REAL-TIME SIMULATION TECHNOLOGY FOR MODERN POWER ELECTRONICS

Hao Bai
Chen Liu
Dusan Majstorovic
Fei Gao

Real-Time Simulation Technology for Modern Power Electronics

This page intentionally left blank

Real-Time Simulation Technology for Modern Power Electronics

Hao Bai
Northwestern Polytechnical University, Xi’an, China

Chen Liu
Zhengzhou University, Zhengzhou, China

Dusan Majstorovic
Typhoon HIL, Inc., Somerville, MA, United States

Fei Gao
University of Technology of Belfort-Montbeliard, Belfort, France

ELSEVIER

ACADEMIC PRESS
An imprint of Elsevier

Academic Press is an imprint of Elsevier
125 London Wall, London EC2Y 5AS, United Kingdom
525 B Street, Suite 1650, San Diego, CA 92101, United States
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, United Kingdom

ISBN: 978-0-323-99541-2

Publisher: Joseph P. Hayton
Acquisitions Editor: Rachel E. Pomery
Editorial Project Manager: Helena Beauchamp
Production Project Manager: Kamesh R.
Cover Designer: Miles Hitchen

Typeset by MPS Limited, Chennai, India

Contents

Preface

Power electronics plays increasingly essential roles in power systems and transportation applications. The high penetration of the power electronic system generates great demands in high-fidelity digital tools for the design, validation, and maintenance in the full life cycles of power electronics equipment. Among these tools, real-time simulation is the most efficient and effective one that can fulfill these tasks and is well accepted by the power electronics practitioners. Real-time simulation creates a replica of the power electronics in the digital world running with the same rate as the physical entities, creating the possibility of interaction of the virtual model with the real world. The most common use of real-time simulation is hardware-in-the-loop (HIL) testing, where the power electronics are simulated in real time, connecting the surrounding system via the interface circuit to constitute a semiphysical testing environment. HIL provides a fast, safe, and low-cost way to validate the performance of power electronics and its controller, which significantly speeds up the development cycle and reduces the time to market. In the commissioning stages the real-time simulation serves as the digital twin of the power electronics system enabled by the reality—virtual interaction. State monitoring, prognosis, health management, optimal operation, and maintenance can be implemented. In addition, due to the real-time characteristics, the real-time simulation technique is favorable and can also be applied to the simulation when designing large-scale power electronics system to break through the constraints between the simulation granularity and the speed.

Power electronics practitioners are usually well familiar with commercial real-time simulators manufactured by Typhoon HIL, Opal-RT, Speedgoat, RTDS, etc. With the development of modern power electronics a more complex topology, a larger scale, and a higher switching frequency will be distinguished features, which spawn the need to better use or customize such a digital tool to facilitate their work. However, real-time simulation involves multidisciplinary knowledge, including power electronics modeling, circuit simulation, numerical analysis, linear/nonlinear system theory, embedded system, electronics circuit design, and power electronics control. There are barriers for the students, researchers, and engineers to understand, get familiar, and grasp the real-time simulation. In addition, there is no published book yet that systematically introduces this technology, which is adverse to popularizing and exploring the most valuable features of real-time simulation.

This book aims at the systematic introduction of the real-time simulation technologies for modern power electronics, including the modeling approaches, the numerical solvers, the real-time implementations, and the applications and the outlooks of future technologies. The book starts with a basic introduction, state of the art, and the challenges of real-time simulation (Chapter 1). The modeling approaches for the real-time simulation are presented in detail, including the review of the operating mechanism and characteristics of power electronics devices (Chapter 2), followed by the modeling of power electronics devices (Chapter 3), the modeling of circuit components (Chapter 4), and the modeling of the power electronics converters (Chapter 5). The numerical solver changes the power electronics converter models to the discrete-time representation, which is solved step by step by the computing unit. Both the general numerical methods and the paralleled solving strategies dedicated to the power electronics simulation are presented (Chapter 6). As one of the most significant uses of real-time simulation the HIL simulation of a power electronics system is analyzed with regard to its time constraints and interface issues (Chapter 7). Afterward the processor-based and field programmable grid array−based power electronics real-time simulation is presented with detailed descriptions of the structures, the implementation methods, the optimization approaches, and the illustrative case studies (Chapters 8 and 9). Meanwhile the real industrial applications of power electronics real-time simulation are demonstrated in an electric vehicle application, a terrestrial microgrid, and a battery energy storage system (Chapter 10). Furthermore, recent advances (Chapter 11) and future outlooks (Chapter 12) are introduced to provide a deeper and wider view of this subject. This book covers the general knowledge, professional skills, and possible guidance in real-time simulation of power electronics and provides the readers with an easy way to know, learn, master, and be creative in this field.

The authors would like to sincerely thank Dr. Caio Osório from Typhoon HIL for his valuable contributions to this book. The authors would also like to thank Mr. Gang Huang, Ms. Nan Wang, and Ms. Jiaxin Tang from Northwestern Polytechnical University and Dr. Qian Li from the University of Technology of Belfort-Montbeliard for their contributions. The authors welcome the valuable suggestions and advice from the readers regarding the book.

Hao Bai, Chen Liu, Dusan Majstorovic and Fei Gao

Roles and challenges of power electronics real-time simulation

1

1.1 Introduction

Power electronics is an interdisciplinary engineering branch, covering semiconductor devices, power conversion circuits, and associated control strategies. Power electronic circuit design requires accurate tools in evaluating the performance. Because of the enormous complexity of modern power electronic systems, simulation is essential and can provide information about circuit performance that is costly or sometimes impossible to obtain with laboratory prototype measurements. However, traditional simulation is restricted to virtual testing and theoretical verification, with limited coverage of the real device features. Real-time simulation has revolutionized the power electronics system testing chain by bridging the gap between the development of the physical device and the deployment of a simulated virtual environment. It enables the device under test to be thoroughly tested and improved during the early development stages, thereby shortening the time to market of power electronic products. However, real-time simulation of power electronics proves to be technically challenging since the requirements for simulation performance approach the boundaries of the current simulation technology.

In this chapter, we will first briefly introduce the background of the power electronic converter and the importance of the simulation in the design of power electronic converters and their controllers. The definitions and the applications of real-time simulation are presented, and the state of the art is briefly reviewed. The main challenges of the power electronic real-time simulation are discussed, and the scope and the structure of this book are sketched.

1.1.1 What is power electronics?

Power electronics is the technical field that uses power semiconductors to implement efficient electric power conversion [1]. Dr. William E. Newell defined power electronics for the first time in June 1973 as a technology that is interstitial to all three of the major disciplines of electrical engineering, electronics, power, and control, and used a triangle shown in Fig. 1.1 to illustrate it [2].

The circuit that converts the electrical power is called the power electronic converter. The power electronic converters can be classified into four types, that is, the DC-DC converter, the DC-AC converter (inverter), the AC-DC converter (rectifier), and the AC-AC converter. The energy flows of each type of power electronic converter can be either unidirectional or bidirectional.

Real-Time Simulation Technology for Modern Power Electronics. DOI: https://doi.org/10.1016/B978-0-323-99541-2.00003-X

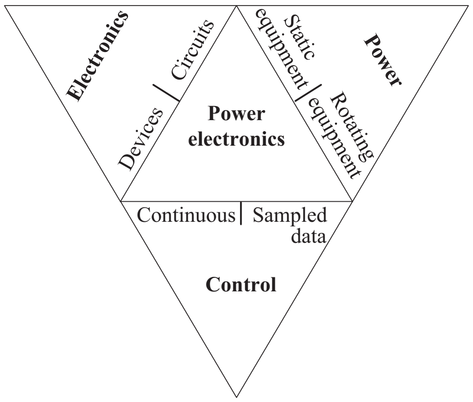


Figure 1.1 The definitions of power electronics.

Power electronic converters have advantages in converting efficiency, power density, reliability, and cost. They are widely applied in almost all industrial sectors, especially in power systems, electrified transportation, information technology, aeronautics and astronautics, renewable energy systems, and domestic appliance. With the shift to e-mobility and ever-increasing energy efficiency requirements, there is a huge market demand for ever more capable power electronic converters as well as power electronic systems. The development efficiency will determine the time to market of the power electronic products. Model-based simulation technology proved to be an efficient, low-cost, and safe tool in the design, prototyping, and validation of power electronic converters and the controllers.

1.1.2 Role of simulation

A model is the representation of a physical object or an entire system. Through the built model, the simulation can be performed to investigate how a system would operate under certain conditions. The power electronic converters combine the power semiconductors, the electric circuits, and the control to perform the power conversion. The simulation thus has different scopes corresponding to the specific target, and multiple levels of detail are considered.

Based on the adopted models, the simulation can be classified into device level and system level. Just as its name implies, the device-level simulation focuses on the individual components' behaviors such as the power semiconductors and the passive components or their behaviors interacting with the converter's performances. The device-level simulation can adopt the physics-based model to perform the multiphysics simulation that combines the structural mechanics, heat transfer, electromagnetics, acoustics, fluid flow, chemical reaction phenomena, and so on that can be described with partial differential equations (PDEs). The typical software is COMSOL and Ansys, and the applications focus on the power semiconductor electrical and thermal designs, module packaging, and failure mechanism analysis.

Behavioral models can also be used in the device-level simulation instead of the complex multiphysics models to reduce the computational burden. Some of the behavioral models are extracted from the physics-based models as the equivalent circuit models integrated into the simulated circuits, while some are built based on the terminal characteristics of the devices without diving deep into the physical mechanisms. Most of commonly used simulation tools can be classified into this category, such as Saber, PLECS, LTspice, and so on. The characteristics of the individual components can be user-defined, and both the device-level behaviors and the system-level integrations can be simulated. The device-level simulation with behavioral models is suitable for verifying the design, validating circuit parameters, tuning, optimizing the design, and performing reliability analysis. The worst-case conditions and corner cases, and the faults and stress effects can be simulated and evaluated.

However, to be able to achieve faster simulation, the device-level power electronic models are not suitable anymore, and the system-level models are used in some simulation tools. The system-level models refer to the situation where the power electronic components are represented by their ideal models or piecewise linearized models. In the system-level simulation the concept of the designs can be validated fast at the beginning of the development stage since not many details are yet defined. Moreover the system-level simulation is suitable for evaluating the system behavior referring to the topology design, control validation, system response analysis, fault diagnosis and protections, system management, and optimizations. Software such as PSCAD, PSIM, and MATLAB Simulink are usually used in the system-level simulation.

The system-level simulation provides a high simulation speed at the cost of the simulation fidelity. The device-level simulation has advantages in the simulation fidelity but can be very slow, especially when the scale of the power electronic system is large. As for the power electronic developments, the different levels of simulation can be selected flexibly to accommodate different design requirements, as shown in Fig. 1.2. We can classify the simulation requirements into three

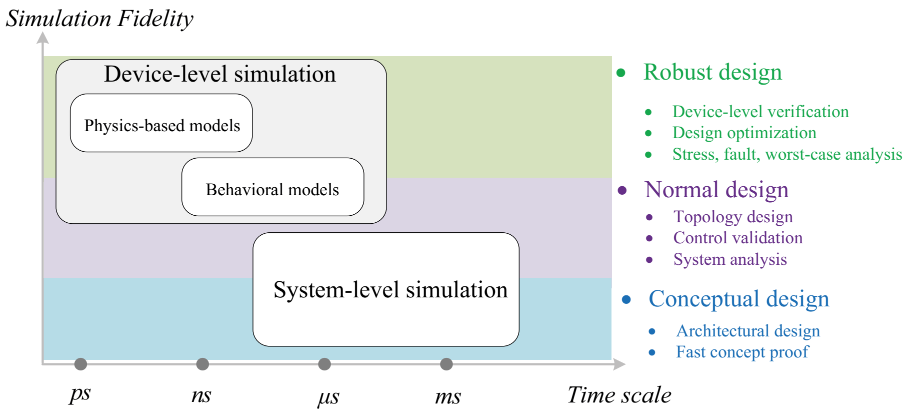


Figure 1.2 Levels and targets of power electronic simulations.

abstraction levels, namely, conceptual design, normal design, and robust design. When the design is in the concept phase the system-level simulation can effectively support the architectural design and allow fast iterations of the concept. Meanwhile the system requirements can be specified, which transform the concept into a set of design objectives. In the normal design stage the topology is evaluated, the control is designed, and the system integrations is tested virtually by using both the system-level simulation and the behavioral model-enabled device-level simulation. The levels of simulation can be chosen based on the design scope. For example, if the driving performances of the power semiconductors or the parasitic parameters need to be considered, the device-level simulation should be performed. When evaluating the control performances the system-level simulation is a good choice. In the robust designs stage the power electronics evaluated at the device level will enhance the design by considering the multiphysics behaviors, such as the EMI, the losses, and the thermal behaviors. The design can thus be optimized according to its application scenarios. Moreover, the fault, device stresses, and the worst case can also be analyzed through the device-level simulation to improve the design robustness.

1.1.3 Role of real-time simulation

To perform the simulation in the time domain, the mathematical models of the physical system are solved numerically in a computer, and the continuous behaviors of the system are approximated by the numerical results in discrete time. There are two time axes in the simulation. The first one is the simulation time of the model, which is defined by the time interval between the start time and stop time of the simulation. It refers to how long the system behaviors are simulated. The simulation time step is the amount of time by which the simulation advances in discrete steps. Based on whether the time step is fixed or variable the simulation can be classified into the fixed-time-step simulation and variable-time-step simulation.

The second one is the execution time, which is the real time that the simulation takes to compute the model with respect to the defined simulation time. For example, running a simulation for a 10 s time interval may take 1 min for a computer to finish. The execution time of a simulation not only depends on the simulation time but also depends on the model's complexity, the step size, and the computer's performance.

When implementing the simulation on a PC, people usually do not want the execution time to be too long and search for the speed-up of the simulation by means of simplifying the model, optimizing the solver, or switching to a more powerful computer. In these cases the execution time can be either shorter or longer than the simulation time, and no constraints are imposed on the execution time.

As for the physical system, there is only one time axis, that is, the real time. Therefore, if we want to perform the simulation running at the same rate as the physical system, the execution time should be equal to the simulation time to make the computer model run in real time. That is to say, the execution time corresponding to each simulation time step should be strictly equal to the simulation time step. This kind of simulation is called the real-time simulation. By considering the input and output signal conversion time of the real-time simulation, the model should be solved within a period even less than the simulation time step, which imposes strict time constraints to the model computation (Fig. 1.3).

The ability to run a model in real time enables the real-time simulation to interact with the physical world. One typical application of real-time simulation is to perform the hardware-in-the-loop (HIL) simulation. In the HIL simulation, one part of the system is simulated in the digital (virtual) world and the other part is a physical entity and they compose a semiphysical testing environment. The simulation interacts with the physical world via an interfacing system, such as digital I/O, analog I/O, power amplifier, and communication. The HIL is a very useful tool in the early development stages of the power electronic system. First, it stimulates the paralleled development flows of power electronics. The controller can be tested in parallel to the development of converter without the need of waiting for the converter prototype. The power electronic converter can also be tested in an integrated environment even when the other converters are not well developed. Second, it provides a time-efficient, low-cost, and safe testing tool to validate the power electronics and its controllers. One can perform the testing without coordinating the experimental equipment, sites, and schedules with other departments. One can also perform different kinds of testing for different products with a real-time simulator, eliminating the need in building various test benches. The HIL creates a safe testing environment that enables exploring the system boundaries, testing the ultimate performance, and emulating the fault condition.

In addition to the HIL applications, real-time simulation plays an important role even after the power electronic system is put into operation. The real-time simulation technology can also be applied in faster-than-real-time simulation to predict the system behaviors for better system management and maintenance. The real-time simulation-enabled digital twin technology can be used in monitoring, prognosis, and health management of power electronics. Moreover the real-time simulation technology can be used to speed up offline simulation, especially when large-scale topology is targeted and long-term performance is of concern. To summarize the real-time simulation provides a digital development tool for the full life cycle of power electronic products (Fig. 1.4).

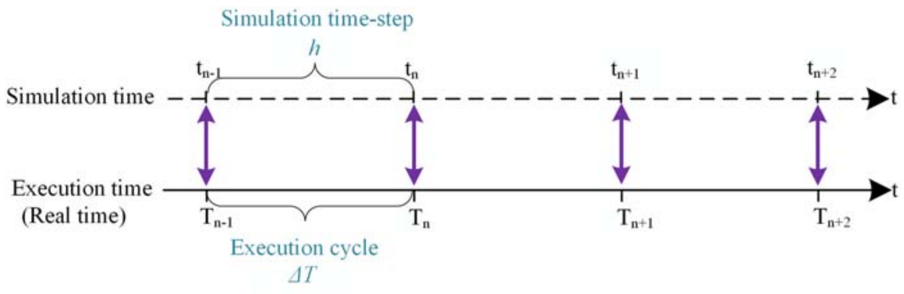


Figure 1.3 The time constraint in real-time simulation.

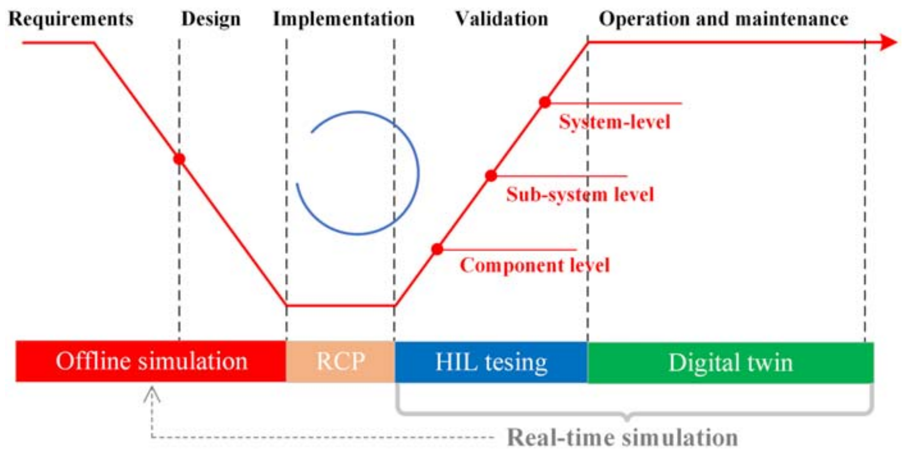


Figure 1.4 Real-time simulations in the full life cycle of the power electronic system.

1.1.4 State-of-the-art power electronics real-time simulation

There are many different types of real-time simulators available in the market, targeting power electronic applications. The mainstream manufacturers such as Typhoon HIL, RTDS, Opal-RT, Speedgoat, and Plexim have released different kinds of HIL platforms for various application fields. The real-time simulator usually has a powerful computing unit to solve the power electronics models in real time. In addition the simulator is equipped with an I/O interface needed to connect to external equipment. Real-time simulation software is usually developed by manufacturers for their own simulators to provide tools for designing and implementing the models, monitoring the simulations, and analyzing the simulation results.

The computing unit is the core of real-time simulator. It solves the power electronic models in real time. The performances of the computing unit determine the size and the topology of models that can be simulated in real time and how accurate the simulation will be. There are two different kinds of computing units used in the real-time simulator, that is, processor (e.g., CPU) and FPGA. The processor-based real-time simulation appeared first, and at that time, the FPGA is usually used as the interface between the processors and I/Os. With the development of FPGA devices, the scale of FPGA and its ability to do arithmetical operations have greatly improved. FPGAs are today commonly used as computational units and offer some unique advantages. Nowadays, processors and FPGAs coexist in the state of the art and they are typically oriented toward different applications. Sometimes the cosimulation structures are used to make the best of both.

The modern real-time simulator dedicated to the power electronic system usually has a minimum simulation time step smaller than 1 μs. For example the commercial Typhoon HIL 404 real-time simulator can realize a minimum simulation time step of 200 ns [3]. In the customized real-time simulation, many efforts have been made to further reduce the simulation time step. A real-time simulation method of a high-switching-frequency power converter is proposed in [4] based on the state-space model, where a 75 ns time step is implemented for simulating a Boost converter and an 80 ns time step is for a two-level inverter. A matrix-inversion technique is proposed in [5] for FPGA-based real-time simulation of power converters where an average of 36 ns time step is achieved in simulating a three-phase back-to-back converter. A predictive real-time simulation method is proposed in [6] to achieve a 25 ns time step in simulating an AC-DC-AC topology. A direct mapped method is proposed in [7] for the real-time simulation of high-switching frequency resonant converters, where a minimum 25 ns simulation time step is achieved. Although the customized real-time simulation can realize an ultra-small time step, they only focus on specific topology where special treatment can thus be used to optimize the computing efficiency. As for a generic real-time simulator, it is still hard to realize a time step lower than 100 ns.

The real-time simulation can also be classified into the system-level and the device-level according to the adopted models. In the system-level simulation the switches are usually represented by their steady-state turn-on and turn-off states. Therefore power electronic circuits can be formulated by various approaches, such as the state-space (SS) method, the nodal analysis (NA) method, and so on. Considering the computational efficiency in real-time simulation, the paralleled modeling and simulation techniques are used to split the large topology into different independent parts, which further constitute a paralleled computing structure to implement a nanosecond-level real-time simulation. Two categories can be identified based on the partitioning principles, the electric network partitioning methods, and the delay-based network decoupling methods. In the former one, all the state variables are discretized by the same implicit integration approach, and the network is partitioned into several subnetworks following the principles of electric network tearing. Therefore each subnetwork interacts with other subnetworks using interface variables that are represented by linear combinations of the components inside them. Hence the discrete system equation is exactly equivalent to that of the network without partitioning. In the latter one, the decoupling inserts the delays in discrete time to decouple the network into different subnetworks. Consequently the involved variables are not coupled between subnetworks at one discrete time step. The overall discrete system equation is no longer equivalent to the formulation of the nondecoupled network model. However, this method suffers from numerical stability since the pole positions of the discrete system have been changed.

In the device-level simulation the power electronic switch is replaced by a more detailed representation that concerns the switching transients. By revisiting the offline switch models, it can be seen that the behavioral model is more suitable for real-time simulation than the physics-based model. Nevertheless, directly employing the offline behavioral switch model in real-time simulation can still be time-consuming. The nonlinearities introduced by the device-level switch model can aggravate the computational burden at each simulation time step. Meanwhile the time scale of the switching transient can range from several microseconds down to tens of nanoseconds, which imposes a demand for a very short simulation time step. Therefore the device-level real-time simulation can be very difficult to implement. In the literature, four types of IGBT device-level real-time models have been proposed so far, including the nonlinear equivalent circuit model, piecewise linear transient model, curve-fitting model, and two-level quasi-transient model. The accuracies of these models have been compromised to different extents in exchange for computational efficiency.

1.1.5 Challenges of power electronic real-time simulation

The real-time simulation has strict time constraints in model computation. The high-frequency switching characteristic of power electronic converters makes the available computing time very short, in the range from tens of nanoseconds to several microseconds. The computational workload thus has to be limited to a relatively low level, which means that the topology scale, the model complexity, and the numbers of switches are very restricted in the real-time simulation. To realize it the following challenges are raised.

1.1.5.1 Improve the modeling accuracy

Due to the strictly limited computing time, the complex model cannot be used in real-time simulation, which hinders the attempts to improve simulation accuracy. For example the device-level simulation needs a behavioral transient switch model to be implemented, but the computing burden is increased significantly due to the nonlinearities in the switch models. A more detailed model would not only improve the simulation accuracy but also stimulate the multiphysics simulation of power electronics, which could broaden the functionality and applications of real-time simulation.

1.1.5.2 Reduce the simulation time step

A shorter simulation time step will lead to a lower numerical discretization error. More importantly, with the popularization of wide-bandgap (WBG) devices, the switching frequency becomes much higher than that of the Si-based devices. The reduction of the simulation time step is imperative for accurate modeling of the WBG devices. However, a shorter simulation time step aggravates the time constraint on the model computing, which may sacrifice the modeling accuracy and the topology scale.

1.1.5.3 Interfacing issues in the HIL real-time simulation

The HIL real-time simulation introduces artificial latencies in the semiphysical testing loop. On one hand the real-time simulation is in discrete time and it is natural to have latency for the real-time model to sample the input signals. On the other hand the input and output of the real-time simulation need time to complete the signal conversion or power amplifier between the virtual and physical devices in the loop. The latency will reduce the accuracy of the real-time simulation results, even threatening the stability of the HIL simulation. The latencies should be minimized in the design process and further compensated in the application process.

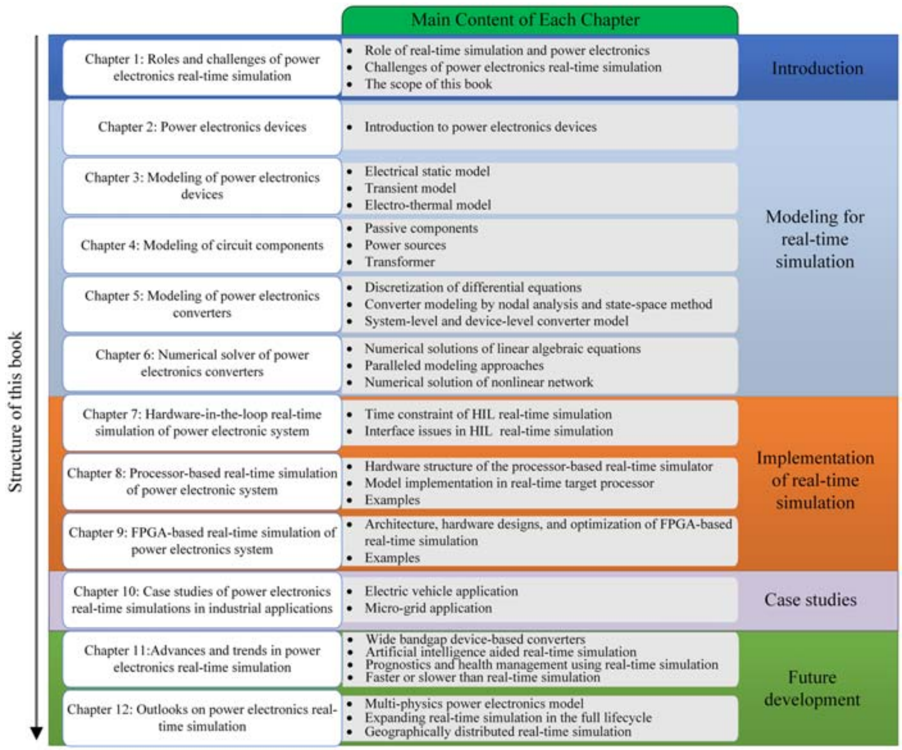
1.1.5.4 Balance the generality and performance

The processor and FPGA have different characteristics and development workflows, resulting in different real-time simulation design flows and simulation performances. The real-time simulation hardware should be carefully chosen by fully considering the requirement of the applications. Since the computing time is of concern the underlying digital circuit design should be considered in together with the upper layer computing algorithm, such as the execution time, data throughput, data representation, transfer mechanism, bus communications, and so on. Besides the hardware implementation of real-time simulation is also diverse with the modeling methodology and the numerical solving algorithm. The designs are prone to be customized to the specific application and lack of generality. In the meantime the generic design is in great demand since the users cannot design the real-time simulation by themselves in most cases. However, the generic design and implementation of real-time simulation in terms of modeling, simulation, hardware structures, and development environment are tough to balance the generality and performance.

1.1.6 Scope and structure of the book

This book aims to provide a systematic introduction to real-time simulation technologies for modern power electronics, including the modeling approaches, the numerical solvers, the real-time implementations, the applications, and the outlooks of future technologies. It starts with an introduction to power electronic device modeling, component modeling, and power converter modeling before addressing numerical methods to solve the converter models, emphasizing computational efficiency and accuracy. This book discusses both processor-based and FPGA-based real-time implementations and provides a detailed review of current applications, including hardware-in-the-loop and its case studies in the electric vehicles and microgrid applications. The book closes with a review of the near and long-term outlooks for the evolving technologies. Collectively the work will provide a systematic resource for students, researchers, and engineers in electrical engineering and other closely related fields.

This book is organized into 12 chapters. The first part is the introductory chapter that briefly presents the role, the state of the art, and the challenges of the power electronic real-time simulation. The second part of the book discusses the modeling methodology for power electronic real-time simulation. It comprises five chapters that cover the introduction of power electronic devices (Chapter 2), modeling the power electronic devices (Chapter 3), modeling of the circuit components (Chapter 4), modeling the power converters (Chapter 5), and the numerical solvers of power converters (Chapter 6). The third part of the book presents the implementation of real-time simulations, covering the introduction of the hardware-in-the-loop simulation (Chapter 7) and the implementation methods for both processor-based (Chapter 8) and FPGA-based



Structure of this book	Main Content of Each Chapter	
Chapter 1: Roles and challenges of power electronics real-time simulation	• Role of real-time simulation and power electronics • Challenges of power electronics real-time simulation • The scope of this book	Introduction
Chapter 2: Power electronics devices	• Introduction to power electronics devices	Modeling for real-time simulation
Chapter 3: Modeling of power electronics devices	• Electrical static model • Transient model • Electro-thermal model	
Chapter 4: Modeling of circuit components	• Passive components • Power sources • Transformer	
Chapter 5: Modeling of power electronics converters	• Discretization of differential equations • Converter modeling by nodal analysis and state-space method • System-level and device-level converter model	
Chapter 6: Numerical solver of power electronics converters	• Numerical solutions of linear algebraic equations • Paralleled modeling approaches • Numerical solution of nonlinear network	
Chapter 7: Hardware-in-the-loop real-time simulation of power electronic system	• Time constraint of HIL real-time simulation • Interface issues in HIL real-time simulation	Implementation of real-time simulation
Chapter 8: Processor-based real-time simulation of power electronic system	• Hardware structure of the processor-based real-time simulator • Model implementation in real-time target processor • Examples	
Chapter 9: FPGA-based real-time simulation of power electronics system	• Architecture, hardware designs, and optimization of FPGA-based real-time simulation • Examples	
Chapter 10: Case studies of power electronics real-time simulations in industrial applications	• Electric vehicle application • Micro-grid application	Case studies
Chapter 11:Advances and trends in power electronics real-time simulation	• Wide bandgap device-based converters • Artificial intelligence aided real-time simulation • Prognostics and health management using real-time simulation • Faster or slower than real-time simulation	Future development
Chapter 12: Outlooks on power electronics real-time simulation	• Multi-physics power electronics model • Expanding real-time simulation in the full lifecycle • Geographically distributed real-time simulation	

Figure 1.5 Main content of each chapter.

(Chapter 9) real-time simulation. The fourth part of the book presents case studies of power electronic real-time simulation in electric vehicles and microgrid applications (Chapter 10). The last part of the book discusses the advances and recent trends (Chapter 11) and provides an outlook on the future development of power electronic real-time simulation (Chapter 12). Fig. 1.5 gives an overview of the structure of this book with brief descriptions of the main contents in each chapter.

References

[1] X. Xin, Power Electronic Technology, China Machine Press, China, 2021 (in Chinese) (Chapter 1).

[2] W.E. Newell, Power electronics—emerging from limbo, in: IEEE Transactions on Industry Applications, IA-10, 1, pp. 7–11, 1974.

[3] Typhoon HIL, Inc., Typhoon-HIL404-Brochure, 2022. [Online]. Available from: https://typhoon-hil.com/doc/brochures/Typhoon-HIL404-Brochure.pdf.

[4] H.F. Blanchette, T. Ould-Bachir, J.P. David, A State-Space Modeling Approach for the FPGA-Based Real-Time Simulation of High Switching Frequency Power Converters,

IEEE Transactions on Industrial Electronics 59 (12) (2012). Available from: https://doi.org/10.1109/TIE.2018.2833058.

[5] A. Hadizadeh, M. Hashemi, M. Labbaf, M. Parniani, A matrix-inversion technique for FPGA-based real-time EMT simulation of power converters, IEEE Transactions on Industrial Electronics 66 (2) (2019) 1224–1234. Available from: https://doi.org/10.1109/TIE.2018.2833058.

[6] C. Liu, H. Bai, S. Zhuo, X. Zhang, R. Ma, F. Gao, Real-time simulation of power electronic systems based on predictive behavior, IEEE Transactions on Industrial Electronics 67 (9) (2020) 8044–8053. Available from: https://doi.org/10.1109/TIE.2019.2941135.

[7] H. Chalangar, T. Ould-Bachir, K. Sheshyekani, J. Mahseredjian, A direct mapped method for accurate modeling and real-time simulation of high switching frequency resonant converters, IEEE Transactions on Industrial Electronics 68 (7) (2021) 6348–6357. Available from: https://doi.org/10.1109/TIE.2020.2998746.

Power electronic devices

2

2.1 Introduction

Power electronics can be defined as the use of static converters to control and process electric power. A key component of these power converters is the power semiconductor devices, also called power electronic devices. Different from the semiconductors used in the electronic field, power electronic devices generally have rated currents greater than 1A and blocking voltage ranges from several volts to tens of kilovolts. The power rating of the power electronic devices also covers a wide range, from several watts to several megawatts, targeting lower power applications such as domestic appliances, middle power applications such as electric propulsion systems, and high-power applications such as the high voltage DC power transmission.

The type of power semiconductor device, topology, and control system are the three main elements that characterize a power electronic converter. A basic structure is shown in Fig. 2.1. The control system calculates the control signals and sends them to the driving circuit, which drives the power electronic devices. The power electronic devices are the basis of the power electronic converters. The development of power electronic devices drives the revolution of power converters and plays an important role in the reliability, cost, and performance of power electronic converters.

The power electronic devices can be classified into three types according to the controlling features, that is, uncontrolled devices, half-controlled devices, and fully controlled devices. The uncontrolled devices mainly refer to different types of power diodes. They have only two terminals, and the turn-on and turn-off states are determined by the power circuit, namely, the voltage between the anode and cathode of the diode. Positive voltage leads to conduction, and negative voltage leads to blocking. The half-controlled devices mainly refer to the thyristors, which also have unilateral conductivity. Different from the diode, the thyristor has three terminals. A gate terminal is added in addition to the semiconductor's anode and cathode. The turn-on of thyristor not only relies on the positive voltage between the anode and cathode but also relies on the positive voltage between the gate and the cathode. Therefore the turn-on event is controllable. The reason why it is a half-controlled device is that once the device is turned on, the turn-off cannot be controlled by the gate terminal and is determined by the power circuit. The fully controlled devices refer to the three-terminal semiconductors in which the control terminal can determine both the turn-on and turn-off states. The power MOSFET (metal-oxide-semiconductor field-effect transistor), IGBT (insulated gate bipolar transistor), GTO (gate-turn-off thyristor), and Power BJT (bipolar junction transistor) are all fully controlled devices.

Real-Time Simulation Technology for Modern Power Electronics. DOI: https://doi.org/10.1016/B978-0-323-99541-2.00012-0

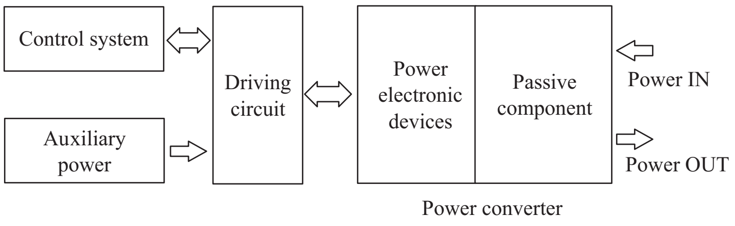


Figure 2.1 Basic structure of power electronic converters.

This chapter gives a brief introduction of commonly used power electronics devices, including the physics structures, operating mechanism, static characteristics, and dynamic characteristics of the power diode, thyristor, power BJT, power MOSFET, and IGBT.

2.2 Power diode

2.2.1 Operating principle of PN junction

The basic structure of the power diode is like that of the electronic diode, which is based on the semiconductor PN junctions shown in Fig. 2.2. The operating mechanism of a PN junction is simple. When the positive voltage is applied to the PN junction (the PN junction is forward-biased) the diffusion current will be established and the PN junction will be forward-conducting. Conversely, when the negative voltage is applied to the PN junction (the PN junction is reverse-biased) the PN junction will be cut-off. The power diode usually adopts a vertical structure shown in Fig. 2.3, different from the electronic diode. The vertical structure can make full use of the conductivity of the silicon slice to increase the current conducting capability. Moreover an additional lightly doped drift region is added between the P and N regions, allowing the power diode to support high voltage in the off-state. The thicker the drift region, the higher the voltage capability of the diode. However, the drift region has a high resistivity due to the low doping density. Thanks to the conductivity modulation the forward voltage drop will be maintained at around 1V even when the high current is conducted by the diode [1].

The PN junction can stand the reverse voltage to some extent. The current is negligibly small when the diode is reverse-biased. When the reverse voltage exceeds the device's ability to block it, the reverse current will increase significantly, and the diode will break down.

2.2.2 Characteristics of power diodes

2.2.2.1 Static characteristics

When the voltage applied to the diode exceeds the threshold voltage, the current starts to increase and the diode will be in a steady turn-on state. The forward

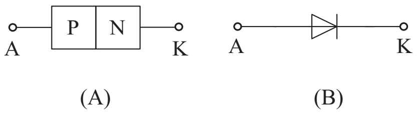


Figure 2.2 (A) Basic structure of the diode; (B) electrical symbol of the diode.

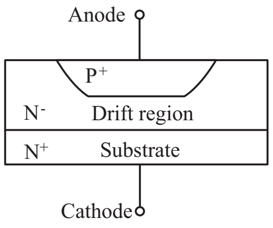


Figure 2.3 Cross-section diagram of the vertical structure diode.

voltage V_F is the voltage drop of a diode across A (anode) and K (cathode) at a defined current I_F.

The static current–voltage relationship is depicted in Fig. 2.4, which can be represented by a resistor R_{on} in series with an ideal diode. The voltage and current relationship across the ideal diode V_j is expressed by the exponential equation shown in (2.1), where I_s is the leakage current and V_b is the junction barrier potential.

$$I_d = I_s\left(e^{\frac{v_j}{V_b}} - 1\right) \tag{2.1}$$

The I_s and R_{on} can be extracted by the static characteristic curve shown in Fig. 2.4. V_b can be obtained by applying (2.1) at the point of 1A in Fig. 2.4, as shown in (2.2), where V_{on} is the diode voltage at $I_d = 1\text{A}$.

$$V_b = \frac{V_{on} - R_{on}}{\ln\left(1 + \frac{1}{I_s}\right)} \tag{2.2}$$

2.2.2.2 Dynamic characteristics

The reverse recovery phenomenon is the typical dynamic characteristic of the diode. The typical reverse recovery curve of the diode is depicted in Fig. 2.5. When the diode turns off the current flowing through the diode will decrease to zero. After it reaches zero a reverse current will be flowing through the diode due to the minority carrier storage effect. After reaching its peak value I_{rrm} the reverse current will finaly decay to zero. The t_{rr} is the time interval when the diode conducts the reverse current.

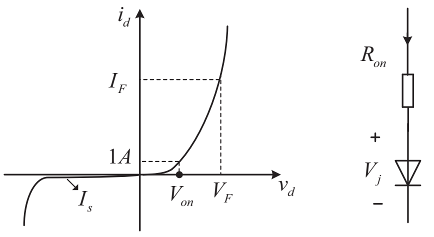


Figure 2.4 Current−voltage relationship of power diodes.

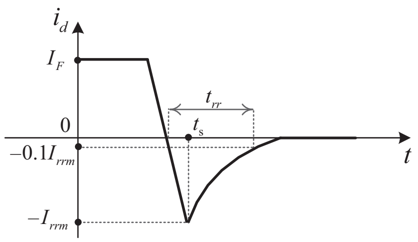


Figure 2.5 Diode reverse recovery curve.

According to Lauritzen's diode model with reverse recovery the diode dynamic characteristics can be expressed by (2.3) to (2.6), where $q_e(t)$ is the junction charge variable, $q_m(t)$ is the charge in the lightly doped region, $i_d(t)$ and $v_d(t)$ are the current and voltage across the diode, respectively, T_m is the diffusion transit time, τ is the carrier lifetime, I_s^* is the diffusion leakage current, $v_j(t)$ is the junction voltage, V_t^* is the junction barrier potential, and R_{on} is the internal series resistance [2,3].

$$i_d(t) = \frac{q_e(t) - q_m(t)}{T_m} \tag{2.3}$$

$$i_d(t) = \frac{dq_m(t)}{t} + \frac{q_m(t)}{\tau} \tag{2.4}$$

$$q_e(t) = I_s^* \tau \left[e^{\frac{v_j(t)}{2V_t^*}} - 1 \right] \tag{2.5}$$

$$v_d(t) = v_j(t) + R_{on} i_d(t) \tag{2.6}$$

Under static operating conditions, Eq. (2.7) can be obtained by combining (2.3) and (2.4).

$$q_e = (\tau + T_m) I_d \tag{2.7}$$

Substituting (2.7) into (2.5), we can get (2.8), which is equivalent to (2.1) in terms of the static operating conditions.

$$I_d = \frac{I_s^* \tau}{(\tau + T_m)} \left[e^{\frac{v_j(t)}{2V_t^*}} - 1 \right] \tag{2.8}$$

Regarding the reverse recovery characteristics of the diode, Fig. 2.6 is an equivalent circuit of the Lauritzen's diode model which is usually used in the circuit simulation [2]. In Fig. 2.6 the diode current is the sum of three branches' currents.

$$i_d = K \cdot v_{Lrr} + \frac{v_{Lrr}}{R_{rr}} + i_{Lrr} \tag{2.9}$$

Replacing the inductor's voltage v_{Lrr}, Eq. (2.10) is obtained.

$$i_d = KL_{rr} \left(1 + \frac{1}{KR_{rr}} \right) \frac{di_{Lrr}}{dt} + i_{Lrr} \tag{2.10}$$

Comparing (2.10) and (2.4), we can express $q_m(t)$ and τ using the parameters and variables in Fig. 2.6.

$$\begin{cases} q_m(t) = KL_{rr} \left(1 + \dfrac{1}{KR_{rr}} \right) i_{Lrr} \\ \tau = KL_{rr} \left(1 + \dfrac{1}{KR_{rr}} \right) \end{cases} \tag{2.11}$$

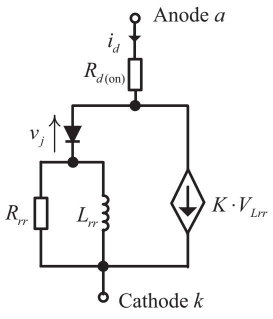


Figure 2.6 Diode equivalent circuit model with reverse recovery behavior.

We can also rewrite (2.9) as (2.12) and (2.13), based on which we can derive the expression of $i_d$in the formula of (2.14).

$$i_d = K i_{Rrr} R_{rr} + i_{Rrr} + i_{Lrr} \tag{2.12}$$

$$i_d = KR_{rr}\left[\left(1+\frac{1}{KR_{rr}}\right)(i_{Rrr}+i_{Lrr}) - i_{Lrr}\right] \tag{2.13}$$

$$i_d = \frac{KL_{rr}\left(1+\frac{1}{KR_{rr}}\right)^2 (i_{Rrr}+i_{Lrr}) - KL_{rr}\left(1+\frac{1}{KR_{rr}}\right) i_{Lrr}}{\frac{L_{rr}}{R_{rr}}\left(1+\frac{1}{KR_{rr}}\right)} \tag{2.14}$$

Comparing (2.14) and (2.3), we can express $q_e(t)$ and T_m as (2.15) using the parameters and variables in Fig. 2.6.

$$\begin{cases} q_e(t) = KL_{rr}\left(1+\frac{1}{KR_{rr}}\right)^2 (i_{rr}+i_{Lrr}) \\ T_m = \frac{L_{rr}}{R_{rr}}\left(1+\frac{1}{KR_{rr}}\right) \end{cases} \tag{2.15}$$

Thus it has been demonstrated that the equivalent circuit model shown in Fig. 2.6 is equivalent to the dynamic model in (2.3) to (2.6).

Refering to the reverse recovery phenomenon shown in Fig. 2.5, if the i_d is assumed to decrease linearly, the voltage across the L_{rr} will be a constant negative value and induce a constant current equal to I_{rrm} in the controlled current source. When $t < t_s$the ideal diode in Fig. 2.6 conducts and the current is determined by the surrounding circuit. When $t = t_s$ the i_d reaches peak I_{rrm} and the ideal diode is blocked. When $t > t_s$ the ideal diode is cut-off, and branches of R_{rr} and L_{rr} are independent from the circuit. The diode current is governed by the controlled current source of which the value is decaying from I_{rrm}.

2.3 Thyristor

2.3.1 Structure and operating mechanism of Thyristors

The thyristor is a semiconductor with four layers, denoted as P1, N1, P2, and N2 in the conceptual view in Fig. 2.7. The anode, the cathode, and the gate are connected to P1, N2, and P2 regions, respectively. Four semiconductor regions constitute three PN junctions, denoted as J1, J2, and J3 in Fig. 2.7. When a positive voltage is applied to the anode with respect to the cathode, J2 is reverse-biased. The device is blocked, and only a small leakage current flows through it. When a negative voltage is applied to the thyristor, J1 and J3 are reverse-biased, and the device is also blocked.

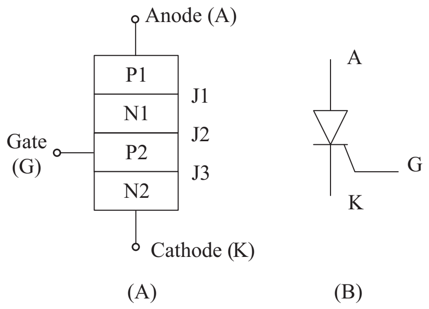


Figure 2.7 (A) Structure of the thyristor; (B) symbol of the thyristor.

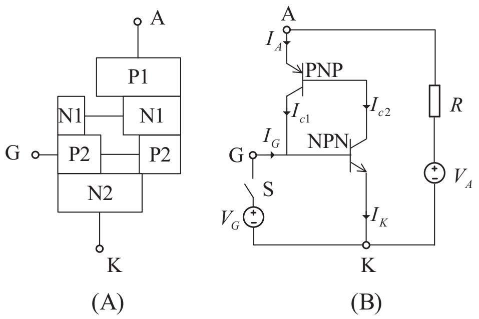


Figure 2.8 (A) Equivalent model of the thyristor; (B) operating mechanism of the thyristor.

By observing the structure in Fig. 2.7 the thyristor can be seen as equivalent to two transistors. One is PNP type, and the other is NPN type, as shown in Fig. 2.8. When the current is injected into the gate, the current flows into the base of the NPN transistor. The resulting collector current of the NPN will then become the base current of the PNP transistor, which will be magnified as the collector current of PNP. The current I_{c1} will then feed into the base of NPN, and a positive feedback mechanism is formed. The two transistors will finally both enter the saturation region, and the thyristor will be conducting even if the gate current is removed. Therefore the turn-on can be controlled by the gate, but the turn-off cannot. Only if the positive voltage applied to A-K is removed or the conducting current is forced to zero, the thyristor will turn off.

2.3.2 Characteristics of thyristors

2.3.2.1 Static characteristics

Fig. 2.9 depicts the current–voltage characteristics of the thyristor. The forward conduction characteristics are shown in quadrant I, and the reverse characteristics are shown in quadrant III [4].

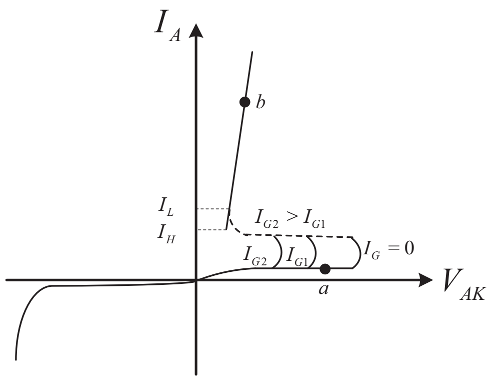


Figure 2.9 Static characteristics of the thyristor.

If I_G is zero, the thyristor will be forward-blocking when a positive voltage is applied between A and K. A small leakage current flows through the thyristor due to the reverse bias of junction *J2*, shown as the operating point *a* in Fig. 2.9. If the applied voltage V_{AK} exceeds the forward-blocking voltage, the leakage current will increase dramatically and the thyristor will be conducting. The operating points will move along with the dashed line from the high-impedance area to the lower-impedance area. The increase of I_G will lower the blocking voltage, as illustrated for I_{G_1} and I_{G_2}. The characteristics of a forward-conducting thyristor are like those of the diode. The forward voltage drop will be stably retained at a relatively low value (typically around 1V), even when a large anode current flows through.

In the forward conduction state the external circuit should provide sufficient anode current to keep the device conducting. If the thyristor is already in forward conduction but the anode current is decreased, the thyristor will switch to the forward-blocking state when the anode current is low enough. The minimum anode current to support the thysistor in forward conduction is called the holding current I_H. If the thyristor is just moving from the blocking state to the conduction state, the minimum current to keep the thysistor in forward conduction when the gate current is removed is called the latching current I_L. The holding current is of a small value ranging from tens to hundreds of milliamperes and is temperature-dependent. The latching current is usually two or three times higher than the holding current for the same thyristor.

The thyristor will switch to the reverse blocking state if the negative voltage is applied between A and K. The reverse characteristics are analogous to those of the diode. When the reverse voltage exceeds the limit the reverse leakage current will grow emergently and the thyristor will break down.

2.3.2.2 *Dynamic characteristics*

The typical switching waveforms of the thyristor are shown in Fig. 2.10. The ideal gate trigger current is applied at the initial time t = 0. Since the positive feedback

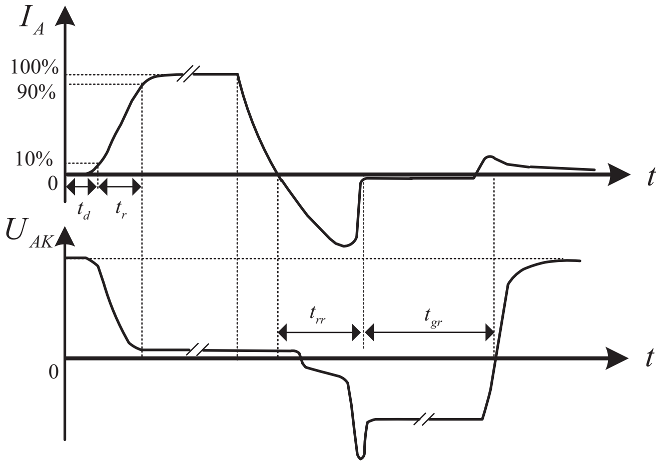


Figure 2.10 Turn-on and turn-off transient waveforms of the thyristor.

mechanism needs time to excite, the anode current will not rise at the exact moment when the trigger arrives. Constrained by the inductive loading current the current will rise with a given slope. The time interval between the instant when the gate trigger current arrives and the instant when the anode current reaches 10% of the steady-state value is defined as the turn-on delay time t_d, and the time interval that the current takes to rise from the 10% to 90% of the steady-state values is defined as the rise time t_r. The turn-on time will be the sum of t_d and t_r. During this period the forward voltage across the device will decrease until it reaches steady values around 1V.

The turn-off transient starts when the applied voltage changes its polarity. The current cannot be removed immediately usually due to the inductive load in the outer circuit. The anode current will decay toward zero. Similar to the diode the reverse recovery phenomenon occurs when the anode current passes through zero. The reverse recovery current will cause a voltage spike across the thyristor due to the inductance in the circuit. The time from the current passing through zero to the reverse recovery current almost decaying to zero is defined as the recovery time of reverse blocking t_{rr}. After the reverse recovery stage the thyristor will take time to restore the forward blocking capacity due to the slow combinations of the carriers, which is known as the recovery time of forward blocking t_{gr}.

2.4 Power bipolar junction transistor

2.4.1 Structure and operating mechanism of BJT

The operating mechanism of power BJT is the same as that of the general BJT except for the higher voltage withstanding and current carrying capability. The power BJT is usually used in the saturation region instead of the active region. The cross-section of a typical power BJT is shown in Fig. 2.11. It can be observed that compared to the general BJT an N-drift region is added that is lightly doped. Similar to the power diode the drift region is used to support high voltage. Meanwhile the on-state voltage can be reduced by conductivity modulation.

The power BJT is used in the common-emitter configuration. The collector current i_c and the base current i_b have a relationship described by (2.16)

$$\beta = \frac{i_c}{i_b} \tag{2.16}$$

where β is the current amplification coefficient that reflects the ability of the base current to control the collector current. When considering the leakage current between the collector and emitter I_{ceo} the relationship can be modified as (2.17).

$$i_c = \beta i_b + I_{ceo} \tag{2.17}$$

2.4.2 Characteristics of BJT

2.4.2.1 Static characteristics

The output characteristics of the BJT under common-emitter configuration are shown in Fig. 2.12. Three operating regions are distinct, namely, the cut-off region, the saturation region, and the active region. In power electronics applications the BJT is used as a switch and operates at the cut-off region or the saturation region. However, the transient behaviors between the cut-off region and the saturation regions will go through the active region.

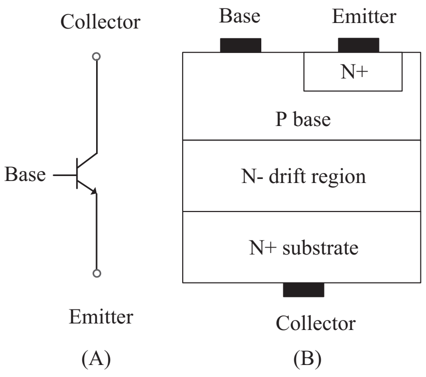


Figure 2.11 (A) Electrical symbol of NPN-type BJT; (B) vertical cross-sectional view for a BJT.

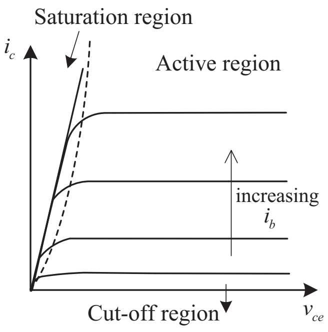


Figure 2.12 Output characteristics of power BJT.

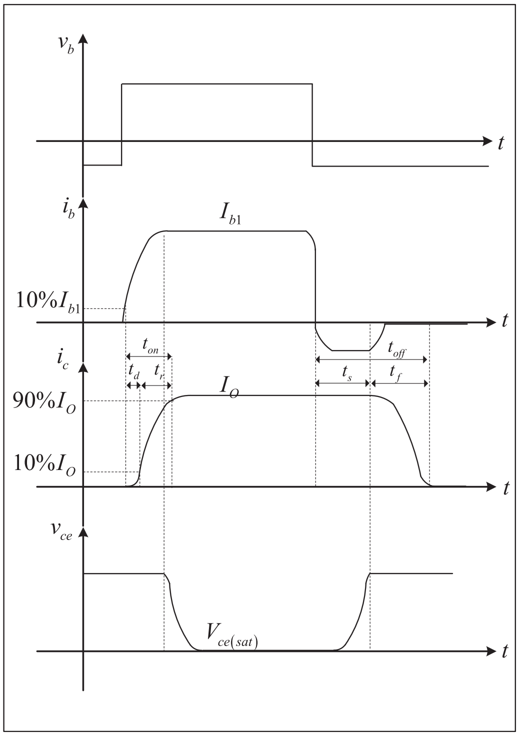


Figure 2.13 Switching transient waveforms of power BJT.

2.4.2.2 Dynamic characteristics

Fig. 2.13 shows typical switching waveforms for an inductive load. The turn-on transient includes the turn-on delay time t_d and the current rise time t_r(from 10% to 90% of maximum value I_O), and the sum of them is the turn-on time denoted as t_{on}. The turn-off transient includes the turn-off delay time t_s and the current fall time t_f, and the sum of them is the turn-off time t_{off}. The overlapping of the transient current and voltage leads to the switching power losses.

2.5 Power MOSFET

2.5.1 Structure and operating mechanism of power MOSFET

The MOSFET uses only the majority carriers in conduction mode, and it is a unipolar device. The device symbols for a p- and n-channel enhancement and depletion types are shown in Fig. 2.14. Most of the MOSFET devices used in power electronics applications are of the n-channel, enhancement-type like that which is shown in Fig. 2.14A.

To enhance the power rating the power MOSFET adopts a vertical channel structure. Fig. 2.15 shows a vertical cross-sectional view for a power MOSFET. When the drain and source terminals are connected to a positive voltage source the PN junction J1 between the P-based and n-drift regions is reverse-biased. When the gate and source terminals are of the same potential ($V_{gs} = 0$), there is no conducting

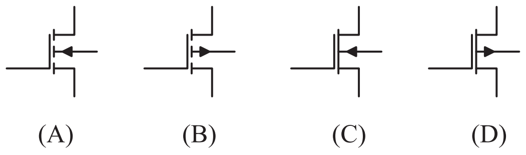


Figure 2.14 (A) n-Channel enhancement type, (B) p-channel enhancement type, (C) n-channel depletion type, and (D) p-channel depletion type.

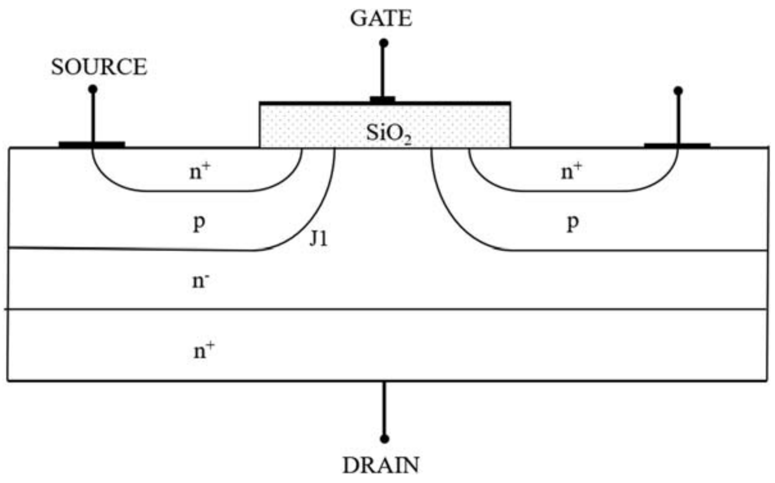


Figure 2.15 Vertical cross-sectional view for a power MOSFET.

channel and the PN junction J1 provides the forward blocking capability of the MOSFET. If the positive voltage is applied to the gate and source, there is no current flowing through the gate and the source due to the insulation of the gate. However, the positive voltage V_{gs} will push the holes away and attracts the electrons to the surface of the P-based region right near the gate. When V_{gs} is larger than the threshold voltage V_{th} the density of the electron in the surface of the P-based region will dominate and the P region will be inverse to the N type, which eliminates the PN junction J1. A conduction channel is thus formed between the drain and the source. The larger V_{gs} will further enhance conductivity and induce a larger drain current i_d.

Based on the same mechanism, the MOSFET introduces the slightly doped N-region to increase the voltage endurance. However, the conduction modulation cannot be fulfilled since the MOSFET relies only on the majority carriers to conduct. Therefore the side effect of the drift region is the increase of the conduction resistance as well as the power losses. The MOSFET is usually adopted in applications under 1000V.

2.5.2 Characteristics of power MOSFET

2.5.2.1 MOSFET static characteristics

The output characteristic curves of the n-channel, enhancement-type MOSFET is shown in Fig. 2.16A. There are three distinct operation regions labeled as linear region, saturation region, and cut-off region. When it is used as a switching device, only linear and cut-off regions are used.

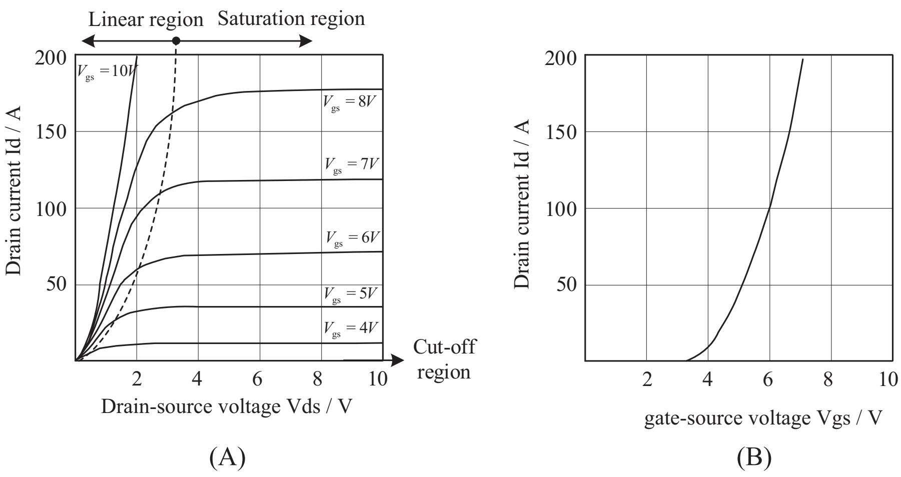


Figure 2.16 (A) Output characteristics of power MOSFET; (B) transfer characteristics of power MOSFET.

The transfer characteristic curves shown in Fig. 2.16B reflect the relationship between the drain current I_d and the gate-source voltage V_{gs}, which is also the relationship between the input voltage and output current of MOSFET. The relationship between I_d and V_{gs} is approximately linear when I_d is of a relatively larger value, and the slope can be defined as the transconductance. The MOSFET is a voltage-controlled device, and the input impedance is high.

1. Cut-off region

   When $V_{gs} < V_{th}$, there is no induced channel and the device is cut-off.
2. Linear region

   Once $V_{gs} > V_{th}$ the channel is formed between the drain and the source. The MOSFET can operate either in the linear region or in the saturation region, which is determined by the drain-source voltage. The boundary between operating regions can be expressed by

   Cut off region: $v_{gs} < V_{th}$

   Linear region: $v_{gs} > V_{th}$, and $v_{ds} < v_{gs} - V_{th}$

   Saturation region: $v_{gs} > V_{th}$, and $v_{ds} > v_{gs} - V_{th}$

The output characteristics of MOSFET have strong nonlinearities, and accurate representations can be complex. One mathematical approximation of the output characteristics is given by (2.18)

$$i_{ds} = \begin{cases} 0 & \text{Cut-off region} \\ K\left[2\left(v_{gs} - V_{th}\right)v_{ds} - v_{ds}^2\right] & \text{Linear region} \\ K\left(v_{gs} - V_{th}\right)^2 & \text{Saturation region} \end{cases} \tag{2.18}$$

where $K = \frac{1}{2}\mu_n C_{OX}\left(\frac{W}{L}\right)$, μ_n is the electron mobility, C_{OX} is the oxide capacitance per unit area, L is the length of the channel, and $W$is the width of the channel.

The power MOSFET has an internal diode between the source and the drain. As shown in Fig. 2.15 the P base together with N− drift and N+ regions forms a power diode antiparalleled to the MOSFET. When the drain-source voltage is negative the MOSFET will conduct due to the imbody diode.

2.5.2.2 MOSFET dynamic characteristics

The switching dynamic characteristics of MOSFET are deeply tied with the parasitic capacitances between three terminals. The parasitic capacitances can be represented by three interterminal capacitors, namely, the gate-to-source capacitor C_{gs}, the gate-to-drain capacitor C_{gd}, and the drain-to-source capacitor C_{ds}. The charging and discharging behaviors play important roles in the switching transient. The values of these capacitances are nonlinear and dependent on the device voltage. The C_{gs} variation is relatively small and can be treated as a constant when analyzing the transient.

