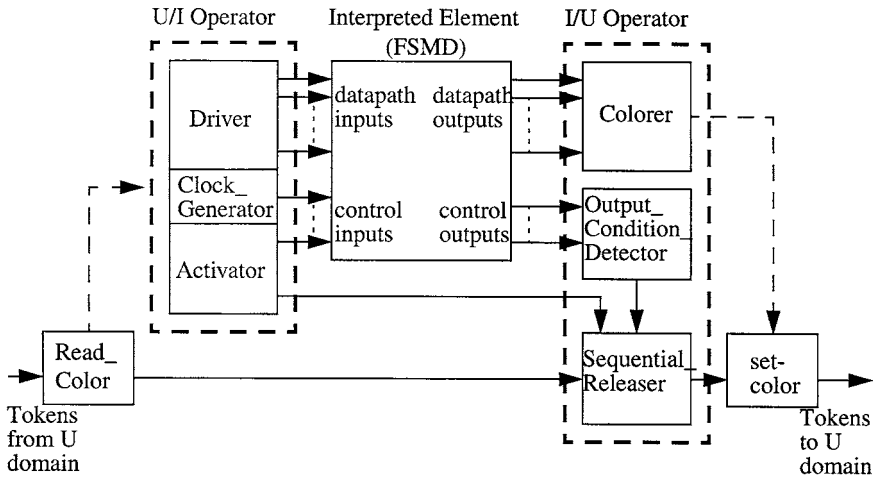


FIGURE 76.28 The mixed-level element structure.

Given this structure, the operation of the mixed-level model can be described using the general example in Fig. 76.28. Upon arrival of a new token to the mixed-level interface (U/I operator), the Read_color module triggers the activator component and passes the token to the Sequential_Releaser where it is stored until the interpreted component is finished with its operation. Once triggered by the Read_Color module, the activator notifies the driver to start the data conversion operation on a new packet of data. In the case of a globally asynchronous system-level model, the activator will notify the Clock_Generator to start generating a clock signal. Since the interpreted element is a sequential machine, the driver may need to drive sequences of values onto the inputs of the interpreted element. This sequence of values is supplied to the interpreted element, while the original token is held by the Sequential_Releaser. This token is released back to the uninterpreted model only when the Output_Condition_Detector indicates the completion of the interpreted element operation. The Output_Condition_Detector is parameterized to recognize the completion of data processing by the particular FSMD interpreted component. Once the Output_Condition_Detector recognizes the completion of data processing, it signals the Sequential_Releaser to release the token into the performance model. Once the token is released, it passes through the colorer, which maps the output data of the interpreted element onto color fields of the token. The new color information on the token may be used by the uninterpreted model for such things as delays through other parts of the model or for routing decisions.

Finding Values for the "Unknown Inputs" in an FSMD-Based Mixed-Level Model

As described previously, because of the abstract nature of a performance model, it may not be possible to derive values for all of the inputs to the interpreted component from the data present in the performance model. Typically, the more abstract the performance model (i.e., the earlier in the design cycle), the higher the percentage of input values that will be unknown. In some cases, particularly during the very early stages of the design process, it is possible that the abstract performance model will not provide any information to the interpreted element, other than the fact that new data has arrived. In this case, the data abstraction gap will be large. This section describes an analytical technique developed to determine values for these "unknown inputs" such that a meaningful performance metric — best or worst case delay — can be derived from the mixed-level model.

If some (or all) inputs are not known from the performance model, some criteria for deriving the values on the unknown inputs must be made. Choosing a criterion is based on the objective of the mixed-level model. For the objective of timing verification, delays (number of clock cycles) through the interpreted element are of interest. The most common criterion in such cases is the worst-case processing delay. In some cases, best-case delay may be desired. If the number of unknown inputs is small, an exhaustive search for worst/best case may be practical. Therefore, it is desirable to minimize the number of unknown inputs that can affect the delay through the interpreted element. The methods for achieving this objective are described conceptually in the next section. By utilizing these methods, the number of unknown inputs is likely to be reduced, but unknown inputs will not be eliminated completely. In this case, the performance metrics of best- and worst-case delay can be provided by a "traversal" of the state graph of the SDE component. The next section describes the traversal method developed for determining the delays through a sequential element.

Note that in the algorithms subsequently described, the function of the SDE component is represented by the State Transition Graph (STG) or State Table. These two representations are essentially equivalent and are easily generated from a behavioral VHDL description of the SDE, either by hand, or using some of the automated techniques that have been developed for formal verification.

Reducing the Number of Unknown Inputs

Although it is not essential, reducing the number of unknown inputs can simplify the simulation of a mixed-level model significantly. Since the FSMD elements being considered have output signals that can be monitored to determine the completion of data processing as discussed previously, other outputs may not be significant for performance analysis. Therefore, the "nonsignificant" (insignificant) outputs can

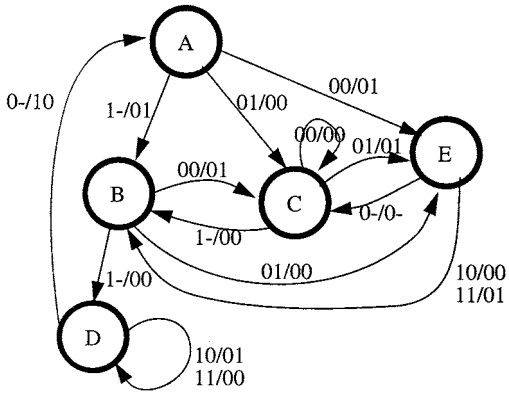


FIGURE 76.29 STG of a 2-inputs, 2-outputs state machine.

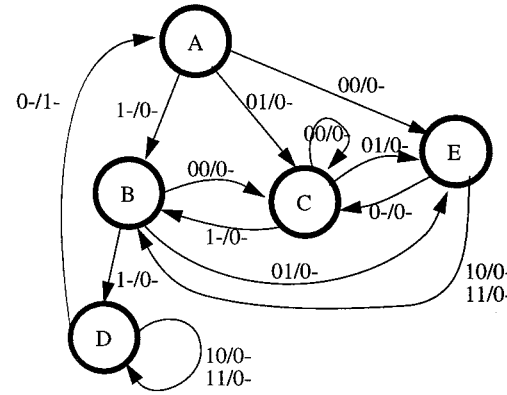


FIGURE 76.30 STG with "significant" output values only.

be considered as "don't-cares." By determining which inputs do not affect the values on the significant outputs, it is possible to minimize the number of unknown delay affecting inputs (DAIs).

The major steps in the DAI detection algorithm are

Step 1: Select the "insignificant" outputs (in terms of temporal performance).
Step 2: In the STG of the machine, replace all values for these outputs with a don't-care to generate the modified state machine.
Step 3: Minimize the modified state machine and generate the corresponding state table.
Step 4: Find the inputs which do NOT alter the flow in the modified state machine by detecting identical columns in the State Table and combining them by implicit input enumeration.

This method is best illustrated by an example. Consider the state machine which is represented by the state transition graph shown in Fig. 76.29. This simple example is a state machine with two inputs, X_1 and X_2, and two outputs, Y_1 and Y_2. This machine cannot be reduced, i.e., it is a minimal state machine. Assume that this machine is the control unit of an FSMD block and that the control output Y_1 is the output which indicates the completion of the "data processing" when its value is 1. Thus, output Y_2 is an "insignificant output" in terms of delay in accordance with Step 1. Therefore, a don't-care value is assigned to Y_2 as per Step 2. The modified STG is shown in Fig. 76.30. As per Step 3, the modified STG is then reduced. In this example, states A, C, and E are equivalent (can be replaced by a single state, K) and the minimal machine consists of three states, K, B, and D. This minimal machine is described by the State Table shown in Table 76.2.

TABLE 76.2 Next-State and Output Y_1 of the Minimal Machine

P.S.\X_1X_2	00	01	10	11
K = (ACE)	K, 0	K, 0	B, 0	B, 0
B	K, 0	K, 0	D, 0	D, 0
D	K, 1	K, 1	D, 0	D, 0

All possible input combinations appear explicitly in Table 76.2. However, it can be seen that the first two columns of the table are identical (i.e., the same next state and output value for all possible present states). Similarly, the last two columns of the table are identical. Therefore, in accordance with Step 4, these columns can be combined yielding the reduced State Table shown in Table 76.3.

TABLE 76.3 The Minimal Machine with Implicit Input Enumeration

P.S. \ X1X2	0-	1-
K = (ACE)	K, 0	B, 0
B	K, 0	D, 0
D	K, 1	D, 0

The reduced state table reveals that the minimal machine does not depend on the value of input X_2. Therefore, the conclusion is that input X_2 is not a DAI. This result implies that by knowing only the value of input X_1, the number of clock cycles (transitions in the original STG) required to reach the condition that output $Y_1 = 1$ can be determined regardless of the values of X_2. This is the case for any given initial state. It is important to emphasize that the paths in the original STG and their lengths are those which must be considered

during the traversal process and that the modified state machine is used only for the purpose of detecting non-DAIs. The machine which is actually being traversed during the mixed-level simulation is the original state machine with all its functionality.

To demonstrate the meaning of an input which is not a DAI, consider the original state machine represented by the graph in Fig. 76.29 and assume that the initial state is A. Consider, for example, one possible sequence of values on input X_1 to be 0, 1, 0, 0, 1, 1, 0. By applying this input sequence, the sequence of values on output Y_1 is 0, 0, 0, 0, 0, 0, 1, regardless of the values applied to input X_2. Therefore, two input sequences which differ only in the values of the non-DAI input X_2 will produce the same sequence of values on the "significant" output Y_1. For example, the sequence X_1X_2 = 00, 10, 00, 01, 10, 10, 01 will drive the machine from state A to E, B, C, E, B, D, and back to A, and the sequence of values on output Y_1 will be as above. Another input sequence, X_1X_2 = 01, 11, 01, 00, 11, 11, 00, in which the values of X_1 are identical to the pervious sequence, will drive the machine from state A to C, B, E, C, B, D, and back to A, while the sequence of values on output Y_1 is identical to the previous case. Therefore, the two input sequences will drive the machine via different states but will produce an identical sequence of values on the "significant" output (which also implies that the two paths in the STG have an equal length).

Traversing the STG for Best and Worst Delay

After extracting all possibilities for minimizing the number of unknown inputs, a method for determining values for those unknown inputs which are DAIs is required. The method developed for mixed-level modeling is based on the traversal of the original STG of the sequential interpreted element. As explained earlier, some combination of output values may signify the completion of processing the data. The search algorithm will look for a minimum or maximum number of state transitions (clock cycles) between the starting state of the machine and the state (Moore machine) or transition (state and input combination — Mealy machine), which generates the output values which signify the completion of data processing. Once this path is found, the input values for the unknown DAIs that will cause the SDE to follow this state transition path will be read from the STG and applied to the interpreted component in the simulation to generate the required best- or worst-case delay. Since the state machine is represented by the STG, this search is equivalent to finding the longest or shortest path between two nodes in a directed graph (digraph).

The search for the shortest-path utilizes a well-known algorithm. Search algorithms exist for both single-source shortest-path and all-pairs shortest-path. One of the first and most commonly used algorithm is Dijkstra's algorithm,[42] which finds the shortest-path from a specified node to any other node in the graph. The search for all-pairs shortest-path is also a well-investigated problem. One such algorithm by Floyd[43] is based on work by Warshall.[44] Its computation complexity is $O(n^3)$ when n is the number of nodes in the graph, which makes it quite practical for moderate-sized graphs. The implementation of this algorithm is based on Boolean matrix multiplication, and the actual realization of all-pairs shortest-paths can be stored in an $n \times n$ matrix. Utilizing this algorithm required some enhancements in order to make it applicable to mixed-level modeling. For example, if some of the inputs to the interpreted element are known (from the performance model), then the path should include transitions that include these known input values.

On the other hand, the search for the longest path is a more complex task. It is an NP-complete problem that has not attracted significant attention. Since most digraphs contain cycles, the cycles need to be handled during the search in order to prevent a path from containing an infinite number of cycles. One possible restriction that makes sense for many state machines that might be used in mixed-level modeling is to construct a path that will not include a node more than once. Given a digraph $G(V,E)$ which consists of a set of vertices (or nodes) $V = (v_1, v_2, \ldots\}$ and a set of edges (or arcs) $E = \{e_1, e_2, \ldots)$, a simple-path between two vertices v_{init} and v_{fin} is a sequence of alternating vertices and edges $P = v_{init}$, e_n, v_m, e_{n+1}, v_{m+1}, e_{n+2}, …, v_{fin} in which each vertex does not appear more than once. Although an arbitrary choice was made to implement the search allowing each vertex to appear in the path only once, the same algorithm could be easily modified to allow the appearance of each vertex a maximum of N times.

Given an initial node and a final node, the search algorithm developed for this application starts from the initial node and adds nodes to the path in a depth-first-search (DFS) fashion until the final node is reached. At this point, the algorithm backtracks and continues looking for a longer path. However, since the digraph may be cyclic, the algorithm must avoid the possibility of increasing the path due to a repeated cycle, which may produce an infinite path.

The underlying approach for avoiding repeated cycles in the algorithm dynamically eliminates the cycles while searching for the longest-simple-path. Let u be the node that the algorithm just added to the path. All the in-arcs to node u can be eliminated from the digraph at this stage of the path construction. The justification for this dynamic modification of the graph is that, while continuing in this path, the simple-path cannot include u again. While searching forward, more nodes are being added to the path and more arcs can be removed temporarily from the graph. At this stage, two things may happen: (1) either the last node being added to the path is the final node, or (2) the last node has no out-arcs in the dynamically modified graph. These two cases are treated in the same way except that in the first case the new path is checked to see if it is longer than the longest one found so far. If it is, the longest path is updated. However, in both cases the algorithm needs to backtrack.

Backtracking is performed by removing the last node from the path, hence decreasing the path length by one. During the process of backtracking, the in-arcs to a node being removed from the path must be returned to the current set of arcs. This process will enable the algorithm to add this node when constructing a new path. At the same time, whenever a node is removed from the path, the arc that was used in order to reach that node is marked in the dynamic graph. This process will eliminate the possibility that the algorithm repeats a path that was already traversed. Therefore, by dynamically eliminating and returning arcs from/to the graph, a cyclic digraph can be treated as if it does not contain cycles. The process of reconnecting nodes, i.e., arcs being returned to the dynamic graph, requires that the original graph be maintained. A more detailed description of this search algorithm can be found in Ref. 45.

In some mixed-level modeling cases, more realistic restriction on the longest-path than that it must not include any node more than once is that it not include any transition (arc) more than once. A longest-path with no repeated arcs may include a node multiple times as long as it is reached via different arcs. In the case of more than one transition that meets the condition on the output combination, a search for the longest-path should check all paths between the initial state and all of these transitions. However, such a path should include any of these transitions only once, and it should be the last one in the path.

Performing a search with the restriction that no arc is contained in the path more than once requires maintaining information on arcs being added or removed from the path. Maintaining this information during the search makes the algorithm and its implementation much more complicated relative to the search for the longest path with no repeated nodes. Therefore, a novel approach to this problem was developed. The approach used is to map the problem to the problem of searching for the longest path with no repeated nodes. This mapping can be accomplished by transforming the digraph to a new digraph, to be referred to as the transformed-digraph (or Tdigraph). Given a digraph, $G(V, E)$, which consists of a set of nodes $V = (v_1, v_2, \ldots, v_k)$ and a set of arcs $E = (e_1, e_2, \ldots, e_l)$ the transformation τ maps $G(V, E)$ into a Tdigraph $TG(N, A)$, where N is its set of nodes and A is its set of arcs. The transformation is defined as $\tau(G(V, E)) = TG(N, A)$ and contains the following steps:

Step 1: $\forall\ (e_i \in E)$ generate a node $n_i \in N$
Step 2: $\forall\ (v \in V)$ and $\forall\ (e_p \in d_{in}{}^v, e_q \in d_{out}{}^v)$ generate an arc $a \in A$ such that A: $n_p \rightarrow n_q$.

The first step is used to create a node in the Tdigraph for each arc in the original digraph. This one-to-one mapping defines the set of nodes in the Tdigraph to be $N = \{n_1, n_2, \ldots, n_l\}$ which has the same number of elements found in the set E.

The second step creates the set of arcs, $A = \{a_1, a_2, \ldots, a_u\}$, in the Tdigraph. For each node in the original digraph and for each combination of in-arcs and out-arcs to/from this node, an arc in the Tdigraph is created. For example, given a node v with one in-arc e_i and one out-arc e_j, e_i is mapped to a node n_i, e_j is mapped to a node n_j, and an arc from n_i to n_j is created. In Step 2 of the transformation

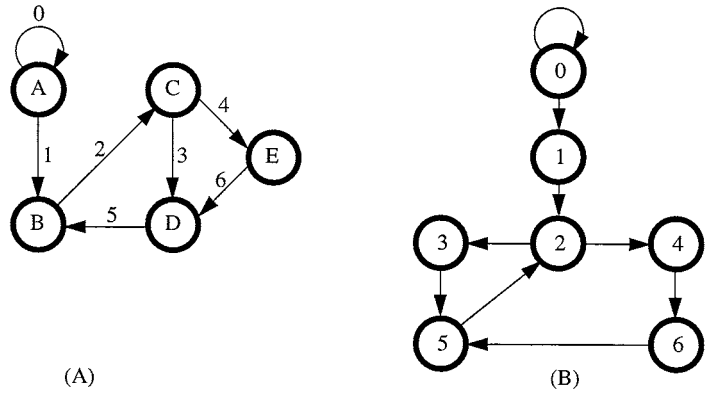


FIGURE 76.31 Transformation of a digraph.

process, it is guaranteed that all possible connections in the original digraph are preserved as transitions between nodes in the Tdigraph. As a result of this transformation, the restriction on not visiting an arc more than once in the original digraph is equivalent to not visiting a node more than once in the Tdigraph. Therefore, by using this transformation, the problem of searching for the longest path with no repeated arcs in the original digraph is mapped to a search for the longest path with no repeated nodes in the Tdigraph. The algorithm described above can then be used to search the Tdigraph.

This transformation is best illustrated by a simple example. Consider the digraph shown in Fig. 76.31(A). The arcs in this digraph are labeled by numbers 0 to 6. The first step of the transformation is to create a node for each arc in the original digraph. Therefore, there will be seven nodes, labeled 0 to 6, in the Tdigraph as shown in Fig. 76.31(B). The next step is to create the arcs in the Tdigraph. As an illustration of this step, consider node C in the original digraph. Arc "2" is an in-arc to node C, while arcs "3" and "4" are out-arcs from node C. Applying Step 2 results in an arc from node "2" to node "3" and a second arc from node "2" to node "4" in the Tdigraph. Considering node B in the original digraph, the Tdigraph will include an arc from node "1" to node "2" and an arc from node "5" to node "2". This process is repeated for all the nodes in the original digraph until the Tdigraph, as shown in Fig. 76.31(B), is formed. A search algorithm can now be executed to find the longest path with no repeated nodes.

A mixed-level modeling methodology, which is composed of all the methods described previously, has been integrated into the ADEPT environment. The steps for minimizing the unknown inputs can be performed prior to simulating the mixed-level model. On the other hand, the search for longest/shortest possible delay must be performed during the simulation itself. This requirement arises because each token may carry different information, which may alter the known input values and, therefore, alter the search of the STG. The STG traversal process has been integrated into the ADEPT modeling environment using the following steps: (1) when a token arrives to the mixed-level interface, the simulation is halted and the search for minimum/maximum number of transitions is performed; and (2) on completion of the search, the simulation continues while applying the sequence of inputs found in the search operation. The transfer of information between the VHDL simulator and the search program, which is implemented in C, is done by using the simulator's VHDL/C interface. A component called the Stream_Generator has been created that implements the STG search process via this interface.

Since mixed-level models are part of the design process and are constructed by refining a performance model, it is likely that many tokens will carry identical relevant information. This information may be used for selective application of the STG search algorithm, hence increasing the efficiency of the mixed-level model simulation. For example, if several tokens carry exactly the same information (and assuming the same initial state of the FSM), the search is performed only once, and the results can be used for the following identical tokens.

An Example of Mixed-Level Modeling with an FSMD Component

This section presents an example of the construction of a mixed-level model with an FSMD interpreted component. The example is based on the performance model of an execution unit of a particular processor. This execution unit is composed of an Integer Unit (IU), a Floating-Point Unit (FPU), and a Load-Store Unit (LSU). These units operate independently, although they receive instructions from the same queue (buffer of instructions). If the FPU is busy processing one instruction and the following instruction requires the FPU, it is buffered, waiting for the FPU to be free again. Meanwhile, instructions which require the IU can be consumed and processed by IU at an independent rate. Both the FPU and the IU have the capability of buffering only one instruction. Therefore, if two or more consecutive instructions are waiting for the same unit, the other units cannot receive new instructions (since the second instruction is held in the main queue). One practical performance metric that can be obtained from this model is the time required for the execution unit to process a given sequence of instructions.

Because the Floating-Point Unit was identified as the most complex and time critical portion of the design, a behavioral description of a potential implementation of it was developed. At this point, a mixed-level model, in which the behavioral description of the FPU is introduced into the performance model, was constructed using the interface described above. The mixed-level model is shown in Fig. 76.32. The mixed-level interface is constructed around the interpreted block which is the behavioral description of the FPU. This FPU is an FSMD type of element, and the interpreted model consists of a clock cycle accurate VHDL behavioral description of this component. The inputs to the FPU include the operation to be performed (Add, Sub, Comp, Mul, MulAdd, and Div), the precision of the operation (single or double) and some additional control information. The number of clock cycles required to complete any instruction depends on these inputs.

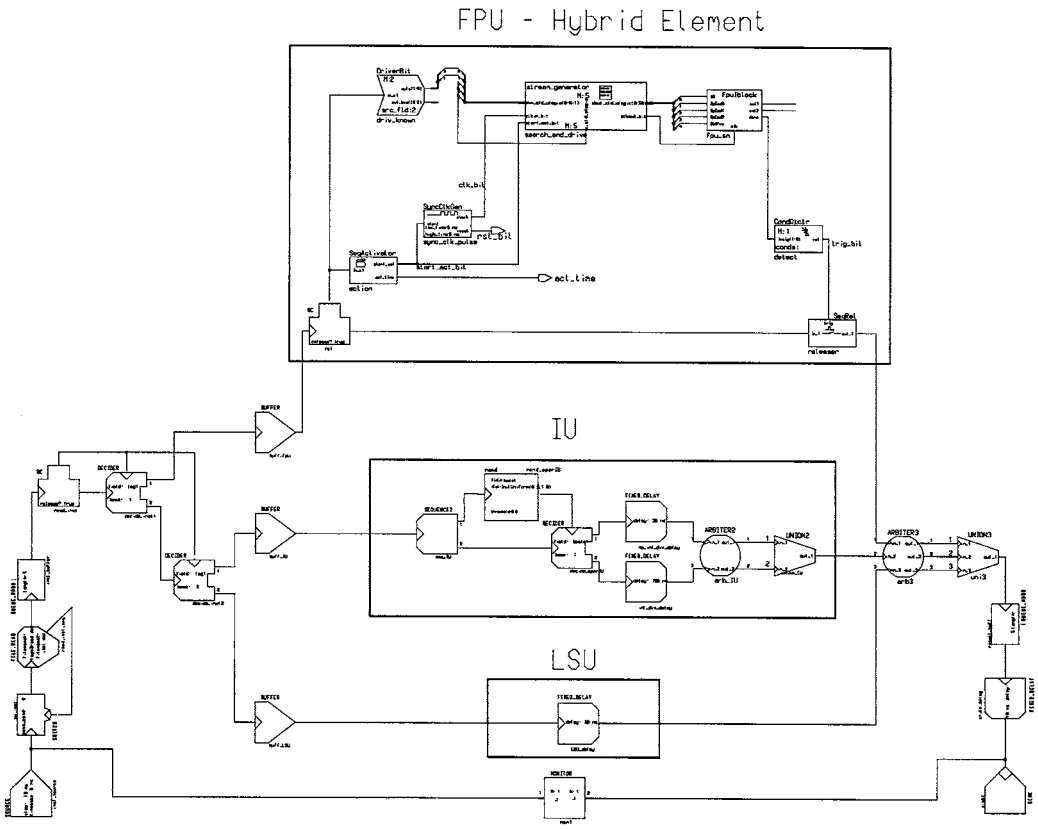


FIGURE 76.32 Mixed-level model with the FPU behavioral description.

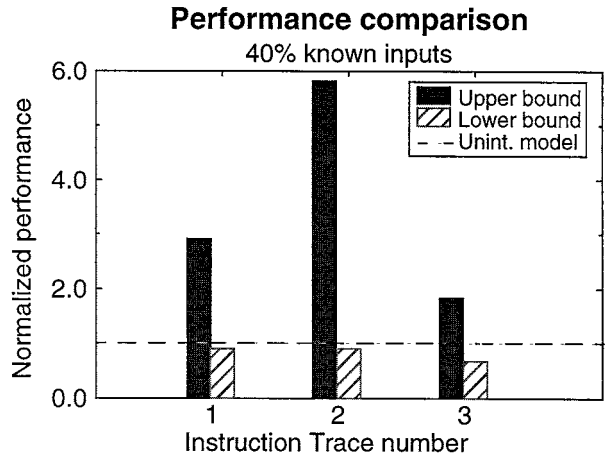


FIGURE 76.33 Performance comparison of the execution unit for three instruction traces.

Figure 76.33 shows the execution unit performance derived from the mixed-level model for three different instruction traces. In this case, only 40% of the inputs have values that are supplied by the abstract performance model; the remainder of the values for the inputs are derived using the techniques described in the section "Finding Values for the "Unknown Inputs" in an FSMD Based Mixed-Level Model." The performance value is normalized by defining unity to be the amount of time required to process a trace according to the initial uninterpreted performance model. In this example, the benefit of the simulation results of the mixed-level model is clear. It provides performance bounds in terms of best- and worst-case delays for the given implementation of the FPU.

An Interface for Mixed-Level Modeling with Complex Sequential Components

A methodology and components for constructing a mixed-level interface involving general sequential interpreted components that can be described as FSMDs was detailed in the previous section. However, many useful mixed-level models can be constructed that include sequential interpreted components that are too complex to be represented as FSMDs, such as microprocessors, complex coprocessors, network interfaces, etc. In these cases, a more "programmable" mixed-level interface that is able to deal with the additional complexity in the timing abstraction problem was needed. This section describes the "watch-and-react" interface that was created to be a generalized, flexible interface between these complex sequential interpreted components and performance models.

The two main elements in the watch-and-react interface are the trigger and the driver. Figure 76.34 illustrates how the trigger and driver are used in a mixed-level interface. Both elements have ports that can connect to signals in the interpreted components of a model. Collectively, these ports are referred to as the probe. The primary job of the trigger is to detect events on the signals attached to its probe, while the primary job of the driver is to force values onto the signals attached to its probe. The driver decodes information carried in tokens to determine what values to force onto signals in the interpreted model (the U/I interface), and the trigger encodes information about events in the interpreted model onto tokens in the performance model (the I/U interface).

The trigger and driver were designed to be as generic as possible. A command language was designed that specifies how the trigger and driver should behave to allow users to easily customize the behavior of the trigger and driver elements without having to modify their VHDL implementation. This command language is interpreted by the trigger and driver during simulation. Because the language is interpreted, changes can be made to the trigger and driver programs without having to recompile the model, thus minimizing the time required to make changes to the mixed-level model.

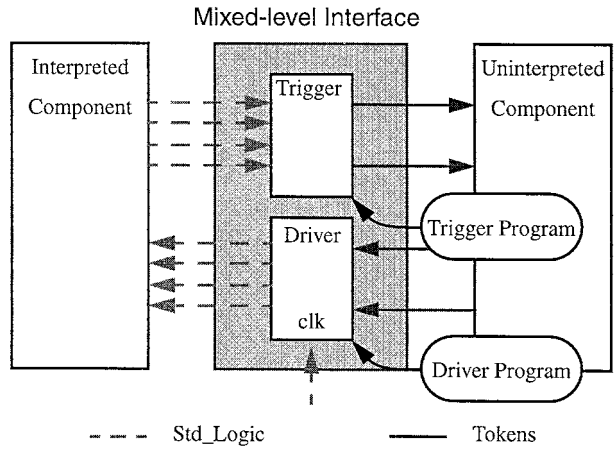


FIGURE 76.34 The watch-and-react mixed-level interface.

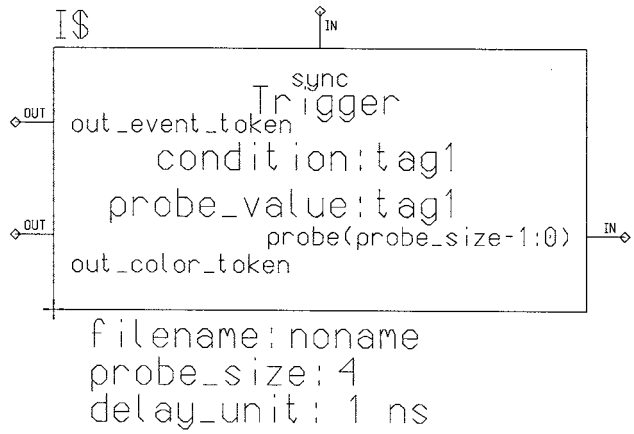


FIGURE 76.35 Schematic symbol for the trigger.

The schematic symbol for the trigger is shown in Fig. 76.35. The primary job of the trigger is to detect events in the interpreted model. The probe on the trigger is a bus of std_logic signals `probe_size` bits wide, where `probe_size` is a generic on the symbol. There is one token output called `out_event_token` and one token output called `out_color_token`. Tokens generated by the trigger when events are detected are placed on the `out_event_token` port. The condition number (an integer) of the event that caused the token to be generated is placed on the `condition` tag field, which is specified as a generic on the symbol. Also, each time a signal changes on the probe, the color of the token on the `out_color_token` port changes appropriately regardless of whether an event was detected or not. The probe value is placed on the `probe_value` tag field of the `out_color_token` port. The `sync` port is used to synchronize the actions of the trigger element with the driver element as explained below.

The name of the file containing the trigger's program is specified by the `filename` generic on the symbol. The `delay_unit` generic on the symbol is a multiple that is used to resolve the actual length of an arbitrary number of delay units specified by some of the interface language statements.

The schematic symbol for the driver element is shown in Fig. 76.36. The primary job of the driver is to create events in the interpreted model by driving values on its probe. These values come from either the command program, or from the values of tag fields on the input tokens to the driver. The probe on the driver is a bus of std_logic signals `probe_size` bits wide, where the `probe_size` is a generic on

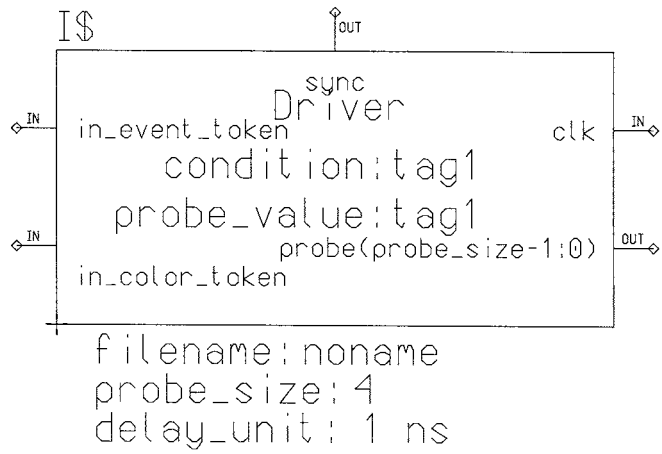


FIGURE 76.36 Schematic symbol for the driver.

the symbol. There is one token input called `in_event_token`, one token input called `in_color_token`, and a special input for a std_logic type clock signal called `clk`. The `clk` input allows driver to synchronize its actions with an external interpreted clock source. The `sync` port is used to synchronize the actions of the driver element with the trigger element. The `filename` and `delay_unit` generic on the symbol function the same as for the trigger.

As discussed previously, a command language was developed to allow the user to program the actions of the trigger and driver. This command language is read by the trigger and driver at the beginning of the simulation. Constructs are available within the command language to allow waiting on events of various signals and driving values or series of values on various signals, either asynchronously or synchronously, with a specified clock signal. In addition, several looping and go-to constructs are available to implement complex behaviors more easily. A summary of the syntax of the command language constructs is shown in Table 76.4.

Example of Mixed-Level Modeling with a Complex Sequential Element

This section presents an example which demonstrates how the trigger and driver elements can be used to interface an interpreted model of a complex sequential component with an uninterpreted model. In this example, the interpreted model is a microprocessor-based controller and the uninterpreted model is that of a motor control system that includes a motor controller and a motor. The motor controller periodically asserts the microcontroller's interrupt line. The microcontroller reacts by reading the motor's current speed from a sensor register on the motor controller, calculating the new control information, and writing the control information to the motor controller.

The microcontroller system consists of interpreted models of eight-bit, RISC-like microprocessors; RAM; memory controller; I/O controller; and clock. The memory controller handles read and write requests issued by the processor to the RAM, while the I/O controller handles read and write request issued by the processor to an I/O device. In the system model, the I/O device is the uninterpreted model of a motor controller. A schematic of the model is shown in Fig. 76.37.

Three triggers and two drivers are used to construct the mixed-level interface for the control system model. One of the triggers is used to detect when the I/O controller is doing a read or write. The other two triggers are used to collect auxiliary information about the operation, such as the address on the address bus and data on the data bus. One of the drivers is used to create a microcontroller interrupt, and the other driver is used to force data onto the data bus when the processor reads from the speed sensor register on the motor controller.

The interrupt driver's program is listed in Fig. 76.38. The program begins by forcing the interrupt line to 'Z' and then waiting for a token from the uninterpreted model of the motor controller to arrive. Once

TABLE 76.4 Trigger and Driver Command Programming Language Constructs

Command	Element	Meaning
--<comment>	both	comment — no action
alert_user	both	print message in simulation window
delay_for <T>	both	delay for <T> time units where the time unit is specified as a generic on the module's symbol
end	both	end the command program
for <N> <loop body> **next**	both	iterate N times over the sequence of statement in the loop body
goto <L>	both	go to line <L> in the command program
output_sync	both	generate a token on the sync output of the module
wait_on_sync	both	wait for an occurrence of a token on the sync port of the module
case_probe_is **when** <STD_LOGIC_VAL> <sequence of statements> **when**… **end_case**	trigger	conditionally execute some sequence of statements depending on the STD_LOGIC value of the probe signal
output <INTEGER_VAL> **after** <T>	trigger	generate a token on the trigger's output with the value of <INTEGER_VAL> on the tag field specified by the generic on the symbol after <T> time units
trigger	trigger	must appear as the first statement in the trigger's command program
wait_on <STD_LOGIC_VAL>	trigger	wait until the probe signal takes on the specified STD_LOGIC value
wait_on_probe	trigger	wait until there is ANY event on the probe signal
case_token_is **when** <INTEGER_VAL> <sequence of statements> **when**… **end_case**	driver	conditionally execute some sequence of statements depending on the integer value of the input token's tag field specified by the generic on the symbol
driver	driver	must appear as the first statement in the driver's command program
dynamic_output_after <T>	driver	force the value from the specified tag field of the input token onto the probe after <T> time units
output <STD_LOGIC_VAL> **after** <T>	driver	force the specified STD_LOGIC value onto the probe signal after <T> time units
wait_on <INTEGER_VAL>	driver	wait until a token with the given integer value on the tag field specified by the generic on the symbol arrives on the in_event_token signal
wait_on_fclk	driver	wait until the falling edge of the clock occurs
wait_on_rclk	driver	wait until the rising edge of the clock occurs
wait_on_token	driver	wait until a token arrives on the in_event_token signal

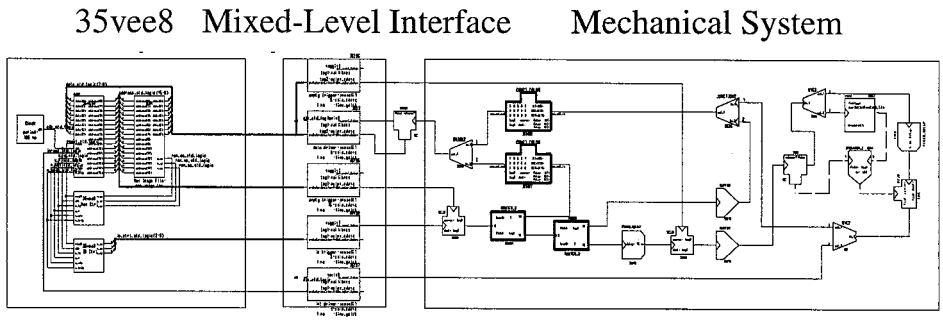


FIGURE 76.37 Schematic of control system model.

a token arrives, the program forces the interrupt line high for ten clock cycles. This condition is accomplished by using a `for-next` statement with a `wait_on_rclk` as the loop body. After ten clock cycles, the program jumps to line 10 where the cycle begins again.

The data driver's program is listed in Fig. 76.39. The program begins by also waiting for a token from the uninterpreted model of the motor controller to arrive. If the value on the condition tag field of the token is 1, then "ZZZZZZZZ" is forced onto the data bus. If the condition tag field value is 3, then the value on the `probe_value` tag field of the `in_color_token` input is forced on the data bus. This process is repeated for every token that arrives.

The I/O trigger's program is listed in Fig. 76.40. This trigger waits until there is a change on the probe. Once there is a change, the program checks to see if the I/O device is being unselected, written to, or read from. If one of the `when` statements matches the probe value, then its corresponding `output` statement is executed. An output of 1 corresponds to the I/O device not being selected. An output of 2 corresponds to the processor writing control information to the motor controller. An output of 3 corresponds to the processor reading the motor's speed from the sensor register on the motor controller.

```
driver
10 output Z after 0
20 wait_on_token
30 output 1 after 0
40 for 10
50    wait_on_rclk
60 next
70 goto 10
80 end
```

FIGURE 76.38 The interrupt driver's program for the control system.

Figure 76.41 shows the results from the mixed-level model as a plot of the sensor output and the processor's control response. Some random error was introduced to the sensor's output to reflect variations in the motor's load as well as sensor noise. The target speed for the system was 63 ticks per sample time (a measure of the motor's RPM). The system oscillates slightly around the target value because of the randomness introduced into the system.

```
driver
10 wait_on_token
20 case_token_is
30     when 1
40        -- sensor not selected
50        output ZZZZZZZZ after 0
60     when 3
70        -- sensor selected for reading
80        dynamic_output_after 0
90 end_case
100 goto 10
110 end
```

FIGURE 76.39 The data driver's program for the control system.

```
trigger
10 wait_on_probe
20 case_probe_is
30     when 111
40        -- sensor not selected
50        output 1 after 0
60     when 001
70        -- sensor selected for writing
80        output 2 after 0
90     when 010
100       -- sensor selected for reading
110       output 3 after 0
120 end_case
130 goto 10
140 end
```

FIGURE 76.40 The I/O trigger's program for the control system.

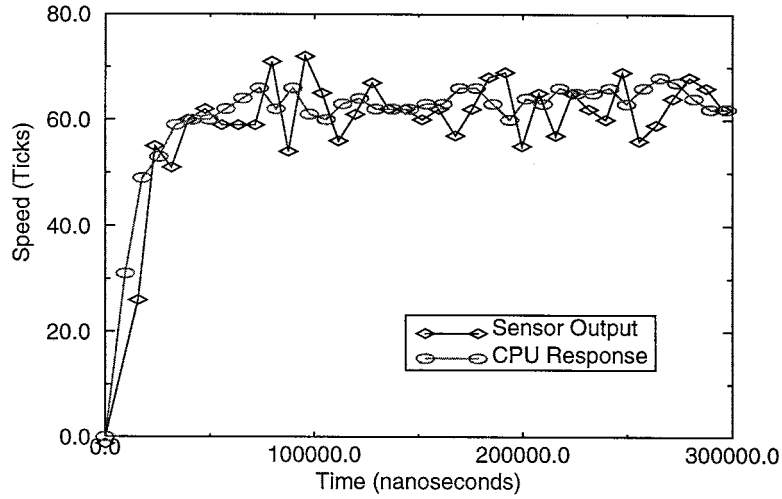


FIGURE 76.41 Sensor and processor outputs for the control system.

76.5 Conclusions

Integration of performance modeling into the design process such that it can actually drive the refinement of the design into an implementation has clear advantages in terms of design time and quality. The ability to cosimulate detailed behavioral models and abstract system-level models is vital to the development of a design environment that fully integrates performance modeling into the design process. One methodology and implementation for cosimulating behavioral models of individual components with an abstract performance model of the entire system was presented. This environment results in models that can provide estimates of the performance bounds of a system that converge as the refinement of the overall model increases.

This chapter has only scratched the surface on the possible improvements that performance or system-level modeling can have on the rapid design of complex VLSI systems. As more and more functionality can be incorporated into the embedded VLSI systems and these systems find their way into safety-critical applications, measures of dependability such as reliability and safety at the system-level are becoming of great interest. Tools and techniques are being developed that can use the performance model from which to derive the desired dependability measures. In addition, behavioral fault simulation and testability analysis are finding their way into the early phases of the design process. In summary, the more attributes of the final implementation that can be determine from the early and often incomplete model, the better the resulting design and the shorter the design cycle.

References

1. ASIC & EDA, Jan. 1993
2. Franke, D. W. and Purvis, M. K. Hardware/software codesign: a perspective, *Proc. 13th Int. Conf. on Software Eng.,* 344, 1991.
3. Franke, D. W. and Purvis, M. K. Design automation technology for codesign: status and directions, *Int. Symp. Circuits Syst.,* 2669, 1992.
4. Frank, G. A., Franke, D. L., and Ingogly, W. F., An architecture design and assessment system, *VLSI Design*, 30, 1985.
5. Terry, C., Concurrent hardware and software design benefits embedded systems, *EDN*, 148, 1990.
6. Napper, S., Embedded-system design plays catch-up, *IEEE Comput.,* 118, 1998.
7. Zurcher, F. W. and Randell, B., Iterative multi-level modeling — a methodology for computer system design, *Proc. IFIP Congr. '68*, 867, 1968.