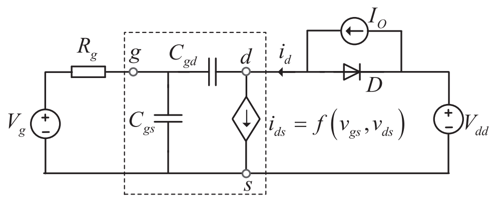


Figure 2.17 Equivalent circuit of MOSFET to analyze the switching transient.

The device datasheet usually gives the curves of these capacitances but in the form of the input capacitance C_{iss}, output capacitance C_{oss}, and reverse capacitance C_{rss}, as given in (2.19).

$$\begin{cases} C_{gd}=C_{rss} \\ C_{gs}=C_{iss}-C_{rss} \\ C_{ds}=C_{oss}-C_{rss} \end{cases} \tag{2.19}$$

Fig. 2.17 shows the equivalent circuit to analyze the switching transient of a power MOSFET where the inductive load current is considered and approximated as a current source in the turn-on and turn-off transients [5]. The output capacitance between the drain and source, C_{ds}, has little effects on the turn-on and turn-off MOSFET switching characteristics and is neglected in Fig. 2.17.

Fig. 2.18 depicts the typical switching transient waveforms of power MOSFET. To facilitate the analysis the switching transient is divided into different stages [4]. By taking the turn-on transient as an example, four stages are recogonized and each of them is explained as follows:

1. t_0–t_1.

The positive gate voltage is applied at $t=t_0$. The gate voltage is below the V_{th}, and no current flows through the MOSFET. Due to the conduction of the diode, the voltage across the MOSFET is clamped. The gate voltage charges the capacitor C_{gs}, and C_{gd}, as given in (2.20), and v_{gs} will grow exponentially, as shown in (2.21).

$$\frac{V_g-v_{gs}}{R_g}=\left(C_{gs}+C_{gd}\right)\frac{dv_{gs}}{dt} \tag{2.20}$$

$$v_{gs}(t)=V_g\left(1-e^{\frac{t-t_0}{\tau}}\right) \tag{2.21}$$

where $\tau=R_g\left(C_{gs}+C_{gd}\right)$.

The time interval of this period is

$$\Delta t_{10}=-\tau\ln\left(1-\frac{V_{th}}{V_g}\right) \tag{2.22}$$

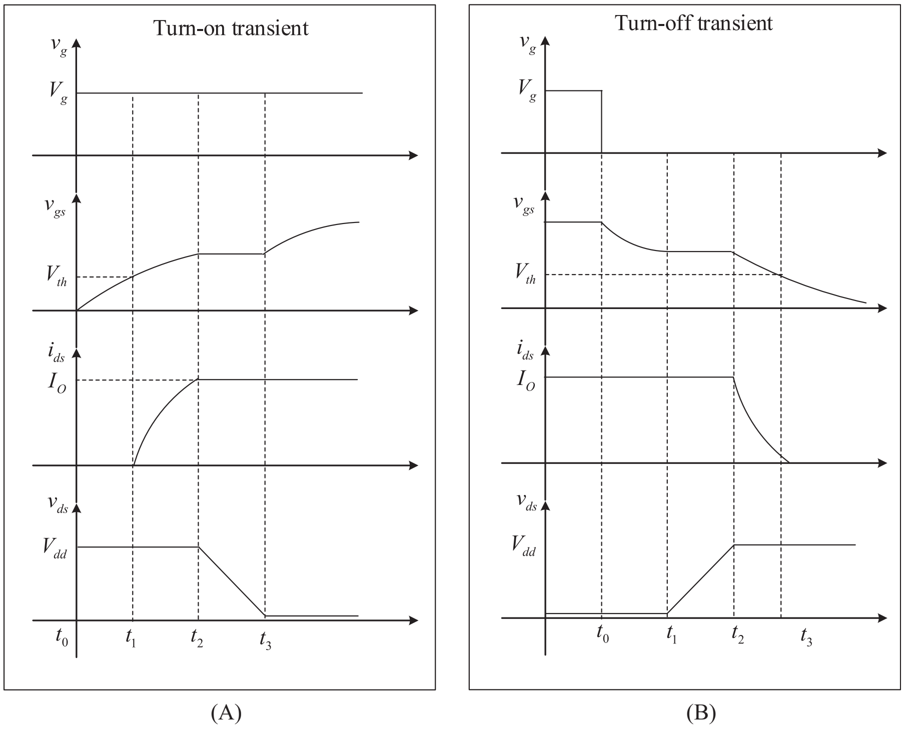


Figure 2.18 Switching transient waveforms of power MOSFET. (A) Turn-on transient waveforms and (B) turn-off transient waveforms.

2. $t_1 - t_2$.

When $v_{gs} > V_{th}$ the MOSFET starts to conduct and the drain current increases as a function of v_{gs} and V_{th}, as given in (2.18). In order to analyze the current rising behavior the transfer characteristics can be approximated by a linear Eq. (2.23).

$$i_{ds} = g_m\left(v_{gs} - V_{th}\right) \tag{2.23}$$

Since $i_{ds} < I_o$ the diode is still conducting and the v_{ds} is clamped at its off value. The v_{gs}will grow as in (2.21) and i_{ds} will increase exponentially as in (2.24). The time interval is thus obtained by (2.25).

$$i_{ds} = g_m\left(V_g - V_{th}\right) - g_m V_g e^{-\frac{(t-t_1)}{\tau}} \tag{2.24}$$

$$\Delta t_{21} = \tau \ \ln \frac{g_m V_g}{g_m\left(V_g - V_{th}\right) - I_o} \tag{2.25}$$

3. $t_2 - t_3$.

i_{ds} reaches the steady on-state value, and the diode thus turns off. The drain-source voltage v_{ds}then decreases. Before v_{ds} drops near the steady on-state value the MOSFET is still in the saturation region. According to the transfer characteristics, the constant i_{ds} leads to a constant V_{gs} in this period as in (2.26), which is called a miller plateau.

$$V_{gs} = \frac{I_o}{g_m} + V_{th} \tag{2.26}$$

The analytical expression of v_{ds} can be obtained by

$$v_{ds} = -\frac{V_g - V_{th}}{R_g C_{gd}}(t - t_2) + V_{dd} \tag{2.27}$$

The time interval of this period can be thus calculated by letting $v_{ds} = V_{dd}$ in (2.28).

$$\Delta t_{32} = R_g C_{gd} \frac{V_{dd} - I_o r_{ds(on)}}{V_g - V_{th}} \tag{2.28}$$

4. $t_3 - t_4$.

The v_{ds} is of constant turn-on value, and the MOSFET enters the linear region. The gate current continues charging the C_{gd}, and v_{ge} grows exponentially by (2.29) until it reaches V_g.

$$v_{gs} = V_g\left(1 - e^{-\frac{t-t_3}{\tau}}\right) + V_{gs}(t_3)e^{-\frac{t-t_3}{\tau}} \tag{2.29}$$

The turn-on transient is thus analyzed stage by stage and can be drawn as Fig. 2.18A. The turn-off transient can be regarded as the inversion process of the turn-on transient. The detailed analysis can be implemented referring to the turn-on process, and the transient waveforms are drawn in Fig. 2.18B.

2.6 Insulated gate bipolar transistor

2.6.1 Structure and operating mechanism of IGBT

The vertical cross-section view of an n-channel IGBT is shown in Fig. 2.19. It has an additional P+ layer compared to the power MOSFET. Due to this P+ layer a wide junction of P+N is formed, which makes the P+ layer to inject the minority carriers into the N− drift region when the IGBT is conducting. The conductivity modulation can thus be realized to improve the current density of the IGBT. The conflict has been solved between the high voltage endurance and high on-state resistance that are all introduced by the N− drift region. It should be mentioned that for the punch through (PT) IGBT an N+ buffer layer is added between the P+ substrate and N− drift region.

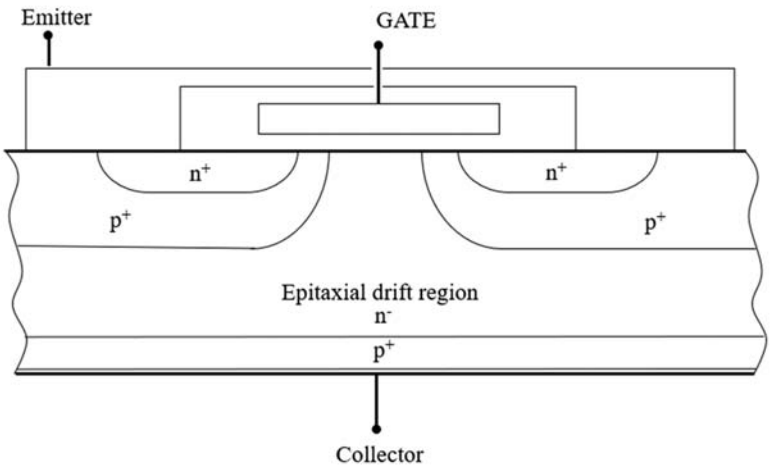


Figure 2.19 Vertical cross-sectional view for an IGBT.

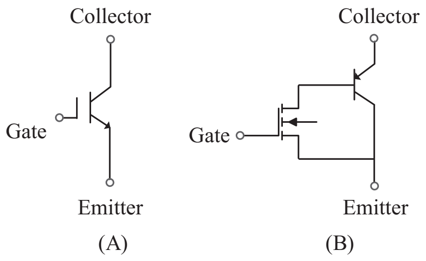


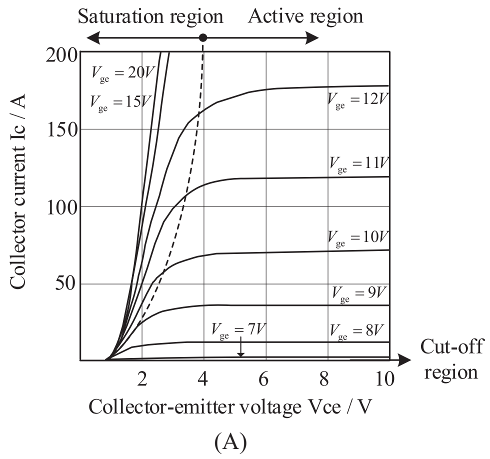
Figure 2.20 (A) IGBT electrical symbol; (B) Darlinton equivalent model of an IGBT.

The p + substrate, n − drift layer, and p + emitter constitute a BJT of which the base current is controlled by the MOSFET gate voltage. A Darlinton configuration shown in Fig. 2.20 is usually used to analyze the IGBT behaviors in which the IGBT is equivalent to PNP BJT driven by a MOSFET. The IGBT is a field-effect transistor as the MOSFET. The turn-on and turn-off are determined by the gate voltage v_{ge}. When v_{ge} is higher than the threshold voltage V_{th} the channel will be induced in the MOSFET and provide the base current to the BJT and the IGBT is thus turned on. When the v_{ge} is applied with a negative voltage or zero the channel disappears and the base current is cut-off, and the IGBT is turn-off.

2.6.2 Characteristics of IGBT

2.6.2.1 Static characteristics

Fig. 2.21A shows the output characteristics of an IGBT. It describes the relationship between the collector current i_c and the collector–emitter voltage v_{ce}. The IGBT output characteristics contain three operating regions. When $v_{ge} < V_{th}$ the IGBT is



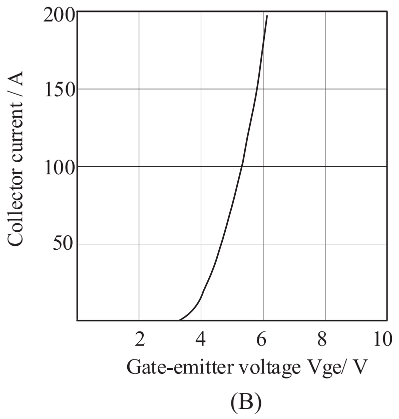


Figure 2.21 (A) Output characteristics of IGBT; (B) transfer characteristics of IGBT.

in the cut-off region. Along with the increase of v_{ge}, there will be current flowing through the IGBT. The device can be operating in the saturation region or the active region. It is important to mention that the definition of the saturation region is different from that of the MOSFET. The boundary between the saturation region and the active region can be identified by the voltage v_{cg}. In power electronic circuits, IGBTs switch between the cut-off region and the saturation region.

Cut off region: $v_{ge} < V_{th}$

Saturation region: $v_{ge} > V_{th}$, and $v_{ce} < v_{ge} - V_{th}$

Active region: $v_{ge} > V_{th}$ and $v_{ce} > v_{ge} - V_{th}$

The transfer characteristics of the IGBTs are shown in Fig. 2.21B, which describes the relationship between the v_{ge} and the collector current i_c under a specific collect-emitter voltage.

2.6.2.2 Dynamic characteristics

The typical switching transient of IGBT is shown in Fig. 2.22. The turn-on transient is similar to that of MOSFET as presented in the last section, which will not be elaborated here. However, due to the recombination of the minority carriers injected into the drift region when IGBT is turned on, the turn-off transient is slightly different from that of the MOSFET.

If the gate voltage is changed to zero or a negative value, the gate-emitter voltage will decrease. The voltage v_{ce} and current i_c remain unchanged until the v_{ge} decreases to a minimum value to sustain the load current [6]. The voltage will then increase dramatically, while the current still remains constant. When v_{ce} reaches the steady turn-off values the diode in the commutating unit will forward-conduct again, which leads to the decrease of i_c. v_{ge} will leave the plateau and decrease to the threshold voltage V_{th}. When $v_{ge} < V_{th}$ the MOS channel is cut-off, and the collector current will reduce significantly but not to zero. The excess stored charge in

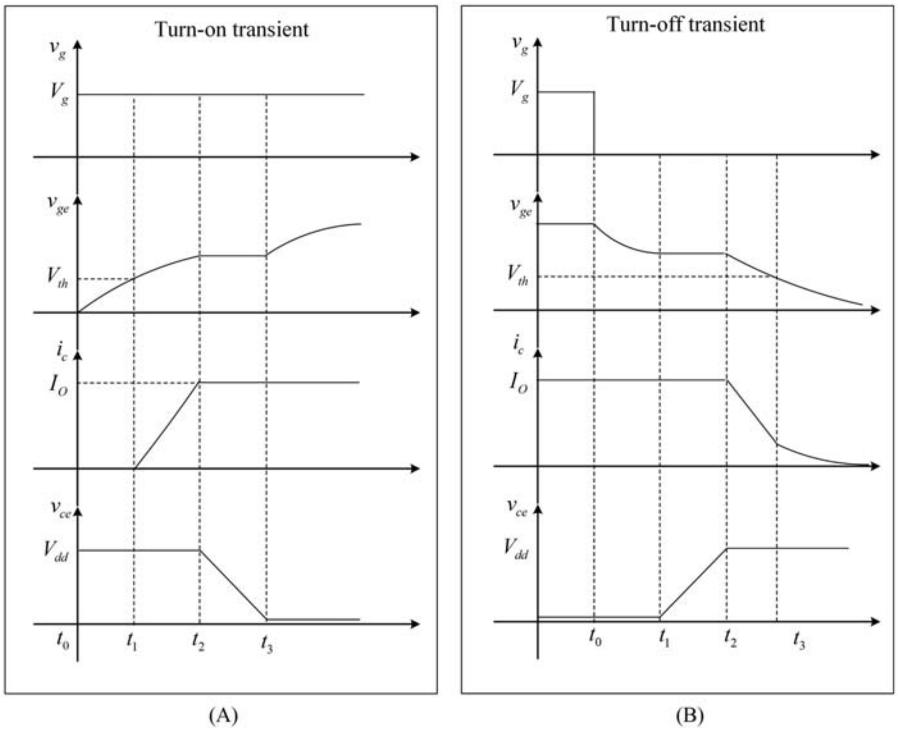


Figure 2.22 Switching transient waveforms of IGBT. (A) Turn-on transient waveforms and (B) turn-off transient waveforms.

the n − drift region during the turn-on state will take a time to recombine until the device is completely cut-off, which will exhibit a tail current at the last stage of the IGBT turn-off transient.

The duration of the tail current is non-neglectable since the lifetime of the excess carrier in the drift region is kept high to decrease the conduction voltage drop. Therefore the power losses are generated in this period because the voltage has reached the turn-off value. Due to the tail current the switching frequency of IGBT is limited, and the benefits of reduced conduction losses can be treated as in cost of the slower switching transient.

Eq. (2.30) gives the expressions of the tail current when the current i_c drops from the steady-state turn-on value I_{on}.

$$i_c(t) = \alpha I_{on} e^{-\frac{t}{\tau_{HL}}} \tag{2.30}$$

where $\alpha = \sec\left(h\frac{l}{L_a}\right)$ is the gain of the bipolar PNP transistor, l is the undepleted base width, L_a is the ambipolar diffusion length, and τ_{HL} is the high-level lifetime of minority carrier.

References

[1] Z. Wang, J. Liu, Power Electronic Technology, fifth ed., China Machine Press, China, 2009 (in Chinese) (Chapter 2).

[2] A. Courtay, MAST power diode and thyristor models including automatic parameter extraction, in: Proceeding of SABER User Group Meeting, Brighton, 1995, pp. 1−10.

[3] Z. Shen, V. Dinavahi, Dynamic variable time-stepping schemes for real-time FPGA-based nonlinear electromagnetic transient emulation, IEEE Transactions on Industrial Electronics 64 (5) (2017) 4006−4016.

[4] M.H. Rashid, Power Electronics Handbook. Devices, Circuits, and Applications, third ed., Butterworth-Heinemann, USA, 2011 (Chapters 2, 3, 4, 5,6).

[5] M. Turzynski, W.J. Kulesza, A simplified behavioral MOSFET model based on parameters extraction for circuit simulations, IEEE Transactions on Power Electronics 31 (4) (2016) 3096−3105.

[6] H. Bai, H. Luo, C. Liu, D. Paire, F. Gao, A. Device-Level, Transient modeling approach for the FPGA-based real-time simulation of power converters, IEEE Transactions on Power Electronics 35 (2) (2020) 1282−1292.

Modeling of power electronic devices

3

3.1 Introduction

As presented in Chapter 2, power electronic devices are the basis of power electronics converters. Therefore to implement accurate modeling of converters, the behaviors of the power electronic devices should be modeled with sufficient accuracy. Different switch models can be used to represent power electronic devices in the simulation. The choices are usually made based on the required accuracy, the acceptable complexity level, and the required computational workload. The more accurate the models are, the more complex and computationally demanding they get. Due to the computing time constraints of the real-time simulation, the power electronic device models should be carefully selected to offer a good balance between accuracy and computational efficiency. Since they behave like a switch in the circuit, nonlinearities and discontinuity will be introduced in the electric network, which will make the network difficult to solve. In real-time applications, two types of electric models are typically used, that is, the static model and the transient model. The static model includes average model, switching function model, ideal model, binary resistor model, and associate discrete circuit model. The average model averages the voltages or the currents in one switching period, which varies with the switching actions, and only the average values of the variables are concerned. The switching function model uses a function to relate the input to the output of the branches containing the switches. The ideal model treats the turn-on state as a short-circuit branch and the turn-off state as an open-circuit branch. The binary resistor model treats the switch as a small resistor in turn-on status and as a large resistor in turn-off status. The associate discrete circuit (ADC) model uses a small inductance to represent the turn-on status and uses a capacitor to represent the turn-off status, while the selected inductance and the capacitance share the same conductance in the discrete-time domain. All the static models describe steady-state turn-on and turn-off states discretely in the simulations and neglect the switching transient. In contrast the transient model includes the switching transient behavior of the power electronic devices in the simulation, bringing the possibility of observing higher-order effects like semiconductor losses and improving the real-time simulation accuracy to the device level.

Furthermore, Electrical and thermal behaviors of power semiconductors are coupled. The junction temperatures will affects the electrical behaviors of the devices, and the power losses in the devices will further induce a temperature rise. A model including electrothermal behavior can improve the accuracy of the power electronic real-time simulation and widen its applications in simulation-based prototyping, validation, and health management.

In this chapter the electric models of the power electronic devices including the static model and the transient model are described. In addition, the thermal models and the electrothermal models of the devices are presented (Fig. 3.1).

3.2 Static model

3.2.1 Average model

The state variables in power electronic converters are usually inductor currents and capacitor voltages. Due to the switching actions, the circuit topology changes, introducing dynamic variations in these state variables. The PWM technique is used for current or voltage modulation. The average model is one of the most used approaches in converter modeling. It relies on separating a state's dynamics into low-frequency and high-frequency dynamics. If the switching effects are averaged over a complete switching cycle, only the low-frequency components of the states remain, and the switching actions will not appear in the system state equations [1,2]. Since often the control involves tracking average power/current and higher-order frequency components are assumed not to affect the power exchange, the average model is usually used in designing the controller of power electronic converters.

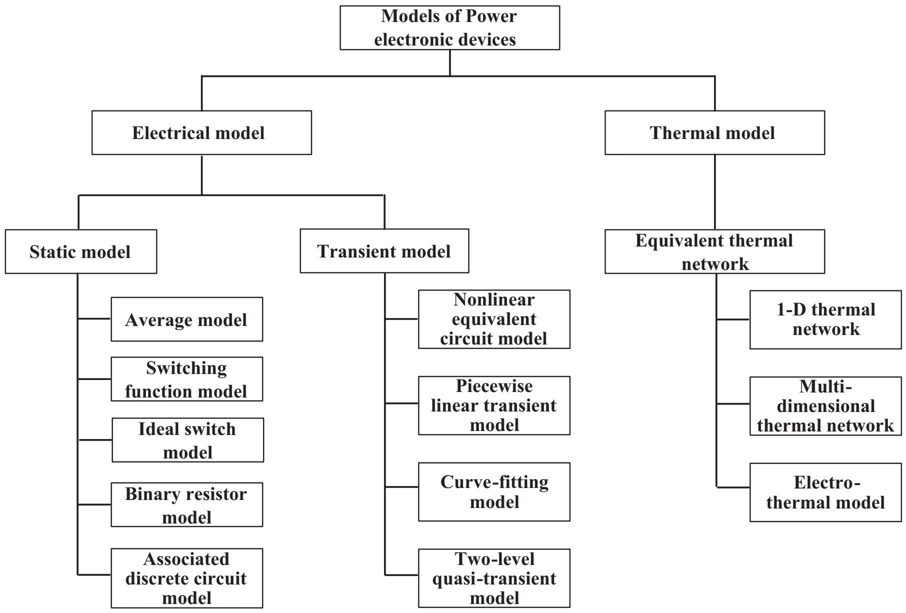

Models of Power electronic devices
Electrical model
Thermal model
Static model
Transient model
Equivalent thermal network
Average model
Switching function model
Ideal switch model
Binary resistor model
Associated discrete circuit model
Nonlinear equivalent circuit model
Piecewise linear transient model
Curve-fitting model
Two-level quasi-transient model
1-D thermal network
Multi-dimensional thermal network
Electro-thermal model

Figure 3.1 Classifications of the power electronic device models.

Let us consider a Boost converter shown in Fig. 3.2 to see how the average model works. It has two discrete states accompanying the switch's on and off statuses, as shown in Fig. 3.3A and B. We can get (3.1) and (3.2) for the turn-on and turn-off periods, respectively. The waveforms are drawn in Fig. 3.4. If the voltage v_{sw} and current i_d are averaged over one switching period using (3.3), the switching unit of the controlled switch and the diode are modeled by a voltage-controlled voltage source and a current-controlled current source, and the boost converter can be modeled by the equivalent circuit in Fig. 3.5.

$$\begin{cases} v_{sw} = 0 \\ i_d = 0 \end{cases} \tag{3.1}$$

$$\begin{cases} v_{sw} = V_{out} \\ i_d = I_L \end{cases} \tag{3.2}$$

$$\begin{cases} \langle v_{sw} \rangle_{T_s} = \dfrac{1}{T_s} \displaystyle\int_t^{t+T_s} v_{sw}(\tau) d\tau = (1-D)V_{out} \\ \langle i_d \rangle_{T_s} = \dfrac{1}{T_s} \displaystyle\int_t^{t+T_s} i_d(\tau) d\tau = (1-D)I_L \end{cases} \tag{3.3}$$

Based on the derivation of the average model-based boost converter, the inductor current is assumed to be continuous, and the model in Fig. 3.5 is adopted for the continuous conduction mode (CCM). The model is supposed to be modified if the discontinuous conduction mode (DCM) is to be considered.

Let us extend the average model to the general half-bridge switching circuit shown in Fig. 3.6A, consisting of two ideal switches in complementary states, a

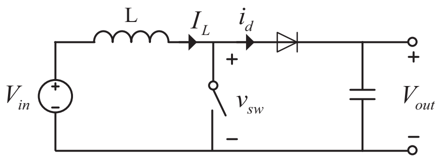


Figure 3.2 Boost converter.

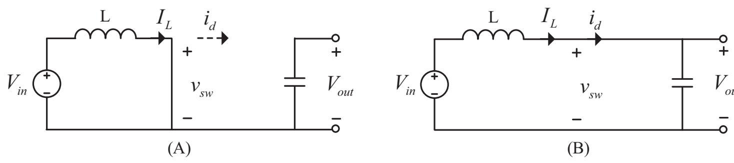


Figure 3.3 Two discrete states of the boost converter.

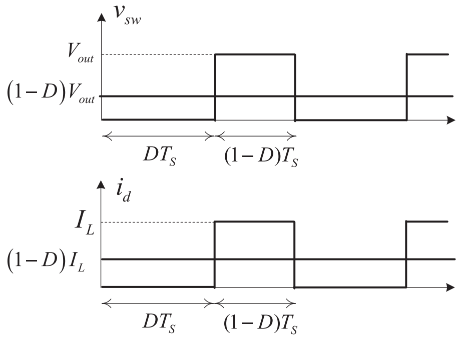


Figure 3.4 Waveforms of v_{sw} and i_d in turn-on and turn-off states.

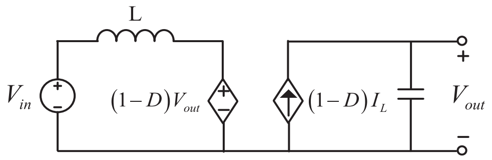


Figure 3.5 The equivalent circuit of the boost converter using the average model.

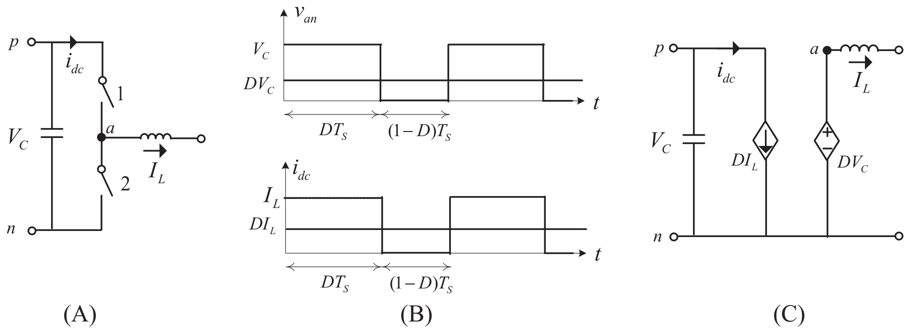


Figure 3.6 Half-bridge switching circuit modeling using the average model. (A) half-bridge switching circuit, (B) DC side current and the output voltage waveforms and (C) the equivalent circuit of half-bridge using average switch model.

capacitor connected to the DC link, and an inductor connected to the middle point of the leg. When the PWM technique is used, the values of the DC side current and the output voltage change along with the switching states, as shown in Fig. 3.6B. By using the average model, the V_{an} and I_{dc} are averaged over one switching cycle, and the equivalent circuit shown in Fig. 3.6C can be obtained, where D is the duty cycle of the upper switch.

Therefore a single-phase inverter can be modeled using the average switch model as shown in Fig. 3.7. D represents the duty cycle of switches No.1 and No.4.

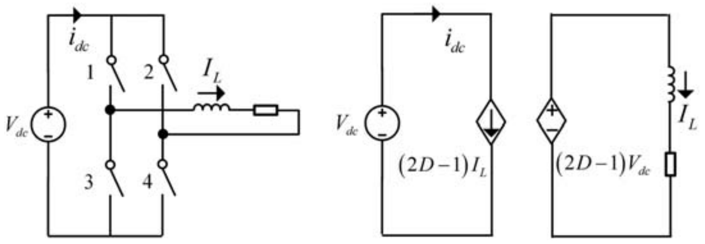


Figure 3.7 A single-phase inverter modeling using the average model.

The use of the average model relies on the analysis of the network states in different switch combinations. This is feasible when the switch number is relatively small and the topology is easy to analyze. When encountering a complex converter model with numerous switches the average model may be inefficient. More importantly, average models may not be a favorable choice for hardware-in-the-loop simulation of power electronics given that the gate signals are usually connected to the models for testing purposes, but the switching actions cannot be represented by the average model in the simulation.

3.2.2 Switching function model

The circuit containing the switches can be modeled with the help of the switching function. The switching function is the representation of a switch as a function taking two values, 1 for the turn-on state and 0 for the turn-off state, represented by *S* with respect to time as given in (3.4). The switching function relates the input to the output in a similar way in (3.5) as the transfer function does.

$$S(t) = \begin{cases} 1 & \text{switch is on} \\ 0 & \text{switch is off} \end{cases} \tag{3.4}$$

$$\text{Output}(t) = \text{Input}(t) \cdot S(t) \tag{3.5}$$

Taking a circuit in Fig. 3.8 as an example the input and output relationship is expressed by (3.6).

$$\begin{cases} V_{out}(t) = V_{in}(t) & S(t) = 1 \\ V_{out}(t) = 0 & S(t) = 0 \end{cases} \tag{3.6}$$

Then, using the superposition theorem to combine two states the output voltage can be expressed by the product of the switching function and the input voltage in (3.7).

$$V_{out}(t) = S(t)V_{in}(t) \tag{3.7}$$

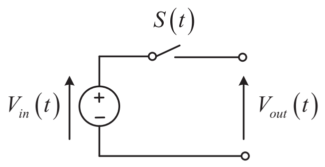


Figure 3.8 Definition of the switching function.

We can use the switching function to describe the general half-bridge circuit in Fig. 3.6A. Since the S_1 and S_2 switches are in a complementary manner if no dead zone is considered, the switching function can be defined by $S_1(t)$.

$$\begin{cases} v_{an} = S_1(t)V_C \\ i_{dc} = S_1(t)I_L \end{cases}, S_1(t) \in \{1, 0\} \tag{3.8}$$

The switching functions in (3.8) are unipolar since they only have values of 0 and 1. In a bridge configuration shown in Fig. 3.7, S_1 and S_4 are switching pairs and they have the same switching function denoted as S_{14}, which is also unipolar. So do S_2 and S_3, with the switching function denoted as S_{23}. The full bridge in Fig. 3.7 includes two half-bridges in Fig. 3.6, and each of them can be expressed by switching functions in (3.9) and (3.10). Therefore a single-phase full bridge can be represented by (3.11). Notice that S_{14} and S_{23} cannot be equal to 1 simultaneously but can be equal to 0, corresponding to the dead zone. $S_B(t)$ thus has the values of 1, -1, and 0, and it is a bipolar switching function.

$$\begin{cases} v_{an} = S_{14}(t)V_{dc} \\ i_{dc1} = S_{14}(t)I_L \end{cases} \tag{3.9}$$

$$\begin{cases} v_{bn} = S_{23}(t)V_{dc} \\ i_{dc2} = -S_{23}(t)I_L \end{cases} \tag{3.10}$$

$$\begin{cases} v_{ab} = [S_{14}(t) - S_{23}(t)]V_{dc} = S_B(t)V_{dc} \\ i_{dc} = [S_{14}(t) - S_{23}(t)]I_L = S_B(t)I_L \end{cases}, S_B(t) \in \{1, -1, 0\} \tag{3.11}$$

3.2.3 Ideal switch model

Power semiconductor devices can be naturally treated as single-pole, single-throw ideal switches, as shown in Fig. 3.9. Consequently the circuit containing the ideal switches can be modeled based on a piecewise linear approach, where for every switch permutation a specific topology is used to represent the power converter.

As an advantage the ideal switch model does not include physical behavior in the model, which may be the case of simplified equivalent models. Moreover the ideal switch approach represents each switch individually and can cover the effects

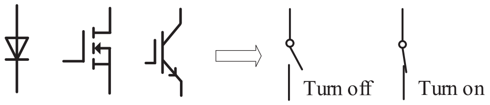


Figure 3.9 Ideal switch model.

of dead time and freewheeling. However, the ideal switch model relies on the dedicated analysis of the models derived from each switching combination and the number of permutation states grows exponentially with the number of switches. Fortunately the models can be precomputed and stored in the solver memory during code compilation, and the exponential growth can be mitigated by employing circuit partitioning methods, as will be presented in Chapter 6.

3.2.4 Binary resistor model

The ideal switch model represents the turn-on state by the short circuit and the turn-off state by the open circuit, which makes the topology vary with the switching actions. Considering that n switches are there in the circuit, there will be theoretically 2^n variations of the topology. It is inconvenient to analyze every possibility of a varying topology in the simulation especially when the switches are numerous. Therefore in the simulation the short-circuit branch is usually replaced by a neglectable resistance, while the open-circuit branch is approximated by a large resistance, as shown in Fig. 3.10. The switches in the circuit can be modeled by a binary resistor in which the value is either small enough or large enough to describe the turn-on and turn-off states.

If we apply the binary resistor switch model to the boost converter, the IGBT and diode shown in Fig. 3.11 will be replaced by two variable resistors in which the values are determined based on the switching status. The inverter represented by the binary resistor model is shown in Fig. 3.12. It should be noted that the combinations of IGBT and its antiparalleled diode in Fig. 3.12 are modeled by a binary resistor since the power flowing through the resistor is bidirectional. The topology of the power converter model is invariant when using the binary resistor model, which reduces the efforts of circuit analysis and departs the identification of the switch status from the network modeling.

3.2.5 Associated discrete circuit model

Using the binary resistor model of switches makes the power electronics topology invariant when the switching events take place and creates the opportunities to use standard network analysis methods to solve the power electronic converters, such as the nodal analysis method. However, the parameters of the network will change with the switching status. Taking the nodal analysis as an example, the nodal voltage equations of a power electronic converter using a binary resistor switch model

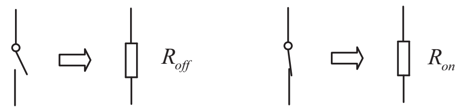


Figure 3.10 Binary resistor switch model.

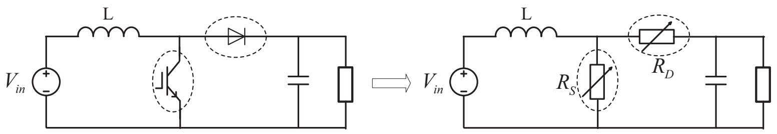


Figure 3.11 Boost converter modeled by the binary resistor model.

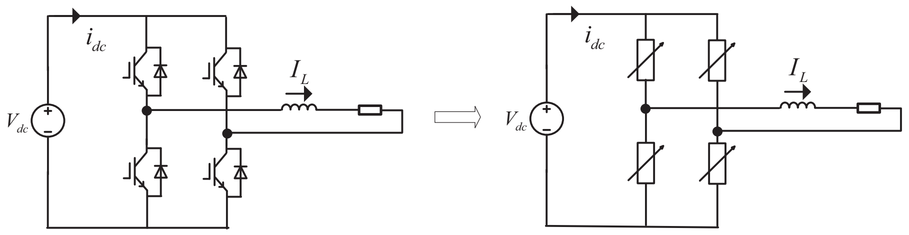


Figure 3.12 Inverter modeled by the binary resistor model.

can be expressed by (3.12). Noting that the nodal analysis will be presented in Chapter 5, please refer to Section 5.4 for detailed explanations.

$$\boldsymbol{Y} \cdot \boldsymbol{v} = \boldsymbol{i} \tag{3.12}$$

The admittance matrix $\boldsymbol{Y}$ has the entries containing the binary resistors. The values of the binary resistors will change with the switching events. As a consequence, the inversion of admittance matrix $\boldsymbol{Y}$ must be recalculated at simulation steps when switching occurs to get the updated value of the nodal voltage, which increases the computing burden in the real-time simulation. Therefore an ADC model has been proposed for the switches by treating them as a small inductor in the turn-on state and a small capacitor in the turn-off state. In the simulation, the voltage and current relationship (VCR) of the inductor and capacitor shown in (3.13) are modeled in the discrete-time domain. Depending on the different methods to discretize the inductor current and capacitor voltage the VCR of these components can be approximated by (3.15) and (3.16).

$$\begin{cases} L\dfrac{di_L}{dt} = v_L \\ C\dfrac{dv_C}{dt} = i_C \end{cases} \tag{3.13}$$

Using the backward Euler method, Eq. (3.14) is obtained.

$$\begin{cases} i_L(t) = \dfrac{\Delta t}{L} v_L(t) + i_L(t-\Delta t) \\ v_C(t) = \dfrac{\Delta t}{C} i_C(t) + v_C(t-\Delta t) \end{cases} \tag{3.14}$$

Rearranging capacitor equations, Eq. (3.15) is equivalent to (3.14)

$$\begin{cases} i_L(t) = \dfrac{\Delta t}{L} v_L(t) + i_L(t-\Delta t) \\ i_C(t) = \dfrac{C}{\Delta t} v_C(t) - \dfrac{C}{\Delta t} v_C(t-\Delta t) \end{cases} \tag{3.15}$$

Using the trapezoidal method, Eq. (3.16) is obtained.

$$\begin{cases} i_L(t) = \dfrac{\Delta t}{2L} v_L(t) + i_L(t-\Delta t) + \dfrac{\Delta t}{2L} v_L(t-\Delta t) \\ i_C(t) = \dfrac{2C}{\Delta t} v_C(t) - \dfrac{2C}{\Delta t} v_C(t-\Delta t) - i_C(t-\Delta t) \end{cases} \tag{3.16}$$

The discrete equations in (3.15) and (3.16) can be reflected in the electric network using a conductance G_S in parallel with a current source J_S, as shown in Fig. 3.13, where

$$i_S = G_S v_S - J_S \tag{3.17}$$

The values of the conductance and the current source corresponding to the discretization methods used are listed in Table 3.1.

The values of current sources depend on the state at the previous simulation step $t - \Delta t$ and is located in vector $\boldsymbol{i}$ in (3.12). To make the admittance matrix $\boldsymbol{Y}$ constant, we can make the conductance in turn-on and turn-off states equal as in (3.18) and (3.19). By carefully selecting the values of L and C to meet (3.18) and (3.19), G_S can be determined to constitute an admittance matrix with fixed entries. With this benefit, Eq. (3.12) can be converted to (3.20) by precomputing the inversion of matrix $\boldsymbol{Y}$. Eq. (3.20) can be computed by the dot product operation, which is very favorable to achieve highly paralleled computing structures to reduce the simulation latency.

The backward Euler method is represented by

$$\frac{\Delta t}{L} = \frac{C}{\Delta t} \Rightarrow LC = \Delta t^2 \tag{3.18}$$

Table 3.1 Parameters of an associate discrete circuit model of switches.

Discrete method	**Turn-on state (inductor)**		**Turn-off state (capacitor)**	
	Conductance G_S	**Current source** J_S	**Conductance** G_S	**Current source** J_S
Backward Euler	$\frac{\Delta t}{L}$	$-i_L(t-\Delta t)$	$\frac{C}{\Delta t}$	$G_S v_C(t-\Delta t)$
Trapezoidal method	$\frac{\Delta t}{2L}$	$-i_L(t-\Delta t)-G_S v_L(t-\Delta t)$	$\frac{2C}{\Delta t}$	$G_S v_C(t-\Delta t)+i_C(t-\Delta t)$

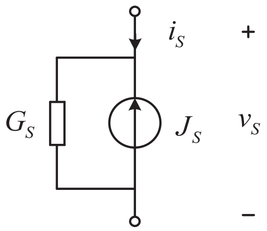


Figure 3.13 Associate discrete circuit model of switches.

The trapezoidal method is represented by

$$\frac{\Delta t}{2L} = \frac{2C}{\Delta t} \Rightarrow 4LC = \Delta t^2 \tag{3.19}$$

$$\boldsymbol{v} = \boldsymbol{Y}^{-1}\boldsymbol{i} \tag{3.20}$$

The main defect of the ADC model is the introduction of virtual losses. The charging and discharging processes of the fictive inductor and capacitor produces virtual power losses, which can be higher than the actual losses and become unacceptable especially in high switching frequency applications.

3.2.6 Determination of switching status

The switch status should be updated in each simulation step to determine the exact value of the switch model. The turn-on and turn-off statuses of the forced commutated switches, such as IGBT and MOSFET, are straightforward to determine according to the gate signals, as shown in Fig. 3.14A. On the other hand, when confronting the naturally commutating switches, such as the diodes, things get more complicated. The diode will turn on when the forward biased voltage is applied, and it will turn off when the current flowing through it decreases to zero, as shown in Fig. 3.14B. The diode status is determined by the voltage across it and current through it, but the instantaneous solutions of the voltage and current in the circuit rely on the diode's status identifications. Therefore an iterative algorithm may be necessary to update the diode status in the simulation. The status determination of the IGBT with an antiparalleled diode is also shown in Fig. 3.14C by combining the logics in Fig. 3.14A and Fig. 3.14B.

The controlled switch statuses are set by the input gate signals, which are deterministic in the current computing time step. The naturally commutated switch statuses are initially set by the voltage and current values computed in the previous time step, called the history terms. The iteration will then start with the switch combinations determined by the history items, the network is solved, and the results will be used to update the statuses of the naturally commutated switches. The iterations will end either when the statuses coincide with that set by the last iteration results or when the iteration number exceeds the predefined maximal value

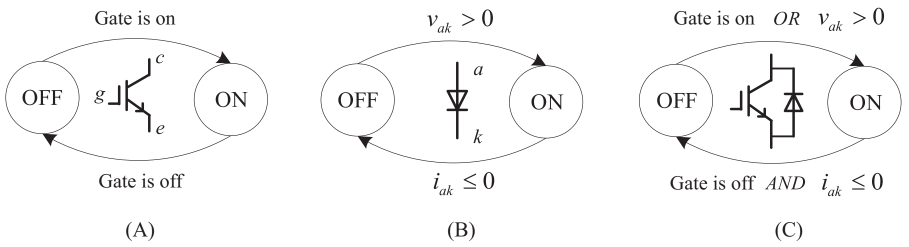


Figure 3.14 Determination of the switching statues.

constrained by the real-time simulation time step. Otherwise the iteration will continue to find the exact switch status. The simulation algorithm in determining the switching statuses is shown in Fig. 3.15.

Let us take the boost converter model shown in Fig. 3.16 using the binary resistor switch model as an example to demonstrate the determination of switching status:

1. Read the input variables V_{in} and the gate signal V_g. If V_g is positive, R_s selects the small resistance value; if not, R_s selects the large resistance value.
2. Load the history values of the variable v_d and i_d, namely $v_d^{t-\Delta t}$ and $i_d^{t-\Delta t}$. Determine the diode status and the value of R_d based on $v_d^{t-\Delta t}$ and $i_d^{t-\Delta t}$ according to Fig. 3.14B.
3. Solve the network and get the values of v_d^t and i_d^t. If the diode switching status determined in step 2 is in accordance with its v_d^t and i_d^t values, the iteration will stop. Otherwise the diode state is changed and the network is resolved again to find new v_d^t and i_d^t.
4. When the iteration ends, update the switch status according to the network solving results and wait for the simulation to step toward.

3.3 Transient model

The static model can only describe the steady-state turn-on and turn-off statuses of the switches in the simulation, which neglects the switching transient and its effects on the network behaviors. The accuracy of the static switch model can be acceptable especially when a large-scale power electronic network is simulated. With the high penetration of the power converters in the power system and electrified transportation, the power electronic devices' transient behaviors will have more effects on the system and the accurate representation of the device in the real-time simulation may become necessary.

The transient model represents the detailed current and voltage behaviors in the simulation which can significantly improve the simulation accuracy of power converters since the model responses will be closer to the real behavior of the converters. The real-time simulation adopting the transient switch model can be treated as the device-level real-time simulation since more detailed device behaviors are described in the simulation. Consequently the advantages of real-time simulation can be enriched by benefiting from the transient models, such as (1) simulating the

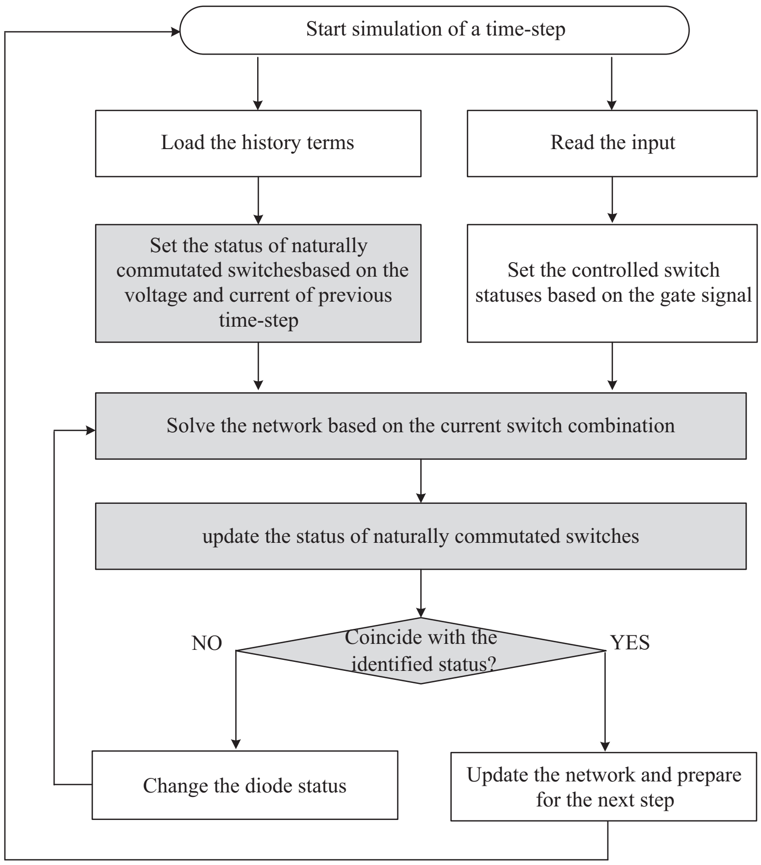

Start simulation of a time-step
Load the history terms
Read the input
Set the status of naturally commutated switchesbased on the voltage and current of previous time-step
Set the controlled switch statuses based on the gate signal
Solve the network based on the current switch combination
update the status of naturally commutated switches
NO
Coincide with the identified status?
YES
Change the diode status
Update the network and prepare for the next step

Figure 3.15 Flowchart describing the simulation algorithm to determine the switch statues.

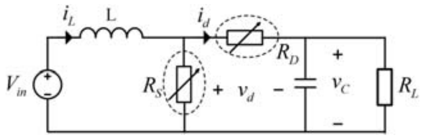


Figure 3.16 Boost converter modeled by the binary resistor model

switching transients and estimating the device current and voltage stresses, (2) calculating device power losses and encompassing the effect of power losses in the simulation, (3) evaluating the electromagnetic interference (EMI) noise, (4) creating opportunities to simulate the thermal and electrothermal characteristics, and (5) assessing the effects of device parasitic parameters. These advantages will further enrich the functionalities of the hardware-in-the-loop simulation. However, implementing the transient models in the real-time simulation can be much more complex than the static models. The computational burden will be heavier since the models usually involve more nonlinearities and discontinuities, and the device-level simulation usually requires a relatively small simulation time step to accurately capture the fast switching transient that is in the range from tens of nanoseconds to hundreds of nanoseconds. Therefore the time constraint imposed in the real-time simulation will be hard to meet since the increased computational workload meets the shorter simulation time step. The trade-off has to be made between the modeling complexity and simulation speed.

The transient switch models can be classified into two categories, the physics-based model and the behavioral model. In the physics-based models, the intrinsic mechanism of the semiconductors is represented by the nonlinear ordinary or partial differential equations. Solving such equations is very computationally intensive in real time. Moreover the parameter extraction requires a deep understanding of semiconductor manufacturing and imposes barriers for researchers outside that field to use them. The physics-based switch model is thus not used in real-time simulation.

Compared to the physics-based model, the behavioral models are more suitable for real-time simulation. The complex physical mechanism is no longer considered, and the macro model and the empirical expression are used instead to describe the switch characteristics. The typical IGBT and diode behavioral models that are widely used in offline simulation software are shown in Fig. 3.17. They use the equivalent circuit components to describe the device behaviors, where the current source i_{mos} models IGBT output characteristics, the controlled current source i_{tail} models tail current, and C_{cg} and C_{ce} model nonlinear interterminal capacitances. For diode, the controlled current source $K \cdot V_{Lrr}$ models diode reverse recovery (RR) current, and the components $R_{d(on)}$ and V_j model diode forward characteristics. By using these equivalent circuits to describe the switches, the model complexity is reduced significantly, making it easier for power electronics practitioners to understand and use the models. Nevertheless, implementing the behavioral model in the real-time simulation is still not easy. The nonlinearities and discontinuities involved require iterative algorithms to accurately solve the model, which is still too complex

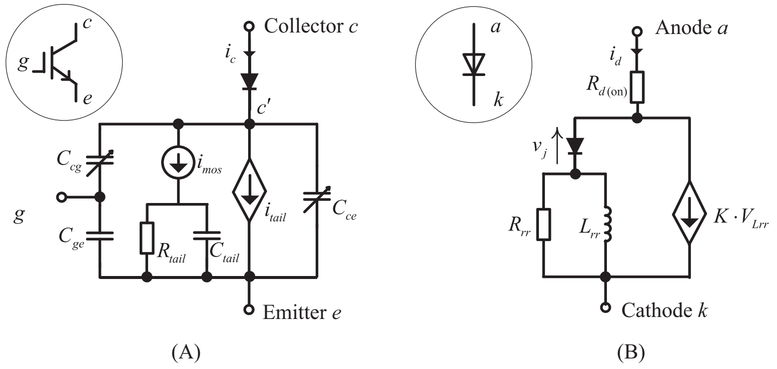


Figure 3.17 (A) IGBT behavioral model; (B) diode reverse recovery model.

to be directly implemented in real-time simulation. Even so, the behavioral models are more suitable for real-time simulations, and with certain approximations and simplifications, they can be applied in real-time simulation to improve the simulation accuracy.

3.3.1 Nonlinear equivalent circuit model

Although the nonlinear behavioral model depicted in Fig. 3.17 is difficult to be directly implemented in real-time simulation, after the feasible simplification and parallelization, it can be suitable for the real-time simulation of the slow switching transient, such as the microsecond-level switching transient of the devices in the high-voltage applications [3].

The equivalent circuit model of IGBT has five nodes. If the model is directly used in the nodal analysis-based simulation (which will be presented in detail in Chapter 5), one IGBT will introduce four nodal voltages in the nodal voltage equations, which enlarges the scale of the admittance matrix. But if we look into the equivalent circuit, the branches related to the tail current and C_{ce} are weakly coupled to the rest of the network. This can be proved by the Jacobian sensitivity analysis of the admittance matrix [3]. The model can be further detached into three parts: an ideal voltage source representing the forward threshold voltage, the tail current unit computing the tail current in turn-off transient, and the MOSFET behavior unit representing the switch in the network, as shown in Fig. 3.18.

As a result, only the MOSFET behavior unit with three nodes participates in the network nodal analysis, and the scale of the admittance matrix is thus reduced. The IGBT simulation results are updated by superposing the MOSFET behavior, the tail current, and the on-state voltage to recover the realistic transient of IGBT. The network containing the MOSFET behavior unit is solved by the Newton–Raphson (N-R) algorithm. The N-R method will be presented in Chapter 6 for a detailed explanation.

The nonlinear equivalent circuit model can be as accurate as the counterparts in the offline simulation. However, the accuracy is at the cost of computing speed. The computational latency grows enormously with the scale of the computing unit and the N-R iteration number. Therefore the applications of nonlinear equivalent circuit model should only be considered for small-scale circuits with a microsecond level switching transient.

3.3.2 Piecewise linear transient model

The difficulties in the nonlinear equivalent circuit model come from two aspects. One is that either IGBT or the diode has different operating regions which cause the discontinuous points in the model; the other is that the voltage and current relationship in each operating region is nonlinear. In order to avoid solving the nonlinear equivalent circuit iteratively, the nonlinear V–I relationship in each operating

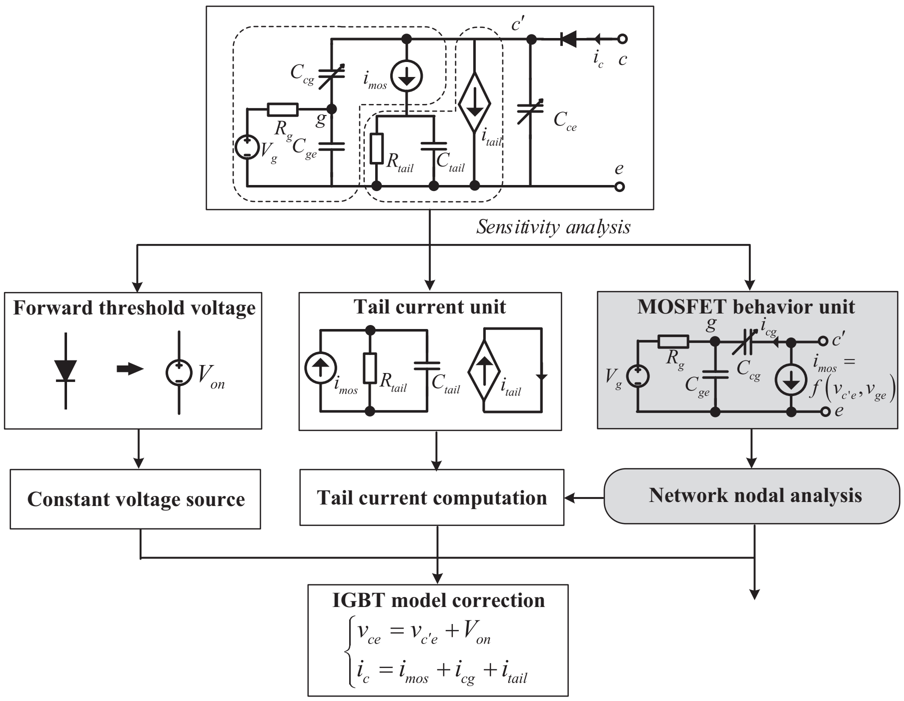


Figure 3.18 Nonlinear equivalent circuit model of IGBT for real-time simulation.

region can be linearized, while the discontinuities between the operating regions can be handled by discrete event-driven mechanisms such as the finite state machine, which leads to a piecewise linear transient switch model [4].

An IGBT–diode commutating pair shown in Fig. 3.19 fed with the constant input current and output voltage is used to analyze the switching transient. The IGBT model shown in Fig. 3.17 has three operating regions, that is, the cut-off region, the active region, and the saturation region, and the diode model shown in Fig. 3.17B has two operating regions separated by whether the ideal diode is turn-on or turn-off. A typical inductive switching transient can be divided into four phases according to IGBT and diode operating conditions, as depicted in Fig. 3.20. Phase 1 and phase 2 are distinguished by the boundary between the IGBT cut-off region and active region; phase 2 and phase 3 are distinguished by the boundary between ideal diode on and off states; phase 3 and phase 4 are distinguished by the boundary between the active region and saturation region.

3.3.2.1 Phase 1 and Phase 2 modeling

In phase 1 and phase 2, IGBT is not fully conducting and the ideal diode is still in the turn-on state. Therefore the diode voltage can be obtained by linearizing the

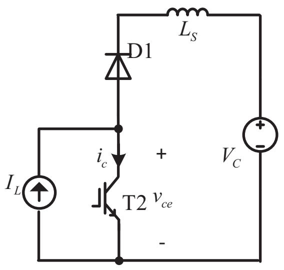


Figure 3.19 IGBT−diode commutating pair.

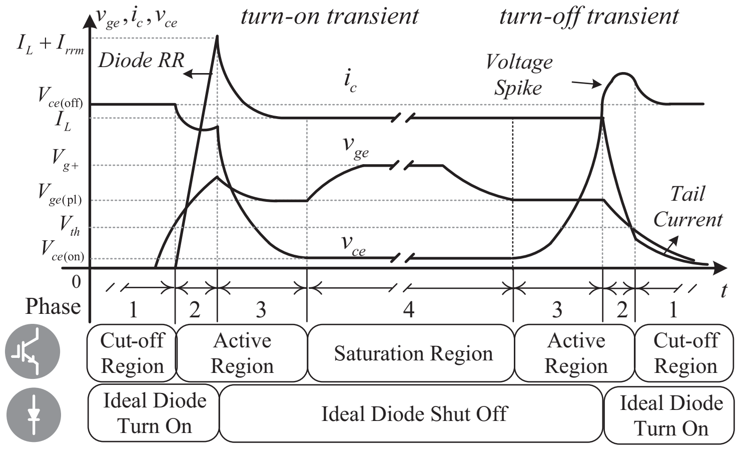


Figure 3.20 Inductive switching transient of IGBT−diode commutating pair.

diode's forward characteristics. The diode voltage V_d can thus be replaced by a voltage source when analyzing the IGBT behaviors, as shown in Fig. 3.21.

$$V_d = V_{d(on)} + R_{d(on)} I_L \tag{3.21}$$

In phase 1, v_{ge} is below the threshold V_{th}, and the IGBT is in the cut-off region. $i_{mos} = 0$, and the equivalent circuit of IGBT is presented in Fig. 3.21.

v_{ce} variations can be represented by (3.22).

$$v_{ce} = V_C + V_d - L_s \frac{di_c}{dt} \tag{3.22}$$

By applying Kirchoff's current law (KCL) to node g, Eq. (3.23) can be obtained,

$$\left(C_{ge} + C_{cg}\right) \frac{dv_{ge}}{dt} = C_{cg} \frac{dv_{ce}}{dt} + \frac{V_g - v_{ge}}{R_g} \tag{3.23}$$

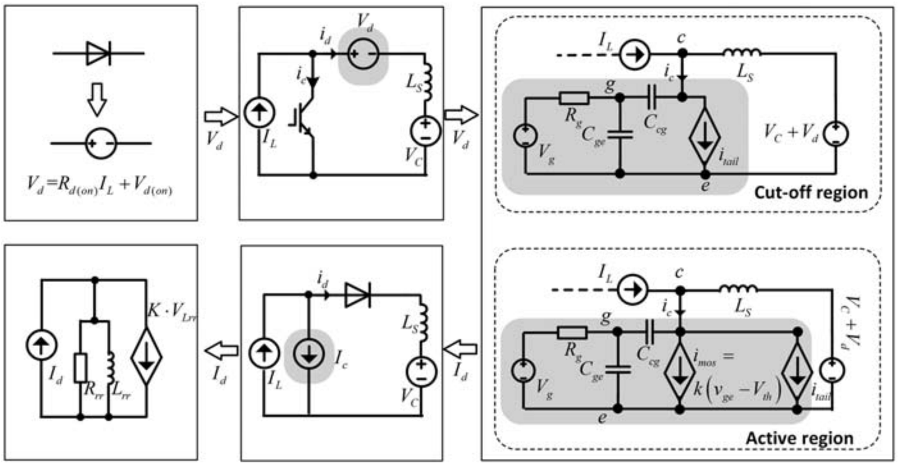


Figure 3.21 Phase 1 and Phase 2 modeling processes.

By neglecting the derivative of v_{ce} in phase 1, Eq. (3.23) can be approximated by

$$\frac{dv_{ge}}{dt} = \frac{1}{R_g\left(C_{ge} + C_{cg}\right)}\left(-v_{ge} + V_g\right) \tag{3.24}$$

The tail current i_{tail} can be unified by (3.25). The coefficient α represents the position in which the tail current starts and can be set as zero in turn-on transient.

$$i_{tail} = \begin{cases} 0 & \text{turn} - \text{on} \\ \alpha\left(\dfrac{V_{Ctail}}{R_{tail}} - i_{mos}\right) & \text{turn} - \text{off} \end{cases} \tag{3.25}$$

v_{Ctail} can be modeled by (3.26). In phase 1, v_{Ctail} is independent from the circuit since $i_{mos} = 0$.

$$\frac{dv_{Ctail}}{dt} = -\frac{1}{R_{tail}C_{tail}} v_{Ctail} + \frac{1}{C_{tail}} i_{mos} \tag{3.26}$$

In phase 2, IGBT operates in the active region, and the relationship between i_{mos} and v_{ge} is subjected to the IGBT transfer characteristics. The transfer characteristic is linearized by (3.27).

$$i_{mos} = k\left(v_{ge} - v_{th}\right) \tag{3.27}$$

By combining (3.22), (3.23), and (3.27) the relationship between v_{ge} and v_{ce} can be formulated as (3.28)

$$\begin{bmatrix} \dot{v}_{ge} \\ \dot{v}_{ce} \end{bmatrix} = M \begin{bmatrix} v_{ge} \\ v_{ce} \end{bmatrix} + N_1 \begin{bmatrix} I_L \\ V_C \end{bmatrix} + N_2 \begin{bmatrix} V_g \\ V_{d(on)} \end{bmatrix} \tag{3.28}$$

where

$$M = \begin{bmatrix} 0 & -\frac{1}{L_s k} \\ \frac{1}{R_g C_{cg}} & -\gamma \frac{1}{L_s k} \end{bmatrix}, \quad N_1 = \begin{bmatrix} \frac{R_{d(on)}}{L_s k} & \frac{1}{L_s k} \\ \gamma \frac{R_{d(on)}}{L_s k} & \gamma \frac{1}{L_s k} \end{bmatrix}, \quad N_2 = \begin{bmatrix} 0 & \frac{1}{L_s k} \\ -\frac{1}{R_g C_{cg}} & \gamma \frac{1}{L_s k} \end{bmatrix},$$

and $\gamma = \frac{C_{ge} + C_{cg}}{C_{cg}}$.

Based on the solution of (3.28), i_c can be obtained by combining (3.27) and (3.25). The current flowing through C_{cg} is neglected since the C_{cg} is of small values in phases 1 and 2.

After obtaining i_d, the diode behaviors in phase 1 and phase 2 can be modeled as (3.29) by treating i_d as an input source to the diode reverse recovery model, where $\tau_{rr} = \frac{L_{rr}}{R_{rr}}$ is the time constant of the diode reverse recovery phenomenon.

$$\begin{cases} \frac{di_{Lrr}}{dt} = \frac{1}{\tau_{rr}(1 + KR_{rr})}(-i_{Lrr} + I_d) \\ V_{Lrr} = \frac{R_{rr}}{(1 + KR_{rr})}(-i_{Lrr} + I_d) \end{cases} \tag{3.29}$$

3.3.2.2 Phase 3 and Phase 4 modeling

In phase 3 and phase 4 the ideal diode in the reverse recovery model is shut off. i_d decays independently as (3.30) and can be treated as an input current source to the IGBT model, as shown in Fig. 3.22.

$$\begin{cases} \frac{di_{Lrr}}{dt} = -\frac{1}{\tau_{rr}} i_{Lrr} \\ i_d = KV_{Lrr} = -KR_{rr} i_{Lrr} \end{cases} \tag{3.30}$$

In phase 3, IGBT still operates in the active region. Applying KCL to node c, Eq. (3.31) can be obtained.

$$\left(C_{ce} + C_{cg}\right)\frac{dv_{ce}}{dt} - C_{cg}\frac{dv_{ge}}{dt} + i_{mos} = i_c \tag{3.31}$$

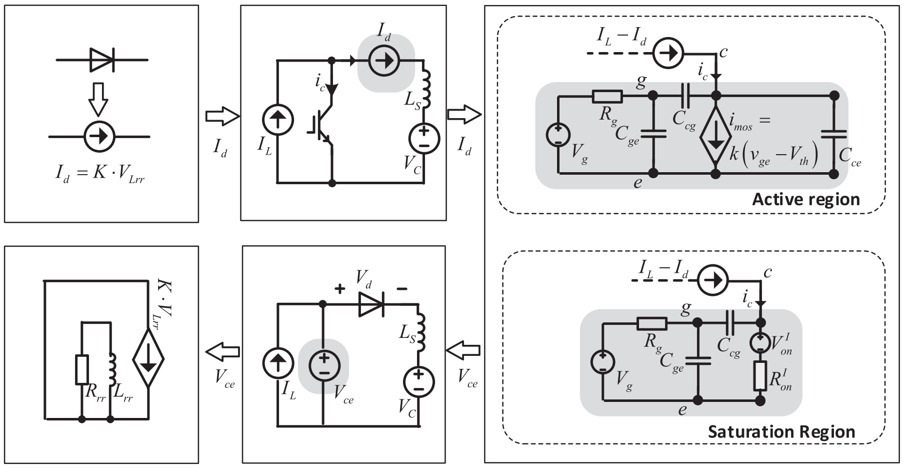


Figure 3.22 Phase 3 and Phase 4 modeling processes.

Combining (3.23), (3.27), and (3.31), v_{ge} and v_{ce} can be formulated as (3.32) and (3.33).

$$\frac{dv_{ge}}{dt} = \delta\left(-\frac{kR_gC_{cg} + C_{oes}}{R_g}v_{ge} + C_{cg}i_c + \frac{C_{oes}}{R_g}V_g + kC_{cg}V_{th}\right) \tag{3.32}$$

$$\frac{dv_{ce}}{dt} = \delta\left(-\frac{kR_gC_{ies} + C_{cg}}{R_g}v_{ge} + C_{ies}i_c + \frac{C_{cg}}{R_g}V_g + kC_{ies}V_{th}\right) \tag{3.33}$$

where $\delta = (C_{cg}C_{ce} + C_{cg}C_{ge} + C_{ge}C_{ce})^{-1}$ and $C_{ies} = C_{ge} + C_{cg}$, $C_{oes} = C_{cg} + C_{ce}$.

i_{mos} can be thereby obtained, and v_{Ctail} can be updated according to (3.26).

In phase 4, IGBT transitions to the saturation region. By linearizing the saturation region the IGBT v_{ce} can be approximated by (3.34).

$$v_{ce} = V_{ce(sat)} = V_{s(on)} + R_{s(on)}i_c \tag{3.34}$$

The diode current is still decaying exponentially as (3.30), and $i_c = I_L - i_d$. v_{ge} is larger than the minimum value to sustain the load current I_L, which is denoted as $V_{ge(pl)}$. The variation of v_{ge} can be modeled by (3.24). Phase 4 will last for the whole on-state period.

3.3.2.3 Phase identifications

The IGBT model is developed into four phases, and each phase is characterized by a continuous model. Four phase models are distinct due to their different operating regions. Therefore each phase can be identified by the specific discrete event that triggers the transition between regions. The solution of the piecewise IGBT/diode transient model involves two aspects, the continuous phase model computation and the effective phase identification. The continuous phase model can be discretized and solved by the implicit integration method. The phase identification process is implemented by zero-crossing detection, which detects the occurrence of the boundary condition and triggers the transition between phases.

The phase identification process can be designed based on the finite state machine in Fig. 3.23. In turn-on transient the solver is first initialized in phase 1. v_{ge} increases in phase 1. Once v_{ge} exceeds the threshold V_{th} the condition of quitting phase 1 is met, and the state machine will transition to phase 2. i_c grows and i_d decreases toward its peak reverse recovery current $-I_{rrm}$ in phase 2. Once i_d is less than KV_{Lrr} the ideal diode shuts off and the state machine will transition to phase 3. v_{ce} decreases toward the saturation value in phase 3. If v_{ce} reaches $V_{ce(sat)}$, the state machine will transition to phase 4 until IGBT turns off.

In turn-off transient the state machine is initialized in phase 4. If v_{ge} decreases below $V_{ge(pl)}$, the IGBT will enter the active region, and the state machine will transition to phase 3. v_{ce} increases toward the off-state value in phase 3. If v_{ce} reaches $V_C + V_{d(on)}$, the diode will turn on, and the state machine will transition to phase 2. v_{ge} decreases in phase 2. If v_{ge} decays below the threshold V_{th}, the state machine will transition to phase 1 until the next switching cycle.

It can be observed that the state machine is initialized based on the gate driving signal, which is independent of the IGBT/diode model. Moreover the transition conditions are designed based on the boundaries of different operating regions. The transitioning between phases is unidirectional in both turn-on and turn-off transients. In the hard-switching condition the IGBT model transitions from phase 1 to phase 4 and then transitions back from phase 4 to phase 1 in a complete switching cycle. Therefore the initialization of the state machine is definite, and the phase transition is unidirectional. The operation of the phase identification can be considered stable, and no chattering is permitted.

3.3.3 Curve-fitting model

The curve-fitting model (CFM) produces the switching waveforms by rescaling the offline measured transient behaviors instead of numerically solving the switch equivalent models as in the nonlinear equivalent circuit model and piecewise linear transient model. The switching waveforms can be measured from either the

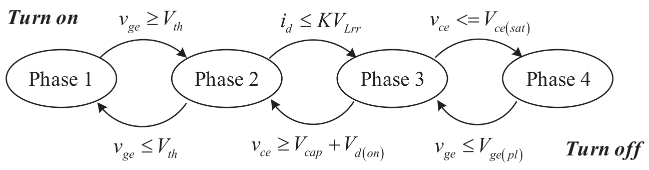


Figure 3.23 The finite state machine of the phase transition in the IGBT/diode model.

experimental results or the offline simulation results of the same topology, as shown in Fig. 3.24. An assumption has been made in CFM that the switching times defined in Fig. 3.24 are constant, which means that the changing rate of transient voltage/current is proportional to their steady-state values. Therefore the measured waveforms can be unified as the per-unit function and stored in the memory unit with a resolution corresponding to the simulation time step. The switching waveforms can thus be rescaled in the real-time simulation by multiplying the per-unit function with the steady-state current and voltage.

The rise and fall times of the waveforms are not constant for different operating conditions. They can also be influenced by the gate resistor, collector current, and device operating temperature. In the CFM, when reshaping the transient waveforms, the overall factors can be considered as well to adapt the waveforms.

Fig. 3.25 depicts the workflow of CFM [5–7]. The steady-state values of IGBT/diode should be computed at first. It relies on either the analysis of all possible modes of the converter or decoupling the half-bridge from the network.

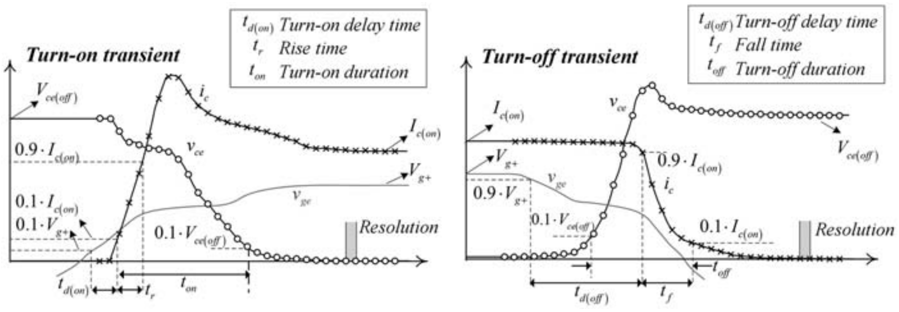


Figure 3.24 Offline measured switching transient waveforms.

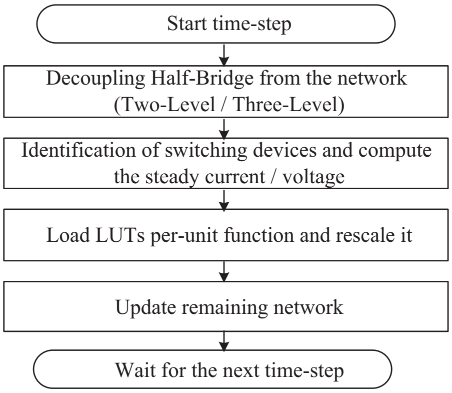

Start time-step
Decoupling Half-Bridge from the network (Two-Level / Three-Level)
Identification of switching devices and compute the steady current / voltage
Load LUTs per-unit function and rescale it
Update remaining network
Wait for the next time-step

Figure 3.25 General workflows of the curve-fitting model of switches.

Furthermore the per-unit function is loaded and rescaled to produce the waveforms. At last the remaining network will be updated. However, the decoupling of the network may cause the numerical instability of the network solver [8]. Therefore the switching waveforms can be represented by the Norton equivalent model based on the predicted transient behaviors, and a global nodal equation is formulated stably embracing all the nonlinearities in the circuit without network decoupling [8].

3.3.4 Two-level quasi-transient model

As discussed previously, the behaviors of switches in the power converters contain two different time scales. One includes the microsecond-level steady-state turn-on and turn-off characteristics, and the other is the nanosecond-level switching transient. By making use of their different time constants, a two-level modeling structure can be used in different applications, referred to as a quasi-transient model (QTM) here. Fig. 3.26 illustrates the modeling structure of QTM. There are two levels of simulation computed in parallel: in the system-level simulation the network adopts the static switch model and is simulated with a microsecond-level time step; in the device-level simulation, switches are represented by the transient model generating the transient waveforms with a nanosecond-level time step to connect the steady-state turn-on and turn-off values. It is called quasi-transient since the transient behaviors are produced based on the system-level simulation results and do not participate in the network solving process. Consequently, we can separate the computation of switching waveforms from the network solver.

3.3.4.1 System-level model

As shown in Fig. 3.27 the system-level model is represented by a combination of resistance R_{on} and voltage source V_{on}, in which the parameters are extracted by linearizing the IGBT saturation region and diode forward characteristics.

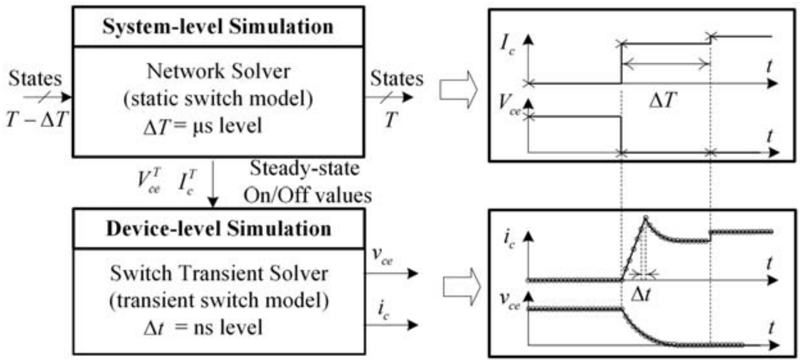


Figure 3.26 Modeling structure of the quasi-transient model.

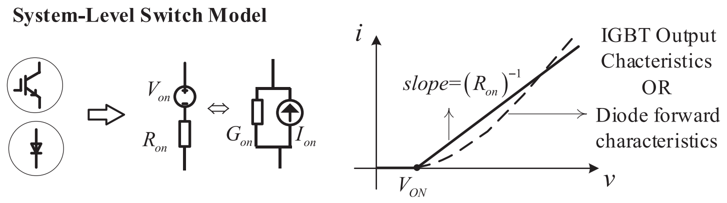

System-Level Switch Model
V_{on}
R_{on}
G_{on}
I_{on}
i
$slope=(R_{on})^{-1}$
V_{ON}
v
IGBT Output Chacteristics
OR
Diode forward characteristics

Figure 3.27 System-level switch model.

3.3.4.2 Device-level model

QTM modeling framework gives the flexibility to implement different models in the device-level simulation. One typical model is called the datasheet-driven model (DDM) [9−11], where the rise and fall processes in the transient waveforms are modeled by straight lines, as shown in Fig. 3.28. The rise time and fall time of the current waveforms are collected from IGBT datasheet, and those of voltage waveforms are computed with the help of switching power losses in the datasheet [10]. Therefore the DDM can provide a precise estimation of power losses. The switching times are assumed to be proportional to the final values of current and voltage, according to which the switching times are revised and used for the waveform generation [11]. The DDM simulates the switching transient with an ultra small time step (such as 10 ns). However, in the DDM the modeled switching waveforms are too ideal and lose some essential transient features such as the current tail and voltage spikes.

Another typical model is called the high-resolution quasi-transient model (HRQTM) [12,13], where the transient behaviors are divided into different stages and modeled by several linear or exponential segments, as shown in Fig. 3.29. The HRQTM computes the slopes, time constants, and intervals of each stage, based on which HRQTM generates switching waveforms with small computing efforts. This can enable the simulation of the fast switching transient under 20 ns. However, the computation of transient parameters introduces additional computation amounts to the system-level simulation. The stray inductance is not modeled either in HRQTM.

The third typical model of QTM is the Hammerstein configuration model (HCM)-based switching transient model. Hammerstein configuration models a physical system by representing the static and dynamic characteristics separately. It is usually constituted by a nonlinear static block followed by a linear dynamic block, which can be used to model the switch's transient behaviors. However, the nonlinearity of the static model requires an iterative numerical solver. A modified HCM replaces the nonlinear static block with a linear one to reduce the computing burden in real-time simulation [14,15]. The system-level model is used as the static block, and the first-order RC circuit is used as the dynamic block, as shown in Fig. 3.30. By feeding the first-order circuit with the values of the system-level switch, the dynamic variation of the equivalent conductance in turn-on transient and the dynamic variation of the companion current source in turn-off transient can be approximated. The parameters of the first-order circuit are obtained from the

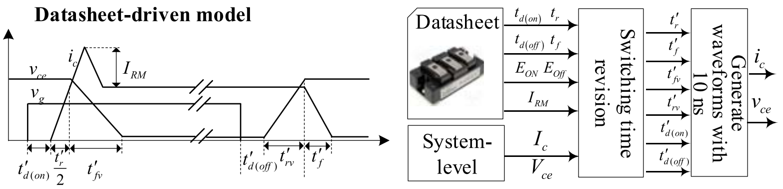


Figure 3.28 Illustrative diagrams of the datasheet-driven model.

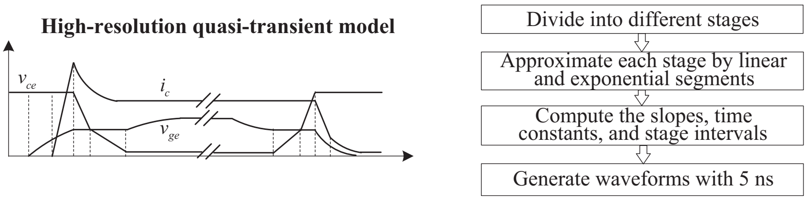

High-resolution quasi-transient model
v_{ce}
i_c
v_{ge}
Divide into different stages
Approximate each stage by linear and exponential segments
Compute the slopes, time constants, and stage intervals
Generate waveforms with 5 ns

Figure 3.29 Illustrative diagrams of the high-resolution quasi-transient model.

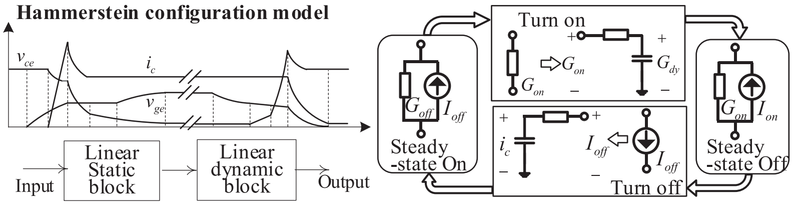


Figure 3.30 Illustrative diagrams of Hammerstein configuration model.

current switching times in the datasheet, and the current waveforms can then be accurately reproduced. Moreover the voltage waveforms are reconstructed based on power losses. A 100 ns device-level simulation time step is realized in [14,15]. HCM has more sophisticated transient waveforms but has lower transient resolution than previous counterparts.

In general, the QTM can handle the fast switching transient very well due to the small transient time step. Nevertheless, QTM can be accurate only if steady-state current and voltage have much larger time constants than the switching transient. Moreover, it has few contributions to the improvement of the network simulation accuracy since the network is only solved by the static model.

3.3.4.3 High-resolution quasi-transient model example

Let us take the HRQTM as an example to demonstrate the QTM models. The system-level simulation uses the static switch model shown in Fig. 3.27 and produces the steady-state turn-on and turn-off values of the switches.

The IGBT transient model and its transient parameter computation are essential to generating the switching waveforms. A simplified IGBT behavior model from Fig. 3.17A is shown in Fig. 3.31. The voltage across and the current through IGBT (v_{ce} and i_c) are modeled separately to describe its port characteristics. The capacitance C_{ge} is treated as a constant value and the C_{cg} is modeled into two different values by piecewise linearizing the capacitance curves from the datasheet. Thus the relationships between v_{ce} and v_{ge} can be derived by (3.35).

$$C_{ge}\frac{dv_{ge}}{dt} = C_{cg}\frac{d\left(v_{ce} - v_{ge}\right)}{dt} + \frac{V_g - v_{ge}}{R_g} \tag{3.35}$$

The transient current waveforms are derived mainly based on the transfer characteristics when IGBT operates in the active region. Moreover the tail current at the end of the turn-off transient is modeled by the exponential decay based on the model in Fig. 3.17A. The typical transient turn-on and turn-off waveforms of an IGBT are depicted in Fig. 3.32.

IGBT switching transient is divided into five stages in both turn-on and turn-off transients. The current and voltage waveforms in each stage are represented by the linear segments and exponential segments. By analyzing the switching transient

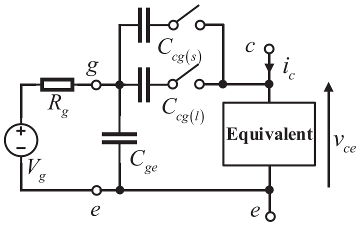


Figure 3.31 IGBT simplified equivalent model.

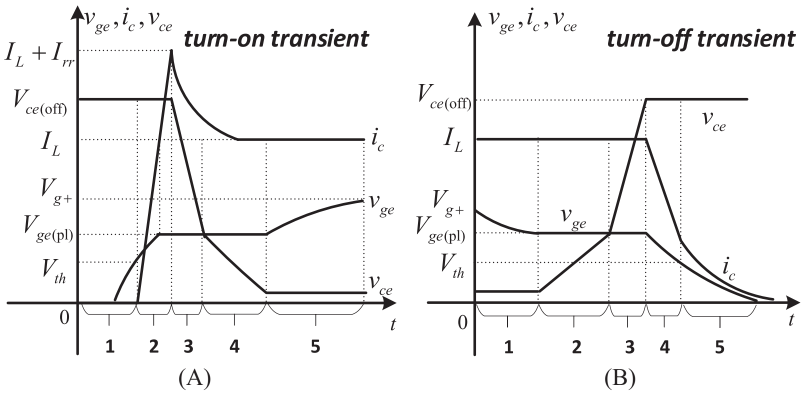


Figure 3.32 Switching transient waveforms modeled by HRQTM.

stage by stage according to the simplified equivalent model, the essential parameters that are needed to generate the transient waveforms can be computed, including the stage intervals, the slope of the linear variations, the time constant, and initial values of the exponential variations. Therefore the computation of the dynamic model consists of only two formulas, (3.36) and (3.37), where x represents the linear segment, y represents the exponential segment, x_0, y_0 are the initial values, u is the input of the first-order circuit, k is the slope, and τ is the time constant.

$$x(t) = kt + x_0 \tag{3.36}$$

$$y(t) = (y_0 - u)e^{-\frac{1}{\tau}t} + u \tag{3.37}$$

Therefore the computation of generating dynamic switching waveforms now concentrates on the two Eqs. (3.36) and (3.37). This creates the conditions of generating transient waveforms with a small time step in the device-level simulation, as shown in Fig. 3.26.

To concise the illustration, we list the equations to obtain the transient parameters in Table 3.2 and Table 3.3 for both the turn-on and turn-off transients. For the readers who are interested in the detailed derivation of these equations, referring to [13] is suggested.

3.4 Thermal model

The thermal behavior of the power electronic devices is essential for the accuracy and fidelity of the power electronic real-time simulation. The thermal behavior emulated in the real-time simulation can help the designer to evaluate the junction temperature to ensure that a device will stay in the safe operating region. Meanwhile the junction temperature variations will affect the electrical behaviors of the power electronic devices. An accurate thermal model as well as the electrothermal model improves the fidelity of real-time simulation and its effectiveness in real-time simulation-based testing.

The simulations of junction temperatures of the power electronic devices are usually accomplished by the finite-element analysis (FEA) and thermal network modeling methods. The FEA can acquire the temperature at any position of the module, but there is intensive computing time and the dynamic thermal behaviors varying with the load cannot be predicted. The thermal network model is established based on the thermal—electrical analogy. By using the RC network to model the thermal resistance and thermal capacitance, the thermal parameters are converted to the electrical parameters, in which the junction temperatures can be simulated using the electric network analysis approaches.

Table 3.2 Transient parameters required to generate turn-on waveforms.

	Δt_{on}	v_{ge}	v_{ce}	i_c
Stage1	$-\tau_1 ln\left(\frac{V_{g+}-V_{th}}{V_{g+}-V_{g-}}\right)$	$\tau_1 = R_g\left(C_{ge}+C_{cg(s)}\right)$	$v_{ce} = V_{ce(off)}$	$i_c = 0$
Stage2	$-\left(1+\frac{I_{rr}}{I_L}\right)\tau_1 \ln\left(\frac{V_{g+}-V_{ge(pl)}}{V_{g+}-V_{th}}\right)$			$k_{ir} = \frac{I_L}{-\tau_1 \ln\left(\frac{V_{g+}-V_{ge(pl)}}{V_{g+}-V_{th}}\right)}$
Stage3	$\frac{V_{ge(pl)}-V_{ce(off)}}{k_{vf1}}$	$v_{ge} = V_{ge(pl)}$	$k_{vf1} = \frac{V_{ge(pl)}-V_{g+}}{R_g C_{cg(s)}}$	$i_c = I_L + I_{rr}e^{-\frac{1}{\tau_{rr}}t}$
Stage4	$\frac{V_{ce(on)}-V_{ge(pl)}}{k_{vf2}}$		$k_{vf2} = \frac{V_{ge(pl)}-V_{g+}}{R_g C_{cg(l)}}$	$i_c = I_L + I_{rr}e^{-\frac{1}{\tau_{rr}}t}$
Stage5	/	$\tau_2 = R_g\left(C_{ge}+C_{cg(l)}\right)$	$v_{ce} = V_{ce(on)}$	$i_c = I_L$

Table 3.3 Transient parameters required to generate turn-off waveforms.

	Δt_{off}	v_{ge}	v_{ce}	i_c
Stage1	$-\tau_2 ln\left(\frac{V_{g-} - V_{ge(pl)}}{V_{g-} - V_{g+}}\right)$	$\tau_2 = R_g\left(C_{ge} + C_{cg(l)}\right)$	$v_{ce} = V_{ce(on)}$	$i_c = I_L$
Stage2	$\frac{V_{ge(pl)} - V_{ce(on)}}{k_{vr1}}$		$k_{vr1} = \frac{V_{ge(pl)} - V_{g-}}{R_g C_{cg(l)}}$	
Stage3	$\frac{V_{ce(off)} - V_{ge(pl)}}{k_{vr2}}$	$v_{ge} = V_{ge(pl)}$	$k_{vr2} = \frac{V_{ge(pl)} - V_{g-}}{R_g C_{cg(s)}}$	
Stage4	$-\tau_1 ln\left(\frac{V_{g-} - V_{th}}{V_{g-} - V_{ge(pl)}}\right)$	$\tau_1 = R_g\left(C_{ge} + C_{cg(s)}\right)$	$v_{ce} = V_{ce(off)}$	$k_{imos} = \frac{-I_L}{\Delta t_{off4}}$ $i_0 = \alpha k_{imos}\tau_{tail}\left(e^{-\frac{\Delta t_{off4}}{\tau_{tail}}} - 1\right)$ $k_{if} = \frac{i_0 - I_L}{\Delta t_{off4}} = \left(1 - \frac{i_0}{I_L}\right)k_{imos}$
Stage5	/	$\tau_1 = R_g\left(C_{ge} + C_{cg(s)}\right)$		$i_c = i_0 e^{-\frac{1}{\tau_{tail}}t}$

3.4.1 Equivalent thermal network

3.4.1.1 Elements of thermal networks

Two fundamental physical elements constitute the thermal network, thermal resistance and thermal capacitance. There are also the sources of heat and temperature.

Considering that there is an object with a thermal resistance of R_{12}, one side of the object is at the temperature T_2 and the other side is at T_1. The corresponding thermal network based on the thermal electrical analogy can be depicted in Fig. 3.33. Noticing that the temperature differences are usually of interest rather than the absolute temperature, the reference is taken to measure the temperatures relative to it, and the reference temperature is assumed to be constant, which is in analogy with the fact that we select a node to be the ground and assign the ground to be zero volts in the electric network.

In addition to the heat transfer, the object can also store the heat. Thermal capacitance is used to model the stored heat in an object. The temperatures will rise as the heat flows into it, and the temperature will decrease as the heat flows out. The thermal–electrical analogy of thermal capacitance is depicted in Fig. 3.34, where one end of the capacitor is connected to the constant ambient temperature.

The thermal sources in the thermal model represent the heat sources in the system. It is represented by a current source that can be constant or variant with time, such as the current source q in Fig. 3.34.

The temperature source used in the thermal model represents a constant temperature. An ideal temperature source maintains a given temperature independent of the

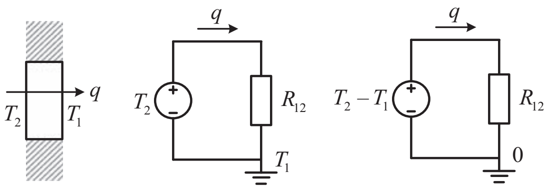


Figure 3.33 Thermal resistance modeling.

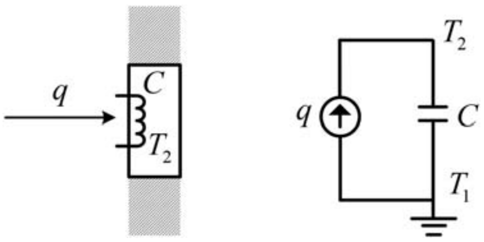


Figure 3.34 Thermal capacitance modeling.

amount of heat required. We will assume that the temperature of the ambient surroundings is constant regardless of the heat flow in or out.

3.4.1.2 Cauer model and Foster model

The thermal behavior of semiconductor components can be described using various equivalent circuit models. The continued fraction circuit which is also called the Cauer model reflects the real, physical setup of the semiconductor based on thermal capacitances with intermediary thermal resistances, as shown in Fig. 3.35A [16]. The model can be set up, where the material characteristics of the individual layers are known. However, the correct mapping of the thermal spreading on the individual layers is problematic. The individual RC elements can then be assigned to the individual layers of the module (chip, chip solder, substrate, substrate solder, baseplate). The network nodes, therefore, allow access to the internal temperatures of the layer sequence.

The partial fraction circuit which is called the Foster model is another commonly used way to build the thermal network. In contrast to the Cauer model, the individual RC elements of the Foster model no longer represent the layer sequence. The network nodes do not have any physical significance. This illustration is usually used in datasheets as the coefficients can be easily extracted from a measured cooling curve of the module.

The thermal impedance of a Foster model can be expressed as (3.38)

$$Z_{th}(t) = \sum_{i=1}^{n} R_i\left(1 - e^{-\frac{1}{\tau_i}t}\right) \tag{3.38}$$

where $\tau_i = R_i C_i$.

With given switching and forward losses $P_L(t)$ and under the assumption of a known case temperature $T_c(t)$ the junction temperature $T_j(t)$ can be determined as follows:

$$T_j(t) = P_L(t)Z_{th(j-a)}(t) + T_{amb} \tag{3.39}$$

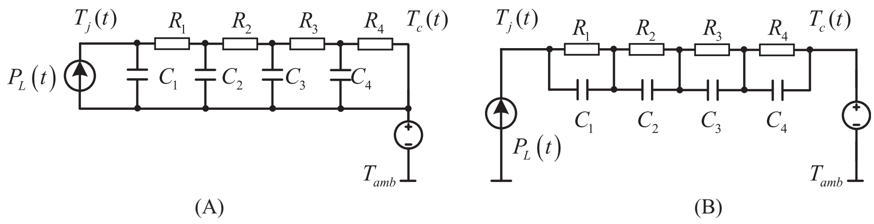


Figure 3.35 (A) Cauer model; (B) Foster model.

3.4.1.3 Thermal system model

The Cauer model and Foster model in Fig. 3.35 can represent the heat transfer from the junction to case. To create a complete thermal system model of the power semiconductor module used in real applications, the thermal transfer from the case to the ambience should be added into the device thermal model.

Fig. 3.36 shows the thermal system model based on the Cauer model and Foster model. The Cauer model is set up by material analysis and FEM simulation of the individual layers of the entire setup, and the Foster model is set up by curve-fitting the transient thermal impedance in (3.38). In addition to the thermal model of the IGBT devices, the thermal model of the grease and heat sink are included.

3.4.1.4 Thermal effect corrections

Neither the Cauer thermal model nor the Foster model considers the effects that the temperature variations can cause on the device material. In fact, the thermal conductivity of each layer in power modules is affected by the operating temperature. Therefore errors will be introduced in the junction temperature estimation, if the thermal effects of materials is not properly considered [17].

Fig. 3.37 shows that the Si and SiC thermal conductivity varies with the temperature, based on which the thermal resistance of the device can be corrected by curve-fitting the function between the thermal resistance and the temperature. The thermal network model considering the materials' thermal effects is shown in Fig. 3.38. The thermal resistance R_1 is modeled by a voltage-controlled resistance, in which the value varies with temperature to simulate the thermal resistance–temperature relationship.

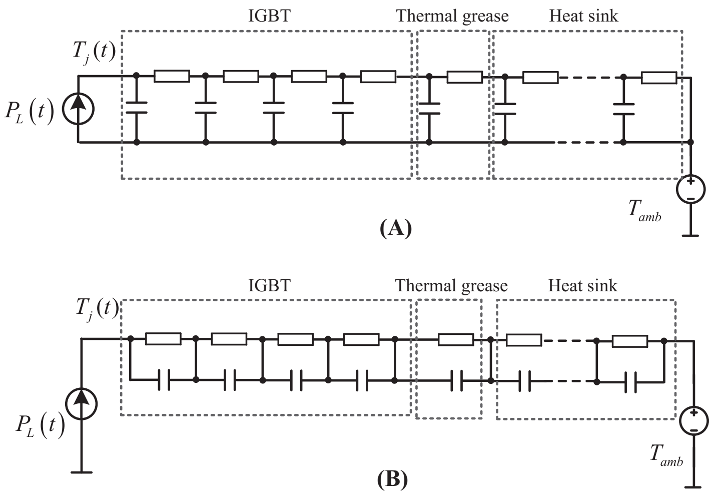


Figure 3.36 Thermal system model based on (A) Cauer model and (B) Foster model.

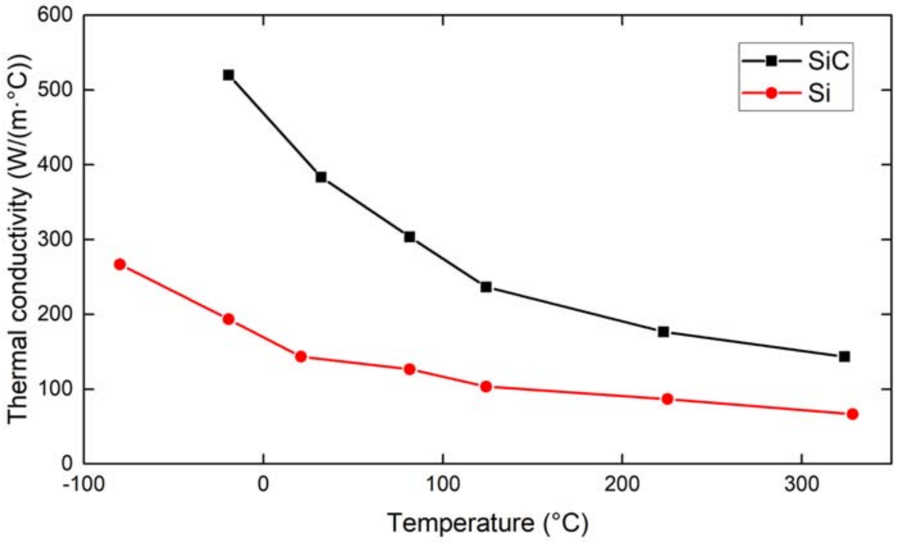


Figure 3.37 Relationships between thermal conductivity and temperature of materials.

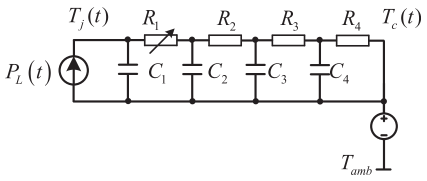


Figure 3.38 Cauer model considering the thermal effect.

3.4.2 Multidimensional thermal network

The Cauer model and Foster model presented in the previous sections are unidimensional since they only describe the thermal diffusions in the perpendicular directions of the power module. If we consider the thermal coupling between different chips and different base plates the thermal model can be treated as two-dimensional (2D) since transverse resistances are included.

3.4.2.1 Paralleled device thermal model

In high-power applications, the power electronic devices are usually used in a paralleled manner and mounted on the same heat sink. The thermal coupling between the paralleled chips will have significant effects on the junction temperature simulation results. If transverse resistances are added to the traditional thermal network model to simulate the thermal coupling in the cooling path, the thermal model will be expanded to the 2D model and the junction temperature can be simulated more accurately.

A modified thermal network of the paralleled devices is shown in Fig. 3.39 [17]. Two modules are modeled, each of which contains one IGBT chip and one antiparalleled diode chip. The power sources represent the power losses of four chips. The thermal resistance in the gray box describes the heat conductions in the transverse direction.

3.4.2.2 Multiple heat source coupling thermal model

A paralleled device thermal model only considers the thermal coupling between two modules mounted on the same heat sink. The high-power module is usually made by packaging multiple paralleled chips, and the heat source coupling between chips cannot be neglected.

The equivalent coupling thermal impedance can be introduced to describe the junction temperature rise of the target chip when the unit power losses are applied from any other chip in the surrounding, as given in (3.40). It represents that the steady-state maximum junction temperature of chip n changes from T_{amb} to T_{jn} when the power losses P_m are applied to chip m.

$$Z_{th(n,m)} = \left(T_{jn} - T_{amb}\right)/P_m \tag{3.40}$$

The thermal network model will contain the self-thermal-impedance $\boldsymbol{Z_{self}}$ and the coupling thermal impedance $\boldsymbol{Z_{couple}}$, and the junction temperature can be computed by the equation when considering the thermal coupling between multiple chips.

$$\mathbf{T_j} = \left(\mathbf{Z_{self}} + \mathbf{Z_{couple}}\right)\cdot \mathbf{P_{loss}} + \mathbf{T_{amb}} \tag{3.41}$$

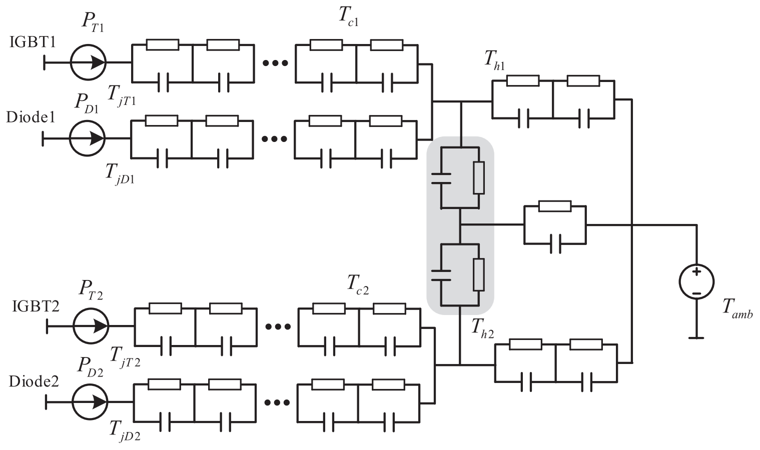


Figure 3.39 Thermal network of paralleled devices.

where

$$Z_{couple} = \begin{bmatrix} 0 & Z_{th(1,2)} & \cdots & Z_{th(1,n)} \\ Z_{th(2,1)} & 0 & \cdots & Z_{th(2,n)} \\ \vdots & \vdots & \ddots & \vdots \\ Z_{th(n,1)} & Z_{th(n,2)} & \cdots & 0 \end{bmatrix}, Z_{self} = \begin{bmatrix} Z_{th(1,1)} & 0 & \cdots & 0 \\ 0 & Z_{th(2,2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Z_{th(n,n)} \end{bmatrix}.$$

Based on (3.41) the thermal network considering the coupling of multiple heat sources can be built as Fig. 3.40 [17]. $P_{loss1},\ldots,P_{lossn}$ represent the power losses of each chip, which constitute vector $\mathbf{P_{loss}}$ in (3.41). $Z_{th\_jc1},\ldots,Z_{th\_jcn}$ represent the thermal impedance from the junction to case of each chip. $Z_{th\_ch1},\ldots,Z_{th\_chn}$ represent the thermal impedance from the case to heat sink of each chip. Z_{th_ha} represents the thermal impedance of the heat sink. $T_{j1},\ldots,T_{jn}$ represent the junction temperature, $T_{c1},\ldots,T_{cn}$ represent the case temperature, and T_h represents the heat sink temperature. $Z_{th(n,m)}$ is the equivalent cooling thermal impedance.

3.4.2.3 Dimensions of thermal networks

To summerize, the Cauer model and Foster model presented only describe the thermal diffusions in the perpendicular directions of the power module, and it can be treated as a 1D thermal network. The thermal network considering thermal coupling between parallel devices and between multiple heat sources can be regarded as a 2D thermal network since the transverse resistances are added. The heat flow path should be in space dimension according to the heat transfer mechanisms. The most accurate thermal network should be developed in 3D by considering the heat flow in perpendicular, transverse, and longitudinal directions [18]. However, such a model needs dedicated analysis and parameter extractions. The complex network

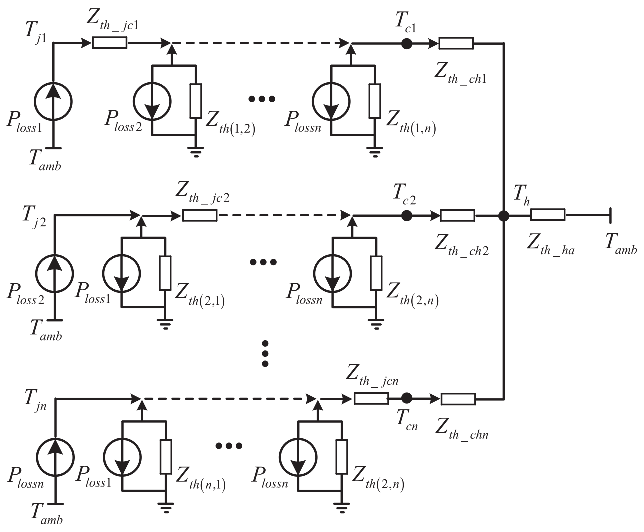


Figure 3.40 Thermal network considering the coupling of multiple heat sources.

will also introduce a large computing amount which is not favorable in real-time simulation. The 3D model will not be presented in this book.

3.4.3 Electrothermal model

The power loss leads to a temperature variation of the junction, which in turn affects the characteristics of switches. Fig. 3.41 shows the IGBT output characteristics of Infineon FF450R12ME4 captured from the datasheet. It can be seen that the IGBT characteristics are strongly affected by the variation of junction temperatures. Therefore, for a given power module, the electrical behaviors and thermal behaviors are coupled, and the more accurate representations of the switching behaviors should synthesize the electrical and thermal characteristics that constitute the electrothermal model.

The basic structure of the electrothermal model is shown in Fig. 3.42. The device voltage, current, and switching status from the electrical network simulation are used for the power loss computation. The power losses are used for the heat source of the thermal model, while the junction temperature obtained by the thermal model can be used for updating the thermal-sensitive electrical characteristics such as the IGBT output characteristics, transfer characteristics, and diode reverse recovery

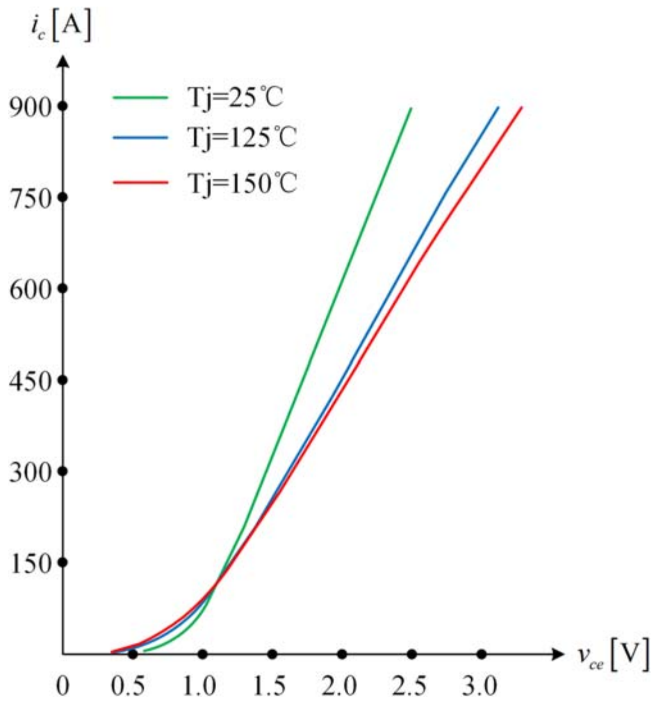


Figure 3.41 Output characteristics of IGBT (Infineon FF450R12ME4).

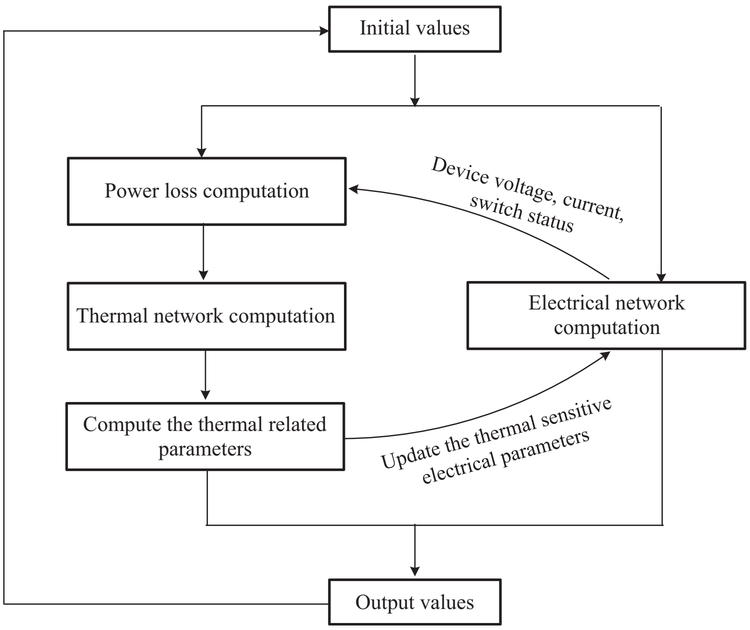

Initial values
Power loss computation
Device voltage, current, switch status
Thermal network computation
Electrical network computation
Compute the thermal related parameters
Update the thermal sensitive electrical parameters
Output values

Figure 3.42 Structure of the power electronic electrothermal model.

characteristics. The electrical behaviors and thermal behaviors are thus coupled and simulated together to restore the power electronic characteristics.

The typical characteristics related to junction temperature are usually given in the datasheet at two or three temperatures, such as the ambient temperature and the maximum operating temperature. The characteristics of other junction temperatures are modeled as a linear interpolation between the top and bottom boundaries, as given in (3.42). Therefore the electrical characteristics interact with thermal behaviors, and the electrothermal converter model can be developed.

$$X^{(T_j)} = \frac{X^{(T_{max})} - X^{(T_{min})}}{T_{max} - T_{min}}\left(T_j - T_{min}\right) + X^{(T_{min})} \tag{3.42}$$

References

[1] I. Etxeberria-Otadui, V. Manzo, S. Bacha, F. Baltes, Generalized average modelling of FACTS for real time simulation in ARENE, in: IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02, vol. 2, 2002, pp. 864–869.

[2] B. Lu, X. Wu, H. Figueroa, A. Monti, A low-cost real-time hardware-in-the-loop testing approach of power electronics controls, IEEE Transactions on Industrial Electronics 54 (2) (2007) 919–931.

[3] N. Lin, V. Dinavahi, Detailed device-level electrothermal modeling of the proactive hybrid HVDC breaker for real-time hardware-in-the-loop simulation of DC grids, IEEE Transactions on Power Electronics 33 (2) (2018) 1118−1134.

[4] H. Bai, H. Luo, C. Liu, D. Paire, F. Gao, A device-level transient modeling approach for the FPGA-based real-time simulation of power converters, IEEE Transactions on Power Electronics 35 (2) (2020) 1282−1292.

[5] A. Myaing, V. Dinavahi, FPGA-based real-time emulation of power electronic systems with detailed representation of device characteristics, IEEE Transactions on Industrial Electronics 58 (1) (2011) 358−368.

[6] N. Lin, V. Dinavahi, Behavioral device-level modeling of modular multilevel converters in real time for variable-speed drive applications, IEEE Journal of Emerging and Selected Topics in Power Electronics 5 (3) (2017) 1177−1191.

[7] N. Lin, V. Dinavahi, Dynamic electro-magnetic-thermal modeling of MMC-based DC−DC converter for real-time simulation of MTDC grid, IEEE Transactions on Power Delivery 33 (3) (2018) 1337−1347.

[8] Z. Huang, V. Dinavahi, A fast and stable method for modeling generalized nonlinearities in power electronic circuit simulation and its real-time implementation, IEEE Transactions on Power Electronics 34 (4) (2019) 3124−3138.

[9] Z. Shen, V. Dinavahi, Real-time device-level transient electrothermal model for modular multilevel converter on FPGA, IEEE Transactions on Power Electronics 31 (9) (2016) 6155−6168.

[10] Z. Shen, T. Duan, V. Dinavahi, Design and im- plementation of real-time MPSoC-FPGA-based electromagnetic transient emulator of CIGRÉ DC grid for HIL application, IEEE Power and Energy Technology Systems Journal 5 (3) (2018) 104−116.

[11] Z. Shen, V. Dinavahi, Real-time MPSoC-based electrothermal transient simulation of fault tolerant MMC topology, IEEE Transactions on Power Delivery 34 (1) (2019) 260−270.

[12] H. Bai, C. Liu, A.K. Rathore, D. Paire, F. Gao, An FPGA-based IGBT behavioral model with high transient resolution for real-time simulation of power electronic circuits, IEEE Transactions on Industrial Electronics 66 (8) (2019) 6581−6591.

[13] H. Bai, C. Liu, S. Zhuo, R. Ma, D. Paire, F. Gao, FPGA-based device-level electro-thermal modeling of floating interleaved boost converter for fuel cell hardware-in-the-loop applications, IEEE Transactions on Industry Applications 55 (5) (2019) 5300−5310.

[14] T. Liang, V. Dinavahi, Real-time system- on-chip emulation of electrothermal models for power electronic devices via Hammerstein configuration, IEEE Journal of Emerging and Selected Topics in Power Electronics 6 (1) (2018) 203−218.

[15] T. Liang, V. Dinavahi, Real-time device-level simulation of MMC-based MVDC traction power system on MPSoC, IEEE Transactions on Transportation Electrification 4 (2) (2018) 626−641.

[16] Thermal Modeling of Power-Electronic Systems Dr. Martin März, Paul Nance Infineon Technologies AG, Munich.

[17] K. Li, Y. He, B. Li, Z. Peng, 'Review and prospect of IGBT power module thermal network model establishment and parameter extraction method, Journal of Electronic Measurement and Instrument 34 (1) (2020) 51−60.

[18] A.S. Bahman, K. Ma, P. Ghimire, et al., A 3-D-lumped thermal network model for long-term load profiles analysis in high-power IGBT modules, IEEE Journal of Emerging and Selected Topics in Power Electronics 4 (3) (2016) 1050−1063.

Modeling of circuit components

4

4.1 Introduction

In Chapter 3 the modeling of power electronic devices is presented in detail. In addition to the switching devices, power electronic systems contain basic electrical circuit components, such as passive components including resistances, inductances, and capacitances; power sources including ideal power sources and controlled power sources; and transformers. With the development of renewable energy generation, power electronic systems are today also commonly fed by batteries and other types of power sources such as photovoltaics, fuel cells, and wind turbine generators. Therefore in this chapter the models of the passive components, power sources, and transformers are presented in detail. These models are essential for the real-time simulation of modern power electronic systems.

4.2 Passive components

4.2.1 Ohm's law

The voltage across and the current through a passive component can be expressed as the functions in (4.1), where di/dt and dv/dt are the derivatives of current I and voltage v, respectively [1].

$$v = f\left(i, \frac{di}{dt}\right) \quad \text{or} \quad i = f\left(v, \frac{dv}{dt}\right) \tag{4.1}$$

If (4.1) does not contain the derivative component, the passive component is to be a resistor with a current and voltage relationship of (4.2).

$$v = f(i) \quad \text{or} \quad i = f(v) \tag{4.2}$$

If the resistor is linear, Ohm's law can be obtained in (4.3), where R is the resistance. Meanwhile the current through a resistor can be expressed by (4.4), where G is the conductance.

$$v = Ri \tag{4.3}$$

Real-Time Simulation Technology for Modern Power Electronics. DOI: https://doi.org/10.1016/B978-0-323-99541-2.00005-3

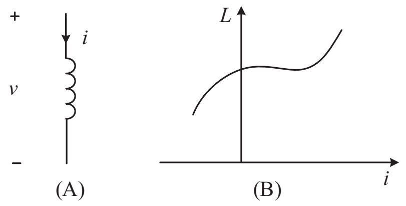


Figure 4.1 (A) Electrical symbol of the inductor; (B) inductance of the nonlinear inductor.

$$i = Gv \tag{4.4}$$

An inductor is a passive component where the magnetic flux is a function of the current through it, namely, $\phi = f(i)$. The induced voltage across the inductor equals to changing rate of the flux, and the inductance is defined by the derivative of flux to current. Therefore for an inductor the voltage and current relationship is expressed by (4.5).

$$v(t) = \frac{d\phi}{dt} = \frac{d\phi}{di}\frac{di}{dt} = L(i)\frac{di}{dt} \tag{4.5}$$

If the inductance is a function of current, the inductor is a nonlinear component as shown in Fig. 4.1. If the inductance is constant, the inductor is said to be linear.

A capacitor is a passive component where the electrical charge is a function of the voltage across it, namely $q = f(v)$. The current is the change rate of charge, and the capacitance is defined by the derivative of charge to voltage. Therefore for a capacitor the current and voltage relationship can be expressed by (4.6).

$$i(t) = \frac{dq}{dt} = \frac{dq}{dv}\frac{dv}{dt} = C(v)\frac{dv}{dt} \tag{4.6}$$

If the capacitance is a function of voltage, the capacitor is a nonlinear component as shwon in Fig. 4.2. If the capacitance is constant, the capacitor is said to be linear.

$$i = C\frac{dv}{dt} \tag{4.7}$$

4.2.2 Equivalent model of the inductor

In the circuit simulation, if backward Euler method is applied to (4.5) to represent the inductor in the discrete time, Eq. (4.8) will be obtained, where h is the discretization time step and n is the number of the time steps.

$$v_{n+1} = \dot{\phi}_{n+1} = \frac{\phi_{n+1} - \phi_n}{h} = \frac{f(i_{n+1})}{h} - \frac{f(i_n)}{h} \tag{4.8}$$

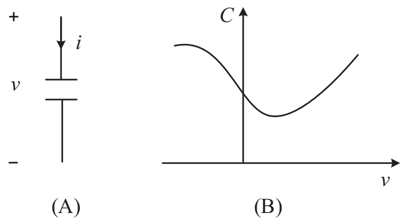


Figure 4.2 (A) Electrical symbol of the capacitor; (B) capacitance of a nonlinear capacitor.

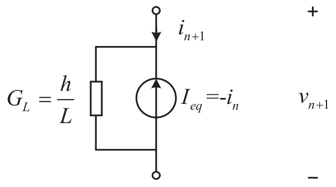


Figure 4.3 Companion circuit model of the linear inductor.

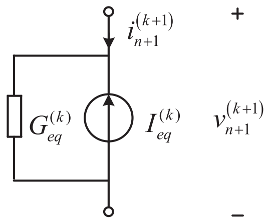


Figure 4.4 Companion circuit model of the nonlinear inductor.

When a linear inductor is considered, $\phi = f(i) = Li$, we can get (4.9), which is equivalent to the companion circuit model shown in Fig. 4.3 in the time step (t_n, t_{n+1}).

$$v_{n+1} = \frac{L}{h} i_{n+1} - \frac{L}{h} i_n \Rightarrow i_{n+1} = \frac{h}{L} v_{n+1} + i_n = G_L v_{n+1} - I_{eq} \tag{4.9}$$

When a nonlinear inductor is considered the current–voltage relationship in discrete time should follow the expression in (4.8). By using the Newton–Raphson (N-R) iteration algorithm, the iterative formula can be constituted as (4.10), which is equivalent to the companion circuit in Fig. 4.4. Where n refers to the number of discretization time steps, and k represents the number of N-R iterations in the current time step (t_n, t_{n+1})

$$v_{n+1}^{(k+1)} = \frac{dv_{n+1}}{di_{n+1}}\bigg|_{i_{n+1}^{(k)}} \left(i_{n+1}^{(k+1)} - i_{n+1}^{(k)} \right) + v_{n+1}^{(k)} \Rightarrow i_{n+1}^{(k+1)} = G_{eq} v_{n+1}^{(k+1)} - G_{eq} v_{n+1}^{(k)} + i_{n+1}^{(k)} \tag{4.10}$$

where

$$G_{eq}^{(k)} = \left[\left. \frac{dv_{n+1}}{di_{n+1}} \right|_{i_{n+1}^{(k)}} \right]^{-1} = \frac{h}{f'\left(i_{n+1}^{(k)}\right)},\ I_{eq}^{(k)} = G_{eq} v_{n+1}^{(k)} - i_{n+1}^{(k)}.$$

4.2.3 Equivalent model of the capacitor

The capacitors in the circuit simulation could also be discretized. If backward Euler method is applied, Eq. (4.6) is expressed by (4.11).

$$i_{n+1} = \dot{q}_{n+1} = \frac{q_{n+1} - q_n}{h} = \frac{f(v_{n+1})}{h} - \frac{f(v_n)}{h}. \tag{4.11}$$

When a linear capacitor is considered, $q = f(v) = Cv$, we can get (4.12) and the capacitor can be modeled by the companion circuit in Fig. 4.5.

$$i_{n+1} = \frac{C}{h} v_{n+1} - \frac{C}{h} v_n = G_C v_{n+1} - I_{eq} \tag{4.12}$$

When a nonlinear capacitor is considered, the current−voltage relationship in discrete time should follow the expression in (4.11). By using the N-R iteration algorithm, the iterative formula can be constituted as (4.13), which is equivalent to the companion circuit in Fig. 4.6. Where n refers to the number of discretization time steps, and k represents the number of N-R iterations in the current time step (t_n, t_{n+1})

$$i_{n+1}^{(k+1)} = \left. \frac{di_{n+1}}{dv_{n+1}} \right|_{v_{n+1}^{(k)}} \left(v_{n+1}^{(k+1)} - v_{n+1}^{(k)} \right) + i_{n+1}^{(k)} = G_{eq}^{(k)} v_{n+1}^{(k+1)} - I_{eq}^{(k)} \tag{4.13}$$

where

$$G_{eq}^{(k)} = \left. \frac{di_{n+1}}{dv_{n+1}} \right|_{v_{n+1}^{(k)}} = \frac{f'\left(v_{n+1}^{(k)}\right)}{h},\ I_{eq}^{(k)} = -\, i_{n+1}^{(k)} + G_{eq}^{(k)} v_{n+1}^{(k)} \tag{4.14}$$

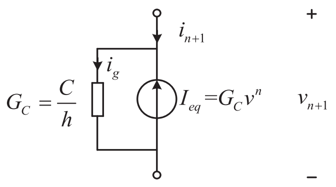


Figure 4.5 Companion circuit model of a linear capacitor.

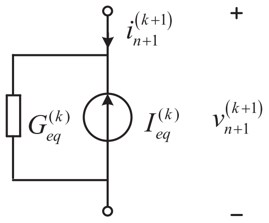


Figure 4.6 Companion circuit model of a nonlinear capacitor.

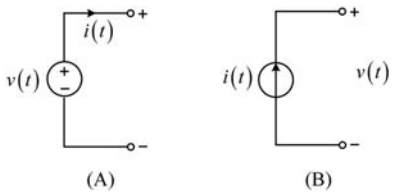


Figure 4.7 (A) Independent voltage source; (B) independent current source.

4.3 Power sources

The power sources provide and sometimes absorb electrical energy in the power electronic circuits. The power sources can be a generator, a battery, and other types of renewable sources, such as PV panels and fuel cell stacks. In the circuit model, the power sources are represented by active elements where equivalent voltage or current is a function of time or a function of the current and voltage of other elements. According to the dependency type, the power sources are classified into independent power sources and controlled power sources.

4.3.1 Independent power source

The independent voltage source can be represented by the symbol shown in Fig. 4.7A. The voltage of an independent voltage source is a constant or a function of time $v(t)$, and $i(t)$ denotes the current flowing through the voltage source branch. The independent current source can be represented by the symbol shown in Fig. 4.7B. The current of an independent current source is a constant or a function of time $i(t)$, and $v(t)$ denotes the voltage across the current source.

4.3.2 Controlled power sources

The controlled voltage source is a voltage source whose value depends on the states of other elements in the circuit. If the voltage is a function of the voltage v_x across any other elements in the circuit, namely, $v = f(v_x)$, the voltage source is a voltage-controlled voltage source (VCVS) and can be represented by the symbol in

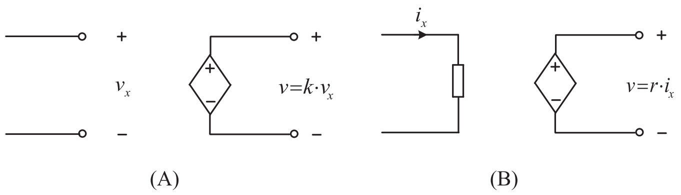


Figure 4.8 (A) Voltage-controlled voltage source; (B) current-controlled voltage source.

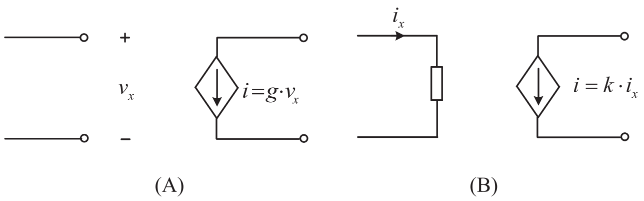


Figure 4.9 (A) Voltage-controlled current source; (B) current-controlled current source.

Fig. 4.8A. If the voltage is a function of the current i_x through any other elements in the circuit, namely, $v=f(i_x)$, the voltage source is a current-controlled voltage source (CCVS) and can be represented by the symbol in Fig. 4.8B.

In analogy, the controlled current source presents a current source in which the value depends on the states of other elements in the circuit. If the current is a function of the voltage v_x across any other elements in the circuit, namely, $i=f(v_x)$, the current source is a voltage-controlled current source (VCCS) and can be represented by the symbol in Fig. 4.9A. If the current is a function of current i_x through any other elements in the circuit, namely, $i=f(i_x)$, the current source is a current-controlled current source (CCCS) and can be represented by the symbol in Fig. 4.9B.

4.3.3 Batteries

The battery cell can be modeled as a controlled voltage source and a resistor connected in series with it. Additionally, multiple parallel resistor–capacitor circuits can be added in series with the internal resistance to increase the fidelity of the model in relation to the physical battery, as can be seen in Fig. 4.10. Finally the model can be extended to support hysteresis effect by adding an additional voltage source. A number of parameters, depending on the desired level of details of the model, can be included in the dynamic calculation of the values of these elements [2]. Battery packs are made by stacking multiple battery cells in parallel, series, or both.

As an energy storage element a battery has a state variable directly proportional to the remaining energy of the battery, called battery state of charge (SOC).

$$SOC(t,T)=\int\frac{i(t)\cdot\eta(T)}{Q(T)}dt \tag{4.15}$$

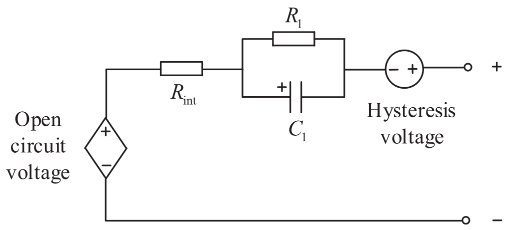


Figure 4.10 First-order equivalent circuit model.

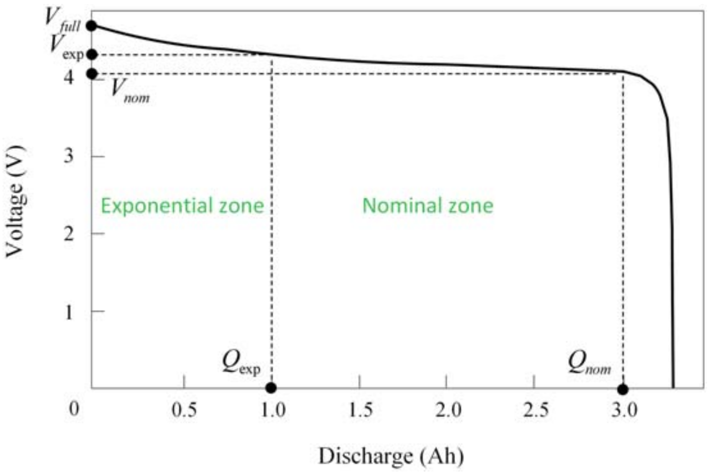


Figure 4.11 Lithium-ion battery cell OCV [V] as a function of the state of discharge [Ah], analytical equation result.

where T is the temperature, $Q(T)$ is the total capacity parameter, $i(t)$ is the battery terminal current, and $\eta(T)$ is the Coulombic efficiency coefficient, which is usually applied only when the cell is being charged (otherwise, its value is set to 100%).