8. Martin Marietta Laboratories, *RASSP First Annual Interim Technical Report* (CDRL A002), Moorestown, October 31, 1994.
9. Wilsey, P. A. and Dasgupta, S., A formal model of computer architectures for digital system design environments, *IEEE Trans. Comput.-Aided Design,* 9(5), 473, 1990.
10. Giumale, C. A. and Kahn, H. J., Information models of VHDL, *Proc. 32nd Design Automation Conf.*, San Francisco, CA, 678, 1995.
11. Kollaritsch, P., Lusky, S., Matzke, D., Smith, D., and Stanford, P., A unified design representation can work, *Proc. 26th Design Automation Conf.*, 811, 1989.
12. Peterson, J., Petri Nets, *Comput. Surv.*, 9(3), 223, 1997.
13. Molloy, M. K., Performance analysis using stochastic Petri Nets, *IEEE Trans. Comput.,* C-31(9), 913, 1982.
14. Holliday, M. A. and Vernon, M. K., A generalized timed Petri Net for performance analysis, *IEEE Trans. Software Eng.,* SE-13(12), 1987.
15. Kleinrock, L., *Queuing Systems, Vol. 1: Theory*, John Wiley & Sons, New York, 1975.
16. Graham, G. S., Queuing network models of computer system performance, *Comput. Surv.,* 10(3), 219, 1978.
17. Balbo, G., Bruell, S. C., and Ghanta, S., Combining queuing networks and generalized stochastic Petri Nets for solutions of complex models of system behavior, *IEEE Trans. Comput.,* 1251, 1988.
18. Frank, G., Software/Hardware codesign of real-time systems with ADAS, *Electron. Eng.,* 95, 1990.
19. *SES/Workbench User's Manual, Release 2.0,* Scientific and Engineering Software Inc., Austin, TX, Jan. 1991.
20. Maliniak, L., ESDA boosts productivity for high-level design, *Electron. Design,* 125, 1993.
21. IDAS Integrated Design Automation System, JRS Research Laboratories Inc., Orange, CA, 1988.
22. Application Note: TRANSCEND/VANTAGE Optium Cosimulation, TD Technologies, pp. 1-33.
23. Application Note for TRANSCEND Structural Ethernet Simulation, TD Technologies, Aug., 1993, pp. 1-13.
24. Bargodia, R. and Shen, C., MIDAS: integrated design and simulation of distributed systems, *IEEE Trans. Software Eng.,* 17(10), 1991.
25. Lee, E. et al., Mini Almagest, University of California at Berkeley, Department of Electrical Engineering and Computer Science, Apr., 1994.
26. Lee, E. A. and Messerschmitt, D. G. Synchronous data flow, *Proc. IEEE*, 75(9), 1235, 1987.
27. VHDL Hybrid Models Requirements, Honeywell Technology Center, Version 1.0, Dec. 27, 1994.
28. Honeywell Technology Center, VHDL Performance Modeling Interoperability Guideline, Version Draft, Aug., 1994.
29. Rose, F., Steeves, T., and Carpenter, T., VHDL performance models, *Proc. First Annu. RASSP Conf.,* Arlington, VA, 60, Aug. 1994.
30. IEEE, IEEE Standard VHDL Language Reference Manual, IEEE Std. 1076-1993, New York, June 6, 1994.
31. Voss, A. P., Klenke, R. H., and Aylor, J. H., The analysis of modeling styles for system level VHDL simulations, *Proc. VHDL Int. Users Forum,* Fall 1.7, 1995.
32. Aylor, J. H., Waxman, R., Johnson, B. W., and Williams, R. D., The integration of performance and functional modeling in VHDL, in *Performance and Fault Modeling with VHDL,* Schoen, J. M., Ed., Prentice-Hall, Englewood Cliffs, NJ, 22, 1992.
33. Jensen, K., Colored Petri Nets: a high-level language for system design and analysis, in *High-level Petri Nets: Theory and application,* Jensen, K. and Rozenberg, G., Eds., Springer-Verlag, Berlin, 44, 1991.
34. Hady, F. T., A Methodology for the Uninterpreted Modeling of Digital Systems in VHDL, Master's thesis, Department of Electrical Engineering, University of Virginia, Jan. 1989.
35. Voss, A. P., Analysis and Enhancements of the ADEPT Environment, Master of Science (Electrical Engineering) thesis, University of Virginia, May 1996.

36. Dennis, J. B., Modular, asynchronous control structures for a high performance processor, *ACM Conf. Rec., Proj. MAC*, MA, 55, 1970.
37. Swaminathan, G., Rao, R., Aylor, J., and Johnson, B., Colored Petri Net Descriptions for the UVA Primitive Modules, *CSIS Technical Report # 920922.0,* University of Virginia, Sept. 1992.
38. *ADEPT Library Reference Manual,* CSIS Technical Report No. 960625.0, University of Virginia, June 6, 1996.
39. Meyassed, M., McGraw, R., Aylor, J., Klenke, R., Williams, R., Rose, F., and Shackleton, J., A framework for the development of hybrid models, *Proc. 2nd Annu. RASSP Conf.*, Arlington, VA, July 1995.
40. MacDonald, R. A., Williams. R., and Aylor, J., An approach to unified performance and functional modeling of complex systems, *IASTED Conf. Modeling Simulation,* Apr. 1995.
41. Gajski, D., Dutt, N., Wu, A., and Lin, S., HIGH-LEVEL SYNTHESIS: introduction to chip and system design, Kluwer Academic Publishers, 1992.
42. Dijkstra, E. W. A note on two problems in connection with graphs, *Numerische Math. 1,* 269, 1959.
43. Floyd, R., Algorithm 97 (shortest path), *Commun. ACM* 5(6), 345, 1962.
44. Warshall, S., A theorem on Boolean matrices, *J. ACM,* 9(1), 11, 1962.
45. Meyassed, M., System-Level Design: Hybrid Modeling with Sequential Interpreted Elements, Ph.D. dissertation, Department of Electrical Engineering, University of Virginia, Jan. 1997.

Wolf, W. "Embedded Computing Systems and Hardware/Software Co-Design"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

77

Embedded Computing Systems and Hardware/Software Co-Design

Wayne Wolf
Princeton University

77.1 Introduction

This chapter describes embedded computing systems that make use of microprocessors to implement part of the system's function. It also describes hardware/software co-design, which is the process of designing embedded systems while simultaneously considering the design of its hardware and software elements.

77.2 Uses of Microprocessors

An embedded computing system (or more simply an embedded system) is any system which uses a programmable processor but itself is not a general purpose computer. Thus, a personal computer is not an embedded computing system (though PCs are often used as platforms for building embedded systems), but a telephone or automobile which includes a CPU is an embedded system. Embedded systems may offer some amount of user programmability — 3Com's PalmPilot, for example, allows users to write and download programs even though it is not a general-purpose computer — but embedded systems generally run limited sets of programs. The fact that we know the software that we will run on the hardware allows us to optimize both the software and hardware in ways that are not possible in general-purpose computing systems.

Microprocessors are generally categorized by their word size, since word size is associated both with maximum program size and data resolution. Commercial microprocessors come in many sizes; the term microcontroller is used to denote a microprocessor which comes with some basic on-chip peripheral devices, such as serial input/output (I/O) ports. Four-bit microcontrollers are extremely simple but capable of some basic functions. Eight-bit microcontrollers are workhorse low-end microprocessors. Sixteen- and 32-bit microprocessors provide significantly more functionality. A 16/32-bit microprocessor

may be in the same architectural family as the CPUs used in computer workstations, but microprocessors destined for embedded computing often do not provide memory management hardware. A digital signal processor (DSP) is a microprocessor tuned for signal processing applications. DSPs are often Harvard architectures, meaning that they provide separate data and program memories; Harvard architectures provide higher performance for DSP applications. DSPs may provide integer or floating-point arithmetic.

Microprocessors are used in an incredible variety of products. Furthermore, many products contain multiple microprocessors. Four- and eight-bit microprocessors are often used in appliances: for example, a thermostat may use a microcontroller to provide timed control of room temperature. Automatic cameras often use several eight-bit microprocessors, each responsible for a different aspect of the camera's functionality: exposure, shutter control, etc. High-end microprocessors are used in laser and ink-jet printers to control the rendering of the page. Many printers use two or three microprocessors to handle generation of pixels, control of the print engine, and so forth. Modern automobiles may use close to 100 microprocessors, and even inexpensive automobiles generally contain several. High-end microprocessors are used to control the engine's ignition system — automobiles use sophisticated control algorithms to simultaneously achieve low emissions, high fuel economy, and good performance. Low-end microcontrollers are used in a number of places in the automobile to increase functionality: for example, four-bit microcontrollers are often used to sense whether seat belts are fastened and turn on the seat belt light when necessary.

Microprocessors may replace analog components to provide similar functions, or they may add totally new functionality to a system. They are used in several different ways in embedded systems. One broad application category is signal conditioning, in which the microprocessor or DSP performs some filtering or control function on a digitized input. The conditioned signal may be sent to some other microprocessor for final use. Signal conditioning allows systems to use less-expensive sensors with the application of a relatively inexpensive microprocessor. Beyond signal conditioning, microprocessors may be used for more sophisticated control applications. For example, microprocessors are often used in telephone systems to control signaling functions, such as determining what action to take based on the reception of dial tones, etc. Microprocessors may implement user interfaces; this requires sensing when buttons, knobs, etc. are used, taking appropriate actions, and updating displays. Finally, microprocessors may perform data processing, such as managing the calendar in a personal digital assistant.

There are several reasons why microprocessors make good design components in such a wide variety of application areas. First, digital systems often provide more complex functionality than can be created using analog components. A good example is the user interface of a home audio/video system, which provides more information and is easier use than older, non-microprocessor-controlled systems. Microprocessors also allow related products much more cost-effectively. An entire product family, including models at various price and feature points, can be built around a single microprocessor-based platform. The platform includes both hardware components common to all the family members and software running on the microprocessor to provide functionality. Software elements can easily be turned on or off in various family members. Economies of scale often mean that it is cheaper to put the same hardware in both expensive and cheap models and to turn off features in the inexpensive models rather than to try to optimize the hardware and software configurations of each model separately. Microprocessors also allow design changes to be made much more quickly. Many changes may be possible simply by reprogramming; other features may be made possible by adding memory or other simple hardware changes along with some additional programming. Finally, microprocessors aid in concurrent engineering. After some initial design decisions have been made, hardware and software can be designed in parallel, reducing total design time.

While embedded computing systems traditionally have been fabricated at the board level out of multiple chips, embedded computing systems will play an increasing role in integrated circuit design as well. As VLSI technology moves toward the ability to fabricate chips with billions of transistors, integrated circuits will increasingly incorporate one or several microprocessors executing embedded software. Using microprocessors as components in integrated circuits increases design productivity, since CPUs can be used as large components which implement a significant part of the system functionality. Single-chip

embedded systems can provide much higher performance than board-level equivalents, since chip-to-chip delays are eliminated.

77.3 Embedded System Architectures

Although embedded computing spans a wide range of application areas, from automotive to medical, there are some common principles of design for embedded systems. The application-specific embedded software runs on a hardware platform. An example hardware platform is shown in Fig. 77.1. It contains a microprocessor, memory, and I/O devices. When designing on a general-purpose system such as a PC, the hardware platform would be predetermined, but in hardware/software co-design the software and hardware can be designed together to better meet cost and performance requirements.

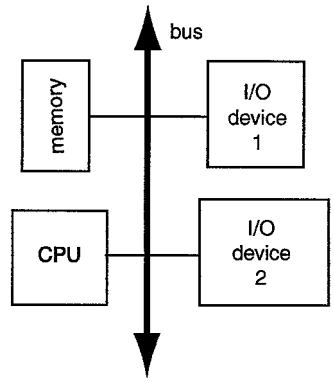


FIGURE 77.1 Hardware structure of a microprocessor system.

Depending on the application, various combinations of criteria may be important goals for the system design. Two typical criteria are speed and manufacturing cost. The speed at which computations are made often contributes to the general usability of the system, just as in general-purpose computing. However, performance is also often associated with the satisfaction of deadlines — times at which computations must be completed to ensure the proper operation of the system. If failure to meet a deadline causes a major error, it is termed a hard deadline. And missed deadlines, which result in tolerable but unsatisfactory degradations are called soft deadlines. Hard deadlines are often (though not always) associated with safety-critical systems. Designing for deadlines is one of the most challenging tasks in embedded system design. Manufacturing cost is often an important criteria for embedded systems. Although the hardware components ultimately determine manufacturing cost, software plays an important role as well. First, the size of the program determines the amount of memory required, and memory is often a significant component of the total component cost. Furthermore, the improper design of software can cause one to require higher-performance, more-expensive hardware components than are really necessary. Efficient utilization of hardware resources requires careful software design. Power consumption is becoming an increasingly important design metric. Power is certainly important in battery-operated devices, but it can be important in wall socket-powered systems as well — lower power consumption means smaller, less-expensive power supplies and cooling and may result in environmental ratings that are advantageous in the marketplace. Once again, power consumption is ultimately determined by the hardware, but software plays a significant role in power characteristics. For example, more efficient use of on-chip caches can reduce the need for off-chip memory access, which consumes much more power than on-chip cache references

Figure 77.1 shows the hardware architecture of a basic microprocessor system. The system includes the CPU, memory, and some I/O devices, all connected by a bus. This system may consist of multiple chips for high-end microprocessors or a single-chip microcontroller. Typical I/O devices include analog/digital (ADC) and digital/analog (DAC) converters, serial and parallel communication devices, network and bus interfaces, buttons and switches, and various types of display devices. This configuration is a complete, basic, embedded computing hardware platform on which application software can execute.

The embedded application software includes components for managing I/O devices and for performing the core computational tasks. The basic software techniques for communicating with I/O devices are polling and interrupt-driven. In a polled system, the program checks each device's status register to determine if it is ready to perform I/O. Polling allows the CPU to determine the order in which I/O operations are completed, which may be important for ensuring that certain device requests are satisfied at the proper rate. However, polling also means that a device may not be serviced in time if the CPU's program does not check it frequently enough. Interrupt-driven I/O allows a device to change the flow

of control on the CPU and call a device driver to handle the pending I/O operation. An interrupt system may provide both prioritized interrupts to allow some devices to take precedence over others and vectored interrupts to allow devices to specify which driver should handle their request.

Device drivers, whether polled or interrupt-driven, will typically perform basic device-specific functions and hand-off data to the core routines for processing. Those routines may perform relatively simple tasks, such as transducing data from one device to another, or may perform more sophisticated algorithms such as control. Those core routines often will initiate output operations based on their computations on the input operations.

Input and output may occur either periodically or aperiodically. Sampled data is a common example of periodic I/O, while user interfaces provide a common source of aperiodic I/O events. The nature of the I/ O transactions affects both the device drivers and the core computational code. Code which operates on periodic data is generally driven by a timer which initiates the code at the start of the period. Periodic operations are often characterized by their periods and the deadline for each period. Aperiodic I/O may be detected either by an interrupt or by polling the devices. Aperiodic operations may have deadlines, which are generally measured from the initiating I/O event. Periodic operations can often be thought of as being executed within an infinite loop. Aperiodic operations tend to use more event-driven code, in which various sections of the program are exercised by different aperiodic events, since there is often more than one aperiodic event which can occur.

Embedded computing systems exhibit a great deal of parallelism which can be used to speed up computation. As a result, they often use multiple microprocessors which communicate with each other to perform the required function. In addition to microprocessors, application-specific ICs (ASICs) may be added to accelerate certain critical functions. CPUs and ASICs in general are called processing elements (PEs). An example multiprocessor system built from several PEs along with I/O devices and memory is shown in Fig. 77.2.

The choice of several small microprocessors or ASICs rather than one large CPU is primarily determined by cost. Microprocessor cost is a nonlinear function of performance, even within a microprocessor family. Vendors generally supply several versions of a microprocessor which run at different clock rates; chips which run at varying speeds are a natural consequence of the variations in the VLSI manufacturing process. The slowest microprocessors are significantly less expensive than the fastest ones, and the cost increment is larger at the high end of the speed range than at the low end. As a result, it is often cheaper to use several smaller microprocessors to implement a function.

When several microprocessors work together in a system, they may communicate with each other in several different ways. If slow data rates are sufficient, serial data links are commonly used for their low hardware cost. The I^2C bus is a well-known example of a serial bus used to build multi-microprocessor embedded systems; the CAN bus is widely used in automobiles. High-speed serial links can achieve

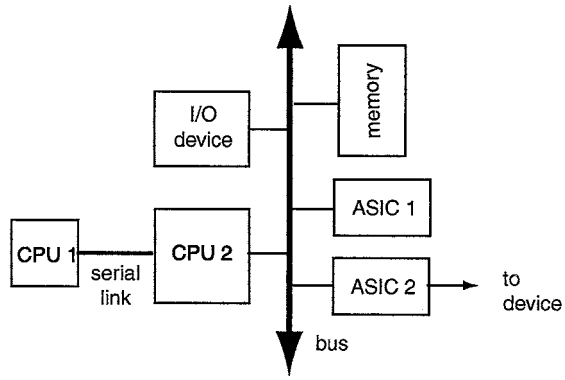


FIGURE 77.2 A heterogeneous embedded multiprocessor.

moderately high performance and are often used to link multiple DSPs in high-speed signal processing systems. Parallel data links provide the highest performance thanks to their sheer data width. High-speed busses such as PCI can be used to link several processors.

The software for an embedded multiprocessing system is often built around processes. A process, as in a general-purpose computing system, is an instantiation of a program with its own state. Since problems complex enough to require multiprocessors often run sophisticated algorithms and I/O systems, dividing the system into processes helps manage design complexity. A real-time operating system (RTOS) is an operating system specifically designed for embedded, and specifically real-time applications. The RTOS manages the processes and device drivers in the system, determining when each executes on the CPU. This function is termed scheduling. The partitioning of the software between application code which executes core algorithms and an RTOS which schedules the times to which those core algorithms are executed is a fundamental design principle in computing systems in general and is especially important for real-time operation.

There are a number of techniques which can be used to schedule processes in an embedded system — that is, to determine which process runs next on a particular CPU. Most RTOSs use process priorities in some form to determine the schedule. A process may be in any one of three states: currently executing (there can obviously be only one executing process on each CPU); ready to execute; or waiting. A process may not be able to execute until, for example, its data has arrived. Once its data arrives, it moves from waiting to ready. The scheduler chooses among the ready processes to determine which process runs next. In general, the RTOS's scheduler chooses the highest-priority ready process to run next; variations between scheduling methods depend in large part on the ways in which priorities are determined. Unlike general-purpose operating systems, RTOSs generally allow a process to run until it is preempted by a higher-priority process. General-purpose operating systems often perform time-slicing operations to maintain fair access of all the users on the system, but time-slicing does not allow the control required for meeting deadlines.

A fundamental result in real-time scheduling is known as rate-monotonic scheduling. This technique schedules a set of processes which run independently on a single CPU. Each process has its own period, with the deadline happening at the end of each period. There can be arbitrary relationships between the periods of the processes. It is assumed that data does not in general arrive at the beginning of the period, so there are no assumptions about when a process goes from waiting to ready within a period. This scheduling policy uses static priorities — the priorities for the processes are assigned before execution begins and do not change. It can be shown that the optimal priority assignment is based on period — the shorter the period, the higher the priority. This priority assignment ensures that all processes will meet their deadlines on every period. It can also be shown that at most, 69% of the CPU is used by this scheduling policy. The remaining cycles are spent waiting for activities to happen — since data arrival times are not known, it is not possible to utilize 100% of the CPU cycles.

Another well-known, real-time scheduling technique is earliest deadline first (EDF). This is a dynamic priority scheme — process priorities change during execution. EDF sets priorities based on the impending deadlines, with the process whose deadline is closest in the future having the highest priority. Clearly, the rate of change of process priorities depends on the periods and deadlines. EDF can be shown to be able to utilize 100% of the CPU, but it does not guarantee that all deadlines will be met. Since priorities are dynamic, it is not possible in general to analyze whether the system will be overloaded at some point.

Processes may be specified with data dependencies, as shown in Fig. 77.3, to create a task graph. An arc in the data dependency graph specifies that one process feeds data to another. The sink process cannot become ready until all the source processes have delivered their data. Processes which have no data dependency path between them are in separate tasks. Each task can run at its own rate. Data dependencies allow schedulers to make more efficient use of CPU resources. Since the source and sink processes of a data dependency cannot execute simultaneously, we can use that information to eliminate some combinations of processes which may want to run at the same time. Narrowing the scope of process conflicts allows us to more accurately predict how the CPU will be used.

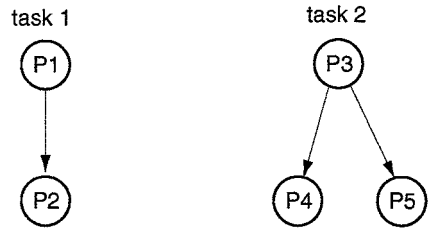


FIGURE 77.3 A task graph with two tasks and data dependencies between processes.

A real-time operating system is often designed to have a small memory footprint, since embedded systems are more cost-sensitive than general-purpose computers. RTOSs are also designed to be more responsive in two different ways. First, they allow greater control over the order of execution of processes, which is critical for ensuring that deadlines are met. Second, they are designed to have lower context-switching overhead, since that overhead eats into the time available for meeting deadlines. The kernel of an RTOS is the basic set of functions that is always resident in memory. A basic RTOS may have an extremely small kernel of only a few hundred instructions. Such microkernels often provide only basic context-switching and scheduling facilities. More complex RTOSs may provide high-end operating system functions such as file systems and network support; many high-end RTOSs are POSIX (a Unix standard) compliant. While running such a high-end operating system requires more hardware resources, the extra features are useful in a number of situations. For example, a controller for a machine on a manufacturing line may use a network interface to talk to other machines on the factory floor or the factory coordination unit; it may also use the file system to access a database for the manufacturing process.

77.4 Hardware/Software Co-Design

Hardware/software co-design refers to any methodology which takes into account both hardware and software during the design of an embedded computing system. When the hardware and software are designed together, the designer has more opportunities to optimize the system by making tradeoffs between the hardware and software components. Good system designers intuitively perform co-design, but co-design methods are increasingly being embodied in computer-aided design (CAD) tools. We will discuss several aspects of co-design and co-design tools, including models of the design, co-simulation, performance analysis, and various methods for architectural co-synthesis. We will conclude with a look at design methodologies that make use of these phases of co-design.

Models

In designing embedded computing systems, we make use of several different types of models at different points in the design process. We need to model basic functionality. We must also capture nonfunctional requirements: speed, weight, power consumption, manufacturing cost, etc.

In the earliest stages of design, the task graph is an important modeling tool. The task graph does not capture all aspects of functionality, but it does describe the various rates at which computations must be performed and the expected degrees of parallelism available. This level of detail is often enough to make some important architectural decisions. A useful adjunct to the task graph are the technology description tables, which describe how processes can be implemented on the available components. One of the technology description tables describes basic properties of the processing elements, such as cost and basic power dissipation. A separate table describes how the processes may be implemented on the components, giving execution time (and perhaps other function-specific parameters like precise power consumption) on a processing element of that type. The technology description is more complex when ASICs can be used as processing elements, since many different ASICs at differing price/performance points can be designed for a given functionality, but the basic data still applies.

A more detailed description is given by either high-level language code (C, etc.) for software or hardware description language code (VHDL, Verilog, etc.) for software components. These should not be viewed as specifications — they are, in fact, quite detailed implementations. However, they do provide a level of abstraction above assembly language and gates and so can be valuable for analyzing performance, size, etc. The control-data flow graph (CDFG) is a typical representation of a high-level language: a flowchart-like structure describes the program's control, while data flow graphs describe the behavior within expressions and basic blocks.

Co-simulation

Simulation is an important tool for design verification. The simulation of a complete embedded system entails modeling both the underlying hardware platform and the software executing on the CPUs. Some of the hardware must be simulated at a very fine level of detail — for example, buses and I/O devices may require gate-level simulation. On the other hand, the software can and should be executed at a higher level of abstraction. While it would be possible to simulate software execution by running a gate-level simulation of the CPU and modeling the program as residing in the memory of the simulated CPU, this would be unacceptably slow.

We can gain significant performance advantages by running different parts of the simulation at different levels of detail: elements of the hardware can be simulated in great detail, while software execution can be modeled much more directly. Basic functionality aspects of a high-level language program can be simulated by compiling the software on the computer on which the simulation executes, allowing those parts of the program to run at the native computer speed. Aspects of the program which deal with the hardware platform must interface to the section of the simulator which deals with the hardware. Those sections of the program are replaced by stubs which interface to the simulator. This style of simulation is a multi-rate simulation system, since the hardware and software simulation sections run at different rates: a single instruction in the software simulation will correspond to several clock cycles in the hardware simulation. The main jobs of the simulator are to keep the various sections of the simulation synchronized and to manage communication between the hardware and software components of the simulation.

Performance Analysis

Since performance is an important design goal in most embedded systems, both for overall throughput and for meeting deadlines, the analysis of the system to determine its speed of operation is an important element of any co-design methodology. System performance — the time it takes to execute a particular aspect of the system's functionality — clearly depends both on the software being executed and the underlying hardware platform. While simulation is an important tool for performance analysis, it is not sufficient, since simulation does not determine the worst-case delays. Since the execution times of most programs are data-dependent, it is necessary to give the simulation of the program the proper set of inputs to observe worst-case delay. The number of possible input combinations makes it unlikely that one will find those worst-case inputs without the sort of analysis that is at the heart of performance analysis.

In general, performance analysis must be done at several different levels of abstraction. Given a single program, one can place an upper bound on the worst-case execution time of the program. However, since many embedded systems consist of multiple processes and device drivers, it is necessary to analyze how these programs interact with each other, a phase which makes use of the results of single-program performance analysis.

Determining the worst-case execution time of a single program can be broken into two subproblems: determining the longest execution path through the program and determining the execution time of that program. Since there is at least a rough correlation between the number of operations and the actual execution time, we can determine the longest execution path without detailed knowledge of the instructions being executed — the longest path depends primarily on the structure of conditionals and loops. One way to find the longest path through the program is to model the program as a control-flow graph and use network flow algorithms to solve the resulting system.

Once the longest path has been found, we need to look at the instructions executed along that path to determine the actual execution time. A simple model of the processor would assume that each instruction has a fixed execution time, independent of other factors such as the data values being operated on, surrounding instructions, or the path of execution. In fact, such simple models do not give adequate results for modern high-speed microprocessors. One problem is that in pipelined processors, the execution time of an instruction may depend on the sequence of instructions executed before it. An even greater cause of performance variations is caching, since the same instruction sequence can have variable execution times, depending on whether the code is in the cache. Since cache miss penalties are often 5X or 10X, the cost of mischaracterizing cache performance is significant. Assuming that the cache is never present gives a conservative estimate of worst-case execution time, but one that is so over-conservative that it distorts the entire design. Since the performance penalty for ignoring the cache is so large, it results in using a much faster, more expensive processor than is really necessary. The effects of caching can be taken into account during the path analysis of the program — path analysis can determine bound how often an instruction present in the cache.

There are two major effects which must be taken into account when analyzing multiple-process systems. The first is the effect of scheduling multiple processes and device drivers on a single CPU. This analysis is performed by a scheduling algorithm, which determines bounds on when programs can execute. Ratemonotonic analysis is the simplest form of scheduling analysis — the utilization factor given by ratemonotonic analysis tells one an upper limit on the amount of active CPU time. However, if data dependencies between processes are known, or some knowledge of the arrival times of data is known, then a more accurate performance estimate can be computed. If the system includes multiple processing elements, more sophisticated scheduling algorithms must be used, since the data arrival time for a process on one processing element may be determined by the time at which that datum is computed on another processing element.

The second effect which must be taken into account is interactions between processes in the cache. When several programs on a CPU share a cache, or when several processing elements share a second-level cache, the cache state depends on the behavior of all the programs. For example, when one process is suspended by the operating system and another process starts running, that process may knock the first program out of the cache. When the first process resumes execution, it will initially run more slowly, an effect which cannot be taken into account by analyzing the programs independently. This analysis clearly depends in part on the system schedule, since the interactions between processes depends on the order in which the processes execute. But the system scheduling analysis must also keep track of the cache state — which parts of which programs are in the cache at the start of execution of each process. Good accuracy can be obtained with a simple model which assumes that a program is either in the cache or out of it, without considering individual instructions; higher accuracy comes from breaking a process into several sub-processes for analysis, each of which can have its own cache state.

Hardware/Software Co-Synthesis

Hardware/software co-synthesis tries to simultaneously design the hardware and software for an embedded computing system, given design requirements such as performance as well as a description of the functionality. Co-synthesis generally concentrates on architectural design rather than detailed component design — it concentrates on determining such major factors as the number and types of processing elements required and the ways in which software processes interact.

The most basic style of co-synthesis is known as hardware/software partitioning. As shown in Fig. 77.4, this algorithm maps the given functionality onto a template architecture consisting of a CPU and one or more ASICs communicating via the microprocessor bus. The functionality is usually specified as a single program. The partitioning algorithm breaks that program into pieces and allocates pieces either to the CPU or ASICs for execution. Hardware/software partitioning assumes that total system performance is dominated by a relatively small part of the application, so that implementing a small fraction of the application in the ASIC leads to large performance gains. Less performance-critical sections of the application are relegated to the CPU.

The first problem to be solved is how to break the application program into pieces; common techniques include determining where I/O operations occur and concentrating on the basic blocks of inner loops. Once the application code is partitioned, various allocations of those components must be evaluated. Given an allocation of program components to the CPU or ASICs, performance analysis techniques can be used to determine the total system performance; performance analysis should take into account the time required to transfer necessary data into the ASIC and to extract the results of the computation from the ASIC. Since the total number of allocations is large, heuristics must be used to search the design space. In addition, the cost of the implementation must be determined. Since the CPU's cost is known in advance, that cost is determined by the ASIC cost, which varies as to the amount of hardware required to implement the desired function. High-level synthesis can be used to estimate both the performance and hardware cost of an ASIC which will be synthesized from a portion of the application program.

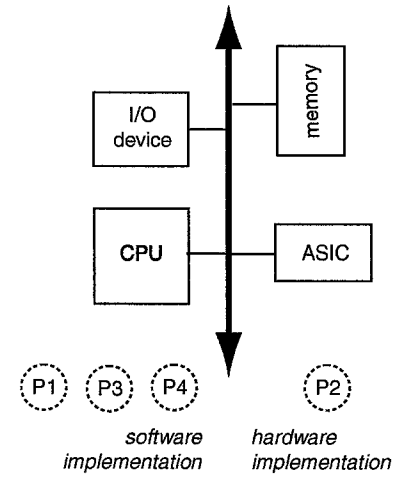


FIGURE 77.4 Hardware/software partitioning.

Basic, co-synthesis heuristics start from extreme initial solutions: We can either put all program components into the CPU, creating an implementation which is minimal cost but probably does not meet performance requirements, or put all program elements in the ASIC, which gives a maximal-performance, maximal-expense implementation. Given this initial solution, heuristics select which program component to move to the other side of the partition to either reduce hardware cost or increase performance, as desired. More sophisticated heuristics try to construct a solution by estimating how critical a component will be to overall system performance and choosing a CPU or ASIC implementation accordingly. Iterative improvement strategies may move components across the partition boundary to improve the design.

However, many embedded systems do not strictly follow the one CPU, one bus, n ASIC architectural template. These more general architectures are known as distributed embedded systems. Techniques for designing distributed embedded systems rest on the foundations of hardware/software partitioning, but they are generally more complicated, since there are more free variables. For example, since the number and types of CPUs is not known in advance, the co-synthesis algorithm must select them. If the number of busses or other communication links is not known in advance, those must be selected as well. Unfortunately, these decisions are all closely related. For example, the number of CPUs and ASICs required depends on the system schedule. The system schedule, in turn, depends on the execution time of each of the components on the available hardware elements. But those execution times depend on the processing elements available, which is what we are trying to determine in the first place. Co-synthesis algorithms generally try to fix several designs and vary only one or a few, then check the results of a design decision on the other parameters. For example, the algorithm may fix the hardware architecture and try to move processes to other processing elements to make more efficient use of the available hardware. Given that new configuration of processes, it may then try to reduce the cost of the hardware by eliminating unused processing elements or replacing a faster, more expensive processing element with a slower, cheaper one.

Since the memory hierarchy is a significant contributor to overall system performance, the design of the caching system is an important aspect of distributed system co-synthesis. In a board-level system with existing microprocessors, the sizes of second-level caches is under designer control, even if the first-level cache is incorporated on the microprocessor and therefore fixed in size. In a single-chip embedded system, the designer has control over the sizes of all the caches. Co-synthesis can determine hardware elements such as the placement of caches in the hardware architecture and the size of each cache. It can

also determine software attributes such as the placement of each program in the cache. The placement of a program in the cache is determined by the addresses used by the program — by relocating the program, the cache behavior of the program can be changed. Memory system design requires calculating the cache state when constructing the system schedule and using the cache state as one of the factors to determine how to modify the design.

Design Methodologies

A co-design methodology tries to take into account aspects of hardware and software during all phases of design. At some point in the design process, the hardware and software components are well-specified and can be designed relatively independently. But it is important to consider the characteristics of both the hardware and software components early in design. It is also important to properly test the system once the hardware and software components are assembled into a complete system.

Co-synthesis can be used as a design planning tool, even if it is not used to generate a complete system architectural design. Because co-synthesis can evaluate a large number of designs very quickly, it can determine the feasibility of a proposed system much faster than a human designer. This allows the designer to experiment with what-if scenarios, such as adding new features or speculating on the effects of lower component costs in the future. Many co-synthesis algorithms can be applied without having a complete program to use as a specification. If the system can be specified to the level of processes with some estimate of the computation time required for each process, then useful information about architectural feasibility can be generated by co-synthesis.

Co-simulation plays a major role once subsystem designs are available. It does not have to wait until all components are complete, since stubs may be created to provide minimal functionality for incomplete components. The ability to simulate the software before completing the hardware is a major boon to software development and can substantially reduce development time.

Cottrell, D. "Design Automation Technology Roadmap"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

78

Design Automation Technology Roadmap

Donald Cottrell
Silicon Integration Initiative, Inc.

78.1 Introduction

No invention in the modern age has been as pervasive as the semiconductor, and nothing has been more important to its technological advancement than Electronic Design Automation (EDA). EDA was born in the 1960s both for the electronic computer and because of it. It was the advent of the computer that allowed for the development of specialized programs that perform the complex management, design, and analysis operations associated with semiconductors and electronic systems. At the same time, it was the design, management, and manufacture of the thousands (now tens of millions) of devices that make up a single electronic assembly that made EDA an absolute requirement to fuel the semiconductor progression. Today, EDA programs are used on electronic packages for all business markets from computers to games, telephones to aerospace guidance systems, toasters to automobiles. Across these markets, EDA supports many different package types, such as integrated circuit (IC) chips, multi-chip modules (MCM), printed circuit boards (PCB), and entire electronic assemblies of several different packages.

- No electronic circuit package is as challenging to those EDA as the integrated circuit. The growth in complexity in ICs has placed tremendous demands on EDA. Mainstream EDA applications such as simulation, layout, and test generation have had to improve their speed and capacity characteristics with this ever-increasing growth in the number of circuits to be processed. New types of design and analysis applications, new methodologies, and new design rules have been necessary to keep pace. Yet, even with the technological breakthroughs that have been made in EDA across the past three decades, it is still having difficulty keeping up with the breakthroughs being made in the electronic technologies that it supports. The increase in the chip's die size, coupled with the decrease in the size of features on the chip, is causing the number of design elements per IC to increase at a tremendous rate. The decrease in feature size and spacing coupled with the increase in operating frequency is causing additional levels of complexity to be approximated in the models used by design and analysis programs.

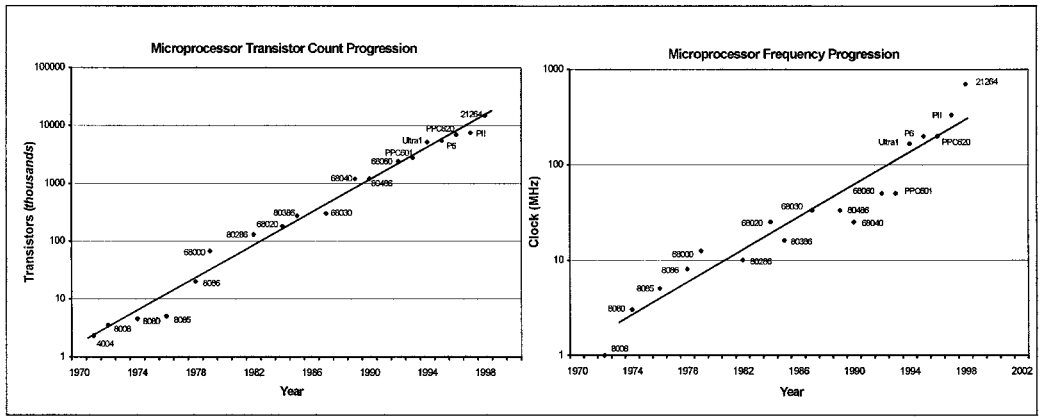


FIGURE 78.1 Microprocessor development.

In the period from 1970 to the present, the following semiconductor advances occurred:

- IC integration has grown from tens of transistors on a chip to over ten million.
- The feature size on ICs has shrunk from 10 microns to 0.18 microns.
- On-chip clock frequency has increased from a few MHz to over 600 MHz.
- Die sizes have increased from less than 20 mm^2 to over 400 mm^2.

Playing an essential part in the advancement of EDA have been advances in computer architectures that run the EDA applications. These advances have included the following:

- Computer CPU speed: from less than a million instructions per second (MIPS) of shared mainframe to hundreds of MIPS on a dedicated workstation.
- Computer memory: from fewer than 32 kilobytes to over 500 gigabytes.
- Data archive: from voluminous reels of (rather) slow speed tape to virtually limitless amounts of high-speed electronic storage devices.

Nevertheless, these major improvements in computing power alone would not have been sufficient to meet the EDA needs of the even more significant advances in semiconductors. Major advances also had to be made in fundamental algorithms used by EDA programs, and entirely new design techniques and design paradigms had to be invented and established for the technology advances to be supportable. This chapter will trace the more notable advancements made in EDA across the past three decades. From there, it will go on to discuss the technology trends predicted across the next decade along with the impact they will have on EDA of the future. It is important to understand these trends and projections, because if the EDA systems cannot keep pace with the semiconductor projections, then these projections will never be realized. Although it may be possible to build manufacturing processes that can produce ultra-deep submicron and find the billions of dollars of capital required, without the necessary EDA support these factories will never be fully utilized. SEMATECH reports that chip design productivity has increased at a compounded rate of 21%, while Moore's Law predicts the number of transistors on a chip to increase at a compound rate of 56%. This means that at some time building manufacturing plants that can produce smaller, denser chips will reach a point of diminishing returns, because the ability to design chips with that many transistors will not be possible.

78.2 Design Automation — An Historical Perspective

The 1960s — The Beginnings of Design Automation

Early entries into design automation were made in the areas of design records, PCB wiring, and manufacturing test generation. A commercial EDA industry did not exist, and developments were made within

companies with the need, such as IBM[1] and Bell Labs, on mainframe computers such as the IBM 7090. The 7090 had addressable 36-bit words and a limit of 32,000 words of main storage (magnetic cores). By today's standard, this would be equivalent to the storage in a small electronic address book, far less than the typical 32 megabytes of RAM on the average notebook PC and certainly no match for a high-function RISC workstation with 512 megabytes of main store. Computer limitations continue to be the curse of EDA development, particularly when the EDA support is targeted for design of the next generation computer. However, the limitations of the computers of the 1960s was particularly acute.

In retrospect, the limit of computers in the 1960s was a blessing for the development of design automation. Because of these limitations, design automation developers were forced to invent highly creative algorithms that operated on very compact data structures. Many of the algorithms developed during this decade are still in use within commercial EDA systems today, with only minor differences in fundamental concepts. Notable advances during this period were

- The fundamental "stuck-at" model for manufacturing test and a formal algebra for the generation of tests and diagnosis of faults
- Parallel fault simulation, which provided simulation of many fault conditions in parallel with the good-machine (nonfaulty circuit) to reduce fault-simulation run-times
- A three-valued algebra for simulation which yields accurate results using simple delay models, even in the presence of race conditions within the design
- Development of fundamental algorithms for the placement and wiring of components.

Moreover, there was development of fundamental heuristics for placement and wire routing, and for divide and conquer concepts. One such concept was the hierarchical division of a wiring image into cells, globally routing between cells and then performing detailed routing within cells, possibly subdividing them further. Many of these fundamental concepts are still applied today, although the complexities of physical design of today's LSI is vastly more complex.

The 1960s represented the awakening of design automation and provided the proof of its value and need for electronic design. It would not be until the end of this decade when the explosion of the number of circuits designed on a chip would occur and the term LSI (large-scale integration) would by coined. EDA development in the 1960s was primarily focused on printed circuit assemblies, but the fundamental concepts developed for design entry, test generation, and physical design provided the basics for EDA in the LSI era.

Design Entry

Before the use of computers in electronic design, the design schematic was a drawing. This drawing was a draftsman's rendering of the notes and sketches provided by the designer. The drawings provided the basis for manufacturing or repair operations in the field. As automation developed, it became desirable to store these drawings on media usable by computers so that the creation of input to the automated processes could itself be automated. So, the need to record the design of electronic products and assemblies in computers was recognized in the late 1950s and early 1960s. In the early days of design automation, the electronic designer would develop the design using paper and pencil and then transcribe it to a form suitable for keyed entry to a computer. Once keyed into the computer, the design could be rendered in a number of different formats to support the manufacturing and field support processes. It recognized further that these computerized representations of the circuit design drawing could also drive design processes, such as the routing of printed circuit traces or the generation of manufacturing test patterns. From there, it was but a short step to the use of computers to generate data in the form required to drive automated manufacturing and test equipment.

Early design entry methods involved the keying of the design into transcription records that were read into the computer and saved on a persistent storage device. This became known as the design's database, and it is the start of the design automation system. From the database, schematic diagrams and logic diagrams were rendered for use in engineering, manufacturing, and field support. This was typically a two-step process, whereby the designer drew the schematic by hand and then submitted it to another

for conversion to the transcription records, keypunch, and submission to the computer. Once in the computer, the formal automated drawings were generated, printed, and returned to the designer. Although this process seems archaic by today's standards, it did result in a permanent record of the design in computer readable format. This could be used for many forms of records management, engineering change history, and as input to design, analysis, and manufacturing automation that would soon follow.

With the introduction of the alphanumeric terminal, the keypunch was replaced as the window into the computer. With this, new design description languages were developed and the role of the transcription operator dissolved. Although these description languages still represented the design at the device or gate level, they were free-format and keyword-oriented and engineers were willing to use them. The design engineer now had the tools to enter design descriptions directly into the computer, thus eliminating the inherent inefficiencies of the "middle-man." Thus, a paradigm shift began to evolve in the method by which design was entered to the EDA system. Introduction of the direct access storage devices (disks) in the late 1960s also improved the entry process as well as the entire design system by providing on-line, high-speed direct access to the entire design or any portion of it. This was also important to the acceptance of design entry by the designer, as the task was still viewed as a necessary overhead rather than a natural part of the design task. It was necessary to get access to the other evolving design automation tools, but typically, the real design thought process took place with pencil and paper techniques. Therefore, any changes that made the entry process faster and easier were eagerly accepted.