The controlled voltage source mimics the battery pack *open-circuit voltage* (OCV) property, which represents the cell's terminal voltage when the diffusion process has subsided and when there is no current flow. OCV can be represented as an analytical function, or more commonly using a lookup table, which varies significantly with the battery type. In most battery models, OCV is calculated solely as a function of the SOC, while more detailed models usually consider temperature dependency too.

In the case of an analytical function, the following experimental data can be used to extract the model parameters, as illustrated in Fig. 4.11: the fully charged voltage V_{full}, the end of the exponential zone voltage V_{exp} and charge Q_{exp}, and the end of the nominal zone voltage V_{nom} and charge Q_{nom} [3]. OCV can then be described by (4.16),

$$OCV = OCV_0 - K\frac{Q_{max}}{Q_{max} - it} + \left(V_{full} - V_{exp}\right) \cdot e^{\frac{-3 \cdot it}{Q_{exp}}} \tag{4.16}$$

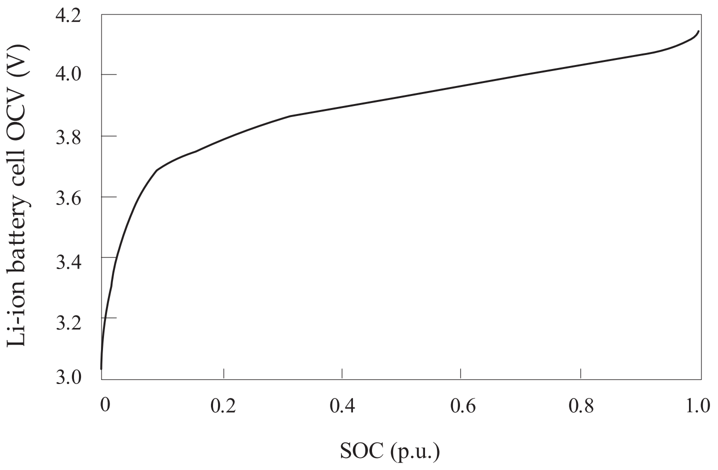


Figure 4.12 Lithium-ion battery cell OCV as a function of the state of charge, experimental results.

where Q_{max} is the maximum battery capacity; *it* is the extracted capacity, $K = \frac{\left(V_{full} - V_{nom} + \left(V_{full} - V_{exp}\right)\cdot\left(e^{\frac{-3\cdot Q_{nom}}{Q_{exp}}} - 1\right)\right)\left(Q_{max} - Q_{nom}\right)}{Q_{nom}}$ $OCV_0 = V_{full} + K + i_{nom} \cdot \frac{V_{full}\cdot 0.005}{0.2\cdot Q_{nom}} - V_{full} + V_{exp}$, and i_{nom} is the nominal discharge current.

These analytical equations can give approximate results for the OCV curve. For lithium-ion battery cells, one such approximation is shown in Fig. 4.11.

The common shape of experimentally derived results of $OCV = f(SOC)$ curve for the lithium-ion battery cell is shown in Fig. 4.12.

The resistor in series with the controlled voltage source represents the battery pack's internal resistance. The model comprising the voltage source in series with the internal resistance is comprehensive enough to represent a battery pack without acknowledging chemical diffusion processes (e.g., diffusion of lithium ions through the membrane in Li-ion battery cells). To simulate these processes, it is common to add up to five parallel resistor—capacitor circuits in series with the basic model. Setting the correct values for these resistors and capacitors, however, is a difficult and iterative process of comparing the simulation to laboratory results and correcting the resistor—capacitor parameters.

Usually, battery manufacturers provide datasheets with experimentally obtained parameters and results:

- One of the most important laboratory results is the *OCV curve* as a function of SOC. These results vary based on battery types, the most common of which are lead-acid, lithium-ion, nickel-cadmium, and nickel-metal-hydride. OCV also has a temperature dependence.
- To model the state of charge, aside from the battery pack terminal input current, parameters Q and η are needed. However, the Coulombic efficiency coefficient can be easily calculated; the total capacity (the total amount of energy a battery pack contains when fully charged) is rarely provided. It is more common to find discharge capacity, which

defines the charge that can be extracted from the fully charged battery, using a particular constant discharge rate before it reaches the minimum/cutoff voltage. Calculations connecting these two capacities can be found in the relevant literature.

- Internal resistance is typically provided and often for multiple temperature points. Internal resistance is one of the parameters greatly influenced by the battery temperature.
- Hysteresis effect voltage is important to replicate the battery cell behavior. When measuring the voltage on the cell terminals, hysteresis behavior of the cell voltage in relation to the cell current can be observed. One of the analytical equations that can be used to describe this behavior, called *One state hysteresis*, is described below:

$$V_{hysteresis}(t,T) = M0(T) \cdot sig(t) + h(t) \tag{4.17}$$

where

$$\dot{h}(t,T) = \left| \frac{\eta(T) \cdot i(t) \cdot \gamma}{Q(T)} \right| \cdot h(t,T) + \left| \frac{\eta(T) \cdot i(t) \cdot \gamma}{Q(T)} \right| \cdot M(T)$$

$M(T)$, $M0(T)$, and $\gamma$are user-customizable parameters, and $sig(t)$ is a sign of the cell current.

Previously mentioned parameters are only a few of many parameters typically provided by the manufacturers. Battery cells and battery pack operating parameters are influenced by multiple environmental factors as well as the number of charging and discharging cycles of the pack. Depending on the application, different levels of model fidelity might be required.

4.3.4 Fuel cell

One of the renewable power sources on the rise is the *proton exchange membrane (PEM) fuel cell.* It can be modeled as a controlled voltage source, whose value is calculated by a mathematical model that mimics real-world behavior of such devices.

The controlled voltage source value can be calculated as

$$V_{cell} = E_{cell} - V_{ohm} - V_{act} \tag{4.18}$$

where E_{cell} is the thermodynamic voltage of the fuel cell, V_{ohm} is the voltage drop on the electrolyte ionic resistance, and V_{act} is the cathode activation loss voltage.

The thermodynamic voltage of the fuel cell can be calculated using the Nernst equation:

$$E_{cell} = 1.229 - 0.85 \cdot 10^{-3}(T_{cell} - 298.15) + \frac{R \cdot T}{2F} \ln\left(\sqrt{P_{O_2 cata}} \cdot P_{H_2 cata}\right) \tag{4.19}$$

where T is the temperature of the cell in Kelvins, $P_{O_2 cata}$ is the oxygen pressure in atm at the cathode catalytic interface, $P_{H_2 cata}$ is the hydrogen pressure in atm at the cathode catalytic interface, $R = 8.314$ is the universal gas constant in J/(mol · K), and $F = 96,485.3$ is the Faraday constant in C/mol.

The differences between the gas supply pressures and the electrode interface pressures are due to the gas diffusion phenomenon through the porous electrodes in a PEM fuel cell, which can be calculated using Fick's law:

$$P_{xcata} = P_x - \frac{N_x \cdot \delta_{GDL} \cdot R \cdot T}{D_{x-\mathrm{H_2O}eff} \cdot A} \ \mathrm{x} \in (\mathrm{H_2}, \mathrm{O_2}) \tag{4.20}$$

where δ_{GDL} is the gas diffusion layer thickness, $D_{x-\mathrm{H_2O}eff}$ is the effective gas diffusion coefficient, and the reactant gas molar flow rates for hydrogen and oxygen are directly related to the fuel cell current:

$$N_{\mathrm{H_2}} = \frac{i}{2 \cdot F} \tag{4.21}$$

$$N_{\mathrm{O_2}} = \frac{i}{4 \cdot F} \tag{4.22}$$

The effective gas diffusion coefficient $D_{x-\mathrm{H_2O}eff}$ can be obtained using the Slattery−Bird formula with the Bruggemann correction:

$$D_{x-\mathrm{H_2O}eff} = D_{x-\mathrm{H_2O}} \cdot \varepsilon^{\tau} \ x \in (\mathrm{H_2}, \mathrm{O_2}) \tag{4.23}$$

where $D_{x-\mathrm{H_2O}}$ is the binary gas diffusion coefficient in m^2/s, between the reactants H_2, O_2 and product H_2O, ε is the corresponding gas diffusion layer porosity, and τ is the corresponding gas diffusion layer tortuosity.

Cathode activation loss voltage can be calculated using the following differential equation:

$$\frac{d}{dt} V_{act} = \frac{i}{C_{dl}} \left(1 - \frac{1}{\eta_{act}} V_{act} \right) \tag{4.24}$$

where C_{dl} is the single-fuel cell cathode double-layer capacitance in Farads, i is the fuel cell current in Amperes, and η_{act} is the cathode steady-state activation voltage drop, calculated using the well-known Tafel equation:

$$\eta_{act} = \frac{R \cdot T}{2 \cdot \alpha \cdot F} \ln \left(\frac{i}{J \cdot A} \right) \tag{4.25}$$

where A is the fuel cell catalytic layer geometric surface area in m^2, α is the electrochemical reaction symmetry factor (usually between 0.2 and 0.5), and J is the cathode exchange current density in A/m^2, calculated by the following empirical equation:

$$J = \gamma \cdot P^{\beta}_{\mathrm{O_2}cata} \cdot e^{-\frac{E_C}{R \cdot T}\left(1 - \frac{T}{298.15}\right)} \tag{4.26}$$

where γ and β are two empirical parameters that need to be identified through fuel cell experimental tests, and E_c is the constant representing oxygen activation energy at the electrode platinum interface (66,000 J/mol).

Electrolyte ionic resistance voltage drop can be calculated from the membrane conductivity equation:

$$V_{ohm} = \frac{i \cdot \delta_{mem}}{\sigma_{mem} \cdot A_{mem}} \tag{4.27}$$

where δ_{mem} is the PEM thickness and σ_{mem} (S/m) is the proton conductivity of Nafion, which is highly related to the water content of the membrane. Its value can be calculated by

$$\sigma_{mem} = 0.5139 \cdot \lambda_{mem} - 0.326 \cdot e^{\left[1268\left(\frac{1}{303}-\frac{1}{T}\right)\right]} \tag{4.28}$$

where λ_{mem} is the membrane water content: $\lambda_{mem} \approx 7$ for a dry membrane, $\lambda_{mem} \approx 14$ for a well-humidified membrane, and $\lambda_{mem} \approx 22$ for a flooded membrane.

4.3.5 Photovoltaic panel

The photovoltaic (PV) panel is a DC power source that converts the absorbed solar energy into electricity. The basic device of a PV panel is the PV cell. A PV panel comprises multiple PV cells connected in series and/or parallel in order to achieve higher output power. The PV cell has a semiconductor structure, commonly silicon. The conversion is based on the photoelectric effect in the PV cell, in which electrons excited by the absorbed solar energy are emitted from the surface of the PV cell, which is in close vicinity of the semiconductor p-n junction.

The PV cell can be modeled with a single-diode equivalent circuit model [4]. An independent current source provides the photocurrent, which models irradiance flux, while the diode current represents current escaping due to diffusion in the semiconductor p-n junction. Series and parallel resistances can be included to model junction and contact losses and diode leakage current, respectively. In the remainder of this section the resistance connected in parallel is not considered as the current through is assumed to be negligible. In more detailed models an additional parallel connected diode can be included to model current loss due to charge recombination. The equivalent circuit diagram of a PV cell, with an independent current source I_{ph}, diode current I_D, and series resistance R_{pv}, is given in Fig. 4.13.

The photocurrent I_{ph} is dependent on both temperature T(K) and solar irradiance $G\left(\mathrm{W/m^2}\right)$ conditions. The relationship between the PV cell output current I and the terminal voltage V is given by the equation

$$I(G,T) = I_{ph}(G,T) - I_D = I_{ph} - I_0\left(e^{\frac{q\left(V+IR_{pv}\right)}{mkT}} - 1\right) \tag{4.29}$$

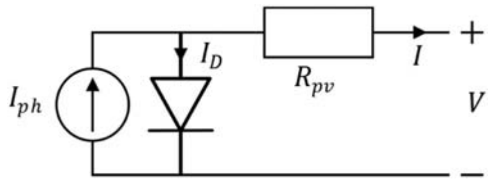


Figure 4.13 Equivalent circuit diagram of a PV cell.

where I_0 is the reverse diode saturation current, q is the electron charge ($q = 1.6021 \times 10^{-19}$C), k is the Boltzmann constant ($k = 1.3865 \times 10^{-23}$J/K), and m is the diode ideality factor ($1 \leq m \leq 2$). The diode ideality factor models how closely the diode follows the ideal diode equation.

Short-circuit current $I_{SC(STC)}$ and open-circuit voltage $V_{OC(STC)}$ at standard test condition (STC) operating points can be calculated given the short-circuit current $I_{p\_SC(STC)}$ and open-circuit voltage $V_{p\_OC(STC)}$ of the PV panel, which is usually provided in the current–voltage (I-V) curve. For a panel comprising n_p parallel connected cells and n_s series connected cells, their relationship is expressed with $I_{SC(STC)} = I_{p_SC(STC)}/n_p$and $V_{OC(STC)} = V_{p\_OC(STC)}/n_s$. At operating condition $V = 0$,

$$I_{ph}(G,T) = I(G,T) = I_{SC}(G,T) \tag{4.30}$$

At climatic conditions determined by an irradiance G and temperature T, the short-circuit current $I_{SC}(G,T)$ can be used to specify the value of the photocurrent $I_{ph}(G,T)$. In particular,

$$I_{ph(STC)} = I_{SC(STC)} \tag{4.31}$$

The series-connected PV resistance R_{pv} can be derived by exploiting the gradient of the terminal voltage at OCV. The gradient can be approximated from the inverse of the slope of the product I-V curve at OCV.

$$\left.\frac{dV}{dI}\right|_{I=0} \tag{4.32}$$

Terminal voltage V is expressed as a function of the output current in (4.33), while the derivative of the V-I characteristic is given in (4.34).

$$V = \frac{mkT}{q}\ln\frac{I_{ph} - I + I_0}{I_0} - IR_{pv} \tag{4.33}$$

$$\frac{dV}{dI} = -\frac{mkT}{q}\frac{1}{I_{ph} - I + I_0} - R_{pv} \tag{4.34}$$

The terms in the output current Eq. (4.29) can be rearranged as

$$I_{ph} - I + I_0 = I_0 e^{\frac{q(V+IR_{pv})}{mkT}} \tag{4.35}$$

At operating condition $I = 0, T = T_c, V = V_{OC(STC)}$, where T_c is a reference temperature at standard operating conditions, the following equation holds:

$$I_{ph(STC)} + I_0 = I_0 e^{\frac{qV_{OC(STC)}}{mkT_c}} = \frac{I_{ph(SCT)}}{e^{\frac{qV_{OC(STC)}}{mkT_C}} - 1} e^{\frac{qV_{OC(STC)}}{mkT_c}} \tag{4.36}$$

$$I_{0(STC)} = \frac{I_{ph(STC)}}{e^{\frac{qV_{OC(STC)}}{mkT_C}} - 1} \tag{4.37}$$

Finally the expression for the series-connected resistance is given by

$$R_{pv} = -\frac{mkT}{q}\frac{1}{I_{ph} - I + I_0}\bigg|_{I=0} - \frac{dV}{dI}\bigg|_{I=0} = -\frac{mkT_C}{q}\frac{1}{I_{ph}}\frac{e^{\frac{qV_{OC(STC)}}{mkT_c}} - 1}{e^{\frac{qV_{OC(STC)}}{mkT_c}}} - \frac{dV}{dI}\bigg|_{I=0} \tag{4.38}$$

Photocurrent can be expressed using STC temperature $T_{(STC)}$ and irradiance $G_{(STC)}$, which are typically $G_{(STC)} = 1000\ W/m^2$ and $T_{(STC)} = 25^{\circ}$C.

$$I_{ph}(G,T) = I_{ph(STC)}\frac{G}{G_{(STC)}} + K_t\left(T - T_{(STC)}\right) \tag{4.39}$$

where $K_t\left(\mathrm{A}/^{\circ}\mathrm{C}\right)$ is the photocurrent temperature coefficient.

The reverse diode saturation current I_o is temperature-dependent as well, given by

$$I_o(T) = I_{o(STC)}\left(\frac{T}{T_{(STC)}}\right)^{3/m} e^{\frac{qV_{g(STC)}}{mk}\left(\frac{1}{T} - \frac{1}{T_{(STC)}}\right)} \tag{4.40}$$

where $V_{g(STC)}$ is the silicon gap energy of the semiconductor material at STC. For crystalline silicon (c-Si) technology the value of V_g is usually $V_g = 1.12$ V, while for thin-film (TF, amorphous) silicon technology, it is $V_g = 1.75$ V.

Finally the I-V curve can be generated. For a given temperature and irradiance the photocurrent and the reverse diode saturation current are calculated and included in the expression of the output current. The PV panel is modeled as a compound parameterized PV cell, whose output current is obtained by aggregating output currents of individual PV cells. Fig. 4.14 shows the I-V and power–voltage (P-V) curves of a PV panel.

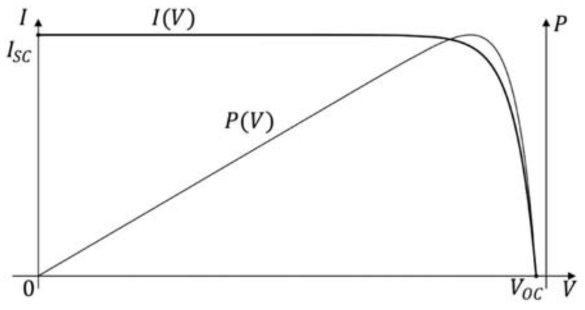


Figure 4.14 I-V and P-V curves of a PV panel.

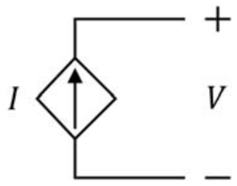


Figure 4.15 PV panel as voltage-controlled current source.

A comparison of OCV and short-circuit current for varied climatic conditions and constant circuit parameters is considered. At constant solar irradiance, exposure to higher temperatures decreases the OCV of the PV panel, while short-circuiting current increases slightly. The short-circuit current is directly proportional to solar irradiance, while the OCV increases less considerably.

The PV panel is typically modeled as a current source controlled by its terminal voltage as shown in Fig. 4.15, in combination with a predefined PV model I-V curve. The nonlinear analytical I-V curve is approximated with a lookup table and is derived from a mathematical model of the PV cell, as described in this section.

4.3.6 Wind power plants

Wind power plants are one of the leading renewable energy sources. Early wind power plants were simple in their construction and operation. However, with the introduction of power electronic converters, their complexity has increased, leading to the need for using more advanced modeling and simulation techniques.

A wind power plant consists of several parts required for effective energy conversion, starting from wind energy harvesting to supplying electrical energy to the grid. Stages typically used in this process are presented in Fig. 4.16.

There are several types of wind power plants, and each type reflects the properties of elements presented in Fig. 4.16. Different types of wind power plants are labeled with a number from 1 to 4 or from A to D, which will be explained throughout this chapter.

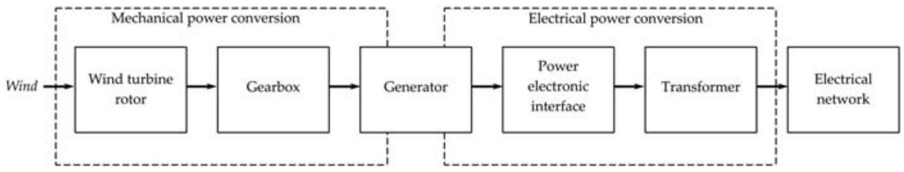

Mechanical power conversion
Electrical power conversion
Wind
Wind turbine rotor
Gearbox
Generator
Power electronic interface
Transformer
Electrical network

Figure 4.16 Stages of power conversion in a wind power plant.

Wind turbines harvest the power from the wind through rotating blades, which is then converted to rotating mechanical power at the shaft. The wind power collected by the turbine can be expressed as

$$P_t = \frac{1}{2}\pi R^2 \rho V_w^3 C_p \tag{4.41}$$

where R is the radius of the turbine (length of the blades), ρ is the air density, V_w is the velocity of the wind, and C_p is the power coefficient.

The power coefficient is a dimensionless parameter that expresses the effectiveness of the wind turbine in the transformation of the kinetic energy of the wind into mechanical energy. Intuitively, if this coefficient is 1, there would be no wind after the turbine. As this is not feasible, there is a theoretical limit to the value of the power coefficient. This maximum is called the Betz limit, and its value is 0.593.

The value of the power coefficient depends on the pitch angle of the blades β, and the tip speed ratio λ. This ratio is defined as

$$\lambda = \frac{R\Omega_t}{V_w} \tag{4.42}$$

where Ω_t is the angular speed of the rotor.

Fig. 4.17 shows an example of the power coefficient function of different tip speed ratios for several blade pitch angles. These curves are mainly dependent on the physical characteristics of wind turbine blades.

For expressing mechanical torque, the expression for power in the turbine can be used:

$$T_m = \frac{P_t}{\Omega_t} = \frac{1}{2}\frac{\rho\pi R^2 V_w^3}{\Omega_t}C_p = \frac{\rho\pi R^2 V_w^2}{2\lambda}C_p \tag{4.43}$$

Most often, wind turbines include a gearbox in order to increase the speed of the generator rotor shaft. An exception is when synchronous machines with a large number of pole pairs are used as generators, and in that case, the gearbox could be excluded.

To express speed and mechanical torque on the rotor shaft the gear ratio must be included in the calculation.

$$T_{turbine} = \frac{T_{rotor}}{N} \tag{4.44}$$

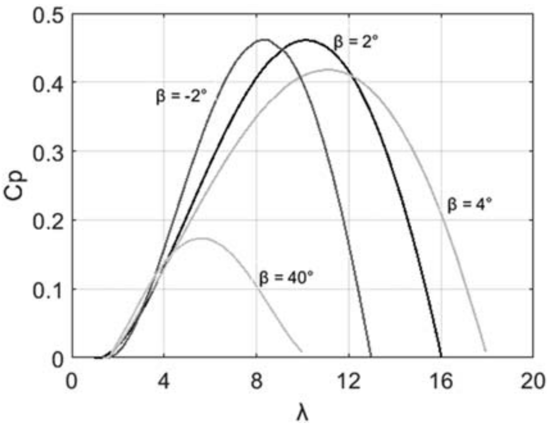


Figure 4.17 Power coefficient as a function of tip speed ratio for several different blade pitch angles.

$$\Omega_{turbine} = \Omega_{rotor}N \tag{4.45}$$

where N is the gearbox ratio.

Apart from the gearbox, the rest of the wind turbine mechanical system is usually modeled as a two-mass system, where one mass represents the turbine, while the second one represents the electrical machine. This model is primarily used for determining the equivalent inertia of the system. However, advanced models can also include friction of the machine and turbine as well as flexible couplings between the two masses.

Multiple electrical machine types can be used as electrical generators. If the wind power plant is of Type 1, mostly often a squirrel cage induction generator is used. Type 2 and 3 plants typically use induction generators too; however, it is an induction machine with available rotor windings via slip rings. This machine is often referred to as a doubly fed induction machine (DFIG). Type 4 can employ various types of machines, most often synchronous machines with or without permanent magnets, but induction machines can be used as well.

The interface between the electric generator and the grid is usually achieved through a power converter. The power converter usually comprises a power electronic device and a transformer. The type of the power electronic device used depends on the type of wind power plant, while the transformer is present in most applications.

In the case of Type 1 plants the electric machine is directly connected to the grid with an optional soft starter for smoother transients. Type 2 plants share the same grid connection concept; however, slip rings of DFIG are connected to variable resistors, which enables partial adjustment of the rotational speed of the wind turbine.

Both Type 3 and Type 4 employ three-phase inverters with two or more levels. They are connected in back-to-back topology and convey part or complete energy produced by the generator. In the case of Type 3, a back-to-back converter connects the rotor windings of DFIG with the grid, while the stator is connected directly to the grid. In Type 4, the back-to-back converter connects the stator with the grid, and the complete power produced by the wind turbine is conveyed through the converter. If the converter is capable of a direct grid connection, then this type can be implemented without the transformer.

Depending on the application requirements, the complexity of the wind power plant model can vary. Usually, when representing a wind power plant inside a larger distribution model the model can be rather simple and is typically based on current or voltage sources, as explained in Section 4.3.2. If more comprehensive insight into wind power plant operation is needed, more detailed models are required. Such models usually comprise a detailed model of the electric generator and the power electronic switching converter.

4.4 Transformers

A transformer is a device that changes the voltage level of the AC power through the magnetic field and is widely used in power electronic converters. Fig. 4.18 gives the structural diagram of a two-winding transformer where two windings are wrapped around a magnetic core. The winding connected to the input voltage is the primary winding, and that connected to the output is the secondary winding. The transformer has N_p turns on its primary winding and N_s turns on the secondary winding.

4.4.1 Ideal transformer

The simplest way to model a transformer is to consider it as an ideal one, where infinite permeability, zero excitation current, zero winding resistance, and zero leakage flux are assumed and the core losses and saturation are neglected. The equivalent circuit of an ideal transformer is shown in Fig. 4.19. The relationship between the input voltage and output voltage of the ideal transformer is given by (4.46), where a is the turn ratio of the transformer.

$$\frac{v_p(t)}{v_s(t)} = \frac{N_p}{N_S} = a \tag{4.46}$$

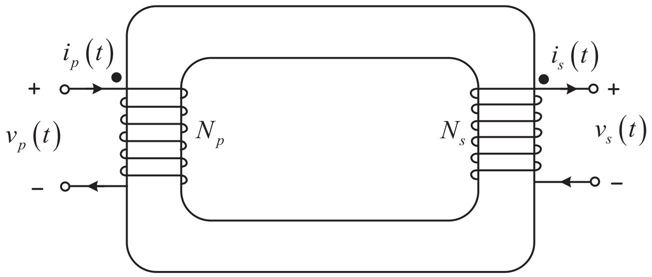


Figure 4.18 A two-winding transformer.

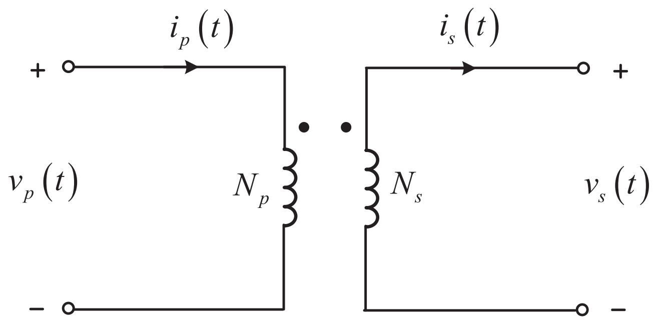


Figure 4.19 The equivalent circuit of an ideal transformer.

The relationship between the input current i_p and output current i_s is given by (4.47).

$$N_p i_p(t) = N_s i_s(t) \rightarrow \frac{i_p(t)}{i_p(t)} = \frac{1}{a} \tag{4.47}$$

The current and voltage magnitude relationship between the primary winding and secondary winding are expressed by (4.46) and (4.47). However, the polarity of the voltage and current in the secondary winding is not yet provided. The dot convention is used to specify the polarity. If the primary voltage is positive at the dotted end, the secondary voltage will be positive at the dotted end, and vice versa. If the current flows into the dotted end in the primary winding, the current will flow out of the dotted winding in the secondary winding, and vice versa.

4.4.2 Equivalent model of a real transformer

In practical applications, the ideal transformer does not exist. The ferromagnetic core and windings are not ideal, and there are many parasitic parameters that need to be considered to constitute a real transformer model.

According to Faraday's law, the applied voltage across the primary winding will produce the magnetic flux in the primary coil. In a real transformer, not all the flux produced on the primary side will reach the secondary one and participate in the energy conversion. Some of the flux will leave the iron and pass through the air. Therefore the leakage flux is defined as the flux that only goes through one coil of the transformer, and the mutual flux is defined as the flux that remains in the core and couples both coils. Similarly the flux in the secondary windings can also be divided into the mutual flux and the leakage flux.

$$\phi_p = \phi_{lp} + \phi_m \tag{4.48}$$

$$\phi_s = \phi_{ls} + \phi_m \tag{4.49}$$

The voltages in the primary and secondary windings can be thus expressed by (4.50) and (4.51).

$$v_p(t) = N_p \frac{d\phi_p}{dt} = N_p \frac{d\phi_m}{dt} + N_p \frac{d\phi_{lp}}{dt} = e_p(t) + e_{lp}(t) \tag{4.50}$$

$$v_s(t) = N_s \frac{d\phi_s}{dt} = N_s \frac{d\phi_m}{dt} + N_s \frac{d\phi_{ls}}{dt} = e_s(t) + e_{ls}(t) \tag{4.51}$$

From (4.50) and (4.51) the voltages related to the mutual flux on both sides of the transformer have the relationship in (4.52), which means that the ratio of voltages in the primary and secondary windings related to the mutual flux is the turn ratio. Therefore the ideal transformer model can be used in the real transformer model to represent such a voltage relationship.

$$\frac{d\phi_m}{dt} = \frac{e_p(t)}{N_p} = \frac{e_s(t)}{N_s} \Rightarrow \frac{e_p(t)}{e_s(t)} = \frac{N_p}{N_s} = a \tag{4.52}$$

Considering that most of the leakage flux passes through the air, the leakage flux can be represented by (4.53), where F is the magnetomotive force and R is the reluctance of air.

$$\begin{aligned} \phi_{lp} &= \frac{F_p}{R_{air}} = \frac{N_p i_p}{R_{air}} \\ \phi_{ls} &= \frac{F_s}{R_{air}} = \frac{N_s i_s}{R_{air}} \end{aligned} \tag{4.53}$$

Combining (4.52) and (4.53), Eq. (4.54) can be obtained, where the L_{lp} and L_{sp} are the self-inductances of primary and secondary windings. Therefore the leakage flux can be modeled by an inductor in series with the primary winding and secondary winding.

$$\begin{aligned} e_{lp}(t) &= N_p \frac{d\phi_{lp}}{dt} = \frac{N_p^2}{R_{air}} \frac{di_p}{dt} = L_{lp} \frac{di_p}{dt} \\ e_{ls}(t) &= N_s \frac{d\phi_{ls}}{dt} = \frac{N_s^2}{R_{air}} \frac{di_s}{dt} = L_{sp} \frac{di_s}{dt} \end{aligned} \tag{4.54}$$

In addition, the AC voltage is applied to the transformer, and there will be a current flow in the primary windings even when no load is connected to the secondary winding. This current is called the excitation current, and it is composed of the magnetization current I_m and the core-loss current I_{h+e}. The magnetization current I_m produces the flux in the ferromagnetic core, and the core-loss current I_{h+e} compensates for the hysteresis and eddy current. The exact equivalent model should

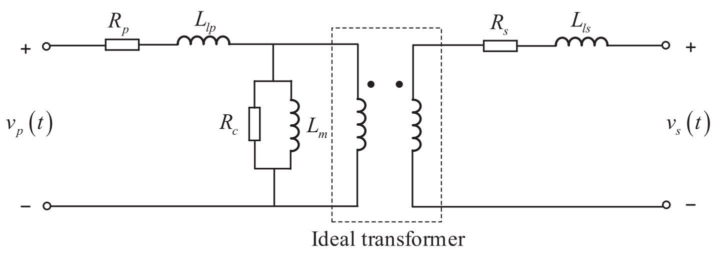


Figure 4.20 Equivalent circuit of a real transformer.

also represent the excitation effects. Considering Faraday's law, we can get the flux in the core [5].

$$\phi = \frac{1}{N_p}\int v_p(t)dt \tag{4.55}$$

If the applied voltage $v_p(t) = V_m \cos\omega t$, the resulting flux will be equal to (4.56), which is 90 degrees lagging the voltage. Considering the magnetization curve of the transformer core, the magnetizing current is in phase with the flux and has a linear relationship in the unsaturation region. The magnetizing current thus has a phase lag of 90 degrees to the applied voltage in the primary winding. Therefore the magnetizing current I_m can be modeled by an inductance across the primary winding.

$$\phi(t) = \frac{V_m}{\omega N_p}\sin\omega t \tag{4.56}$$

Different from the magnetizing current, the core-loss current I_{h+e} is in phase with the applied voltage and can be assumed to be proportional to the voltage. The core-loss current is thus modeled by resistance across the primary winding.

The copper losses refer to the heating losses in the windings and can be modeled by a resistor in series with the windings. By modeling the nonideal effects, including the leakage flux, the copper losses, the excitation effects, and the core losses, the equivalent circuit of a real transformer can be developed as Fig. 4.20. R_p and R_s model the copper losses, L_{lp} and L_{sp} model the leakage flux, R_c models the core losses, L_m models the magnetizing effects, and the ideal transformer models the voltage conversion.

References

[1] F.N. Najm, Circuit Simulation, John Wiley & Sons, Inc., New Jersey, 2010 (Chapter 2).
[2] X. Hu, S. Li, H. Peng, A comparative study of equivalent circuit models for Li-ion batteries, Journal of Power Sources 198 (2012).

[3] O. Tremblay, L.A. Dessaint, A.I. Dekkiche, Models A Generic Battery Model for the Dynamic Simulation of Hybrid Electric Vehicles, IEEE, 2007.

[4] A.D. Hansen, P. Sørensen, L.H. Hansen, H. Bindner, Risø-R no. 1219(EN) Models for a Stand-Alone PV System, Forskningscenter Risø, Denmark, 2001.

[5] S.J. Chapman, Electric Machinery Fundamentals, Tsinghua Press, Beijing, 2008 (Chapter 2).

This page intentionally left blank

Modeling of power electronic converters

5

5.1 Introduction

In Chapters 3 and 4, we noticed that the components in power electronic converters, such as the switches, the sources, and the transformers, can be finally modeled by combining resistors, inductors, capacitors, power sources, and ideal transformers. Therefore finding how to model such a network constituted by those circuit elements will lead to the solution of modeling power electronic converters.

This chapter introduces the theorems, principles, and approaches used in modeling power electronic converters. Kirchhoff's laws are introduced first, followed by the numerical discretization methods of continuous states in the circuit. Two mainstream modeling approaches used in the power electronic real-time simulation are presented in detail, that is, the nodal analysis method and the state-space method. Moreover the system-level converter model and device-level converter model are presented, and the electrothermal converter model is derived. Illustrative examples of converter models are provided to demonstrate the modeling procedure step by step.

5.2 Kirchhoff's laws

Kirchhoff's laws are the basic laws used to analyze the electric network. To demonstrate simply these laws, we will define each two-port component as a *branch* and the connection of branches as a *node*. A closed path composed of branches is called a *loop*. If we consider the boost converter model shown in Fig. 5.1 where the binary resistor switch model is used as an example, six branches and four nodes are involved, and the combinations of branches (1,2,3), (3,4,5), (5,6), (1,2,4,5), (1,2,4,6), and (3,4,6) constitute loops.

When we model the circuit, the branches' currents and branches' voltages are usually selected as the variables that we intend to solve. In a circuit, the branch currents and branch voltages are constrained by two factors. One is the voltage−current relationship (VCR) of the component, which means that the voltage and current of the component should meet a relationship of $v = f(i)$ or $i = f(v)$, as presented in Chapters 3 and 4; the other is the topology that defines how the components are connected, and the topology constraints should be modeled by the Kirchhoff's laws. Kirchhoff's laws include the current law and the voltage law.

Real-Time Simulation Technology for Modern Power Electronics. DOI: https://doi.org/10.1016/B978-0-323-99541-2.00007-7

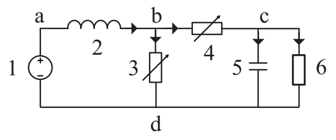


Figure 5.1 Boost converter modeled by the binary resistor switch model.

5.2.1 Kirchhoff's current law

Different branches are coupled in a node, and Kirchhoff's current law (KCL) represents the relationship of the branch currents flowing into or out of that node. KCL can be expressed as *at any time, for any node, the algebraic sum of all the branch currents that flow out of the node is equal to zero*. For any node the branch currents connected to the node will meet the equation

$$\sum i_x = 0. \tag{5.1}$$

The algebraic sum of the current is determined by their reference directions, which in Fig. 5.1 are represented using arrows. If the reference direction of the current is flowing into the node, the current flowing into the node will be positive and that flowing out of the node will be negative in (5.1); if the reference direction of the current is flowing out of the node, the current flowing into the node will be negative and that flowing out of the node will be positive. Considering the boost converter and applying KCL in (5.1) for node b leads to Eq. (5.2). Alternatively, this equation can be represented as (5.3) where the branch currents flowing into node b are equal to those flowing out of node b.

$$-i_2 + i_3 + i_4 = 0 \tag{5.2}$$

$$i_2 = i_3 + i_4 \tag{5.3}$$

For node c, KCL is given by

$$-i_4 + i_5 + i_6 = 0 \tag{5.4}$$

KCL can be extended to a closed surface that is composed of several nodes. If we define a surface containing nodes b and c in the boost converter, as shown in Fig. 5.2, we can get (5.5) by adding (5.2) and (5.4). Eq. (5.5) expresses that *the algebraic sum of all the branch currents that flow out of the surface is equal to zero*. Equivalently the branch currents flowing into the surface and those flowing out of the surface are equal.

$$-i_2 + i_3 + i_5 + i_6 = 0 \tag{5.5}$$

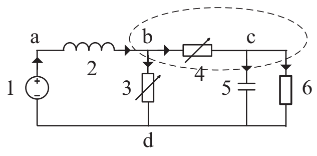


Figure 5.2 Define a closed surface in the boost converter.

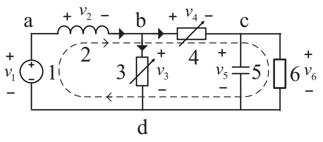


Figure 5.3 Applying KVL to the boost converter.

5.2.2 *Kirchhoff's voltage law*

The relationship between branches' voltages represented by Kirchhoff's voltage law (KVL) is that *the algebraic sum of all the branch voltages in a loop is equal to zero*. For any loop in the circuit the branch voltages will meet (5.6).

$$\sum v_x = 0 \tag{5.6}$$

The algebraic sum of the voltages is determined by the loop direction. Once a loop direction is chosen, if the reference polarity of a branch voltage is the same as the loop direction, the branch voltage will be positive in (5.6), and if the reference polarity of branch voltage is opposite to the loop direction, the branch voltage will be negative in (5.6).

Taking the boost converter again as an example where the reference polarities of branch voltages are defined in Fig. 5.3. The KVL equations of loop (1,2,4,6) will be given by (5.7) if a clockwise loop direction is chosen. Consequently, Eq. (5.8) can be obtained, which means that the voltage rise equals the voltage drop along the loop.

$$v_1 - v_2 - v_4 - v_6 = 0 \tag{5.7}$$

$$v_1 = v_2 + v_4 + v_6 \tag{5.8}$$

To summerize, KCL applies the constraints to the branch currents, and KVL applies the constraints to the branch voltages no matter whether linear or nonlinear, time-invariant, or time-variant components are targeted. Based on VCR, KCL, and KVL, the circuit can be modeled by using the branch current and voltage as state variables.

5.3 Discretization of differential equations

The inductor and capacitor are essential components in power electronic converters because they not only serve as the components in the converter circuit, but also are parts of the equivalent model of other components. Referring to Chapter 4 the VCR of a linear inductor and linear capacitor is given in (5.9) and (5.10).

$$v_L = L\frac{di_L}{dt} \tag{5.9}$$

$$i_c = C\frac{dv_c}{dt} \tag{5.10}$$

The VCR of the inductor and capacitor contains the derivatives, and the equations to model the network containing them are the ordinary differential equations (ODEs). In the simulation, the differential equations must be solved to get the results of the variables of interest with respect to time. However, most of the ODEs cannot be solved analytically. Their results are usually approximated by numerical methods that provide solutions with enough accuracy in a finite time interval, which is accomplished by the discretization of differential equations.

5.3.1 Explicit and implicit methods

The idea of the numerical methods for solving ODEs is that from the initial state of the equations (initial values at $t=0$), one can discretize the "derivative" into the "difference" approximation with a given time interval (called time step) between two points of calculation. The results obtained by numerical methods are a series of discrete values with respect to time [1].

In the remainder of this section, we will use the "standard expression" of the first-order ordinary differential equation, shown in (5.11), to discuss different numerical methods. Explanations and examples in this section are based on a single ODE. However, the developed results and conclusions can obviously be applied to systems of equations.

$$\frac{dy(t)}{dt} = f(t, y(t)) \tag{5.11}$$

The simplest discretization method is to replace the derivative in (5.11) by the following formula:

$$\frac{dy(t)}{dt} = \frac{y(t_n + h) - y(t_n)}{h} \tag{5.12}$$

where t_n is the time of the n^{th} calculation point, $t_{n+1} = t_n + h$ is the time of the next calculation point, and h is the discrete-time interval (time step). It should be noted that when h is sufficiently small, we obtain the definition of the derivative.

Thus, for a given ODE, its numerical solution can be approximated by (5.13) with $n = 0, 1, 2, \ldots$ and $y(0) = y_0$ as the initial value.

$$y_{n+1} = y_n + h \cdot f(t_n, y_n) \tag{5.13}$$

The y_{n+1} and y_n here represent the approximated values of the exact solution $y(t_{n+1})$ and $y(t_n)$.

This discretization method is called the *explicit (forward) Euler method.*

If we consider the definition of the integral,

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))\,dt \tag{5.14}$$

By considering the right-side rectangle rule the above integral form of the ODE can be then approximated by

$$y_{n+1} = y_n + h \cdot f(t_{n+1}, y_{n+1}) \tag{5.15}$$

This discretization method is called the *implicit (backward) Euler method.*

It can be noted that, compared to the explicit Euler method, the term y_{n+1} also appears on the right side of the formula. The formula represents an implicit algebraic equation that needs to be solved in each time step.

Furthermore, in the latter integral form, if we replace the rectangle rule with the trapezoidal rule for the approximation, we can obtain

$$y_{n+1} = y_n + \frac{h}{2} \cdot [f(t_n, y_n) + f(t_{n+1}, y_{n+1})] \tag{5.16}$$

This discretization method is called the *trapezoid method.*

This method also uses an implicit formula (appearance of y_{n+1} at both sides) that needs to be solved in each time step.

For any implicit methods, implicit algebraic equations must be solved in each time step. The exact analytical solution of the implicit formula such as (5.15) and (5.16) can be calculated. However, even with a little more complex implicit formula, it would be impossible to obtain directly its explicit form. In practice, the iterative approximation (estimation) methods have always been used to solve the implicit formula.

For implicit trapezoid method, the functional iteration approximation in each time step can be expressed as

$$y_{n+1}^{(k+1)} = y_n + \frac{h}{2} \cdot \left[f(t_n, y_n) + f\left(t_{n+1}, y_{n+1}^{(k)}\right) \right] \tag{5.17}$$

with $y_{n+1}^{(0)} = y_n + h \cdot f(t_n, y_n)$ as the initial estimation.

If we fix the number of iterations to 1 instead of using an error threshold ε, the above function iteration becomes (5.18) and (5.19), which is also called *improved Euler method.*

$$y_{n+1} = y_n + \frac{h}{2} \cdot [f(t_n, y_n) + f(t_{n+1}, y_n + h \cdot f(t_n, y_n))] \tag{5.18}$$

or

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f(t_n + h, y_n + h \cdot k_1) \\ y_{n+1} = y_n + \frac{h}{2} \cdot (k_1 + k_2) \end{cases} \tag{5.19}$$

Explicit and implicit methods are both used in the numerical solution of ODEs. Explicit methods calculate the state of the system at a later time only from the state of the system at the current time, while implicit methods calculate the state at a later time from both the current time and later one.

Mathematically, if the term y_{n+1} appears on both sides of the initial numerical formula of a method, it is then considered an implicit method; otherwise, the method is explicit.

Explicit formula:

$$y_{n+1} = D(y_n, h)$$

Implicit formula:

$$y_{n+1} = D(y_n, y_{n+1}, h)$$

5.3.2 Taylor's expansion and Runge–Kutta methods

As discussed previously, the numerical methods for solving ODEs are about to find successively the ODE discrete solution y at a later time $(t_n + h)$ next to the current time (t_n). Thus Taylor's expansion can be considered:

$$y(t_{n+1}) = y(t_n + h) = y(t_n) + h \cdot \frac{dy(t_n)}{dt} + \frac{1}{2}h^2 \frac{d^2y(t_n)}{dt^2} + \ldots \frac{1}{m!}h^p \frac{d^{(p)}y(t_n)}{dt^p} + O\left(h^{p+1}\right) \tag{5.20}$$

which is expressed as an infinite Taylor polynomial of h with derivatives of y. $O\left(h^{p+1}\right)$ is the truncation error and is proportional to the h^{p+1}.

If we stop at the order $O\left(h^2\right)$, we will find again the explicit Euler method in (5.13). To solve ODEs with higher accuracy, we can extend Taylor's expansion to some higher orders $O\left(h^{p+1}\right)$. The most famous numerical methods deduced from this approach are the family of "Runge–Kutta methods". Here are some important numerical methods from the Runge–Kutta (R-K) family for solving ODEs.

Heun's method:

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f\left(t_n + \frac{2}{3}h, y_n + \frac{2}{3}h \cdot k_1\right) \\ y_{n+1} = y_n + h \cdot \left(\frac{1}{4}k_1 + \frac{3}{4}k_2\right) \end{cases} \tag{5.21}$$

In the R-K family the intermediate computation steps (denoted by $k_{(v)}$) before the final formula of y_{n+1} are called the "stages." For example, Heun's method is a two-stage R-K method.

Bogacki–Shampine method (three-stage R-K):

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot k_1\right) \\ k_3 = f\left(t_n + \frac{3}{4}h, y_n + \frac{3}{4}h \cdot k_2\right) \\ y_{n+1} = y_n + h \cdot \left(\frac{2}{9}k_1 + \frac{1}{3}k_2 + \frac{4}{9}k_3\right) \end{cases} \tag{5.22}$$

Classic RK4 Runge–Kutta method (four-stage R-K):

$$\begin{cases} k_1 = f(t_n, y_n) \\ k_2 = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot k_1\right) \\ k_3 = f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}h \cdot k_2\right) \\ k_4 = f(t_n + h, y_n + h \cdot k_3) \\ y_{n+1} = y_n + \frac{1}{6}h \cdot (k_1 + 2k_2 + 2k_3 + k_4) \end{cases} \tag{5.23}$$

Other explicit families of R-K methods will not be presented in the reminder of this section.

By using the numerical methods, the solution of $y(t)$ at different time points is calculated successively from one time point (t_n) to another time point ($t_n + h$). The value of h is then called the major time step of numerical methods.

In the simplest forward Euler method the value of $y(t_n + h)$ is calculated directly from the value of $y(t_n)$. However, in order to improve numerical accuracy, in some numerical methods the calculation for the value of $y(t_n + h)$ is based on not only the value of the previous major time step $y(t_n)$ but also the value of some intermediate time steps between (t_n) and $(t_n + h)$. For example, in "Heun's method," we can find the intermediate time step values of $\frac{1}{4}h, \frac{2}{3}h, \frac{3}{4}h, \ldots$; in the "Classic RK4 Runge–Kutta method," we can find the intermediate time step values of $\frac{1}{6}h, \frac{1}{3}h, \frac{1}{2}h, \ldots$ and so on.

These intermediate time step values, presented as a fraction of the major time step h, are called the minor time step and are not "visible" in the final results.

5.3.3 Adams–Bashforth method

In the previous section, we have seen the different numerical methods for ODEs which are developed from the approximation of Taylor's expansion. For those methods (R-K family) the calculation of the state y_{n+1} at the immediate next time step depends only on the state y_n at the current time step.

There is another group of numerical methods for ODEs, which uses not only the value of the immediate neighbor state y_n but also the values of previous states $y_{n-1}, y_{n-2}, \ldots, y_n$, to calculate y_{n+1}. This is the idea of linear multistep discretization methods.

Reconsidering the solution of the ODEs by the definition of the integral,

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t))\,dt \tag{5.24}$$

To solve this formula at the time (t_{n+1}), we need to know the value f_{n+1} of the derivative function $f(t, y(t))$ at the time (t_{n+1}). This value can be approximated by a polynomial interpolation function from its values at previous time points (thus already known) $(t_n, f_n), (t_{n-1}, f_{n-1}), \ldots$. The methods using this approach are known as the family of Adams methods. It should be noted that methods in this family are developed from the interpolation of the derivative function f_n at different times t_n, not from the state y_n.

In order to express explicitly the corresponding polynomial interpolation in the function of $(t_n, f_n), (t_{n-1}, f_{n-1}), \ldots$ the Lagrangian interpolation form is used.

The following are some common numerical methods from the Adams family for solving ODEs.

Explicit Adams–Bashforth method (four-step):

$$y_{n+1} = y_n + \frac{h}{24} \cdot (55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}) \tag{5.25}$$

The explicit Adams–Bashforth method with other step numbers also exists (from 1 to 20). For example the two-step explicit Adams–Bashforth formula is

$$y_{n+1} = y_n + \frac{h}{2} \cdot (3f_n - f_{n-1}) \tag{5.26}$$

In practice, the step number in the Adams–Bashforth method could be adapted to different values during the numerical calculation for ODEs based on the numerical accuracy requirement.

At the beginning of the numerical calculation, due to the lack of "previous" points, a lower-step Adams–Bashforth method could be applied first to initiate calculation.

Implicit Adams–Moulton method (three-step):

$$y_{n+1} = y_n + \frac{h}{24} \cdot (9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}) \tag{5.27}$$

Like the explicit method the implicit Adams–Moulton method with other step numbers also exists (from 1 to 19). For example the three-step implicit Adams–Moulton formula is given in (5.27) and two-step implicit Adams–Moulton formula is given in (5.28)

$$y_{n+1} = y_n + \frac{h}{12} \cdot (5f_{n+1} + 8f_n - f_{n-1}) \tag{5.28}$$

Compared to the explicit method, the implicit method uses additionally the value of the derivative function f_{n+1}. By definition, this value does not count as a "step." Thus a "k-step" Adams–Moulton method has "k + 1" terms of the derivative function f.

5.3.4 Predictor–corrector method

The predictor–corrector (P-C) method is a numerical method for solving (by approximation) the implicit formula with a fixed number of calculation steps:

P-step: Compute the predictor $\hat{y}_{n+1}$ by an explicit k-step Adams–Bashforth method;

E-step: Evaluate the derivative at this approximation: $\hat{f}_{n+1} = f\left(t_{n+1}, \hat{y}_{n+1}\right)$;

C-step: Apply the $\hat{f}_{n+1}$ in an implicit (k-1)-step Adams–Moulton method to obtain the corrector y_{n+1}

E-step: Evaluate the new $f_{n+1} = f(t_{n+1}, y_{n+1})$, which will be used in the next time step.

In practice, one can use only the first three steps to solve the implicit formula in each time step. This approach is called the "PEC method." If all four steps are used, it is called the "PECE method." Furthermore, one can repeat the last two steps again to form a six-step iteration, which is called the "PECECE method." In a general way the coupling of the "P-step" and the "C-step" of the P-C method can be used to approximate any implicit algebraic formula. For example, if the PECE method is applied by using 4(3)-step Adams–Bashforth–Moulton, we can get the following solution:

P-step

$$\hat{y}_{n+1} = y_n + \frac{h}{24} \cdot (55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}) \tag{5.29}$$

E-step

$$\hat{f}_{n+1} = f\left(t_{n+1}, \hat{y}_{n+1}\right) \tag{5.30}$$

C-step

$$y_{n+1} = y_n + \frac{h}{24} \cdot \left(9\hat{f}_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}\right) \tag{5.31}$$

E-step

$$f_{n+1} = f(t_{n+1}, y_{n+1}) \tag{5.32}$$

5.3.5 Backward differentiation formulas

For a given ODE, backward differentiation formulas (BDFs) or a family of gear methods approximate the derivative of the ODEs at a later time step by using previously calculated ODE state variables at current and past time steps.

In contrast to the family of Adams methods, which are developed by integrating the polynomial which interpolates past values of f (thus dy/dt), the BDF methods are developed by differentiating the polynomial which interpolates past values of y and setting the derivative at t_{n+1} to $f(t_{n+1}, y_{n+1})$.

By the definition of the BDF (gear) methods, they need to use the value of y_{n+1} to evaluate the differential at the next time step t_{n+1}. Thus the BDF (gear) methods are, by their nature, implicit numerical methods.

In order to express explicitly the corresponding polynomial interpolation in the function of $(t_{n+1}, y_{n+1}), (t_n, y_n), (t_{n-1}, y_{n-1}), \ldots$ the Lagrangian interpolation form is used.

Here are some common numerical methods from BDF (gear) family for solving ODEs.

BDF4 method (four-step):

$$y_{n+1} = \frac{25}{48}y_n - \frac{36}{25}y_{n-1} + \frac{16}{25}y_{n-2} - \frac{3}{25}y_{n-3} + \frac{12}{25}h \cdot f(t_{n+1}, y_{n+1}) \tag{5.33}$$

BDF2 method (two-step):

$$y_{n+1} = \frac{4}{3}y_n - \frac{1}{3}y_{n-1} + \frac{2}{3}h \cdot f(t_{n+1}, y_{n+1}) \tag{5.34}$$

In numerical methods for solving ODEs, if a numerical method uses only the solution point y_n at the current time step and y_{n+1} to compute the next solution point y_{n+1} (thus does not use any previously computed solution points), it is called the one-step method.

For example, the previously studied Euler method (forward and backward), trapezoid method, and R-K method families are all *one-step methods*.

In contrast to one-step methods, if a numerical method uses a series of previously computed solution points $y_n, y_{n-1}, \ldots .y_{n-k}(k > 1)$ to compute the solution point y_{n+1}, it is called the *multistep method*.

For example the family of Adams methods and the family of BDF methods are both composed of multistep methods.

One-step methods can be considered as numerical methods without "memory." They always use the state y_n as an "initial value" to compute the next state y_{n+1}.

5.3.6 Variable order method and variable step method

5.3.6.1 Fixed order method and variable order method

In most cases the numerical methods for solving ODEs use the same mathematical formula to calculate the ODEs states in each time step.

However, in some cases, the accuracy of the numerical solutions may vary in each time step even with the same formula. Therefore it is natural to think if it is possible to "adapt" the numerical methods with the requirement of accuracy during numerical calculation. This means that if the accuracy in a step cannot be achieved by, say, a three-step Adams–Bashforth method, we may instead use a higher-order method, say a four-step method in that step, and vice versa.

If a numerical method for solving ODEs allows this "order adaptation" with different formulas from a numerical methods family, it is called the *variable order method.*

If a numerical method always uses the same numerical formula, it is called the *fixed order method*.

5.3.6.2 Fixed step method and variable step method

In a numerical method for solving ODEs, if a time step h is given and kept for the entire time range, that is, the solution y is calculated at $y(h), y(2h), \ldots, y(kh)$, it is called the *fixed step method*.

In contrast, if an error tolerance is given and the time step is newly selected based on the local error in each time step, that is, the solution y is calculated at $y(h_1), y(h_2), \ldots, y(h_k)$, it is called the *variable step method.*

In general, a variable step method is more computationally intensive (due to the extra order for error estimation). However, this drawback can be overly balanced by the gain of time for the problem with different time scales.

In a variable step method, if we want to obtain a regular time-spaced solution series (similar to that of a fixed step method), we can simply use the interpolated values between the different variable time points.

5.3.7 Solver accuracy

The numerical methods for solving ODEs are based on mathematical function approximations. Compared to the analytical (exact) solution of the ODEs, the numerical methods introduce numerical errors in each time step.

For an ODE of the form $\frac{dy(t)}{dt} = f(t, y(t))$, let us assume that the exact solutions of the ODEs at different times are $y(t_1), y(t_2), \ldots, y(t_n)$, and its corresponding numerical method solutions are $y_1, y_2, \ldots, y_n$.

The *local truncation error (LTE)* of a numerical method is the error introduced in a single step of the method. If we assume $y_n = y(t_n)$ (no error at t_n) and calculate y_{n+1}, the local truncation error is then defined as

$$\begin{cases} y_n = y(t_n) \\ \tau_n = y(t_{n+1}) - y_{n+1} \end{cases} \tag{5.35}$$

The *global truncation error (GTE)* of a numerical method is the error accumulated from the initial step of the method. If we begin from $y_0 = y(t_0)$ (no error at the initial step) and calculate $y_1, y_2, \ldots, y_n$, the global truncation error at step n is then defined as

$$\begin{cases} y_0 = y(t_0) \\ \varepsilon_n = y(t_n) - y_n \end{cases} \tag{5.36}$$

By the definition of the local truncation error, the exact solution of the ODEs after a time step h can be expressed by Taylor's expansion as $h \rightarrow 0$:

$$y(t_{n+1}) = y(t_n + h) = y(t_n) + h \cdot \frac{dy(t_n)}{dt} + \frac{1}{2}h^2 \frac{d^2y(t_n)}{dt^2} + \ldots \frac{1}{m!}h^p \frac{d^{(p)}y(t_n)}{dt^p} + O\left(h^{p+1}\right) \tag{5.37}$$

where $O\left(h^{p+1}\right)$ is a term of order $p + 1$ as a function of both h and derivatives of $y(t_n)$ and is proportional to h^{p+1}. The numerical method is said to be of the order p if the local truncation error (LTE) satisfies $\tau_n = O\left(h^{p+1}\right)$ as $h \rightarrow 0$. The orders of some commonly used numerical methods are as follows (Table 5.1):

5.3.8 Solver stability

In addition to accuracy, numerical stability is another important property of numerical methods. By the definition of the order of numerical methods, the numerical

Table 5.1 Order of numerical methods.

	Numerical method	**Order of accuracy**
1	Forward Euler method	1
2	Backward Euler method	1
3	Trapezoid method	2
4	Improved Euler method	2
5	k-stage Runge–Kutta methods	k
6	k-step Adams–Bashforth methods	k
7	k-step Adams–Moulton methods	k + 1
8	k-step BDF (gear) methods	k

accuracy is only guaranteed when h is "sufficiently small" ($h \to 0$). However, that definition does not give us enough information about how "small" the value of h should be in practice or, in other words, the range of h to let the numerical results converge (or behave in a stable way). This introduces the notion of "stability" of numerical methods.

The numerical method is called "absolutely stable" in a region of h if the global and local truncation errors of the method with the h in that region are bounded as $t \to \infty$, that is,

$$\varepsilon_n = \left| y(t_n) - y_n \right| < C, t \to \infty \tag{5.38}$$

where C is a positive constant. The corresponding region of h is called the "stability region" of the numerical methods.

For a given ODE problem, the stability of the numerical methods depends not only on the h but also on the "time scales" (or dynamics) of the ODEs.

Consider that ODEs can be written (or approximated) as a first-order state space of dimension m:

$$\frac{dx}{dt} = A \cdot x + f(t) \tag{5.39}$$

where A is a matrix of dimension m with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_m$.

The values of λ represent the different "time scales" of the ODEs. If λ is a real number, the corresponding transient will decay exponentially in time; if λ is a complex number, the corresponding transient will decay sinusoidally in time. A larger value of λ means a faster decay.

For example the stability region of the forward Euler method is defined as

$$|1 + h \cdot \lambda| \leq 1 \tag{5.40}$$

considering the ODE: $\frac{dy(t)}{dt} = -2y(t)$ with $y(0) = 1$. The eigenvalue of the ODE is $\lambda = -2$. Thus to ensure the stability of the forward Euler method the value of time step needs to satisfy

$$h \leq \frac{2}{-\lambda} = 1 \tag{5.41}$$

Table 5.2 and Fig. 5.4 show the numerical results of $\frac{dy(t)}{dt} = -2y(t)$ obtained by the explicit Euler method when the time step h is chosen to be $h = 0.9$ and $h = 1.1$.

In theory, for a specific numerical method, if the expression of the stability region is known, the "stable" time step h can then be calculated by determining the eigenvalues of the ODEs. The region of numerical stability of some simple numerical methods is shown in Table 5.3.

It should be noted that most of the implicit numerical methods (until second order for multistep implicit methods) are stable for any value of $h > 0$, if $\mathrm{Re}(\lambda) < 0$.

Table 5.2 Numerical results of $\frac{dy(t)}{dt} = -2y(t)$.

$h = 0.9$		$h = 1.1$	
time	y	**time**	y
0	1.0000	0	1.0000
0.9	−0.8000	1.1	−1.2000
1.8	0.6400	2.2	1.4400
2.7	−0.5120	3.3	−1.7280
3.6	0.4096	4.4	2.0736
4.5	−0.3277	5.5	−2.4883

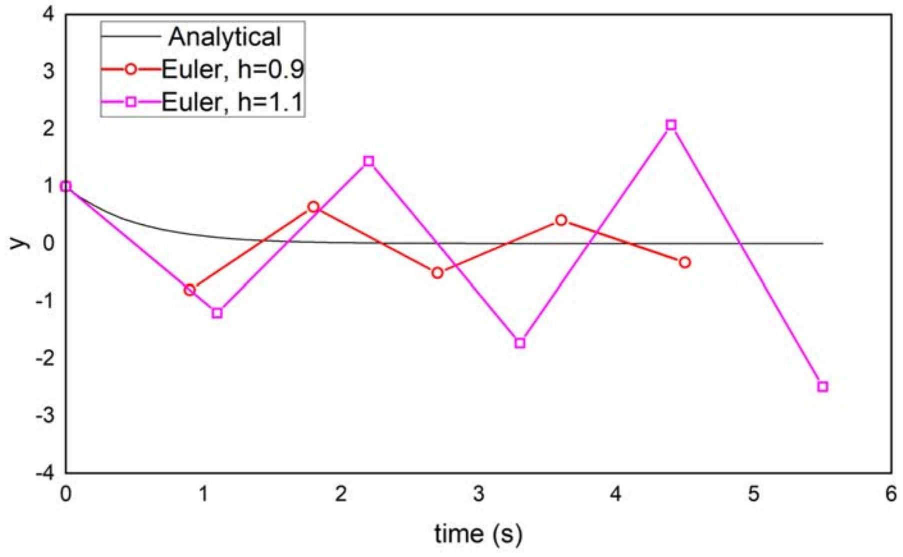


Figure 5.4 Comparison of the explicit Euler results for $h = 0.9$ and $h = 1.1$.

Table 5.3 Region of numerical stability.

Methods	**Stability region**
Explicit Euler	$\lvert 1 + h \cdot \lambda \rvert \leq 1$
Implicit Euler	$1/\lvert 1 - h \cdot \lambda \rvert \leq 1$
Trapezoid (implicit)	$\lvert (2 + h \cdot \lambda)/(2 - h \cdot \lambda) \rvert \leq 1$
Fourth order Runge–Kutta	$-2.78 < h \cdot \lambda < 0$
Explicit second order Adams–Bashforth	$-1 < h \cdot \lambda < 0$
Implicit second order BDF	$-\infty < h \cdot \lambda < 0$

In practice, we may not need to calculate the eigenvalues of the ODE every time and apply them to the region of stability formula. In fact the eigenvalues of the ODEs represent the different "time scales" of the system. If we have a rough understanding of the ODEs that we want to solve, we can, in some cases, estimate the fastest time scale of the ODEs and then choose a value of h which will be much smaller than it. We can also consider the numerical accuracy at the same time.

It is important to distinguish two notions: the "accuracy" and the "stability" of numerical methods. The stability of numerical methods for solving ODEs should always be the first thing to be considered. If a numerical method is not "stable," the results will have no sense. Fortunately, most of the implicit methods (implicit one-step of any order and implicit multistep up to order 2) are stable for any value of the time step h (if the ODEs themselves are "stable"). However, an implicit method is more time-consuming than an explicit one during calculation. The second thing to be considered is the accuracy of a numerical method. Even if the method is "stable" for any value of h, it does not necessarily mean that the results are also accurate (refer to the Euler example before). In general, better numerical accuracy can be achieved by using a smaller time step and a higher-order method. Furthermore a cleverer way to achieve better accuracy is to use the variable step approach with a specified error tolerance.

5.4 Converter model using the nodal analysis method

The switches and circuit components in the power electronics can be modeled by their equivalent circuit model, as presented in Chapters 3 and 4. In general, as shown in Fig. 5.5, after the modeling procedure the power electronics will be represented by a combination of resistors, inductors, capacitors, and power sources. Among these components, the V-I relationship of inductors and capacitors includes derivative terms. By discretizing them using the implicit method presented previously, they can be modeled by companion models, which are equivalent circuit models where a

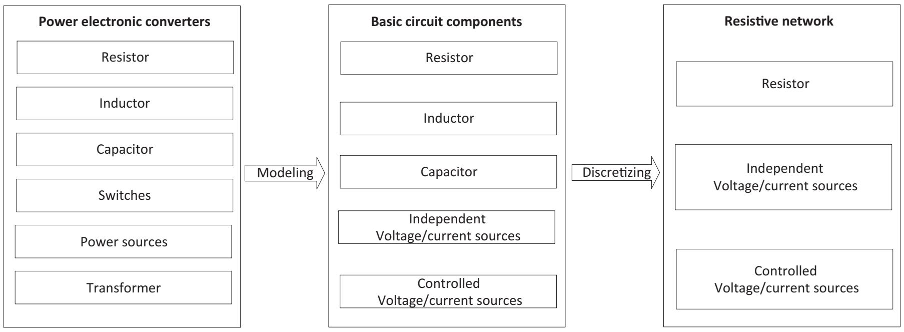

Power electronic converters
Resistor
Inductor
Capacitor
Switches
Power sources
Transformer
Modeling
Basic circuit components
Resistor
Inductor
Capacitor
Independent Voltage/current sources
Controlled Voltage/current sources
Discretizing
Resistive network
Resistor
Independent Voltage/current sources
Controlled Voltage/current sources

Figure 5.5 Modeling and discretizing of power electronic converters.

current source is in parallel to a resistor. Therefore all the components in the power electronic converter can be modeled by equivalent resistors and power sources in discrete time, which constitute the resistive network at each discrete time step. The resistive network analysis method can thus be carried out to solve the circuit, which provides a unified way to get the instantaneous solution of the circuit in each time step. This makes the nodal analysis method particularly efficient and thus widely used in the real-time simulation of power electronic converters.

5.4.1 Nodal voltage analysis method

As an example, a typical boost converter is shown in Fig. 5.6, where R_{in} is the equivalent internal resistance of the voltage source V_{in}. The power switches are modeled by the binary resistor model.

If the inductor and capacitors are discretized by the backward Euler method, considering the companion circuit, the boost converter model will be equivalent to the network in Fig. 5.7. The voltage source and its internal resistance are replaced by their Norton equivalents. All the resistances are modeled in the conductance manner.

If we selected one of the nodes as the reference node, the voltage of any other nodes referred to the reference one is called the nodal voltage. The polarity of the nodal voltage is selected to be negative at the reference node and positive at the other node. In this way the branch voltage can be expressed by the difference between two nodal voltages. Furthermore the branch current can be expressed by the nodal voltages if it can be expressed by the branch voltage. If we apply KCL to

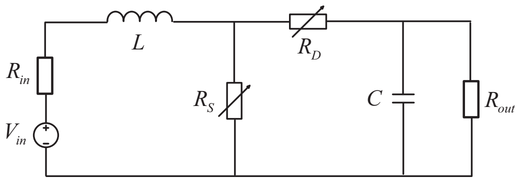


Figure 5.6 Boost converter.

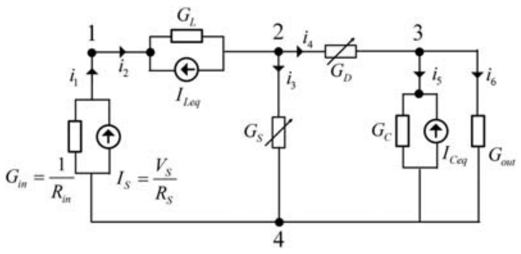


Figure 5.7 Boost converter's equivalent model in discrete time.

every node except for the reference node, for a network containing N nodes, we can write $N-1$ independent equations called nodal voltage equations, in which the nodal voltages are the independent variables. By solving these nodal voltage equations, the branch voltage and branch current of the network can be obtained.

Taking the boost converter model in Fig. 5.7 as an example the circuit contains four nodes. If we select node 4 as the reference, nodes 1, 2, and 3 will be independent. If we merge the conductance in parallel to the current source in one branch, the boost contains six branches. Writing KCL at nodes 1, 2, and 3, we can get

$$\begin{cases} i_1 - i_2 = 0 \\ i_2 - i_3 - i_4 = 0 \\ i_4 - i_5 - i_6 = 0 \end{cases} \tag{5.42}$$

Then, replacing the branch currents with the nodal voltages,

$$\begin{cases} (-G_{in}v_{n1} + I_S) - \left[G_L(v_{n1} - v_{n2}) - I_{Leq}\right] = 0 \\ \left[G_L(v_{n1} - v_{n2}) - I_{Leq}\right] - G_S v_{n2} - G_D(v_{n2} - v_{n3}) = 0 \\ G_D(v_{n2} - v_{n3}) - \left(G_C v_{n3} - I_{Ceq}\right) - G_{out}v_{n3} = 0 \end{cases} \tag{5.43}$$

Organizing (5.43), we can get the nodal voltage equation,

$$\begin{cases} (G_{in} + G_L)v_{n1} - G_L v_{n2} = I_S + I_{Leq} \\ -G_L v_{n1} + (G_L + G_S + G_D)v_{n2} - G_D v_{n3} = -I_{Leq} \\ -G_D v_{n2} + (G_D + G_C + G_{out})v_{n3} = I_{Ceq} \end{cases} \tag{5.44}$$

Define $G_{11} = G_{in} + G_L$, $G_{22} = G_L + G_S + G_D$, and $G_{33} = G_D + G_C + G_{out}$ as the self-conductance of nodes 1, 2, and 3. Define $G_{12} = G_{21} = -G_L$ and $G_{23} = G_{32} = -G_D$ as the mutual conductance between nodes 1, 2 and nodes 2, 3. The self-conductance of one node equals the sum of the conductances in branches connected to it and is always positive; the mutual conductance equals the opposite sum of the conductance in the branches connected between two nodes. Define $I_{S11} = I_S + I_{Leq}$, $I_{S22} = -I_{Leq}$, and $I_{S33} = -I_{Ceq}$ as the injected current of nodes 1, 2, and 3. The injected current equals the algebraic sum of the current sources flowing into the node. By using these definitions the nodal voltage equations can be formulated directly by observing the network rather than starting from the beginning with KCL.

The approaches to formulate the nodal voltage equation can be extended to a network that contains N nodes. $N-1$ independent nodal voltage equations can be formulated, as given in (5.45).

$$\begin{cases} G_{11}v_{n1} + G_{12}v_{n2} + \ldots + G_{1(N-1)}v_{n(N-1)} = I_{s11} \\ G_{21}v_{n1} + G_{22}v_{n2} + \ldots + G_{2(N-1)}v_{n(N-1)} = I_{s22} \\ \ldots \\ G_{(N-1)1}v_{n1} + G_{(N-1)2}v_{n2} + \ldots + G_{(N-1)(N-1)}v_{n(N-1)} = I_{s(N-1)(N-1)} \end{cases} \tag{5.45}$$

The matrix form is given as (5.46).

$$\mathbf{Y_N V = I} \tag{5.46}$$

where $\mathbf{V = [V_1 ... V_N]^T}$ is the vector of nodal potentials, $\mathbf{Y_N}$ is the nodal admittance matrix, and $\mathbf{I = [I_1 ... I_N]^T}$ is the vector of the current source.

After solving the nodal voltages, the branch currents can be easily obtained by the voltage—current relationship of each component.

The nodal analysis method has limitations. For example, if the voltage source is not accompanied by a resistor, the formulation of the nodal voltage equation is problematic. The branch current of the voltage source cannot be expressed by the nodal voltage. Moreover, if the network has a controlled source, the dependence of nodal voltage on the sources will be broken. Therefore, in the practical simulation solver, the modified nodal analysis method is applied.

5.4.2 Modified nodal voltage analysis method

In the modified method, if an ideal voltage source without resistance is present in a branch, the nodal voltage equation can be formulated by treating the branch current of the voltage source as a new variable. The branch current is defined as an injected current. Considering the boost in Fig. 5.8 whose input is an ideal voltage source, we define the branch current of the voltage source V_{in} as a new variable i_1 and write the nodal voltage equation as (5.47).

$$\begin{cases} G_L v_{n1} - G_L v_{n2} = i_1 + I_{Leq} \\ -G_L v_{n1} + (G_L + G_S + G_D) v_{n2} - G_D v_{n3} = -I_{Leq} \\ -G_D v_{n2} + (G_D + G_C + G_{out}) v_{n3} = I_{Ceq} \end{cases} \tag{5.47}$$

The ideal voltage source V_{in} introduces a new constraint of v_{n1} and v_{n2} as given in (5.48).

$$v_{n1} = V_{in} \tag{5.48}$$

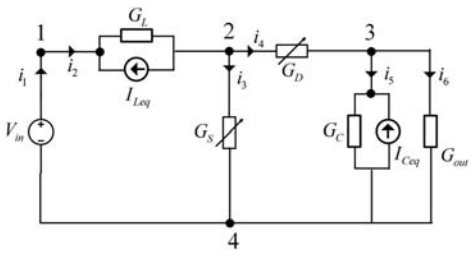


Figure 5.8 Boost converter with an ideal voltage source.

By combining (5.47) and (5.48) the boost converter model can then be correctly solved.

The converter can also be modeled as a network containing a controlled voltage source. For example, for a converter containing a transformer, controlled voltage and current sources will be present in the network. Therefore the nodal analysis method has to deal with the controlled sources.

The main idea to deal with controlled sources is to first treat them as independent sources and write the nodal voltage equations. Then additional equations are written to relate the controlling variables of the controlled sources to the nodal voltages [2]. Taking the network in Fig. 5.9 containing the controlled current source and controlled voltage source as an example, the nodal voltage equation can be formulated in the following manner.

First the controlled sources are treated as the independent sources and the nodal voltage equations are written.

Node 1:

$$\frac{1}{R_1}v_{n1} - \frac{1}{R_1}v_{n2} = I_2 - g_m V \tag{5.49}$$

Node 2:

$$-\frac{1}{R_1}v_{n1} + \left(\frac{1}{R_1} + \frac{1}{R_2}\right)v_{n2} = I_3 \tag{5.50}$$

Node 3:

$$\frac{1}{R_3}v_{n3} = g_m V - I_3 \tag{5.51}$$

The additional relationship can be

$$\begin{cases} v_{n1} = V_S \\ v_{n2} - v_{n3} = R_m I_1 \end{cases} \tag{5.52}$$

Among them the controlling variables V and I_1 can be expressed by the nodal voltage, as given in

$$\begin{cases} V = v_{n2} \\ I_1 = \dfrac{v_{n1} - v_{n2}}{R_1} \end{cases} \tag{5.53}$$

Reorganizing the equations, we can get the nodal voltage equations of the circuit in Fig. 5.9.

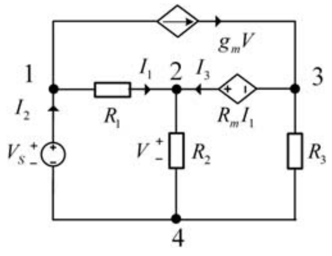


Figure 5.9 An example circuit containing the controlled sources.

5.5 Converter model using the state-space method

In the nodal analysis method, the components in the power converter are first discretized using the numerical integration method and then transformed into the companion circuit. Therefore the power electronic converter will be equivalent to a resistive network in each time step and the nodal voltage method can be used to solve the converter model. In contrast to the nodal analysis method, the state-space method first formulates the circuit into the state-space equation and then discretizes it to be solved step by step.

5.5.1 State-space representation of electric networks

The state-space model of a linear system is given by

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{5.54}$$

where **A** is the state matrix, **x** is the state vector, **B** is the input matrix, and **u** is the input vector.

The state variables in the power electronic converters are usually selected as the inductor currents and capacitor voltages. Since the derivative of the inductor current is voltage and the derivative of capacitor voltage is current, the state-space equations can be formulated in the following two steps [3]:

1. Applying KVL to the loops in the converter to formulate the voltage relationship for the derivative of the inductor currents;
2. Applying KCL to the nodes connected to the capacitors to formulate the current relationship for the derivative of capacitor voltages.

We use the same boost converter shown in Fig. 5.10 to demonstrate this modeling process. The switches are modeled by the binary resistor model.

Step 1: Write the voltage equations in loops 1 and 2 using KVL

Loop 1:

$$-V_{in} + L\frac{di_1}{dt} + R_S(i_1 - i_2) = 0 \tag{5.55}$$

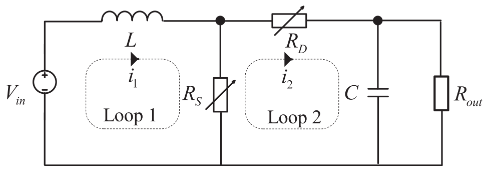


Figure 5.10 Boost converter.

Loop 2:

$$R_D i_2 + v_C + (i_2 - i_1)R_S = 0 \tag{5.56}$$

i_1 is the state variable i_L, and i_2 can be substituted by the state variables and independent sources. Loop 2 can be rewritten by

Loop 2:

$$i_2 = \frac{R_S}{R_S + R_D} i_L - \frac{1}{R_S + R_D} v_C \tag{5.57}$$

Then we get

$$L\frac{di_L}{dt} = -\frac{R_S R_D}{R_S + R_D} i_L - \frac{R_S}{R_S + R_D} v_C + V_{in} \tag{5.58}$$

Step 2: Write the current equations regarding the capacitor node using KCL

$$i_2 = C\frac{dv_C}{dt} + \frac{v_C}{R_{out}} \tag{5.59}$$

Eliminate i_2 by (5.57); then we get

$$C\frac{dv_C}{dt} = \frac{R_S}{R_S + R_D} i_L - \left(\frac{1}{R_S + R_D} + \frac{1}{R_{out}}\right) v_C \tag{5.60}$$

Organize (5.58) and (5.60) in the state-space form

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{5.61}$$

where

$$\mathbf{x} = \begin{bmatrix} i_L \\ v_C \end{bmatrix}, \mathbf{u} = V_{in}, \quad \mathbf{A} = \begin{bmatrix} L & 0 \\ 0 & C \end{bmatrix}^{-1} \begin{bmatrix} -\dfrac{R_S R_D}{R_S + R_D} & -\dfrac{R_S}{R_S + R_D} \\ \dfrac{R_S}{R_S + R_D} & -\left(\dfrac{1}{R_S + R_D} + \dfrac{1}{R_{out}}\right) \end{bmatrix}, \quad B = \begin{bmatrix} L^{-1} \\ 0 \end{bmatrix}.$$

5.5.2 Discrete-time state-space model of electric networks

Discretize (5.61) using backward Euler method, and Eq. (5.62) can be got.

$$\mathbf{x_{n+1}} = \mathbf{x_{n+1}} + h[\mathbf{Ax_{n+1}} + \mathbf{Bu_{n+1}}] \tag{5.62}$$

Organizing (5.62) as (5.63) leads to the main challenge in implementing the real-time simulation, which is to solve the system of linear equations given by (5.64).

$$[\mathbf{I}-h\mathbf{A}]\mathbf{x_{n+1}} = \mathbf{x_n} + h\mathbf{Bu_{n+1}} \tag{5.63}$$

$$\mathbf{Mx_{n+1}} = \mathbf{z} \tag{5.64}$$

where $\mathbf{M} = [\mathbf{I}-\mathbf{hA}], \mathbf{z} = \mathbf{x_n} + \mathbf{hBu_{n+1}}$

Regardless of the approach used in the converter modeling, both nodal analysis method and state-space method will finally lead to solving the system of linear equations ($\mathbf{Y_N V} = \mathbf{I}$ for nodal analysis method and $\mathbf{Mx_{n+1}} = \mathbf{z}$ for state-space method as shown in this chapter). If the nodal analysis method is applied, the order of the linear equations equals the node number subtracting 1. If the state-space method is used, the order of the linear equations equals the number of independent state variables (Fig. 5.11).

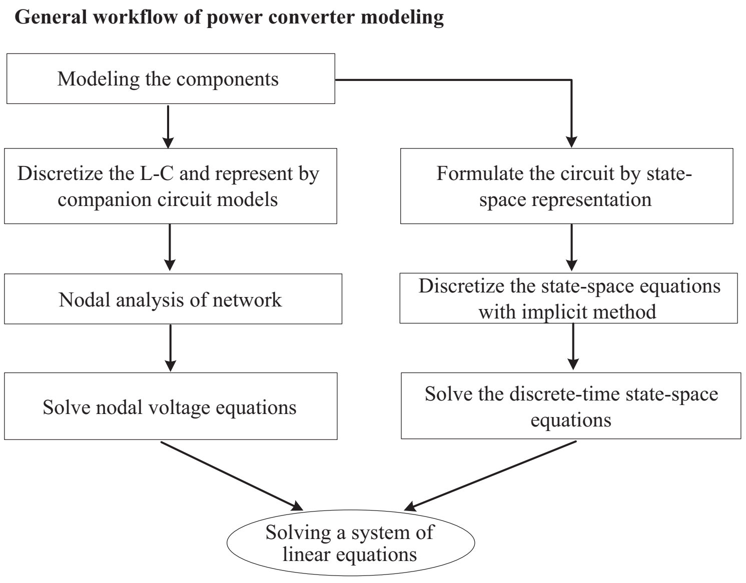

General workflow of power converter modeling
Modeling the components
Discretize the L-C and represent by companion circuit models
Formulate the circuit by state-space representation
Nodal analysis of network
Discretize the state-space equations with implicit method
Solve nodal voltage equations
Solve the discrete-time state-space equations
Solving a system of linear equations

Figure 5.11 Workflow of power electronic converter modeling.

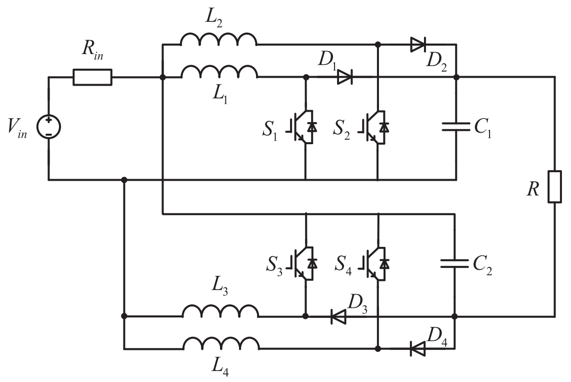


Figure 5.12 Four-phase floating interleaved boost converter.

5.6 System-level converter model

Let us consider more complex power electronic converters to demonstrate the modeling procedure using the nodal analysis method and the state-space method. A four-phase floating interleaved boost converter (FIBC) shown in Fig. 5.12 is used as an illustrative example. In this system-level model a binary resistor switch model is used. The parasitic parameters of the switches, the capacitors, and the inductors are not considered.

5.6.1 System-level model using the nodal analysis method

The backward Euler method is first used to discrete the inductor and capacitor. The inductor and capacitor are then modeled by the companion circuit, as shown in Fig. 5.13. The parameters of the companion circuit model are given in (5.65) and (5.66).

$$\begin{cases} G_L = \dfrac{h}{L} \\ J_L^{n+1} = -\, i_L^n \end{cases} \tag{5.65}$$

$$\begin{cases} G_C = \dfrac{C}{h} \\ J_C^{n+1} = g_C v_c^n \end{cases} \tag{5.66}$$

By replacing the inductor, capacitor, and switches with their equivalent models the FIBC will be represented by the network in Fig. 5.14.

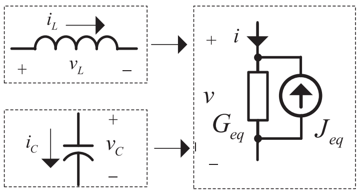


Figure 5.13 The companion circuit model of inductor and capacitor.

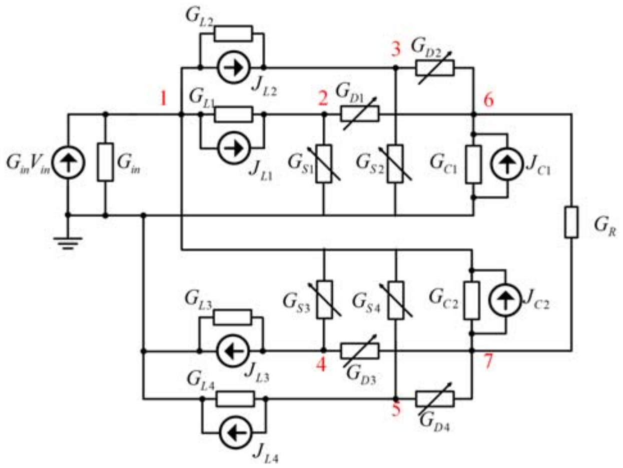


Figure 5.14 FIBC networks modeled by the companion circuit and binary resistor switch model.

Using the nodal analysis method the network can be formulated by

$$\mathbf{Y} \cdot \mathbf{v} = \mathbf{I} \tag{5.67}$$

$v = \begin{bmatrix} v_1 & v_2 & \dots & v_7 \end{bmatrix}^T$ represents the node potential.

$I = \begin{bmatrix} G_{in}V_{in} - J_{L1} - J_{L2} + J_{C2} & J_{L1} & J_{L2} & -J_{L3} & -J_{L4} & J_{C1} & -J_{C2} \end{bmatrix}^T$ represents the injected current from current sources into different nodes.

$$\begin{cases} J_{L1}^{n+1} = i_{L1}^n = J_{L1}^n + G_{L1}\left(v_1^n - v_2^n\right) \\ J_{L2}^{n+1} = i_{L2}^n = J_{L2}^n + G_{L2}\left(v_1^n - v_3^n\right) \\ J_{L3}^{n+1} = i_{L3}^n = J_{L3}^n + G_{L3}v_4^n \\ J_{L4}^{n+1} = i_{L4}^n = J_{L4}^n + G_{L4}v_5^n \end{cases} \tag{5.68}$$

$$\begin{cases} J_{C1}^{n+1} = G_{C1} v_6^n \\ J_{C2}^{n+1} = G_{C2}\left(v_1^n - v_7^n\right) \end{cases} \tag{5.69}$$

Y is the admittance matrix describing the admittance connected to every node.

$$Y = \begin{pmatrix} \begin{matrix} G_{in} + G_{L1} + G_{L2} \\ + G_{S3} + G_{S4} + G_{C2} \end{matrix} & -G_{L1} & -G_{L2} & -G_{S3} & -G_{S4} & 0 & -G_{C2} \\ -G_{L1} & G_{L1} + G_{S1} + G_{D1} & 0 & 0 & 0 & -G_{D1} & 0 \\ -G_{L2} & 0 & G_{L2} + G_{S2} + G_{D2} & 0 & 0 & -G_{D2} & 0 \\ -G_{S3} & 0 & 0 & G_{L3} + G_{S3} + G_{D3} & 0 & 0 & -G_{D3} \\ -G_{S4} & 0 & 0 & 0 & G_{L4} + G_{S4} + G_{D4} & 0 & -G_{D4} \\ 0 & -G_{D1} & -G_{D2} & 0 & 0 & \begin{matrix} G_{D1} + G_{D2} \\ + G_{C1} + G_{R} \end{matrix} & -G_{R} \\ -G_{C2} & 0 & 0 & -G_{D3} & -G_{D4} & -G_{R} & \begin{matrix} G_{D3} + G_{D3} + \\ G_{C2} + G_{R} \end{matrix} \end{pmatrix}$$

5.6.2 System-level model using state-space equation method

We can also use the state-space equation to model the FIBC. For simplicity the internal resistance of the input voltage source is omitted, and the switches are modeled by the binary resistor model (Fig. 5.15).

By following the method presented in Section 5.5.1, we can formulate the state equations step by step. The first step is to use KVL in the loop to represent the derivative of the inductor current. The FIBC has four inductors, and

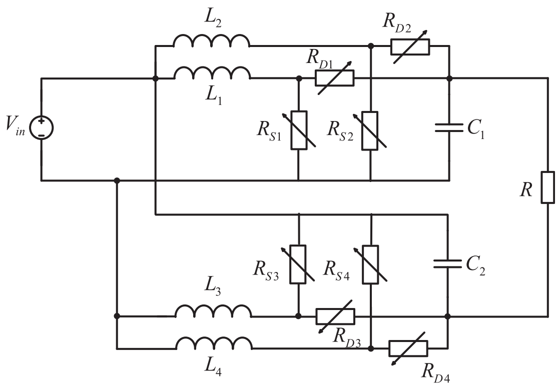


Figure 5.15 Four-phase floating interleaved boost converter modeled by the binary resistor model.

the state equations can be written as (5.70) with the help of the boost converter shown in (5.58).

$$\begin{cases} L_1\dfrac{di_{L1}}{dt} = -\dfrac{R_{S1}R_{D1}}{R_{S1}+R_{D1}}i_{L1} - \dfrac{R_{S1}}{R_{S1}+R_{D1}}v_{C1} + V_{in} \\ L_2\dfrac{di_{L2}}{dt} = -\dfrac{R_{S2}R_{D2}}{R_{S2}+R_{D2}}i_{L2} - \dfrac{R_{S2}}{R_{S2}+R_{D2}}v_{C1} + V_{in} \\ L_3\dfrac{di_3}{dt} = -\dfrac{R_{S3}R_{D3}}{R_{S3}+R_{D3}}i_{L3} - \dfrac{R_{S3}}{R_{S3}+R_{D3}}v_{C2} + V_{in} \\ L_4\dfrac{di_4}{dt} = -\dfrac{R_{S4}R_{D4}}{R_{S4}+R_{D4}}i_{L4} - \dfrac{R_{S4}}{R_{S4}+R_{D4}}v_{C2} + V_{in} \end{cases} \tag{5.70}$$

The second step is to use the KCL in the nodes connected to the capacitors, as given in (5.71) and (5.72).

$$i_{d1} + i_{d2} - \frac{v_{out}}{R_L} = C_1\frac{dv_{C1}}{dt} \tag{5.71}$$

$$i_{d3} + i_{d4} - \frac{v_{out}}{R_L} = C_2\frac{dv_{C2}}{dt} \tag{5.72}$$

The diode current and the output voltage can be represented by the state variables as shown in (5.73) and (5.74).

$$\begin{cases} i_{d1} = \dfrac{R_{S1}}{R_{S1}+R_{D1}} i_{L1} - \dfrac{1}{R_{S1}+R_{D1}} v_{C1} \\ i_{d2} = \dfrac{R_{S2}}{R_{S2}+R_{D2}} i_{L2} - \dfrac{1}{R_{S2}+R_{D2}} v_{C1} \\ i_{d3} = \dfrac{R_{S3}}{R_{S3}+R_{D3}} i_{L3} - \dfrac{1}{R_{S3}+R_{D3}} v_{C2} \\ i_{d4} = \dfrac{R_{S4}}{R_{S4}+R_{D4}} i_{L4} - \dfrac{1}{R_{S4}+R_{D4}} v_{C2} \end{cases} \tag{5.73}$$

$$v_{out} = v_{C1} + v_{C2} - V_{in} \tag{5.74}$$

Reorganize (5.71) to (5.74), and the state equations regarding the capacitor current are formulated by (5.75).

$$\begin{cases} C_1 \dfrac{dv_{C1}}{dt} = \dfrac{R_{S1}}{R_{S1}+R_{D1}} i_{L1} + \dfrac{R_{S2}}{R_{S2}+R_{D2}} i_{L2} - \left(\dfrac{1}{R_L} + \dfrac{1}{R_{S1}+R_{D1}} + \dfrac{1}{R_{S2}+R_{D2}} \right) \\ \quad v_{C1} - \dfrac{1}{R_L} v_{C2} + \dfrac{1}{R_L} V_{in} \\ C_2 \dfrac{dv_{C2}}{dt} = \dfrac{R_{S3}}{R_{S3}+R_{D3}} i_{L3} + \dfrac{R_{S4}}{R_{S4}+R_{D4}} i_{L4} - \dfrac{1}{R_L} v_{C1} \\ \quad - \left(\dfrac{1}{R_L} + \dfrac{1}{R_{S3}+R_{D3}} + \dfrac{1}{R_{S4}+R_{D4}} \right) v_{C2} + \dfrac{1}{R_L} V_{in} \end{cases} \tag{5.75}$$

Combining (5.70) and (5.75) the state-space representation of the FIBC can be obtained,

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \tag{5.76}$$

$$\text{where } \mathbf{x} = \begin{bmatrix} i_{L1} \\ i_{L2} \\ i_{L3} \\ i_{L4} \\ v_{C1} \\ v_{C2} \end{bmatrix}, \mathbf{u} = V_{in}, \mathbf{W} = \begin{bmatrix} L_1 & & & & & \\ & L_2 & & & & \\ & & L_3 & & & \\ & & & L_4 & & \\ & & & & C_1 & \\ & & & & & C_2 \end{bmatrix}^{-1}, \mathbf{B} = \mathbf{W} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \dfrac{1}{R_L} \\ \dfrac{1}{R_L} \end{bmatrix}$$

$$\mathbf{A} = \mathbf{W}\begin{bmatrix} -\frac{R_{S1}R_{D1}}{R_{S1}+R_{D1}} & 0 & 0 & 0 & -\frac{R_{S1}}{R_{S1}+R_{D1}} & 0 \\ 0 & -\frac{R_{S2}R_{D2}}{R_{S2}+R_{D2}} & 0 & 0 & -\frac{R_{S2}}{R_{S2}+R_{D2}} & 0 \\ 0 & 0 & -\frac{R_{S3}R_{D3}}{R_{S3}+R_{D3}} & 0 & 0 & -\frac{R_{S3}}{R_{S3}+R_{D3}} \\ 0 & 0 & 0 & -\frac{R_{S4}R_{D4}}{R_{S4}+R_{D4}} & 0 & -\frac{R_{S4}}{R_{S4}+R_{D4}} \\ \frac{R_{S1}}{R_{S1}+R_{D1}} & \frac{R_{S2}}{R_{S2}+R_{D2}} & 0 & 0 & -\left(\frac{1}{R_L}+\frac{1}{R_{S1}+R_{D1}}+\frac{1}{R_{S2}+R_{D2}}\right) & -\frac{1}{R_L} \\ 0 & 0 & \frac{R_{S3}}{R_{S3}+R_{D3}} & \frac{R_{S4}}{R_{S4}+R_{D4}} & -\frac{1}{R_L} & -\left(\frac{1}{R_L}+\frac{1}{R_{S3}+R_{D3}}+\frac{1}{R_{S4}+R_{D4}}\right) \end{bmatrix}$$

By discretizing (5.76) with backward Euler method, (5.77) can be obtained to solve the instant values of the FIBC state variables.

$$\mathbf{Mx^{n+1} = z} \tag{5.77}$$

where $\mathbf{M = [I - hA], z = x^n + hBu}$.

Here is a simulation test of an FIBC converter by using both the nodal analysis modeling method and state-space method. Eqs. (5.67) and (5.77) are finally formulated and solved by the MATLAB® matrix inversion function. The simulation is implemented in the Simulink environment, and the models are solved in a programmed M-function according to the flow chart shown in Fig. 5.16. The backward Euler method is used for both the component discretization in nodal analysis and the discretization of state-space equation with a time step of 1 μs. An FIBC is also modeled by using the Simulink

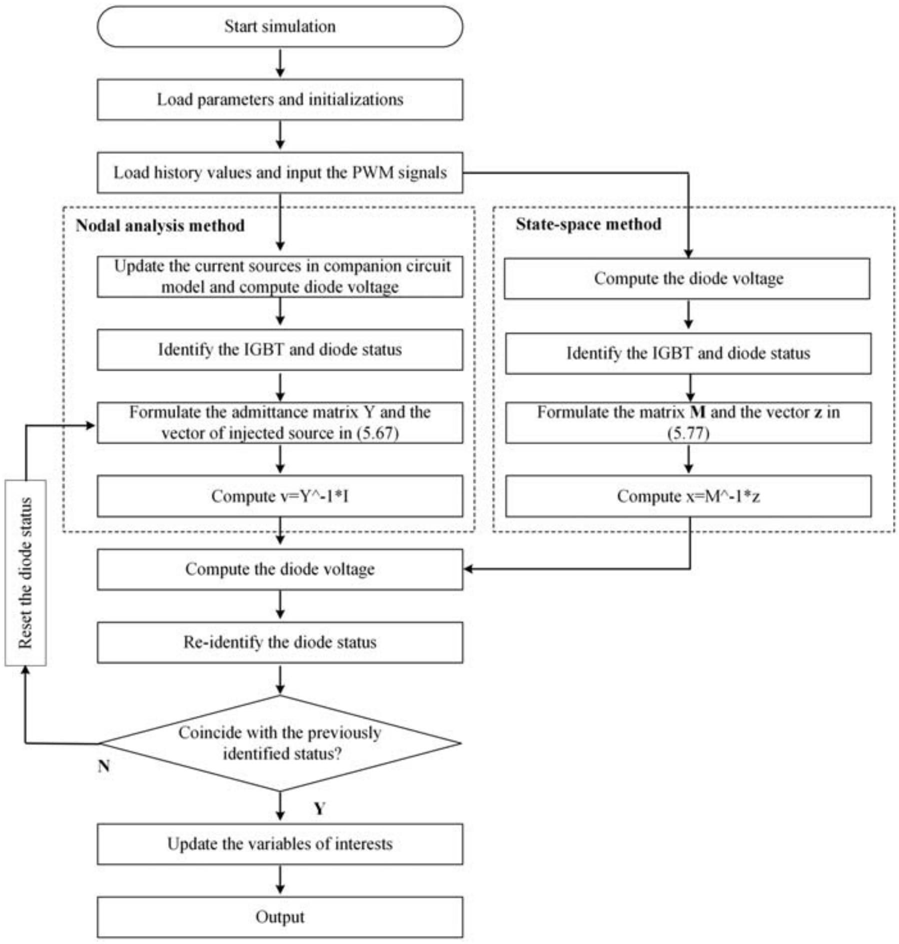

Start simulation
Load parameters and initializations
Load history values and input the PWM signals
Nodal analysis method
Update the current sources in companion circuit model and compute diode voltage
Identify the IGBT and diode status
Formulate the admittance matrix Y and the vector of injected source in (5.67)
Compute v=Y^-1*I
State-space method
Compute the diode voltage
Identify the IGBT and diode status
Formulate the matrix M and the vector z in (5.77)
Compute x=M^-1*z
Compute the diode voltage
Re-identify the diode status
Coincide with the previously identified status?
N
Y
Reset the diode status
Update the variables of interests
Output

Figure 5.16 Converter model solving flow chart in the nodal analysis method and state-space method.

SimPowerSystem (SPS) library and solved by the Simulink solver (backward Euler is chosen) with a time step of 1 μs, which is set as the reference to compare with the simulation results produced by the nodal analysis and state-space equation. The FIBC is operating under the open-loop condition with a 0.6 duty cycle. The PWM signals of four boost units in FIBC have a phase shift of 90 degrees to each other. The parameters of the FIBC are given in Table 5.4, and the simulation environment is shown in Fig. 5.17.

Figs. 5.18 and 5.19 show the FIBC simulation results of Simulink solver, nodal analysis method, and state-space method. It can be observed that the results of nodal analysis and state-space method are consistent with those of Simulink solver, which proves that the nodal analysis method and state-space method are equivalent in the discrete time domain.

Table 5.4 Parameters of FIBC.

Parameters	Values	Parameters	Values
V_{in}	72 V	R_{in}	1e − 6 Ω
$L_1 \sim L_4$	400 μH	$C_1 \sim C_2$	1 mF
R_L	8 Ω	Switch R_{ON}	0.0010 Ω
Switching frequency	10 kHz	Switch R_{OFF}	100 kΩ
Duty cycle	0.6	Simulation time step	1 μs

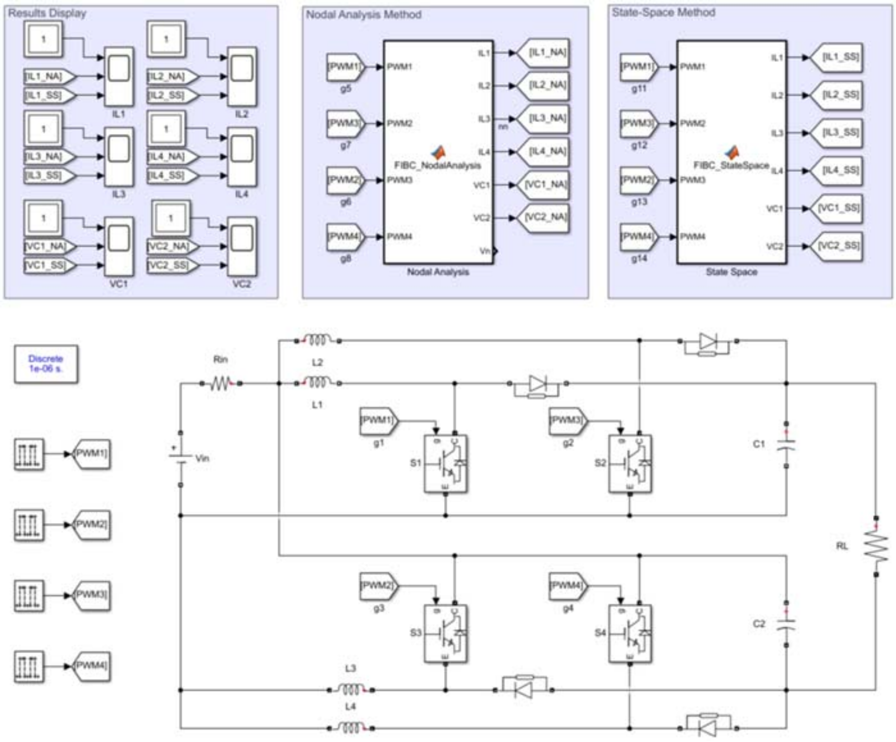


Figure 5.17 Simulation environment in MATLAB/Simulink.

5.7 Device-level converter model

As discussed in Chapter 3, the binary resistor switch model represents a system-level switching behavior and neglects the switching transient. When the transient switch models such as the curve-fitting model are applied to the converter real-time simulation, the detailed device behaviors can be further described, and the simulation can be treated as a device-level real-time simulation.

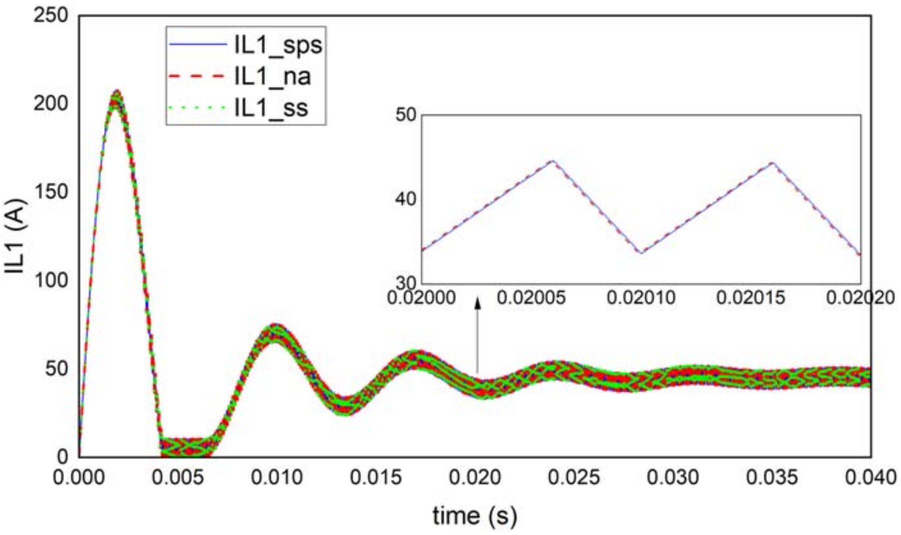


Figure 5.18 Simulation results of inductor current of i_{L1}. Solid line: the results of SPS model; dashed line: the results of nodal analysis; dotted line: the results of state-space method.

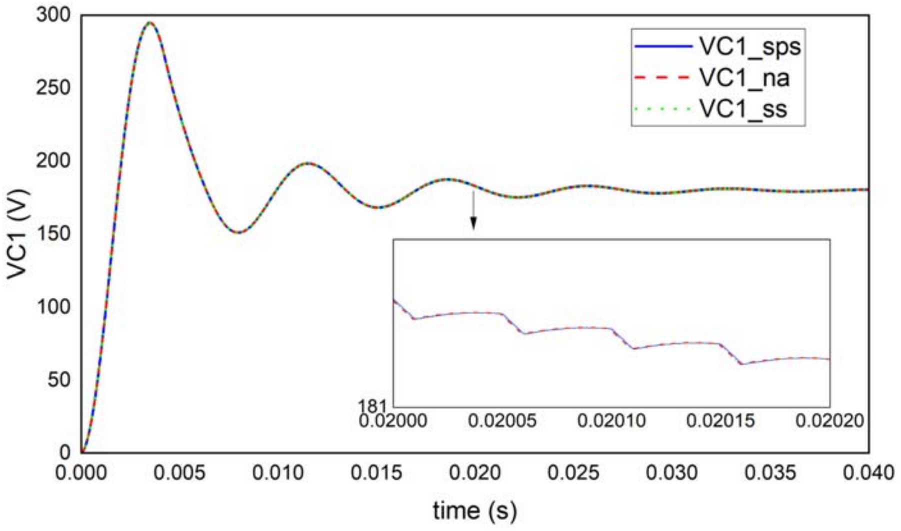


Figure 5.19 Simulation results of capacitor voltage of v_{C1}. Solid line: the results of SPS model; dashed line: the results of nodal analysis; dotted line: the results of state-space method.

In the reminder of this section, we will also focus on the FIBC topology to introduce the device-level converter modeling using the curve-fitting switch model [4] as well as the two-level quasi-transient switch model [5].

5.7.1 Device-level model using curve-fitting-based switch model

5.7.1.1 Steady-state modeling

In the steady turn-on state, the IGBT and diode behaviors can be modeled by a serial of the voltage source and resistance representing the forward threshold voltage and the conduction resistance, respectively, as shown in Fig. 5.20. The parameters are usually extracted by linearizing the saturation region of IGBT output characteristics and the diode forward characteristics. In the steady turn-off state, the conduction resistance is replaced by a large resistance to represent the block state of the switch branch. The converter modeled by the steady-state switches can be solved by either the nodal analysis method or the state-space equation method.

5.7.1.2 Switching transient modeling

The half-bridge (HB) unit in Fig. 5.21 will be used as the computing unit to model the switching transient by treating the input current and dc-link voltage as the constants in each simulation time step. It is because in FIBC the inductor current and capacitor voltage have much slower dynamics than the switching transient. The

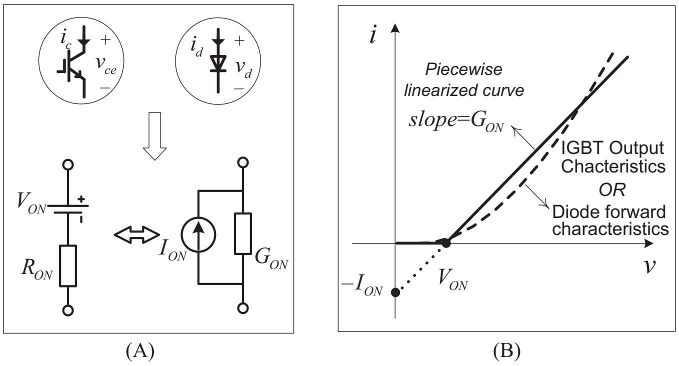


Figure 5.20 (A) Steady-state IGBT/diode model; (B) parameter characterization.

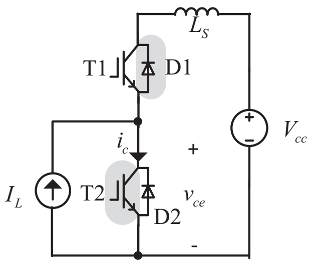


Figure 5.21 Half-bridge circuit.

upper IGBT and lower diode constitute one commutating unit when the input current is flowing out of the HB, while the lower IGBT and upper diode constitute the other commutating unit when the input current is injecting into the HB. The offline measured waveforms of the inductive switching transient are shown in Fig. 5.22, obtained under the nominal current and voltage, denoted by superscript *. One of the simplest curve-fitting approaches is to rescale the offline measured waveforms according to the magnitude of the steady-state current and voltage, and by doing so, the transient collector current i_c and the collector-emitter voltage v_{ce} can be expressed by (5.78). I_c^{st} is the steady-state collector current where the turn-on transient ends and V_{ce}^{st} is the steady-state collector-emitter voltage where the turn-off transient begins, and they can be obtained by the results of steady-state switch model.

$$\begin{cases} i_c = \dfrac{I_c^{st}}{I_c^*} i_c^* \\ \\ v_{ce} = \dfrac{V_{ce}^{st}}{V_{ce}^*} v_{ce}^* \end{cases} \tag{5.78}$$

The curve-fitting-based device-level converter model can be solved following the flow chart in Fig. 5.23. The switching status should be identified first, which decides whether a steady-state model or transient model is used to get the solution of the converter. If the switch is in the steady-state turn-on and turn-off states, the steady-state switch mode shown in Fig. 5.20 is used and the converter network can be solved by the nodal analysis method for example. If the switch event just happens, the transient model will be adopted, which rescales the offline measured waveforms to get the transient solution of the HB. The results of HB are then fed back to the converter network model and used for the solution of the rest of the network.

In this example, we selected the FF450R12ME4 module as the modeled IGBT and diode, and the measured transient waveforms and switching times are collected

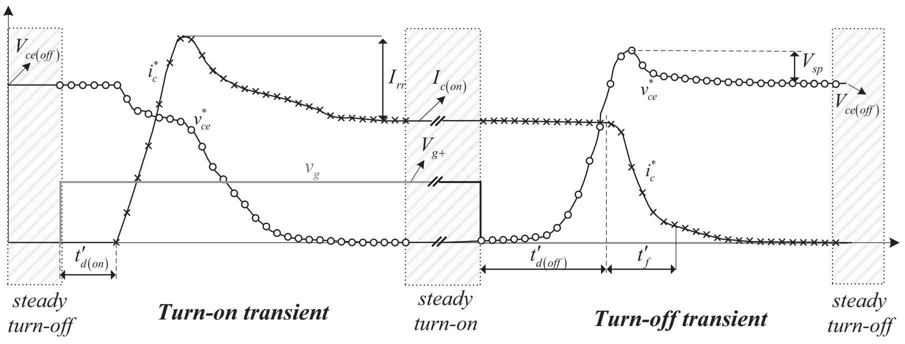


Figure 5.22 The inductive switching transient of the IGBT/diode commutating unit.

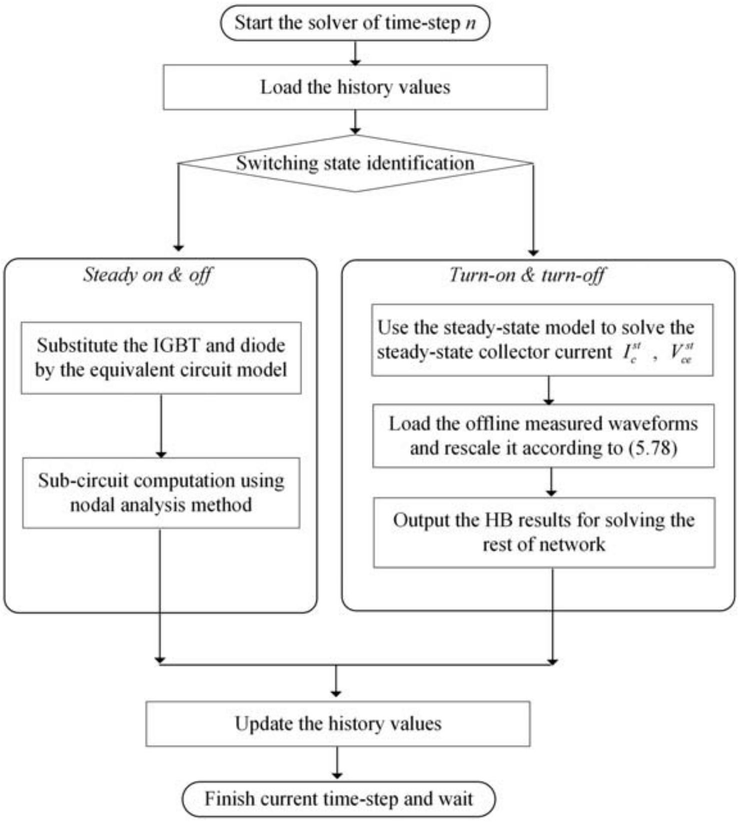

Start the solver of time-step n
Load the history values
Switching state identification
Steady on & off
Substitute the IGBT and diode by the equivalent circuit model
Sub-circuit computation using nodal analysis method
Turn-on & turn-off
Use the steady-state model to solve the steady-state collector current I_c^{st} , V_{ce}^{st}
Load the offline measured waveforms and rescale it according to (5.78)
Output the HB results for solving the rest of network
Update the history values
Finish current time-step and wait

Figure 5.23 The flow chart for solving curve-fitting-based device-level converter model.

based on the results of the Saber Model Architect tool IGBT model under 600 V/ 400 A rated values. The curve-fitting-based converter model is simulated by using M function in Simulink with a time step of 25 ns.

Figs. 5.24 and 5.25 show the simulation results of inductor currents and capacitor voltages, which represent the system-level behaviors. Meanwhile, Fig. 5.26 shows the current and voltage of IGBT, which represent the device-level behaviors in the simulation.

5.7.2 Device-level model using two-level quasi-transient switch model

As presented in Chapter 3, the basic modeling structure of the two-level quasi-transient switch model is shown in Fig. 5.27. It contains two levels of simulation, the system-level simulation and the device-level simulation. In the system-level simulation the IGBT static model participates in the converter network solutions to obtain the branch currents and voltages at each time step. The IGBT static model is

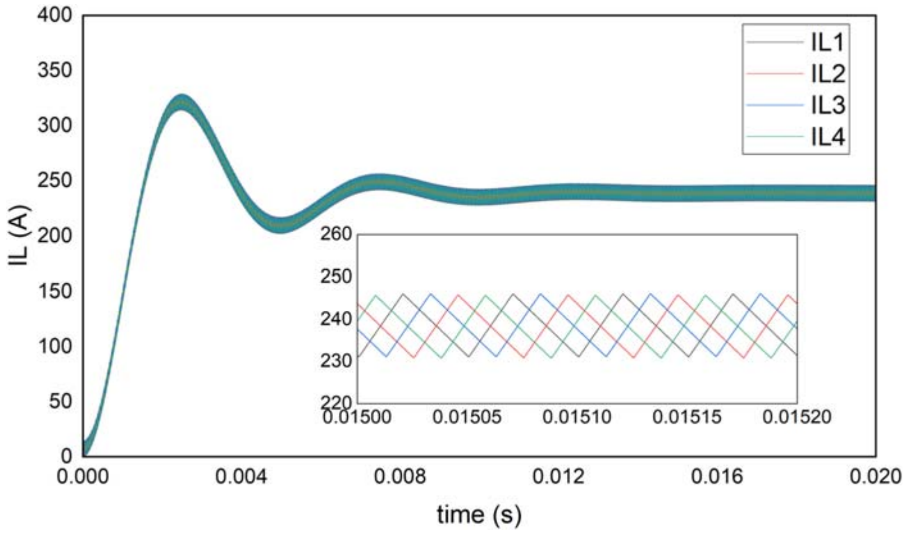


Figure 5.24 Simulation results of four inductor currents.

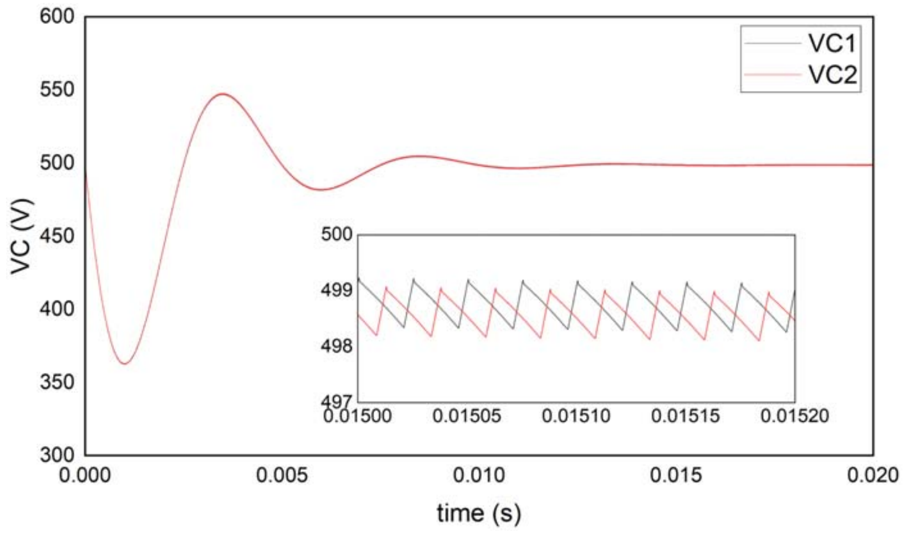


Figure 5.25 Simulation results of two capacitor voltages.

represented by a serial connection of a voltage source and a resistor as given in Fig. 5.20. Meanwhile the parameters required by the IGBT dynamic model in the device-level simulation can be computed at the same time, including the time intervals of the different transient stages, and the corresponding slopes/time constants of the current and voltage variations in the transient waveforms.

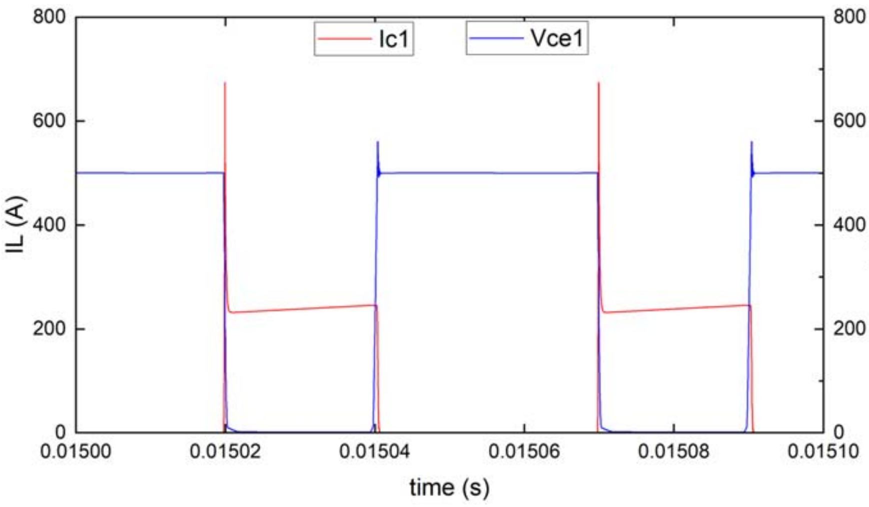


Figure 5.26 Transient waveforms of IGBT 1 in FIBC device-level simulation.

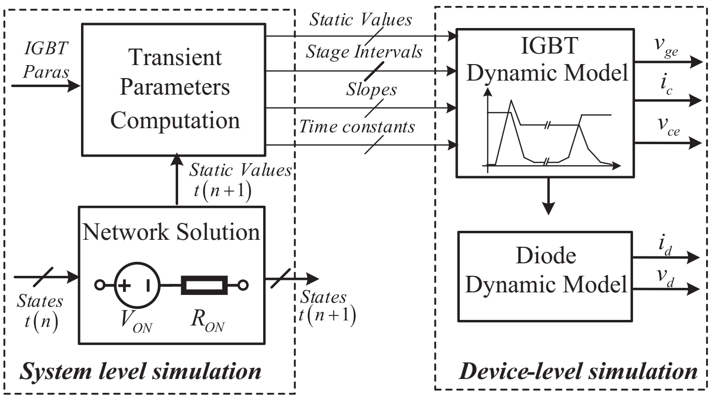


Figure 5.27 Modeling structure of two-level quasitransient switch model.

5.7.2.1 FIBC network solution with static IGBT model

The network is formulated by the nodal analysis method. The network solution will be formulated by a system of linear equations as shown in previous sections.

5.7.2.2 Quasi-transient switching modeling by the high-resolution model

The four switching pairs in FIBC are modeled by the quasi-transient high-resolution model presented in Chapter 3. Based on the transient parameters computed in the system-level simulation the transient waveforms of IGBT can be generated by simply using the linear function and exponential function with respect to time. Thus the switching waveforms can be generated with a small computing effort and have a high resolution in discrete time. To verify the accuracy of the model, the system-level simulation results are compared to the results from the SPS model implemented in Simulink where IGBT and the diode are represented by the forward voltage in series with a resistance. Meanwhile the device-level performances are verified by the Saber offline simulation tool. The used IGBTs and diodes are modeled based on the datasheet of CM100DY-12H IGBT module from MITSUBISHI. FIBC operates first under the open-loop condition.

5.7.2.3 System-level simulation results

The system-level simulation results are depicted in Fig. 5.28 and Fig. 5.29, including four boost inductor currents ($i_{L1} \sim i_{L4}$) and two output capacitor voltages ($v_{C1} \sim v_{C2}$). The SPS model is executed with the same parameters in Simulink, and the results are also presented. Consistent system-level results are obtained by the quasi-transient high-resolution model compared to those of SPS model.

5.7.2.4 Device-level simulation results

The transient waveforms of IGBT during turn-on and turn-off are depicted in Fig. 5.30. It can be observed that the model can produce consistent waveforms compared with the offline Saber simulation reference results. However, some differences can be noticed since the proposed model is simplified for real-time simulation purposes and the waveforms are approximated as linear and exponential

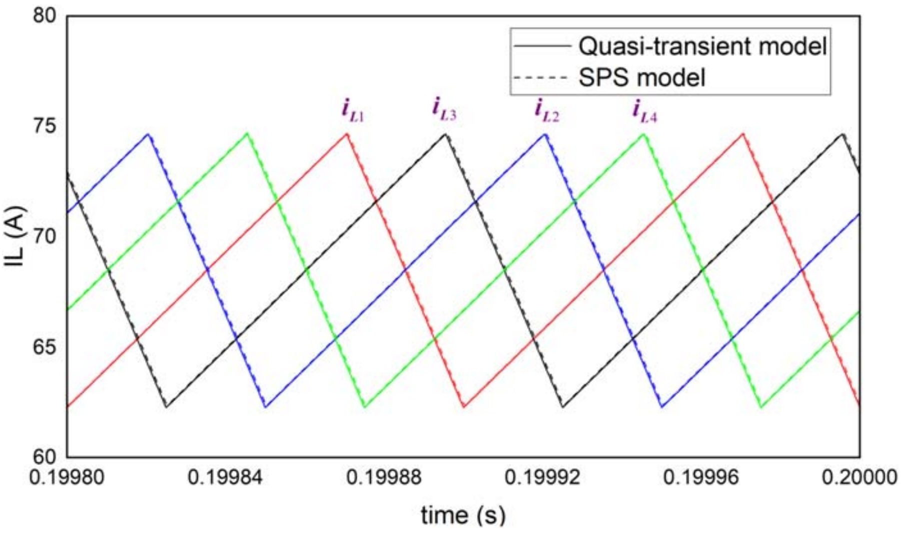


Figure 5.28 FIBC simulation results of inductor currents.

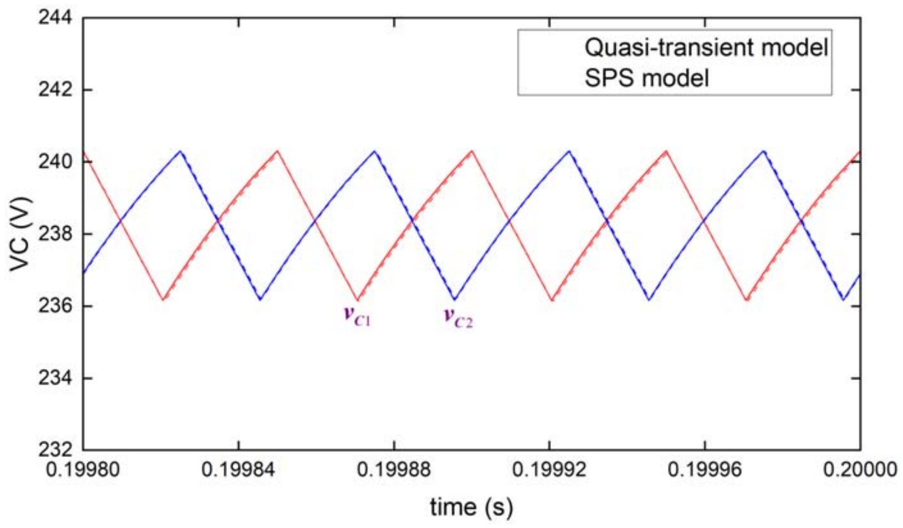


Figure 5.29 FIBC system-level simulation results of capacitor voltages.

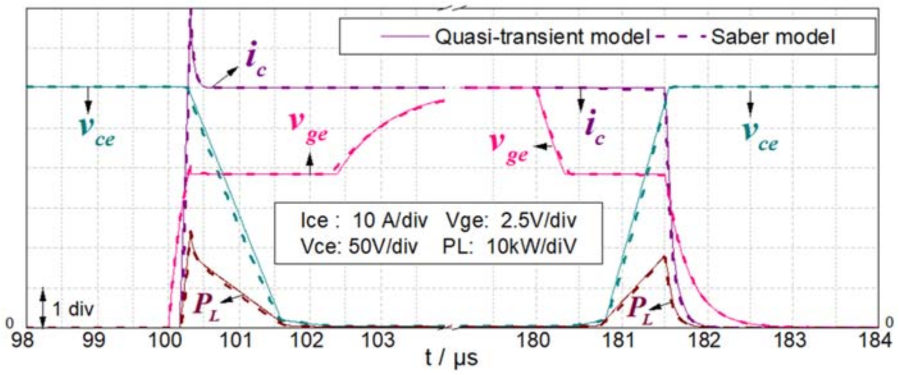


Figure 5.30 Device-level simulation results.

variations in each stage. The essential switching information is calculated based on Fig. 5.30 and is listed in Table 5.5.