With the evolution of design tools, new design description languages began to be introduced that were capable of representing the functional design intent, but without the implementation details. The design engineer was now able to represent his design ideas at a more abstract level with less detail. Register Transfer Level (RTL) design languages were introduced, and with new simulators capable of simulation directly from these, the designer was able to verify design intent much earlier in the design cycle.

The next shift occurred in the later part of the 1970s with the introduction of graphics terminals. With these, the designer could enter a design into the database in schematic form. This form of design entry was a novelty, but not a clear performance improvement. In fact, until the introduction of the workstation and dedicated graphics support, graphic entry was detrimental to design productivity in many cases. Negative effects such as less-than-effective transaction speed and time lost in making the schematic aesthetically pleasing, and the low level of detail all added to less-than-obvious advances. On the other hand, use of the graphics display to view the design and make design change proved extremely effective and was a great improvement over the red-lined prints, and for this reason alone, graphics represent a major advance. However, even with the negative virtues of graphic schematic entry, this style took off with the introduction of the workstation in the 1980s. In fact, the graphic editor glitz and capability often was a major decision point in the purchase of one commercial EDA system over another, and to be considered a commercially viable system, graphics entry was required. Nevertheless, as the density of ICs grew and grew, graphics entry of schematics would begin to yield to the advantages of alphanumeric entry languages. As EDA design and analysis tool technology advanced, entry of design at the RTL level would become commonplace. Today, design entry using RTL descriptions is the generally accepted approach for original design of standard cell design, although schematic entry is the accepted method for PCB design and many elements of custom ICs.

There is no doubt that the introduction of graphics into the design automation system represents a major advance and a major paradigm shift. Use of graphics to visualize design details, wiring congestion, timing diagrams, etc. is of major importance. Use of the graphics to perform certain edit functions is standard operating procedure. Large system design often entails control circuitry, dataflow, and functional modules. Classically, these systems span across several chips and boards and employ several styles of entry for the different physical packages. These may include

- Schematics — graphic
- RTL and behavorial level languages — alphanumeric
- Timing diagrams — graphic

- State diagrams — alphanumeric
- Flowcharts — graphic

Today, these entry techniques can be found in different EDA tools, and each is particularly effective for different types of design problems. Schematics are effective for the design of "glue" logic that interconnects functional design elements, such as modules on a PCB. Behavorial languages are useful for all design, but particularly effective for dataflow behavior. Timing diagrams lend themselves well to describe the functional operations of "black-box" components at their I/Os without needing to describe their internal circuitry. State diagrams are a convenient way to express the logical operation of combinational circuits. Flowcharts are effective for describing the operations of control logic, much like use of flowcharts for specification of software program flow. With technology advances, the IC is engulfing more and more of the entire system, and all of these forms of design are prevalent on a single chip. It is even expected that the design of "black-box" functions will be available from multiple sources to be embedded onto the chip similar to the use of modules on a PCB. Thus, it is likely that future EDA systems will support a mixture of design description forms to allow the designer to represent sections of the design in a manner most effective to each. After all, design is described in many forms by the designer outside the design system.

Test Generation

Testing of manufactured electronic subassemblies entails the use of special test hardware that can provide stimulus (test signals) to selected [input] pins of the part under test and measure for specified responses on selected [output] pins. If the measured response matches the specified response, then the part under test passed that test successfully. If some other response is measured, then the part failed that test and the presence of a defect is indicated. Manufacturing testing is the successive application of test patterns that causes some measurable point on the part under test to be sensitized to the presence of a manufacturing defect. That is, some measurable point on the part under test will result in a value if the fault is present that is different from what it would be if the fault were not present. The collection of test patterns cause all (or almost all) possible manufacturing failures to render a different output response than would the non-defective part. For static dc testers, each stimulus is applied, and after the part under test settles to steady state, the specified outputs are measured and compared with the expected results for a nondefective part.

To bound the test generation problem, a model was developed to represent possible defects at the abstract gate level. This model characterizes the effects of defects as a stuck-at value. This model is fundamental to most of the development in test generation and is still in use today. It characterizes defects as causing either a stuck-at-one or a stuck-at-zero condition at pins on a gate. Additionally, it assumes that a fault is persistent and that only one fault occurs at a time. Thus, this model became known as the single stuck-at fault model. Stuck-at fault testing assumes that the symptom of any manufacturing defect can be characterized by the presence of a stuck-at fault some place within the circuit. By testing for the presence of all possible stuck-at faults that can occur, all possible manufacturing defects can thus be tested.

The stuck-fault models for NAND and NOR gates are shown in Fig. 78.2. For the NAND gate, the presence of an input stuck-at-one defect can be sensitized (made detectable) at the gate's output pin by setting the good-machine state for that input to zero and setting the other input values to 1. If a 0-state is observed at the gate's output node, then the fault is detected. A stuck-at-zero condition on any specific input to the NAND gate is not distinguishable from a stuck-at-zero on any other input to the gate, thus is not part of the model. However, a stuck-at-one is modeled at the gate's output to account for such a fault or a defect on the gate's output circuitry.

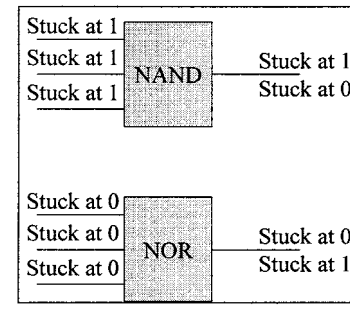


FIGURE 78.2 Stuck fault models.

The fault model for the NOR gate is similar except that here the input condition is stuck-at-zero only, as the stuck-at-one defect cannot be isolated to a particular input.

Later in time, additional development would attack defects not detectable with this stuck-at fault model. For example, bridging faults where nodes are shorted together, and delay faults where the output response does not occur within the required time. The stuck-at fault model cannot detect these fault types, and they became important as the development of CMOS progressed. Significant work was performed in both of these areas beginning in the 1970s, but it did not have the impact on test generation development that the stuck-at fault model did.

The creation of the fault model was very important to test generation, as it established a realistic set of objectives to be met by the test generator that could be achieved in a realistic amount of compute time. A formal algebra was developed by Roth,[2] called the D-ALG, that formalized an approach to test generation and fault diagnosis.

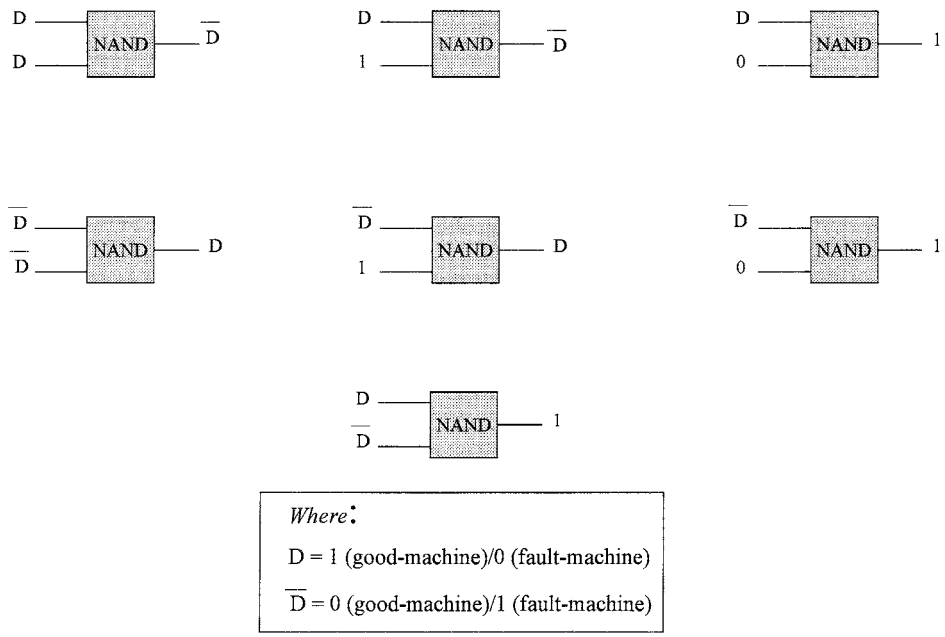


FIGURE 78.3 D-ALG calculus.

The test generation programs could choose a fault based on the instances of gates within the design and the fault models for the gates. It could then trace back from that fault to the input pins of the design and, using the D-ALG calculus, find a set of input states that would sensitize the fault. Then, it could trace forward from that fault to the design's output pins, sensitizing the path (cause the good-machine value to be the opposite from the stuck-at-fault value along that path) to at least one observable pin.

Use of functional patterns as the test patterns in lieu of heuristic test generation was another approach for manufacturing testing. However, this required an extreme number of tests to be applied and worse, depended on the experience and skill of the designer to create quality tests. The use of fault models and automatic test generation produced a minimum set of tests and greatly reduced manually intensive labor.

The exhaustive method to assure coverage of all possibly detectable defects would be to apply all possible input states to the design under test and compare the measured output states with the simulated good-machine states. For a design with n input pins, however, it requires the simulation of 2^n input patterns for combinatorial logic and at least 2^{n+m} for sequential logic (where m is the number of independent storage elements). For even a relatively small number of input pins, this amount of simulation

would not be possible even on today's computers, and the time to apply this number of patterns at the tester would be grossly prohibitive.

The amount of time that a part resides on the tester is critical in the semiconductor business, as it impacts the number of parts that can be produced in a given amount of time and the capital cost for testers. This implies that the number of test patterns that need to be applied at the tester should be keep to a minimum. Early work in test generation attacked this problem in two ways. First, when a test pattern was generated for a specific fault, it was simulated against all possible faults. In many cases, the application of a test pattern that is targeted for a specific fault will also detect several other faults at the same time. The test for the specific fault may be detectable on one output pin, for example, but additional faults may be observable at other output pins. The use of fault simulation detected these cases and provided a mechanism to mark as tested those faults that were "accidentally" covered. This meant that the test-generation algorithm did not have to generate a specific test for those faults. Second, schemes were developed to merge input patterns together into a smaller number of test vectors to minimize the number of patterns that need to be applied at the tester. This is possible when two adjacent test patterns require the application of specific stimulus values at different input pins, each allowing all other input pins to be at the don't-care state. In these cases, the patterns can be merged into a single test vector. With successive analysis in this way, all pairs of test patterns (pattern n with $n + 1$, or the merger of m and $m + 1$ with pattern $m + 2$) are analyzed and merged into a reduced set of patterns.

Sequential design elements severely complicate test generation, as they require the analysis of previous states and the application of sequences of patterns. Early work in test generation broke feedback nets, inserted a lumped delay on them, and analyzed the design using a Huffman model. Later work attempted to identify the memory elements within the design using sophisticated topological analysis and then used a Huffman model to analyze each. State tables for each sequential element were generated and saved for later use as lookup tables in the test generation process. Use of three-value simulation within the analysis both reduced the analysis time and guaranteed that the results were always accurate. Huffman analysis required 2^x simulations (where x is the number of feedback nets) to determine if critical hazards existed in the sequential elements. Using three-valued simulation[3] (all value transitions go through an X state), this was reduced to a maximum of $2x$ simulations.

This lumped delay model did not account for the distribution of delays in the actual design, thus it often caused pessimistic results. Often simulated results yielded don't-know (X-state) conditions when a more accurate model could yield a known state. This made it difficult to generate patterns that would detect all faults in the model. As the level of integration increased, the problems associated with automatic test generation for arbitrary sequential circuits became unwieldy. This necessitated that the designer be called back into the problem of test generation most often to develop tests that would detect those missed by the test generation program. New approaches were developed that ranged from random pattern generation to advanced algorithms and heuristics which used sequences of different approaches. However, by the mid-70s the need to design-for-test was becoming evident to many companies.

During the mid-70s, the use of scan-design was developed.[4] Scan design provides external control and observability to points within a design that might otherwise be inaccessible. This is accomplished inserting special scan registers into the design at the points to be controlled and observed. These registers are connected into a shift register chain that has each of its data input and its data output connected to an accessible pin. Under normal conditions, signals from the design are passed through individual registers. Under test conditions, test vectors can be scanned in from the input pin of the scan register and applied to the appropriate points within the design. Similarly, values on points within the design are captured in individual registers and scanned out to the output pin of the scan register.

The development of scan design was important for two reasons. First, level-sensitive scan design (LSSD) allowed for external control and observability of sequential elements within the design. With specific design-for-test rules, this reduced the problem of test generation by allowing the design to be analyzed as a combinational circuit. In LSSD, rigid design rules such as the following were developed, and checking programs or scan generation algorithms were implemented to assure they were adhered to before entering test generation:

- all internal storage elements implemented in hazard-free polarity-hold latches
- absence of any global feedback loops
- latches may not be controlled by the same clock as latches feeding them
- externally controllable clocks to shift register latches

Second, scan design allowed for external control and observability at otherwise nonprobable points between modules on an MCM or PCB. During the 1980s, an industry standard for this was developed called Boundary Scan (IEEE 1149.1 Joint Test Action Group).[5] Use of a standard technique and shift register latches greatly improved the transfer of complex modules into a random design by simplifying the test problem. Each of these uses of scan techniques to provide additional control and observability have a price. They require additional real estate in the IC design and have a level of performance overhead. However, with the achievable transistor density levels on today's ICs and the test benefits accrued, these penalties are easily justified in all but the most performance critical designs. In fact, with today's IC densities and speeds, the use of circuitry on-chip is often used to generate test vectors.

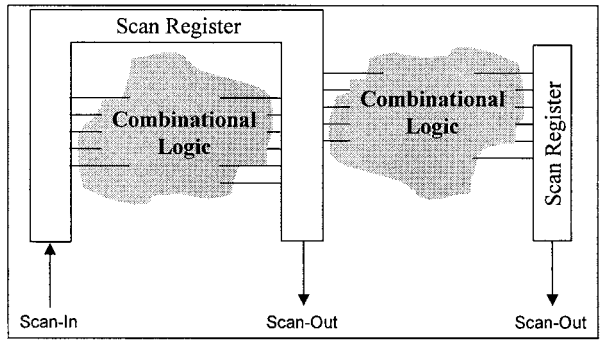


FIGURE 78.4 Scan design.

During the 1980s, IC density allowed for the design of built-in self-test (BIST) circuitry on the chip itself. Operating at hardware speed, it became feasible to generate complete exhaustive tests and large numbers of random tests never before possible with software and stored program testers. BIST tests are generated on the tester by the device under test, reducing the management of data transfer from design to manufacturing. Use of a binary-counter or linear feedback shift-registers (LFSR) generate the patterns for exhaustive or random tests, respectively. In the latter case, a pseudo-random bit sequence is formed by the exclusive-OR of the bits on the LFSR, and this result is then fed back into the LFSR input. Thus, a pseudo-random bit sequence can be generated whose sequence length is based on the number of LFSR stages and the initial LFSR state. The design can be simulated to determine the state conditions for the generated test patterns, and these simulated results compared with the actual device-under-test results are observed at the tester.

Today, BIST techniques are becoming common for the test of on-chip RAM and ROS. BIST is also used for logic sections of chips either with fault-simulated, weighted random test patterns or good machine simulated exhaustive patters. In the latter case, logic is partitioned into electrically isolated regions with a smaller number of inputs in order to reduce the number of test patterns. Partitioning of the design reduces the number of exhaustive tests from 2^n (where n is the total number of inputs) to

$$\sum_{i=1}^{m} 2^{n^i}$$

where m is the number of partitions and n^i ($n^i < n$) is the number of inputs on each logic partition.

Since the use of BIST implies extremely large numbers of tests, the simulated data transfer and test measurement time is reduced greatly by the use of a compressed signature to represent the expected and actual test results. Thus, only a comparison of the simulated signatures for each BIST region needs to be made with the signatures derived by the on-chip hardware, rather than the results of each individual test pattern. This is accomplished by feeding the output bit sequence to a single input serial input LFSR after exclusive-OR with the pseudo-random pattern generated by that LFSR. In this way, a unique pattern can be observed for a sequence of test results, which is a function of the good-machine response and a pseudo-random number.

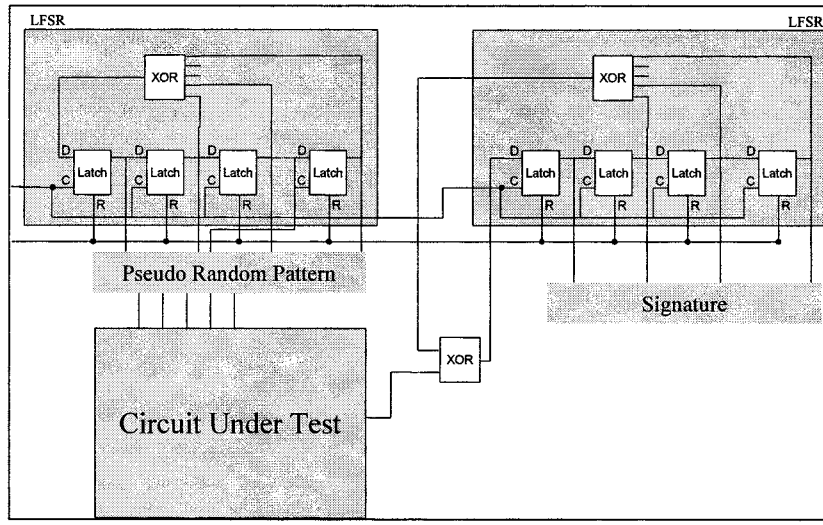


FIGURE 78.5 BIST logic.

Today, testing of ICs typically consists of combinations of different test strategies. These may include stored program stuck fault tests, BIST tests, dc and delay tests (which test for signal arrival times in addition to state), and I_{DDQ} tests (direct drain quiescent current testing checks for CMOS defects that cause current leakage). These latter two test techniques are used to detect defect conditions not identified by stuck-at fault tests such as shorts (bridging faults) between signal nets or gate oxide defects which cause incorrect device operation. The progression of inventions in stuck-fault modeling and scan design, that have taken place over the last 30 years have been key to the success in this field. Moreover, as will be seen later in this chapter, these techniques will continue to be essential ingredients for IC testing in the future.

Fault Simulation

Fault simulation is used to predict the state of a design at observable points in the presence of a defect. This is used in manufacturing testing to detect faulty parts and for field diagnostics of entire assemblies. Early work in fault simulation relied on the stuck-fault model and performed successive simulations of the design with each single fault independent of any other fault; thus a single stuck-at-fault model was assumed. Because even these early designs consisted of thousands of faults, it was too time-consuming to simulate each fault serially, and it was necessary to create high-speed simulation algorithms.

For manufacturing testing, fault simulation took advantage of three-valued zero-delay simulation. The simulation model of the design was levelized and compiled into an executable program. Levelization assured that driver-gates were simulated before the gates receiving the signals, thus allowing a state resolution in a single simulation pass. Feedback loops were cut and the X-transition of three-valued simulation resolved race conditions. The inherent instruction set of the host computer (such as AND, OR, XOR, etc.) allowed the use of a minimum set of instructions to simulate a gate's function.

The parallel-fault simulation algorithm was developed during the 1960s, which allowed many faults to be simulated in parallel. Using the 32-bit word length of the IBM 7090 computer architecture, for example, simulation of 31 faults in a single pass (using the last bit for the good-machine) was possible. For each gate in the design, two host machine words were assigned to represent its good-machine state and the state for 31 single stuck-at faults. A bit position in the first word was set to one or zero to represent the state of the node represented for the good or faulty machine, which that bit position represented. The corresponding bit position in the second word was set to zero if it was a known state and one if it was an X-state. The entire fault list was divided into n partitions of 31 faults, and each partition was then simulated against all input patterns. In this way, the run-time for simulation is a function of

$$\sum_{i=1}^{n=F/31} Patterns$$

where, F is the total number of single stuck-at faults in the design. Specific faults are injected within the word representing their location within the design by the insertion of a mask that is AND'd or OR'd at the appropriate word. For stuck-at-one conditions, the mask contains a 1-bit in the position representing the fault (and 0-bits at others) and it is OR'd to the gate's memory location. For stuck-at-zero faults, the mask contains a 0-bit at the positions representing the fault (and 1-bits at others) and it is AND'd with the gate's memory location.

As the level of integration increased, however, so did the number of faults. The simulation speed improvement realized from parallel-fault simulation is limited to the number of faults simulated in parallel and because of the algorithm overhead, it was typically less than that factor. Also, during the 1970s the challenges to testing became those of the IC, which was major because of the rapidly increasing circuit density. However, the IC also allowed relaxed goals in test.

When manufacturing testing is performed on PCB packages, it is desirable to not only detect the presence of a fault, but also to isolate it to the faulty module, which can then be replaced. With IC's the notion of fixing faulty parts is not realistic. To isolate faults to a particular replaceable unit, it is necessary to simulate all generated tests against all faults exhaustively. When fault detection is the only goal, a fault-machine need only be simulated up to the point where it is detectable at an observable pin and need not be simulated against any other patterns. Parallel-fault simulation did not lend itself to be able to drop out simulation of detected faults because even when individual faults within a word were detected, others were not, thus, simulation of the entire set had to continue. Only when all faults in the word were detected could simulation of that parallel set cease. This, coupled with the limiting factor of a speedup of the number of parallel faults, led to the development of event-based fault simulation and improved algorithms.

Deductive-fault simulation was developed early in the 1970s and required only one simulation pass per test pattern. This is accomplished by simulating only the good-machine behavior and using deductive techniques to determine each fault that is detectable along the simulated paths. Because lists of faults detectable at every point along the simulated path need to be kept, this algorithm requires extensive memory, far more than the parallel algorithm. However, with increasing memory on host computers and the inherent increase in fault simulation speed, this technique won the favor of many fault simulators.

Concurrent-fault simulation refined the deductive algorithm by recognizing the fact that paths in the design quickly become insensitive to the presence of most faults, particularly after some initial set of test patterns is simulated. [An observation made in the 1970s was that a high percentage of faults is detected in a low percentage of the initial test patterns, even if these patterns are randomly generated.] The concurrent-fault simulation algorithm simulates the good-machine and "concurrently" simulates a number of fault-machines. Once it is determined that a particular fault-machine state is the same as the good-machine state, simulation for that fault ceases. Since in logic paths most faults will become insensitive rather close to the point where the fault is located, the amount of simulation for these fault machines was kept small. This algorithm required even more memory, particularly for the early test patterns; however, host machine architectures of the late 1970s were supporting what then appeared as massive amounts of addressable memory.

With the use of scan-design, all sequential elements are controllable from the tester. Therefore, the design can be simulated as a combinational circuit and its state is deterministic based on any single test pattern — and not dependent of previous patterns or states. Parallel-Pattern fault simulation was developed in the late 1970s to take advantage of this and simulate multiple test patterns in parallel against a single fault. A performance advantage is achieved because compiled simulation could again be utilized as opposed to the more general event-based approach. In addition, because faults not detected by the initial test patterns are typically detectable by only a few patterns, simulation of many sets of patterns does not require a complete pass across the design. In most cases, the required sensitized path to detect these faults disappears within a close proximity to the fault. Adjustment of the number of patterns to simulate in parallel, therefore, could effect the required simulation time.

Because of the increasing number of devices on chips, the test generation and fault simulation problem continued to face severe challenges. During the 1980s, BIST began to evolve as a method for manufacturing testing. Use of circuitry on the IC itself to generate test vectors meant that exhaustive tests could be generated and exercised at the tester. With exhaustive tests, and no need to provide fault isolation diagnostics, fault simulation can be eliminated. With BIST approaches to testing, only the good machine behavior needs to be simulated and compared with the actual results at the tester. To decrease the amount of data passed from the simulation to the tester and the time on the tester, special hardware is designed on the chip to capture the results and compress them into a single value (signature). This signature is compared with the simulated signature only at the end of a sequence of tests, thus minimizing the number of comparisons required.

Physical Design

As with test generation, physical design has evolved from early work on PCBs. Physical design (PD) automation programs generate the physical routing (wiring) for the interconnections of the logical nets in a design. To accomplish this, the suite PD programs must be capable of assigning physical package pins (pin assignment) to nets, placing logic functions (placement) on the package, and performing myriad electrical and topographical checks to assure the quality of the solution. The challenge for PD has become ever greater since the early days. It started with PCBs, where the goal was simply to route all the nets, typically with the shortest paths. Any nets that were not auto-routed were routed (embedded) manually, or with discrete wires as a last resort. As the problem moved to ICs, the ability to use discrete wires to finish routing was no more. For nets that could not be auto-routed, manual routing (embedding) was required. Now, wiring congestion in areas on the IC had to be considered by PD programs. Too much congestion in an area could block the passage of more wires through that area and complicate or negate a possible solution, and less than a 100% solution is unacceptable. As the IC densities increased, so did the number of nets. This necessitated the invention of smarter wiring programs and heuristics, for even a small number of incomplete (overflow) routes became too complex of a task for manual solutions.

Later, as IC device sizes shrank and the gate delays decreased, the delay caused by interconnect wiring became an important factor. No longer was any wiring solution that connected the nodes on the nets necessarily a correct solution. Now, the wiring problem was complicated by the need to find wiring solutions that fall within acceptable timing limits. Thus, the wiring lengths and thickness need to be considered. As IC features become packed closer together, cross-coupled capacitance (crosstalk) effects between them is now an important consideration as well. For advanced ICs even today, effects of cross coupling have expanded into a three-dimensional space.

As if this challenge were not enough, the decrease in power supply (V_{dd}) voltage levels with smaller, more closely spaced features has introduced a new problem with which physical design must contend. This is called noise, the unwanted introduction of voltage on a signal. Noise can cause both increases in signal delays and incorrect logical behavior. Noise is a function of both the physical layout of the design and the signal switching characteristics. It results from design factors such as signal termination, simultaneous switching of circuits, and crosstalk between adjacent lines. These conditions result from characteristics such as spacing between signal lines, switching frequency, rise times, cross-sectional area of wires, etc. Design tools for PCBs and multi-chip modules have had to contend with noise for some time.

However, it is now becoming a significant factor in the design of ICs. Future PD solutions must consider these complex factors and still achieve a 100% solution that meets the designer specified timing for IC designs that contain millions of nets.

Because of these increasing demands on PD, major paradigm changes have taken place in design methodology. In the early days, there was a clear separation of logic design and physical design. The logic designer was responsible to create a netlist that correctly represented the logic behavior desired. Timing was a function of the drive capability of the driving circuit and the number of receivers. Different power levels for drivers could be chosen by the logic designer to match the timing requirements, based on the driven circuits. The delay imposed by the time-of-flight along interconnects and due to the parasitics on the interconnect was insignificant. Therefore, the logic designer could hand-off the physical design to another, more adept at using the PD programs and manually embedding overflow wires. As the semiconductor technology progressed, however, there needed to be more interactions between the logic designer and the physical designer, as the interconnect delays became a more dominant factor across signal paths. The logic designer had to give certain timing constraints to the physical designer, and if these could not be met, the design was often passed back to the logic designer. The logic designer, in turn, had to choose different driver gates or a different logical architecture to meet his design specification. In many cases, the pair had to become a team or there was a merger of the two previously distinct operations into one "IC designer."

This same progression of merging logic design and physical design into one operational responsibility has also begun at the EDA system architecture level. In the 1960s and 1970s, front-end (design) programs were separate from back-end (physical) programs. Most often they were developed by different EDA development teams, and designs were transferred between them by means of data files. Beginning in the 1980s and into today, the data transferred between front-end programs and back-end included specific design constraints that must be met by the PD programs — the most common being a specific amount of allowed delay across an interconnect or signal path. As the number of constraints that must be met by the PD programs increases, however, the likelihood that a 100% solution can be found is less and less. This results in additional design cycles between front-end and back-end programs coupled with longer processing times within any cycle. This will result in another necessary paradigm shift in the architecture and design of physical design systems and algorithms, which will be discussed in more detail later in this chapter.

Ignoring the complexities of checking and electrical constraints that modern-day physical design programs have to deal with, however, many of the fundamental wiring heuristics and algorithms spawned from work done in the 1960s for PCBs.

Before global routing is performed, placement of the modules, cells, or gates is performed. Early placement algorithms were developed to minimize the total length of the interconnect wiring using Steiner Trees and Manhattan wiring graphs. In addition, during these early years, algorithms were developed to analyze wiring congestion that would occur as a result of placement choices and minimize it to give routing a chance to succeed. Later work in the 1960s led to algorithms that performed a hierarchical division of the wiring image and performed global wiring between these subdivisions (then called cells) before routing within the cells.[6] This divide and conquer approach simplified the problem and led to quicker and more complete results. Min-cut placement algorithms often used today are a derivative of this divide and conquer approach. The image is divided into partitions and placement of these partitions are swapped to minimize interconnect length and congestion. Then, placement is performed within the partitions using the same objectives. Many current placement algorithms are based on these early techniques although they need to consider more constraints, such as crosstalk, timing requirements, and power dissipation.

Development of efficient maze routing algorithms also began in the 1960s, and many of today's routers use enhanced derivatives of these early techniques. The Lee algorithm[7] finds a solution by emitting a "wave" out from both the source and target points to be wired. This wave is actually a ordered identification of available channel positions — where the available positions adjacent to the source or destination are numbered 1, and the available positions adjacent to them are numbered 2, etc. Successive moves and

sequential identification is made (out in all directions as would a wave) until the source and destination moves meet (the waves collide). Then a backtrace is performed from the intersecting position in reverse sequential order along the numbered track positions back to the source and destination. At points where a choice is available (that is, there are two adjacent points with the same order number), the one which does not require a change in direction is chosen.

Other techniques, such as the Hightower Line-Probe technique,[8] were developed during this period and speeded up maze routing by use of emanating lines rather than waves. This algorithm emanated a line out from both the source and destination points toward each other. When either line encounters an obstacle, then another line is emanated from a point just missing the edge of the obstacle on the original line at a right angle to the original line, and toward the target or source. Thus, the process is much like walking blindly in an orthogonal line toward the target and changing direction only after bumping into a wall. This process continues until the lines intersect at which time the path is complete.

In today's ICs, the challenge of completed wiring that meets all constraints is of crucial importance. Unlike test generation, which can be considered successful when a very high percentage of the faults are detected by the test patterns, 100% of the nets must be wired. Further 100% of the wires must fall within the required electrical and physical design constraints; nothing less than 100% is acceptable. Today these constraints include timing, power consumption, noise, and electromigration, and this list will become more complex as IC feature sizes and spacing are reduced further.

The 1970s — The Awaking of Verification

Before the introduction of LSI components it was common practice to build prototype hardware and then verify the design correctness. PCB packages containing small-scale integrated modules allowed for engineering rework of real hardware within the verification cycle. Prototype PCBs were built, and engineering used drivers and oscilloscopes to determine whether the correct output conditions resulted from input stimuli. As design errors were detected, they were repaired on the PCB prototype, validated, and recorded for later engineering change (EC) into the production version of the design. Use of the wrong logic function within the design could easily be repaired by replacing the low cost component(s) in error with the correct one(s). Incorrect connections could easily be repaired by cutting a printed circuit and replacing it with a discrete wire (as opposed to a printed circuit). Using wire-wrap tools, these discrete wires could easily connect the desired pins of the modules on the PCB. Thus, design verification was a sort of trial and error process and the final rework of a PCB could contain hundreds of these wires.

The introduction of LSI drastically changed the design verification (DV) paradigm. Although use of software simulation to verify design correctness began during the 1960s, it was not until the advent of LSI that this concept became widely accepted. With LSI, it became impossible to accurately model the design with a populated PCB, and it was not possible to rework gates and wires on the chip. Thus, the 1970s are best represented as a quantum leap into software design verification and verification before manufacture. This represented a major paradigm shift in electronic design, and it was a difficult change for some to accept. Design verification on hardware prototypes resulted in a tangible result that could be touched and held. It was a convenient tangible, which could be shown to management to represent real progress. Completed design verification against a software model did not produce the same level of touch and feel. Further, since the use of computer models was a relatively new concept, it met with the distrust of many. However, the introduction of LSI demanded this change and today, software design verification is commonly accepted practice for all levels of electronic components, subassemblies, and systems.

Early DV simulators simulated gate-level models of the design with two-valued simulation. Since gates had nearly equal delays and the interconnect delay was insignificant by comparison, use of a unit delay value of each gate was common. Later simulators exploited the use of three-valued simulation to resolve race conditions and identify oscillations within the design more quickly. With the emergence of LSI, however, these simple models had to become more complex, and simulators had to become more flexible and faster much faster. In the first half of the 1970s, important advances were made to design verification simulators that included

- Use of abstract (with respect to the gate-level) models to improve simulation performance and allow for verification throughout the design cycle (not just at the end)
- More accurate representations of gate and interconnect delays to enhance the simulated accuracy

In the latter half of the decade, significant contributions began as a result of design constraints such as scan design, which reduced the problem to combinational circuits. During this period work began on separating functional characteristics from timing, and more formal approaches verification. However, the fundamental DV tool is simulation, and the challenge to make simulators faster and more flexible continues even today.

Simulation

Although some early DV simulators used compiled models, these quickly gave way to interpretative event-driven approaches. Compiled simulators have the advantage of higher-speed simulation of individual gates because host machine instructions are compiled in-line and are directly executed with minimum simulator overhead. Event-based simulators require more overhead to manage the simulation operations, but they provide a level of flexibility and generality not possible with the compiled model. This flexibility was necessary to provide for simulation of timing characteristics as well as function, and to handle general sequential designs. Therefore, this approach was generally adopted for DV simulators over the compiled approach used by early fault simulators. There are four main concepts to an event-based simulator:

- The Netlist, which provides the list of blocks (gates at first, but any complex function later), connections between blocks, and delay characteristics of the blocks.
- Event time queues, which are lists of events that need to be executed (blocks that need to be simulated) at specific points in [simulation] time. Event queues contain two types of events — update and calculate. Update events change the specified node to the specified value, then schedule calculate-events for the blocks driven from that node. Calculate events call the simulation behavior of the specified block and, on return from the behavior routine, schedule update events to change the states on the output nodes to the new values.
- Block simulation behavior (the instructions that, when passed the block's input states will compute its output states — possibly also scheduling some portion of itself to be simulated at a later time).
- Value list — the current state of each node in the design.

Simulation begins with the stimulus generator, which schedules update events in the time-zero queue. After all update events are stored for all time-zero stimuli, the time-zero event queue is traversed and, one-by-one, each update event in the queue is executed. Update events update the node in the value list and, if the new value is different from the current value, schedule calculate events for blocks driven from the updated node. These calculate events may be placed back in the time-zero queue or in other time queues based on delay specifications (which will be discussed later). After all update events are executed and removed from the queue, the simulator traverses the queue, selects the next calculate event, interprets its function, and passes control to the appropriate block simulation behavior for calculation. Thus, event-based simulation is interpretative as opposed to the compiled simulator, which is ignorant of the function because it is compiled in-line with host instructions.

Execution of a calculate event causes simulation of a block to take place. This is accomplished by passing control to the simulation behavior of the block with pointers to the current state-values on its inputs (in the value list). When complete, the simulation routine of the block will pass control back to the simulator with the new state condition for its output(s). The simulator then schedules the block output(s) value update by placing an update event for it in the appropriate time queue. The decision of which time queue the update events are scheduled within is based on what the delay value is for the block.

Once all calculate events are executed and removed from the queue, the cycle begins again, but first the stimulus generator is again called. The stimulus generator may store new update-events in the next time queue or (if required) insert a new time queue for a time between the present and that of the next existing queue. Then the process of executing update events followed by calculate events for the current time queue repeats. This cycle repeats until there are no more events or until some specified maximum

simulation time. To keep update events separated from calculate events within the linked-list queues, it is common to add update events at the top of the linked-list and calculate events at the bottom, updating the chain-links accordingly.

Because it is not possible to predetermine how many events will reside in a queue at any time, it is common to create these as linked lists of dynamically allocated memory. Additionally, time queues are linked, since the required number of time queues cannot be determined in advance and similar to events. New time queues can be inserted into that chained list as required.

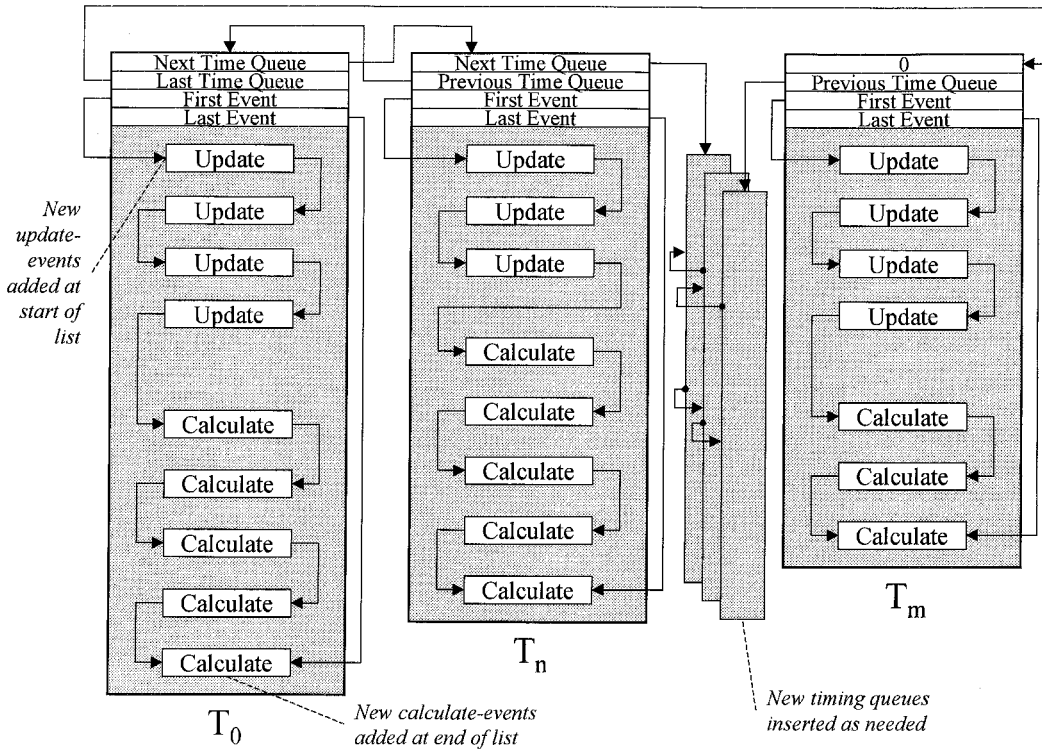


FIGURE 78.6 Event queues.

A number of techniques have been developed to make queue management fast and efficient, as it is the heart of the simulator, and because of its generality, the event-based algorithm was the clear choice for DV. One advantage of event-based simulation over the compiled approach is that it easily supports the simulation of delay. Delays can be assigned to blocks, and they are simulated by choosing the appropriate time queue(s) in which to schedule the update event(s) for the block output(s) change. Delays can be assigned to nets, and they are simulated by choosing the appropriate time queues in which to schedule calculate events for the fan-out nodes. (Note that this allows different delays to be associated with each path in the net as receiver block calculation events can be scheduled in different time queues.) Therefore, even early simulators typically supported a very complete delay model, even before sophisticated delay calculators were available to take advantage of it.

The delay model included

- Intrinsic block delay (T_{block}) — the time required for the block output to change state relative to the time that a controlling input to that block changed state.
- Interconnect delay (T_{int}) — the time required for a specific receiver pin in a net to change state relative to the time that the driver pin changed state.
- Input-output delay (T_{io}) — the time required for a specific block output to change state relative to the time the state changed on a specific input to that block.

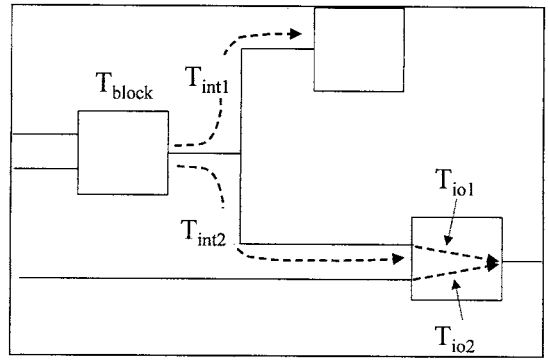


FIGURE 78.7 Simulation delay types.

Treatment of delay has had to evolve and expand since the beginnings of software DV. Unit delay was the first model used — where each gate in the design is assigned one unit of delay and interconnect delays are zero. This was a crude approximation but allowed for high-speed simulations because of the simplifying assumptions and it worked reasonably well. As the integrated circuit evolved, however, the unit delay model was replaced by a lumped delay model in which each gate could be assigned a unique value for delay — actually, a rise-delay and a fall-delay. This was assigned by the technologist based on some average load assumed. At this time, also, the beginnings of development of delay calculators began. These early calculators used simple equations, adding the number of gates driven by the gate being calculated and then adding the additional delay to the intrinsic delay values of that gate. As interconnect wiring became a factor in the timing of the circuit, the pin-to-pin delay came into use. At first, delay calculation for the interconnect contribution to delay was crude, associating a fixed value for pin-to-pin delay for very long interconnects. Today's ICs, however, require delay calculation based on very accurate, distributed RC-models for interconnects, as these delays have become more significant with respect to gate delays. Tomorrow's ICs will require even more precise modeling for delays, as will be discussed later in this chapter, and consider inductance (L) in addition to the RC parasitics. Additionally, transmission line models will be necessary for the analysis of certain critical global interconnects. However, the delay model defined in the early years of DV and the capabilities of the event-based simulation algorithm stand ready to meet this challenge.

Another significant advantage of the event-based simulation algorithm commonly used today evolved from development that began in the early 1970s. This is the mixed-level simulation model. Recall that one of the fundamental components of event simulation is the Block Simulation Behavior. A calculation-event for a block passes control to the sub-routine that simulates the behavior along with the block's input states. For gate-level simulation, these behavior subroutines are quite simple: AND, OR, NOR, etc. However, since the behavior is actually a program sub-routine, it can be arbitrarily complex as well. Realizing this, early work took place to generalize the interface between the simulator control program and the Block Simulation Behavior. A small number of functional-level commands were formalized to access input states, schedule execution at a specific entry to the behavior for a later time, and update the block output states. Thus, a Block Simulation Behavior might look like the following:

```
FUNCTION (CNOR);
/* Complimentary output NOR function with 3 inputs. Input delay  = 2,
   intrinsic delay for true output = 10, intrinsic delay for complement
   output = 12*/
INPUT (a,b,c);/*Declare for inputs a, b, c*/
OUTPUT (x,y);
DECLARE input1, input2, input3;/*Declare storage for inputs*/
DECLARE out, cout;/*Declare storage for outputs*/
GET a,input1, b,input2, c,input3;
```

```
DELAY 2, Entry1;
Entry1: out = input1|input2|input3;
cout = ¬out;
DELAY 10,(x = out);/*Schedule this update-event for 10 time units later */
DELAY 12 (y = cout);/*Schedule this update-event for 12 time units later */
END Entry1;
ENDCNOR;
```

Sophisticated use of these behavorial commands supported the direct representation not only for simple gates, but also for complex functions such as registers, MUXs, RAM, ROS, etc. In doing so, simulation performance was improved because what before was treated as a complex interconnection of gates could now be simulated as a single block.

By generalizing the simulator system, block simulation routines could be loaded at simulation run-time if required (by the use of a block with that function in the netlist) and dynamically linked with the simulator control program. This meant that the block simulation behaviors could be developed by anyone, compiled independently of the simulator, stored in a library of simulation behavioral models, and used as needed in simulation. This is common practice in DV simulators today, but this concept in the early 1970s represented a significant breakthrough and supported a major paradigm shift in design — namely, the concept of top-down-design and verification.

With top-down-design, the designer no longer had to design the gate-level netlist before verification could take place. Now, high-level models could be written to represent the behavior of sections of the design not yet complete, and they could be used in conjunction with gate-level descriptions of the completed parts to perform full system verification. Now, simulation performance could be improved by swapping in the use of gate-level models or behavior models for system elements based on what the verification goals were. Now, concurrent design and verification could take place across design teams. And now there was a formal method to support design reuse of system elements without the need to expose internal design details for reusable elements. In the extreme case, the system designer could write a single functional model for the entire system and verify it with simulation. The design could then be partitioned into sub-system elements, and each could be described with a behavorial model before being handed off for detailed design. During the detailed design phase, individual designers could verify their sub-system in the full system context even before the other sub-systems were completed. This was particularly valuable for generation of the functional patterns to be simulated, as they could now be generated by the simulation of the other system components. Thus, verification of the design no longer had to wait until the end; it could now be a continuous process throughout the design.

Throughout the period, improvements were made to DV simulators to improve performance in the formulation and capability of behavorial description languages. In addition, designers found more and more novel ways to use behavorial models to describe non-digital system elements such as card readers, and analog assemblies. For analog devices, the digital-to-analog interface was modeled on entry to the behavior and the analog-to-digital interface when returning to the simulator, as well as the analog behavior within the modeled element. Punch-card reader behavior could be modeled with software that would access stored memory containing the data on the simulated cards and return the bit patterns back to the simulator — on the proper nodes and at the proper simulation times. Late in the 1970s and across the 1980s, this was formalized into:

- VHDL[9] (VHSIC [Very High speed Integrated Circuit] High-Level Description Language), sponsored by DARPA, and
- Verilog,[10] a commercially developed RTL-level description language.

These two languages are now both industry-accepted standard hardware-description languages. Simulator performance improved by means of creative queue algorithms and the use of compiled representations of standard functions, etc. Additionally, methodologies and tools were developed to compare simulated results of simulations against the high-level models with those of the detail-level designs at

selected outputs, and top-down-design verification using event-based simulation became common practice. This was a natural evolution for large system designers. It was, however, the introduction of synthesis and the commercial ASIC that evolved in the 1980s that made its use the norm today.

With the advent of structured scan design, such as LSSD, the use of compiled simulation returned for verification of the combinational sections of the design between scan latches. A simulation technique, called cycle-simulation, was developed that yielded a major performance advantage over event-based simulation. Cycle-simulation treats the combinational sections of a scan-based design as compiled zero-delay models. The simulator executes each section at each simulated cycle by passing control to the compiled routine for it along with its input states. The resulting state values at the outputs of these sections are assumed to be correctly captured into their respective latch positions. That is, the clock circuitry and path delays are assumed correct and are not simulated during this phase of verification. The (latched) output values are used as the input states to the sections they drive at the next cycle, and the process repeats for the each simulated cycle. Each simulation pass across the compiled models represents one cycle of the design's system clock, starting with an input state and resulting in an output state. To assure only one pass is required, the gates or RTL-level statements for the combinational sections are levelized before compilation into the host-machine instructions. This assures that value updates occur before calculations, and only one pass across a section of the model is required to achieve the correct state response at the outputs.

Simulation performance was greatly improved with cycle-simulation because of the compiled model and because the clock circuitry did not have to be simulated repeatedly with each simulated-machine cycle. Early cycle-simulators were developed on mixed-level event simulators and by unique implementations. For cycle-simulation on top of event simulators, the design model consisted of a netlist containing an interconnected set of the combinational sections. Each of the combinational sections was described in an RTL-level language and treated as a general compiled behavioral model by the simulator. The event simulator operated as normal, except that the control program precluded simulated state changes at the outputs of simulated models were not propagated to the inputs of the driven sections until the start of the next simulated clock cycle. Later implementations used look-up table models[11] in place of compiled in-line host instructions, which provided additional speed improvements, as they did not require the execution of compiled host-instructions for every element within a combinational logic section for each cycle.

Cycle-simulation did not completely replace event simulation, as the clock circuitry needs to be verified. Its use, however, was effective for control logic of synchronous computer designs. Cycle-simulation is less useful of dataflow circuitry and not useful for unstructured asynchronous designs. It requires use of structured design constraints so the design can be viewed as combinational. Thus, its use was limited to a few companies with designs large enough to justify the perceived overhead of the design constraints. However, early use of cycle-simulation was very effective for many large systems and these early developments provided the foundations for modern cycle-simulators that are now beginning to gain acceptance for complex ICs.

Simulation has a couple of severe inherent problems. First, unlike test generation or physical design, there is no precise measure of completeness. Test generation has the stuck-fault model, and physical design has a finite list of nets that must be routed. However there was no equivalent metrics to determine when verification of the design is complete, or when enough simulation is done. During the 1970s, research began to develop a metric for verification completeness, but to this day none has been generally accepted. Use is made of minimum criteria, such as all nets must switch in both directions, and statistical models using random patterns. Recent work in formal verification is applying algorithmic approaches to validate coverage of paths and branches within the model. However, the generally accepted goal is to "do more." Second, it is a hard and time-consuming task to create effective simulation vectors to verify a complex design. DV simulators typically support a rich high-level language for the stimulus generator, but it still requires the thought, experience, and ingenuity of the design verification engineer to develop and debug these programs. Therefore, it is desirous to simulate the portion of the design being verified in as much of the total system environment as possible and have the simulation create functional stimulus for the portion to be verified. Likewise, it is hard to validate the simulated results for correctness, making it desirable to use them in the

full system environment where it is easier to validate them. Ideally, the owner of an ASIC chip being designed for a computer could simulate that chip within a model of all of the computer's hardware, the microcode, and the operating system, and use example software programs as the ultimate simulation experiment. To even approach this goal, however, the simulators have to become faster and faster.

During the 1980s, research and development took place on custom hardware simulators and accelerators. Special purpose hardware-simulators use massively parallel instruction processors with much customized instruction sets to simulate gates. These provide simulation speeds that are orders of magnitude faster than software simulation on general-purpose computers. However, they are expensive to build, lack the flexibility of software simulators, and the hardware technology they are built in soon becomes outdated (although general parallel architectures may allow the incremental addition of additional processors). Hardware accelerators use custom hardware to simulate portions of design in conjunction with the software simulator. These are more flexible, but they still have the inherent problems that their big brothers have. Nonetheless, use of custom hardware to tackle the simulation performance demands has gained acceptance in many companies and they are commercially available today using both gate-level and HDL design descriptions.

Timing Analysis

The practice of divide and conquer in design verification started in the 1970s with the introduction of the behavorial model. Another divide-and-conquer style born in the 1970s, that has become generally popular in the 1990s, is to separate verification of the design's function from its timing. With the invention of scan design, it became possible to verify logic as a combinational circuit using high-speed compiled cycle-simulators. Development of path tracing algorithms to verify timing resulted in a technique to verify timing without simulation, thus providing a complete solution rather than an experimental one. This became known as static timing analysis. Static timing analysis (TA) is used to analyze signal paths from primary inputs to latches, latches to latches, latches to primary outputs, and primary inputs to primary outputs. At each gate, the TA program computes the min-max time that gate will change in state based on the min-max arrival times of its input signals. TA tools do not simulate the gate function; they add only its contribution to the path delay, although the choice of using rise or fall times for the gate is based on whether it has a complimentary output, or not. Because the circuitry between the latches is combinational, only one pass needs to be made across the design. The addition can be based on the minimum rise or fall delay for gates or both, providing a min-max analysis. The designer specifies the required arrival times for paths at the latches or primary outputs, and the TA program compares these with the actual arrival times. The difference between the required arrival and the actual arrival is defined as slack. The TA tool computes the slack at the termination of each path, sorts them numerically, and provides a report. The designer then verifies the design correctness by analysis of all negative slacks.

Engineering judgment is applied during the analysis, including the elimination of false-path conditions. A false-path is a signal transition that will never occur in the real operation of the design. Since the TA tool does not simulate the behavior of the circuit, it cannot automatically eliminate all false paths. Through knowledge of the signal polarity, TA can eliminate false paths caused by the fan-out and reconvergence of certain signals. However, other forms of false-paths are only identifiable by the designer.

Formal Verification

Development of a formal method to verify the equivalence of two different representations of a design began during the 1970s. Boolean verification analyzes a circuit against a known good reference and provides a mathematical proof of equivalence. An RTL model, for example, of the reference circuit is verified using standard simulation techniques. The Boolean verification program then compiles the known-good reference design into a canonical NAND-NOR equivalent circuit. This equivalent circuit is compared with the gate-level hardware design using sophisticated theorem provers to determine equivalence. User controls map the logically corresponding design signals in the RTL to their equivalent node in the gate-level model. To reduce processing times, formal verification pre-processes the two circuits to create an overlapping set of smaller logic cones to be analyzed. These cones are simply the set of logic

traversed by backtracing across the circuit from an output node (latch positions or primary outputs) to the controlling input nodes (latch positions or primary inputs). User controls specify the nodes that are supposed to be logically equivalent between the two circuits.

Early work in this field explored the use of test generation algorithms to prove equivalence. Two cones to be compared can be represented as $F_{cone1} = f$(a,b,c,X,Y,Z..) and $F_{cone2} = f$(d,e,f,.,X′,Y′,Z′..). F_{cone1} and F_{cone2} are user defined output nodes to be compared for equivalence. The terms a,b,c and d,e,f represent the set of input nodes for the function, and X,Y,Z are the computational sub-functions. User input defines the equivalence between a,b,c and d,e,f. If F_{cone1} and F_{cone2} are functionally equivalent, then the value at of G must be 0 for all possible input states. If the two cones are equivalent, then use of D-ALG test generation techniques for G = F_{cone1} XOR F_{cone2} will be unable to derive a test for the stuck-at-zero fault on the output of G. Similarly, the use of random pattern generation and simulation can be applied to prove equivalence between cones by observing the value on G for all input states.

Research across the 1980s provided improvements in Boolean equivalence checking techniques and models (such as binary decision diagrams), and modern Boolean equivalence checking programs may employ a number of mathematical and simulation algorithms to optimize the overall processing. However, Boolean equivalence checking methods require the existence of a reference design against which equivalence is proved. This implies there must be a complete validation of the reference design against the design specification. Validating the reference design has typically been a job for simulation. However, this leads to the problems of assuring coverage and completeness of the simulation experiment. Consequently, formal methods to validate the correctness of functional-level models have become an important topic in EDA. Modern design validation tools use a combination of techniques to validate the correctness of a design model. These typically include techniques used if software development to measure completeness of simulation test cases, such as

- checking for coverage of all instructions [in the model]
- checking to assure all possible branch conditions [in the model] were exercized.

They may also provide more formal approaches to validation, such as

- checking [the model] against designer asserted conditions (or constraints) that must be met
- techniques that construct a proof that the intended functions are realized by the design.

These concepts and techniques continue to be the subject of research and will gain more importance as the size of IC designs stretch the limits of simulation based techniques.

Verification Methodologies

With the use of structured design techniques, the design to be verified can be treated as a set of combinational designs. With the arsenal of verification concepts that began to emerge during this period, the user had many verification advantages not previously available. Without structured design, delay simulation of the entire design was required. Use of functional models intermixed with gate-level descriptions of subsections of the design provided major improvements, but it was still very costly and time-consuming. Further, to be safe, the practical designer would always attempt to delay simulate the entire design at the gate-level.

With structured design techniques, the designer could do the massive simulations at the RTL level, focusing on the logic function only, without regards for timing. Cycle-simulation improved the simulation performance by one to two orders of magnitude by eliminating repetitive simulations of the clock circuitry and use of compiled (or table look-up) simulation. For some, the use of massively parallel hardware-simulators offered even greater simulation speeds — the equivalent of hundreds of millions events per second. Verification of the logic function of the gate-level design could be accomplished by formally proofing its equivalence with the simulated RTL-level model instead of additional simulation. Static timing analysis provided the basis to verify the timing of all data paths in a rigorous and methodical manner. Functional and timing verification of the clock circuitry could be accomplished with the standard delay-simulation techniques using the event simulator. Collectively, these tools and techniques provided

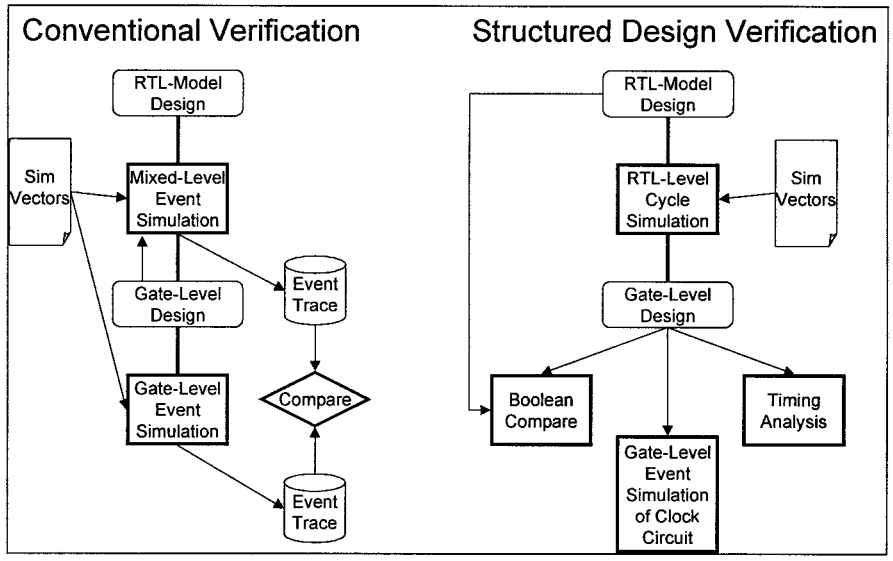


FIGURE 78.8 Verification methodology.

major design productivity improvements. However, the IC area and performance overhead required for these structured design approaches limited the number of designs taking advantage of them. Across the 1980s, as the commercial EDA business developed, these new verification tools and techniques remained as in-house tools in a few companies. Commercial availability did not emerge until the 1990s when the densities and complexities of ICs began to demand the change.

The 1980s — Birth of the Industry

Up to the 1980s design automation was, for the most part, developed in-house by a small number of large companies for their own proprietary use. Almost all EDA tools operated against large mainframe computers using company-specific interfaces. High-level modeling languages were unique and most often proprietary, technology rules formats were proprietary, and user interfaces were unique. As semiconductor foundries made their manufacturing lines available for customer-specific chip designs, however, the need for access to EDA tools grew. With the expansion of the application specific integrated circuit (ASIC), the need for commercially available EDA tools exploded. Suddenly, a number of commercial EDA companies began to emerge and electronic design companies had the choice of developing tools in-house or purchasing them from a variety of vendors. The EDA challenge now often became one of integrating tools from multiple suppliers into a homogeneous system. Therefore, one major paradigm shift of the 1980s was the beginnings of EDA standards to provide the means to transfer designs from one EDA design system to another or from a design system to manufacturing. VHDL and Verilog matured and became the industry-standard hardware description languages (HDL). EDIF[12] (Electronic Data Interchange Format) was developed as an industry standard for the exchange of netlist, and GDSII became a standard interface for shapes data to manufacturing. This was only the start of a necessary change, however. As time progressed, the need for standard interfaces between EDA tools would become evident in order to provide efficient design flows containing EDA tools from a number of different vendors.

A second paradigm shift of the 1980s was the introduction of the interactive workstation as the platform for EDA tools and systems. Although some may view it as a step backwards, the "arcade" graphics capabilities of this new hardware caught the attention of enough designers to make it a clear choice over the mainframe wherever possible. For a time, it appeared that many of the advances made in alphanumeric HDLs were about to yield to pizzazz of graphical schematic editors. Nevertheless, although the graphics pizzazz may have dictated the purchase of one solution over another, the dedicated processing, and the

ability to incrementally add compute power made the move from the mainframe to the workstation inevitable. Early commercial EDA entries such as those from Daisy (Logician) and Valid (SCALD) were developed on custom hardware using commercial microprocessors from Intel and Motorola. This soon gave way, however, to commercially available workstations using RISC-based microprocessors, and Unix became the defacto operating system standard. During the period, there was a rush of redevelopment of many of the fundamental algorithms for EDA to the workstation. However, as commercial products with the new power of high-function graphics, these applications were vastly improved along the way. Experience and new learning streamlined many of the fundamental algorithms. The high-function graphic display provided the basis for enhanced user interfaces to the applications. The commercial ASIC business provided focus on the need for technology libraries from multiple manufacturers. Finally, there was significant exploration into custom EDA hardware such as hardware simulators and accelerators and parallel processing techniques.

From an IC design perspective, however, the major paradigm shift of the 1980s was synthesis. With the introduction of synthesis, automation could be used to reduce a HDL description of the design to the final hardware representation. This provided major productivity improvements for ASIC design, as chip designers could work at the HDL level and use automation to create the details. Also, there was a much higher probability that the synthesis-generated design would be correct than for manually created schematics. The transgression from the early days of integrated circuit design to the 1980s is similar to what occurred earlier in software. Early computer programming was done at the machine language level. This could provide optimum program performance and efficiency, but at the maximum labor cost. Programming in machine instructions proved too inefficient for the vast majority of software programs, thus the advent of assembly languages. Assembly language programming offers a productivity advantage over machine instructions because the assembler abstracts up several of the complexities of machine language. Thus, the software designer works with less complexity, using the assembler to add the necessary details and build the final machine instructions. However, assembly language programming is still at a low level of abstraction. Introduction of functional level program languages and compilers provided even more productivity improvements by providing a set of programming statements that each provided function that would otherwise require many machine instructions to implement. Thus the level at which the programmer could now work was even higher, allowing him to construct programs with far fewer statements. The analog in IC design is the progression of transistor-level design (machine level), to gate level design (assembler level), to HDL-based design. Synthesis provided the basis for HDL-based design, its inherent productivity improvements, and major changes to the IC design methodology.

Synthesis

Fundamentally, synthesis is a three-step process:

- Compile a HDL description of a design into an equivalent NAND-NOR description.
- Optimize the NAND-NOR description based on design targets.
- Map the resulting NAND-NOR description to the building blocks supported for the final package.

Although work on synthesis techniques has occurred on and off since the beginnings of design automation and back to TTL-based designs, it was not until gate array style ICs reached a significant density threshold that it found production use. In the late 1970s and 1980s, considerable research and development in industry and universities took place on high-speed algorithms and heuristics for synthesis. In the early 1980s, IBM exploited the use of synthesis on ICs used in the 3090 and AS400 computers. These computers used chips that were designed from qualified sets of pre-designed functions interconnected by personalized wiring. These functions (now called cells) represented the basic building blocks for each specific IC family. The computers used a high number of uniquely personalized chip designs, so it was advantageous to design at the HDL level and use automation to compile to the equivalent cell-level detail. The result was significant improvements to overall design productivity. Today, synthesis is a fundamental cornerstone of the design methodology for ASICs (both gate-array and standard-cell) supporting both VHDL and Verilog as the standard input.

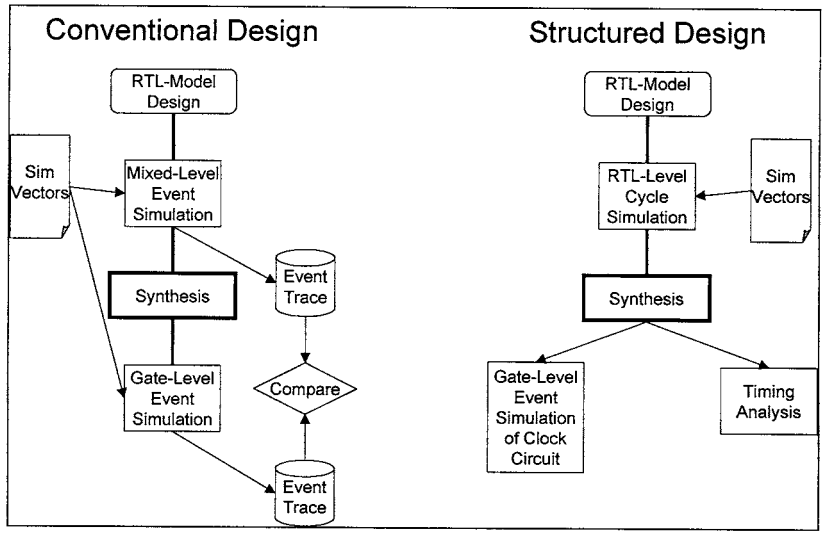


FIGURE 78.9 Synthesis methodology.

As the complexities of IC design have increased, so have the challenges for synthesis. Early synthesis had relatively few constraints to be observed in its optimization phase. Based on user controls, the design was optimized for minimum area, minimum fan-out (minimum power), or maximum fan-out (maximum performance), etc. This was accomplished by applying a series of logic reduction algorithms (transforms) that provide different types of reduction and refinement.[13] Cell behavior could also be represented in primitive-logic equivalent form, and a topological analysis could find matches between the design gates and equivalent cell patterns. The mapping phase would then select the appropriate cells based on function and simple electrical and physical parameters for each cell in the library, such as drive strength, area, etc. Early on, synthesis was not overly concerned with signal delays in the generated design or other electrical constraints that the design may be required to meet. As IC feature sizes decrease, however, these constraints are getting more demanding and automation is required to achieve them. Thus, synthesis is now required to make intelligent choices not only based on the HDL function and simple controls, but also on a number of constraints that must be met by its result.

The design input to modern synthesis tools is not the functional HDL design alone, but now includes constraints such as the maximum allowed delay along a path between two points in the design. This complicates the synthesis decision process, as it must now generate a solution that meets the required function with a set of cells and interconnections that will fall within the required timing constraints. Therefore, additional tradeoffs must be made between the optimization and mapping phases, and the effects of interconnection penalty need to be considered. Additionally, synthesis must now have additional technology characteristics available to it such as delay for cells and wiring. To determine this delay, it is necessary to understand the total load seen by a driver cell as a result of the input capacitance of the cells it drives and the RC parasitics on the interconnect wiring. However, the actual wiring of the completed IC is not yet available (since routing has not taken place) and estimates for interconnect parasitics must be made. These are often based on wire load models that are created from empirical analysis of other chip designs in the technology. These wire load models are tables that provide an estimate of interconnect length and parasitic values based on the fan-out and net density.

As interconnect delay became a more dominant portion of path delay, it became necessary to refine the wire load estimations based on more than total net count for the chip. More and more passes through the synthesis-layout design loop became necessary to find a final solution that achieved the required path delays. The need for a new design tool in the flow became apparent as this estimation of interconnect delay was too coarse. The reason is that different regions on a typical chip have different levels of real

estate (net) congestion. This means that wire load estimates differ across different regions on the chip because the average amount of wiring length for a net with a given pin count is different by region. Conversely, the actual placement of cells on the chip effect the congestion and therefore the resulting wire load models. Consequently, a new tool, called floor planning, was inserted in the flow between synthesis and layout.

Floorplanning

The purpose of floorplanning is to provide a high-level plan for the placement of functions on the chip, which is used to both refine the synthesis delay estimations and direct the final layout. In effect, it inserts the designer into the synthesis-layout loop by providing a number of automation and analysis functions used to effectively partition and place functional elements on the chip. As time progressed, the number of functions integrated into the floorplanner grew to a point where today; it is more generally thought of as design planning.[14] The initial purpose of the floorplanner was to provide the ability to partition the chip design functions and develop a placement of these partitions that optimized the wiring between them. Optimization of the partitioning and placement may be based on factors such as minimum wiring or minimum timing across a signal path. A typical scenario for the use a design planner is as follows:

- Run synthesis on the RTL description and map to the cell level
- Run the design planner against the cell-level design with constraints such as path delays and net priorities:
  - partition and floorplan the design, either automatically or manually, with the design planner
  - create wire load models based on the congestion within the physical partitions and empirical data
- Rerun synthesis to optimize the design at the cell level using the partitioning, global route, and wire load model data.

User-directed graphics and placement-wiring software algorithms are used to place the partitions and route the inter-partition (global) nets. One key to the planner is the tight coupling of partitioning and routing capability with electrical analysis tools, such as power analysis, delay calculation, and timing analysis. Checking of the validity of a design change (such as placement) can be made immediately on that portion of the design that changed. Another is that it needs to be reasonably tightly connected to synthesis, as it is typical to pass through the synthesis-planner-synthesis loop a number of times before reaching a successful plan for the final layout tools.

After a pass in the floorplanner, more knowledge is available to synthesis. With this new knowledge, the optimization and mapping phases can be rerun against the previous results to produce a refined solution. The important piece of knowledge now available is the gate density within different functions in the design (or regions on the chip). With this, the synthesis tool can be selective about which wire load table to use and develop a more accurate estimate of the delay resulting from a specific solution choice.

Floorplanning has been a significant enhancement to the design process and is widely used today. In addition to floorplanning, modern design planners include support for clock tree, design power bus design, I/O assignment, and a wealth of electrical analysis tools. As will be discussed later, semiconductor technology trends will place even more importance on the design planner and dictate an ever-tightening integration between it with both synthesis and layout. Today, synthesis, design planning, and layout are three discrete steps in the typical design flow. Communication between these steps is accomplished by file-based interchange of the design data and constraints. As designs become more dense and with the growing complexities of electrical analysis required, it will become necessary to integrate these three steps into one tight process as are the functions within the modern design planners.

The 1990s — The Age of Integration

Before EDA tools were commercially available, software architecture standards could be established to assure smooth integration of the tools into complex EDA systems supporting the entire design flow.

Choice of languages, database and file exchange formats, host hardware and operating systems, and user interface conventions could be made solely by the developing company. Moreover, the need for comprehensive EDA systems supporting the entire flow through design and manufacturing release in a managed process is paramount. Design of ICs with hundreds of thousands of gates (growing to the millions) across a design team requires gigabytes of data contained within thousands of files. All of this data must be managed and controlled to assure its correctness and consistency with other design components. Design steps must be carried out and completed in a methodical manner to assure correctness of the design before release to manufacturing. The EDA system that encompasses all of this must be efficient, effective, and manageable. This was always a challenge for EDA system integrators, but the use of EDA tools from external vendors compounded the problem. EDA vendors may develop their tools to different user interfaces, database and file formats, computer hardware, and operating systems. In turn, it is the EDA system integrator who must find a way to connect them in a homogeneous flow that is manageable, yet provides the necessary design efficiency across the tools. By the late 1980s, it was said that the cost to integrate an EDA tool into the in-house environment was often over twice that of the tool itself. Consequently, the CAD framework initiative (CFI) was formed whose charter was to establish software standards for critical interfaces necessary for EDA tool integration. Thus began the era of the EDA Framework.

The EDA framework[15] is a layer of software function between the EDA tool and the operating system, database, or user. CFI defined a set of functional elements for this software layer each of which had a standard interface and provided a unique function in support of communication between different tools. The goal was to provide EDA system integrators the ability to choose EDA tools from different vendors and be able to plug them into a homogeneous system to provide an effective system operation across the entire design flow. The framework components included

- Design information access — a formal description of design elements and a standard set of software functions to access each, called an API (application program interface)
- Design data management — a model and API to provide applications the means to manage concurrent access and configuration of design data
- Inter-tool communication — an API to communicate control information between applications and a standard set of controls (called messages)
- Tool encapsulation — a standard for specifying the necessary information elements required to integrate a tool into a system such as, input files, output files, etc.
- Extension language — specification of a high-level language that can be used external to the EDA tools to access information about the design and perform operations on it.
- Session management — a standard interface used by EDA tools to initiate and terminate their operation and report errata, etc.
- Methodology management — APIs used by tools so that they could be integrated into software systems to aid the design methodology management
- User interface — a set of APIs to provide consistency in the graphical user interface (GUI) across tools.

Each of these functional areas needs to be considered for integration of EDA tools and provide interoperability between them. In its first release, CFI published specifications for design representation and access (DR), intertool communication (ITC), tool encapsulation (TES), and extension language (EL). Additionally, the use of MOTIF and X-Windows were specified as the GUI tool kit. However, for many practical, business, and human nature reasons, these standards did not achieve widespread adoption across the commercial EDA industry beyond the use of MOTIF and X-Windows. Offerings for DR, ITC, and EL became available in products such as CDB (Mentor Graphics' DR based product), ToolTalk (Sun Microsystems' ITC product), and SCHEME (the CFI EL standard). Because, however, use of these required EDA tools to modify their internal structures to interact directly with the committee-derived APIs, their use across the industry was less than desired and the "Plug-and-Play" motto of CFI was not realized. However, the work at CFI did serve to formalize the science of integration and communication, and a number of the information models developed for the necessary components are used today but often are slightly modified and within a proprietary interface.

As it was learned, not only was it costly to change EDA tools to use a different internal interface (API), but it was not considered good business to allow tools to be integrated into commercial system offerings from competitors. For a period, several framework offerings became available, but these quickly became the end product rather than a means to an end. These products were marketed as a means to bring outside EDA tools into the environment, but few vendors modified their tools to support this by adopting the CFI APIs. Instead, to meet the need for tool integration, the commercial industry focused on data flow only and chose an approach that provides communication between tools by file generation and translation. Consequently, today's EDA systems are typically a disparate set of heterogeneous tools stitched together by the creation of data files. These files are created from one tool and translated by another tool [within the design flow] to its internal data structures. Many of these formats evolved from proprietary formats developed within commercial companies that were later transferred or licensed for use across the industry. Common EDA file formats in use today are

- Electronic data interchange format (EDIF) — netlist exchange (EIA→IEC)
- Physical data exchange format (PDEF) — floorplan exchange (Synopsys →IEEE)
- Standard delay file (SDF) — delay data exchange (Cadence Design Systems→IEEE)
- Extended standard parasitic file (ESPF) — parasitic data exchange (Cadence Design Systems→IEEE)
- Library exchange file (LEF) — physical charactistics information for a circuit family (Cadence)
- Data exchange format (DEF) — physical data exchange (Cadence)
- Library data format (.LIB) — ASIC cell characterization (Synopsys)
- ASIC library format (ALF) — ASIC cell characterization (Open Verilog International)
- VISIC high level design language (VHDL) — behavorial design description language (IEEE)
- Verilog — RTL-level design description language (Cadence Design Systems→IEEE)

These "standard" formats have provided the desired engineering and business autonomy necessary to get adoption and their use across the industry has improved the ability to integrate tools into systems drastically compared with the 1980s. It has proved to be an effective method for the design of today's commodity ICs above .25 micron. However, the inherent ambiguities, translation requirements, rapidly increasing file sizes (due to increasing IC density), and the fundamental nonincremental sequential nature of design flows based on them will soon give way to IC technology advances.

78.3 The Future

It may be said that the semiconductor industry is the most predictable industry on earth. Whether it was an astute prediction or the cause, the prediction of Gordon Moore (now known as Moore's Law) has held steadily across the past two decades. Every 18 months, the number of transistors or bits on an IC doubles. To accomplish this, feature sizes are shrinking, spacing between features is shrinking, and chip die sizes are increasing. In addition, the fundamental physics that govern the electrical properties of these devices and between them has been documented in electrical engineering textbooks for some time. However, there are points within this progression where paradigm shifts occur in the elemental assumptions and models required to characterize these circuits, and where fundamental EDA design and analysis must change. Earlier we discussed paradigm shifts in design verification because of inability to repair. We also discussed shifts resulting from the number of elements in a design versus the necessary tool performance and designer productivity. Across the past three decades, we have seen shifts in test, verification, design abstraction, and design methodologies. Now, as semiconductor technology crosses the .25 micron barrier, another paradigm shift is required. Feature packing, decreased rise times, increased clock frequencies, increased die sizes, and the explosion of the number of switching transistors are all interacting to place new demands of models, tools, and EDA systems. Thus, the next decade will require new work in design tools, rules, systems, and schools for problematic areas of delay, signal integrity, power, test, and mixed technology.

SIA National Technology Roadmap for Semiconductors

The semiconductor industry association and SEMATECH periodically publish a report called the "National Technology Roadmap for Semiconductors"[16] (NTRS). This report characterizes planned semiconductor directions and advances across the next decade and the necessary technology advancements in all support areas including

- Design and test
- Process integration devices and structures
- Front-end [manufacturing] processes
- Lithography
- Interconnect
- Factory integration
- Assembly and packaging
- Environment, safety, and health
- Defect reduction
- Metrology
- Modeling and simulation

It provides a wealth of information on the semiconductor future, as well as the problematic areas that will arise and the inventions that will be required. Semiconductor advancements for high-function microprocessors that are documented in the NTRS that will impact the next-generation EDA tools and systems are shown in Table 78.1. The reader is encouraged to study the technology characteristics in the NTRS, as it details the roadmap across all intermediate years for high-volume ASICs as well. For simplicity, only the years 1997 and 2012 are given in Table 78.1, as these trends lead and represent the design points required. However, ASIC advancements do not lag far behind in time.

TABLE 78.1 NTRS Microprocessor Roadmap

Characteristic	1997	2001	2006	2012
Transistor Gate Length	.20 μm	.12 μm	.07 μm	.035 μm
Feature Size Scale Factor ($S_{feature}$)	1	.6	.35	.175
Die Size	300 mm^2	385 mm^2	520 mm^2	750 mm^2
Chip Size Scale Factor (S_{die})	1	1.13	1.32	1.58
Transistors per Chip	11×10^6	40×10^6	200×10^6	1.4×10^9
Gate Scale Factor (S_{gates})	1	3.6	18	127
Local Net Clock Frequency	750 MHz	1500 MHz	3500 MHz	10000 MHz
Local Cycle Scale Factor (S_{freq})	1	.5	.21	.075
Global Net Clock Frequency	750 MHz	1400 MHz	2000 MHz	3000 MHz
Global Cycle Scale Factor (S_{freq})	1	.54	.38	.25
Supply Voltage (V_{dd})	1.8 Volts	1.2 Volts	0.9 Volts	0.5 Volts
Maximum Power	70 Watts	110 Watts	160 Watts	175 Watts
Current Scale Factor ($S_{current}$)	1	2.3	4.6	9

In order to understand the implications of the NTRS data, it is convenient to review the effects of scaling on a number of electrical parameters.[17]

Delay is typically specified as the time from when the switching input to a driver gate reaches 50% of its voltage to the time the input to a receiver gate reaches its 50% value. It may be viewed as the driver gate delay plus the interconnect delay. Unloaded gate delay is a function of the gate capacitance and transistor resistance. The gate capacitance is a function of the gate width and length, and the gate-oxide thickness (T_{gox}). Decreasing the size or transistors by a factor of $S_{feature}$ decreases the capacitance by the same scale, since

$$C_{gate} = \varepsilon_{ox}\left(W_{gate}L_{gate}\right)/T_{gox} \alpha \left(S_{feature}S_{feature}\right)/S_{feature} = S_{feature}$$

The transistor on-resistance (R_{tr}) is proportional to supply voltage and current which both scale down proportionally, so the resistance is constant. The intrinsic gate delay, decreases by a factor of $S_{feature}$, since

$$T_{gate} \alpha R_{transistor}C_{gate} \alpha 1S_{feature} = S_{feature}$$

Thus, unloaded gate delays decrease in direct proportion with feature size ($S_{feature}$). Loading imposed on a driving gate effects this delay, however. And for cases where the interconnect resistance $R_{int} < R_{tr}$, this load is the sum of the capacitances of the receiver gates. As R_{int} gets larger, however, the actual delay of a driving gate must take into account the interconnect resistance and capacitance as well.

Interconnect delay is a function of the RC-time constant, which is a result of resistance and distributed capacitance of the interconnect. As feature sizes decrease by a factor of $S_{feature}$, the local interconnects are likewise scaled down in cross-sectional area, thus the width and height of the interconnect metal are each scaled down by a factor of $S_{feature}$. Additionally, the interconnection insulator layer (t_{ox}) is assumed to scale with feature size.

Interconnect resistance is proportional to the length and inversely proportional to the cross-sectional area of the interconnect

$$R_{int} \alpha L_{int}/\left(W_{int}H_{int}\right) \alpha L_{int}/S_{feature}S_{feature} = L_{int}/S_{feature}^{2}$$

where L_{int}, H_{int}, and W_{int} are the length, width, and height of the interconnect, respectively.

Capacitance is a function of the plate area and the spacing between the plates. The self-capacitance to ground of the interconnect is proportional to length and the width of the conductor and inversely proportional to t_{ox}:

$$C_{self} = \varepsilon_{ox}L_{int}\, W_{int}/t_{ox} \alpha L_{int}S_{feature}/S_{feature} = L_{int}$$

The resulting interconnect RC constant, therefore, scales by

$$RC_{int} \alpha \left(L_{int}/S_{feature}^{2}\right)\left(L_{int}\right) = L_{int}^{2}/S_{feature}^{2}$$

Scaling down of transistor sizes is accompanied by closer spacing. For local nets, the interconnect length (L_{int}) is scaled down by a factor of $S_{feature}$, thus the RC constant remains constant:

$$RC_{local} \alpha \left(L_{int}S_{feature}\right)^{2}/S_{feature}^{2} \alpha S_{feature}^{2}/S_{feature}^{2} = 1$$

Therefore, the delay along local interconnects is constant. However, as seen above, the gate delay is reduced by a factor of $S_{feature}$, so the relative impact of interconnect delay with respect to gate delay on the overall timing across a signal path is increasing. Below 1 μm, this became a significant factor. And around .35 μm, the delay of local nets approaches that of the gate. As feature sizes scale down further, the local interconnect delays will exceed the gate delays.

For global nets, the interconnect length does not scale down with the feature sizes. As the chip die size increases, the global net length increases. For these, the resulting RC constant scales like

$$RC_{global} = L_{int}^{2} / S_{feature}^{2} \alpha S_{die}^{2} / S_{feature}^{2}$$

Thus, the RC delay for global nets increases proportional to the square of the die size and inversely proportional to the square of the feature size decrease.

Applying simple RC approximation for delay against the NTRS data indicates that interconnect delay has become an important factor in the overall delay characterization of digital circuits (see Table 78.2). Thus, the need for accurate models for interconnect delay will continue to increase in importance. In practice, this type of uniform scaling across all interconnects (particularly global interconnects) is not applied; however, the importance of accurately modeling interconnect delay is evident in any case.

TABLE 78.2 Interconnect Delay Scaling

Characteristic	1997	2001	2006	2012
Feature size scale factor ($S_{feature)}$)	1	.6	.35	.175
Chip (die) size scale factor (S_{die})	1	1.13	1.32	1.58
T_{int}/T_{gate} local nets	1	1.7	2.8	5.7
T_{int}/T_{gate} global nets	1	5.9	40	466

Plate capacitance to ground is not the only consideration in today's ICs. Capacitance results whenever two conducting bodies are charged to different electric potentials and may be viewed as the reluctance of voltage to build or decay quickly in response to injected power. To reduce the interconnect resistive effects of decreasing metal cross-sectional area, the height of signal lines has not decreased at the same proportion. Thus, the plate areas effecting mutual capacitance do not scale down proportionally. This combined with the closer spacing between signal lines leads to an increased amount of mutual capacitance relative to self-capacitance. As a result, effects of mutual capacitance between signal lines is now as important in the overall design of ICs as is the self-capacitance of these lines. To get an accurate measure of the total capacitance along any interconnect wire, it is necessary to consider all conducting bodies within a sufficiently close proximity. This includes not only the ground plane, but also adjacent signal wires, as there is a mutual capacitance between these.

Figure 78.10 depicts three interconnect wires on a signal plane. An approximate formula for the capacitance[18] of the center wire is

C = Self Capacitance + Mutual Capacitance = $C_{int\text{-}ground} + 2\ C_m$. The mutual capacitance is a function of the plate area along the sides of the interconnects and the spacing between them and may be simply approximated as

$$C_m = \varepsilon_{ox} L_{int} H_{int} / S_{int}$$

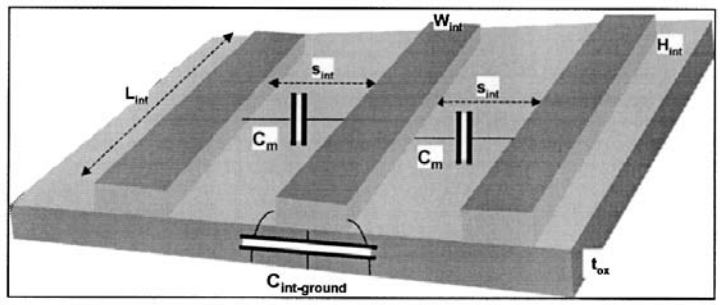


FIGURE 78.10 Mutual capacitance.

More accurate approximations for capacitance must take into account fringing effects, and the referenced literature gives the more accurate expression for capacitance as

$$C_{int-ground} = \left[\varepsilon_{ox}\left(1.15\left(W_{int}/t_{ox}\right)+2.8\left(H_{int}/t_{ox}\right)^{0.222}\right)\right]L_{int}$$

$$C_{m} = \left[\varepsilon_{ox}\left(0.03\left(W_{int}/t_{ox}\right)+0.83\left(H_{int}/t_{ox}\right)-0.07\left(H_{int}/t_{ox}\right)^{0.222}\right)\left(t_{ox}/s_{int}\right)^{1.34}\right]L_{int}$$

This mutual capacitance increases the RC delay for the interconnect. However, these simplifying equations assume the capacitive effect between the plates and ground. The actual effect of mutual capacitance is a function of the voltage potential between the conductors (Miller effect) and, therefore, a complex function of the signals on each line.

Noise is the introduction of unwanted voltage onto a signal line and is proportional to mutual capacitance and mutual inductance and inversely proportional to signal rise times. Mutual capacitance injects current into adjacent (victim) signal lines proportional to the rate of change in voltage along the exciting (aggressor) signal line. The amount of induced current can be approximated by the following equation:[19]

$$I_{mC} = C_{m}\left(dV_{aggressor}/dt\right)$$

Therefore, it can be shown that the voltage introduced (crosstalk noise) into a victim net by a switching aggressor is proportional to the mutual capacitance and inversely proportional to the rise time (τ_r) of the aggressor waveform:

$$V_{mC} = I_{m}Z_{victim} = C_{m}\left(\Delta V_{aggressor}/\tau_{r}\right)R_{victim}$$

This induced voltage can increase or even decrease[20] delays along victim nets, depending on the relative switching activities between the aggressor and victim. Further, if the crosstalk noise is of sufficient magnitude, it can give rise to faulty operation in the design. Accurate models to approximate the total effect of this crosstalk between signals for high-speed EDA processing is a current topic of much research.