5.8 Electrothermal converter model

During the operations of semiconductors the heat produced at the junction is conducted outward through the die bond and the case. Meanwhile, some of the heat will be kept in the device. Therefore the device exhibits impedance to the heat flow, which is time-dependent.

Table 5.5 Switching information in transient waveforms.

Items	HRQTM	Saber	Items	HRQTM	Saber
$t_{d(on)}$	143 ns	163 ns	I_{rr}	18.3 A	20.1 A
t_r	92 ns	82 ns	$P^{on}_{L(\max)}$	23.5 kW	24.1 kW
$t_{d(off)}$	1441 ns	1430 ns	$P^{off}_{L(\max)}$	18.0 kW	17.7 kW
t_f	419 ns	391 ns	/	/	/

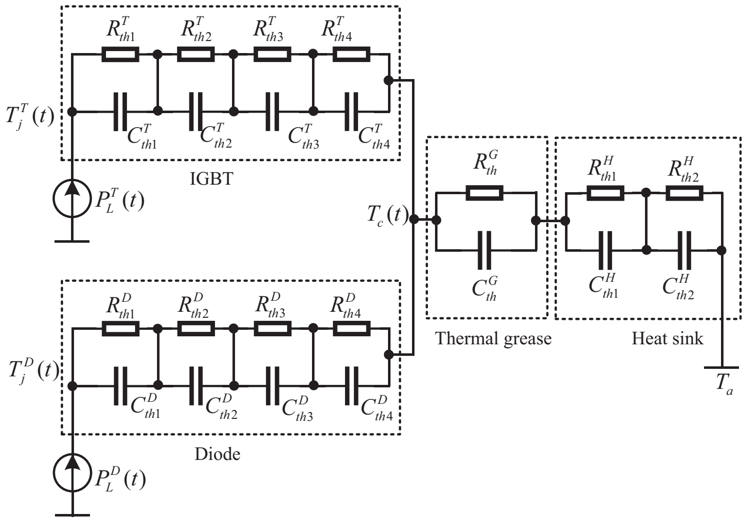


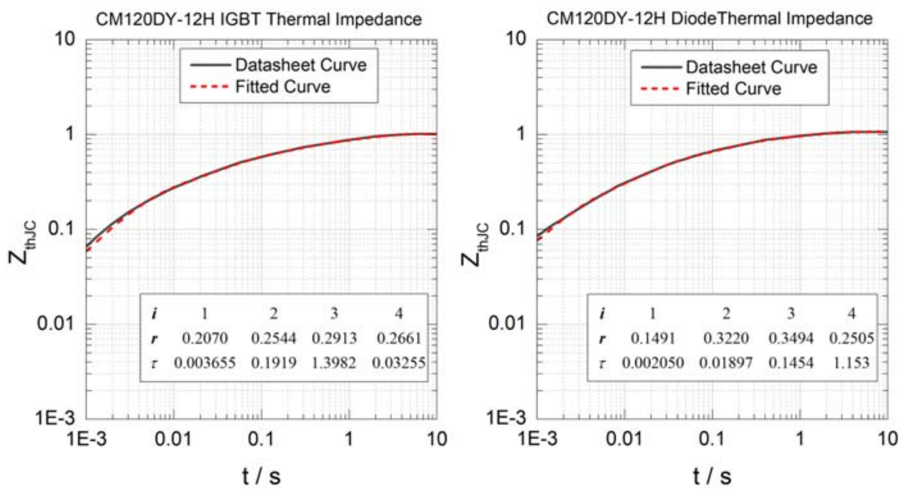
Figure 5.31 Foster model of transient thermal impedance.

Fig. 5.31 depicts the equivalent circuit model of the device thermal impedance, where P_L is the IGBT/diode power loss, T_j is the junction temperature, T_c is the case temperature, and T_a is the ambient temperature. The thermal impedance from case to ambient, such as that of thermal grease and heat sink, can be merged after the node of T_c in Fig. 5.31.

The thermal impedance of the model can be thus derived as (5.79), where, $\tau_i = R_i C_i$.

$$Z_{th}(t) = \sum_{i=1}^{n} R_i\left(1 - e^{-\frac{1}{\tau_i}t}\right) \tag{5.79}$$

The transient thermal impedance curves provided by the device datasheet can be credible since it is tested experimentally before being put on the market. Therefore the parameters of the Foster model are extracted by fitting the transient thermal impedance curves from the datasheet with four items of (5.79). The fitting results are given in Fig. 5.32.



i	1	2	3	4
r	0.2070	0.2544	0.2913	0.2661
τ	0.003655	0.1919	1.3982	0.03255

i	1	2	3	4
r	0.1491	0.3220	0.3494	0.2505
τ	0.002050	0.01897	0.1454	1.153

Figure 5.32 The curve fitting of the transient thermal impedance.

The Foster model in Fig. 5.31 can be discretized by the backward Euler method. The junction temperature of IGBT and the diode can be thus obtained by (5.80).

$$T_j(t) = T_c(t) + \sum_{i=1}^{4} \left[M_{(i)} P_L(t) + N_{(i)} v_{Cth(i)}(t - \Delta t) \right] \tag{5.80}$$

where $M_{(i)} = \frac{hR_{th(i)}}{h + \tau_{th(i)}}$, $N_{(i)} = \frac{\tau_{th(i)}}{h + \tau_{th(i)}}$.

The power loss leads to a temperature variation of the junction, which in turn affects the characteristics of the IGBT. The IGBT static model and the transfer characteristics are affected by the variation of junction temperatures. However, the typical characteristics related to junction temperature are usually given in the datasheet at two temperatures, ambient temperature and the maximum operating temperature. The characteristics of other junction temperatures are modeled as a linear interpolation between the top and bottom boundaries, as given in (5.81). Therefore the electrical characteristics interact with thermal behaviors, and the electrothermal converter model can be developed.

$$X^{(T_j)} = \frac{X^{(T_{\max})} - X^{(T_{\min})}}{T_{\max} - T_{\min}} \left(T_j - T_{\min} \right) + X^{(T_{\min})} \tag{5.81}$$

To perform the electrothermal simulation, the electric model presented in Section 5.7 will be used in combination with the thermal model. By doing the electrothermal simulation of FIBC converter the junction temperature variation of IGBT is depicted in Fig. 5.33 and compared to those from Saber simulation results over 0.5 s simulation time. Three switching frequencies are used in the simulation, 5, 10, and 20 kHz. The switching power losses are more significant with the higher

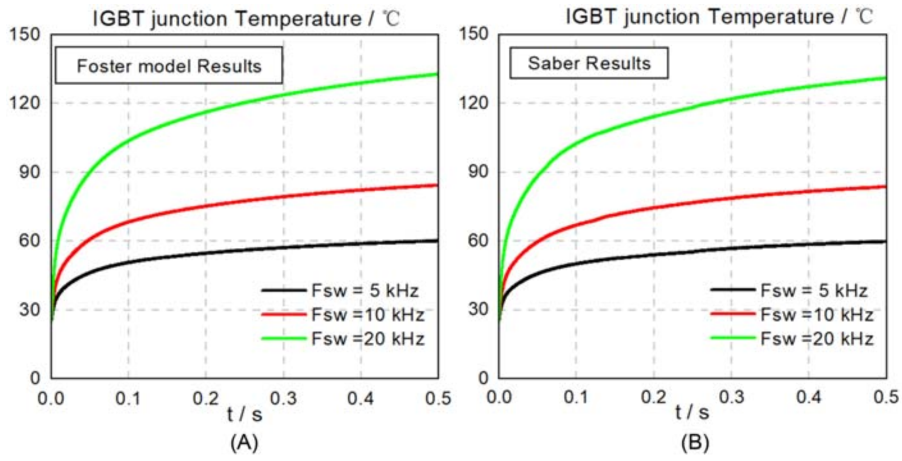


Figure 5.33 IGBT junction temperature waveforms.

switching frequency, which leads to a higher junction temperature. As can be seen clearly from the figure the electrothermal model can predict the thermal behaviors consistently compared with the reference model.

References

[1] F. Gao, Methods for Ordinary Differential Equations System—How to Configure Solvers in Matlab-Simulink, Report, University of Technology of Belfort-Montbeliard, 2016.

[2] G. Qiu, Electric Circuit, fifth ed., Higher Education Press, Beijing, 2006 (in Chinese) (Chapter 3).

[3] H.F. Blanchette, T. Ould-Bachir, J.P. David, A state-space modeling approach for the FPGA-based real-time simulation of high switching frequency power converters, IEEE Transactions on Industrial Electronics 59 (12) (2012) 4555−4567.

[4] H. Bai, C. Liu, S. Zhuo, R. Ma, D. Paire, F. Gao, FPGA-based device-level electro-thermal modeling of floating interleaved boost converter for fuel cell hardware-in-the-loop applications, IEEE Transactions on Industry Applications 55 (5) (2019) 5300−5310.

[5] H. Bai, Q. Li, H. Luo, Y. Huangfu, F. Gao, Nonlinear behavioral modeling and real-time simulation of electric propulsion system for the high-fidelity X-in-the-loop applications, IEEE Transactions on Transportation Electrification (2022).

Numerical solver of power electronic converter models

6

6.1 Introduction

After modeling the components as well as the network of power electronic converters, the numerical solver of the converter model is ascribed to solving a system of linear equations. For networks modeled using the nodal analysis method, the nodal voltage equation must be solved to get the nodal voltage and further the branch voltage and current. For networks modeled by state-space equations, the discretized state-space equations must be solved to get the values of the state variables and further the output variables. In this chapter, we will focus on the methods to solve a system of linear algebraic equations. For clarity, the system of linear algebraic equations is represented as $\mathbf{Ax} = \mathbf{b}$ in this chapter.

The speed of solving $\mathbf{Ax} = \mathbf{b}$ will dominate the computing time of the simulation in one time step. Based on the knowledge from linear algebra, $\mathbf{Ax} = \mathbf{b}$ can be solved by direct methods such as Gaussian elimination method and LU decomposition to get the analytical results. Meanwhile, based on the knowledge from numerical analysis methods, iterative methods such as the Gauss−Seidel iteration can also be used to find the approximated results. However, both direct and iterative methods may be time-consuming and not suitable for the power electronics real-time simulation, since the simulation time step can be down to the nanosecond level. For this reason the precomputing method is usually adopted, where all the possible values of $\mathbf{A}^{-1}$ are precomputed and stored in the memory before the simulation starts. The real time calculation speed is improved but at the cost of memory usage since the number of matrices can be extremely high when numerous switches are involved. In addition, an inversion updating method can be used to update $\mathbf{A}^{-1}$ without redoing the entire inversion of $\mathbf{A}$ when switching events happens, which could reduce the computing complexity.

To speed up the process of solving $\mathbf{Ax} = \mathbf{b}$, on one hand, the parallel solver can be used to speed up the computation. The related technologies are massively studied when encountering a sparse matrix $\mathbf{A}$ for the simulation of a very large-scale circuit [1,2]. On the other hand, if the network is naturally modeled in parallel, parallel computing can be easily implemented. In this chapter, we will explore the paralleled network modeling method including the equivalent network partitioning method and the delay-based network decoupling method. By splitting the electric network into several subcircuits, the scale of $\mathbf{Ax} = \mathbf{b}$ can be significantly reduced while the subcircuits can be computed in parallel. The computing time can be reduced, which gives the possibility of simulating high-switching frequency converters and integrating more complex power electronic models in the real time simulation.

When the power electronic circuit contains nonlinear components, the iterative solver should be used to approximate the analytical solutions. In this chapter the nonlinear electric network-solving strategy using the Newton–Raphson (N-R) method is also presented. The nonlinear components are linearized and represented by the companion circuit model in each N-R iteration and inserted in the global nodal analysis. When the error criterion is met or the maximum iteration number is reached, the iteration will stop and the solution of the current time step will be found. The iterative solving process is not suitable for the short-time-step real-time simulation; in the state-of-the-art real-time simulation of power electronics, the nonlinearities in the circuit network can be solved by the lookup-table (LUT)-based method. A DC-DC-AC topology is selected as an example in this chapter to illustrate the parallel network modeling and its implementation process. Both the system-level simulation and device-level simulation are demonstrated, and the results are evaluated by the commercial simulation tool.

6.2 Numerical solutions of linear algebraic equations

6.2.1 Direct method

6.2.1.1 Gaussian elimination method

Consider the system of linear equations presented in Eq. (6.1) or in the matrix format in Eq. (6.2), where **A** is nonsingular. The system can be represented by the augmented matrix representation shown in Eq. (6.3).

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = a_{1,n+1} \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = a_{2,n+1} \\ \qquad\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = a_{n,n+1} \end{cases}. \tag{6.1}$$

$$\mathbf{Ax} = \mathbf{b} \tag{6.2}$$

where $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \mathbf{b} = \begin{pmatrix} a_{1,n+1} \\ a_{2,n+1} \\ \vdots \\ a_{n,n+1} \end{pmatrix}$.

$$(\mathbf{A}|\mathbf{b}) = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n1} & \cdots & a_{nn} & a_{n,n+1} \end{array}\right) = \left(\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & a_{2,n+1}^{(1)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & a_{n,n+1}^{(1)} \end{array}\right). \tag{6.3}$$

The basic idea of the Gaussian elimination method is to perform the elementary row operation of $(\mathbf{A}|\mathbf{b})$ to make the entries below the diagonal line equal zero. The solution of the linear equations will then transform into that of an upper diagonal matrix. The procedure of the Gaussian elimination method is demonstrated as follows:

Step 1: Assuming $a_{11}^{(1)} \neq 0$, define $l_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, (i = 2, 3, \ldots, n)$. Add row 1 multiplied by $-l_{i1}$ to row i ($i = 2,3,\ldots,n$).

$$a_{ij}^{(2)} = a_{ij}^{(1)} - \frac{a_{i1}^{(1)}}{a_{11}^{(1)}} a_{1j}^{(1)}, (j = 2, 3, \ldots, n + 1). \tag{6.4}$$

The augmented matrix $(\mathbf{A}|\mathbf{b})$ will then be transformed to Eq. (6.5).

$$(\mathbf{A}|\mathbf{b}) = \left(\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ & \cdots & \cdots & \cdots & \cdots \\ & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & a_{n,n+1}^{(2)} \end{array}\right). \tag{6.5}$$

Step 2: Assuming $a_{22}^{(2)} \neq 0$, define $l_{i2} = \frac{a_{i2}^{(2)}}{a_{22}^{(2)}}, (i = 3, 4, \ldots, n)$. Add row 2 multiplied by $-l_{i2}$ to row i ($i = 3,4,\ldots,$n).

$$a_{ij}^{(3)} = a_{ij}^{(2)} - \frac{a_{i2}^{(2)}}{a_{22}^{(2)}} a_{2j}^{(2)}, (j = 3, 4, \ldots, n + 1). \tag{6.6}$$

The augmented matrix $(\mathbf{A}|\mathbf{b})$ will then be transformed to

$$(\mathbf{A}|\mathbf{b}) = \left(\begin{array}{ccccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ & & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & a_{3,n+1}^{(3)} \\ & & \vdots & \cdots & \cdots & \cdots \\ & & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} & a_{n,n+1}^{(3)} \end{array}\right). \tag{6.7}$$

Step n: After finishing steps 3 to n-1 the augmented matrix will be finally transformed to Eq. (6.8), which means that the linear equations in Eq. (6.1) will be equivalently transformed to the upper diagonal equations shown in Eq. (6.9).

$$(\mathbf{A}|\mathbf{b}) = \left(\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} & a_{1,n+1}^{(1)} \\ & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & a_{2,n+1}^{(2)} \\ & & \ddots & \cdots & \cdots \\ & & & a_{nn}^{(n)} & a_{n,n+1}^{(n)} \end{array}\right). \tag{6.8}$$

$$\begin{cases} a_{11}^{(1)}x_1 + a_{12}^{(1)}x_2 + \cdots\cdots + a_{1n}^{(1)}x_n = a_{1,n+1}^{(1)} \\ a_{22}^{(2)}x_2 + \cdots\cdots + a_{2n}^{(2)}x_n = a_{2,n+1}^{(2)} \\ a_{33}^{(3)}x_3 + \cdots + a_{3n}^{(3)}x_n = a_{3,n+1}^{(3)} \\ \cdots\cdots \\ a_{nn}^{(n)}x_n = a_{n,n+1}^{(n)} \end{cases}. \tag{6.9}$$

Finally, Eq. (6.9) can be performed to get the solution of the original linear equations, as given in Eq. (6.10).

$$\begin{cases} x_n = \dfrac{a_{n,n+1}^{(n)}}{a_{nn}^{(n)}} \\ x_k = \dfrac{1}{a_{kk}^{(k)}}\left[a_{k,n+1}^{(k)} - \displaystyle\sum_{j=k+1}^{n} a_{kj}^{(k)}x_j\right], k = n-1, n-2, \cdots, 1 \end{cases}. \tag{6.10}$$

If matrix A is of n order, the number of the multiplication/division operations in the Gaussian elimination method is $\frac{n^3}{3} + n^2 - \frac{n}{3} \approx \frac{n^3}{3}$, and the number of add/subtract operations is $\frac{n(n-1)(2n+5)}{6} \approx \frac{n^3}{3}$.

It should be noted that the precondition of using the Gaussian elimination method is that the leading principle minor of A for any order is nonzero.

6.2.1.2 LU decomposition method

If matrix **A** can be decomposed by a lower triangular matrix **L** and an upper triangular matrix **U**, where **A** = **LU**, **Ax** = **b** will be equivalent to Eq. (6.11)

$$\mathbf{Ax} = \mathbf{b} \Leftrightarrow \mathbf{LUx} = \mathbf{b} \Leftrightarrow \begin{cases} \mathbf{Ux} = \mathbf{y} \\ \mathbf{Ly} = \mathbf{b} \end{cases}. \tag{6.11}$$

The solution of **Ax** = **b** can be performed by first solving **Ly** = **b** to get $y_1, y_2, \ldots, y_n$ as in Eq. (6.12)

$$\begin{pmatrix} 1 & & & \\ l_{21} & 1 & & \\ \cdots & \cdots & \cdots & \\ l_{n1} & l_{n2} & & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{pmatrix} \Rightarrow \begin{cases} y_1 = b_1 \\ y_k = b_k - \displaystyle\sum_{m=1}^{k-1} l_{km}y_m, k = 2, 3, \ldots, n \end{cases}. \tag{6.12}$$

and then solving **Ux** = **y** to get $x_n, x_{n-1}, \ldots, x_1$, as in Eq. (6.13).

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \cdots & \cdots \\ & & & u_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{pmatrix} \Rightarrow$$

$$\begin{cases} x_n = \dfrac{y_n}{u_{nn}} \\ x_k = \dfrac{1}{u_{kk}}(y_k - \displaystyle\sum_{m=k+1}^{n} u_{km}x_m), k = n-1, n-2, \cdots, 1 \end{cases}. \tag{6.13}$$

In this case, **L** is a unit lower triangular matrix while **U** is an upper triangular matrix. Such a decomposition **A** = **LU** is called Doolittle decomposition. Actually, **A** = **LU** may have another form where **L** is a lower triangular matrix while **U** is a unit upper triangular matrix, which is called Crout decomposition. If A is an invertible matrix, the LU decomposition of **A** (for both Doolittle and Crout decomposition) will exist only if the leading principle minor of **A** for any order is nonzero.

The decomposition of **A** = **LU** in Eq. (6.14) can be performed by the following procedures:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & & & \ddots & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix}. \tag{6.14}$$

Step 1: Observing the first row, $a_{1j} = u_{1j}$; the first row of **U** is obtained by

$$u_{1j} = a_{1j}, (j = 1, 2, \ldots, n). \tag{6.15}$$

Observing the first column, $a_{i1} = l_{i1}u_{11}$, and the first column of **L** is obtained by

$$l_{i1} = \frac{a_{i1}}{u_{11}}, (i = 2, 3, \ldots, n). \tag{6.16}$$

Step k (*k* = 2,3,. . .*n*)

Let us assume that the columns from 1 to *k*-1 of **L** and the rows from 1 to *k*-1 of **U** are obtained.

Observing the *k*th row of **A** in Eq. (6.17),

$$a_{kj} = \sum_{m=1}^{n} l_{km}u_{mj} = \sum_{m=1}^{k-1} l_{km}u_{mj} + l_{kk}u_{kj} + \sum_{m=k+1}^{n} l_{km}u_{mj} = \sum_{m=1}^{k-1} l_{km}u_{mj} + u_{kj}. \tag{6.17}$$

row k of **U** is obtained by Eq. (6.18)

$$u_{kj} = a_{kj} - \sum_{m=1}^{k-1} l_{km} u_{mj}, \quad (j = k, k+1, \cdots, n). \tag{6.18}$$

Observing the *k*th column of **A** in Eq. (6.19),

$$a_{ik} = \sum_{m=1}^{n} l_{im} u_{mk} = \sum_{m=1}^{k-1} l_{im} u_{mk} + l_{ik} u_{kk} + \sum_{m=k+1}^{n} l_{im} u_{mk} = \sum_{m=1}^{k-1} l_{im} u_{mk} + l_{ik} u_{kk}. \tag{6.19}$$

Column *k* of **L** is obtained by Eq. (6.20)

$$l_{ik} = \frac{1}{u_{kk}} \left(a_{ik} - \sum_{m=1}^{k-1} l_{im} u_{mk} \right). \tag{6.20}$$

By combing Eqs. (6.18) and (6.20) the data produced in the progressive solution of **L** and **U** can be stored in a compact manner as shown in Fig. 6.1, which can save the memory cost in the computing procedure in the processor.

After obtaining **L** and **U**, **Ly = b** is solved recursively by Eq. (6.20),

$$\begin{cases} y_1 = b_1 \\ y_k = b_k - \sum\limits_{m=1}^{k-1} l_{km} y_m k = 2, 3, \cdots, n \end{cases}. \tag{6.21}$$

Ux = y is solved recursively by Eq. (6.21).

$$\begin{cases} x_1 = \dfrac{y_n}{u_{nn}} \\ x_k = \left(y_k - \sum\limits_{j=k+1}^{n} u_{kj} x_j \right) / u_{kk} k = n-1, n-2, \ldots, 1 \end{cases}. \tag{6.22}$$

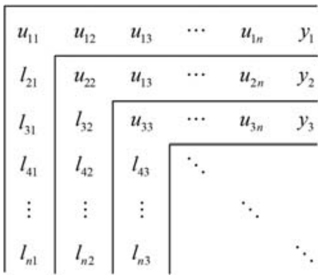


Figure 6.1 The compact storing format for L and U.

It can be proved that the Gaussian elimination method is used to transform **A** into an upper triangular matrix $\mathbf{U}=\mathbf{L}^{-1}\mathbf{A}$ and transform **b** into $\mathbf{y}=\mathbf{L}^{-1}\mathbf{b}$. The operations of the Gaussian elimination method can be treated as the same as the LU decomposition. Therefore the number of multiplication/division operations in both methods is similar.

6.2.1.3 Cholesky decomposition method

If **A** is a symmetric positive definite matrix, the LU decomposition can be accomplished in a simpler manner $\mathbf{A}=\mathbf{L}\mathbf{L}^{\mathbf{T}}$ where **L** is a nonsingular lower triangular matrix. The $\mathbf{A}=\mathbf{L}\mathbf{L}^{\mathbf{T}}$ given in Eq. (6.23) is called the Cholesky decomposition of **A**

$$\mathbf{A}=\mathbf{L}\mathbf{L}^{\mathbf{T}}=\begin{bmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & & \ddots & \\ l_{n1} & & \cdots & l_{nn} \end{bmatrix}\begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ & l_{22} & & \\ & & \ddots & \\ & & & l_{nn} \end{bmatrix}. \tag{6.23}$$

Similar to the procedure in LU decomposition, matrix multiplication is used to reversely derive the entries in **L**.

Step 1

Observing the first column of **A**, where $a_{11}=l_{11}^2$, $a_{i1}=l_{11}\cdot l_{i1}$

The first column of **L** is obtained by

$$\begin{cases} l_{11}=\sqrt{a_{11}} \\ l_{i1}=\dfrac{a_{i1}}{l_{11}}, i=2,3,\ldots,n\cdot \end{cases} \tag{6.24}$$

Step k ($k=2,3,\ldots n$)

Considering that columns 1 to k-1 of **L** are obtained, observing the kth column of **A** shown in Eq. (6.25),

$$a_{ik}=\sum_{m=1}^{n}l_{im}l_{km}=\sum_{m=1}^{k-1}l_{im}l_{km}+l_{ik}l_{kk}+\sum_{m=k+1}^{n}l_{im}l_{km}=\sum_{m=1}^{k-1}l_{im}l_{km}+l_{ik}l_{kk}. \tag{6.25}$$

The kth column of **L** can be obtained by Eq. (6.26).

$$\begin{cases} a_{kk}=\sum_{m=1}^{k-1}l_{km}^2+l_{kk}^2\Rightarrow l_{kk}=\sqrt{a_{kk}-\sum_{m=1}^{k-1}l_{km}^2}, & i=k \\ a_{ik}=\sum_{m=1}^{k-1}l_{im}l_{km}+l_{ik}l_{kk}\Rightarrow l_{ik}=\left(a_{ik}-\sum_{m=1}^{k-1}l_{im}l_{km}\right)/l_{kk}, & i>k, i=k+1,k+2,\cdots,n \end{cases}. \tag{6.26}$$

The number of multiplication/division operations in Cholesky decomposition is about $\frac{1}{6}n^3$, which is less than that in LU decomposition.

It should be noted that the nodal analysis will formulate a system of linear equations where the admittance matrix $\mathbf{Y}$ is symmetric and usually regarded to be positive definite in the electric network analysis, which is suitable for using the Cholesky decomposition method in the circuit simulation.

6.2.2 Iterative methods

To solve $\mathbf{Ax}=\mathbf{b}$, we can also construct an iterative formula to get approximated solutions. If the equivalent is made to transform $\mathbf{Ax}=\mathbf{b}$ into $\mathbf{x}=\mathbf{Bx}+\mathbf{g}$, we can construct an iterative formula as shown in Eq. (6.27).

$$\mathbf{x}^{(\mathbf{k+1})}=\mathbf{Bx}^{(\mathbf{k})}+\mathbf{g}, k=0,1,\ldots. \tag{6.27}$$

where $\mathbf{x}^{(\mathbf{0})}$ is the initial vector, and Eq. (6.27) will produce a sequence $\{\mathbf{x}^{(k)}\}$. If $\{\mathbf{x}^{(k)}\}$ converges to a vector $\mathbf{x}^*$, namely, $\mathbf{x}^*=\mathbf{Bx}^*+\mathbf{g}$, $\mathbf{x}^*$ is the solution of the equation $\mathbf{Ax}=\mathbf{b}$. We can say that the iterative formula $\mathbf{x}^{(\mathbf{k+1})}=\mathbf{Bx}^{(\mathbf{k})}+\mathbf{g}, k=0,1,\ldots$ converges with regard to the initial vector $\mathbf{x}^{(\mathbf{0})}$.

The transformation from $\mathbf{Ax}=\mathbf{b}$ to $\mathbf{x}=\mathbf{Bx}+\mathbf{g}$ is not unique. Different iterative formulas can be either convergent or not. The way to constitute the iterative formula with convergency, the convergency speed, and errors are the important things to consider when using the iterative method to solve linear equations.

6.2.2.1 Gauss–Seidel iteration method

Consider an iterative formula $\mathbf{x}^{(\mathbf{k+1})}=\mathbf{Bx}^{(\mathbf{k})}+\mathbf{g}, k=0,1,\ldots$, if $\mathbf{B}$ can be decomposed by $\mathbf{B}=\mathbf{B_1}+\mathbf{B_2}$, where

$$\mathbf{B}=\mathbf{B_1}+\mathbf{B_2}=\begin{pmatrix} 0 & & & \\ b_{21} & 0 & & \\ \vdots & & \ddots & \\ b_{n1} & b_{n2} & \cdots & 0 \end{pmatrix}+\begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ & b_{22} & \cdots & b_{2n} \\ & & \ddots & \vdots \\ & & & b_{nn} \end{pmatrix}. \tag{6.28}$$

the iterative formula will be

$$\mathbf{x}^{(\mathbf{k+1})}=\mathbf{B}_1\mathbf{x}^{(\mathbf{k})}+\mathbf{B}_2\mathbf{x}^{(\mathbf{k})}+\mathbf{g}, k=0,1,\ldots. \tag{6.29}$$

The formula written regarding each component of the vector is

$$\begin{cases} x_1^{(k+1)} = b_{11}x_1^{(k)} + b_{12}x_2^{(k)} + b_{13}x_3^{(k)} + \cdots + b_{1n}x_n^{(k)} + g_1 \\ x_2^{(k+1)} = b_{21}x_1^{(k)} + b_{22}x_2^{(k)} + b_{23}x_3^{(k)} + \cdots + b_{2n}x_n^{(k)} + g_2 \\ x_3^{(k+1)} = b_{31}x_1^{(k)} + b_{32}x_2^{(k)} + b_{33}x_3^{(k)} + \cdots + b_{3n}x_n^{(k)} + g_3 \\ \cdots \\ x_n^{(k+1)} = b_{n1}x_1^{(k)} + b_{n2}x_2^{(k)} + b_{n3}x_3^{(k)} + \cdots + b_{n,n}x_n^{(k)} + g_n \end{cases}, k = 0, 1, 2, \ldots.. \tag{6.30}$$

If the newly updated values $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$ are used in the computing $x_i^{(k+1)}$, where $1 < i \leq n$ as shown in Eq. (6.31),

$$\begin{cases} x_1^{(k+1)} = b_{11}x_1^{(k)} + b_{12}x_2^{(k)} + b_{13}x_3^{(k)} + \cdots + b_{1n}x_n^{(k)} + g_1 \\ x_2^{(k+1)} = b_{21}x_1^{(k+1)} + b_{22}x_2^{(k)} + b_{23}x_3^{(k)} + \cdots + b_{2n}x_n^{(k)} + g_2 \\ x_3^{(k+1)} = b_{31}x_1^{(k+1)} + b_{32}x_2^{(k+1)} + b_{33}x_3^{(k)} + \cdots + b_{3n}x_n^{(k)} + g_3 \\ \cdots \\ x_n^{(k+1)} = b_{n1}x_1^{(k+1)} + b_{n2}x_2^{(k+1)} + b_{n3}x_3^{(k+1)} + \cdots + b_{n,n}x_n^{(k)} + g_n \end{cases}, k = 0, 1, 2, \ldots. \tag{6.31}$$

the iterative formula can be updated to Eq. (6.32), which represents the Gauss−Seidel iteration method

$$\mathbf{x}^{(\mathbf{k}+1)} = \mathbf{B}_1\mathbf{x}^{(\mathbf{k}+1)} + \mathbf{B}_2\mathbf{x}^{(\mathbf{k})} + \mathbf{g}, k = 0, 1, \ldots. \tag{6.32}$$

6.2.2.2 *Jacobi iteration method*

For a system of linear equations in Eq. (6.1), if we transform it to Eq. (6.33),

$$\begin{cases} x_1 = \dfrac{1}{a_{11}}(-a_{12}x_2 - a_{13}x_3 - \cdots - a_{1n}x_n + b_1) \\ x_2 = \dfrac{1}{a_{22}}(-a_{21}x_1 - a_{23}x_3 - \cdots - a_{2n}x_n + b_2) \\ \cdots \\ x_n = \dfrac{1}{a_{nn}}(-a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1} + b_n) \end{cases}. \tag{6.33}$$

The iterative formula can be constituted by Eq. (6.34), and it is called the Jacobi iteration formula.

$$\begin{cases} x_1^{(k+1)} = \dfrac{1}{a_{11}}(-a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)} + b_1) \\ x_2^{(k+1)} = \dfrac{1}{a_{22}}(-a_{21}x_1^{(k)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_n^{(k)} + b_2) \\ \cdots \\ x_2^{(k+1)} = \dfrac{1}{a_{nn}}(-a_{n1}x_1^{(k)} - a_{n2}x_2^{(k)} - \cdots - a_{n,n-1}x_{n-1}^{(k)} + b_n) \end{cases} ,k = 0, 1, 2, \cdots \tag{6.34}$$

The simplified representation is

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i}^{j=1^n} a_{ij}x_j^{(k)}\right) \tag{6.35}$$

where $i = 1, 2, \cdots, n$ and $k = 0, 1, 2, \ldots.$

Consider $\mathbf{A = D + L + U}$ shown in Eq. (6.36), where $\mathbf{L}$ is the upper triangular matrix, $\mathbf{U}$ is lower triangular matrix, and $\mathbf{D}$ is the diagonal matrix,

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & \ddots & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & 0 & a_{nn} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ a_{21} & 0 & \ddots & \ddots & 0 \\ a_{31} & a_{32} & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ 0 & \ddots & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{6.36}$$

$\mathbf{Ax = b}$ can be equivalent to $\mathbf{(L + D + U)x = b}$. Therefore the Jacobi iterative formula in matrix can be expressed by Eq. (6.37)

$$\mathbf{x}^{(k+1)} = -\mathbf{D}^{-1}(\mathbf{L + U})\mathbf{x}^k + \mathbf{D}^{-1}\mathbf{b} \quad k = 0, 1, 2, \ldots. \tag{6.37}$$

If we make $\mathbf{B_J} = -\mathbf{D}^{-1}(\mathbf{L + U})$ and $\mathbf{g = D}^{-1}\mathbf{b}$, $\mathbf{B_J}$ is the iteration matrix of the Jacobi iterative method. The Jacobi iterative formula is further expressed by Eq. (6.38)

$$\mathbf{x}^{(k+1)} = \mathbf{B_J}\mathbf{x}^k + \mathbf{g} \quad k = 0, 1, 2, \ldots. \tag{6.38}$$

6.2.2.3 Gauss–Seidel iteration corresponding to the Jacobi iteration formula

Newly updated values $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$ are used in computing $x_i^{(k+1)}$, where $1 < i \le n$

The Jacobi iterative formula shown in Eq. (6.34) can be revised as (6.39) by using the newly updated values $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{i-1}^{(k+1)}$ in computing the rest component $x_i^{(k+1)}(1 < i \le n)$ in the same way as the Gauss–Seidel iteration method.

$$\begin{cases} x_1^{(k+1)} = \dfrac{1}{a_{11}}(-a_{12}x_2^{(k)} - a_{13}x_3^{(k)} - \cdots - a_{1n}x_n^{(k)} + b_1) \\ x_2^{(k+1)} = \dfrac{1}{a_{22}}(-a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)} - \cdots - a_{2n}x_{n-1}^{(k)} + b_2) \\ \cdots \\ x_n^{(k+1)} = \dfrac{1}{a_{nn}}(-a_{n1}x_1^{(k+1)} - a_{n2}x_2^{(k+1)} - \cdots - a_{n,n-1}x_{n-1}^{(k+1)} + b_n) \end{cases} , k = 0, 1, 2, \cdots. \tag{6.39}$$

We can thus get the Gauss–Seidel iterative formula corresponding to the Jacobi iteration as shown in Eqs. (6.39) and (6.40).

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right). \tag{6.40}$$

Reconsider $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$, $\mathbf{Ax} = \mathbf{b}$ can be equivalent to $\mathbf{Dx} = \mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x}$
The Gauss–Seidel iteration can thus be implemented by

$$\mathbf{x}^{(k+1)} = -\mathbf{D}^{-1}\mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{D}^{-1}\mathbf{U}\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{b} \quad k = 0, 1, 2, \ldots. \tag{6.41}$$

Thus

$$(\mathbf{D} + \mathbf{L})\mathbf{x}^{(k+1)} = -\mathbf{U}\mathbf{x}^{(k)} + \mathbf{b} \quad k = 0, 1, 2, \ldots. \tag{6.42}$$

The Gauss–Seidel iteration can be expressed by

$$\mathbf{x}^{(k+1)} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\mathbf{x}^{(k)} + (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b} \quad k = 0, 1, 2, \ldots. \tag{6.43}$$

If we make $\mathbf{B_S} = -(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}$ and $\mathbf{g_s} = (\mathbf{D}+\mathbf{L})^{-1}\mathbf{b}$, then we get

$$\mathbf{x}^{(k+1)} = \mathbf{B_S}\mathbf{x}^{(k)} + \mathbf{g_s} \quad k = 0,1,2,\ldots. \tag{6.44}$$

where $\mathbf{B_S}$ is the iteration matrix. It should be noted that in this book the Gauss–Siedel (G-S) iterative method refers to that corresponding to the Jacobi iterative formula.

6.2.2.4 *Successive over-relaxation method*

Revise the Gauss–Siedel iterative formula in Eq. (6.40) to a new formula as in Eq. (6.45),

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \ldots, n. \tag{6.45}$$

Denote the item as Eq. (6.46), and $r_i^{(k+1)}$ is in general nonzero.

$$r_i^{(k+1)} = b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}, \quad i = 1, 2, \ldots, n. \tag{6.46}$$

If we multiply a factor $\omega = 1$ by the remnant between $x_i^{(k+1)}$ and $x_i^{(k)}$, the successive over-relaxation (SOR) iterative formula is obtained by Eq. (6.47), where ω is the relaxation factor.

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \ldots, n. \tag{6.47}$$

Eq. (6.48) is another expression of SOR iteration. When $\omega = 1$ the SOR iterative formula is the G-S iterative formula. In other words the SOR iterative method can be regarded as the weighted average between the G-S method and the $x_i^{(k)}$.

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \ldots, n.. \tag{6.48}$$

Rewrite the formula as Eq. (6.49); we can derive the matrix representation of the SOR iteration.

$$a_{ii}x_i^{(k+1)} = (1-\omega)a_{ii}x_i^{(k)} + \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right). \tag{6.49}$$

Consider $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$; we can get

$$\mathbf{D}\mathbf{x}^{(k+1)} = (1-\omega)\mathbf{D}\mathbf{x}^{(k)} + \omega(\mathbf{b} - \mathbf{L}\mathbf{x}^{(k+1)} - \mathbf{U}\mathbf{x}^{(k)}). \tag{6.50}$$

$$(\mathbf{D} + \omega\mathbf{L})\mathbf{x}^{(k+1)} = ((1-\omega)\mathbf{D} - \omega\mathbf{U})\mathbf{x}^{(k)} + \omega\mathbf{b}. \tag{6.51}$$

Then,

$$\mathbf{x}^{(k+1)} = (\mathbf{D}+\omega\mathbf{L})^{-1}[(1-\omega)\mathbf{D} - \omega\mathbf{U}]\mathbf{x}^{(k)} + \omega(\mathbf{D}+\omega\mathbf{L})^{-1}\mathbf{b}. \tag{6.52}$$

If we make $\mathbf{L}_\omega = (\mathbf{D}+\omega\mathbf{L})^{-1}((1-\omega)\mathbf{D} - \omega\mathbf{U})$ and $g = \omega(\mathbf{D}+\omega\mathbf{L})^{-1}\mathbf{b}$, then the matrix representation of SOR will be given by

$$\mathbf{x}^{(k+1)} = \mathbf{L}_\omega\mathbf{x}^{(k)} + \mathbf{g}_\omega, \quad k = 0, 1, 2, \ldots. \tag{6.53}$$

where $\mathbf{L}_\omega$ is the iteration matrix of the successive over-relaxation method.

The iterative method to solve $\mathbf{Ax} = \mathbf{b}$ has to make sure that the computation is convergent to the neighborhood of the exact values. Therefore in the solving process the error tolerance is chosen as the criterion to stop the iteration. Once the absolute difference between successive iterations falls within the error tolerance, $\mathbf{x}^{(k+1)}$ can be treated as the approximated values of the linear equations and the iteration can be stopped. However, the iteration number should be confined in case the computation enters an endless loop.

In offline simulations the error tolerance can be chosen small enough to get accurate results and the number of iterations can be adjusted to get the desired speed of convergence. However, in real-time simulations, since the model computation time in each time step is limited and fixed, the feasible number of iterations is also limited. At the same time the limited number of iterations cannot always ensure the numerical accuracy of the approximated results. As a consequence the numerical errors in each simulation time step may become uncontrolled under the iterative method, which will lead to risks of simulation instability.

6.2.3 Precomputing method

For the real-time simulation of power electronics, where the simulation time step is in the microsecond level or nanosecond level, the direct method or the iterative method may not be a feasible choice.

To accommodate the low latency requirement of the simulations, $\mathbf{Ax} = \mathbf{b}$ can be realized by a precomputing method. The inversion of $\mathbf{A}$ is computed offline according to the topology and parameters of the simulated power converters. Due to the switching events, $\mathbf{A}^{-1}$ has in theory 2^n possibilities for a converter containing n switches when using the binary resistor model or ideal switch model. All the possible solutions of $\mathbf{A}^{-1}$ are stored in the memory unit in the real-time processor or FPGA and called up in the real-time simulation. One possible solution of $\mathbf{A}^{-1}$ will be selected in each time step based on the switch status and used for the solution of $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

When the associate discrete circuit (ADC) switch model is used, the conductance of the ADC model will not change with the switching event. Therefore matrix **A** will have constant entries which create a favorable condition to precompute $\mathbf{A}^{-1}$. $\mathbf{A}^{-1}$ thus will not change in the simulation, and the required memory will be reduced.

The precomputing method to solve $\mathbf{Ax} = \mathbf{b}$ is time-efficient and is well used in the FPGA-based real-time simulation. Very low real-time simulation time steps can be achieved (down to tens of nanoseconds). However, the precomputing of $\mathbf{A}^{-1}$ costs extra memory resources that will increase exponentially with the number of switches if the ideal switch model and binary resistor switch model are used. Besides, all achievable topologies of the simulated model should be defined before the implementation of the real-time simulation. As for the ADC model the precomputing of $\mathbf{A}^{-1}$ seems to be tailored, but the main deficiency comes from the accuracy of the ADC model itself, such as the virtual losses introduced by ADC models.

6.2.4 Inversion updating method

If the inversion $\mathbf{A}^{-1}$ can be updated when some of the entries in **A** change, the precomputed inversion can always be used to accommodate the topology change, and only the initial value of $\mathbf{A}^{-1}$ need to be stored.

The Sherman–Morrison–Woodbury (SMW) formula allows updating the inverse of a matrix **A** after a small perturbation from the inverse of the original matrix [3]. It will provide an alternative to solving $\mathbf{Ax} = \mathbf{b}$ which significantly reduces the needs in the storing the matrices compared to the precomputing method.

The SMW formulas is given by Eq. (6.54),

$$\mathbf{A}_{\mathbf{new}}^{-1} = \mathbf{A}_{\mathbf{old}}^{-1} - \mathbf{A}_{\mathbf{old}}^{-1}\mathbf{U}\mathbf{D}^{-1}\mathbf{V}\mathbf{A}_{\mathbf{old}}^{-1}. \tag{6.54}$$

where **A**, **U**, **D**, and **V** are matrices of sizes $n \times n$, $n \times d$, $d \times d$, and $d \times n$, respectively. d is equal to the number of columns of **A** which has been perturbed. **U** and **V** are computed based on perturbation in matrix **A** as Eq. (6.55).

$$\mathbf{UV} = \mathbf{A}_{\mathbf{new}} - \mathbf{A}_{\mathbf{old}}. \tag{6.55}$$

D can be computed by Eq. (6.56).

$$\mathbf{D} = \mathbf{I} + \mathbf{V}\mathbf{A}_{\mathbf{old}}^{-1}\mathbf{U}. \tag{6.56}$$

To compute $\mathbf{A}^{-1}$ using Eq. (6.54), $\mathbf{A}_{\mathbf{old}}^{-1}$ is known, **U** and **V** can be constructed based on Eq. (6.55), and only $\mathbf{D}^{-1}$ needs additional efforts to compute. When a small perturbation is involved the scale of **D**, $(d \times d)$ can be small, which is much easier to solve than the direct inversion of A. Therefore by using Eq. (6.54) the computing time can be reduced and a small simulation time step is prone be achieved.

However, the switching events cannot be predicted and the numbers of perturbed columns in **A** can be either small or large in different simulation time steps. Therefore when matrix **D** is of large scale, computational benefits of the inversion updating methods are diminished.

6.3 Electric network partitioning methods

Since solving Ax = b is either time-consuming or memory intensive, the paralleled computing is pursued to reduce the simulation latency. One feasible way to explore the parallel computation is to split the network into independent subcircuits and solve them simultaneously. Based on whether the solution of subcircuits are simultaneous, we can divide the parallel network modeling into equivalent network partitioning methods and delay-based network decoupling methods, as shown in Fig. 6.2

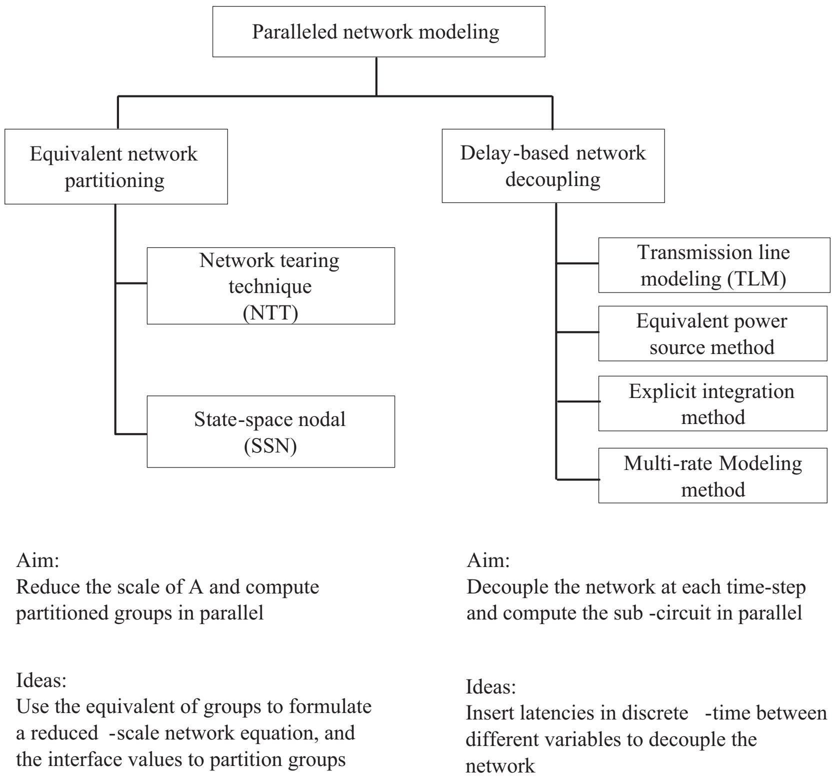

Paralleled network modeling
Equivalent network partitioning
Delay-based network decoupling
Network tearing technique (NTT)
State-space nodal (SSN)
Transmission line modeling (TLM)
Equivalent power source method
Explicit integration method
Multi-rate Modeling method
Aim:
Reduce the scale of A and compute partitioned groups in parallel
Aim:
Decouple the network at each time-step and compute the sub -circuit in parallel
Ideas:
Use the equivalent of groups to formulate a reduced -scale network equation, and the interface values to partition groups
Ideas:
Insert latencies in discrete -time between different variables to decouple the network

Figure 6.2 Paralleled network modeling methods.

The equivalent network partitioning tears a network into different subcircuits while providing a synchronous and stable solution for the entire network. The aim is to reduce the matrix scale in the network equation and compute partitioned subcircuits in parallel. The approach is to find the equivalents of the subcircuits and use them to formulate a reduced-scale network equation.

Two methods in the state of the art can be categorized into this type. One is the network tearing technique (NTT); the other is the state-space nodal (SSN) method.

6.3.1 Network tearing technique

Considering a network abstracted from a power electronic converter, the inductors and capacitors are all discretized and represented by their companion circuit models. The basic steps of the network tearing technique can be concluded as follows:

1. identify the subcircuits to be partitioned;
2. write the multiport equivalents of these subcircuits;
3. constitute the reduced network equations combining the multiport equivalents of partitioned groups and the remaindering network;
4. solve the network equations and compute the interface currents/voltages of the equivalents;
5. compute the subcircuits by using the newly computed interface variables.

In chapter 5, we have derived the formulation of an FIBC when the nodal analysis is applied. Let us still consider an FIBC converter as an example to demonstrate how the NTT is applied and how the real-time simulation can benefit from it. To recall, the switches are modeled by the binary resistor model, and the FIBC can be represented by the network shown in Fig. 6.3.

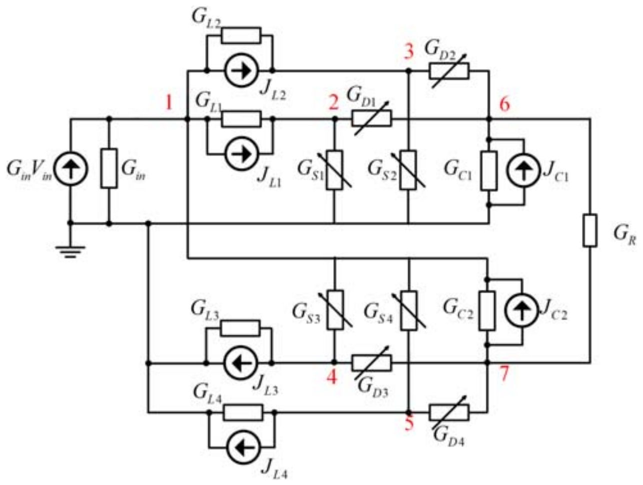


Figure 6.3 FIBC networks modeled by the companion circuit and binary resistor switch model.

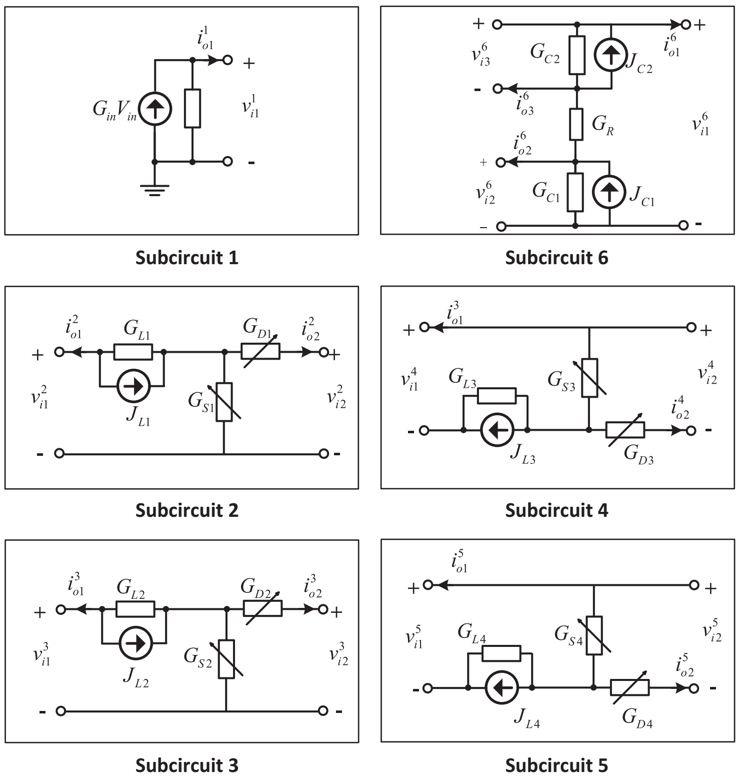


Figure 6.4 FIBC network partitioning into six subcircuits.

Step1 Select the subcircuits to be partitioned. Considering the physical composition of FIBC, the subcircuits to be partitioned are selected as the power source (subcircuit 1), the four switching units (subcircuits 2 to 5), and the output unit (subcircuit 6), as shown in Fig. 6.4.

Step2 Write the multiport hybrid equivalent (MPHE) of the subcircuits. Notice that the equivalent circuit can have the voltage port and current port simultaneously. A general formulation called multiport hybrid equivalent is shown in Eq. (6.57), combining the Norton equivalents and Thevenin equivalents to formulate the subcircuits in this step. $\mathbf{i_o}$ and $\mathbf{v_o}$ are vectors of output currents and voltages at the partitioned ports; $\mathbf{v_i}$ and $\mathbf{i_i}$ are vectors of imposed voltages and currents at these ports; $\mathbf{i_{eq}}$ and $\mathbf{v_{eq}}$ are vectors of equivalent current and voltage sources of the MPHE; $\mathbf{G_N}, \mathbf{R_{Th}}$ are matrices of equivalent conductance and resistance, respectively; $\mathbf{K_I}$ and $\mathbf{K_V}$ are coefficient matrices.

$$\begin{bmatrix} \mathbf{i_o} \\ \mathbf{v_o} \end{bmatrix} = \begin{bmatrix} \mathbf{i_{eq}} \\ \mathbf{v_{eq}} \end{bmatrix} - \begin{bmatrix} \mathbf{G_N} & \mathbf{K_I} \\ \mathbf{K_V} & \mathbf{R_{Th}} \end{bmatrix} \begin{bmatrix} \mathbf{v_i} \\ \mathbf{i_i} \end{bmatrix}. \tag{6.57}$$

In the subcircuits of FIBC, the port voltages are selected as the imposed voltages and the port currents are selected as the output currents of the subcircuit. Therefore the MPHEs of subcircuits can be formulated as follows:

Subcircuit 1:

$$i_{o1}^1 = i_{eq}^1 - g_{11}^1 v_{i1}^1 = V_{fc}G_{fc} - G_{fc}v_{i1}^1. \tag{6.58}$$

Subcircuit 2:

$$\begin{bmatrix} i_{o1}^2 \\ i_{o2}^2 \end{bmatrix} = \begin{bmatrix} i_{eq1}^2 \\ i_{eq2}^2 \end{bmatrix} - \begin{bmatrix} g_{11}^2 & g_{12}^2 \\ g_{21}^2 & g_{22}^2 \end{bmatrix} \begin{bmatrix} v_{i1}^2 \\ v_{i2}^2 \end{bmatrix}$$
$$= \begin{bmatrix} -\dfrac{J_{L1}(G_{S1}+G_{D1})}{G_{L1}+G_{S1}+G_{D1}} \\ \dfrac{J_{L1}G_{D1}}{G_{L1}+G_{S1}+G_{D1}} \end{bmatrix} - \begin{bmatrix} \dfrac{G_{L1}(G_{S1}+G_{D1})}{G_{L1}+G_{S1}+G_{D1}} & -\dfrac{G_{D1}G_{L1}}{G_{L1}+G_{S1}+G_{D1}} \\ -\dfrac{G_{D1}G_{L1}}{G_{L1}+G_{S1}+G_{D1}} & \dfrac{G_{D1}(G_{S1}+G_{L1})}{G_{L1}+G_{S1}+G_{D1}} \end{bmatrix} \begin{bmatrix} v_{i1}^2 \\ v_{i2}^2 \end{bmatrix}. \tag{6.59}$$

Subcircuit 3:

$$\begin{bmatrix} i_{o1}^3 \\ i_{o2}^3 \end{bmatrix} = \begin{bmatrix} i_{eq1}^3 \\ i_{eq2}^3 \end{bmatrix} - \begin{bmatrix} g_{11}^3 & g_{12}^3 \\ g_{21}^3 & g_{22}^3 \end{bmatrix} \begin{bmatrix} v_{i1}^3 \\ v_{i2}^3 \end{bmatrix}$$
$$= \begin{bmatrix} -\dfrac{J_{L2}(G_{S2}+G_{D2})}{G_{L2}+G_{S2}+G_{D2}} \\ \dfrac{J_{L2}G_{D2}}{G_{L2}+G_{S2}+G_{D2}} \end{bmatrix} - \begin{bmatrix} \dfrac{G_{L2}(G_{S2}+G_{D2})}{G_{L2}+G_{S2}+G_{D2}} & -\dfrac{G_{D2}G_{L2}}{G_{L2}+G_{S2}+G_{D2}} \\ -\dfrac{G_{D2}G_{L2}}{G_{L2}+G_{S2}+G_{D2}} & \dfrac{G_{D2}(G_{S2}+G_{L2})}{G_{L2}+G_{S2}+G_{D2}} \end{bmatrix} \begin{bmatrix} v_{i1}^3 \\ v_{i2}^3 \end{bmatrix}. \tag{6.60}$$

Subcircuit 4:

$$\begin{bmatrix} i_{o1}^4 \\ i_{o2}^4 \end{bmatrix} = \begin{bmatrix} i_{eq1}^4 \\ i_{eq2}^4 \end{bmatrix} - \begin{bmatrix} g_{11}^4 & g_{12}^4 \\ g_{21}^4 & g_{22}^4 \end{bmatrix} \begin{bmatrix} v_{i1}^4 \\ v_{i2}^4 \end{bmatrix}$$
$$= \begin{bmatrix} -\dfrac{J_{L3}(G_{S3}+G_{D3})}{G_{L3}+G_{S3}+G_{D3}} \\ -\dfrac{J_{L3}G_{D3}}{G_{L3}+G_{S3}+G_{D3}} \end{bmatrix} - \begin{bmatrix} \dfrac{G_{L3}(G_{S3}+G_{D3})}{G_{L3}+G_{S3}+G_{D3}} & -\dfrac{G_{D3}G_{L3}}{G_{L3}+G_{S3}+G_{D3}} \\ \dfrac{G_{D3}G_{L3}}{G_{L3}+G_{S3}+G_{D3}} & -\dfrac{G_{D3}(G_{S3}+G_{L3})}{G_{L3}+G_{S3}+G_{D3}} \end{bmatrix} \begin{bmatrix} v_{i1}^4 \\ v_{i2}^4 \end{bmatrix}. \tag{6.61}$$

Subcircuit 5:

$$\begin{bmatrix} i_{o1}^5 \\ i_{o2}^5 \end{bmatrix} = \begin{bmatrix} i_{eq1}^5 \\ i_{eq2}^5 \end{bmatrix} - \begin{bmatrix} g_{11}^5 & g_{12}^5 \\ g_{21}^5 & g_{22}^5 \end{bmatrix} \begin{bmatrix} v_{i1}^5 \\ v_{i2}^5 \end{bmatrix}$$
$$= \begin{bmatrix} -\dfrac{J_{L4}(G_{S4}+G_{D4})}{G_{L4}+G_{S4}+G_{D4}} \\ -\dfrac{J_{L4}G_{D4}}{G_{L4}+G_{S4}+G_{D4}} \end{bmatrix} - \begin{bmatrix} \dfrac{G_{L4}(G_{S4}+G_{D4})}{G_{L4}+G_{S4}+G_{D4}} & -\dfrac{G_{D4}G_{L4}}{G_{L4}+G_{S4}+G_{D4}} \\ \dfrac{G_{D4}G_{L4}}{G_{L4}+G_{S4}+G_{D4}} & -\dfrac{G_{D4}(G_{S4}+G_{L4})}{G_{L4}+G_{S4}+G_{D4}} \end{bmatrix} \begin{bmatrix} v_{i1}^5 \\ v_{i2}^5 \end{bmatrix}. \tag{6.62}$$

Subcircuit 6:

$$\begin{bmatrix} i_{o1}^6 \\ i_{o2}^6 \\ i_{o1}^6 \end{bmatrix} = \begin{bmatrix} i_{eq1}^6 \\ i_{eq2}^6 \\ i_{eq3}^6 \end{bmatrix} - \begin{bmatrix} g_{11}^6 & g_{12}^6 & g_{13}^6 \\ g_{21}^6 & g_{22}^6 & g_{23}^6 \\ g_{31}^6 & g_{32}^6 & g_{33}^6 \end{bmatrix} \begin{bmatrix} v_{i1}^6 \\ v_{i2}^6 \\ v_{i3}^6 \end{bmatrix}$$
$$= \begin{bmatrix} 0 \\ J_{C1} \\ -J_{C2} \end{bmatrix} - \begin{bmatrix} G_R & -G_R & -G_R \\ -G_R & G_{C1}+G_R & G_R \\ G_R & -G_R & -(G_{C2}+G_R) \end{bmatrix} \begin{bmatrix} v_{i1}^6 \\ v_{i2}^6 \\ v_{i2}^6 \end{bmatrix}. \tag{6.63}$$

Step 3 Formulate the network equations using the equivalents

Considering the interconnections between subcircuits in the FIBC converter shown in Fig. 6.3 the following relationship can be obtained.

Based on KVL, we can get the relationship between different port voltages in Eq. (6.64).

$$\begin{cases} v_{i1}^1 = v_{i1}^2 = v_{i1}^3 = v_{i1}^4 = v_{i1}^5 = v_{i1}^6 \\ v_{i2}^6 = v_{i2}^2 = v_{i2}^3 \\ v_{i3}^6 = v_{i3}^4 = v_{i3}^5 \end{cases}. \tag{6.64}$$

$v_{i1}^1, v_{i2}^6, v_{i3}^6$ can be selected as the independent input voltages of the six subcircuits.

Moreover, we can formulate the KCL equations shown in Eq. (6.65)

$$\begin{cases} i_{o1}^1 + i_{o1}^2 + i_{o1}^3 + i_{o1}^4 + i_{o1}^5 + i_{o1}^6 = 0 \\ i_{o2}^2 + i_{o2}^3 + i_{o2}^6 = 0 \\ i_{o3}^4 + i_{o3}^5 + i_{o3}^6 = 0 \end{cases}. \tag{6.65}$$

By replacing the output current with equivalent Eqs. (6.58)–(6.63) a reduced set of network equations is obtained in Eq. (6.66).

$$\begin{bmatrix} g_{11}^1+g_{11}^2+g_{11}^3+g_{11}^4+g_{11}^5+g_{11}^6 & g_{12}^2+g_{12}^3+g_{12}^6 & g_{12}^4+g_{12}^5+g_{13}^6 \\ g_{21}^2+g_{21}^3+g_{21}^6 & g_{22}^2+g_{22}^3+g_{22}^6 & g_{23}^6 \\ g_{21}^4+g_{21}^5+g_{31}^6 & g_{22}^4+g_{22}^5+g_{32}^6 & g_{33}^6 \end{bmatrix} \begin{bmatrix} v_{i1}^1 \\ v_{i2}^6 \\ v_{i3}^6 \end{bmatrix} = \begin{bmatrix} i_{eq}^1+i_{eq1}^2+i_{eq1}^3+i_{eq1}^4+i_{eq1}^5+i_{eq1}^6 \\ i_{eq2}^2+i_{eq2}^3+i_{eq2}^6 \\ i_{eq2}^4+i_{eq2}^5+i_{eq3}^6 \end{bmatrix}. \tag{6.66}$$

Step 4 Solve the network equations in Eq. (6.66). Compared to the network equation formulated directly by the nodal analysis method, the network equation in Eq. (6.66) is reduced from 7-order to 3-order. The scale is reduced, and the computing amount involved in solving the linear equations decrease significantly. Eq. (6.66) can be solved by various approaches presented in Sections 6.2 and 6.3.