Mutual inductance can also be the cause of noise. Using approximations and analysis similar to the used for noise induced by mutual capacitance yields the following relationships for mutual inductance:

$$V_{mL} = L_{m}\left(dI_{aggressor}/dt\right) = L_{m}\left(\Delta I/\Delta t\right)$$

Thus, rapidly changing current on an aggressor signal can also cause noise on a victim. These effects have classically not been of great importance on ICs, but as switching frequencies along wide lines increase, this noise must also be considered as well. This is particularly true for the power bus. As the number of transistors on the IC scales up, the amount of current required by switching transistors increases and rise times decrease.

$$V_{mL} = L_{m}\left(I_{total}/\tau_{r}\right)\alpha\left(S_{gates}/S_{features}\right)$$

Thus, the amount of induced noise resulting from mutual inductance becomes more significant as ICs scale down.

TABLE 78.3 Noise Scaling

Characteristic	1997	2001	2006	2012
Gate Scale Factor (S_{gates})	1	3.6	18	127
Feature Size Scale Factor ($S_{feature)}$)	1	.6	.35	.175
Relative Rise Time (τ_r) α $S_{feature}$	1	.6	.35	.175
Relative I_{total} α $S_{gates}S_{feature}$	1	2.16	6.3	22.2
Relative I_{tot} (NTRS)	1	2.3	4.6	9
Relative V_{mL}	1	3.8	13	51.4

EDA Impact

The references point to a number of texts that discuss the complexities of interacting electrical effects resulting from decreasing IC feature sizes and increased die sizes. These give rise to new design rules that design and analysis applications must deal with. The importance of crosstalk effects gives rise to the need for accurate parasitic extractors that can work in 2-D and 3-D space. Mutual capacitance gives rise to the need for more exacting delay calculators, and noise gives rise for new analysis tools that need to be aware of the switching activity of the IC in order to perform complete analysis. Switching activity is particularly important to power and clock bus design analysis and may be required on critical signal paths. Noise, crosstalk, Delta-I, reflection, and electromigration are well-defined properties, but their introduction into the CMOS IC design space is just beginning. Further, while the design complexities due to electrical effects are increasing, the number of circuits on the IC continues to increase. More required analysis on many more circuit elements is the fundamental challenge for EDA to meet.

EDA System Integration

With the design size increases that are projected, it will be necessary for the next-generation EDA systems to formalize the use all of the techniques that have been learned over the past three decades. It is not conceivable that current loosely coupled EDA system flows, which support only sequential design and analysis steps on a full design, will yield the required cycle times. Thus, it will be necessary to integrate EDA systems in a way that provides major decreases in cycle times and provides the increased accuracy and scope of electrical analysis to assure designs that meet specifications. Beyond the extensions to the application algorithms and development of new algorithms, the exploitation of hierarchy, abstraction, shared access, incremental analysis, and concurrent processing across design flows will become essential.

Hierarchy is the term used to describe the logical top-down organization of the design as a set of elements that can be designed and verified with minimum regards to other elements. Different levels within the hierarchy represent different levels of abstraction for the design, where each successively lower level in the hierarchy provides additional design detail. This hierarchy allows decomposition of the design into individual elements, sets of elements at different levels of abstraction, or sets of elements at the same level of abstraction. This allows for large teams of designers to work on different elements in parallel and reduces the overall processing times required individual EDA tools ($2^n > n2^m$ as discussed earlier). However, hierarchy preserves the integrity of the overall design and provides a system-level view when necessary.

Many steps in the design process can operate on an individual element without the specifics of other elements. Still, others may require an abstract representation or model of other elements within the hierarchy but not the complete design detail. And others will depend on the details of the portion of the design they are analyzing as well as for other hierarchical nodes and require an expanded occurrence-level description of the design. For example, functional-level design of a hierarchical element can be performed in a large part without the details of other elements. On the other hand, early delay calculation needs a representation of the actual loading imposed by elements connected to outputs of the element being analyzed. Yet detailed parasitic extraction requires the detailed wiring on all elements in the design

hierarchy. Design systems will need to recognize and support each of these levels of representation for the applications they serve.

Abstraction is used to describe the representation of a design at a less detailed (higher) level. Through abstraction, analysis tools deal with fewer elements, thus improving performance and memory requirements. The support for abstraction also provides the basis for a continuum of design analysis across the entire design-cycle. This top-down approach to design is common within simulation, but it will need to be expanded up and out in future EDA systems. Expanding formal design support up will provide the designer formal support in the very early stages of systems design where design algorithms, instruction set analysis, global timing, hardware-software tradeoffs, design reuse, and functional partitioning are evaluated. Thus, there will be necessary movement to higher-level (with respect to Verilog and VHDL) system design languages and analysis tools. Today, the use of simulation and synthesis at the RTL abstracted level is considered commonplace. Support at an even more abstract behavorial level will provide an additional level of efficiency. Expansion out provides for more design and analysis tools operating off abstractions of the physical IC. Accurate abstract models for physical design elements may be key to providing necessary efficiencies in physical design and analysis.

Shared relates to the architectural principle that the design tools operate off a common representation and a single shared instance of the design. That is to say, there is one common definition of the design elements, and physical store of design data is nonredundant. Today's EDA system architectures communicate data between different design tools via file transfers. These files typically represent design information already existing in the permanent storage of other files or databases in the system. Because there is no common information model for these design elements, there are many different definitions used across these different files. With the increasing number of design elements on an IC, an explosion in the number of bytes of permanent storage and run times required by applications to read and translate the data will occur. Use of standard file formats (such as EDIF, SDF, ESPF, LEF, DEF, etc.) to transfer different types of design information is becoming stressed to the limit even today. Many large designs require gigabytes of storage for these individual files and often hours of processing time to translate them into the form required by individual design applications. Across the next decade, the number of transistors will scale up by two orders of magnitude, and these interchange files can be expected to expand at the same rate. Further, more and more design tools will need to access additional types of design data to analyze increasing important electrical effects of delay and noise, making it more difficult to compartmentalize data elements into independent files. Thus, the need for a more data-centric view in design systems will need to replace today's tool-centric architectures.

Incremental means simply that only the effected portion of the design needs to be processed to analyze a change. The goal of this is to make the analysis time required for a design change proportional to the size of that change and not the size of the entire circuit. With the increase in design elements, it will be too time-consuming to re-analyze the entire IC, or even a hierarchical node within it, for every design change. Analysis applications will need enhanced algorithims to support analysis of changed portions of the design, and EDA systems will need methods to keep efficient records of design changes. This implies the tagging of records within the design database that have changed since some reference point in time. Using this information, applications can then determine what portions of the design to process (or reprocess). Strictly speaking, however, it is not necessary to keep these change tags directly with the data records. Change history can be kept and managed external to the design data either as change flags or design change orders to be processed on demand.

Structuring design algorithms to perform incrementally is no longer a simple goal. Historically, for example, a wire router could reroute a net by knowing only the open channels. It would not require detailed information about logical nets and physical wires occupying blocked channels—only that they are blocked. Now, because of the interaction of electrical effects between features on the IC, the router needs details about wire in occupied channels to correctly predict delay and noise. Further, just re-routing the changed net may not be sufficient. It is possible that the rerouted net produces a mutual capacitance with a previously correctly routed net that results in signal timing of the path that net is within beyond the critical threshold. Similar effects occur in other long-running applications, such as parasitic extraction. Here, not only does

the changed net need to be re-extracted, but so may the nets within its sphere of influence. For applications such as this, a conceptual region of influence around the change needs to be defined and other elements within that region need to be considered. Of course, in the case of the router, this region of influence could grow as additional rework is performed to correctly route the original changed net.

Concurrent is the term used to describe a closer coupling (tighter integration) between design and analysis. Typical EDA system flows of today are highly sequential in nature. That is, a set of design changes is made and this is followed by a set of analysis applications to verify their validity within the design environment. Typically, in today's systems, individual tools are integrated by file-based data exchange, and each analysis tool needs to be started, parse the design changes, perform the analysis, and terminate before another type of analysis is performed. As design size scales up, cycle times across design and analysis will scale accordingly. Furthermore, as the complexities of achieving an accurate design increase, the number of design-analyze cycles may increase, thus compounding the overall design time increase. Future EDA design systems will need to provide analysis engines that can be called dynamically, on demand, by design programs to provide incremental analysis of changes (conceptually) concurrent with the change. Development and use of a common set of these sub-routines (engines) working off of the same design model, will not only streamline the design cycle time, but will also provide better convergence in the cycle. Using common analysis engines throughout the flow will assure that different applications will predict the same results for complex computations that are key to accurate prediction of the hardware. Common engines for delay calculation, noise prediction, extraction, design rule checking, etc. will greatly improve the design cycle by yielding the same answer independent of the design application calling them.

To summarize, future EDA systems will be required to take a data-centric view of the design process as opposed to the tool-centric view common today. There will necessarily be a tighter linkage between design planning and synthesis and design planning and layout. Links will not be to entire tools as we think of them today, but rather, to functional modules within those tools. For example, a design planner may realize that a floorplan cannot be derived to meet the required design points or constraints given the current netlist. Therefore, it may call up a synthesis transform to re-optimize the troublesome sub-section of the design. Global routing of nets between partitions may require the design planner to call on extraction and delay calculation engines, and local simulation to predict noise spikes in adjacent routes in order to find a correct result. Final layout will require tight concurrent integration with all forms of analysis engines as the end design becomes more dependent on complex inter-relationships between netlist, interconnect properties, feature spacing, frequency, and signal transition. A tightly integrated model that uses a common view of the design data and common engines will be more than just a productivity enhancement, it may be essential to converge all the constraints, rules, and physics to a valid result.

This issue is recognized today, and the movement toward data-centric EDA system architectures has begun in several major EDA system companies. Additionally, SEMATECH has launched a program to develop an industrywide model and access API, enabling all of the above architectural features across design-analysis functions. An additional goal of the SEMATECH chip hierarchical design system technical data (CHDStd) program is that the design model and API be open and available for use across the EDA industry. In this way, design flows can be developed using tools and engines from different suppliers and integrated into a homogeneous system. This is in contrast to the proprietary architectural solutions now being developed by the commercial EDA system companies which prevent the necessary level of integration with competing tools.

Delay

Simplifying assumptions and models for delay have provided the foundation for high-speed event-driven delay simulation since its initial use in the 1970s. More accurate waveform simulation such as that provided by SPICE implementations has played a crucial role in the characterization of IC devices across over three decades and continues to do so. SPICE-level simulations are important for characterization and qualification of ASIC cells and custom macros of all levels of complexity. However, the run times required for this level of device modeling are too large to support its use across entire ICs. SPICE is often used to analyze the most critical signal paths, but SPICE-level simulation on circuits with millions of

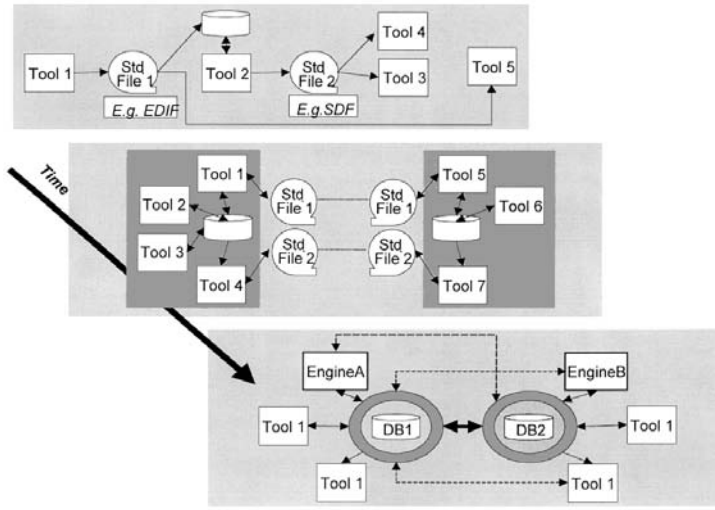


FIGURE 78.11 EDA system architecture progression.

transistors is not practical. Consequently, event-based simulation at the abstracted gate-level, or higher, is essential for IC verification. However, simplifying models used to characterize delay for discrete-event simulation and other timing analysis algorithms have had to improve and become more complex as feature sizes on ICs have shrunk and the importance of interconnect resistance and crosstalk have increased. In the future, these models will need to improve even more.

When ICs were well above 1 μm, gate delay was the predominant factor in determination of timing. Very early simulation of TTL logic used a simple model which assigned a fixed unit of delay to each gate and assumed no delay across the interconnects. Timing was based only on the number of gates through which a signal traversed along its path. As LSI advanced, this was refined to a lumped-capacitance model. With this model, gate delay was based on a foundry specified intrinsic value (actually a rise delay and a fall delay) that was adjusted up based on the capacitive load of the gates it fanned out to (receiver gates). At integration levels above 1 μm, the load seen by a gate was dominated by the capacitance of its receivers, so this delay model was sufficiently accurate. As feature sizes crossed below 1 μm, however, the parasitic wire resistance and capacitance became a significant factor. More precise modeling of the total load effect on gate delay and interconnect delay had to be taken into account. By .5 μm the delay attributed to global nets about equaled that of gates, and by .35 μm the delay attributed to short nets equaled the gate delay.

Today, a number of models are used to represent the distributed parasitics along interconnects using lumped-form equations. The well-known π-model, for example, is a popular method to model these distributed RCs. Different degrees of accuracy can be obtained by adding more sections to the equivalent lumped RC model for interconnect.

Delay calculators are developed to analyze the circuit topology, in particular, the length of interconnect wires and gate loads and compute delay across circuit elements based on the characterization model used. Calculated delay for gates and interconnects are used by design tools such as synthesis, floorplanning, and routing to constrain design choices, and analysis tools such as simulators and STV for circuit verification.

As the importance of timing-based design tools grows, integration of multiple-sourced design tools into an EDA flow becomes problematic. As more EDA vendors use built-in models for gate and interconnect delays, these different delay models often result in different calculated delays for the same circuit. This results in difficult and time-consuming analysis on the designers' part to correlate the different results. In mid-1994, an industry effort began to develop a common delay-calculation engine.[21] The goal

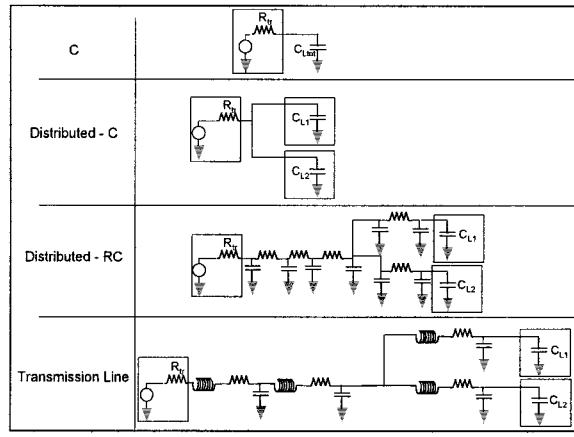


FIGURE 78.12 Delay models.

of this effort was to provide a standard language usable by ASIC suppliers to specify the delay calculation models and equations (or table look-ups) for their technology families. In this way, the complexities of deciding appropriate models to be used to characterize circuit delay and the resulting calculation expressions could be placed in the hands of the technology supplier rather than across the EDA vendors. These calculation models and expressions were to be rendered into a form that could be directly used by any EDA application requiring delay calculation and in a way that would protect the intellectual property these specifications represent. By providing a single source for the calculation of delay, all applications in the design flow could provide consistency in the computed results. The delay calculation language (DCL) technology developed by IBM was contributed to industry as the basis for this. Today, DCL has been extended to cover power calculation in addition to delay, and it has been released to IEEE as an open industry standard (delay and power calculation system — DPCS, IEEE 1481).

DPCS is an example of an industrywide available computational engine as described in the previous section. DCL language statements that specify the characterizing models for delay and power are compiled into an executable program that can be used by any EDA tool. The power of the DCL language leads the technologist to a means to characterize the delay model and calculation algorithms. Once compiled, the DCL program becomes a calculation engine for a particular technology family that can be called on demand by any EDA application adhering to the DPCS API. Compilation protects the intellectual property of the semiconductor company while providing a common executable subroutine. With DPCS, the EDA application provides the design details, and the calculation engine returns the calculated delays. The DPCS-API specifies the interface between the compiled engine and the EDA application. EDA applications may call the DPCS engine to compute delay for cells and interconnects individually or traverse across the entire design. Use of the DPCS engine across a broad scope of EDA tools provides the necessary accuracy and consistency in the computed delays resulting in shorter design times.

By .25 μm, mutual capacitance between signal lines begins to have a significant effect on interconnect delay. In this technology generation, it is necessary to consider wire size, length, and the proximity of adjacent wires. When features are significantly close so that mutual capacitance between them becomes sufficient, determination of the RC parasitics along an interconnect is no longer a simple multiplication of RC parasitics per unit length times interconnect length. Now, sophisticated tools must additionally analyze the proximity of features (including wires, vias, pads, etc.) and electrical characteristics of dielectrics between them in order to extract the parasitics from the design. Parasitic Extraction tools derive the total circuit parasitics, including the mutual capacitance in a 2-dimensional space and as ICs shrink further a 3-dimensional space.

Extraction tools are now in the commercial EDA market. They are available as stand-alone products and as part of complete system offerings. Unfortunately, similar to what occurred with delay calculation,

different extractors from different suppliers may yield different results because of different simplifying models used. Also, because of the large of processing time required to perform full extraction, it is becoming desirous to have the capability to mix and match extractors within a design flow. On nets where timing is critical, exacting parasitic models are needed, while for less critical nets, less accuracy is acceptable in favor of shorter run-times. Further, there is a growing need to call up extraction within a design tool to test the validity of a design choice. For these reasons, it is apparent that the industry need for incremental extraction engines will become as important as did the delay calculation engine.

Distributed RC models used to characterize delay are very accurate approximations so long as the rise-time (t_r) of the signals is much larger than the time-of-flight (t_{of}) across the wire. As die size scales up, so does the length of global wires and the resultant t_{of}. As transistor size scales down, so do the signal rise times. As t_{of} approaches t_r, then lossy-transmission line analysis is necessary to model delay. When $t_r > t_{of}$, lumped model approximation is accurate. Between these points is a grey area where transmission line analysis may be necessary depending on the criticality of the timing across a particular path. Published papers[22,23] address the question of where this level of analysis is important and design rules to be considered for time-critical global lines. For future EDA systems, however, it means more complexities in the design and analysis of circuits. Extraction tools will now need to derive inductance for interconnects so that the effects of the magnetic fields surrounding these lines can be factored into the delay calculations. Effects of mutual inductance also become important, particularly along the power lines which will drive large amounts of current in very short times ($e = L di/dt$). Design planners and routers will need to be aware of these effects when designing global nets and power bus structures. Design rules will need to be developed and checked by EDA tools to assure final designs that meet the expected timing. This again gives rise to the need for use of calculation and checking engines within design tools, which are developed with appropriate models and rules for these calculations by those best understanding the particular IC technology.

In summary, although the use of standard files such as SDF and the ESPF to communicate delays and parasitics between tools provides a great advantage to multi-vendor tool flows, it is problematic in the long term. As ICs shrink below .25 μm, the amount of data generated for millions of gates and their interconnects is huge, thus causing large performance overhead due to the read and translation times. Files exceeding 2-gigabytes are common for circuits of less than 1 million gates and will scale up with technology. Additionally, the practice of regenerating the entire set of delays or parasitics each time a design change is made becomes impractical, as the required processing times are too large. Therefore, future IC tools will need the ability to incrementally extract parasitics, calculate delay, and analyze the effects of electrically interacting IC features. However, different models and methods for these complex calculations can yield different conclusions across the design flow and result in design inefficiency. Thus, there will be a growing need for industry-standard calculation engines that are characterized by technologists who best understand their device technology and are made available for EDA tools independent of their source.

Signal Integrity

Beyond delay, the IC roadmap indicates the growing importance of a number of electrical analysis checks. Many of these checks have been necessary for high-performance MCMs and PCBs, but their importance for CMOS ICs is just beginning. The combined effect of decreased feature spacing, signal rise times, and voltage levels will soon reach a point where these new checks are of critical importance to all IC design. Of most importance is the effect of signal noise, which can result from mutual capacitance and mutual inductance between signal lines, and signal reflections. Noise can affect the integrity of signals and result in unacceptable delay or logic failures, and it is a function of the design's topology and functional operation. Therefore, analysis of noise is a complex and time-consuming process. However, because noise analysis is becoming more necessary across the entire design process, new analysis techniques, models, and design rules will become critical in next-generation design systems. Furthermore, its effects are increasing the need for enhanced support for design and analysis of power and clock distribution.

Signal integrity is the term used in relationship to whether or not injected noise will cause a logical fault. As discussed earlier, noise (unwanted voltage) can be introduced onto a signal line as the result of high-speed switching of an adjacent line. This is a result of mutual capacitance and mutual inductance.

If this noise is of sufficient magnitude, it can cause, for example, a latch to be set to an incorrect state. Therefore, design tools must now begin to make decisions based not only on the logical intent, the physical package constraints, and timing, but noise as well. As is the case for delay, design rules will be developed as a first order of defense and for the quick decisions for many design phases. New constraints, such as the maximum noise tolerated on a signal line, will become design rules that must be adhered to. However, this means that design tools, such as routers may need to call on noise calculation engines to determine if a decision falls within the tolerable limits. Unfortunately, for critical nets these (relatively) quick approximations and checks may be overly pessimistic. For these interconnects, it will be necessary to perform more exhaustive analysis of the effects of the noise. Unfortunately, to determine whether or not a signal has integrity requires knowledge of the functional operation in addition to the topological and electrical characteristics of the design and the properties of the materials. Therefore, a new type of EDA analysis is becoming more necessary in the design flows. This is signal integrity verification.

Precise signal integrity verification requires functional patterns to exercise signal paths, in addition to the electrical characteristics of those paths for accurate analysis. Induced noise on a signal may or may not represent a problem, depending on whether that condition can exist in the real machine operation (If a tree falls in the woods and no one is there to hear it, does it make noise?) and whether the path is sensitive to the state of the noisy net. Therefore, simulation of functional patterns is necessary to completely determine if a noise spike causes a fault. This, however, takes on the same problems as functional design verification — pattern generation and completeness analysis. Current and future research in this field must develop formal solutions and more algorithmic solutions to pattern generation — similar to those employed by test generation.

Test

Manufacturing test of ICs today is becoming a heuristic process involving a number of different strategies in combination to assure coverage and maximize generation costs and throughput at the tester. Today, an IC test employs the use of one or more of the following approaches:

- Static stuck-fault test using stored program testers

  The test patterns may be derived algorithmically or from random patterns, and the test is based on final steady state conditions independent of arrival times. The goal of stuck-fault test is to achieve 100% detection of all stuck-at faults (except for redundancies and certain un-testable design situations that can result from the design). Algorithmic test generation for all faults in general sequential circuits is often not possible, but test generation for combinational circuits is. Therefore, the use of design-for-test strategies (such as LSSD) is becoming more accepted.

- Delay test using stored program testers

  Delay test patterns may be algorithmically generated or functional patterns derived from design verification. The tester measures output pins for the logic state at the specified time after the test pattern is applied.

- Built-in self test (BIST)

  BIST tests are generated at the tester by the device under test, and output values are captured in scan latches for comparison with simulated good-machine responses. BIST tests may be exhaustive or random patterns, and the expected responses are determined by simulation. BIST requires special circuitry on the device-under-test (linear feedback shift registers and scan-latches) for pattern generation and results capture. To improve tester times, output responses are typically compressed into a signature and comparison of captured results and simulated results is made only at the end of sequences of tests.

- Quiescent current test (I_{DDQ})

  I_{DDQ} measures quiescent current draw (current required to keep transistors at their present state) on the power bus. This form of testing can detect faults not observable by stuck-fault tests (such a bridging faults), which may cause incorrect system operation.

Complicating semiconductor trends that will affect manufacturing test in the future are the speed at which these future chips operate and the quiescent current required for the large density of transistors on the IC. In order to perform delay test, the tester must operate at speeds faster than the device-under-test. That is, if a signal is to be measured at a time *n* after the application of the test pattern, then the tester must be able to cycle in less than *n* units of time. It is possible that the hardware costs required to build testers that operate at frequencies above future IC trends will be prohibitive. I_{DDQ} tests are very effective means to quickly identify faulty chips, as only a small number of patterns are required. Furthermore, these tests find faults not otherwise detected by stuck-fault test. Electric current measurement techniques are far more precise that voltage measurements. However, the amount of quiescent current draw required for the millions of transistors on future ICs may make it impossible to detect the small amount of excess current resulting from small numbers of faulty transistors. Obviously, new inventions and techniques may be introduced into the test menu. However, at present it appears that in the future manufacturing test must rely on static stuck-fault tests and BIST. Therefore, it is expected that more and more application of scan-design will be prevalent in future ICs. This will be have a second-order benefit for many EDA tools, as the design and analysis of combinational circuits and divide and conquer approaches can be utilized.

Design Productivity

The NTRS points out that future ICs will contain hundreds of million of transistors. Even with the application of the aforementioned architectural features, faster tools, and support for architectural design, it is unclear that design productivity (number of good-circuits designed per person-year) will achieve the necessary level. Designing and managing anything containing over 100,000,000 sub-elements is almost unthinkable. Doing it right and within the typical 18-month product cycles of today seems impossible. Yet, if this is not accomplished, semiconductor fabs may run at less than full production, and the ability to see returns the exorbitant capital expenditures required for new fabs will be at risk. Ultimately, this could negate the predictions in the NTRS.

Over the history of IC design, a number of paradigm shifts have enhanced designer productivity, the most notable being high-level design languages and synthesis. Use of design rules, such as LSSD, to constrain the problem has also represented productivity advances. Algorithm advances will continue to play a crucial role as well, as the design cycle-time is a function of computer resources required. Many of the EDA application algorithms, however, are not linear with respect to design size, and processing times can increase as an exponential function of transistors. Therefore, exploitation of hierarchy, abstraction, shared, incremental, and concurrent EDA system architectures will play an important role to overall productivity. However, even with all of this there is a major concern that industry will not be able to design enough good circuits fast enough. Consequently, there is a major push in the semiconductor industry toward design reuse. There is a major activity in the EDA industry (Virtual Sockets Interface Alliance — VSIA) to define the necessary standards, formats and, test rules to make design reuse real on ICs.

Design reuse is not new to electronic design or EDA. Early TTL modules were an example of reuse, standard cell ASICs, and PCB-level MSI modules. With each, the design of the component is done once, then qualified, then reused repeatedly in application specific designs. Reuse is also common within design teams. A logical function may be designed, verified, then reused in many other sections of the system's design. However, the ability to embed semiconductor designs from one process into chips manufactured on another process, or from one company to another, is just in its beginnings.

The VSIA has defined three different types of reusable property for ICs:

1. Hard macros — these functions have been designed and verified, and have completed layout. They are characterized by being a technology-fixed design and a mapping of manufacturing processes is required to retarget them to another fabrication line. The most likely business example for hard macros is that they will be available from the semiconductor vendor for use in application-specific designs being committed to that supplier's fabrication line. In other words, hard macros will most

likely not be generally portable across fabs except in cases where special business partnerships are established. The complexities of plugging a mask-level design for one process into another process line are a gating factor for further exploitation at present.
2. Firm macros — can be characterized as reusable parts that have designed down to the cell level and through partitioning and floorplanning. These are more flexible than hard macros, since they are not process-dependent and can be retargeted to other technology families for manufacture.
3. Soft macros — these are truly portable design descriptions but are only completed down through the logical design level. No technology mapping or physical design is available.

In order to achieve reuse at the IC level, it will be necessary to define or otherwise establish a number of standard interfaces for design data. The reason for this is that it will be necessary to fit design data from other sources into the design system being used to develop the IC. Therefore, VSIA has been established to determine where standard interfaces or data formats are necessary and to elect what the standard is to be. For soft-macros, these interfaces will need to include behavorial descriptions, simulation models, timing models, etc. For firm macros, the scope will additionally include cell libraries, floorplan information, and global wiring information. For hard macros, GDSII may be supplied.

In order to reuse designs that were developed elsewhere (hereafter called intellectual property blocks, or IP blocks), the first essential requirement is that the functional and electrical characteristics of the available IP blocks be available. Since internal construction of firm and hard IP may highly sensitive and proprietary, it will become more important to describe the functional and timing characteristics at the I/O's. This will drive the need for high-level description methods such as use of VHDL behavorial models, DCL, IBIS models, and dynamic timing-diagrams. Furthermore, the need to test these embedded IP blocks will mandate more use of scan-based test techniques such as JTAG Boundary Scan. Use of standards methods such as these to encapsulate IP blocks will be paramount for broad use across many designs and design systems.

There are risks, however, and reuse of IP will not cover all design needs. Grand schemes for reusable software yielded less-than-desired results. Extra effort to generalize the IP block design points for broad use, and describing its characteristics in standard formats is compounded with the ability to identify an IP block that fits a particular design need. Design and characterization of reusable IP will need to be robust in timing, power, reliability, and noise in addition to function and cost. Furthermore, the broad distribution of IP information may conflict with business objectives. Additionally, even if broadly successful, use of reusable IP will not negate the fundamental need for major EDA advances in the areas described. First, tools are needed to design the IP. Second, tools are needed to design the millions of circuits that will interconnect IP. Finally, tools and systems will require major advances to accurately design and analyse the electrical interactions between, over and under IP and application-specific design elements on the IC. Standards are essential, but they are not sufficient for success. Tools, rules, and systems will be the foundation for future IC design as they have been over the past. Nevertheless, the potential rewards are substantial, and there is an absence of any other clear EDA paradigm shift. Consequently, new standards will emerge for design and design systems, and new methods for characterization, distribution, and look-up of IP will become necessary.

78.4 Summary

Over the past three decades, EDA has become a critical science of many disciplines. Where it may have begun as a productivity enhancement, it is now a fundamental requirement for all electronic design. The criticality of EDA for IC design is the focus of much attention because of the expanding semiconductor advancements. Fundamental approaches in test, simulation, and physical design are constantly being tested by these semiconductor advancements. New EDA disciplines are opening-up. Fundamental electrical models are becoming more important and at the same time more complex. New problems in design and IC failure modes will surely surface.

The next decade of EDA should prove to be as exciting as the past three!

References

1. Case, P. W., Correia, M., Gianopulos, W., Heller, W. R., Ofek, H., Raymond, T. C., Simel, R. L., Stieglitz, C. B., Design automation in IBM, *IBM J. Res. Dev.,* 25(5), 631, 1981.
2. Roth, J. P., Diagnosis of automata failures: a calculus and a method, *IBM J. Res. Dev.,* 10, 278, 1966.
3. Eichelberger, E. B., Hazard detection in combinational and sequential switching circuits, *IBM J. Res. Dev.,* 9, 90, 1965.
4. Eichelberger, E. B. and Williams, T. W., A logic design structure for LSI testability, *Proc. 14th Design Automation Conf.,* 462, 1977.
5. *IEEE Std 1149.1-1990, IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE, 1990.
6. Hitchcock, R. B., Cellular wiring and the cellular modeling technique, *Proc. 6th Annu. Design Automation Conf.,* 25, 1969.
7. Lee, C. Y., An algorithm for path connection and its applications, *IRE Trans. Electron. Comput.,* 346, 1961.
8. Hightower, D. W., The intelrconnection problem, a tutorial, *Proc. 10th Annu. Design Automation Workshop,* 1, 1973.
9. *1076-1993 IEEE Standard VHDL Language Reference Manual* , IEEE, 1993.
10. *1364-1995 IEEE Standard Hardware Description Language Based on the Verilog® Hardware Description Language*, IEEE.
11. Parasch, G. J. and Price, R. L., Development and application of a designer oriented cyclic simulator, *Proc. 13th Annu. Design Automation Conf.,* 1976.
12. *Electronic Design Interchange Format (EDIF)* , Electronic Industry Associates.
13. Stok, L., Kung, D. S., Brand, D., Drumm, A. D., Sullivan, A. J., Reddy, L. N., Heiter, N., Geiger, D. J., Chao, H. H., and Osler, P. J., BooleDozer: logic synthesis for ASICs, *IBM J. Res. Dev.,* 40(4), 407, 1996.
14. Sayah, J. Y., Gupta, R., Sherlekar, D. D., Honsinger, P. S., Apte, J. M., Bollinger, S. W., Chen, H. H., DasGupta, S., Hseih, E. P., Huber, A. D., Hughes, E. J., Kurzum, Z. KM., Rao, V. B., Tabteing, T., Valijan, V., and Yang, D. Y., Design planning for high-performance ASICs, *IBM J. Res. Dev.,* 40(4), 431, 1996.
15. Barnes, T. J., Harrison, D., Newton, R. A., and Spickelmier, R. L., *Electronic CAD Frameworks*, Kluwer Academic Publishers, 1992, chap. 10.
16. Semiconductor Industry Association, *The National Technology Roadmap for Semiconductors, Technology Needs, 1997 Edition*, World Wide Web www.sematech.org, 13, 1997.
17. Bakoglu, H. B., *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, Reading, MA, 1990.
18. Goel, A. K., *High-Speed VLSI Interconnections, Modeling, Analysis & Simulation,* John Wiley & Sons, New York, 1994, chap. 2.
19. Johnson, H. W. and Graham, M., *High-Speed Digital Design, A Handbook of Black Magic*, Prentice-Hall, Englewood Cliffs, NJ, 1993, chap. 1.
20. Pandini, Davide, Scandolaro, Primo, Guardiani, Carlo, *Network Reduction for Crosstalk Analysis in Deep Submicron Technologies*, 1997 ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems, 280, 1997.
21. Cottrell, D. R., *Delay Calculation Standard and Deep Submicron Design*, Integrated Systems Design, 1996.
22. Fisher, P.D., *Clock Cycle Estimates for Future Microprocessor Generations*, IEEE 1997 ISIS: International Conference on Innovative Systems in Silicon, 1977.
23. Deutsh, A., Kopcsay, G. V., Restle, P., Katopis, G., Becker, W. D., Smith, H., Coteus, P. W., Surovic, C. W., Rubin, B. J., Dunne, R. P., Gallo, T., Jenkins, K. A., Terman, L. M., Dennard, R. H., Sai-Halasz, G. A., and Knebel, D. R., When are transmission-line effects important for on-chip interconnections, *1997 Electron. Components Technol. Conf.,* 704, 1997.

Yao, K. "Algorithms and Architectures for Multimedia and Beamforming Communications"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

79

Algorithms and Architectures for Multimedia and Beamforming in Communications

Flavio Lorenzelli
ST Microelectronics, Inc.

Kung Yao
University of California at Los Angeles

79.1 Introduction

In recent years, we have experienced many explosions in various technologies in information processing, transmission, distribution, and storage. In this chapter, we will address two distinct but equally important algorithms and architectures of multimedia and beamforming array processings in communication systems. The first problem is motivated by our need for efficient and real-time presentation of still image, live-action video, and audio information commonly called "multimedia." To most users, multimedia presentation is easier and more natural to comprehend compared with the traditional textual form of presentation on paper. The second problem is motivated by a desire to transfer tremendous amounts of information from one location to another over limited frequency-polarization-space-time channel constraints. Beamforming array processing technologies are used to coherently transmit or receive information over these channel constraints that can significantly improve the performance of a single transmit or receive antenna system. Since both of these problems are of fundamental interest in VLSI and practical implementations of modern communication systems, we consider in detail the basic signal processing algorithmic and architectural limitations of these problems.

In Secection 79.2, we first consider the intense computational requirements for displaying images and video using a general purpose PC. The pros and cons of using a separate processor or accelerator to enhance the main CPU are also introduced. Then, "Extended Instruction Set and Multimedia Support for General Purpose Processors" discusses issues related to the extended instruction set for media support

for a general purpose PC, while "Multimedia Processors and Accelerators" considers in some detail media processors and accelerators. "Architectures and Concepts for Multimedia Processors" deals with the architectures and concepts for multimedia processors. The very long instruction word (VLIW) architecture and the SIMD and associates subword parallelism issues are presented and compared. Four tables are used in Section 79.2 to summarize some of the involved basic operations.

In Section 79.3, we discuss the use of beamforming operation originally motivated by various military and aerospace applications, but more recently in many civilian applications. In "Interface Rejecting Beamforming Antenna Arrays", we consider some early simple, sidelobe-canceller, beamforming arrays based on the LMS adaptive criterion, then we discuss in some detail various aspects of a recursive least-squares criterion QR decomposition-based beamforming array. In "Smart Antenna Beamforming Arrays", we consider the motivation and evolution of the smart antenna system and its implication.

79.2 Multimedia Support for General Purpose Computers

In any computer store, it is fairly common nowadays to see rows of computer screens running video and audio clips. Voices, sounds, and moving pictures are attracting large numbers of users who now require as basic features software applications, which include voice mail and videoconferencing. Advertisements for home computers virtually never miss the buzz words "multimedia," "interactive," and "Internet." Over time, audio and video, as well as network connectivity, have become integral parts of many user applications. This change in users' expectations has been the cause for a major shift in the operations required of a general-purpose computer. The focus of computer designers and vendors has moved to plain word processing and spreadsheet applications to highly demanding tasks, such as real-time compression and decompression of audio and video streams. CPUs are now required to process large amounts of data fast, and allow for different processes to run simultaneously. For instance, one might have different windows open at the same time, one running a streaming video, another playing an audio file, another showing an open internet connection, etc. In addition, PCs are now expected to provide high-quality graphics, 3-D moving pictures, good quality sound, and full-motion support (e.g., MPEG-2 decoders, etc.). Most computer vendors and processor designers are therefore making huge efforts to alter or enrich their products so as to be able to provide these new multimedia services.

In order to better envision the complexity required by some multimedia tasks, consider what is involved in operations such as 3-D image rotation/zooming, etc. Each 3-D object consists of hundreds or even thousands of polygons, usually triangles. The smaller the polygons — and consequently the more numerous — the more detailed the object. When an object moves on the screen (rotates or moves forward/backward), the program must recalculate every vertex of every polygon by performing a number of matrix computations. A typical 3-D object might have 1000 polygons, which means that each time it moves, at least 84,000 multiplies and adds are executed. The new MPEG-4 audio/video standard (release date November 1998) addressed, among other things, the need for high interactive functionality, i.e., interactive access to and manipulation of audio-visual data, and content-based interactivity. The standard will provide an object-layered bit-stream. Each object can be separately decoded and manipulated (scaled, rotated, translated). Even 300-MHz Pentium II's cannot supply enough coordinates for more than a million polygons per second. Accelerator cards or graphics chips are often required for high-end, graphics intensive applications, to render polygons faster than any CPU can keep up.

One of the major drives to multimedia PCs was due to Microsoft's APIs, particularly the recent DirectX interfaces. With DirectX, it is easier to replace audio and video hardware while retaining software compatibility. However, as long as older applications remain alive, multimedia PCs will have to provide compatibility with existing register level standards, e.g., superVGA and SoundBlaster. Register-level interfaces were not designed to support more advanced audio/video features, e.g., 3-D graphics. Most game programs avoid old APIs because of their poor performance. They run directly under DOS and access audio and video chips directly. Microsoft created DirectX, a new set of APIs, in order to avoid having to modify existing programs to support new graphics devices as they show up on the market.

Multimedia is likely to reach the electronics consumer market also in ways that are still unclear, for instance, in what has been dubbed home entertainment. Vendors envision gamuts of devices which combine the functions of a PC with traditional TV sets. PCs with dual-purpose monitors and set-top boxes will soon offer functionalities ranging from DVD to videophone and interactive 3-D gaming. The PC might become the basis for a living room entertainment center which includes a TV tuner, DVD, Dolby audio, 3-D graphics, etc.

Different companies have chosen different approaches to multimedia and made widely different decisions. Some companies, notably Intel and Sun Microsystems among others, have added circuitry to their new generation processor and enhanced their instruction sets in order to handle multimedia data. Other companies, e.g., Philips and Chromatics, have preferred to build whole new processors from the ground up which could work either independently or in alliance with another processor, solely in charge of audio, video, and network connectivity. There are also those who have been thinking of thoroughly new architectures and concepts, which could open entirely different perspectives and computation paradigms. In the following, we will provide a (necessarily incomplete and temporary) snapshot of the situation as it appears at the end of the 1990s, when the authors are writing.

The solutions pursued by the various companies display tradeoffs. What will be the most successful approach is open to debate and will probably be understood only in hindsight. Nonetheless, some considerations can be made now.

The main drawback of external multimedia accelerators is undoubtedly the cost.[15] The home PC market has been very sensitive to cost, and most vendors are unwilling to add expensive features which could deter possible customers and therefore reduce their market penetration. On the other hand, software developers are unlikely to generate applications tailored for multimedia processors and accelerators unless there is a sufficiently large installed based.

The alternative to a separate processor or accelerator is to enhance the main CPU to allow it to process multimedia data.[23] As of today, there are few CPUs which have enough "horsepower" to simultaneously handle the operating system, all standard applications, as well as audio decoding and mixing, video decoding, 2-D and 3-D graphics, and modem functions. A general-purpose CPU like one based on a Pentium processor Pentium is simply unable to handle digitized audio samples and pixel data. Moreover, the multimedia additions translate into larger chips with lower yield. On the other hand, if the CPU is enhanced for multimedia, no optional hardware is required to perform the multimedia tasks and the installed base can grow naturally, so to speak. Software developers will be more willing to write applications for established platforms (e.g., Pentium), whereas software vendors can target these systems with higher sales expectations. The additional hardware required for multimedia enhancement is usually only a small fraction of chip space, with minimal added cost to the overall system. As processors become faster and smaller, eventually this is the solution that most foresee prevailing, at least in low- to mid-range systems. Already, existing multimedia enhanced systems show great promise, especially when compared with the cost and performance of add-in chips and boards. What can be said in favor of separate graphics' subsystems is that the same money spent on a better graphics accelerator often buys much larger effective performance improvement (on multimedia applications) than one would get by upgrading the CPU. Users may be more willing to spend money on a system with greater multimedia performance than in a faster number-crunching CPU. In fact, graphics' subsystems have accounted for an increased percentage of the total cost of multimedia PCs, often at the expense of the central processor. Faster CPUs are still unavoidable where FP geometry calculations and rendering performance are crucial, but geometry calculations are being included in graphics chips as well.

Extended Instruction Set and Multimedia Support for General Purpose Processors

The approach of processing the multimedia data on the CPU itself has been named native signal processing (NSP). This is the approach favored by Intel, Sun, Motorola, Hewlett-Packard, and the makers of the PowerPC line. The rationale is that processors soon will be sufficiently powerful to perform tasks

such as real-time video compression and decompression, video teleconferencing, etc., without the aid of any add-in chips or cards. Tomorrow's host CPUs will be able to execute in software most if not all the tasks that today are handled by media processors and accelerators. Certainly, media processors' performance will increase over time just as much as regular CPUs. The question, though, is what is required of a media processor? Multimedia is by definition directly related to human perception. Once a multimedia algorithm reaches the limit of people's perception, there is little reason to make any further improvements. Of course, users always will be asking for more features, simultaneously running applications in real-time, etc., but it seems reasonable that eventually the drive for performance increase will level off. Both audio (including 3-D sound, up- and down-sampling, 32-channel mixing, Dolby ac-3) and 2-D graphics (GUI acceleration) have already reached the limits of human perception on any high-end Pentium system. Algorithms that still need a good deal of improvement are 3-D graphics and video compression (although in the latter, bandwidth is the key, not human perception). CPUs may soon catch up where multimedia algorithms still have not reached their limits. Intel's 233-MHz Klamath is able to execute full DVD decoding, including Dolby ac-3 audio and MPEG-2 video, and future processors will only do better. In the past, the market for hardware accelerators has been destroyed by software decoders that run on any fast CPU, i.e., MPEG-1 decoders.

NSP requires the expansion of the instruction set architecture (ISA), as well as the possible modification of the existing CPU. The additional circuitry for multimedia data processing takes advantage of the fact that most multimedia applications require calculations in lower precision (16- or 32-bit) than provided by ALUs used for standard integer and floating point (FP) data. This allows for the generation of two to eight results per clock cycle, thereby offering sufficient data rate for intensive multimedia applications. Fast computation may require additional optimized hardware: faster and possibly split buses, parallel computational units or superpipeline, larger L1 caches, small misprediction penalty, faster CPUs at lower voltage supply, etc. As an example, Cyrix has offered a chip, MediaGX,[16] that delivers Pentium-class performance and software compatibility, while adding an integrated memory controller, graphics accelerator, and a PCI interface. MediaGX improves performance by taking advantage of the tight coupling between CPU and system logic. The effects of high bandwidth demands are minimized by use of advanced compression techniques, which reduce the size of the frame buffer when stored in memory. By this technique, up to 95% of what the frame buffer reads from memory can be eliminated. MediaGX also provides full compatibility with VGA graphics and SoundBlaster audio. Later generations of this chip also include MMX support and acceleration feature for MPEG-1 video, such as pixel scaling and color scale conversion.

Intel's MMX

Intel improved its architecture's multimedia performance by extending the x86 instruction set.[13] The 57 new instructions, see Table 79.1, are referred to as MMX and are devised to accelerate calculations common in audio, 2-D and 3-D graphics, video, speech synthesis and recognition, and data communication. The overall performance improvement is estimated to be 50 to 100% for most multimedia applications (e.g., MPEG-1 decoding, pixel manipulation). Other companies, such as AMD and Cyrix, have incorporated Intel's MMX. MMX obviously has no impact on the operating system. Applications can take advantage of MMX in either of two ways: by scaling MMS-enabled drivers, or by adding MMX instructions to critical routines. One advantage of relying on drivers is that an application can automatically take advantage of a hardware accelerator for 3-D graphics, sound, or MPEG decoding if one is installed.

MMX instructions can be used in any processor mode and at any privilege level. They generate no new interrupts or exceptions. The eight new MMX registers are mapped onto the existing floating point registers, so that programs cannot use MMX and FP instructions simultaneously in the same routines. For 3-D graphics, MMX is typically used for 3-D rendering routines, while geometry calculations remain in FP.

The new instructions use a SIMD model, i.e., the same instruction operates on many values (eight bytes, four words, or two double words). On a two-way superscalar machine such as a P55C,[19] MMX

instructions can be paired with integer instructions or other MMX instructions as long as they do not use the same functional units.

One drawback of MMX is the lack of a multiply or a multiply-add for 32-bit operands; this choice is probably due to real-estate considerations. This drawback prevents routines such as 3-D geometry calculations and wavetable sound from taking advantage of MMX.

The problem of managing separate versions of each application for MMX and non-MMX systems can be solved by checking bit 23 of the CPUID to determine whether a processor implements MMX. The check can be performed via software.

While MMX can boost multimedia performance of low-end systems with no accelerators, it will probably eliminate media processors and accelerator chips from low- and mid-range systems as processor speed increases.

Intel has announced another extension, KNI, which will consist of 70 new instructions that emphasize parallel FP computations. In the meantime, AMD, Cyrix, and IDT have formally introduced a new x86 instruction set extension for speeding 3-D graphics applications. Their new set is dubbed 3DNow![4] and consists of instructions that pack two 32-bit FP values into a single MMX register and compute two results in parallel. 3DNow! provides basic add, subtract, divide, multiply, and square-root operations. The 3Dnow! single-precision FP format is IEEE-754 compatible, but results computed by 3DNow! do not comply with the IEEE standard. In particular, not all four rounding modes dictated by the IEEE standard are supported. When KNI becomes available in Intel's chips, 3DNow! is likely to be soon forgotten, since no software developer can justify supporting 3DNow! instead of KNI.

Other IS Extensions

Companies other than Intel have proposed their instruction set extensions. Sun proposed VIS for its UltraSparc. Similar to MMX, VIS includes 8-, 16-, and 32-bit parallel operations, uses FP registers, and performs saturating and unsaturating arithmetic. VIS adds some specialized instructions, which accelerate MPEG motion estimation, and video compression algorithms (MPEG-1). Some instructions are included to accelerate the discrete Fourier transform (DCT), pixel masking, and 3-D rendering. VIS instructions are encoded as three-operand and operate on 32 registers.

The PowerPC IS architecture is extended by the AltiVec instructions, which in many respects goes far beyond its 3DNow! counterparts. AltiVec offers more specialized instructions, such as 2-to-the-power, base-2 logarithm estimation, data permutation, various multiply-accumulate. Overflow and other events are more faithful to the IEEE standards than 3DNow!, a feature that helps meet requirements of advanced audio applications. The number of dedicated registers is 32.

Multimedia Processors and Accelerators

Media processors are whole new processors that can handle multimedia data efficiently.[9,12,14] By definition, they are programmable processors which can simultaneously accelerate the processing of multiple data types, including digital video, digital audio, computer animation, text, and graphics. They are required to offer high performance on these tasks, while the CPU is allowed to concentrate on other applications, be they spreadsheets or word processors, uninterrupted. Media processors combine the advantages of hardwired solutions, such as low cost and high performance, with the flexibility of software implementations. They also eliminate the need for a number of different audio/video subsystems (3-D acceleration, MPEG playback, wavetable audio, etc.)

There is no unique way to build a media processor.[26] The first ones to reach the market (NVidia NV1 and Philips TriMedia, TM-1) would combine powerful DSPs for 2-D and 3-D acceleration, audio (channels of CD-quality sound), possibly MPEG decoding, and broadband demodulation. Philips TM-1 has also units for video preprocessing (variable length decoding and image scaling and color space conversion). Other makers, notably Chromatics with its MPact Media Engine, preferred a single powerful processor that performs both audio and video tasks and telephony.[20] MPact2 also adds DVD support,

TABLE 79.1 The MMX Opcodes

Group	Mnemonic	Description
Data Transfer, Pack, Unpack	MOV[D,Q]	Move [double/quad] to/from MMX reg
	PACKUSWB	Pack W → B with unsigned sat
	PACKSS[WB,DW]	Pack W → B, D → W w/signed sat
	PUNPCKH[BW,DW,DQ]	Unpack high-order [B,W,D] from MMX reg
	PUNPCKL[BW,DW,DQ]	Unpack low-order [B,W,D] from MMX reg
Arithmetic	PADD[B,W,D]	Packed add on [B,W,D]
	PADDS[B,W]	Saturating add on [B,W]
	PADDUS[B,W]	Unsigned saturating add on [B,W]
	PSUB[B,W,D]	Packed subtraction on [B,W,D]
	PSUBS[B,W]	Saturating sub. on [B,W]
	PSUBUS[B,W]	Unsigned sat. sub. on [B,W]
	PMULHW	Multiply packed words, get high bits
	PMULLW	Multiply packed words, get low bits
	PMADDWD	Multiply packed words, add pairs of products
Shift	PSLL[W,D,Q]	Packed shift, left, logical
	PSRL[W,D,Q]	Packed shift, right, logical
	PSRA[W,D]	Packed shift, right, arithmetical
Logical	PAND	Bitwise AND
	PANDN	Bitwise AND NOT
	POR	Bitwise OR
	PXOR	Bitwise XOR
Compare	PCMPEQ[B,W,D]	Packed compare if equal
	PCMPGT[B,W,D]	Packed compare if greater than
Misc	EMMS	Empty MMX state

Brackets indicate options.
B = byte.
W = word.
D = double word.
Q = quad word.

videophone operation, as well as MMX support to reduce host processor overhead.[25] The specifics regarding clock speed, core performance, etc., are given in Tables 79.2 and 79.3.

Most media processors use VLIW and SIMD techniques to achieve high performance. These techniques are summarized in a later section. Among the issues that designers have to face are the following:

- The load balance between the media processor and the host CPU. Some of the media processors require significant preprocessing on the host CPU, which can cause a noticeable slowing of the system. For instance, MPEG-1 decoding with a high resolution display can consume as much as half the processing power of a 100-MHz Pentium when Chromatics MPact1 is used.
- Compatibility with legacy code, such as SoundBlaster and VGA emulation. Some makers guarantee some kind of compatibility, while others (e.g., Philips) do not.
- Software flexibility. Some designs are reprogrammable and thus have the advantage of being upgradable when new audio, video and graphics standards change and improve.
- Connections to outside devices. Media processor should also provide connection to PCI, external DAC, and audio codec.

The software development for the different media processors usually depends on both makers and outside vendors (with the notable exception of Chromatics, which did not make it possible for anyone else to develop software for MPact). Makers usually offer compilers and code libraries for the most common multimedia algorithms, while outside partners and OEMs develop custom software. All vendors of media processors are more or less at Microsoft's mercy for the APIs needed to run their processors under Windows.

Not all media accelerators are so ambitious as to target the complete multimedia market. Indeed, recently there has been a trend where media processors have started to specialize. Many companies have narrowed their scope to specific tasks. To name a few, C-Cube's Video RISC processor performs variable-length coding, data compression estimation, and motion estimation for use in MPEG-1 and MPEG-2 encoding by digital satellite TV broadcasters; Mitsubishi's D30V includes audio and video circuitry and a variable-length decoder; Fujitsu's multimedia assist (MMA) integrates a graphics controller and audio interfaces and can be applied to DVD; Rendition's Vérité V1000 is a 32-bit RISC processor acting as a programmable 3-D accelerator, used for antialiazing algorithms and special effects; NVidia's NV3, succeeding the unsuccessful NV1, does not support audio and programmability in favor of 3-D performance. Many makers have included special DVD playback support (Chromatic's MPact2, Digital's SA-1500, as well as Mitsubishi's D30V). Philips has focused on videoconferencing systems, which include echo cancellation, voice-tracking camera control, Web server, GUI programs.

TABLE 79.2 Comparison of Some Media Processors

	NVidia NV1	Chromatic MPact R/3000	Chromatic MPact 2/6000
Clockspeed (MHz)	50	62.5	125
Peak Int. Perf. (GOPS)	0.35	3.0	6.0
Peak FP perf. (GFLOPS)	n/a	n/a	0.5
Memory Bandwidth (MB/s)	100-200	500	1200
3-D Geometry (Mpolys/s)	n/a	n/a	1.0
3-D Setup (Mpolys/s)	n/a	n/a	1.2
3-D Fill rate (Mpel/s)	n/a	5	42
2-D Acceleration	Yes	Yes	Yes
MPEG-1 decode	No	Yes (S/W)	Yes (S/W)
MPEG-1 encode	No	Yes (H/W)	Yes (H/W)
MPEG-2 decode	No	Yes (S/W)	Yes (S/W)
Videoconferencing	No	Yes	Yes
Telephony	No	Yes	Yes
	Philips TM-1	**Philips TM-PC**	**Samsung SMP-1G**
Clockspeed (MHz)	100	100	160
Peak Int. Perf. (GOPS)	3.8	3.8	10.2
Peak FP perf. (GFLOPS)	0.5	0.5	1.6
Memory Bandwidth (MB/s)	400	400	1280
3-D Geometry (Mpolys/s)	0.75	0.75	0.75
3-D Setup (Mpolys/s)	1.0	1.0	0.75
3-D Fill rate (Mpel/s)	n/a	n/a	n/a
2-D Acceleration	No	Yes	Yes
MPEG-1 decode	Yes (S/W)	Yes (S/W)	Yes (S/W)
MPEG-1 encode	Yes (S/W)	Yes (S/W)	Yes (S/W)
MPEG-2 decode	Yes (S/W)	Yes (S/W)	Yes (S/W)
Videoconferencing	Yes (S/W)	Yes (S/W)	Yes (S/W)
Telephony	Yes	Yes	Yes

Recently, a number of companies have announced 3-D chips, among others 3DLabs, ATI, Number Nine, NVidia, S3.[10,11] 3-D chips have to face two main issues to provide the required performance levels: computational resources and memory bandwidth. The calculations necessary for 3-D consist of scene management (preparation of a 3-D object database along with information on light source and virtual camera), geometry calculations (transforms and setup), and rendering (shading and texturing applied to the pixels) (see Table 79.4). The host processor is usually in charge of scene definition, whereas rendering is handled by 3-D graphics hardware. As for geometry calculations, they are usually split equally

TABLE 79.3 Comparison of Some Media Processors (Continued.)

	Fijitsu MMA	Mitsubishi D30V	Rendition Vérité
Clockspeed (MHz)	180	250	50
Peak Int. Perf. (GOPS)	1.08	1.0	0.1
Peak FP perf. (GFLOPS)	n/a	n/a	n/a
Memory Bandwidth (MB/s)	720	n/a	400
3-D Geometry (Mpolys/s)	n/a	n/a	n/a
3-D Setup (Mpolys/s)	n/a	n/a	0.15
3-D Fill rate (Mpel/s)	n/a	n/a	25
2-D Acceleration	n/a	n/a	Yes (H/W)
MPEG-1 decode	Yes (S/W)	Yes (S/W)	No
MPEG-1 encode	No	No	No
MPEG-2 decode	Yes (S/W)	Yes (S/W)	No
Videoconf. Teleph.	n/a	n/a	n/a

between host processor (transforms) and graphics chips (setup). Thanks to the evolution of 3-D chips, the host processor is often relieved of many of the tasks. The result is higher polygon throughput and better visual quality. The performance that all makers have tried to reach, and exceed, is that of a million polygons per second. The increased bandwidth demands are met by on-chip texture caches and faster, wider local memory arrays in the graphics subsystems. Memory bandwidth demands can be reduced by applying texture compression, usually based on vector quantization algorithms for still images. The bandwidth problem will also require larger internal SRAM arrays for texture caches and embedded DRAM for on-chip frame buffers. Off-chip memory will still be necessary for larger frame buffers. On-chip setup engine and motion compensation logic may be added, as in ATI's RagePro design, to accelerate MPEG-2 decoding and DVD playback.

TABLE 79.4 Basic 3-D Pipeline: Scene → Geometry → Rendering

Scene	Geometry	Rendering
Database of 30D Objects	Projection	Shading
Scene	Clipping	Texturing
Location of Light Source	Slope Calculation	Remove Invisible Pixels
Position of Virtual Camera	Rasterizing	

Architectures and Concepts for Multimedia Processors

Many, if not all, multimedia applications are required to perform very computationally demanding computations. Custom architectures that support digital signal processing can satisfy the computational needs of these applications, but they are usually lacking the flexibility required in an environment where standards and algorithms change continuously. Programmable devices, such as traditional DSPs, in turn lack computing power and bandwidth necessary to perform more than a multimedia task at a time. The media processors narrow the gap between DSPs and general-purpose processors by extending the instruction-level parallelism (ILP) of traditional DSPs to a general, very long instruction word (VLIW) architecture.[24] Analogously to superscalar processors, VLIW media processors rely on hardware dispatch units to dynamically schedule the operations and evaluate data dependences, VLIW architectures rely on the compiler to statically schedule instructions at compile time. Most media processors also take advantage of SIMD concepts in order to improve computational throughput. Both VLIW and SIMD concepts are reviewed in the following.[17]

The VLIW Archictecture

Very Long Instruction Word (VLIW) architectures[3,5,7,8] are one of two categories of multiple-issue processors, the other one being referred to as superscalar architecture. Superscalar architectures issue more

than one instruction per clock, which can be either statically scheduled at compile time or dynamically scheduled using various scoreboarding techniques or Tomasulo's algorithm. VLIWs in contrast are statically scheduled by the compiler, and fixed-size instruction packets are issued at each clock cycle.

In a superscalar processor[18] the functional units are structured so that a number of instructions, typically one to eight, can be scheduled simultaneously per clock tick. An example is an architecture where one integer and one floating point instruction can be issued together. A sufficient number of read and write ports have to be provided to avoid structural hazards. Of course, simultaneously issued instructions cannot be interdependent. Moreover, no more than one memory reference can be issued per clock. Other restrictions come from latencies inherent in memory accesses or branches, e.g., it may be required that the result of a load instruction may not be used before a number of clock cycles have elapsed. When dynamic scheduling is employed, a superscalar processor requires dedicated hardware to provide for hazard detection, reservation tables, queues for loads and stores, and possibly an out-of-order scheme. Whenever the behavior is unknown or unpredictable, the architecture has to allow for conditional or predicated instructions, which can eliminate the need for certain branches altogether, and the use of speculation, whereby an instruction can be issued before knowing whether the instruction should really execute. A number of additional registers is typically required to support speculative execution and register renaming. Additionally, bits may have to be added to registers and instructions to indicate whether the instruction is speculative. In summary, a superscalar architecture may require a significant amount of additional hardware with respect to a single-issue processor, with the consequent cost of a larger die area and complex circuitry. The advantage is a low CPI (clock cycles per instruction), obtained with good efficiency (for instance, hardware-based branch prediction is usually superior to software-based prediction done at compile time). Another great advantage of superscalar architectures is compatibility to legacy code: unscheduled programs or those compiled for single-issue processors can be run on a superscalar processor.

VLIWs use multiple independent functional units. Instructions are packed into a very long instruction word. The burden for ordering the instructions and packing them together falls on the compiler, and no hardware is needed to make dynamic, run-time, scheduling and decisions. In order to keep all functional units as busy as possible, there must be enough parallelism in the application. Not all inherent parallelism is immediately available. Some has to be extracted by various software techniques, including loop unrolling and software pipelining.[1,2,21] Instructions may have to be moved even across branch boundaries, always maintaining the correct dependences among data. Some additional hardware is required also of a VLIW processor due to the increase in memory bandwidth and register file bandwidth. As the number of data ports increases, so does the complexity of the memory system. The VLIW compiler is required to perform very aggressive operations on the original code, and it generates an output code that is usually much larger than in traditional processors, e.g., due to loop unrolling. Memory size can be reduced by use of compression techniques in main memory and/or in cache. Object code compatibility is obviously the main drawback. For many types of applications, one cannot guarantee that all functional units will be used efficiently, depending on the available instruction level parallelism (ILP) of the application itself. Moreover, branch mispredictions can cause significant performance loss.[27]

Multimedia applications have characteristics which make them more suitable for VLIW processors. DSP applications as well as multimedia, display abundant amounts of available ILP, thanks to the highly regular structure of the algorithms implemented, i.e., matrix-vector manipulations or discrete cosine transforms. Not surprisingly, DSP architectures were among the first programmable devices to rely on long instruction words (LIWs) to improve parallelism.[22] Signal processing applications are characterized by

- large numbers of combination of instructions, e.g., the pair of instructions multiply and accumulate (very commonly used in filtering and linear transform algorithms)
- low-overhead counted loops, which can be supported by simple decrement-and-branch instructions
- multiple accesses to memory in a single cycle. Memory which supports multiple accesses is usually partitioned or interleaved.

Typical DSP and multimedia applications include parallel and independent computations, e.g., computations of different pixel values, which allow one to extract considerable ILP. Among the most common algorithms that may be encountered in these applications are filtering (FIR, IIR); frequency transforms (Fourier, DCT), are pixel computations.

DSP hardware has been hard to use and program through libraries and hand-tailored code. The principle was for DSPs and is for VLIWs that the processor provides some parallelism and the programmer (or compiler) must provide code that matches as much as possible the CPU's ILP. The first VLIWs were built with the intention of bridging the gap between general-purpose processors and DSPs, keeping in mind that compiler technology has to be as efficient as possible in uncovering the application's ILP. VLIWs now combine the cost/performance ratio of DSP processors with the reprogrammability of general-purpose processors. The hardware can provide parallelism in any of the following ways:

- by allowing operations to execute simultaneously on all functional units; sufficient memory and register level bandwidth are required to reduce potential structural hazards conflicts
- by replicating functional units, e.g., by having two integer units
- by pipelining longer operations, e.g., FP operations, so that one can be initiated at every clock cycle

The application's ILP can be uncovered by using any or all the following techniques:

- *Trace scheduling*. The major barrier to exposing ILP was long represented by the so-called "basic blocks," i.e., the single straight line segments of code between branches or jumps.[6] Compilers used to schedule operations of basic blocks and to stop any scheduling when it would encounter a jump or branch. The most ILP that can be gained by basic block scheduling is approximately a factor of two. More ILP can be exposed if instructions are allowed to be moved across basic blocks boundaries. Trace scheduling is based on loop-free sequences of basic blocks, i.e., paths through the program that could conceivably be taken by some set of input data. Traces are selected and scheduled according to their frequency of execution, i.e., according to how likely it is they are taken. In this scenario, the compiler can freely move operations between basic blocks, irrespective of branches. Scheduling proceeds from most to least frequently executed traces, until all the program is scheduled. Instructions can be moved only if all data dependences are guaranteed to remain unchanged. In general it may not be possible to determine all data dependences at compile time (it is actually an NP-complete problem). The compiler, with no other indications from the programmer, has to be conservative in its decisions. Trace scheduling can be successfully applied in conjunction with software pipelining and loop unrolling.[1]
- *Loop unrolling*. This is done by simply replicating the loop body a given number of times, erasing useless intermediate checks, and adjusting the loop termination code. Instructions from different loop iterations can now be scheduled together.
- *Software pipelining*. This technique reorganizes loops so that each iteration in the software-pipelined code is chosen from different iterations of the original loop. In a sense, instructions from different iterations are interleaved with one another. The loop is thus running at maximum overlap or parallelism among instructions, except for start-up and a wind-down overheads. Often it is necessary to combine software pipelining and loop unrolling, depending on the specific application.

SIMD and Subword Parallelism

Most media processors combine the VLIW architecture with another form of parallelism, usually referred to as single-interaction, multiple-data (SIMD), or subword parallelism. SIMD instructions affect multiple pieces of data, compacted in a longer word. A single SIMD instruction has the same effect as many identical instructions on as many data items. ILP can be exploited by packing many lower precision data into a large container. The ALU is set up to perform identical operations on every single subword of the input. The additional hardware required to implement SIMD operations is

- more control logic for the ALU to perform parallel operations
- more opcodes and the corresponding decoding logic

- packing, unpacking, and alignment circuitry

SIMD uses most of the existing ALU architecture to execute multiple instructions in parallel. Moreover, it uses the same register file port to read and write more distinct values. On the other hand, packing and unpacking penalties have to be paid when single items need to be accessed individually. In addition, SIMD instructions require rewriting of the application to identify the ILP and exploit it with appropriate library functions that use SIMD instructions. Many media processors try to take advantage of both approaches by implementing a VLIW architecture with some functional units supporting packed (SIMD) arithmetic.

Architecture Issues

A VLIW/SIMD processor has specific architecture characteristics, different from traditional processors.

- A larger number of registers helps when large amounts of ILP are exposed. Many registers reduce memory traffic. Moreover, the following techniques take advantage of a higher number of registers:
  - Speculative execution: speculative results must be kept until committed or discarded.
  - Loop unrolling and software pipelining: *N* iterations of the same loop, unrolled or interleaved, require more registers.
  - Predication: values computed along both paths of a branch must be kept until the branch is resolved.

  These effects are mitigated by SIMD, where a single register contains many smaller precision values.
- VLIW requires a higher number of data ports and a higher bandwidth. Data ports are costly additions because
  - the area of the register grows quadratically with the number of ports, and
  - the read access time of a register file grows linearly with the number of ports.

  Again, a SIMD approach reduces the need for ports, since a single register may contain several data items.
- The register file has to be appropriately partitioned into separate register banks. Hardware must provide for moving date from bank to bank, when needed.
- SIMD operations must be supported in the following ways:
  - The hardware must support to address individual subwords of a register. Parts of a register may be read or written without affecting the rest.
  - The hardware must support to execute lower precision operations on subwords.
  - Parts of a register may have to be shifted and realigned.
- The memory architecture has to be organized in such a way as to reduce latencies to a minimum. General-purpose processors make large use of data caches, which are very effective for locally or temporally localized data whose access pattern and size are unpredictable. Among data memory accesses, there are two categories that can be classified as follows:
  - Memory accesses with spatial locality, but little temporal locality, e.g., regular access to data structures, matrices, and vectors, which are usually traversed only once sequentially.
  - Memory accesses with temporal locality, but little spatial locality, e.g., loop-up tables, interpolation tables, etc., which are repeatedly accessed in a data-dependent, nonsequential fashion.

  Caches are usually characterized by miss penalties which are at least one order of magnitude higher than cache hits.

  DSP applications display widely different data locality characteristics. Many data streams, such as audio and video, have a high degree of spatial locality, whereas other data accesses have temporal locality. Typically large data structures with only spatial locality tend to disrupt the data items that could benefit from temporal locality. For DSP applications, the large mismatch between miss and hit penalty makes the worst-case performance unacceptable for most applications, especially if

there are real-time requirements. For these reasons VLIW, analogously to DSPs, usually makes use of local memories.

Local memories are mapped to an address space that is separate from main memory. The use of local memories requires explicit compiler support or programmer intervention in order to guarantee known and constant access time.

The programmer may help the compiler manage memory accesses by use of either of the following techniques:

– platform specific language extensions not supported by the native language, or
– compiler annotations, i.e., annotations that can be safely ignored by the compiler, but, if used, help identify which data can be accessed in local memory. These annotations can be used in high-level languages and do not affect code compatibility.

Local memories are of little use when large data structures are used a few times at most and are traversed sequentially. For these cases, it is best to use

– fast access, off-chip memory, either expensive SRAM or slower DRAM; or
– a pre-fetch butter, which works very well when the access pattern is highly predictable.

- Most DSP algorithms spend a large amount of time in simple loops, which are usually repeated a constant and predictable number of times and do not require complex control logic. Traditionally, DSPs implement very efficient and simple ways of handling the control flow of simple loops.
- VLIW code can be quite large because of the following:
  – Compilation techniques make use of loop unrolling, software pipelining, etc., which cause instructions to be replicated to allow code motion across basic block boundaries.
  – Not all VLIW instructions can be filled with useful instructions because of data dependences and because not all functional units can be continuously busy. This means that many of the VLIW instruction fields are bound to be no-operation (NOP).

  Because of the aforementioned, it is common to compress the VLIW code in order to reduce the storage space. Due to the sparse nature of VLIW instructions, there usually is room for compression. Code is kept compressed in main memory and is decompressed before execution. There are two main approaches to compression:
  – The instruction cache contains uncompressed code. In this case, decompression is not on the critical path and is crucial only during cache misses. The cache is less effective, since it must contain uncompressed instructions. Moreover, the cache cannot have synchronized instruction addresses with main memory because main memory stores variable-length instructions while the I-cache stores fixed-size instructions. The decoding scheme must therefore be more complex.
  – Main memory and the I-cache are both compressed. The cache utilization is optimal, and the address translation mechanism from main memory is simple. The disadvantage is due to having decompression on the critical path, thus affecting the hardware cost.

79.3 Beamforming Array Processing and Architecture

Beamforming array concepts were originally formulated over 40 years ago to tackle demanding defense-oriented aerospace and underwater sonar, radar, and communication system problems. These problems combine signal processing with antenna technologies. They involve the use of phase coherent transmission or reception, utilizing array sensor selectively to enhance the gain in some space-polarization-time-frequency domain to reject intentional or unintentional jamming. In recent years, there has been an explosion in civilian mobile communication usages, mainly in the form of the large number of cellular telephone users competing over a limited number of frequency bands. Proper use of beamforming arrays can increase the number of users as well as find the user's cell phone under an emergency 911 condition. The various proposed array processing systems need the availability of modern digital signal processors. These processors start from the high-end programmable DSP, the FPGA, to custom single and wafer-

integration VLSI chips. As the cost of these processors decreases rapidly while their capability increases dramatically, "smart antenna" beamforming array techniques are under serious consideration for practical communication system implementations.

Interference Rejecting Beamforming Antenna Arrays

In order to understand the rationale for beamforming antenna arrays, we briefly consider various possible types of antenna. An omnidirectional (also called isotropic) antenna has equal gains in all directions. A directional antenna has more gain at certain directions relative to other directions. Thus, it needs to be moved mechanically to point at the direction of interest for either transmission or reception. A phased antenna array uses signals from simple (and possibly omnidirectional) antennas combined appropriately to achieve a high gain in some desired direction. The direction of maximum gain is adjustable by controlling the phases among the antennas so that the voltages are coherently added in phase. An adaptive antenna array is a phased antenna array in which the gain and the phase of each antenna may be changed as a function of time depending on external conditions. As an example, a receiving adaptive antenna array not only directs a high gain toward a desired transmitter, but it may also adaptively place spatial nulls (or low gains) toward other moving co-channel interfering sources operating at the same frequency band. An antenna array is said to be optimal if it adjusts the gains and phases of the antenna elements to optimize the array performance. Typical performance of interest may be maximizing the signal-to-noise-ratio (SNR) or signal-to-interference-and-noise ratio (SINR) of the array.

A plot of the gain vs. the angle of an antenna array is called the beam pattern. The process in which signals from different antennas in an array are added coherently is called beamforming. The steering of the direction of maximum gain of an array can be achieved by mechanical means or by changing the gains and phases of the antennas to achieve electronic beam steering. Beam steering for a narrow band array only need phase-shifters, but for a wide band array, both the gain and the phase of each antenna element (typically implemented in the form of a FIR filter) need to be used. For a narrow band array, coaxial cable of different lengths can be used for phase-shifting. By switching different cable lengths, beam switching can be realized.

The earliest adaptive antenna array system is the sidelobe canceller (SLC) motivated by practical radar-jamming problems. This system consists of a high gain mainbeam dish antenna surrounded by few auxilliary low gain omnidirectional antennas. A strong jammer appears on all the auxilliary and mainbeam antennas. By using appropriate complex weights on all the antennas, the output of the auxilliary antennas consisting of the jamming signal is coherently substracted from the sidelobe of the main antenna, thus permitting desired weak signal at the mainbeam to be detected. This scheme, often called the Howells-Applebaum algorithm,[28] was first proposed in the 1960s and was implemented using analog components. It turns out the Widrow-Hoff algorithm[29] for adaptive filtering, conceived independently and at about the same time, is essentially equivalent but has broader applications. It is called the least-mean-square (LMS) algorithm. It is an approximate steepest-descent gradient search algorithm. As such, the convergence time may be slow if the system eigenvalue spread is large. However, due to its relative simplicity, most simple adaptive antenna systems are still based on variations of the LMS algorithm.

A more complex but also faster convergent adaptive antenna array is to use the least-squares (LS) method for solving the weights W of the linear system of equations $XW \approx d$. Here X is a $N \times M$ matrix with each row representing an antenna element input and M represents the time index, d is a $d \times 1$ known steering vector, and W is a $M \times 1$ unknown weights needed in the adaptive system. Various QR decomposition (QRD) methods, such as the Gram-Schmidt, Modified-Gram-Schmidt, Givens, and Householder algorithms, all can be used to solve the LS problem. In practical complex antenna systems, N can be in the low hundreds, and M can be in the thousands. Clearly, direct brute force block LS solution is not feasible. A simple form of parallel processing called systolic processing, using only small number of processing elements (PE) having adjacent neighbor connects, and operating in a synchronous manner, has been proposed to solve the QRD approach to the recursive least-squares (RLS) problem. This approach was originated by McWhirter[30] using the Givens algorithm, and variations have been proposed by Ling-

Proakis,[31] Kalson-Yao,[32] Liu-Yao,[33] and others. Issue related to the complexity of the algorithms and the practical VLSI implementation of such an array have been considered by Rader,[34] Bolstad-Neeld,[35] and Lodhtbody et al.[36]

Smart Antenna Beamforming Arrays

The explosion of cellular telephony and wireless communication services has motivated the consideration of various technologies to increase their efficiency. These systems, in order to support large number of users with increasing data rate demands and operating in difficult propagation conditions, have found "smart antenna" to be highly useful. The term "smart antenna" commonly denotes the use of a RF antenna array system with advanced signal processing techniques to perform space-time operations. While there is some similarities in smart antenna to the advanced adaptive beamforming antenna systems considered earlier for radar applications, there are also many differences. Basic issues of beamforming for signal enhancement, noise reduction, and interference rejection are still of interest. While the earlier radar systems operate in the higher X frequency band, the wireless systems operate in the lower 800-MHz and 2.4-MHz bands with significantly more multipath and fading problems. Perhaps most important is the fact that we have to deal with both up and down link communications. Various advanced modulation and coding operations not needed in radar systems are present now. With code division-multiple access (CDMA) communication, the bandwidths encountered are wider than the previously encountered systems. Furthermore, the hand-held communication devices have limited power, volume, as well as dimension for the placement of several antennas. All these harsh conditions impose demanding requirements on the smart antenna.

A smart antenna may possess adaptive beamforming capability as well as only be able to switch between small number of fixed beams. The switched beam system offers some of the advantages of a more elaborate smart antenna with reduced complexity and cost. However, for greater flexibility and performance, a smart antenna must utilize adaptivity. In an adaptive system, the weighting vector is adjusted in an adaptive manner to minimize some criteria. In the minimum mean-square error (MMSE) criteria, statistical averaging of the observed quantities are used, while in the LS criteria, the observed time samples are used directly. Variations of the stochastic gradient method are used to solve the MMSE criteria problem.

For both of the above approaches, in order to know the output of the spatial filter, training sequences known both to the transmitter and the receiver are often used. An alternative approach is to use decision directed adaptation, where the desired signal sample is estimated from the generally correctly decided sample. Still another approach uses blind adaptive algorithm, which does not require training sequences. This approach exploits various properties of the transmitted signals.

Two other approaches have been used for an adaptive antenna array system. In the maximum SNR criteria, it seeks to maximizes the ratio of the power of the desired signal to the noise power at the output of the array. In the linearly constrained minimum variance (LCMV) criteria, it minimizes the variance of the output of the array subject to linear constraints such as the directions of interferences. Both of these methods must know the direction-of-arrival (DOA) of the signal of interest. Both methods result in the solution of linear system of equations commonly termed the normal equation. Direct solution as well as various LS approximation methods and RLS methods are possible.

One important application of a smart antenna is finding the DOA and possibly the location of a source. The DOA problem has been an ongoing problem of interest in aerospace and avionics for many years. However, due to the recent FCC mandate requiring a cellular telephone system to locate each user to 125 meter accuracy, there is increasing interest in this area. There are many techniques that have been proposed to tackle the DOA and localization problem. Conventional methods use the beamforming and nulling capabilities of an array to determine the DOA of a source in the presence of noise and interference. While these techniques have been used in many practical systems, there are fundamental resolution limitations. These limitations correspond closely to frequency resolution limitations of classical DFT/FFT method in spectral analysis. Just as there are various modern parametric techniques in spectral analysis,

so too are there many parametric techniques for DOA and localization. Most of the modern techniques are based on signal subspace methods. All subspace methods exploit some signal/noise structure in the modeling of the problem and use eigenvalue decomposition (EVD) or singular value decomposition (SVD) computational tools. The earliest and most well-known subspace method is the MUltiple SIgnal Classificat (MUSIC) technique. The MUSIC method exploits the narrow band assumption of the input sources and uses an EVD of the input covariance matrix to determine the DOA of the sources. Variations and extensions of the MUSIC method have been proposed. There are also many maximum likelihood (ML) methods that have been considered to address the DOA and localization problem. These methods are all computationally costly. Most recently, various joint space-time array processing techniques based on advanced linear algebraic methods have been proposed. The goals of these techniques are to perform blind deconvolution, system identification, source separation, and equalization of the communication channels. While many technically interesting results have been discovered, there is still much work to be done before these methods can be used in practical smart antenna system. Relevant references in these areas can be found in Refs. 37–42. We also note that many of the techniques used for smart antenna are also relevant to acoustic and seismic sensor DOA and localization problems in multimedia, industrial, and military applications.[43]

Acknowledgment

This work is partially supported by NASA under Grant NCC 2-374.

References

1. Allan, V. H., Jones, R., Lee, R., and Allan, S. J., Software pipelining, *ACM Comput. Surv.,* Sept. 1995.
2. Callahan, D., Kennedy, K., and Porterfield, A., Software prefetching, *Proc. 4th Int. Conf. Arch. Supp. Prog. Lang. and O.S.'s,* Apr. 1991.
3. Capitanio, A., Dutt, N., and Nicolau, A., Partitioned register files for VLIWs: a preliminary analysis of tradeoffs, *Proc. 25th Annu. Int. Symp. Microarch.,* Dec. 1992.
4. Case, B., 3DNow boosts non-Intel 3D performance, *Microprocessor Rep.,* June 1998.
5. Faraboschi, P., Desoli, G., and Fisher, J. A., The latest word in digital and media processing, *IEEE Signal Processing Mag.,* March 1998.
6. Fisher, J. A., Trace scheduling: a technique for global microcode compaction, *IEEE Trans. Comput.,* 30(7), 478, 1981.
7. Fisher, J. A., Ellis, J. R., Ruttenberg, J. C., and Nicolau, A., Parallel processing: a smart compiler and a dumb processor, *Proc. SIGPLAN Conf. Compiler Construct.,* June 1984.
8. Fisher, J. A. and Rau, B. R., *Journal of Supercomputing*, Jan. 1993.
9. Glaskowsky, P., First media processors reach the market, *Microprocessor Rep.,* Jan. 1997.
10. Glaskowsky, P., 3D chips break megatriangle barrier, *Microprocessor Rep.,* June 1997.
11. Glaskowsky, P., 3D chips take large bite of PC budget, *Microprocessor Rep.,* July 1997.
12. Gwennap, L., Multimedia boom affects CPU design, *Microprocessor Rep.,* Dec. 1994.
13. Gwennap, L., Intel's MMX speeds multimedia, *Microprocessor Rep.,* March 1996.
14. Gwennap, L., Media processors may have short reign, *Microprocessor Rep.,* Oct. 1996.
15. Gwennap, L., New multimedia chips to enter the fray, *Microprocessor Rep.,* Oct. 1996.
16. Gwennap, L., MediaGX targets low-cost PCs, *Microprocessor Rep.,* March 1997.
17. Hennessy, J. L. and Patterson, D. A., *Computer Architecture. A Quantitative Approach*, Morgan Kaufmann, 1996.
18. Johnson, M., *Superscalar Microprocessor Design*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
19. Kagan, M., The P55C microarchitecture: the first implementation of MMX technology, *Hot-Chips 8,* Aug. 1996.
20. Kalapathy, P., Hardware/software interaction on the MPact media processor, *Hot-Chips 8,* Aug. 1996.

21. Lam, M., Software pipelining: an effective scheduling technique for VLIW processor, *Proc. SIGPLAN Conf. Prog. Lang. Design Impl.,* June 1988.
22. Lapsley, P., Bier, J., Shohan, A., and Lee, E. A., *DSP Design Fundamentals — Architectures and Features*, Berkeley Design Technology, Inc., 1996.
23. Lee, R., Accelerating multimedia with enhanced microprocessor, *IEEE Micro,* 15(2), May 1993.
24. Rau B. R. and Fisher, J. A., Instruction-level parallelism, *J. Supercomput.,* May 1993.
25. Song, P., Media processors begin to specialize, *Microprocessor Rep.,* Jan. 1998.
26. Turley, J., Multimedia chips complicate choices, *Microprocessor Rep.,* Feb. 1996.
27. Wall, D.W., Limits of instruction-level parallelism, *Proc. 4th Conf. Arch. Supp. Prog. Lang. O.S.'s,* Apr. 1991.
28. Howells, P. W., Intermediate frequency sidelobe canceller, U.S. patent 3,202,990, Aug. 1965.
29. Widrow, B., Adaptive filters I: fundamentals, Rept. SEL-66-126, Stanford Electronics Laboratory, Dec. 1966.
30. McWhirter, J. G., Recursive least-squares minimization using a systolic array, *Proc. SPIE,* 431, 1983.
31. Ling, F. and Proakis, J. G., A generalized multichannel least-squares lattice with sequential processing stages, *IEEE Trans. Acoustics, Speech, Signal Proc.,* 32, Apr. 1984.
32. Kalson, S. Z. and Yao, K., A class of least-squares filtering and identification algorithms with systolic array architecture, *IEEE Trans. Inf. Theory,* Jan. 1991.
33. Liu, K. J. R., Hsieh, S. F., and Yao, K., Systolic block householder transformation for RLS algorithm with two-level pipelined implementation, *IEEE Trans. Signal Proc.,* Apr. 1992.
34. Rader, C. M., VLSI systolic arrays for adaptive nulling, *IEEE Signal Proc. Mag.,* 13, 29, 1996.
35. Bolstad, G. D. and Neeld, K. B., CORDIC-based digital signal processing (DSP) element for adaptive signal processing, *Digital Signal Processing Technology*, Papamichalis, P. and Kerwin, R., Eds., SPIE Optical Engineering Press, 291, 1995.
36. Lightbody, G., Woods, R., Walke, R., Hu, Y., Trainor, D., and McCanny, J., Rapid design of a single chip adaptive beamformer, *Proc. 1998 Workshop Signal Proc.*, Manolakos, E.S. et al., Eds., 285, 1998.
37. Rapport, T. S., *Smart Antennas — Adaptive Arrays, Algorithms, & Wireless Position Location*, IEEE, 1998.
38. Liberti, J. C. and Rapport, T. S., Smart antennas for wireless communications, Prentice-Hall, Englewood Cliffs, NJ, 1999.
39. Godara, L. C., Applications of antenna arrays to mobile communications, part I: performance improvement, feasibility, and system considerations, *Proc. IEEE*, 85, 1301, 1997.
40. Godara, L. C., Applications of antenna arrays to mobile communications, part II: beam-forming and direction-of-arrival considerations, *Proc. IEEE*, 85, 1195, 1997.
41. Paulraj, A. J. and Papadias, C. B., Space-time processing for wireless communications, *IEEE Personal Commun.*, 14, 49, 1997.
42. Mouline, E., Duhamel, P., Cardoso, J., and Mayrague, S., Subspace methods for the blind identification of multichannel FIR filters, *IEEE Trans. Signal Proc.*, 43, 516, 1995.
43. Yao, K., Hudson, R. E., Reed, C. W., Chen, D., and Lorenzelli, F., Blind beamforming on a randomly distributed sensor array system, *IEEE J. Selected Areas Commun.*, 16, 1555, 1998.