Step 5 Solve the subcircuit. The reduced network equation Eq. (6.66) contains the contributions from all the subcircuits and provides the exact solution of the interface voltages $v_{i1}^1, v_{i2}^6, v_{i3}^6$. Interface variables are further treated as input sources to solve different groups according to Eqs. (6.58)–(6.63). It can be seen that the coupling between groups is still retained. The partitioned model is equivalent to the network without partitioning. Therefore the solution is of the same accuracy and stability.

6.3.2 State-space nodal method

The state-space nodal (SSN) method is a network-solving method that combines the state-space and nodal analyses for the simulation of a large electrical system. SSN method uses the state-space equations to describe the subcircuits and clusters them into a global equation formulated by nodal analysis. The basic steps of the SSN method are similar to that of NTT and can be summarized as follows:

1. Identify the subcircuits to be partitioned;
2. Write the state-space equations of the subcircuits, and derive the output equations by treating the interface current as the output variable;
3. Define the reduced network equations using the nodal analysis by combining the output equations of partitioned subcircuits and the remainder network;
4. Solve the network equations and compute the interface currents/voltages of the equivalents;
5. Compute subcircuits by using the newly computed interface variables.

Let us reconsider the circuit partitioning of the FIBC and use SSN to accomplish it.

Step 1 The first step of the SSN is the same as the NTT, which is to select the group to be partitioned. We can still select the same six subcircuits to be

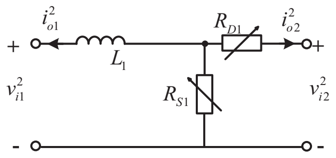


Figure 6.5 Subcircuit 2 of FIBC.

partitioned, according to Fig. 6.4. However, it is important to mention that in Fig. 6.4 the components are discretized in the companion circuit model and prepared for the nodal analysis, while in SSN the subcircuits should be modeled by the state-space equation without discretizing the components in advance.

Step 2

The state-space equations of the subcircuits can be written as Eq. (6.67) following the method presented in Chapter 5.

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}_k\mathbf{x} + \mathbf{B}_k\mathbf{u}\\ \mathbf{y} &= \mathbf{C}_k\mathbf{x} + \mathbf{D}_k\mathbf{u}.\end{aligned} \tag{6.67}$$

x is the vector of the state variables that are selected as the inductor currents and capacitor voltages in the power electronic circuits. **y** is the vector of the output variables in which the interfacing variables between subcircuits are selected.

Let us take subcircuit 2 as an example in Fig. 6.5 to illustrate this step. The inductor current i_{L1} is the state variable. The interfacing variables v_{i1}^2 and v_{i2}^2 are selected as the input sources of the subcircuits. Consequently, i_{o1}^2 and i_{o2}^2 are regarded as the output variables of the subcircuit. Therefore the state-space equations of subcircuit 2 can be written as

$$\begin{cases}\dfrac{di_{L1}}{dt} = -\dfrac{1}{L_1}\dfrac{R_S R_D}{R_S+R_D}i_{L1} + \begin{bmatrix}\dfrac{1}{L_1} & -\dfrac{1}{L_1}\dfrac{R_S}{R_S+R_D}\end{bmatrix}\begin{bmatrix}V_{i1}\\ V_{i2}\end{bmatrix}\\ \begin{bmatrix}i_{o1}\\ i_{o2}\end{bmatrix} = \begin{bmatrix}-1\\ \dfrac{R_{S1}}{R_{S1}+R_{D1}}\end{bmatrix}i_{L1} + \begin{bmatrix}0 & 0\\ 0 & -\dfrac{1}{R_{S1}+R_{D1}}\end{bmatrix}\begin{bmatrix}V_{i1}\\ V_{i2}\end{bmatrix}\end{cases}. \tag{6.68}$$

Discretize the state-space equations into Eq. (6.69) and formulate the output equations as Eq. (6.70).

$$\begin{aligned}i_{L1}^{(t)} = \frac{L_1(R_S+R_D)}{L_1(R_S+R_D)+hR_SR_D}i_{L1}^{(t-\Delta t)} &+ \frac{h(R_S+R_D)}{L_1(R_S+R_D)+hR_SR_D}V_{i1}\\ &- \frac{hR_S}{L_1(R_S+R_D)+hR_SR_D}V_{i2}\end{aligned} \tag{6.69}$$

$$\begin{bmatrix} i_{o1}^2 \\ i_{o2}^2 \end{bmatrix} = \begin{bmatrix} -\dfrac{h(R_{S1}+R_{D1})}{L_1(R_{S1}+R_{D1})+hR_{S1}R_{D1}} & \dfrac{hR_{S1}}{L_1(R_{S1}+R_{D1})+hR_{S1}R_{D1}} \\ \dfrac{hR_{S1}}{L_1(R_{S1}+R_{D1})+hR_{S1}R_{D1}} & -\dfrac{hR_{S1}+L_1}{L_1(R_{S1}+R_{D1})+hR_{S1}R_{D1}} \end{bmatrix} \begin{bmatrix} V_{i1}^2 \\ V_{i2}^2 \end{bmatrix} + \begin{bmatrix} -\dfrac{L_1(R_{S1}+R_{D1})}{L_1(R_{S1}+R_{D1})+hR_{S1}R_{D1}} \\ \dfrac{L_1R_{S1}}{L_1(R_{S1}+R_{D1})+hR_{S1}R_{D1}} \end{bmatrix} i_{L1}^{(t-\Delta t)} \tag{6.70}$$

Therefore in general the output Eq. (6.70) can be formulated by Eq. (6.71)

$$\mathbf{y}_n^{(t)} = \mathbf{W}_n\mathbf{u}_n^{(t)} + \mathbf{y}_{\mathbf{hist}} \tag{6.71}$$

In FIBC subcircuit 2, $\mathbf{y_n}$ represents the current in the interface that can be treated as the injection current, $\mathbf{u_n}$ is the node voltage, $\mathbf{y}_{hist}$ represents the history current sources, and $\mathbf{W}_n$ is the admittance matrix. Thus Eq. (6.70) is called a V-type SSN, which is a Norton equivalent. If the output variables are voltages, $\mathbf{y_n}$ represents the interface voltage, $\mathbf{u_n}$ represents the current flowing into or out of the subcircuits, $\mathbf{y}_{\text{hist}}$ represents the history voltage sources, and $\mathbf{W}_n$ is an admittance matrix. This type is called an I-type SSN, which is a Thevenin equivalent. In general, a subcircuit can have both V-type and I-type SSN representations, as shown in Eq. (6.72).

$$\begin{bmatrix} \mathbf{v}_n^I \\ \mathbf{i}_n^V \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{k_{\text{hist}}} \\ \mathbf{i}_{k_{\text{hist}}} \end{bmatrix} + \begin{bmatrix} \mathbf{W}_{II} & \mathbf{W}_{IV} \\ \mathbf{W}_{VI} & \mathbf{W}_{VV} \end{bmatrix} \begin{bmatrix} \mathbf{i}_n^I \\ \mathbf{v}_n^V \end{bmatrix} \tag{6.72}$$

If we use the conductance $G_S = 1/R_S$, $G_D = 1/R_D$, $G_L = h/L$, and $J_L = I_L^{t-\Delta t}$ to reorganize Eq. (6.70), we can find that the V-type SSN of subcircuit 2 has the same formula as Eq. (6.59) derived by NTT. Therefore the intrinsic ideas of the NTT and SSN when formulating the subcircuits are the same, that is, to use the Norton/Thevenin equivalent to represent the subcircuits from the view of the interface ports.

In order to cluster subcircuits into a global equation formulated by nodal analysis, all types of SSN representations should be transformed to the V-type represented by Eq. (6.73)

$$\begin{bmatrix} \mathbf{i}_n^I \\ \mathbf{i}_n^V \end{bmatrix} = \mathbf{\Gamma}_{k_n} \begin{bmatrix} \mathbf{v}_{k_{\text{hist}}} \\ \mathbf{i}_{k_{\text{hist}}} \end{bmatrix} + \mathbf{Y}_{k_n} \begin{bmatrix} \mathbf{v}_n^I \\ \mathbf{v}_n^V \end{bmatrix} \tag{6.73}$$

The FIBC subcircuit can be represented by the SNN form, where $\mathbf{i^k}$ is the vector of interface current of *k*th subcircuit and $\mathbf{v^k}$ is the vector of interface voltages of the *k*th subcircuit

$$\mathbf{i^k} = \mathbf{i^k_{hist}} + \mathbf{Y^k v^k} \tag{6.74}$$

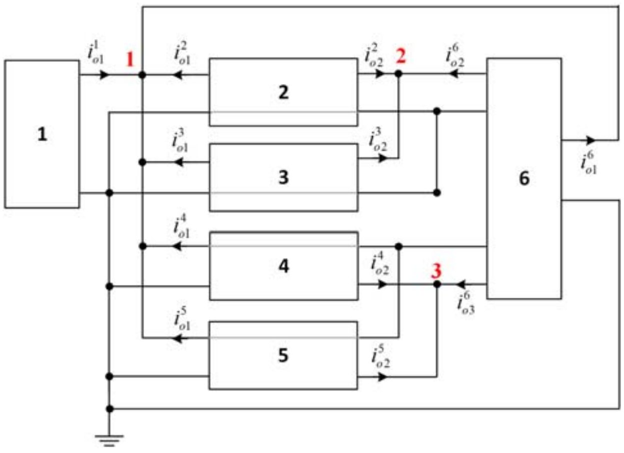


Figure 6.6 The interconnections of six subcircuits in FIBC.

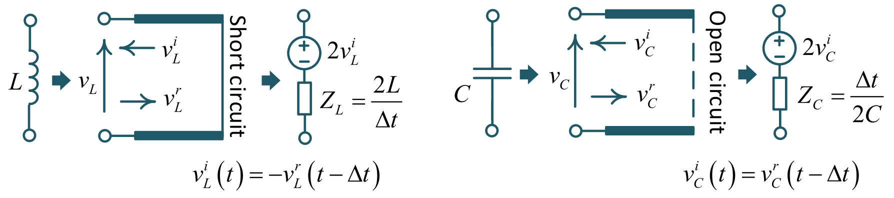
Step 3 Formulate the network equation by nodal analysis. The interconnections of six subcircuits are shown in Fig. 6.6. It can be seen that there are three independent nodes if the subcircuits are considered by their SNN equivalents. The KCL equations are written in Eq. (6.65). The currents flowing into the nodes can be replaced by their equivalent in Eq. (6.73). Since three nodal voltage voltages are involved, which are v_{n1}, v_{n2}, v_{n3} representing the potentials of nodes 1, 2, and 3 referring to the ground, the input voltages in Eq. (6.73) can be represented by the nodal voltage v_{n1}, v_{n2}, v_{n3}. Therefore the nodal voltage equations can be formulated into a three-order system of linear equations, presenting the relationship between the nodal voltage and the history current source. Considering that the subcircuits of the SNN method are the same as those of the NTT method and they are both derived from KCL, in the FIBC case the reduced-order network equations of SNN are exactly those of NTT in Eq. (6.66).

6.4 Delay-based network decoupling methods

The real-time simulation is based on discrete-time solvers. By considering a time delay between selected variables, different parts of the network can be mathematically decoupled within a given time step and therefore calculated independently.

6.4.1 Transmission line modeling method

Taking use of the time delays in the transmission line is a practical method to decouple a large network in the simulation. The reactive components can be represented by a transmission line modeling (TLM) stub or link. The TLM stub is a one-port model used to discretize the capacitance and inductance in the circuit, as



L
v_L
v_L^i
v_L^r
Short circuit
$2v_L^i$
$Z_L = \frac{2L}{\Delta t}$
$v_L^i(t) = -v_L^r(t-\Delta t)$
C
v_C
v_C^i
v_C^r
Open circuit
$2v_C^i$
$Z_C = \frac{\Delta t}{2C}$
$v_C^i(t) = v_C^r(t-\Delta t)$

Figure 6.7 Transmission line modeling stub.

shown in Fig. 6.7. The inductor is modeled by a short-circuit stub, and the capacitor is modeled by an open-circuit stub. According to the transmission line theory, the voltage across the stub is the sum of the incident pulse (denoted by subscript *i*) and the reflected pulse (denoted by subscript *r*) as shown in Eq. (6.75)

$$v = v^i + v^r \tag{6.75}$$

The voltage across the stub can be expressed by the Thevenin equivalent. As for an inductive stub the voltage across the inductor can be expressed by Eq. (6.76), where $Z_L = \frac{2L}{\Delta t}$.

$$v_L = i_L \cdot Z_L + 2v_L^i \tag{6.76}$$

As for a capacitor stub the voltage across the capacitor can be expressed by Eq. (6.77), where $Z_C = \frac{\Delta t}{2C}$.

$$v_C = i_C \cdot Z_C + 2v_C^i \tag{6.77}$$

In the TLM stub, the incident pulse will take one time step to travel and reflect back. Since the far end of the inductive stub is a short circuit, the reflected pulse will be inverted and the inverted pulse becomes the incident pulse for the next time step. Since the far end of the capacitor stub is an open circuit, the reflected pulse will become the incident pulse of the next time step.

$$\begin{cases} v_L^i(t) = -v_L^r(t-\Delta t) \\ v_C^i(t) = v_C^r(t-\Delta t) \end{cases} \tag{6.78}$$

The TLM link shown in Fig. 6.8 is a two-port model used to decouple two subcircuits that connect to it. The voltage across the link is the sum of the incident pulse and the reflected pulse as given in Eq. (6.75). The voltage across the link can be also expressed by the Thevenin equivalent, as shown in Eq. (6.79). For an inductive TLM link, $Z_{lk(L)} = L/\Delta t$, and for a capacitive TLM link, $Z_{lk(C)} = C/\Delta t$.

$$\begin{cases} v_A = i_A \cdot Z_{lk} + 2v_A^i \\ v_B = i_B \cdot Z_{lk} + 2v_B^i \end{cases} \tag{6.79}$$

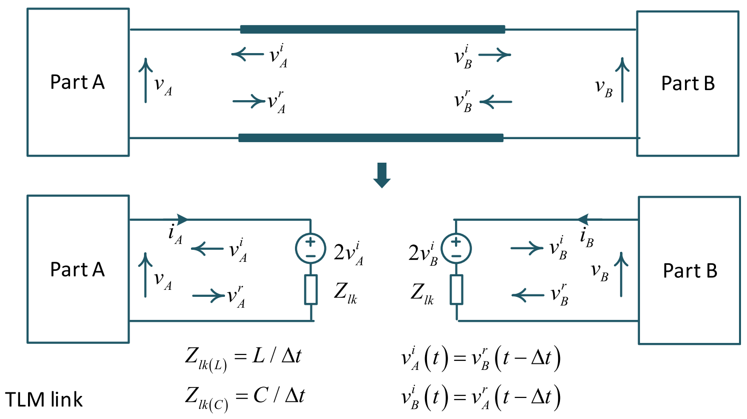


Figure 6.8 Transmission line modeling link.

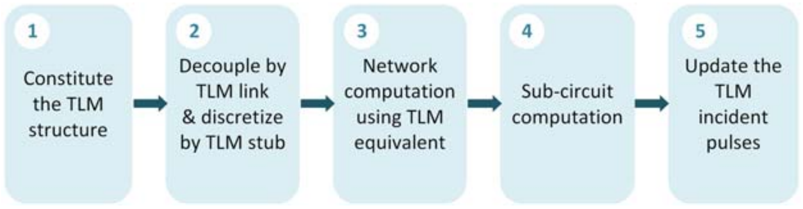

1
Constitute the TLM structure
2
Decouple by TLM link & discretize by TLM stub
3
Network computation using TLM equivalent
4
Sub-circuit computation
5
Update the TLM incident pulses

Figure 6.9 The basic steps of the transmission line modeling decoupling method.

As for the TLM link in Fig. 6.8, the reflected pulse from one side will take one time step to travel to the other side and become its incident pulse in the next time step, expressed by Eq. (6.80).

$$\begin{cases} v_A^i(t) = v_B^r(t-\Delta t) \\ v_B^i(t) = v_A^r(t-\Delta t) \end{cases} \tag{6.80}$$

Therefore the incident pulse will be preknown, and the two parts connected to the TLM link can be decoupled in the simulation. The basic steps of TLM-based decoupling are summarized in Fig. 6.9.

6.4.2 Equivalent power source method

Another intuitive way of inserting delays is to use the values from the last time step to directly feed the computation in the current time step. Considering the decoupling of two interconnected parts in Fig. 6.10. The current and voltage from the last time step are directly used as the sources injected into subcircuits in the current

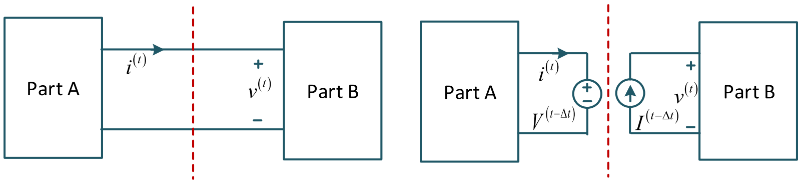


Figure 6.10 Illustrations of the equivalent power source.

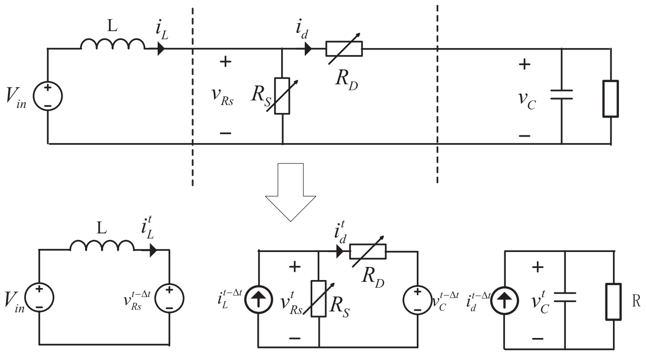


Figure 6.11 Decoupling of the boost converter using equivalent power source method.

time step. By using this method, it is equivalent to that a voltage source and a current source are inserted at the decoupling interface, and this decoupling method can be called the equivalent power source method. It needs to mention that for the practical uses, this decoupling method requires the inclusion of the snubbers to avoid the numberical instability issue.

Consider the decoupling of the boost converter using the equivalent power source methods, as shown in Fig. 6.11.

Step 1: Select the decoupling interface near the inductor and capacitor

Step 2: Use the inductor current $i_L^{t-\Delta t}$, the voltage $v_{Rs}^{t-\Delta t}$ across R_S, the capacitor voltage $v_C^{t-\Delta t}$, and the current $i_d^{t-\Delta t}$ through R_D from the last simulation step as the power sources, and inject them into the decoupling interfaces. The boost converter will be decoupled into three subcircuits.

Step 3: Compute the subcircuits, and obtain the values of i_L^t, v_{Rs}^t, i_d^t, and v_C^t that can be used for the computation of the next step.

6.4.3 Explicit numerical integration method

The explicit integration method can also be used to decouple the network at a discrete time step since it concerns only the values of the previous time step. Therefore the multisolver strategy combining the implicit method and explicit method can be used to realize the paralleled modeling structure [4].

Consider the network represented by the state-space equation Eq. (6.81). If some state variables are discretized by an explicit integration method, such as forward Euler (FE) method, and others are discretized by an implicit integration method, such as backward Euler (BE) method, the discretized system equations can then be represented by Eqs. (6.82) and (6.83), where $\boldsymbol{x_E}$ and $\boldsymbol{x_I}$ represent state variables discretized by FE method and BE method, respectively.

$$\dot{\mathbf{x}} = \mathbf{Ax} + \mathbf{Bu} \Rightarrow \begin{bmatrix} \dot{\boldsymbol{x}}_E \\ \dot{\boldsymbol{x}}_I \end{bmatrix} = \begin{bmatrix} \boldsymbol{A_E} & \boldsymbol{A}_{\mathrm{EI}} \\ \boldsymbol{A}_{\mathrm{IE}} & \boldsymbol{A_I} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_E} \\ \boldsymbol{x_I} \end{bmatrix} + \begin{bmatrix} \boldsymbol{B_E} \\ \boldsymbol{B_I} \end{bmatrix} \boldsymbol{u} \tag{6.81}$$

$$\boldsymbol{x}_{\boldsymbol{E}}^{\boldsymbol{n}+1} = \boldsymbol{x}_{\mathrm{FE}}^{\boldsymbol{n}} + \boldsymbol{h}\left(\boldsymbol{A_E}\boldsymbol{x}_{\boldsymbol{E}}^{\boldsymbol{n}} + \boldsymbol{A}_{\mathrm{EI}}\boldsymbol{x}_{\boldsymbol{I}}^{\boldsymbol{n}} + \boldsymbol{B_E}\boldsymbol{u}^{\boldsymbol{n}}\right) \tag{6.82}$$

$$\boldsymbol{x}_{\boldsymbol{I}}^{\boldsymbol{n}+1} = \boldsymbol{x}_{\boldsymbol{I}}^{\boldsymbol{n}} + \boldsymbol{h}\left(\boldsymbol{A}_{\mathrm{IE}}\boldsymbol{x}_{\boldsymbol{E}}^{\boldsymbol{n}+1} + \boldsymbol{A_I}\boldsymbol{x}_{\boldsymbol{I}}^{\boldsymbol{n}+1} + \boldsymbol{B_I}\boldsymbol{u}^{\boldsymbol{n}+1}\right) \tag{6.83}$$

Since the FE method only needs the previous values of the system, if $\boldsymbol{x}_{\boldsymbol{E}}^{\boldsymbol{n}+1}$ in Eq. (6.83) is replaced by Eq. (6.82), $\boldsymbol{x}_{\boldsymbol{I}}^{\boldsymbol{n}+1}$ is no longer coupled with $\boldsymbol{x}_{\boldsymbol{E}}^{\boldsymbol{n}+1}$ anymore. Therefore $\boldsymbol{x}_{\boldsymbol{E}}^{\boldsymbol{n}+1}$ can be calculated at the beginning of each time step and be treated as a source injected into the state equations Eq. (6.83).

For example, if the inductor current and capacitor voltage are discretized by the FE method, as given in Eq. (6.84), the inductor current and capacitor voltage can be treated as a current source and a voltage source injected into the rest of the circuit. Therefore two parts in series with the inductor and two parts in parallel with the capacitor can be thus decoupled at the present time step, as shown in Fig. 6.12.

$$\begin{cases} I_L^{n+1} = I_L^n + \dfrac{h}{L} V_L^n \\ V_C^{n+1} = V_C^n + \dfrac{h}{C} V_C^n \end{cases} \tag{6.84}$$

Fig. 6.13 demonstrates the procedure of using explicit numerical integration to decouple a DC-AC inverter. The DC link capacitor and the filter inductor are

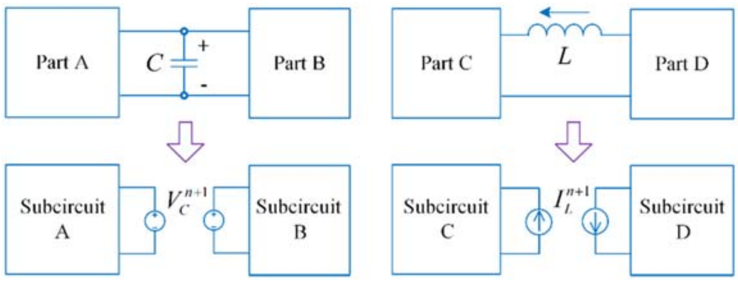


Figure 6.12 Network decoupling by explicit numerical integration method.

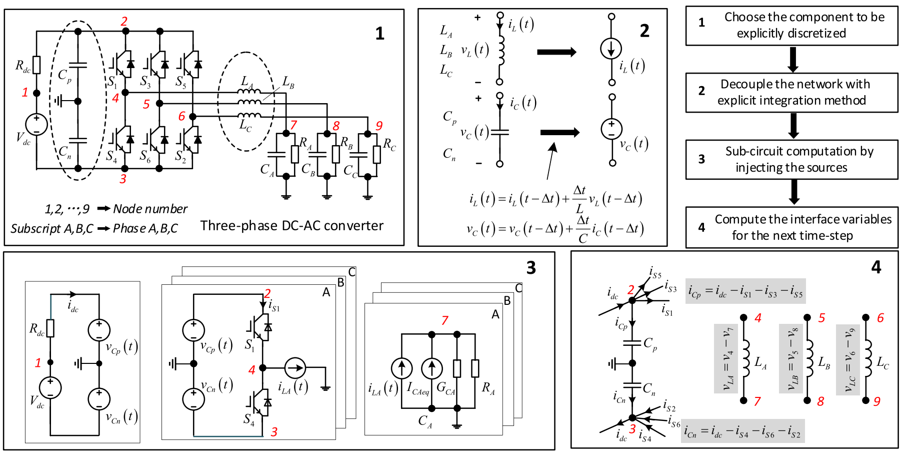

Three-phase DC-AC converter
1,2,⋯,9 → Node number
Subscript A,B,C → Phase A,B,C
1 Choose the component to be explicitly discretized
2 Decouple the network with explicit integration method
3 Sub-circuit computation by injecting the sources
4 Compute the interface variables for the next time-step
$i_L(t)=i_L(t-\Delta t)+\frac{\Delta t}{L}v_L(t-\Delta t)$
$v_C(t)=v_C(t-\Delta t)+\frac{\Delta t}{C}i_C(t-\Delta t)$
$i_{Cp}=i_{dc}-i_{S1}-i_{S3}-i_{S5}$
$i_{Cn}=i_{dc}-i_{S4}-i_{S6}-i_{S2}$
$v_{LA}=v_4-v_7$
$v_{LB}=v_5-v_8$
$v_{LC}=v_6-v_9$

Figure 6.13 Procedure of decoupling the inverter using explicit numerical integration.

discretized by the FE method, leading to seven subcircuits being computed in parallel. At last the variables in the decoupled interfaces are updated and prepared for the FE integration in the next time step.

However, it must be mentioned that, the explicit integration suffers from the issue of numerical stability. Therefore when applying the multisolver strategy, the stability issue should be verified. The global discretized system can be reformulated as Eq. (6.85) from Eqs. (6.82) and (6.83). Therefore if the poles of this discretized system are located within a unit cycle, the solver is considered to be numerically stable.

$$\begin{aligned} &\boldsymbol{x}^{n+1}=\boldsymbol{A}_{pre}\cdot\boldsymbol{x}^{n}+\boldsymbol{A}_{cur}\cdot\boldsymbol{x}^{n+1}+\boldsymbol{B}_{pre}\cdot\boldsymbol{u}^{n}+\boldsymbol{B}_{cur}\cdot\boldsymbol{u}^{n+1}\\ &\Rightarrow\boldsymbol{x}^{n+1}=(\boldsymbol{I}-\boldsymbol{A}_{cur})^{-1}\boldsymbol{A}_{pre}\cdot\boldsymbol{x}^{n}+(\boldsymbol{I}-\boldsymbol{A}_{cur})^{-1}\left(\boldsymbol{B}_{pre}\cdot\boldsymbol{u}^{n}+\boldsymbol{B}_{cur}\cdot\boldsymbol{u}^{n+1}\right)\end{aligned} \tag{6.85}$$

6.4.4 Multirate modeling method

A multirate modeling (MRM) method is usually used to deal with a stiff system. The state variables with markedly different time constants are discretized into different time steps in MRM. Power electronic converters have fast dynamics due to switching behaviors. Thus the multirate simulation can effectively decouple the power electronic converters from the power system, the electric machines, and the renewable energy sources. Therefore each subsystem should choose the most appropriate time step to represent their behaviors to avoid unnecessary computation time. The slower subsystems are computed with a larger time step, while the faster subsystems are computed with a smaller time step. The decoupling of the different rates is realized by introducing delays between them. The basic scheme of multirate simulation is depicted in Fig. 6.14.

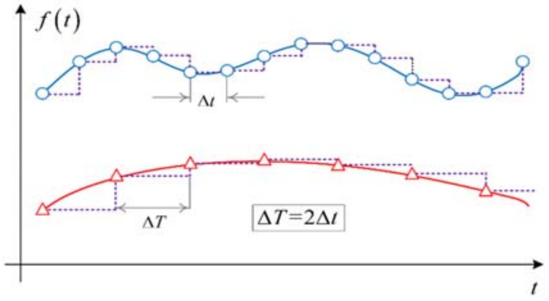


Figure 6.14 Multirate decoupling scheme.

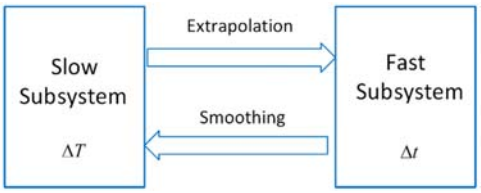


Figure 6.15 Data communications in the multirate system.

The multirate simulation introduces the issue of how to exchange data between two different simulation time steps. From the faster to the slower subsystem the values between two samples can be kept constant with a zero-order holder. More practically, the data produced by the fast subsystems should be smoothed for the slower subsystems, and conversely the data produced by the slow subsystems should be extrapolated for the fast subsystems, as shown in Fig. 6.15.

Another issue is the numerical stability. The unified discretized system representation is not easy to formulate for the stability analysis since the different time steps are involved in the state space equations. In [5] the discretized state-variable equations of a multirate system are linearized around the smallest sampling rate. The different state variables are then unified in a single state-space matrix system, which can be used to validate the accuracy and stability by pole location of the discretized system.

The discretized system of a multirate system can be represented by Eq. (6.86). The subscript S represents the slow-dynamic state variables, and F represents the fast variables. The variable ϱ is of binary values 1 and 0. Both fast and slow states are updated if ϱ equals to 1. On the other hand, if ϱ is equal to 0, only the fast states are updated and the slow states are treated as constant.

$$\begin{bmatrix} X_S^n \\ X_F^n \end{bmatrix} = \begin{bmatrix} \varrho A_S + (I-\varrho) & \varrho A_{SF} \\ A_{FS} & A_F \end{bmatrix} \begin{bmatrix} X_S^{n-1} \\ X_F^{n-1} \end{bmatrix} + \begin{bmatrix} \varrho B_S \\ B_F \end{bmatrix} U^n \tag{6.86}$$

The effects of binary variables ϱ are then averaged during the large time step ΔT with the definition in Eq. (6.87) just as the same concept as the linearization of the effects of switching actions on the power converter. The $\overline{\varrho}$ is obtained with regard to the ratio of the smallest Δt to others.

$$\overline{\varrho}_{1,2,\ldots} = \frac{\Delta t}{\Delta T_{1,2,\ldots}} \tag{6.87}$$

Therefore the multirate system can be formulated as a single-rate discretized state-space equation, as shown in Eq. (6.88). The multirate system is linearized in the smallest time step, and Eq. (6.88) is only represented by a single rate Δt. The pole location can thus be analyzed to verify the accuracy and the stability of the multirate solver [5].

$$\begin{bmatrix} X_S^n \\ X_F^n \end{bmatrix} = \left(M_\varrho \begin{bmatrix} A_S & A_{SF} \\ A_{FS} & A_F \end{bmatrix} + I - M_\varrho \right) \begin{bmatrix} X_S^{n-1} \\ X_F^{n-1} \end{bmatrix} + \left(M_\varrho \begin{bmatrix} B_S \\ B_F \end{bmatrix} \right) U^n \tag{6.88}$$

where $M_\varrho = \begin{bmatrix} \overline{\varrho}_1 & 0 & 0 \\ 0 & \overline{\varrho}_{\ldots} & 0 \\ 0 & 0 & I \end{bmatrix}$.

6.5 Numerical solution of nonlinear electric networks

6.5.1 Newton–Raphson method

In the simulation, the circuit components can be nonlinear, where the voltage–current relationship is nonlinear. In Chapter 4, we have derived the V-I relationship of the resistor, capacitor, and inductor, as given in Eqs. (6.89)–(6.91).

Resistor:

$$v = f(i) \tag{6.89}$$

Capacitor:

$$i(t) = \frac{dq}{dt} = \frac{dq}{dv}\frac{dv}{dt} = C(v)\frac{dv}{dt} \tag{6.90}$$

Inductor:

$$v(t) = \frac{d\phi}{dt} = \frac{d\phi}{di}\frac{di}{dt} = L(i)\frac{di}{dt} \tag{6.91}$$

The resistor does not contain the derivative component. If a nonlinear resistor is considered, we can use the Newton–Raphson (N-R) method to formulate the resistor as the companion circuit model.

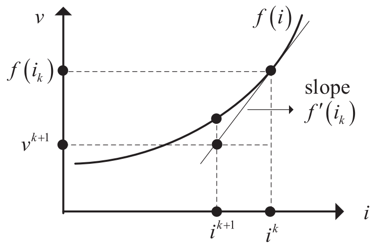


Figure 6.16 Graphical illustrations of the N-R iteration method.

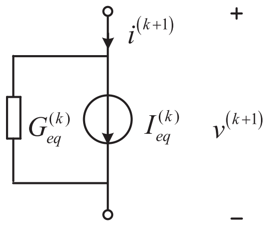


Figure 6.17 Equivalent model of a nonlinear resistor using N-R algorithm.

The main concept of the N-R iteration algorithm is to draw a tangent line to $v = f(i)$ at the point $i = i^{(k)}$ and use this line to represent the V-I relationship of the resistor at the $(k+1)$th iteration, as shown in Fig. 6.16. The linear V-I relationship shown in Eq. (6.92) will be used in the network-solving process, and the resulting status $i^{(k+1)}$ and $f\left(i^{(k+1)}\right)$ will be used for the next iteration to constitute a new line with a closer V-I relationship to the exact operating point.

$$v^{(k+1)} = f'\left(i^{(k)}\right)\left(i^{(k+1)} - i^{(k)}\right) + f\left(i^{(k)}\right) \tag{6.92}$$

The resistor at the $(k+1)$th iteration can be represented by Fig. 6.17, where

$$\begin{cases} G_{eq}^{(k)} = \left[f'\left(i^k\right)\right]^{-1} \\ I_{eq}^{(k)} = i^{(k)} - G_{eq} f\left(i^k\right) \end{cases} \tag{6.93}$$

The nonlinearity of the capacitor is embodied in the relationship between charge and voltage $q = g(v)$. If the BE method is used, we can get the V-I relationship at the step (t_n, t_{n+1}) by Eq. (6.94).

$$i_{n+1} = \dot{q}_{n+1} = q_{n+1} - q_n = \frac{g(v_{n+1})}{h} - \frac{g(v_n)}{h} = f_C(v_{n+1}) \tag{6.94}$$

The N-R iteration algorithm can be formulated by Eq. (6.95)

$$i_{n+1}^{(k+1)} = f_C{}'\left(v_{n+1}^{(k)}\right)\left(v_{n+1}^{(k+1)} - v_{n+1}^{(k)}\right) + f_C\left(v_{n+1}^{(k)}\right) \tag{6.95}$$

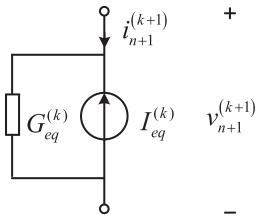


Figure 6.18 Companion circuit model of the nonlinear capacitor/inductor using N-R algorithm.

The capacitor V-I relationship at the $(k+1)$th iteration in the discretized time (t_n, t_{n+1}) can be represented by Fig. 6.18, where

$$\begin{cases} G_{eq}^{(k)} = f_C{}'\left(v_{n+1}^{(k)}\right) = \dfrac{g'\left(v_{n+1}^{(k)}\right)}{h} \\ I_{eq}^{(k)} = f_C\left(v_{n+1}^{(k)}\right) - G_{eq}^{(k)} v_{n+1}^{(k)} \end{cases} \tag{6.96}$$

The nonlinearity of the inductor is embodied in the relationship between flux and current $\phi = l(i)$. If the BE is used, we can get the V-I relationship at the step (t_n, t_{n+1}) by Eq. (6.97)

$$v_{n+1} = \dot{\phi}_{n+1} = \phi_{n+1} - \phi_n = \frac{l(i_{n+1})}{h} - \frac{l(i_n)}{h} = f_L(i_{n+1}) \tag{6.97}$$

The N-R iteration algorithm can be formulated by Eq. (6.98)

$$v_{n+1}^{(k+1)} = f_L{}'\left(i_{n+1}^{(k)}\right)\left(i_{n+1}^{(k+1)} - i_{n+1}^{(k)}\right) + f_L\left(i_{n+1}^{(k)}\right) \tag{6.98}$$

The inductor V-I relationship at the $(k+1)$th iteration in the discretized time (t_n, t_{n+1}) can also be represented by Fig. 6.18, where

$$\begin{cases} G_{eq}^{(k)} = \left[f_L{}'\left(i_{n+1}^{(k)}\right)\right]^{-1} = \dfrac{h}{l'\left(i_{n+1}^{(k)}\right)} \\ I_{eq}^{(k)} = i_{n+1}^{(k)} - G_{eq}^{(k)} f_L\left(i_{n+1}^{(k)}\right) \end{cases} \tag{6.99}$$

As for the nonlinear power sources the N-R iterative algorithm can also be applicable to constitute a companion circuit model. Let us consider the nonlinear equivalent circuit model of an IGBT, as presented in Chapter 3. In this model the MOSFET unit shown in Fig. 6.19 is actually represented by the companion model and inserted into the network nodal analysis.

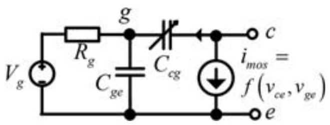


Figure 6.19 MOSFET unit in the nonlinear equivalent circuit model of IGBT.

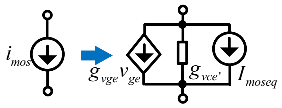


Figure 6.20 Companion circuit model of the nonlinear current source in MOSFET unit.

The current source i_{mos} represents the output characteristics of IGBT where three operating regions are modeled. Based on the different modeling accuracies, the i_{mos} has various mathematical representations. Let us consider the abstraction of i_{mos} as Eq. (6.100)

$$i_{\text{mos}} = f\left(v_{ce}, v_{\text{ge}}\right) \tag{6.100}$$

The N-R iteration algorithm is applied, and the iterative formula is Eq. (6.101).

$$i_{\text{mos}}^{(k+1)} = \left.\frac{\partial i_{\text{mos}}}{\partial v_{\text{ge}}}\right|_{v_{\text{ge}}^{(k)}} \left(v_{\text{ge}}^{(k+1)} - v_{\text{ge}}^{(k)}\right) + \left.\frac{\partial i_{\text{mos}}}{\partial v_{\text{ce}}}\right|_{v_{\text{ce}}^{(k)}} \left(v_{\text{ce}}^{(k+1)} - v_{\text{ce}}^{(k)}\right) + f\left(v_{\text{ce}}^{(k)}, v_{\text{ge}}^{(k)}\right) \tag{6.101}$$

The nonlinear current source in the MOSFET unit can be modeled by Fig. 6.20, where

$$\begin{cases} g_{\text{vge}} = \dfrac{\partial i_{\text{mos}}}{\partial v_{\text{ge}}} \\ g_{vce} = \dfrac{\partial i_{\text{mos}}}{\partial v_{ce}} \\ I_{\text{moseq}} = f\left(v_{\text{ce}}^{(k)}, v_{\text{ge}}^{(k)}\right) - g_{\text{vge}} v_{\text{ge}} - g_{\text{vce}} v_{ce} \end{cases} \tag{6.102}$$

6.5.2 Iterative solver design

The nonlinear components can be represented by the companion circuit model by using the N-R iteration algorithm. Therefore the network containing the nonlinear components can be solved by the nodal analysis at each iteration. If the voltage or current errors of the nonlinear components meet the criterion, the iteration will

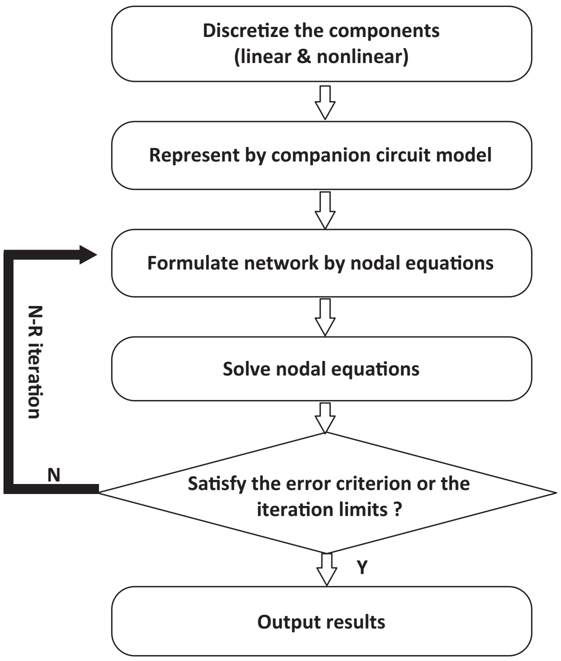

Discretize the components (linear & nonlinear)
Represent by companion circuit model
Formulate network by nodal equations
Solve nodal equations
Satisfy the error criterion or the iteration limits ?
N
N-R iteration
Y
Output results

Figure 6.21 Iterative N-R solver design using nodal analysis method.

stop. However, due to the computing time constraint of the real time simulation, the iteration numbers should have an upper limit. Therefore when the errors are not converged into the error criterion, the iteration will end up if the iteration number exceeds the upper limit. The procedure of the iterative N-R solver design using nodal analysis is depicted in Fig. 6.21.

6.5.3 Lookup-table method

The iterative solver can be time-consuming for the small time-step real-time simulation. There is an alternative way to embrace the nonlinearities of electrical networks in applicable real-time simulations, where the nonlinear functions are represented by the lookup table (LUT).

Let us take the nonlinear inductor as an example. It is implemented as a controlled current source as a function of the inductor's flux. Function $i(\phi)$ (where i is the inductor current and ϕ is the inductor flux) is defined for the inductor, and an LUT is generated based on flux−current curve, as shown in Fig. 6.22. Linear interpolation is applied to the flux values between the defined points, while linear extrapolation is applied in case a current value is outside of the specified range.

The schematic diagram of the nonlinear inductor model is shown in Fig. 6.23. In the electrical circuit the nonlinear inductor is represented as a current source. Voltage across the inductor is measured and integrated, with the result of the

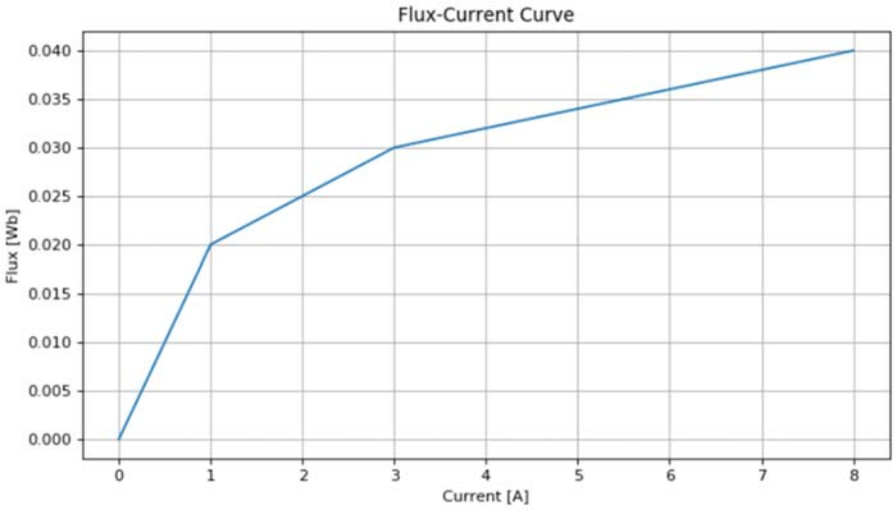


Figure 6.22 The flux—current curve in lookup-table.

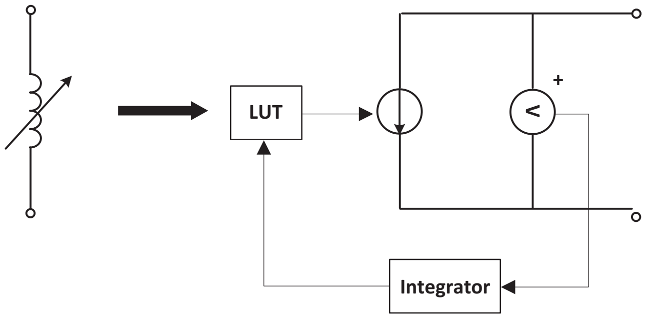


Figure 6.23 Schematic diagram of the nonlinear inductor model.

integration being the inductor's flux. Flux is used to address the LUT. The output of the lookup table is the inductor current which is fed to the controlled current source. The snubber circuit can be added in parallel with the current source using the purely resistive R snubber or the R-C snubber.

6.6 Illustrative example

A DC-DC-AC topology is a commonly used power electronic topology in the power train of fuel cell electric vehicles, where DC-DC steps up the fuel cell output and DC-AC drives the motor. A DC-DC-AC simplified topology shown in

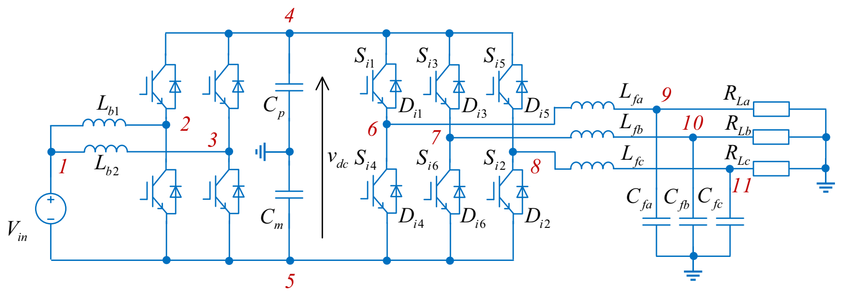


Figure 6.24 Boost–inverter DC-DC-AC topology.

Table 6.1 Parameters of the boost–inverter.

Parameters	Values	Parameters	Values
Input voltage (V_{in})	270V	Inductance (L_o)	1mH
Filter inductance (L_f)	1mH	DC bus capacitance	1mF
Filter capacitance (C_f)	1μF	Load resistance (R_L)	0.8Ω
Boost PWM frequency	10kHz	Carrier frequency	10kHz
Gate voltage (V_g)	-5/15V	Gate resistance	10Ω

Fig. 6.24 includes a two-phase interleaved boost converter and a three-phase two-level inverter, and it is selected as an example to show both the system-level simulation and the device-level simulation using the explicit integration method presented in Section 6.4.3. The simulation parameters are shown in Table 6.1. In this case the power electronic converters are modeled and simulated in the offline environments, and the solver is coded by the M function. Given that the book is focused on HIL for power electronics, offline simulation here is used to demonstrate the use of the methods covered in this chapter.

The explicit-integration method is applied to decouple the switches from the network. After discretizing L_{b1}, L_{b2}, C_P, C_m, L_{fa}, L_{fb}, and L_{fc} with the FE method, i_{Lb1}^{n+1}, i_{Lb2}^{n+1}, i_{fa}^{n+1}, i_{fb}^{n+1}, and i_{fc}^{n+1} can be treated as current sources injected into the network at each time step. Similarly, v_{Cp}^{n+1} and v_{Cm}^{n+1} can be regarded as voltage sources. As a result the circuit in Fig. 6.24 is partitioned into nine subcircuits, as shown in Fig. 6.25. Subcircuits 2–6 are of the half-bridge topology which can be solved by the IGBT/diode models presented in Chapter 3. Subcircuits 7–9 include the filter capacitors and load resistors, where the filter capacitors are modeled by their companion circuits [6]. The subcircuits are of simple structures, and the computation of each part is totally independent at each time step.

In the system-level simulation, the switches in Fig. 6.24 are modeled by the binary switch models. By considering the forward conduction voltage of the IGBT and diode the binary resistor is in series with a voltage source that models the forward voltage in the turn-on status, as shown in Fig. 6.26.

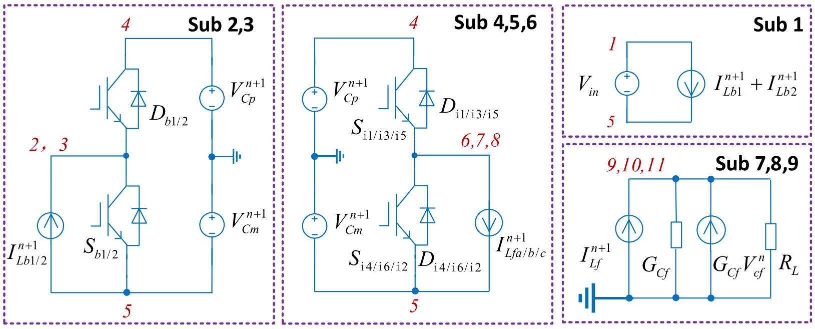


Figure 6.25 Network decoupling of the boost inverter.

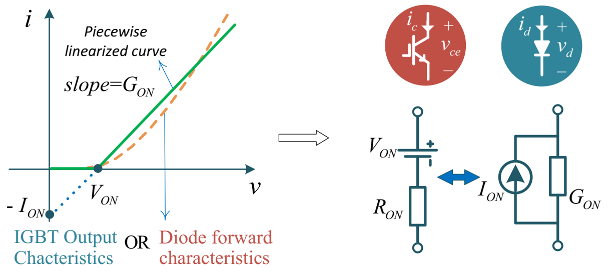


Figure 6.26 Static state IGBT/diode model.

In the device-level simulation the IGBT/diode switching pair is modeled by the piecewise linear transient model presented in Chapter 3. The continuous behaviors of the IGBT/diode model are described in four phases. It can be observed that each phase can be unified and represented by the state-space equation in Eq. (6.103),

$$\dot{x}_i = A_i x_i + B_{i1} u_{i1} + B_{i2} u_{i2} \tag{6.103}$$

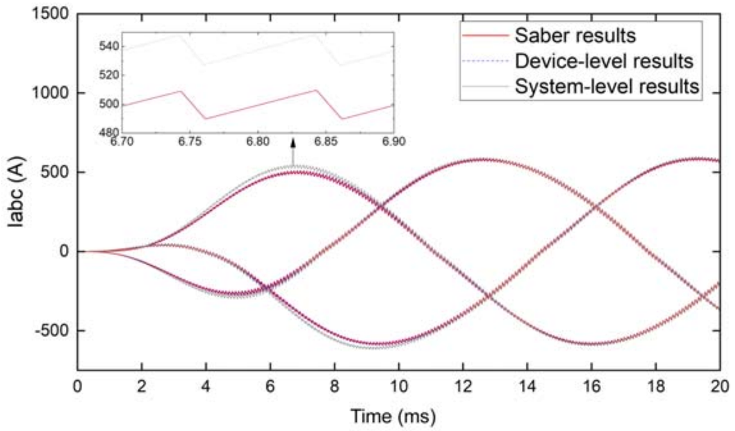
where x_i is the vector of state variables $\begin{bmatrix} v_{ge} & v_{ce} & v_{ctail} & i_{Lrr} \end{bmatrix}^T$, u_{i1} is the vector of internal sources $\begin{bmatrix} V_g & V_{th} & V_{d(on)} & V_{s(on)} \end{bmatrix}^T$, and u_{i2} is the vector of the external sources $\begin{bmatrix} I_L & V_{cap} \end{bmatrix}^T$. The subscript i refers to the IGBT/diode model.

By discretizing Eq. (6.103) with BE integration method, Eq. (6.104) can be obtained,

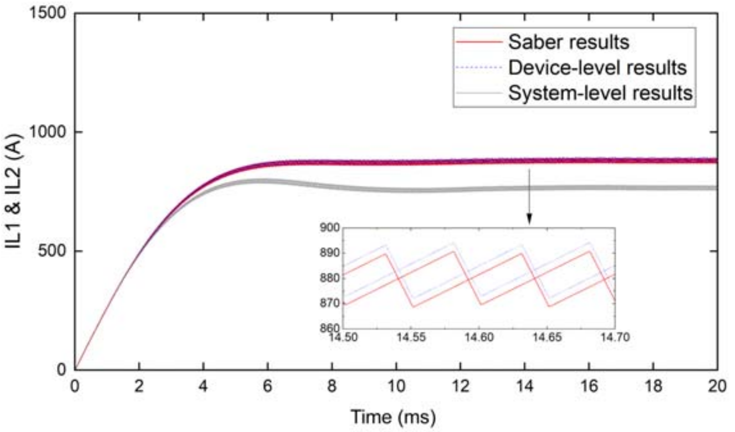
$$x_i^{n+1} = \widehat{A}_i x_i^n + \widehat{B_{i1}} u_{i1}^{n+1} + \widehat{B_{i2}} u_{i2}^{n+1} \tag{6.104}$$

where $\widehat{A}_i = (I - hA_i)^{-1}$, $\widehat{B_{i1}} = h(I - hA_i)^{-1} B_{i1}$, $\widehat{B_{i2}} = h(I - hA_i)^{-1} B_{i2}$.

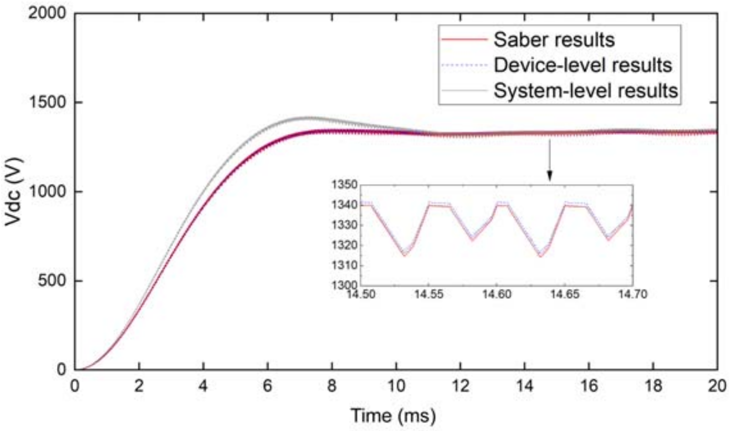
C_{cg} and C_{ce} are nonlinear capacitances changing their values with the voltage across the collector and the gate V_{cg} [7]. In this model, C_{cg} and C_{ce} are represented by a group of discretized values according to the capacitance curves provided by



(A)



(B)



(C)

Figure 6.27 Simulation results of the DC-DC-AC converter from 0 to 20 ms. Solid line, Saber simulation results; dashed line, simulation results using the piecewise linear transient model. (A) Three-phase currents I_{LA}, I_{LB}, and I_{LC}; (B) interleaved boost inductor currents I_{Lb1} and I_{Lb2}; (C) dc bus voltage V_{dc}.

the device datasheet. The capacitance is assumed to be constant between two discrete points. The capacitance values can be obtained by the lookup table method based on the value of V_{cg} from the previous time step.

As indicated in Chapter 3, four phases of the piecewise linear transient model can be identified by the boundary condition between different operating regions of IGBT and the diode, and the transitioning between phases is realized by a state machine that has a definite initial state and a unidirectional working flow.

The simulation of the DC-DC-AC converter is decoupled by using the explicit-integration method. The models are implemented in the Simulink environment, and the models are solved in a programmed M-function according to the flow chart shown in Fig. 6.13. The systems of linear equations $Ax = b$ involved in both system-level simulation and device-level simulation are realized using the precomputing method. The simulation time step is set to 50 ns. A reference model of this

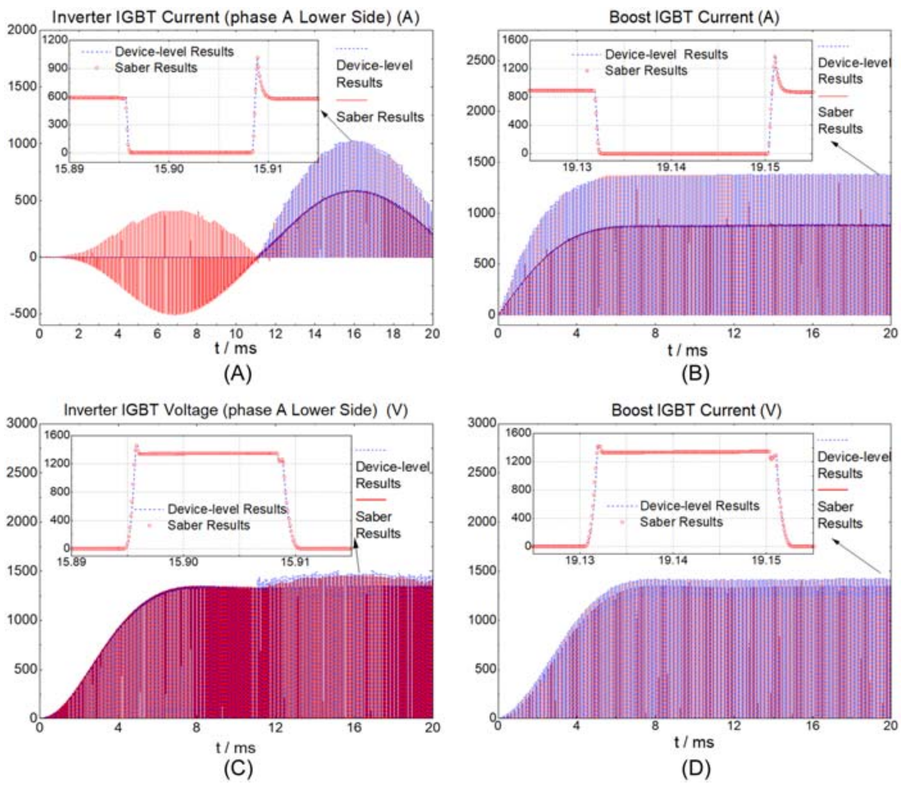


Figure 6.28 Simulation results of switching transient behaviors. Solid line, Saber simulation results; dashed line, simulation results using the piecewise linear transient model. (A) IGBT current of inverter phase A lower side $I_{ce(i4)}$; (B) IGBT current of interleaved boost $I_{ce(b2)}$; (C) IGBT voltage of inverter phase A lower side $V_{ce(i4)}$; (D) IGBT voltage of the interleaved boost $V_{ce(b2)}$.

boost-inverter topology is also modeled in the Saber simulation tool using an IGBT reference model selected from the Saber model architect tool. The Saber model is solved by an N-R algorithm with a variable time-step strategy. The boost converter and three-phase inverter are operating under the open loop conditions to eliminate the influences from the controller. The boost converter operates under a 0.8 duty cycle. The inverter operates under a 50Hz sinusoidal reference and 0.8 amplitude modulation index.

The simulation results of the tested converter are illustrated in Figs. 6.27 and 6.28. The inverter three-phase currents, boost inductor currents, and the dc bus voltage are depicted in Fig. 6.27A−C, produced by the Saber reference model, the device-level model, and the system-level model. The device-level simulation results match the Saber results well because the Saber model actually adopts a transient IGBT/diode model as well. However, the system-level simulation results deviate from the reference to some extent. Therefore we can treat the device-level simulation as a more accurate one than the system-level simulation. The current through and the voltage across the inverter lower arm IGBT are depicted in Fig. 6.28A and C. The boost IGBT waveforms are depicted in Fig. 6.28B and D. The switching transient behaviors represented by the piecewise linear transient model have good agreement compared to the Saber reference model. Based on the simulation results of the tested case, the proper network decoupling can realize a paralleled computing structure in the simulation solver while maintaining good accuracy. The paralleled network modeling will contribute to the possibility of simulating larger networks with more complex device models.

References

[1] T. Nechma, M. Zwolinski, Parallel sparse matrix solution for circuit simulation on FPGAs, IEEE Transactions on Computers 64 (4) (2014) 1090−1103.

[2] C.W. Bomhof, H.A. Van Der Vorst, A parallel linear system solver for circuit simulation problems, Numerical Linear Algebra with Applications 7 (7-8) (2000) 649−665.

[3] A. Hadizadeh, M. Hashemi, M. Labbaf, et al., A matrix-inversion technique for FPGA-based real-time EMT simulation of power converters, IEEE Transactions on Industrial Electronics 66 (2) (2018) 1224−1234.

[4] L.A. Grégoire, H.F. Blanchette, J. Bélanger, et al., Real-time simulation-based multisolver decoupling technique for complex power-electronics circuits, IEEE Transactions on Power Delivery 31 (5) (2016) 2313−2321.

[5] L.A. Grégoire, H.F. Blanchette, J. Belanger, et al., A stability and accuracy validation method for multirate digital simulation, IEEE Transactions on Industrial Informatics 13 (2) (2016) 512−519.

[6] H. Bai, H. Luo, C. Liu, et al., A device-level transient modeling approach for the FPGA-based real-time simulation of power converters, IEEE Transactions on Power Electronics 35 (2) (2019) 1282−1292.

[7] J.T. Hsu, K.D.T. Ngo, Behavioral modeling of the IGBT using the Hammerstein configuration, IEEE Transactions on Power Electronics 11 (6) (1996) 746−754.

7 Hardware-in-the-loop real-time simulation of power electronic systems

7.1 Introduction

Simulations have long been used to support the design, analysis, and validation of power electronic systems, especially when dealing with complex topologies or high-power applications. Although nonreal-time offline simulations are useful to verify control strategies and converter topologies in the early design stages, real-time simulations allow interfacing the simulated models with physical components, constituting a more complete environment for testing of power electronic systems, useful throughout the entire development cycle. Hardware-in-the-loop (HIL) real-time simulation is one of the real-time tools used from the early development stages and the performance testing of the power electronic converter and its controllers all the way to the commission maintenance throughout the full life cycle. In the HIL simulation, the behavior of the power electronics is simulated in a digital real-time simulator (DRTS), which interacts with one or more devices under test (DUTs) through a signal or power interface. Through the HIL-based real-time simulations, various testing scenarios can be safely taken to evaluate the performance of the DUT before connecting it to the real physical system without the risk of equipment damage. The result is a significant reduction in development time and cost, besides the increased flexibility, safety, and reproducibility. In addition the HIL allows for automatic testing procedures, which provide much wider test coverage than manual tests combined with time and cost savings.

HIL real-time simulations must comply with strict time constraints imposed by the real-time computing and interfacing requirements, which imposes significant challenges in the design and use of such a tool. To develop or use a HIL platform, it is inevitable to consider the interaction effects among three basic components: DUT, DRTS, and the signal/power interface. To give an overall understanding to the reader, this chapter first discusses the time constraint in the HIL simulation and then discusses the interface issues in HIL.