Barton, D.L. "Design Languages"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

80

Design Languages

David L. Barton
Intermetrics, Inc.

80.1 Introduction

Hardware and systems engineers are spending less and less time operating on, and manipulating, actual hardware. As the complexity of systems that fit on a single die increases, engineers are spending more of their time operating in "virtual" environments — environments that represent the systems under design at a distance, using selected abstractions, and from which the actual hardware can be created.

Central to these virtual environments are design languages. Design languages provide a structured framework within which designers can specify what their designs are to do, how they are to do them, and finally, how they are to be implemented in hardware. Design languages are becoming the primary mechanism for expressing the complex designs that are now the majority product in an increasingly large, and profitable, silicon industry.

This chapter will give a summary of the major parts of design languages, what they express, and how they may be used to design and implement digital hardware. It will begin by stating what design languages are, how they are used, and how they relate to the hardware being designed. It will then move onto the various parts of design languages, starting from the basics (values and objects that contain them), and move up through the various language structures. (Such structures are used to represent the various parts of a system, and to take those parts and compose them into a hierarchical structure that represents the entire system under design.)

There are two different ways to present design languages and languages for use by computers in general. One emphasizes use and attempts to teach design and/or programming while teaching the language. The

second emphasizes the language and language structures. We have chosen the second course in this chapter; other texts are more user-oriented and may be consulted and used for courses in design that have a language orientation. This chapter is also fairly compressed; it is oriented toward reference, not teaching or casual reading.

Two languages are undeniably more popular than others for digital systems design: VHDL and Verilog. The majority of this chapter will use VHDL in its examples and will draw from VHDL for explicit examplars of general concepts. This is being done to reduce complexity and create a single context for the discussion of general concepts. Chapter 81 will attempt to correct this imbalance in language presentation by explicitly presenting Verilog in a separate section, written by Dr. Zainalabedin Navabi, at the end of the chapter.

Uses of Design Languages

The primary use of design languages is to represent information concerning the design of a complex system. This representation serves both as documentation of the system being designed and as a guide to the instincts and practices of the designer as development progresses. The latter aspect of a design language is frequently overlooked in lists of the benefits of computer-aided design (CAD); however, it is the basis of other tool-provided benefits, since it oriented the thinking of the designer toward forms that can be used by those tools. It also provides documentation concerning the design that is far superior to descriptions written in English; the precise meaning of the description provides an unambiguous specification of just what the device under consideration is supposed to do and how it is to be constructed.

The unambiguous nature of a description written in a design language makes it a prime source of information for the various tools that make up the CAD tool suite. Of these, by far the most common and one of the most useful, is the simulator. A simulator constructs an abstract model of the device that is being developed and shows how it will operate by projecting the inputs and outputs of the device onto values that can be graphed and displayed by a computer. Simulations have value on a number of levels. First and foremost, they serve as a check on whether the device matches the customer's desired behavior. Often, requirements written in English are ambiguous. A simulation trace is the first specific representation of the behavior of the device being produced that can be shown to the customer, and which the customer can judge as either meeting or not meeting his requirements.

Beyond customer verification, a simulation is often used for assuring that the device meets the requirements imposed on it. This means feeding sufficient test cases to the simulator to gain some assurance that the device will operate across its entire range of possible inputs and conditions. The judgment of what constitutes a sufficient number of test cases is a technically complex one. This usually involves some sort of code coverage metric: each statement executed at least once, each branch taken at least once, each loop taken a certain number of times, and so on.

Beyond simulation, design languages are used as input (and output) formats for virtually every tool in the CAD toolset. Most of these work on a subset of the various languages, such as structural descriptions with respect to a given cell library. Synthesis is of particular interest. Synthesis tools work on a subset of the design languages roughly corresponding to register transfer level (RTL) descriptions. They produce structural descriptions whose leaf-level cells belong to a given ASIC cell library (see "System Hierarchy" for a discussion of leaf level cells). Synthesis is a vital arrow in the quiver of the designers of complex systems.

Beyond this are all the tools used by the practicing engineer. These include place and route, timing analysis, layout production, and so on. The end product is a working piece of silicon that is fully documented in the design language used to produce it. This aids in reproducing the part later and in reusing the design in other circumstances. Reuse is of particular interest as the complexity of systems increases. Designs that take advantage of the full capability of existing and future silicon foundries will need to make extensive use of existing designs. Design reuse is the key to effective utilization of the full capability of silicon production facilities.

Syntax and Semantics

Any language, including any design language, has two parts that give it meaning. Syntax controls what a description in the language looks like, and semantics control what the description means. More formally, the syntax of a language specifies what character strings (or, more generally, what strings of binary data, where that data may be strings, graphical representations, or other data formats) constitute legal descriptions in the language. Such descriptions may or may not be meaningful; however, any description that is not syntactically correct is dismissed out of hand.

The syntax of a design language is usually defined by a formal description in one of a small set of standardized formats. Most common is a description in a dialect of the Backus-Naur Form (BNF) that has been used to define syntax for software languages for decades. Such formats can be used as input to compiler building tools such as YACC to produce parsers and other language tools.

The semantics of a design language gives meaning to the descriptions written in the language. While there are formal mechanisms for defining the semantics of a language, at present the most common mechanism for semantic definition is carefully worded English descriptions. The wording of these English documents is carefully structure to avoid ambiguity wherever possible. This tends to make the document difficult to read; language definitions are infamously soporific to those who try to read and understand them.

The syntactic and semantic definitions for a design language are usually combined into a single document called a language reference manual (LRM). LRMs are available from standards organizations if the design languages are standardized; VHDL and Verilog are both standardized by the IEEE. The LRM is the document that is voted on by members of the standards organizations, and the LRM (and not any particular implementation or textbook) is the official definition of the design language that it describes.

Models

A description of a silicon-based system is useful only to the extent that it reflects the actual system once it is implemented. For this reason, a description of a system written in a design language is commonly called a model. The model is a representation of reality that, like any other model, abstracts the real system in various ways.

The semantics of a design language limits the models that can practically be written in the language. This range of models is often called the operational model of the language itself. VHDL and Verilog are both discrete event systems. This means that they both view time, and the actions of a system in time, as a series of discrete events that happen at specific points in time. Other languages have different operational models. VHDL-AMS, for example, integrates a continuous time model with the discrete time model of VHDL so that analog systems, and mixed signal digital/analog systems, can be described.

The usefulness of any language is constrained by its operational model (do not think that software languages are exempt from this). It is a serious error to attempt to apply a design language outside its operational model. The operational model is made up of the values and objects that the language can use to represent real processes, the kinds of actions that can be taken on those objects and values, and how the various actions can be combined into a single model representing a digital design.

The engineering process using a design language is a process of refining the model to become closer and closer to reality. This normally requires more and more detail and a lower level of abstraction on the part of the leaf level nodes in the hierarchy that corresponds to the model (see "System Hierarchy" for a definition of leaf nodes). This may or may not be a product of the top-down hierarchical decomposition process that is so beloved of textbooks; in many cases, the design will work up from libraries of available cells to functions that seem to the designer to be of use in the design before working to-down from the overall device specification. In either case, the produced model must agree with the real device to a given level of accuracy for it to be of use.

80.2 Objects and Data Types

Values, the data types to which they belong, and the objects that contain them are the basic units of the models that are created within a description language. The models are mapped to reality by taking values that are created, manipulated, and displayed by the model and mapping them to measurable phenomena associated with the corresponding device. The model is correct, and faithful to reality, to the extent that the values produced by simulation and defined by the semantics of the language correspond to the measured values of the device as it is implemented by the synthesis, place and route, layout, and other tools that operate on the description.

We will describe the values associated with description languages, the types to which they belong, and the objects that contain them in turn.

Values

Values in a description language are used to represent real phenomena. Since physical phenomena are complex and varied in their manifestation, we need a variety of values to represent them. It is more important for these values to be intuitively meaningful than for them to be precisely correct; a meaningless correct value is useless, while a meaningful but somewhat inaccurate value may have some use in dealing with a device being designed. The kinds of values available in the language are thus important to its ability to flexibility and faithfully reflect reality.

This section reflects more differences between VHDL and Verilog than other sections in this chapter; the values available in VHDL are considerably richer than Verilog. Please refer to Chapter 81 for the restrictions on Verilog values.

Basic Values

Basic values are those values that are defined by the language itself. No user action is necessary to make these values available. They are, in some sense, the ground that the language stands on. There is a surprisingly small number of basic values in most languages; mechanisms for creating user defined values can build a larger number of complex values from the basic values.

Description languages define the following basic kinds of values:

Bits All description languages define some kind of logical value, and some provide several. Indeed, VHDL allows the user to define any logical value it likes, as we shall discuss later. The decision concerning which logical value to use is an important one for a complex description. Entire IEEE standards are devoted to logical types and the operations available on them.[3]

Integers Integers represent the mathematical integer values. Languages differ concerning the size of the integer values provided and the protection from overflow that the user can depend on.

Reals Some form of floating point representation of real numbers is provided by all description languages. Floating point numbers are subject to all the restrictions and numerical analysis issues that are found in software languages providing floating point values. Values provided by description languages tend to be IEEE-compliant.[2]

The different kinds of values, including both basic values and user-defined values, are given by their types. We therefore move on to a discussion of types before addressing user-defined values.

Types

A type is a bunch (collection) of values. Each object is assigned a type, which identifies what values the object may contain. User-defined values are identified by creating user-defined types. All description languages have types corresponding to integer values, floating point values, and bit (logical) values. Just what a type is, and how it is defined, is the subject of the following subsections.

Bunches and Types

A type is a bunch of values, where a bunch is a set that is not containerized, i.e., a bunch may not contain other bunches as sets may contain sets. Bunch theory avoids the potential paradoxes of set theory; see Ref. 1 for an exposition of the theory of bunches. The integer type corresponds to the bunch of all floating point values available in the design language. Values that are members of a given bunch are said to be values of the type that is associated with the bunch.

Bunches have the same kinds of operations defined on them as sets: union, intersection, membership, difference, and subsetting (subbunching). These become important in type definitions, which can often be given in terms of these operations.

Type Definitions and User-Defined Types

A type definition defines a bunch of values that can be manipulated, saved in objects, and which represent phenomena in real systems. In a design language, a type definition is a marker that a specific kind of phenomenon is going to be represented in the model. Thus, type definitions are important pieces to the understanding of a complex model.

Basic types need no explicit type definitions; they are assumed to be predefined (in VHDL, these types are presumed to be defined in a package called package STANDARD). Other type definitions construct different kinds of values; they are called user-defined types. Each type definition for a user-defined type identifies the bunch of values that make up the type. There are a number of ways for a user to define a type.

The first way for a user to define a type is for the user to name the values that make up the bunch corresponding to the type. This is called an enumeration type; the values in the type's bunch are enumerated, or explicitly named. An example of an enumeration type is:

type `COLOR` is (`Red`, `Yellow`, `Green`, `Blue`, `Orange`, `Violet`);

This type definition creates a bunch of six values: `Red`, `Yellow`, `Green`, `Blue`, `Orange`, and `Violet`, corresponding to the colors in the rainbow. There is an implicit order imposed on the values in the bunch that corresponds to the order in which the values are named; `Red` is less than `Green`, which is greater than `Yellow`.

Enumeration types are the only mechanism of creating new scalar values. There are two mechanisms for combining values together to create new values. These combinations take one of two forms: a collection of same typed values which are indexed by number (an array), and a collection of heterogeneously typed values which are indexed by name (a record).

Arrays in design languages are similar to arrays in software languages. Arrays are sequences values of the same type which are indexed either by an integer or by an enumeration value. An example of such a type definition:

type `VECTOR` is array (`Red` to `Violet`) of `BIT`;

This definition creates an array type called `VECTOR` which is indexed by `COLOR` rather than by a number. The values of the array — one for each color — are all of type `BIT`. They are accessed by stating which member is desired, by color: my_vector(`RED`) returns the first bit in the array my_vector of type `VECTOR` (since `Red` is the first element of type `COLOR`). Arrays may be sliced; my_vector (`Red` to `Green`) results in an array value which is three elements long.

Records in design language are also similar to records in software languages. Records are divided into fields. Each field has a specific type and a name by which the value corresponding to the field is accessed. An example is

```
type STOPLIGHT is record
current_color: COLOR;
horizontal_grid, vertical_grid: INTEGER;
end record;
```

This creates a record with three fields. If the user has declared a `STOPLIGHT` called stop, then the current color of that stoplight is given by `stop.current.color`.

Design languages thus give the user the ability to create new basic values via enumeration types, and to combine different values into other values (by combining different types into other types) using arrays and records. As we will see, there are other ways to create types from other types.

Subtypes

A subtype bears the same relation to a given type (called the base type) as a subbunch does to a bunch of which it is a subbunch; in fact, the values that are members of the subbunch are the same as the values that are of the subtype. A subtype is defined by giving a restriction on the values of a type. For example,

```
subtype NATURAL is INTEGER range 0 to INTEGER'HIGH;
```

This defines the subtype of natural numbers. All natural numbers are integers; only integers in the given range are natural numbers. The notation `INTEGER'HIGH` is the largest value in type `INTEGER`. This is called an attribute, and it is used in VHDL to give information about something in the language (such as a type). This may be used with any type that has a range; for example,

```
subtype STOP_COLOR is COLOR range Red to Green;
```

This type definition defines the colors that appear in stoplight as a range of the colors of the rainbow.

User-defined types and subtypes create a powerful and flexible mechanism for specifying the types of a system. If used properly, user-defined types can make a description much more readable and understandable for the user reading the model (and therefore much more easy to debug).

Objects and State

Objects are containers of values. A value may be stored in an object and then referenced later in the simulation of the description. The sum total of all the values in all the objects in the description, along with the system time and the internal objects used to keep track of the simulation cycle, is called the system state. The system state, at a given simulation time, should map to measurable phenomena of the system being modeled at the time corresponding to the given simulation time. To the extent that the mapped system state differs from the measured values, the model is inaccurate.

The semantics of a design language define two things:

1. How each object in the description gets an initial value (what the initial value is for each object in the system).
2. Given a system state and a description, how the values contained by the objects change at the next "step" (where "step" is also defined by the language semantics).

The objects of the model, and the values they contain, are thus the link to the behavior of the modeled device. Proper display of these values (in particular, using familiar graphical concepts like waveforms and user interface devices) allows the user to make sense of the simulation.

Object Kinds

Objects change values as a result of the execution of language statements. Description languages have a much wider range of action than software languages. Different statements affect objects in different ways, and in general they may or may not affect the values in the objects immediately. The way that an object changes value is determined by the kind of the object. The kinds of objects available in VHDL and its analog extension, VHDL-AMS, are

Constant Objects of kind constant never change value; they keep their initial value throughout all time.

Variable Objects of kind variable change their value immediately on the execution of an assignment statement (specifically, a variable assignment statement).

Signal Objects of kind signal change their value during the simulation cycle. A signal assignment statement records its action; in VHDL, this is recorded in a driver for the signal. During the simulation cycle the values of the drivers are combined to determine the new value of the signal. Signals that are assigned to in more than one part of the model (in more than one process) are said to have multiple drivers; in this case, the drivers need to be resolved. This is done by providing a function that takes an array of values and produces a single value of the same type as a result; this function is attached to the subtype of the signal. Such signals are said to be resolved signals, and subtypes that have function names attached to them are called resolved subtypes.

Quantity Objects of kind quantity get values as a result of the solution of a set of simultaneous equations that are derived from the description in the description language. As time progresses, an analog equation solver assigns each quantity a value that is in agreement with the entire system of equations.

Terminal Objects of kind terminal are composed of two related quantities that are of related types, one across quantity and one through quantity. Terminals, and their connections, are governed by conservation equations determined by Kirchhoff's Voltage Law and Kirchhoff's Current Law. The quantities that make up the terminal get their value in the same manner as other quantities. Terminals do not have types per se. Instead, a terminal is said to belong to a nature which specifies the types of the across and through quantities of the terminal.

Object declarations specify the kind and type of the objects being declared. In VHDL, this takes the following form:

```
constant pi: REAL = 3.14159;
variable x, y, z: INTEGER;
signal in1, in2, in3: BIT;
quantity bias_voltage: VOLTAGE;
terminal c1, c2: ELECTRICAL.
```

80.3 Standard Logic Types

All description languages have one or more representations of electrical signals, called standard logic types. Some, such as VHDL, allow the user to define other logic types as well; however, simulator vendors often have specialized features that allow simulations of descriptions expressed in terms of one of the standard logic types to run faster than simulations of descriptions expressed in terms of user-defined types.

Simulator vendors may choose to optimize any logic types that they wish; however, in the case of VHDL there are a number of types that are standardized and are therefore available for optimization by simulator vendors. At the time this chapter was written, these standard logic types derive from two specific sources.

The first is package STANDARD itself. This defines the type BIT, which consists of two values: '0' and '1'. Because this is defined in the package STANDARD, every implementation of VHDL must provide it (although it may not be optimized on all systems).

The second source is the standard utility library called std_logic_1164. It defines a logic type called std_ulogic (and a corresponding resolved type called std_logic) with the following values:

'U' Uninitialized
'X' Forcing Unknown
'0' Forcing 0
'1' Forcing 1
'Z' High Impedance
'W' Weak Unknown
'L' Weak 0
'H' Weak 1
'-' Don't care

These values are also provided by Verilog, and are optimized for various timing optimizations as a part of those simulator vendors that support the VITAL standard. In addition to this standard logic type, there are four subtypes of this type that are defined:

X01 This subtype contains only these three values: 'X', '0', and '1'.
X01Z This subtype adds 'Z' to X01.
UX01 This subtype adds 'U' to X01.
UX01Z This subtype adds 'Z' to UX01.

In addition to these two types, there is a proposed extension to the standard that adds three values to std_logic:

'C' Capacitive unknown.
'D' Discharge 0.
'P' Precharge 1.

This type is not as widely supported across the universe of vendors; therefore, counting on these last three values being provided by a given vendor may not be wise.

Logical Operations

In addition to the standard logic types, there are a number of standard logical operations defined on values of these types. These are defined on all standard logic types. They are

- Logical and (conjunction).
- Logical or (disjunction).
- Logical not.
- Logical nand (not and)
- Logical nor (not or).
- Logical xor (inequality).

These operations may be used on vectors of logical types as well.

Other Standard Operations

In addition to the standard logic types and logical operations, there are a number of standard numeric operations that are provided by VHDL and most VHDL implementations. These are either provided in package STANDARD, or in a standard mathematical package called MATH_REAL. These include

Additive operations Addition, subtraction, and inverse.
Multiplicative operations Multiplication, division, modulus.
Exponentiation Exponent, log.
Transcendental functions Sine, cosine, tangent, arcsine, arccosine, arctangent.
Hyperbolic functions Hyperbolic sine, cosine, and tangent.
Boolean operations And, or, and not. Note that these are separate from the logical operations above.

These operations may be used to build up expressions that appear in assignment statements, sequential statements, simultaneous statements, and statements where expressions are appropriate such as if, loop, and return statements.

80.4 Concurrent Statements

Concurrent statements are the basic units of cooperating action. Hardware involves a number of devices — whether modeled at the gate level, the transistor level, or at a high behavioral level — working simultaneously and exchanging information. Concurrent statements specify the actions that execute in

isolation from one another, exchanging information using signals and shared variables. The basic unit of concurrent action is the process.

Processes

Processes are units of action in a simulation. Processes affect the system state, and therefore affect the model of the device under development, by assigning to signals. These signals are in turn read by other processes and used to determine the signals they assign in turn.

In general, processes may be thought of as tasks or co-routines in a software language. Processes may declare variables which serve as internal state, call procedures, and read from and write to files. Processes also suspend their own execution via wait statements. Processes are composed of sequential statements (see Section 80.5 for a discussion of sequential statements).

Processes begin execution at the beginning of simulation, and also at some combination of the following three conditions:

- when one of a given set of signals changes value
- after a certain period of time
- when a Boolean condition becomes true

When suspending execution, each process states the conditions under which it will next resume execution. Consider the following simple example:

```
process
begin
  a <= b nand c after 50 ns;
  wait on b,c;
end process;
```

The signal assignment statement specifies the value of the signal **a** based on the values of signals **b** and **c**. The following statement specifies that whenever the values of either **b** or **c** change, the process will resume execution. Note that processes always inherently loop; after the wait statement, the process will resume execution at the first statement of the process.

There are several other kinds of concurrent statements; however, all of them that execute are equivalent to processes of one or another. We will consider these subsequently.

Other Concurrent Statements

Concurrent statements may be divided into two categories: those that affect the system state during execution (and which are therefore equivalent to some concurrent process) and those that define the structure of the model being built. We will cover the first category in this section; the second category will be dealt with in other sections below.

Signal Assignment Statements

Signal assignment statements may be expressed directly, without embedding them inside a process statement. Such signal assignment statements are equivalent to a process that contains them, and a wait statement containing the signal names of the signals on the right hand side of the assignment statement. For example, the process statement given above is equivalent to the signal assignment statement:

```
a <= b nand c after 50 ns;
```

The b and c for the wait statement in the process statement are taken from the **nand** expression on the right side of the signal assignment statement.

There are two other forms of the signal assignment statement, each with an equivalent process. The following conditional signal assignment statement

```
output <=in1 after 10ns when cntl == '1' else
         in2 after 10ns
```

is equivalent to the following process:

```
process
begin
  if cntl = = '1' then
    output <= in1 after 10ns;
  else
    output <= in2 after 10ns;
  end if;
  wait on cntl, in1, in2;
end process;
```

which contains an if statement derived from the choices. Note that the list of signals in the wait statement is the union of those found in the assignment statements, along with the expression in the when clause. Similarly, the following selected signal assignment

```
with cntl select
output <= in1 after 10 ns when '1',
           in2 after 10 ns when '0';
```

is equivalent to the following process:

```
process
begin
case cntl is
  '1' => output <= in1 after 10ns;
  '0' => output <= in2 after 10ns;
end case;
wait on cntl, in1, in2;
end process;
```

Again, the list of signals in the wait statement is equivalent to the union of the lists provided by each signal assignment statement, along with the name in the case expression.

Behavior that is defined totally by concurrent signal assignment statements (and particularly by simple signal assignment statements) is often referred to as a register transfer level, or RTL, description. Such descriptions are particularly useful in conjunction with synthesis tools; there are standardized subsets of both VHDL and Verilog for use by synthesis tools.

Concurrent Procedure Calls

The concurrent procedure call is a procedure call that is executed as a process itself. This is useful if a more complex piece of behavior is used repeatedly as a concurrent "chunk," and embedding the single procedure call is confusing and wasteful. The following procedure call

```
select_signal (cntl, in1, in2, output);
```

is equivalent to the following process statement:

```
process
begin
  select_signal (cntl, in1, in2, output);
  wait on cntl, in1, in2, output;
end process;
```

The signals for the wait statement are gathered from those parameters of mode **in** or **inout**.

Concurrent Assertion Statement

Like the concurrent procedure call, the concurrent assertion statement allows the designer to express invariants about his design without having to embed the statement in a process. Unlike some of the other concurrent statements, it is perhaps more common to find the concurrent form of the assertion statement used than the sequential form. The following concurrent assertion statement

```
assert not (in1 == '1' and in2 == '1')
  report "Illegal input combination on flip flop."
  severity WARNING;
```

is equivalent to the following process:

```
process
begin
  assert not (in1 == '1' and in2 == '1')
    report "Illegal input combination on flip flop."
    severity WARNING;
  wait on in1, in2;
end process;
```

Concurrent Break Statement

The concurrent break statement signals a discontinuity to the analog solver, which must then reinitialize the values it is calculating for all quantities and terminals based on information given by the break statement. The following break statement

```
break vel => –vel on ypos'ABOVE(10);
```

is equivalent to the process statement:

```
process
begin
  break vel => –vel;
  wait on ypos'ABOVE(10);
end process;
```

Note that the use of the form ypos'ABOVE(10) creates a signal that changes value when the expression ypos—10 becomes above zero; in other words, when ypos crosses the value 10.

80.5 Sequential Statements

Sequential statements appear in process statements or in subprograms (procedures and functions) which are called by process statements. They are the statements that change the program state and that control execution flow through the process. Statements that change the program state are assignment statements.

Many of the sequential statements in description languages should be familiar to software programmers. Indeed, it may be said that in many ways, designing hardware with a design language is similar to programming. We will summarize constructs similar to software languages here without a tremendous amount of discussion. More time will be spent on those constructs that are unique to, or more common in, description languages.

Assignments

An assignment calculates an expression and assigns it to an object that is part of the system state. There are two kinds of assignment statements: variable assignment statements and signal assignment statements. In both cases, the expression on the right side of the statement is evaluated and then the action specified

by the statement takes place. The two kinds of statements are distinguished by the sign used to mark the assignment:

```
x: = y + z
a <= b nand c;
```

The first statement is a variable assignment, and the second is a signal assignment statement.

A variable assignment statement takes effect immediately; the statement directly following it can read the new value. A signal assignment statement takes effect more indirectly in the following fashion. Consider the following signal assignment statement:

```
clock <= 'X' after 10 ns, '1' after 20 ns;
```

The right-hand side consists of a potentially unbounded number of pairs (two in this case), separated by commas, consisting of an expression and a time (which can also be an expression). The times must be nondecreasing. Each process has an object called a driver for each signal for which it has a signal assignment statement. The update to the driver takes place immediately; the value of the signal does not change until all processes have been executed and the simulation cycle updates the signal values (see 80.8 for a description of the simulation cycle).

Updating the driver is a complex operation. The driver may already contain a number of time value pairs. The time value pairs in the signal assignment statement are overlaid on top of the time value pairs in the driver in one of two ways. The first, called transport delay, takes the first time in the signal assignment statement and eliminates all pairs in the driver with a time equal to or later to the first time. The second, called inertial delay, also reflects the first value back to the first occurrence of that value in the driver, eliminating all pairs in between. For a fuller explanation of this process, see the VHDL LRM; the process in Verilog is considerably simpler.

Control Flow

Within a process statement, control transfers from statement to statement in a sequential fashion, one after another, unless this is interrupted or changed by one or another control flow statement. Most of these are familiar to programmers; however, there is one that is used in description languages. This is the wait statement, which has been mentioned previously. Other statements will be mentioned more briefly.

Wait Statement

The wait statement suspends the execution of a process and specifies the conditions under which the process will resume execution. We have already seen examples of a wait statement; its most general form is

```
wait on <signals> until <condition> for time;
```

The user can leave out any of the parts if he wishes. In this case, these are automatic defaults. The meaning of each of these parts is

on <signals> The process will resume when any of the signals in the list of signals given changes value. If none is given, the list is derived from the signals in the condition; if there is no condition, the list is empty;

until <condition> The process will resume when the condition is true; it is checked only if one of the signals in the list above changes value. If no condition is given, it defaults to TRUE.

for <time> The process will resume execution after the given amount of simulation time has elapsed. If no time is given, it defaults to the highest time in the system (minus the current time; the process will restart "at the end of time," or never).

Thus, the process will resume when one of the signals in the list changes value or when the given time has elapsed, whichever is first. If one of the signals changes value and the condition is false, the process does not resume.

Assertions

An assertion states a condition that must be true at a point in the simulation of the system. If the statement is not true, the message in the report clause is printed out and execution either halts or continues depending on the value in the severity clause. An example is:

```
assert not (in1 == '1' and in2 == '1')
  report "Illegal input combination on flip flop."
  severity WARNING;
```

This statement works just as well inside a process as outside a process; inside a process it is a sequential statement, and outside a process it is a concurrent assertion statement with an equivalent process as described in "Concurrent Assertion Statement."

Often we need no condition for an assertion statement; the simple fact that we have reached a point in the execution flow is enough. In this case, we need only specify the report and the severity clauses in a report statement. The previous assertion statement is equivalent to

```
if in1 == '1' and in2 == '1' then
  report "Illegal input combination on flip flop."
    severity WARNING;
end if;
```

Conditionals

As with most software languages, there are two forms of conditionals: the if statement and the case statement. The following are equivalent

```
if cntl == '1' then
  output <= in1 after 50 ns;
else
  output <= in2 after 50 ns;
end if;
```

```
case cntl is
  '1' => output <= in1 after 50 ns;
  others => output <= in2 after 50 ns;
end case;
```

Loops

There are two forms of loops: for loops and while loops. Associates with loops are two statements for loop control: the next statement, which immediately jumps to the beginning of the loop (when used with a label, to the loop with the label given), and the exit statement, which jumps to the statement following the end of the loop (if labeled, the loop with the label given). Three loops that do the same thing–initialize an array–are

```
for i in 1 to 10 loop
  for j in 1 to 10 loop
    a (i,j): = 0;
  end loop;
end loop;
```

```
i: = 0;
iloop: while i < 10 loop
  i: = i + 1; j: = 1;
  jloop: loop
    a (i,j) = 0;
```

```
    if j == 10 then
      exit jloop;
    else
      j: = j + 1;
    end if;
  end loop;
end loop;
```

```
i: = 0;
iloop: while i < 10 loop
  i: = i + 1; j: = 1;
  jloop: loop
    a (i,j) = 0;
    if j == 10 then
      next iloop;
    else
      j: = j + 1;
    end if;
  end loop;
end loop;
```

Obviously, the second and third are for purposes of illustration only.

Procedure Calls and Returns

A procedure call is an invocation of a given procedure, giving values for all the parameters. The statement simply gives the name of the procedure and the parameter values:

```
fetch_memory (1024);
```

The return statement returns control from a subprogram. If inside a procedure, it consists of the single keyword return; if inside a function, it consists of the keyword return followed by an expression that, when evaluated, becomes the value of the function call.

Null Statement

The null statement consists of the single keyword null and does nothing at all.

Break Statement

The break statement is not a control flow statement per se; control continues with the immediately following statement. The break statement signals that a discontinuity has occurred and that the analog solver must re-initialize its set of solutions before proceeding. An example is

```
break vel => -vel when ypos == 0;
```

The expression gives the new value for the quantity during reinitialization, and the condition denotes when the break is signaled. If the expression evaluates to false, no break occurs.

80.6 Simultaneous Statements

Simultaneous statements determine the value of quantities and terminals. They do not take effect directly; instead, they are used to determine the values of the objects involved via a mathematical process of determining the solution of a system of simultaneous equations. To explain this further, we consider modeling analog circuits in general.

Analog circuits are modeled by specifying, via a series of different algebraic equations (DAEs), the relationship between the values of the quantities involved. These equations are characteristic for each

analog device. It becomes the responsibility of the simulator to assign values such that these mathematical relationships are maintained.

In addition to the relationships specified by the DAEs in a model, the analog system also uses two laws — Kirchhoff's Current Law, which states that the sum of all currents into a single terminal must be zero, and Kirchhoff's Voltage Law, which states that the sum of all voltages in any loop of connected terminals must be zero — to create any additional mathematical relationships needed to solve the system.

Given that these DAEs exist and are specified in the description language, the job of the analog solver is to

1. Collect the DAEs and create a system of simultaneous equations out of them.
2. Determine a set of initial values for the system.
3. Solve the system in increasing time steps, synchronizing with the fixed event simulator of the digital system as needed.

Within this general description there is a great deal of room for differences in specific algorithms. The important task is to maintain the mathematical relationships defined in the model whenever the values of the quantities and terminals are read.

Specifying DAEs is the job of the simultaneous statements. These are not part of VHDL, but are specified as part of the analog extensions to VHDL; the entire language is referred to as VHDL-AMS.

Forms of Simultaneous Statements

The basic form of a simultaneous statement is an equation. It simply sets up the relationship involved. The equation is written identically to a Boolean condition that could be checked in a conditional statement. An example is

```
brvolts == L * brcurr'DOT;
```

where the use of the form `brcurr'DOT` signifies the time derivative of the branch current.

As with the signal assignment statement, there are alternative forms of the simultaneous statement. These select the equation to be used by the analog solver depending on the values of the surrounding objects. One is analogous to a conditional statement:

```
if v> 0.0 use
  v'dot == -g – v ** 2 * resis;
else use
  v'dot == -g + v ** 2 * resis;
end use;
```

The second is analogous to a case statement:

```
case cntl use
  when '1' => outv = 10 * inv;
  when '0' => outv = 2 * inv;
end case;
```

There is a third form which is considerably more complex, and effectively allows the user to insert a function that is called each time a given quantity value is determined. This is the simultaneous procedural statement. It allows the same quantity names to be used as variables and for them to be assigned to as necessary. For example, the procedural equivalent of the conditional example above is

```
procedural is
begin
  if v > 0.0 then
    v'dot: = -g – v ** 2 * resis;
  else
```

```
        v'dot: = -g + v ** 2 * resis;
    end it;
end procedural;
```

Any number of quantities may be assigned to by the procedural statement; the procedural statement then acts as the DAE for all of them.

80.7 Modular Designs

Modern systems that are implemented on a single silicon chip can be incredibly complex. They can include millions of lines of software, millions of gates of digital logic, and thousands of active analog devices. The only mechanism for controlling this kind of complexity is modularity. This means splitting up a very complex design into small pieces that can be understood and controlled. This section begins with some discussion of general issues in modularity, then moves on to mechanisms for creating and enforcing modularity.

General Issues

In dividing a system up into pieces, there are several concerns that must be addressed. Each one of these is important to the ability of the user to understand and use in individual pieces. To a large extent, this differs depending on what can be legitimately divided and separated from other pieces of the system, and this varies between digital and analog descriptions.

Description languages have the task of providing the user with the ability to split up the system in a manner that conforms to engineering and design practice. In general, this means separating interface from implementation and making sure that the interface is uniquely defined. Different mechanisms of partitioning and separation are used, depending on the effect that is desired. Decomposition is a powerful tool, and sometimes a two-edged one. Great attention is needed to partition a design effectively and efficiently.

Forms of Partitioning

There are several forms of partitioning that are made available by description languages:

Algorithmic partitioning This entails the ability to use an algorithm, or rule, without necessarily knowing how it is implemented. This kind of partitioning is used by software languages, as well, and is implemented in general with some sort of packaging and subprogram mechanism.

Physical partitioning A design can be partitioned according to the physical packages that appear within it. This means if a given device has a physical realization — it is placed in a separate package with pins and soldered connections, for example (other physical packaging is possible) — it has a separate definition in the description language.

Design partitioning A designer normally begins his design with a block diagram which shows, in conceptually separate blocks, a number of functions and the connections between them. These blocks may or may not implement identifiable algorithms, and they may or may not have specific physical packages associated with them.

Implementational partitioning During the design of a complex system, multiple implementations (or models) of a given part will be used during the system lifetime. These may include a bus functional model (a model that attempts to give timing of messages flowing through the device at a very high level), a behavioral model, an RTL model, and a detailed structural model.

Description languages must be able to provide each of these partitioning mechanisms.

Name Space Control

In a large design where objects, models, devices, and other parts of the system under development all have their separate names, name space control is a tremendous problem. In general, having unique names

for each of these named things is not possible. Name space control is an important part of the definition of any description language.

Name space control is a problem shared by software languages as well. In general, it consists of importing various sets of names — libraries and packages — into a given description unit and using those names just as any locally defined name.

There are usually two options for importing names:

1. Import all the names defined in a given package or library.
2. Import a list of a selected list of names defined in a given package or library.

Designers often use method 1, which requires less writing. One reason for this discussion is to urge more use of method 2, which is more trouble but provides a great deal more control over the names and more visibility over where they are defined.

Another mechanism of name space control used by many languages is to mark off a section of the description where names may be declared that are not visible outside that section. The general name for such a section, both in software languages and in description languages, is the block. In VHDL, there is an explicit formulation of a block that is called the block statement. The basic format of a block statement is:

```
block
  <declarations>
begin
  <statements>
end block;
```

where the declarations and statements can be anything that may occur in a concurrent declaration or statement context, respectively. Names declared in the block are not visible outside the block.

In fact, it is possible to use a block like a component (see "Interfaces and Implementations" for more discussion) by defining an interface consisting of ports and generics and connecting these to objects outside the block. This is a rather advanced use of the block statement, and we will not discuss it further here.

System Hierarachy

Each unit in a description has two choices concerning how to define the behavior of that unit:

1. It can link together other components — a structural definition.
2. It can define the unit with code specifying how it behaves — a behavioral definition.

Units can mix behavioral and structural definitions.

Given that a description has been split up into a large number of pieces, the pieces must be recombined in order to form a complete system description. This combined description is composed of three kinds of units:

1. Units with only behavioral descriptions. These are called leaf level units; they use no other devices.
2. Units which refer to other units in their definition. These units are called the parents of the units to which they refer, which are called children. Units which have both parents and children are called internal units.
3. Units which refer to other units in their definition but have no parents. These are called roots.

While it is possible for a given design to have multiple roots, in practice there is a single root for each design. The transitive closure — parents using children, who in turn, use their children until nothing but leaf units are left — of the root is called the design hierarchy of the system. The design hierarchy represents the entire system being designed. It almost always represents a testable unit, with a separate physical interface. Test benches and testing occur on design hierarchies and refer to the entire design hierarchy through its root.

Interfaces and Implementations

The first mechanism for partitioning provided by description languages is a separation between interfaces and implementations. In VHDL, this occurs at several levels. This section will cover physical and design partitioning and leave algorithmic partitioning to "Packages and Subprograms."

Any interface must define the interface points, or points where information may be exchanged. These are often divided into two kinds of exchange:

1. interfaces where a physical package itself would be connected, or where actual information is exchanged during system operation (perhaps by coupling);
2. interfaces where information concerning system modification and adaptation is exchanged.

In VHDL, the former is accomplished by ports, which are signal objects, and the latter is accomplished by generics, which are constant objects. The interface is called an entity. An example is

```
entity multiplexor is
  generic (delay: TIME);
  port (cntl, in1, in2: in BIT;
        output: out BIT);
end entity multiplexor;
```

This section of VHDL code gives all the information necessary for a user to understand how to use the device, including setting its delay time and how it is to be connected into a system. It says absolutely nothing about how the device behaves. This is left to the specification of the implementation of the device. In VHDL, this is done by the architecture:

```
architecture behavioral of multiplexor is
begin
  with cntl select
    output <= in1 after delay when '1',
              in2 after delay when '0';
end behavioral;
```

Together, the entity and the architecture define the entire device; each is incomplete without the other. There may be many architectures for a given entity. The appropriate architecture is chosen by a configuration, as seen in "Configuration."

Instances and Instantiations

Given units created by entities and architectures, we must be able to use them. Using an entity architecture pair (or component) is called creating an instance of that component, and the statement that does so is a component instantiation statement. Creating an instance of a component means creating all of the objects defined by that component (which means all of the objects defined by the design hierarchy rooted at that computer) and connecting them appropriately, as shown in the component instantiation statement.
VHDL has an additional level of indirection between the component instantiation statement and the component: the component declaration. If the name in the component declaration is the same as the name of the entity in the library, then no further action is needed. If the designer wishes to change the device used later, then a configuration can be used, as in the following section.

All of these are combined into the (rather stupid) implementation of a nand gate:

```
entity my_nand is
  port (in1, in2: in BIT;
    output: out BIT);
end my_nand;
```

```
architecture sample of my_nand is
  component my_and
    port (in1, in2; in BIT;
      output: out BIT);
    end component;
  component my_not
    port (input: in BIT;
      output: out BIT);
    end component;
  signal t: BIT;
begin
  and1: my_and port map (in1, in2, t);
  not1: my_not port map (t, output);
end sample;
```

This implementation defines two components, my_and and my_not, and uses them to build up the implementation of the nand gate by creating one instance of each. The form of the component instantiation statement begins with a required label, and then names the component. The port map follows, which gives an object of kind signal for each port in the component definition. A generic map clause would be used to provide values for any generics in an entity definition.

Configuration

As stated in the previous section, VHDL provides an extra level of indirection between the component instantiation statement and the component to which it refers in the form of a component declaration. If the name of the component matches the name in the library and if the port and generic lists match as well, then nothing more need be done; a default connection will occur between the component declaration and the component in the library.

In some cases, it is desirable to override this default matching. Indeed, it may be desirable to override this matching all up and down the entire design hierarchy. In VHDL, this is done with a configuration declaration. The configuration intuitively "follows the design hierarchy," specifying which component is used for each component declaration. The configuration allows the user to "step into" the different parts of the design hierarchy. An example is

```
configuration my_config of my_nand is
  for sample
    for my_and use and_123(struct); end for;
    for my_not use not_456(struct); end for;
  end for;
end configuration my_config;
```

Note that the various for statements follow the design hierarchy down; each time the user "steps into" a component, a new for statement is written. The for statement specifies which entity and architecture (the architecture name appears in the parentheses) is to be used for each component, along with configuring any components below it in the design hierarchy.

The configuration declaration is the only unit in VHDL that can "dip into" different units and get visibility over the names in those units as it progresses down in the hierarchy. This makes the configuration declaration potentially the most complex single unit in the description.

Packages and Subprograms

Algorithmic partitioning is accomplished by the use of packages and subprograms. Packages are collections of declarations, including type and subprogram declarations. Packages allow the user to take a

number of related declarations and place them together, using them in many different models. Subprograms allow the user to take a portion of an algorithm and separate it, invoking it in a number of processes. There are two different kinds of subprograms: procedures (which act like sequential statements) and functions (which act like expressions).

Both packages and subprograms maintain the VHDL practice of separating interfaces from implementations. In all cases, the interfaces are called declarations (package declaration, subprogram declaration) and the implementations are called bodies. The bodies may repeat part of the declaration and then give values to the declaration by giving more details.

Since packages are collections of declarations and therefore do not execute per se, they have no interface points, and therefore no parameters. Subprograms do execute and therefore have interface points. These parameters are specified by name, type, and direction; the direction specifies whether the parameter is being used as input to the subprogram or output from the subprogram. Functions cannot use parameters as outputs; they can only return values in an expression.

Subprograms can be overloaded; i.e., different subprograms can share the same name. In this case, the types of the parameters to the subprogram are used to determine which subprogram is to be called. This is useful where the same essential function is defined on a number of different types, thus allowing the user to state the definition for each of the types and therefore avoid using different names for what is the same basic operation.

Consider the following example:

```
package sample_inc is
  constant factor: INTEGER;
  procedure inc (x, y: in INTEGER;
                 z: out INTEGER);
  function inc (x, y: INTEGER) return INTEGER;
end package sample_inc;

package body sample_inc is
  constant factor: INTEGER: = 2;
  procedure inc (x, y: in INTEGER;
                   z: out INTEGER) is
  begin
    z = (x + y) * factor;
  end procedure;
  function inc (x, y: INTEGER) return INTEGER is
  begin
    return (x + y) * factor;
  end function inc;
end package body sample_inc;
```

In the context of this package, the following two statements

```
inc (a, b, c);
c: = inc (a, b);
```

do precisely the same thing. Note that the value of constants as well as subprograms can be deferred to the implementation; it is possible to name constants without giving their value.

80.8 Simulation

Although details vary, each description language determines object values and executes statements in a manner determined by what is called the simulation cycle. The simulation cycle has responsibility for the following tasks:

- initialization of all objects;
- initial execution and resumption of processes;
- determination and advancement of simulation time;
- solving systems of simultaneous DAEs, and setting the values of quantity and terminal objects; and
- determination and propagation of values in signal objects.

In general, while some interleaving may take place, these operations are done in separate steps that make up what is called the simulation cycle.

Initialization of objects is performed prior to the beginning of simulation. This is a one-time operation and has the effect of setting the system to a known state before simulation begins.

The processes to be executed are determined by a combination of three things: the simulation time, changes on various signals, and the conditions set in wait statements. In general, models do not depend on the execution order of processes. Indeed, VHDL models that do depend on this order are deemed to be in error.

Simulation time is determined based on the drivers of all the signals in the model. Each driver has an earliest transition; the minimum of these earliest transitions determines the next simulation time. In some cases, simulation time may not advance at all, but remain at its present value. In VHDL, this is called a delta cycle.

The system of simultaneous equations is determined each time the analog solver is invoked. Because of the action of conditional and selected signal assignment statements, that system may change at any time. The analog solver has the task of determining the time steps it will take (although this may be limited in VHDL by a predefined procedure call). In addition, if a discontinuity has been signaled by a break statement, the analog solver has the task of determining a new set of initial values before continuing to update its own simulation time steps.

Determining the value of signal objects means taking the contribution to that signal value from each process that assigns to it (contained in the driver) and combining them appropriately. If there is only one driver, then the driver value is the value of the signal. If there are multiple driver values, then the values must be resolved. This is the purpose of the resolution function which is specified by the subtype (not the type) of the signal. This resolution function takes an array of values in the type of the signal, and produces a signal value of that type. The values of each of the drivers is invoked, and the result becomes the value of the signal.

The extent to which these operations can be interleaved, and executed in parallel, is determined by the exact definition of the language as contained in the LRM. Simulators are not confined to doing these functions separately, or serially, as long as the results they produce are identical to those defined by the abstract simulation mechanism defined by the LRM. Study groups and standard practices for parallel simulation are a continuing field of research in VHDL and Verilog language development.

80.9 Test Benches

Given a model of a device under development, the designer next has the task of simulating that device. Simulation, and the definition of test vectors to the simulation, is very tool dependent; however, there are two basic approaches:

- The toolset can provide inputs to the simulation and graph (or check) outputs outside the model itself. This approach is totally dependent on the tool features, and we do not deal with it further here.
- The toolset can assist in the construction of a component called a test bench, which instantiates the design to be tested, feeds data to the input pins, collects data from the output pins, and checks it in various ways. The test bench becomes the root of the design hierarchy that is simulated.

Writing a test bench is like writing any other component in the description language; the same language constructs are used and the same principles apply. A test bench does two things. First, it generates

behavior; it causes the device to simulate. In some development methodologies, this is a vital function. The initial checking of the device consists of generating random inputs and visually inspecting the outputs (as waveforms or as more comprehensible outputs) to see if they match those inputs. Such methodologies can go even further, using this mechanism to produce actual test vectors for the resulting device from the simulation itself (and thus using the model as the specification of the device).

Second, a test bench can be used to check that the device actually fulfills its function. This means both generating the behavior and checking that the behavior is correct. Such checking requires an outside specification of that behavior. There are two approaches to writing this kind of test bench: algorithmic, where the specification is provided by an algorithm, and vector-based, where the specification is provided by a set of test vectors.

Algorithmic Test Benches

An algorithmic test bench tests a device by implementing the algorithm that the device is intended to embody. For example, if a multiplier is being designed, an algorithmic test bench would feed two numbers to the input of the device, capture the output, then multiply the numbers together and check that the output matches the result of the multiplication.

Algorithmic test benches can fit in very nicely with a top-down design methodology where each entity has at least two architectures: one behavioral and one structural. If the algorithm in the behavioral architecture is encapsulated in a procedure, the test bench can call the procedure in parallel with the provision of inputs to the structural device. The outputs of the two can then be compared. This can apply to quite complex devices, even to ISP-level simulations of complex microprocessors.

While the function of the device is provided by the algorithm, the exact inputs are not. There are three approaches to choosing inputs. First, the designer may decide (particularly in a memoryless device such as a multiplier) to exhaustively simulate the entire input space. This is implemented using a nested loop, with one level for each input. Second, the designer may choose to select inputs randomly and run the simulation for as long as time allows. This is implemented with a random number generator. Finally, the designer may attempt to provide some level of fault or algorithmic code coverage. This must be accomplished with a test tool that measures such coverage metrics.

Vector-Based Test Benches

Vector-based test benches use a series of inputs and outputs that are, in some sense, defined outside the model itself. The only task of the test bench is to obtain the test vectors (usually from a file), provide the inputs to the input ports, read the outputs from the output ports, and check them against the output values provided by the test vectors.

Vector-based test benches are thus very simple to write, and indeed there are tools that write them automatically based on a list of input pins, output pins, and a matching between the pins and the bits in the test vectors. From this, the code to read the vectors, apply them to the device, and check the output values can be derived automatically.

In vector-based test benches, the complexity is contained in the vectors themselves. Indeed, it could be said that the test vectors are a specification of the behavior that the device must meet. The production of the test vectors therefore becomes of vital importance. Often, the test vectors are produced by simulation as discussed above. In this case, the specification contained in the test vectors can be said to develop iteratively, first by inspection of initial outputs and later by running further inputs and expanding test cases as necessary.

The adequacy of vector-based test benches is usually measured by fault coverage. Fault coverage entails the deliberate introduction of faults into the model, either by altering the actual model code or (more commonly) by having the simulator introduce faults into the model automatically without changing the code and recompiling. Faults are usually simple in nature, the most common being setting the value of a specific object to a given value regardless of the actions of the surrounding devices (a “stuck at” fault).

A set of vectors is said to achieve a certain percentage of fault coverage for a device if that percentage of generated faults are detected by the test bench using the given set of test vectors.

The use of vector-based test benches is sufficiently important that the standardization activities have been devoted to it. In particular, the waveform and vector exchange specification (WAVES) is a standard test vector specification language that is designed to both work with a VHDL model and to provide input to various automatic test equipment (ATE) systems. Tools exist to create test benches from a WAVES description as described previously, making the simulation of a VHDL mode with a set of WAVES specified test vectors easy and convenient. These tools are vendor-independent in the sense that they produce standard VHDL code that runs on any compliant simulator.

References

1. Hehner, E. C. R., *A Practical Theory of Programming*. Springer-Verlag, Berlin, 1993.
2. Institute of Electrical and Electronic Engineers, *IEEE Standard for Radix-Independent Floating Point Arithmetic*, Number IEEE Std 854, Institute of Electrical and Electronic Engineers, New York, 1987.
3. Institute of Electrical and Electronic Engineers, *IEEE VHDL Standard Multi-Value Logic System*, Number IEEE Std 1076-1993.5, Institute of Electrical and Electronic Engineers, New York, 1993.

Navabi, Z. "Hardware Description in Verilog: An Overview"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

81

Hardware Description in Verilog: An Overview

Zainalabedin Navabi
Northeastern University/University of Tehran

This chapter presents an overview of the Verilog hardware description language (VHDL) and the way it may be used in a design environment. Many details of the language will not be discussed in this introduction. However, this chapter will cover the general concepts of Verilog that enable the reader to understand Verilog code and develop simple hardware descriptions in this language. This chapter also facilitates the learning of the full language as presented in the IEEE standards document or any of the available texts on this topic.

In the sections that follow we will first present an overview of Verilog and elements of this language. We will then describe basic component descriptions such as flip-flops and small combinational circuits. Using the coding styles presented, we will then describe more complex hardware component descriptions such as counters and shift registers. The section that follows part descriptions will show a complete component-based design using components from the earlier sections.

Describing complete systems at the RT (Register Transfer) level requires use of data units such as registers, busses, and logic units as well as controller circuitry. While data units facilitate storage, movement, and processing of data, controllers control flow of data and times at which each of the data units handles its given data. Sections 81.2 and 81.3 of this chapter discuss Verilog coding styles for basic components. These styles apply to describing data units and their interconnections. In Section 81.4, we will present Verilog code of a state machine representing a coding style for description of controller circuitry. Together, material in Sections 81.2, 81.3, and 81.4 present coding styles for describing a complete system at register transfer level.

Finally, in Section 581., we will discuss the use of Verilog primitives and the way they can be used for describing basic gate and switch level components. System designers usually deal with more abstract

descriptions than those at the gate or switch level. However, being able to read machine generated gate level descriptions, and being able to develop glue logic at this level of abstraction is necessary for a VLSI designer using Verilog in his or her design environment.

81.1 Elements of Verilog

Constructs of the Verilog language are designed for describing hardware modules and primitives. This section presents language features related to these applications. We will show the general outline of a Verilog code for describing a hardware part and will discuss the use of various coding styles in a design.

Describing Hardware Modules

The Verilog HDL is used to describe hardware components of a system and complete systems. The main component of the language, which is a module, is the main body for describing parts and systems. As shown in Fig. 81.1, a module description consists of the keyword **module**, followed by list of ports of the hardware module, module functionality specification, and the keyword **endmodule**. Following the list of ports of a module, a semicolon separates this part from the next part in which declarations of parameters, ports, and signals of a module are specified. After the declarations, statements in a module specify its functionality. This part defines how output ports react to changes on input ports.

```
module module_name
        list of ports;

        declarations;
        specification of the
        functionality of module;
endmodule
```

FIGURE 81.1 Module specifications.

As in software languages, there is usually more than one way a component or a module may be described. Various descriptions of a component may correspond to descriptions at various levels of abstraction or various levels of detail of functionality of a module. As shown in Fig. 81.2, a module description may be at the behavioral level of abstraction with no timing details, while another description for the same component may include transistor level timing details. A module may be part of a library or predesigned library components and include detailed timing and loading information, while a different description of the same module may be at the behavioral level for input to a synthesis tool. It must be noted that not all descriptions of the same module must necessarily behave exactly the same way, nor, is it required that all descriptions behave correctly. In a fault simulation environment, faulty modules may be developed to study various forms of failure of a component.

```
module module_i
        (ports, ...);
        declarations, ...;

        behavioral
        specification
        of module_i
        with no timing
        specification;
        . . .
        . . .
endmodule
```

```
module module_i
        (ports, ...);
        declarations, ...;

        structural
        specification
        of module_i
        with transistor
        level timing
        information;
        . . .
endmodule
```

```
module module_i
        (ports, ...);
        declarations, ...;

        synthesizable
        specification
        of module_i
        for input to a
         synthesis tool;
        . . .
        . . .
endmodule
```

FIGURE 81.2 Modules with different levels of details.

TABLE 81.1 Verilog Predefined Primitives

LOGIC GATES			PULL	SWITCHES	
n-input	n-output	Tristate	GATES	Unidirectional	Bidirectional
and	buf	bufif0	pulldown	cmos	Rtran
nand	not	bufif1	pullup	nmos	rtranif0
or		notif0		pmos	rtranif1
xnor		notif1		rcmos	tran
xor				rnmos	tranif0
				rpmos	tranif1

As shown in Fig. 81.2, a module begins with a module header that includes a listing of the ports of the module. Although there may be many module descriptions for a hardware component, the same port specification is used in all such modules. Initially in a design process, a behavioral description of a system helps the designer understand the general functionality of the system under design. After the initial phase of the design process and when a design is partitioned into its subcomponents, a structural specification specifying wiring of subcomponents is used. In the last stage of design of a component, RTL (Register Transfer Level) descriptions for manual design, or synthesizable descriptions for automatic design are used. After a design is synthesized, the synthesized output is generated in form of a structural description specifying wiring of primitive components.

Often in a design process various forms of a module are created manually or automatically. These descriptions are analyzed and simulated, and their results are compared for verifying the synthesized circuit functionality or for timing verification. The Verilog hardware description language also provides constructs for generation of test benches. A test bench is a Verilog behavioral module in which a design module is instantiated and test data is applied to it. Test benches can also instantiate several modules describing the same design for functional and timing comparisons.

Primitives

Generally, hardware components are described as modules. Often, however, in a design environment certain bit level functions exist that are used as primitive gates or memory elements. These primitives are used for formation of glue logic in large designs or for complete netlist descriptions of a system. A description based on primitive structures is useful for timing analysis, test applications, and other applications where exact hardware correspondence is important. Verilog provides a set of 25 logic primitives as shown in Table 81.1. These primitives have a fixed interface with output port being listed first, and can be used in a design much the same way as a user-defined module is used. Verilog descriptions that are primarily based on gate and transistor primitives have the same properties as netlist formats used in many CAD tools as intermediate database. Because of this, many commercial Verilog synthesis tools have an option to generate their netlist output in Verilog. This, in effect, translates an abstract description in Verilog to another less abstract form.

In addition to predefined primitives, Verilog allows definition of bit level sequential or combinational user primitives. Figure 81.3 shows the basic structure of a user defined primitive in Verilog. Defining a multiplexer, a special function flip-flop or a CMOS complex gate can be done by user-defined primitives. Generating functional models for a specific ASIC library can take advantage of this Verilog utility.

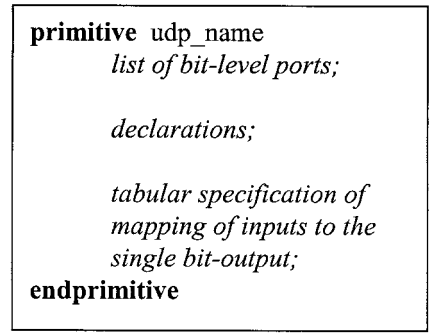


FIGURE 81.3 Verilog user-defined primitives.

81.2 Basic Component Descriptions

To demonstrate basic language features, we will show descriptions of several components in this section. Language features enable a designer to describe a component at the structural, dataflow, or behavioral levels of abstraction, which will be utilized in the descriptions that follow.

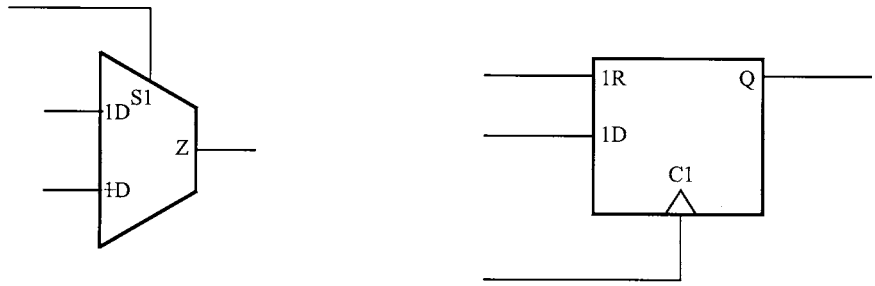
We will show Verilog code for a multiplexer, a D-type flip-flop, a full-adder, a counter, and a shift-register with enable and reset. These examples are chosen such that a wide range of behavioral and dataflow language constructs for combinational and sequential circuits are covered. For each example, the functionality will be described and a corresponding Verilog code will be presented; language constructs used in example codes will be explained. A complete design in the next section uses components of this section in a hierarchical structural description.

Small-Scale Part Descriptions

The components described in Fig. 81.4 are a multiplexer and a flip-flop. The multiplexer will be described at the dataflow level and the flip-flop at the behavioral level. The coding styles used here will be used as models for more complex hardware units. A shift-register and a counter that will be described next use the general format of the multiplexer and the flip-flop with added functionality and expanded number of bits.

The multiplexer is a 2-to-1 multiplexer with active high inputs and outputs. The flip-flop is synchronous D-type with synchronous, active, high reset input. With a delay after the rising edge of the clock if the reset is not active, the input value is clocked into the output. If the reset is active, the output becomes **0** with the falling edge of the clock.

Figure 81.5 shows the Verilog code for the 2-to-1 multiplexer of Fig. 81.4a. The code shown describes the *mux2_1* module. Following the *timescale*, which defines time unit of 1 ns and time precision of 100 ps, the first line of code specifies the name of module and its ports. Four input and output ports of *mux2-1* are named *sel*, *data1*, *data0*, and *z*. The line following this header line specifies that the first three ports are inputs, and the following line declares *z* as an output. This subsection ends declarations



(a) Multiplexer (b) Flip-flop

FIGURE 81.4 Symbolic notations for a multiplexer and a flip-flop.

```
`timescale 1ns/100ps

module mux2_1 (sel, data1, data0, z);
input sel, data1, data0;
output z;
  assign #6 z = (sel == 1'b1) ? data1 : data0;
endmodule
```

FIGURE 81.5 Verilog code for the multiplexer.

of inputs and outputs. If other signals were necessary for describing the operation of *mux2-1*, they would be declared here or anywhere in the code before they are used. It is usually best to put all the declarations at the top after the module header. All declared signals in this description, and in general, in Verilog, are of the bit type. This data type takes four logic values of **0**, **1**, **Z**, and **X**.

Following the declarations, the main body of a Verilog module describes the operation of the module. In this part, a module may be described in terms of its subcomponents, its register, and bus structure, or its behavior. In the *mux2_1* example, a continuous assign statement is used to specify output values for various input combinations. This statement specifies a 6 ns delay for all values assigned to *z*. The right-hand side of this statement selects *data1* or *data0* depending on *sel* value being binary **1** or **0**. Signals, such as *z*, to which assign is done, are presumed to be driven by their right-hand side at all times. Such signals are considered wires that do not need to hold any value. The description of the multiplexer presented here is at the dataflow level of abstraction. In this level, flow of data between buses and registers under control of the control signals is specified.

Figure 81.4.b shows schematic diagram for a flip-flop with synchronous reset and data inputs. Figure 81.6 shows the Verilog code for this small-scale part.

As in Fig. 81.5, the first line in Fig. 81.6 specifies time unit and its precision. Also, as in the description of *mux2-1*, the header line of the *flop* module specifies input and output ports of the flip-flop. Declarations following this header specify which ports are inputs and which are considered outputs of the module. An additional declaration specifies *qout* to be a signal that has capability of holding its values. This becomes clearer in the following paragraph.

```
`timescale 1ns/100ps

module flop (reset, din, clk, qout);
input reset, din, clk;
output qout;
reg qout;
  always @(negedge clk)
  begin
    if (reset) qout = #8 1'd0;
    else qout = #8 din;
  end
endmodule
```

FIGURE 81.6 Verilog code for a D-type flip-flop.

The part of the code in Fig. 81.6 that begins with the **always** keyword specifies values assigned to *qout* in response to *clk* and input changes. As specified by the statement following the at-sign, the body of this always statement is executed at the negative edge of the *clk* signal. At such times, if *reset* is true, *qout* receives 1′d0 (1-bit decimal 0), otherwise *qout* receives *din*. Value assignments to *qout* take place only on the negative edge of the clock. Therefore, in order for this output to hold its value between clock edges, it has been declared as a **reg**. A **reg** declaration is used for variables that hold values, while a net declaration that uses the **wire** keyword is used for continuous assignments and module interconnections.

In all Verilog descriptions a delay value is specified by an integer following a pound sign. In Fig. 81.6, 8 ns delay values specified on the right-hand sides of assignments to *qout* specify time delay between evaluation of the right-hand side expression and assigning them to *qout*.

A software-like, sequential coding style is used for describing the flip-flop model. A modeler writing such a behavioral description is concerned only with assigning appropriate values to circuit outputs. Neither the structure of the circuit nor the details of hardware in which data flows is of any concern. Behavioral constructs of Verilog enable designers or modelers to concentrate on *what* the hardware is supposed to do rather than *how* the task is to be done.

Medium-Scale Part Descriptions

In this section, coding styles described previously will be used to describe more complex hardware structures. A counter will be described at the behavioral level and a shift-register at the dataflow level of abstraction. We will also describe a full-adder circuit that demonstrates the use of functions in Verilog.

```
`timescale 1ns/100ps

module counter (reset, clk, counting);
input reset, clk;
output counting;
reg counting;
integer count;
parameter limit = 8;
  always @(negedge clk)
  begin
    if (reset) count = 0;
    else
    begin
      if (count < limit) count = count + 1;
    end
    if (count == limit) #8 counting = 0;
    else #8 counting = 1;
  end
endmodule
```

FIGURE 81.7 Divide-by 8, counter.

Procedural Coding

The behavioral description of the flip-flop uses procedural statements of Verilog. In general, in a procedural body of Verilog if-then-else, case and various loop constructs may be used. The example that follows shows a more general procedural coding than that of Fig. 81.6.

Figure 81.7 shows Verilog code for a counter with a synchronous *reset* input and a *counting* output. While counting falling clock edges to 8, the *counting* output stays high. When the circuit is reset or when 8 clock edges are counted, *counting* becomes **0**.

The Verilog code of the counter begins with module name and port list. Input and output declarations as well as declaration of *counting* as a **reg** follow the module heading. The *counting* variable is to hold values between activations of assignment of values to this signal, and, therefore, it is declared as **reg** to have the value holding property.

The declared integer *count* to keep the count and parameter limit to set the count limit. As in the description of *flop*, an **always** statement that becomes active on the negative edge of *clk* encloses the statements specifying behavior of the counter. Following the keyword **begin**, an if-then-else statement increments *count* if *reset* is not active and if *count* has not reached the limit. Another if-then-else statement sets the *counting* output of the counter to **1** if the count limit of 8 has been reached. Assignments to *counting* are delayed by 8 ns, specified by integers following pound signs.

User-Defined Functions

In the dataflow description of the multiplexer, continuous assignment statements were used to assign right-hand side expressions to net outputs. Use of functions can eliminate the need for complex right-hand side expressions in behavioral and dataflow descriptions. An example here will demonstrate definition and use of functions.

Figure 81.8 shows Verilog codes for a full-adder. The *fulladder* module describes a combinatorial circuit with three inputs and two outputs. A continuous assign-statement assigns the two-bit output of the *fadd* user-defined function to concatenation of *sum* and *cout*. The *sum* output receives the left-most output of *fadd* and *cout* receives bit 0 of this function. The *fadd* function is defined in the module of Fig. 81.8 as a function with a two-bit output. An expression concatenating two Boolean expressions for the two outputs of the function forms the right-hand assignment to function identifier, *fadd*. On the right-hand side of *fadd*, the symbol ^ is used for exclusive-or, & is used for anding, and the vertical bar is used for Boolean or operation.

```
`timescale 1ns/100ps

module fulladder (a, b, cin, sum, cout);
input a, b, cin;
output sum, cout;
  assign {sum , cout} = fadd (a, b, cin);

  function [1:0] fadd;
  input a, b, c;
    fadd = {(a ^ b ^ c) , ((a & b) | (a & c) | (b & c))};
  endfunction
endmodule
```

FIGURE 81.8 Full-adder description.

```
`timescale 1ns/100ps

module shifter (sin, reset, enable, clk, parout);
input sin, reset, enable, clk;
output [7:0] parout;
reg [7:0] parout;
  always @(negedge clk)
    parout = (reset) ? 8'd0 : ((enable) ? {sin, parout [7:1]} : parout);
endmodule
```

FIGURE 81.9 Shifter Verilog code.

Conditional Operator

In the dataflow description of the multiplexer, conditional operator was used on the right-hand side of a continious assignment. The example that follows will show the use of condition operator in sequential blocks. We will also show how clock control can be incorporated into an assignment statement with complex right-hand side operations.

For this part, a shift-register as shown in Fig. 81.9 is described in Verilog. Following module header and declarations, an always block, the flow into which is controlled by the falling edge of the clock (determined by *negedge clk*), encloses a sequential assignment to *parout* shifter parallel output. The value assigned to *parout* is determined by nested expressions using Verilog conditional operator (?:).

On the falling edge of the clock if *reset* is **1**, an 8-bit decimal 0 is placed on parout. If reset is **0**, a second conditional operator checks for *enable* being **1**. If *enable* is **1**, then *sin* is left concatenated with previous value of parout and will be assigned to parout as its new value. This concatenation causes *parout* to be shifted to the right by one bit. On the other hand, if *enable* is **0**, the second conditional operator in Fig. 81.9 will place the previous value of *parout* back on itself, causing the shift register state to remain unchanged.

81.3 A Complete Design

The previous section presented several styles for describing small- and medium-scale parts. In a complete design, these parts are wired to form a functional unit. Verilog provides constructs for wiring primitives and modules. This form of describing hardware is most often referred to as structural, in which components to be wired are instantiated and their connections are specified by **nets** that may be explicitly declared. The example that illustrates this style of coding is a sequential adder that is implemented using parts described in Section 81.2.

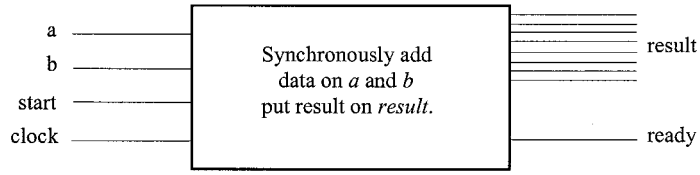


FIGURE 81.10 Serial adder.

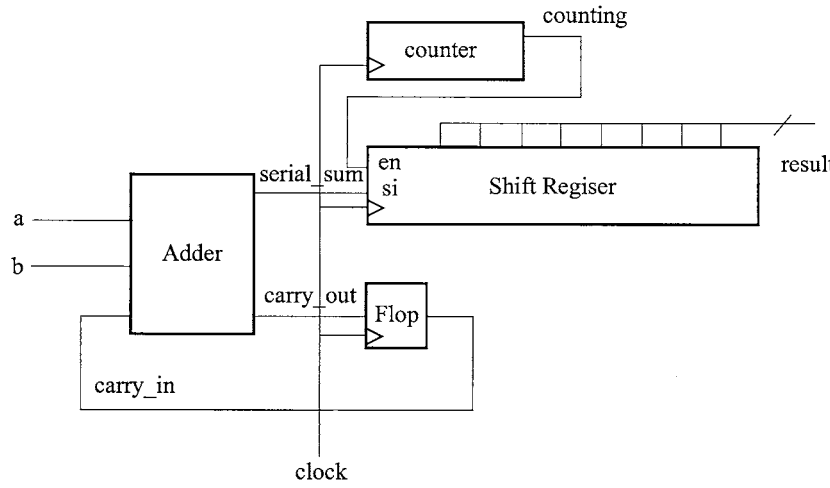


FIGURE 81.11 Implementing the serial adder.

The design we have selected is a sequential adder shown in Fig. 81.10. The circuit has two data inputs *a* and *b*, and a *start* input. The outputs of the circuit are *ready* and *result*. The circuit is a synchronous sequential circuit that uses the *clock* signal for the synchronization clock.

Serial data are synchronized with *clock* and appear on *a* and *b* inputs. Valid data bits on these inputs design after a pulse on start, and the ordering of these bits is from least to most. After eight clock pulses, the *result* output of the circuit will have the add result of the serial data are *a* and *b*. The *ready* output becomes **1** to indicate that one data set has been collected and add result is ready. This output remains **1** until another synchronous *start* pulse is observed.

Figure 81.11 shows an implementation of the serial adder using parts described in Section 81.2. An adder is used for adding serial bits, a flip-flop saves the value of carry for a next add stage, a shift-register stores bit results, and a counter keeps count of bits that are added. When the *counting* output becomes **0**, add result is ready of the *result* output.

Figure 81.12 shows the structural description of *serial_adder* consisting of interconnection of its four subcomponents. The module declaration consists of port list describing inputs and outputs of the serial adder. Declarations in the structural description of Fig. 81.12 declare inputs and outputs of *serial_adder* as well as wires for interconnection of components of which the serial adder is composed. Following the declarations, the *serial_adder* module includes instantiation of *fulladder*, *flop*, *counter*, and *shifter* modules. Instantiation refers to naming a module within another module for use as a subcomponent. Every instantiation begins with the name of module that is being instantiated, followed by an arbitrary label and a list of module port connections.

The format used here for module port connections is according to an ordered list. Signals in port connections of an instantiated module, e.g., *fulladder* instantiation in Fig. 81.12, connect to actual ports of the module, e.g., *fulladder* module in Fig. 81.8, in the order that ports appear in port-list of the module.

```
`timescale 1ns/100ps

module serial_adder (a, b, start, clock, ready, result);
input a, b, start, clock;
output ready;
output [7:0] result;
wire serial_sum, carry_in, carry_out, counting;
  fulladder u1 (a, b, carry_in, serial_sum, carry_out);
  flop u2 (start, carry_out, clock, carry_in);
  counter u3 (start, clock, counting);
  shifter u4 (serial_sum, start, counting, clock, result);
  assign ready = ~ counting;
endmodule
```

FIGURE 81.12 Shifter Verilog code.

Within a module, variables in the port list of the module as well as wires that can be explicitly declared in the module may be connected to ports of an instantiated module. For example, signals *a*, *b*, *start*, *clock*, *ready*, and *result* that appear in port list of *serial_adder* in Fig. 81.12, as well as declared wires in this module, i.e., *serial_sum*, *carry_in*, *carry_out*, and *counting*, can be used for connections to ports of *fulladder*, *flop*, *counter*, and *shifter*. Through instantiation of full-adder, wires visible in *serial_adder* module are connected to ports of the *fulladder*. Elaborating on this connection mechanism, consider the *carry_in* signal that connects to the *cin* input of *fulladder* and at the same time is used in the port map of *flop* to connect to its second port. The *carry_in* wire causes the *cout* output of *fulladder* to connect to *din* of *flop*.

81.4 Controller Description

The main purpose of this introductory tutorial has been the presentation of an overview of Verilog. A complete design usually consists of a data unit and a controller. Because of the size of our design examples, we were not able to discuss controllers and Verilog coding styles for them. A sequence detector is a state machine that represents a simplified controller and represents most characteristics of complex controller circuits.

This section shows Verilog coding style for describing a simple synchronous Moore sequence detector. A sequence detector has several states the transitions between which are decided by input values. Outputs of a detector are issued when one of its states becomes active. All state transitions in a sequence detector are synchronized with a main clock. Verilog code for the simple sequence detector in Fig. 81.13 will be presented here.

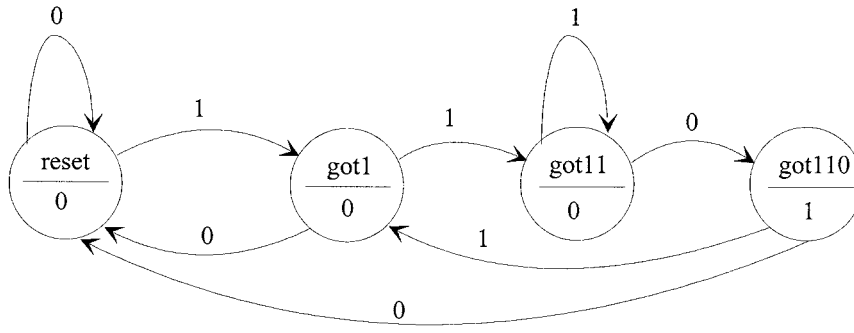


FIGURE 81.13 Moore machine state diagram.

```
module moore_110_detector (x, clk, z);
input x, clk;
output z;
reg [1:0] current;
parameter [1:0] reset = 0, got1 = 1, got11 = 2, got110 = 3;
  initial current = reset;
  always @(posedge clk)
  begin
    if     (current == reset) current = x ? got1 : reset;
    else if (current == got1) current = x ? got11 : reset;
    else if (current == got11) current = x ? got11 : got110;
    else if (current == got110) current = x ? got1 : reset;
  end
  assign z = (current == got110) ? 1'b1 : 1'b0;
endmodule
```

FIGURE 81.14 Moore machine Verilog description.

```
always @(posedge clk)
  begin
    . . .
    else if (current == got1)
        begin
            if (x) current = got11;
            else current = reset;
        end
    . . .
endmodule
```

FIGURE 81.15 Alternative coding style for Moore detector.

The Moore machine in Fig. 81.13 searches on its *x* input for the 110 sequence. When this sequence is detected in three consecutive clock pulses, the output (*z*) becomes **1** and stays **1** for a complete clock cycle. States are named according to bits detected on the *x* input. Starting in the *reset* state, it takes at least three clock pulses for the state machine to get to the *got110* state, in which output becomes **1**.

The Verilog behavioral description for this machine is shown in Fig. 81.14. The list of ports contain *x*, *clk*, and *z*, which match those in the state diagram of Fig. 81.13. A two-bit variable is declared as **reg** to hold the current state of the machine. Four two-bit parameters define state names and their binary assignments. In an **initial** block, the present state is set to the *reset* state. The main flow of the state machine is implemented by an **always** block the flow into which is controlled by the positive edge of the clock. In this statement, **if-else** statements are used to check for each of the four cases of the *current* state. For each state that matches the current state a conditional operator is used to set the current state to the next active state of the machine depending on the value of the *x* input. The condition operators can be expanded to **if-else** statements. For example, code corresponding to state *got1* of the machine could be written as shown in Fig. 81.15. Functionality of the code in Fig. 81.14 would be no different if codes for all four states were replaced with style used in Fig. 81.15.

81.5 Gate and Switch Level Description

As discussed in the previous sections, system designers are mostly concerned with high-level descriptions that can be easily developed, understood, and synthesized. Gate and switch level descriptions are also part of Verilog and are used for cell modeling, glue logic, and machine generated codes. At this level,

```
`timescale 1ns/100ps

module aoi (a, b, c, z);
//
input a, b, c;
output z;
wire w;
  and #(3, 4) n1 (z, ~a, w);
  nand #(3, 4) n2 (w, b, c);
endmodule
```

FIGURE 81.16 A gate level logic function using predefined primitives.

primitives shown in Table 181. are used in a design. These primitive structures are instantiated in a module describing a gate-level circuit or they may be used along side with other coding styles presented in the previous sections of this chapter. In this section we will show modules that consist of pure gate and switch level codes using Verilog primitives.

A Gate Level AOI

Figure 81.16 shows description of an and-or-invert gate using **and** and **nand** primitives. As in all the descriptions we have presented, the module begins with a list of inputs and outputs followed by declaration of these ports. The *aoi* module in this figure has 3 inputs and one output. The **nand** output of *b* and *c* inputs is used by the **and** structure to produce the circuit output. The declared wire, *w*, connects the output of the **nand** gate to the input of the **and** gate. Declaring such intermediate wires is not mandatory in Verilog. Instantiations of these two primitives are done much the same way as modules are instantiated with a few differences that will be discussed here.

Unlike module instantiations, the names used with instantiation of primitives are not required. In our *aoi* module, gate names *n1* and *n2* are optional. Another difference between primitive instantiations and module instantiations is the way ports are ordered. Module ports are determined according the module definition. A user can order inputs and outputs of a module in any order that he or she prefers. In predefined primitives (the same also applies to user-defined primitives), the output of the gate is always first in the port list followed by the inputs. In the primitives we have used in this design any number of inputs can be used. Another feature of primitive instantiations is the specification of delay values. In our *aoi* example a **0** to **1** propagation delay of 3 ns and a **1** to **0** propagation delay of 4 ns are used for each of the gates used in this design. Basic gate primitives take two delay values, while gates with three-state outputs take three delay parameters.

A Switch Level NAND

Shown in Fig 81.17 is a CMOS NAND using n-type and p-type transistors. Although the complete functionality of this circuit is available in a **nand** primitive, there may be cases that the two timing parameters allowed by this primitive do not provide the necessary timing accuracy required in a design. Furthermore, the **nand** predefined primitive only produces **0** and **1** output values that may not be sufficient for designers interested in charge sharing issues and three-state values. In such instances, Verilog allows description of a system at the switch level using switch primitives shown in Table 1. Simulation at the switch level provides detailed information but is slow. In large designs switch level simulation is avoided as much as possible. This level of modeling and simulation is useful when designing cells and evaluating their detailed functionality.

Figure 81.18 shows *cmos_nand* module that corresponds to diagram of Fig. 81.17. Built-in structures **nmos** and **pmos** are used with three delay values. Switches in Verilog can use up to three delay values.

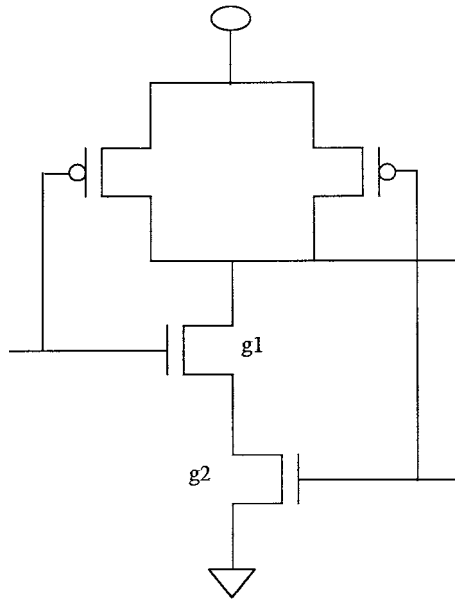


FIGURE 81.17 Switch level CMOS NAND.

```
`timescale 1ns/100ps

module cmos_nand (a, b, z);
//
input a, b;
output z;
supply1 vdd;
supply0 gnd;
wire im1;
  nmos #(0, 0, 4)
    g1 (z, im1, a),
    g2 (im1, gnd, b);
  pmos #(0, 0, 5)
    g4 (z, vdd, a),
    g5 (z, vdd, b);
endmodule
```

FIGURE 81.18 Verilog code for a switch level CMOS NAND.

The first two parameters are for **0** to **1** and **1** to **0** transitions, while the third timing parameter is the delay when the switch opens. Labels *g1*, *g2*, *g4*, and *g5* are optional, and we have used them for making a better correspondence between diagram of Fig. 81.17 and code of Fig. 81.18. The last field of switch instantiations in Fig. 81.18 lists output, input, and control terminal connections, respectively. Because switches are unidirectional, the output-input connections are based on the direction of flow of data out of the switches. For n-type transistors, gnd terminal is input pushing 0 into the switch, and for p-type transistors, vdd terminal is considered as input pushing 1 into the switch. Control connections (transistor gate terminals) are the last of the three terminal connections.

```
`timescale 100 ns/1ns

// Verilog test bench and stimulus for
cmos_nand
module test_cmos_nand;
reg a, b;
wire z;
  cmos_nand n (a, b, z);
  initial  begin
    a = 0; b = 1;
    #25 a = 1; #11 b = 0;
    #10 a = 0; #10 b = 1;
    #11 a = 1; #11 b = 0;
    #30 $stop;
  end
endmodule
```

FIGURE 81.19 Verilog code for a test bench for testing cmos_nand module,

81.6 Test Bench Descriptions

Verilog provides constructs that can be used for applying data to a design module and monitoring its response. A module in which this is done is referred to as a test bench. To show how a test bench is developed and the tasks performed by such modules, a simple test bench is developed for the CMOS NAND gate of Fig. 81.18. The test bench is shown in Fig. 81.19. This is a Verilog module that instantiates the module corresponding to the circuit under test and uses behavioral Verilog constructs to generate data for the inputs of the circuit. In our simple example, an **initial** block applies values to the inputs of cmos_nand module. Assignment of values is timed to allow for the internal propagation delays of the NAND gate. The time control statements that begin with a number sign, e.g., #25, suspend the flow of program for the amount of time given before the next statement is executed. An **initial** block is similar to an **always** block but only runs once and terminates when program flow reaches its last statement. The **$stop** statement is a Verilog system task that stops the simulation when it is reached. Simulation of test_cmos_nand module ends after 108 ns from the start of simulation.

81.7 Summary

This overview introduced the main concepts of the Verilog Hardware Description Language. Entities needed for description of designs were presented. We have shown ways of describing small components, medium-size parts, complete systems, and controller descriptions. In addition we have presented gate and switch level descriptions in Verilog that make this language a very effective tool for modeling and design of VLSI circuits. Descriptions shown here are general in style and represent most coding styles required in a large design environment.

References

IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language, IEEE Std 1364-1995, IEEE, Inc., New York, 1996.

Navabi, Z., *Verilog Digital System Design*, McGraw-Hill Publishing, New York, 1999.

Navabi, Z., *VHDL: Analysis and Modeling of Digital Systems*, 2nd ed., McGraw-Hill Publishing, New York, 1998.

Palnitkar, S., *Verilog HDL: A Guide to Digital Design and Synthesis*, Prentice-Hall, Upper Saddle River, NJ, 1996.

Smith, D. J., *A Practical Guide for Designing, Synthesizing and sim ulating ASICs and FPGA s Using VHDL or Verilog*, Doone Publications, June 1996.

Thomas, D. E. and P. R. Moorby, *The Verilog Hardware Description Languag e*, 3rd ed., Kluwer Academic Publishers, Norwell, MA, 1996.

"Section I: VLSI Technology"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

I

VLSI Technology

Krishna Shenai
University of Illinois

"Section II: Devices and Their Models"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

II

Devices and Their Models

John Choma, Jr.
University of Southern California

"Section III: Circuit Simulations"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

III

Circuit Simulations

Wai-Kai Chen
University of Illinois at Chicago

"Section IV: Amplifiers"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

IV

Amplifiers

Rolf Schaumann
Portland State University

"Section V: Logic Design"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

V

Logic Design

Saburo Muroga
University of Illinois at Urbana-Champaign

"Section VI: Memory, Registers, and System Timing"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

VI

Memory, Registers, and System Timing

Bing J. Sheu
Avant! Corporation

"Section VII: Analog Circuits"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

VII

Analog Circuits

Bang-Sup Song
University of California at San Diego

"Section VIII: Miicroprocessor and ASIC"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

VIII

Microprocessor and ASIC

Steve M. Kang
University of Illinois at Urbana-Champaign

"Section IX: Test and Testability"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

IX

Test and Testability

Nick Kanopoulos
Atmel, Multimedia and Communications

"Section X: Compound Semiconductor Digital Integrated Circuit Technology"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

X

Compound Semiconductor Digital Integrated Circuit Technology

Stephen I. Long
University of California at Santa Barbara

"Section XI: Design Automation"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

XI

Design Automation

Allen M. Dewey
Duke University

"Section XII: Algorithms and Architects"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

XII

Algorithms and Architect

Kung Yao
University of California at Los Angeles

"Section XIII: Design Languages"
The VLSI Handbook.
Ed. Wai-Kai Chen
Boca Raton: CRC Press LLC, 2000

XIII

Design Languages

David L. Barton
Intermetrics, Inc.