7.2 Time constraints of hardware-in-the-loop real-time simulation

A general structure for the HIL simulation is illustrated in Fig. 7.1. Depending on the DUT, there are two types of HIL simulations. If the DUT is a controller, the

Real-Time Simulation Technology for Modern Power Electronics. DOI: https://doi.org/10.1016/B978-0-323-99541-2.00001-6

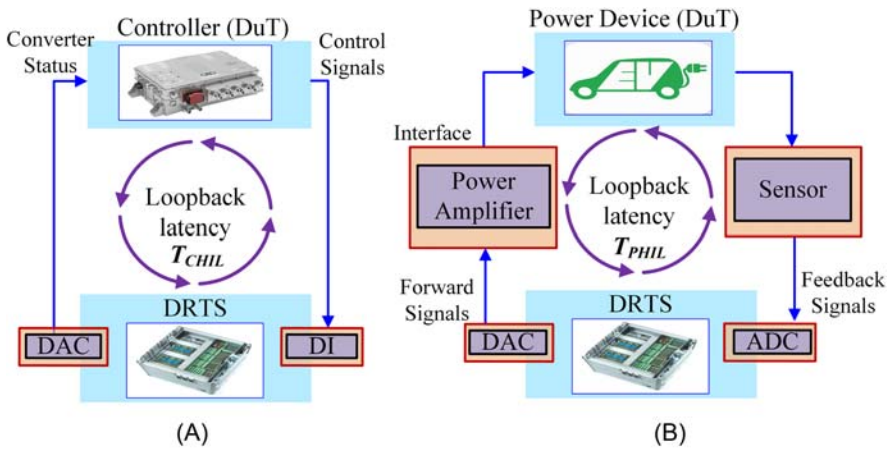


Figure 7.1 Example of signal flow in hardware-in-the-loop systems: (A) signal flow in CHIL; (B) signal flow in PHIL.

type of HIL is the controller HIL (CHIL), as shown in Fig. 7.1A. If the DUT encompasses actual power hardware, like an electrical device that is a part of a power electronic system, the type of HIL is the power HIL (PHIL), as shown in Fig. 7.1B. The main components of the HIL include the DUT, digital real-time simulator (DRTS), and signal/power interface. DRTS must perform the following three processes to interact with the connected DUT:

1. Read the signals from the DUT output;
2. Determine the system response and variables of interest based on the implemented real-time model and the model inputs;
3. Output the numerical results and convert them to the output signals for DUT.

In either CHIL or PHIL the loop-back latency is roughly the sum of the time cost of the above three steps. The loop-back latency is a critical index deciding the overall performance of the HIL real-time simulation. Since the latency is introduced to the simulator, it can lead to different DUT behaviors compared to the real system due to wrongly perceived simulated system dynamics. This is why it is important to minimize loop-back latency and keep it well below the transient period of the actual system. Moreover, inside the DRTS, the model computation must be completed within the simulation time step, otherwise the model cannot be simulated in real time.

7.2.1 Time constraint in CHIL

In CHIL, as shown in Fig. 7.1, the DRTS receives the control signals from the controller and simulates the power electronic behavior, while the controller samples the simulated power electronic variables from the DRTS and generates the control signals. The controller is connected to the simulated system through the I/O ports of

DRTS (DAC/ADC/DIO) known as the signal interface. Since the DRTS and the controller are both discrete-time systems, the loop-back latency T_{CHIL} in the CHIL is the key factor in deciding the system's stability and accuracy.

Firstly, the DRTS samples the controllers' signals (for example PMW signals) that feed the real-time model. A latency T_{in} equal to the sampling time is introduced. The model then computes within a fixed simulation time step T_s, and the idle process compensates for the difference between the actual computing time and the simulation time step. At last, the simulation output is converted to the analog output with a settling time T_{out}. The loop-back latency T_{CHIL} is thus equal to the sum of these three latencies, as indicated in Fig. 7.2A.

$$T_{\mathrm{CHIL}} = T_{\mathrm{in}} + T_s + T_{\mathrm{out}} \tag{7.1}$$

The time frame of the CHIL is shown in Fig. 7.2B. The sampling theorem states that to present the correct signal form, the highest frequency in a signal should be less than half the sampling frequency, but practical evidence prefers that the sampling frequency should be at least tens of times higher than the signal so that the results with acceptable accuracy in the time domain can be obtained for most cases. Let us consider one PWM control period T_{pwm} of the power electronics; the loop-back latency T_{CHIL} should be at least N times less than T_{pwm}, as given in Eq. (7.2).

$$T_{\mathrm{CHIL}} \leq \frac{1}{N} \cdot T_{\mathrm{pwm}} \tag{7.2}$$

The model computing time T_s satisfies Eq. (7.3). The sampling time T_{in} and settling time T_{out} are decided by the hardware of the DRTS, and the sampling and settling processes are designed in an oversampling manner to improve the signal resolution. The loop-back latency T_{CHIL} is thus subject to the computing speed of the power electronic model.

$$T_s = T_{\mathrm{CHIL}} - T_{\mathrm{in}} - T_{\mathrm{out}} \leq \frac{1}{N} \cdot T_{\mathrm{pwm}} - T_{\mathrm{in}} - T_{\mathrm{out}} \tag{7.3}$$

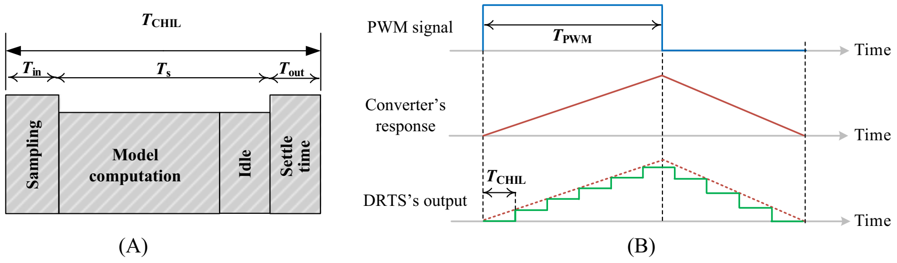


Figure 7.2 Data communication in CHIL. (A) The composition of CHIL loopback latency and (B) the PWM responses of the DRTS.

It needs to mention that the choice of N determines the simulation resolution, which depends on the desired real-time simulation fidelity. The fidelity varies according to the application and the responses of interest as well as with the control and modulation strategies being implemented. For example, if we consider $T_s = 500\text{ns}$, for the relationship of $N = 10$ to hold, the maximum switching frequency would be theoretically limited to 200 kHz. However, in this case the PWM resolution is only 10% of duty cycles. Assuming $T_s = 500\text{ns}$ again the PWM frequency would need to be limited to around 2 kHz in order to achieve a PWM resolution of at least 10 bits ($N = 1024$).

7.2.2 Time constraint in PHIL

As shown in Fig. 7.1B, in PHIL, parts of the power electronic system are simulated in the DRTS, while other parts consist of the actual power hardware in the loop. Unlike CHIL, a power interface is thus required for interacting with the virtually simulated system with the real Power-DUT (P-DUT). A power source or sink (connected through the PHIL interface) is needed for this setup, which will either generate or absorb power. Reference signals generated in the DRTS are sent to the power interface equipped with power amplifiers, in which the matched voltages and currents to the P-DUT are produced. Meanwhile, signals from the DUT are sensed and sent to the DRTS as feedback signals so that the loop is closed. A filter module is usually required to filter the signals captured from the DUT. This configuration allows testing electrical devices at full power.

In the PHIL, the P-DUT is a physical system, while the DRTS is a discrete-time simulator. The loopback latency T_{PHIL} in PHIL applications includes both the time needed for model simulation ($T_{\text{in}} + T_s + T_{\text{out}}$) in the DRTS and the time latency introduced by the power amplifier $T_{\text{Amp.}}$. $T_{\text{Amp.}}$ covers a large proportion of PHIL loopback latency T_{PHIL} such that

$$T_{\text{PHIL}} = T_{\text{in}} + T_{\text{out}} + T_s + T_{\text{Amp.}} \tag{7.4}$$

The signal interface in PHIL is more complicated than in CHIL since the signal exchange between the P-DUT and DRTS involves a power interface. In addition to long loopback latency T_{PHIL}, the problems of noise and a limited bandwidth will also be prominent, and these nonideal effects can result in closed-loop stability issues.

7.3 Interface issues in hardware-in-the-loop

7.3.1 Interface issues in CHIL

In the HIL simulation the real-time simulator receives the gate signals from the controller and simulates the power electronic behaviors, while the controller samples the output signals from the real-time simulation and generates the gate signals to close the loop. The frequency of the gate signals is very high, up to hundreds of kilohertz. However, if the loopback latency of CHIL is not small enough, the number of samples that the DRTS can get in a switching cycle will be small, referring to Eq. (7.3). Therefore sampling errors will occur on the model digital inputs. It is the different rates of the DRTS sampling of the model digital inputs (as same as RT model execution rate) and the digital port signals (usually have higher sampling rate than model one, known as "oversimpling") that dominate the interface issues in CHIL.

7.3.1.1 Sampling rate effects on digital ports

The outputs of the power electronic controller are discrete gate signals, which control the switches in the power converter. Correspondingly, these signals are the inputs for the real-time simulator. In the real-time simulation, the power converter models are solved with a fixed time step. In the view of the models, the gate drive signals are sampled with a sampling period equal to the simulation time step, although the real sampling period of the DIO port is much shorter than the simulation time step. Therefore the shorter the simulation time step is, the closer the sampled gate signals will be to the actual signals. However, in the real-time simulation the simulation time step ranges from hundreds of nanoseconds to the microsecond level, and it is still impossible to precisely identify the instant when the gate signals change from one state to another. Switching could happen between two simulation steps to induce the interstep switching events which cannot be captured by the real-time model in the exact time. When the switch numbers are high, multiple interstep switching events could be induced. The gate signals captured in the real-time simulation will be deviated from the original signals due to the interstep switching events.

Let us take the commonly used pulse width modulation (PWM) signals in a power electronic converter as an example to illustrate the interstep switching events. Fig. 7.3 illustrates interstep events for one switch when the real-time simulation samples PWM signals from the controller. The size of the error area is related to the switching frequency and the sampling clock of the real-time simulation (simulation time step). The errors will reduce the accuracy of the simulation results or in the worst case jeopardize the simulation stability. As a result the error area caused by interstep events needs to be compensated by the algorithms.

The compensating algorithm usually uses two typical numerical methods, that is, double interpolation method and interpolation-extrapolation method, to reconstruct the original signals from the controller in real-time simulation [1]. They are proved to be suitable for accounting of those events accurately and efficiently in the simulation of power electronic systems.

7.3.1.2 Compensation algorithm

As shown in Fig. 7.4, assume that time point t_1 and time point t_2 are two successive simulation points, and the interstep switching events happen at

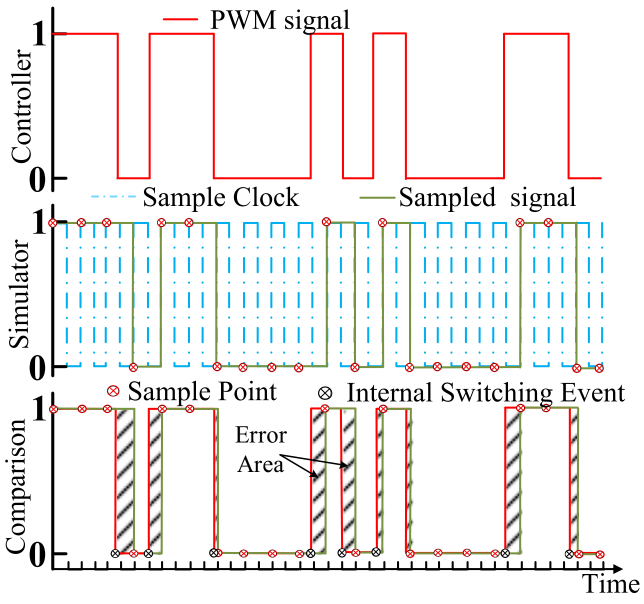


Figure 7.3 Interstep switching events.

t_a between t_1 and t_2. The power electronic converter simulated can be modeled by Eq. (7.5).

$$\dot{y} = \begin{cases} f_{\text{on}}(\text{t}), \text{On state} \\ f_{\text{off}}(\text{t}), \text{Off state} \end{cases} \tag{7.5}$$

If no compensation algorithm is used, the result at t_2 will be computed by turn-on models $\dot{y} = f_{\text{on}}(\text{t})$ expressed by Eq. (7.6) if the Euler method is used, which delays the switching event for a certain time, as indicated in Fig. 7.4.

$$y(t_2) = y(t_1) + h \cdot f_{\text{on}}(t_2) \tag{7.6}$$

If the double interpolation method is used, three steps will be taken to compensate for the interstep switching event.

1. Estimate the results $y(t_a)$ at the switching time point t_a using interpolation in Eq. (7.7).

$$y(t_a) = y(t_1) + \frac{y(t_2) - y(t_1)}{h}(t_a - t_1) \tag{7.7}$$

2. Compute the result at t_b by the turn-off model $\dot{y} = f_{\text{off}}(\text{t})$, which is one simulation time step forward from t_a, as in Eq. (7.8).

$$y(t_b) = y(t_a) + h \cdot f_{\text{off}}(t_b) \tag{7.8}$$

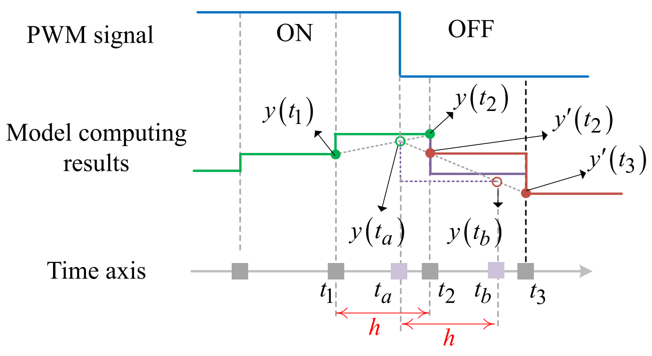


Figure 7.4 Illustration of the interpolation and extrapolation compensation algorithms.

3. Estimate the compensated results at t_2 by interpolating between $y(t_a)$ and $y(t_b)$ as in Eq. (7.9).

$$y'(t_2) = y(t_a) + \frac{y(t_b) - y(t_a)}{h}(t_2 - t_a) \tag{7.9}$$

For the interpolation-extrapolation method the difference lies in the third step. Instead of computing $y'(t_2)$ the extrapolation method is used to directly compute the results at t_3 without revising the results at t_2, as given in Eq. (7.10). Therefore, the interpolation-extrapolation method omits an interpolation computation of $y'(t_2)$, the computation amount is reduced while the accuracy is compromised as well. Noticing that both the double interpolation method and interpolation-extrapolation method rely on the accurate acquisition of the time stamp t_a to indicate when the switching event occurs. The t_a can be obtained by using the PWM oversampling technology, where the PWM is sampled by a period much shorter than the simulation time step. The exact switching time is stamped and used for the compensate the interstep switching events.

$$y'(t_3) = y(t_a) + \frac{y(t_b) - y(t_a)}{h}(t_3 - t_a) \tag{7.10}$$

7.3.2 Interface issues in PHIL

Ideally the power interface in the PHIL shown in Fig. 7.5 should have unity gain with an infinite bandwidth and no time delay, but that is impossible to achieve in practice. The power interface between the DRTS and Power-DUT introduces significant propagation latency and nonlinear characteristics, which can cause inaccuracy and even instability in the PHIL simulation. The discussion of interface issues can be divided into two parts: the power amplifier and the interface algorithms (IAs).

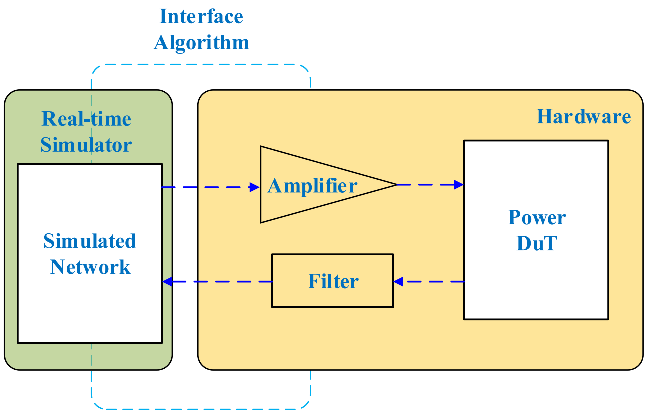
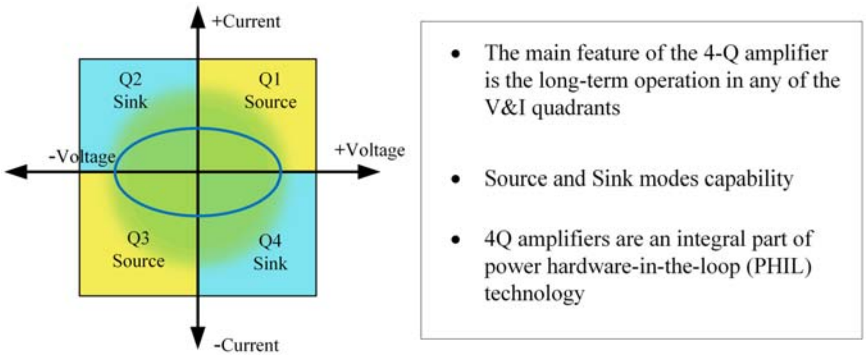


Figure 7.5 The power interface between the DRTS and Power-DUT.



- The main feature of the 4-Q amplifier is the long-term operation in any of the V&I quadrants
- Source and Sink modes capability
- 4Q amplifiers are an integral part of power hardware-in-the-loop (PHIL) technology

Figure 7.6 Four-quadrant power amplifier.

7.3.2.1 Power amplifiers

A power amplifier is an electronic circuit whose output is proportional to its input. It is used to provide and absorb actual power from the P-DUT based on the references provided by the DRTS. Linear power amplifiers and switched-mode power amplifiers are two basic types. Fig. 7.6 illustrates a four-quadrant (4-Q) power amplifier, capable of providing bidirectional power flow.

Linear amplifiers provide continuous output voltage with high accuracy, a wide bandwidth, and a very small time latency but have a low efficiency, which limits the output power and is not suitable for high-power applications. The switched mode amplifier can benefit from its high efficiency in the high-power PHIL simulation but has a limited bandwidth, excessive latencies, and electromagnetic interference problems compared to the linear amplifiers. A brief summary of linear amplifier and switched mode amplifier is given in Table 7.1.

Table 7.1 Summary of power amplifiers.

Technology	Linear	Switched mode
Efficiency	Low	High
Complexity	Low	High (additional control circuits required)
Types	• Current amplifier. • Voltage amplifier.	• Switch-mode amplifier. • Synchronous generator. • Boost amplifier.
Frequency range (FR)	Wide FR and short I/O delay.	Narrow FR and high I/O delay
Cost	High	Relatively low
Application range	• Amplification of voltage and/or current. • Kilowatt power range.	• Amplification of voltage and/or current. • Megawatt power range.

Depending on the application, amplifiers can be used for AC or DC operation. For DC operation the input signal is directly coupled to the amplifier. For AC operation, filters are used to eliminate the DC components in the input signal. Amplifiers with DC or AC operation options provide the extended capability for various PHIL applications.

Although the utilization of the FPGA technology (which will be presented in Chapter 9) has made the DRTS time steps smaller than 1 μs, the latency in the amplifiers is still large (usually > 40 μs). A state-of-art power amplifier, even with high-speed digital connection ports, still has a latency of 20 μs range. To reduce the influence of this large loopback latency, the interface algorithms (IAs) are essential to ensure that the PHIL of the power electronic system is accurate and stable.

7.3.2.2 Interface algorithms

To illustrate the IAs, here we use an equivalent circuit representation expressing the model in the simulator and the physical power hardware connected via a power amplifier. At any time step, the simulator can be expressed as a Thevenin or Norton equivalent circuit. As shown in Fig. 7.7, $Z_{P-\mathrm{DUT}}$ is the equivalent impedance of DUT and Z_S is the equivalent impedance of the simulated power electronic system in DRTS.

The transfer in the interface can be expressed by the mathematical representation. Ideally, for the v_1, i_1 in the simulator and v_2, i_2 in the hardware, we have the following relationship without latencies, as given in Eq. (7.11).

$$\begin{cases} v_1(t) = v_2(t) \\ i_1(t) = i_2(t) \end{cases} \tag{7.11}$$

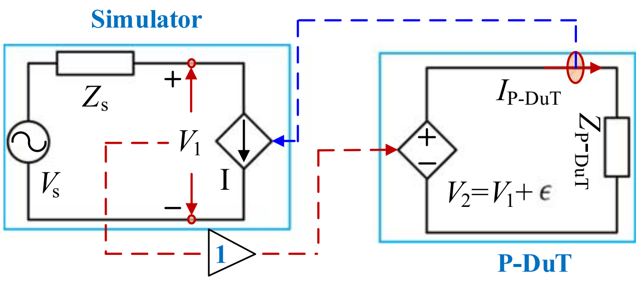


Figure 7.7 The ideal interface algorithms with Thevenin circuit representation.

However, in real cases, we have to consider the latencies in the interface. Therefore Eq. (7.11) should be rewritten as Eq. (7.12) with the latencies (T_{d1}andT_{d2}) and error (ε) caused by latencies.

$$\begin{cases} v_2(t) = v_1(t - T_{d1}) + \varepsilon \\ i_1(t) = i_2(t - T_{d2}) \end{cases} \tag{7.12}$$

If an error ε is introduced in voltage v_2, an error equal to $\frac{\varepsilon}{Z_{P\text{-DUT}}}$ will be introduced to current i_2 as well. Since i_2 will be fed back to the simulator, an error will be further induced to v_1 that is equal to $-\frac{Z_S}{Z_{P\text{-DUT}}}\varepsilon$. A new value v_1 will have an error that is equal to the previous error ε in v_2 amplified by a factor of $-\frac{Z_S}{Z_{P\text{-DUT}}}$ [2]. If $\frac{Z_S}{Z_{P\text{-DUT}}} > 1$, the error will be increasingly magnified, which could lead to the PHIL instability.

Eq. (7.12) can be further shown in Fig. 7.8, where ΔT_{d1} is the latency from the simulator to DUT, ΔT_{d2} is the latency from DUT to the simulator, and $T_{\text{amp}}(s)$ and $T_{\text{filter}}(s)$ are the transfer functions of the amplifier and the filter, respectively.

To evaluate the stability of the interface algorithm in Fig. 7.8, we could further develop the open-loop transfer function in Eq. (7.13), where ΔT_d is the sum of the time delays introduced by the DRTS and interface devices. With the transfer function, the stability issue with different system frequencies can be evaluated by the Nyquist stability criterion [3].

$$G_{OL}(s) = \frac{Z_S}{Z_{P\text{-}DUT}} T_{amp}(s) T_{filter}(s) e^{-s\Delta T_d} \tag{7.13}$$

Fig. 7.9 summarizes several commonly used IAs. In this figure, Z_s represents the impedance of DRTS and $Z_{P-\text{DuT}}$ represents the impedance of P-DUT.

7.3.2.2.1 Ideal transformer method

Ideal transformer method (ITM) is the most used IA due to its simplicity. The transfer function for the ITM is shown in Eqs. (7.14) and (7.15) [2,4,5],

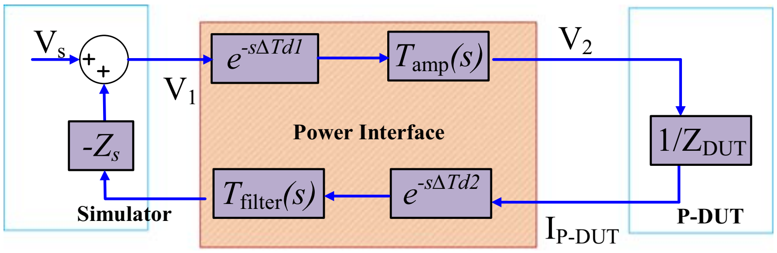


Figure 7.8 Block diagram of interfacing algorithm.

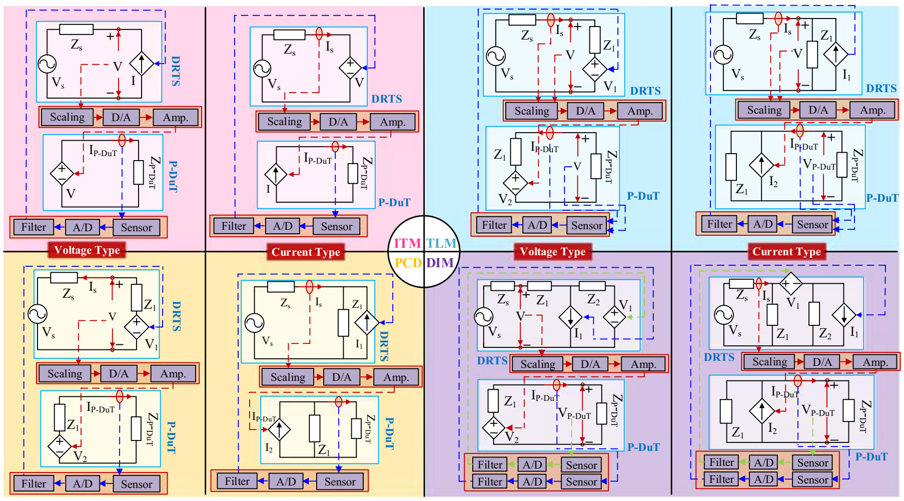


Figure 7.9 Different types of interface algorithms.

$$\text{Voltage type: } -\frac{Z_s}{Z_{P-\text{DUT}}} T_{\text{amp}}(s) \cdot T_{\text{filt}}(s) e^{-s\Delta T_d} \tag{7.14}$$

$$\text{Current type: } -\frac{Z_{P-\text{DUT}}}{Z_s} T_{\text{amp}}(s) \cdot T_{\text{filt}}(s) e^{-s\Delta T_d} \tag{7.15}$$

7.3.2.2.2 Transmission line method

Transmission line method (TLM) describes the power interface as a linking inductor or capacitor using the Bergeron transmission line model, which is expressed as an equivalent Norton circuit or Thevenin circuit [2,6]. Compared with ITM, TLM can consider the response time of PHIL by introducing the length of the transmission line as a parameter. The minimum length of the equivalent transmission line model should satisfy the criteria in Eq. (7.16),

$$\text{length}_{\min}(\text{km}) = \frac{T_{\text{PHIL}}}{\sqrt{L(H/\text{km})C(H/\text{km})}} \tag{7.16}$$

The transfer function for the ITM is shown in Eqs. (7.17) and (7.18),

$$\text{Voltage type: } -\frac{1-\beta e^{-2s\Delta T_d}}{1+\beta e^{-2s\Delta T_d}} \cdot \frac{Z_s}{Z_1} T_{\text{amp}}(s) \cdot T_{\text{filt}}(s) \tag{7.17}$$

$$\text{Current type:} \frac{1-\beta e^{-2s\Delta T_d}}{1+\beta e^{-2s\Delta T_d}} \cdot \frac{Z_s}{Z_1} T_{\text{amp}}(s) \cdot T_{\text{filt}}(s) \tag{7.18}$$

where
$\beta = Z_s - Z_1/Z_s + Z_1, Z_1 = \Delta T_d/L \; or \quad Z_1 = C/\Delta T_d, \text{V}_1 = \text{V}_{\text{DUT}}(t - \Delta T_d) + Z_1 \text{I}_{\text{DUT}}$ $(t - \Delta T_d), \text{V}_2 = \text{V}(t - \Delta T_d) + Z_1 \text{I}_s(t - \Delta T_d)$.

Since solutions of the equivalent Norton circuit or Thevenin circuit of the TLM adopt trapezoid integrations, TLM is numerically stable. However, this model introduces a fictive transmission line loss. Moreover the linking inductance or capacitance has to be appropriately sized as long as there is a change in the circuit configuration or the response time of the PHIL simulation.

7.3.2.2.3 Partial circuit duplication

In ITM, the stability is decided by the impedance ratio. It can be concluded that a large value of the linking impedance helps to improve the stability of the interface [6]. Based on this idea, partial circuit duplication (PCD) places linking impedance Z_1 in the simulated system and the hardware side. The transfer function for the PCD is shown in Eqs. (7.19) and (7.20),

$$\text{Voltage type: } -\frac{Z_s \cdot Z_{P-\text{DuT}}}{(Z_s + Z_1)(Z_{P-\text{DuT}} + Z_1)} \cdot T_{\text{amp}}(s) \cdot T_{\text{filt}}(s) \cdot e^{-s\Delta T_d} \tag{7.19}$$

$$\text{Current type: } -\frac{Z_1^2}{(Z_s + Z_1)(Z_{P-\text{DuT}} + Z_1)} \cdot T_{\text{amp}}(s) \cdot T_{\text{filt}}(s) \cdot e^{-s\Delta T_d} \tag{7.20}$$

Although the stability is increased, the fictive power losses and increased model inaccuracies are the side effects limiting the development of the PCD method.

7.3.2.2.4 Damping impedance method

In order to reduce the side effects of the large impedance introduced in the PCD [7], damping impedance method (DIM) includes a damping impedance Z_2. The transfer function for the DIM is shown in Eqs. (7.21) and (7.22),

$$\text{Voltage type: } -\frac{Z_s(Z_{P-\text{DuT}} - Z_2)}{(Z_1 + Z_{P-\text{DuT}})(Z_s + Z_2 + Z_1)} \cdot T_{\text{amp}}(s) \cdot T_{\text{filt}}(s) \cdot e^{-s\Delta T_d} \tag{7.21}$$

Table 7.2 Summary of interface algorithms.

Interface algorithms method	Stability	Complexity	Power loss	Accuracy
Ideal transformer method	Low	Low	Medium	High
Transmission line method	High	High	High	Low
Partial circuit duplication	Medium	Medium	High	Medium
Damping impedance method	High	Medium	Low	High

$$\text{Current type: } -\frac{Z_1^2(Z_{P-\mathrm{DuT}}-Z_2)}{(Z_1+Z_{P-\mathrm{DuT}})(Z_sZ_1+Z_2Z_1+Z_1Z_s)}\cdot T_{\mathrm{amp}}(s)\cdot T_{\mathrm{filt}}(s)\cdot e^{-s\Delta T_d} \tag{7.22}$$

Based on the transfer functions in Eqs. (7.21) and (7.22), it can be observed that the absolute stable condition (transfer functions becoming zero) occurs when Z_2 equals to $Z_{P-\mathrm{DuT}}$. Under this condition, the error induced in the current step will not transmit into the next simulation step. However, limited by the nonlinear characteristics of the P-DUT, it is not easy to acquire the exact value of $Z_{P-\mathrm{DuT}}$.

Table 7.2 summarizes the specifications of the previously presented four IA methods in terms of stability, complexity, power losses, and accuracy, which provide a direct view for users to pick up the most appropriate method in implementing the PHIL simulation of the power electronic system.

In this chapter, we have presented the HIL simulation associated with its time constraints and the interfacing issues for both the CHIL and PHIL applications. The specific types to implement the real-time simulation (processor-based and FPGA-based) will be further detailed in the next two chapters.

References

[1] F. Li, Y. Wang, Y. Huang, Y. Liu, X. Zhang, M. Mingyao, Review of real-time simulation of power electronics, Journal of Modern Power Systems and Clean Energy 8 (4) (2020) 796–808.

[2] W. Ren, M. Steurer, T.L. Baldwin, Improve the stability and the accuracy of power hardware-in-the-loop simulation by selecting appropriate interface algorithms, IEEE Transactions on Industrial Electronics 44 (4) (2008) 1286–1294.

[3] O. Nzimako and R. Wierckx, "Stability and accuracy evaluation of a power hardware in the loop (PHIL) interface with a photovoltaic micro-inverter," in: Proceedings of the Forty-first Annual Conference of the IEEE Industrial Electronics Society, 2015, pp. 005285-005291.

[4] M. Steurer, C.S. Edrington, M. Sloderbeck, W. Ren, J. Langston, A megawatt-scale power hardware-in-the-loop simulation setup for motor drives, IEEE Transactions on Industrial Electronics 57 (4) (2010) 1254–1260.

[5] N.D. Marks, W.Y. Kong, D.S. Birt, Stability of a switched mode power amplifier interface for power hardware-in-the-loop, IEEE Transactions on Industrial Electronics 65 (11) (2018) 8445−8454.
[6] S. Goyal, G. Ledwich, A. Ghosh, Power network in loop: a paradigm for real-time simulation and hardware testing, IEEE Transactions on Power Delivery 25 (2) (2010) 1083−1092.
[7] A. Riccobono, E. Liegmann, M. Pau, F. Ponci, A. Monti, Online parametric identification of power impedances to improve stability and accuracy of power hardware-in-the-loop simulations, IEEE Transactions on Instrumentation and Measurement 66 (9) (2017) 2247−2257.

Processor-based real-time simulation of power electronic systems

8

8.1 Introduction

Hardware-in-the-loop real-time simulation can greatly speed up power electronics development by providing a flexible, low-cost, and safe environment that enables to investigate the characteristics and validate the performance of the physical control and/or power devices in the loop with the simulated plant. However, practical implementation of power electronics real-time simulation can be very challenging due to the strict time constraints of the model computation. The computing unit must finish the model computation within a time interval smaller than the simulation time step. Power electronic converters usually have high switching frequencies which make the simulation time step very short, below 1μs for example. Therefore the model computing speed should be fast enough.

As presented in Chapter 5, the core of the real-time model computation is to solve a system of linear equations that represent the power electronics behaviors in discrete time. It is obvious that the scale of the linear equations is determined by the topology under simulation, and the required computating capacity will grow significantly when the scale increases. For example the number of multiplication/division operations in the Gaussian elimination method is about $\frac{n^3}{3}$, where n is the scale of the linear system. Moreover, when considering the nonlinearities in the power electronics, the iterative algorithm will aggravate the computing burden in each simulation time step. Therefore the computing burden can be significant and the time constraint can be very hard to meet, especially for the modern power electronic system with complex topologies and high switching frequencies.

The successful implementation of real-time simulation relies on the computing capacity of the simulator. There are two mainstream types of computing hardware in the real-time simulation, that is, the processor and the field programmable gate array (FPGA).

The processor-based real-time simulation uses general computational hardware resources to compute the model. The common processors used in real-time simulations are the RISC processor, the central processor unit (CPU), and the digital signal processor (DSP). The processor typically uses float-point numerical representations and can be used for the real-time simulation of power electronic systems with relatively low dynamics and hence relatively large simulation timestep requirements, typically in the range from tens of microseconds to hundreds of microseconds.

Real-Time Simulation Technology for Modern Power Electronics. DOI: https://doi.org/10.1016/B978-0-323-99541-2.00009-0

In addition, multiple processors can work together to constitute a paralleled computing structure that is efficient in dealing with large-scale power electronic systems, such as the high voltage direct current transmission (HVDC) system.

Another type of computational hardware is the FPGA, in which the user-defined logic can be designed using either the low-level HDL approach (Verilog HDL and VHDL) or high-level approaches. The intrinsic parallel structure of logic resources permits massively paralleled data processing to reduce the real time simulation time step down to the nanosecond level, which is ideal for the real time simulation of power electronics, especially for the high switching-frequency converters. Shortcomings of the FPGA-based approach are the tedious process of programming that requires extensive specialist knowledge and needs a long time to build the FPGA hardware design. Additionally, FPGA-based simulations have lower accuracy in numerical representation, where the usage of fixed-point and low-precision floating-point arithmetic are quite common.

It is also possible to combine the processor- and the FPGA-based approaches and thus leverage the advantages of both platforms in their respective domains. For example the slow dynamics can be simulated in CPU with a larger time step, and the fast dynamics can be simulated in FPGA with a smaller time step. However, this hybrid approach comes at the cost of model partitioning and additional efforts to manage synchronization and data exchange in such a time-critical heterogeneous system. Hence the choice of the hardware platform should be made based on the specific simulation requirements and compromise between accuracy, speed, and scale, to get the best performance of the real-time simulation.

Although power electronic models presented in Chapters 3 to 6 lay the mathematical ground to implementing the real-time simulation, the details related to the hardware implementation of these models on processors or FPGA are still to be discussed. In practice the most important parameter in a real-time simulation is the simulation time step. It should be chosen carefully because from the modeling perspective, it determines the discretization accuracy of the model, and from the numerical solving point of view, it determines the numerical stability. In a real-time simulation, the simulation time step is directly constrained by the hardware computing capacity, meaning that the simulation time step cannot be smaller than the minimum computing time required for one model simulation step. The accuracy, stability, and hardware computing capacity are thus tightly coupled by the simulation time step, which strongly affects the hardware implementation of the models. This chapter focuses on the processor-based real-time simulation, and Chapter 9 is devoted to the FPGA-based real-time simulation of power electronic systems.

In this chapter the hardware structure of a processor-based simulator is first given. Then the model implementation method in the real-time target processor is discussed. Afterward the choice of the numerical method is discussed using an R-L circuit as an example. Moreover the design process of implementing a power converter model in a real-time simulation is presented in an example by implementing a AC-DC-AC power electronic system in a DSP. The general procedures, the involved problems, and the potential solutions are also discussed.

8.2 Hardware structure of the processor-based commercial real-time simulator

Nowadays the real-time simulators of power electronics are well developed, and various types of commercial simulators are available for users to select based on their specific needs. The model computing hardware of the simulators can be a processor, FPGA, or both. In this section, we will give a brief introduction to the processor-based real-time simulators. It needs to be mentioned that the processor-based real-time simulator could also be equipped with additional FPGA-based computing unit for co-simulation purpose. But only the processor-based hardware structure will be focused in this chapter.

Although the hardware structure of commercial products shows different capability and performance, there are still some common characteristics that can be summarized as follows:

1. The simulation tasks can be separated and implemented in multiple cores in processor or multiple processors and run in real time.
2. The host computer installed with the real-time simulation software generates the code-based mathematical model of the power electronic system and involves compilation tools to compile them into executable codes. The codes are then downloaded into the target processor. The host computer also serves as the control, monitoring, and analysis center of the real-time simulation.
3. The peripheral interfaces (I/O ports, DAC/ADC ports, communication ports) are designed to be reconfigurable and may use the FPGA board to control the signal flows and process the data preliminarily.
4. High-speed data buses such as PCIe are equipped to ensure data exchanges between multiple processors, between the processor and FPGA I/O, and in some cases, between the processor and the host computer.

Here, we will take the commercial real-time simulators from RTDS and Opal-RT to illustrate some hardware structures.

8.2.1 Hardware structure of the RTDS

Fig. 8.1 shows one of the hardware architectures of the RTDS simulator [1], which utilizes rack units for parallel simulation. Each rack unit contains several processor cards called PB5 cards. A PB5 card is a computing unit to solve the modeled network and auxiliary components, and each PB5 card contains two Freescale MC7448 RISC processors, each operating at 1.7 GHz [2]. A maximum of six PB5 cards with 12 processors can be put into one rack to simulate a subsystem with 90 nodes. If larger networks with more nodes are simulated, multiple racks can be used to implement the simulation.

In addition to PB5, each rack contains a card called a GTWIF. It is used to build the Ethernet communications between the Host PC and racks, as well as the communications between racks. Besides, it provides a time-step clock to synchronize the operation of different processors. The real-time synchronization of different racks is implemented by the global bus hub. The GTIO card provides the connection to the external hardware via analog and digital I/O ports.

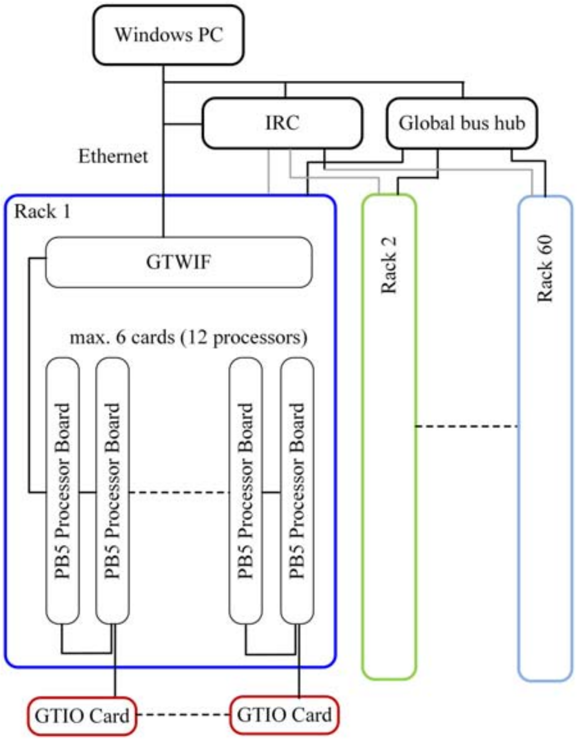


Figure 8.1 Hardware structure of RTDS [1].

8.2.2 Hardware structure of Opal-RT

Fig. 8.2 shows the hardware architecture of the Opal-RT simulator OP5707XG. In the processor-based part, it uses multicore multithread Intel CPUs to compute very large models in real time. Meanwhile, it has a Xilinx Virtex − 7 FPGA with a timer resolution of 5 ns to simulate fast models with a time step as low as 145 ns. Based on the specific need, the number of CPUs can be extended to enlarge the simulation scale.

Fig. 8.2 shows that the data exchanges between CPUs are implemented by the shared RAM. The Host PC connects the simulator via the Ethernet port. The CPU and FPGA are interacted by means of the PCIe bus. The simulator runs a real-time operating system called real-time OPAL-RT-Linux.

8.2.3 Hardware structure of digital signal processor-based real-time simulators

There are many customized real-time simulators built in many laboratories that came out of research projects for serving specific purpose. One of the customized structures is shown in Fig. 8.3. The main computing unit of the real-time simulation is the DSP (such as TMS320F28377D or TMS320F28335D from TI Company). In addition to the DSP component, FPGAs are used to provide timing, synchronization, and

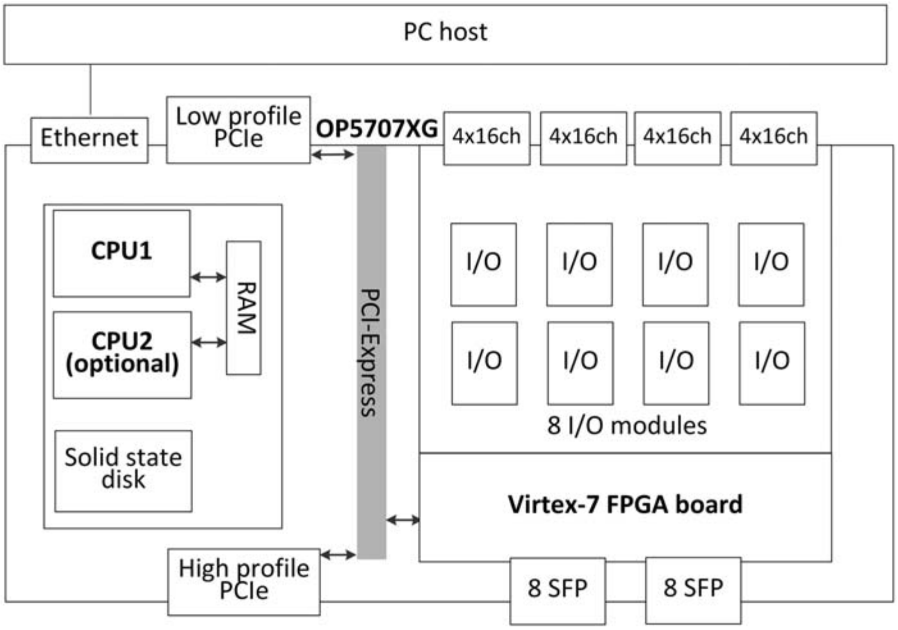


Figure 8.2 Hardware structure of Opal-RT [3].

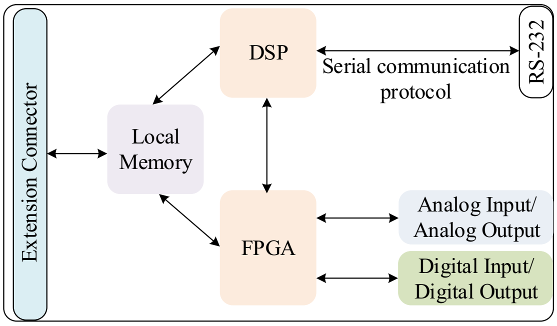

Extension Connector
DSP
Serial communication protocol
RS-232
Local Memory
FPGA
Analog Input/ Analog Output
Digital Input/ Digital Output

Figure 8.3 Hardware structure of the customized simulator.

coprocessing as well as communication to the attached AIO and DIO interfaces [1]. Although the computing power of DSP is very limited, for a small-scale power electronic converter simulation, it is a cost-effective choice.

8.3 Model implementation in real-time target processors

8.3.1 Preparing the mathematical model

Before the real-time implementations of the power electronic models, the models should be converted in the discrete-time domain. The discretization is implemented by the numerical method of differential equations such as the Euler method and the Trapezoid method as presented in Chapter 5. If the discretization is applied to the individual inductor and capacitor, the nodal analysis method can be used to model the converter in the discrete time. Meanwhile, if the state-space equation is used to represent the converter in the continuous time domain, the discretization can be applied to develop a discrete-time state-space model. Regardless of the method used, the power electronic converter model is simulated by computing its states at successive steps over a specified period, that is, the simulation time step. From the view of the discretized model, the time step will affect the discretization accuracy and the numerical stability.

Since the model computing process must be finished within the real time that the simulation time step corresponds to, the selection of the time step of the models should consider the computing power and potential computing latency of the target processor. As presented in Chapter 6, the solution of a discretized power electronic converter model depends on the solution of a linear equation system, which dominates the model computing latency. Possible paralleled modeling can be used to realize the paralleled computing in multiple cores of processors or in multiple processors, while the communication latency should be considered. Moreover, the simulation input, output, and history terms should be identified. The input and output variables will be interfaced with the I/O, and the history terms should be prepared as the persistent variables.

After obtaining the mathematical model for real-time simulation, the remaining tasks are (1) the coding on the target processor, (2) the offline verifications of the mathematical model, and (3) the validation in real-time simulation.

8.3.2 Coding

Users can use various kinds of programming languages at different levels to design the code for the model. It depends on the target and the compilation tools. To avoid errors in coding, the automatic code generator (ACG) tool can be used. The ACG tool could automatically generate the codes in different programming language, such as C++/C. By using compilation tools, the codes are compiled into executables that can be loaded on the target processor.

8.3.3 Offline verifications

The mathematical model as well as the associated codes need to be verified before its real-time implementation. First the accuracy of the power electronic converter model needs to be evaluated. Second the parameters need to be tuned to relfect the physical entities. Third the effectiveness and accuracy of the solving strategies need to be verified to ensure the correct representation of the power converter model in discrete time. Last the codes might need to be debugged to eliminate the error in the coding process.

To achieve these objectives, the codes can be firstly executed in offline mode to analyze the simulation results. The reference model is usually selected in this stage using offline simulation tools or experimental results. For example the Simulink model using the components from the Simpower system library can be used as the reference model for the system-level simulation of the power electronic converter, as demonstrated in previous chapters. A variable time-step simulation strategy can be used in Simulink to ensure the accuracy of the reference model. If the nonlinearities are involved, the iterative solving algorithm with a strict error criterion can be used to ensure the converged results of the reference model. The simulation tools such as Saber, PLECs, PSIM, and so on are all possible choices to be used to build reference model. By comparing the simulation results with the reference, the mathematical model, parameters, solving methods, and generated code can be verified comprehensively.

8.3.4 Real-time validations

The verified codes are executed in the target processor in real time with a fixed time step. In this stage the real-time constraint should be examined. If the execution time is larger than the simulation time step, overrun occurs and the time step needs to be reselected or the solving algorithm needs to be optimized. The possible paralleled computing can be also explored to speed up the simulation. If the iteration algorithm is involved, the maximum iteration numbers can be adjusted by balancing the convergency and the speed. In the meantime, a large simulation time step can also be tested to see whether the requirement of simulation accuracy is still met. If the execution time is much shorter than the simulation time step, a smaller simulation time step is preferred to gain in the accuracy and stability of the real-time simulation.

In addition to the time constraint, the results of the real-time simulation should be validated. In this step the offline simulation of the same codes can be selected as the reference. The errors will also be introduced after transplanting the model into the real-time target. For example the results of some overrun steps may not be accurate, which may affect the simulation stability. Moreover the iteration convergency may not be met when the maximum iteration number is reached, which will accumulate errors in the relevant variables. The last step is to validate the effectiveness of the real-time simulations by implementing the HIL testing to see whether it meets testing requirements. The HIL can also be put in the field tests and evaluated for its performance.

To summarize, the processor-based real-time simulation of power electronic converters usually contains the following steps:

1. Build the mathematical models;
2. Model adaption for the real-time simulation;
3. Code the model for target processor.
4. Verify the models in offline simulation;
5. Conduct the real-time simulation in the target hardware;
6. Validate the time constraints and accuracy of the real-time simulations.
7. Put the real-time simulation into the field HIL tests and evaluate its performance.

8.4 Case studies

In this section, we will present two case studies. In case study 1, we will illustrate the comparison among different numerical results using an RL circuit as an example. This is an offline simulation using Matlab®/Simulink. With this case, we can have an overall understanding of the simulation time step and the numerical solvers presented in previous chapters. In case study 2, we will build an AC-DC-AC power electronic system model for DSP-based real-time simulation. The procedures for converting the mathematical model to the real-time model are presented in detail, and the simulation results are demonstrated, with a special focus and discussion on numerical "chattering" issue induced by fixed discrete time step of the model.

8.4.1 Case study 1: the choice of the numerical method in an R-L circuit

As stated previously, the commonly used numerical discretization methods in the processor-based simulation are the forward Euler (FE) method, backward Euler (BE) method, trapezoid (TR) method, and Runge-Kutta 4th (RK4) method. Here, we give the choice of the numerical methods from the aspects of accuracy and stability using the R-L circuit as an example (Fig. 8.4).

The state-space equation of the R-L circuit is given in (8.1)

$$v(t) = \mathrm{L}\frac{di(t)}{dt} + \mathrm{R}i(t) \tag{8.1}$$

Using different numerical methods, the discrete formations of (8.1) are given from (8.2) to (8.5).

$$i_{\mathrm{FE}} = i(t+\Delta t) = \left(1 - \frac{R\Delta t}{L}\right)i(t) + \frac{\Delta t}{L}v(t) \tag{8.2}$$

$$i_{\mathrm{BE}} = i(t+\Delta t) = \left(\frac{1}{1+\frac{R\Delta t}{L}}\right)i(t) + \frac{\Delta t}{L}\frac{1}{1+\frac{R\Delta t}{L}}v(t+\Delta t) \tag{8.3}$$

$$i_{\mathrm{TR}} = i(t+\Delta t) = \left(\frac{2-\frac{R\Delta t}{L}}{2+\frac{R\Delta t}{L}}\right)i(t) + \frac{\Delta t}{L}\frac{1}{2+\frac{R\Delta t}{L}}[v(t+\Delta t)+v(t)] \tag{8.4}$$

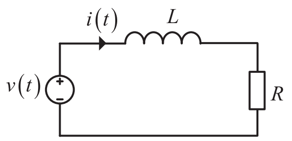


Figure 8.4 An RL circuit.

$$\begin{cases} k_1 = f(x_n, u_n) \\ k_2 = f(x_n + \Delta t k_1/2, (u_t + u_{t+\Delta t})/2) \\ k_3 = f(x_n + \Delta t k_2/2, (u_t + u_{t+\Delta t})/2)) \\ k_4 = f(x_n + \Delta t k_3, u_{t+\Delta t}) \\ i_{\mathrm{RK}} = x_n + (k_1 + 2k_2 + 3k_3 + k_4)\Delta t/6 \end{cases} \tag{8.5}$$

With a unit step-up of the input voltage $v(t)$ and the parameters of $R = 1\Omega, L = 0.05\mathrm{mH}$, the different results of the current $i(t)$ under different time steps are shown in Fig. 8.5 and the errors are shown in Table 8.1.

It can be seen from Fig. 8.5 and Table 8.1 that when the time step is less than $1\mu s$, the errors among the FE, BE, TR, and RK methods have little difference. With the increase of the time step, the FE method becomes unstable. On the contrary the BE method keeps stable while the accuracy decreases. Among these methods the RK4 method is a fourth-order method, which has a local truncation error of $\mathrm{O}(h^5)$ and the total accumulated error of $\mathrm{O}(h^4)$. A high-order numerical method usually leads to an advantage in accuracy, but it is more complex and time-consuming to implement in real-time simulation. Regarding the TR method, we can find that it shows good accuracy and stability when the time step increases. As a result, a rough experience-based estimation can be given for the processor-based real-time simulation of power electronics to choose the numerical methods. TR method and BE method could be considered as the first choices for model implementation with a time step at microsecond level. Moreover the FE method is hard to be used in the processor-based real-time simulation since the time step should be further reduced.

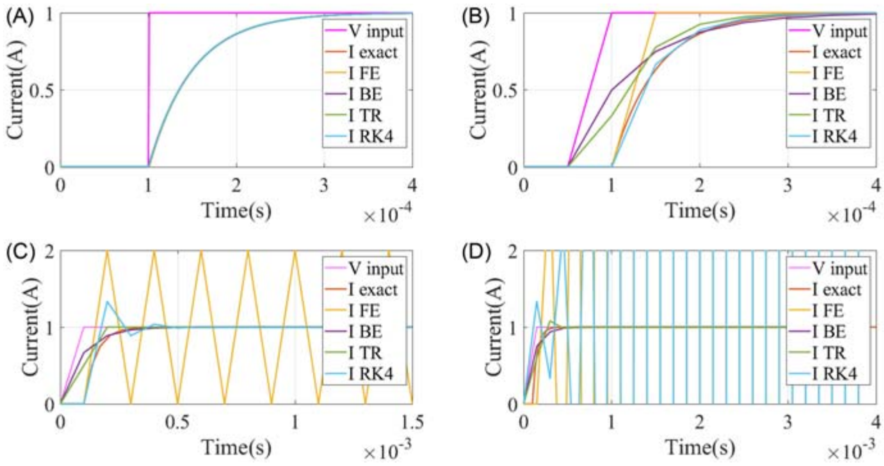


Figure 8.5 The unit step response results of the R-L circuit under different time steps. (A) 1μs time step, (B) 50μs time step, (C) 100μs time step, and (D) 150μs time step.

Table 8.1 The error comparison in different numerical methods at 50μs time step.

Time (μs)	I-exact (A)	Forward Euler		Backward Euler		Trapezoidal rule		Runge-Kutta	
		I-FE (A)	error %	I-BE (A)	error %	I-TR (A)	Error %	I-RK (A)	error %
100	0.6321	0.000	100.000	0.5000	20.9012	0.3333	47.2674	0.3750	40.6759
150	0.8647	1.000	15.6518	0.7500	13.2612	0.7778	10.0486	0.7656	11.4541
200	0.9502	1.000	5.2396	0.8750	7.9154	0.9259	2.5560	0.9121	4.0100
250	0.9817	1.000	1.8657	0.9375	4.5009	0.9753	0.6495	0.9670	1.4917
300	0.9933	1.000	0.6784	0.9688	2.4678	0.9918	0.1503	0.9876	0.5660
350	0.9975	1.000	0.2485	0.9844	1.3179	0.9973	0.0265	0.9954	0.2161
400	0.9991	1.000	0.0913	0.9922	0.6907	0.9991	0.0003	0.9983	0.0827
450	0.9997	1.000	0.0336	0.9961	0.3572	0.9997	0.0031	0.9993	0.0316

8.4.2 Case study 2: AC-DC-AC power conversion system

In this section a case study of the AC-DC-AC structure (shown in Fig. 8.6) is modeled and discussed using a customized DSP-based real time simulation.

8.4.2.1 Mathematical modeling

In this example, the switches are modeled by the switching function, which depends on the control gate signal. The gate signals are expressed as S_k^+ and S_k^- ($k = \{a, b, c, d, e\}$), and S_k^+ and S_k^- have complementary logic. $U_{\mathrm{an}}, U_{\mathrm{bn}} U_{\mathrm{cn}}, U_{\mathrm{dn}}, U_{\mathrm{en}}$ are voltages to node n, which can be computed by the switching functions in (8.6).

$$\begin{cases} U_{\mathrm{an}} = S_a^+ \times U_{C_d} \\ U_{\mathrm{bn}} = S_b^+ \times U_{C_d} \\ U_{\mathrm{dn}} = S_c^+ \times U_{C_d} \\ U_{\mathrm{dn}} = S_d^+ \times U_{C_d} \\ U_{\mathrm{en}} = S_e^+ \times U_{C_d} \end{cases} \tag{8.6}$$

Meanwhile the current through the switches can be represented by the switching function, and the dc link current can be computed by (8.7) and (8.8).

$$i_{dc1} = -\left(i_{S_d} + i_{S_e}\right) = \left(S_d^+ - S_e^+\right) \times i_{L_s} \tag{8.7}$$

$$i_{dc2} = i_{S_a} + i_{S_b} + i_{S_c} = S_a^+ \times i_{L_a} + S_b^+ \times i_{L_b} + S_c^+ \times i_{L_c} \tag{8.8}$$

The inductor current i_{L_s} and $i_{L_{(a,b,c)}}$ and the capacitor voltage U_{C_d} can be expressed by the state equations in (8.9)–(8.12).

$$\frac{di_{L_s}}{dt} = \frac{1}{L_s} \times (U_s - U_{\mathrm{dn}} + U_{\mathrm{en}} - R_s \times I_{L_s}) \tag{8.9}$$

$$\frac{di_{L_{(a,b,c)}}}{dt} = \frac{1}{L_{(a,b,c)}} \times (U_{(a,b,c)m} - R_{(a,b,c)} \times i_{L(a,b,c)}) \tag{8.10}$$

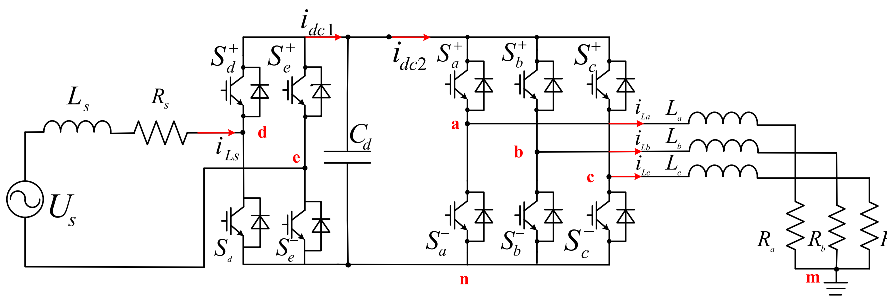


Figure 8.6 AC-DC-AC structure.

$$\frac{dU_{C_d}}{dt} = \frac{1}{C_d} \times (i_{dc1} - i_{dc2}) \tag{8.11}$$

$U_{(a,\mathrm{b,c})m}$ is computed by (8.12) under the balanced condition of three phases.

$$U_{(a,\mathrm{b,c})m} = U_{(a,\mathrm{b,c})n} + U_{\mathrm{nm}} = U_{(a,\mathrm{b,c})n} - \frac{1}{3}(U_{\mathrm{an}} + U_{\mathrm{bn}} + U_{\mathrm{cn}}) \tag{8.12}$$

The switching status of the forced commutated IGBT switch is determined by gate signals. As for the naturally commutating switch, such as diode, the switching status is determined by the voltage or current across it.

In this case study, there are no gate signals to the $[S_a^+, S_a^-, S_b^+, S_b^-]$, so they will operate as the uncontrolled rectifier with only diodes. The status of $[S_a^+, S_a^-, S_b^+, S_b^-]$ is then determined by the direction of current i_{L_s} at the last time step. The status of $[S_c^+, S_c^-, S_d^+, S_d^-, S_e^+, S_e^-]$ can be determined in a half-bridge environment, and the gate signals and current $[i_a, i_b, i_c]$ decide the turn-on or turn-off statuses of $[S_c^+, S_c^-, S_d^+, S_d^-, S_e^+, S_e^-]$.

8.4.2.2 Offline verification

In order to compare the effect of switching events on the different solvers, here we give the results of the forward Euler (FE) method and the improved Euler method (IEM). The SimPower System (SPS) model in Simulink is chosen to be the reference model. The simulation parameters are shown in Table 8.2.

Since the variable-step solver can provide more accurate results than the fixed-step simulation, the reference model is computed by the variable-step solver in Simulink. The block diagram of the Simulink is shown in Fig. 8.7.

As seen in the simulation results in Fig. 8.8, the chattering happens when the current i_{L_s} is close to zero. There is a significant error in either the FE method or IEM. Moreover, the chattering errors introduce potential instability and cause fictive power losses in simulation results.

Figs. 8.9 and 8.10 are the simulation results of the voltage U_{C_d} and current i_{L_a}. The IEM is closer to the SPS than the FE method since the IEM is to be a method of order 2 while FE is to be the method of order 1. Although the simulation results are very close, there are still differences between the built models and the reference model.

Table 8.2 Simulation parameters.

Symbol	**Value**	**Symbol**	**Value**
Time step	$20\mu s$	C_d	$2.3e^{-4}F$
L_s	$1e^{-3}H$	$R_\mathrm{a}, R_\mathrm{b}, R_\mathrm{c}$	1Ω
R_s	$1e^{-3}\Omega$	$L_\mathrm{a}, L_\mathrm{b}, L_\mathrm{c}$	$2.3e^{-3}H$

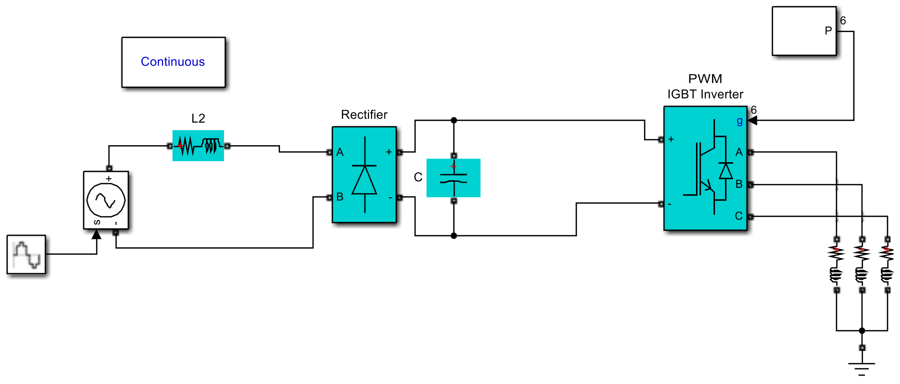


Figure 8.7 Reference model in SimPower system.

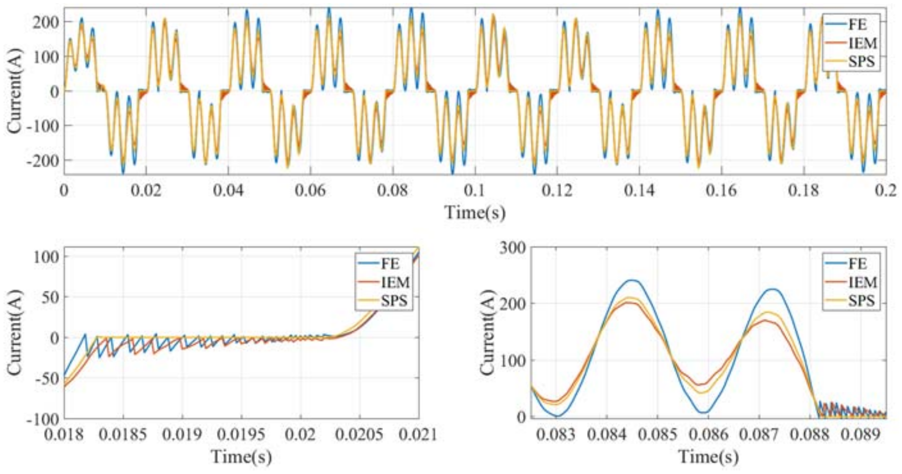


Figure 8.8 The simulation results of current i_{L_s}.

8.4.2.3 The chattering problem due to the blocking mode

To reduce the error, we have to deal with the chattering observed in Fig. 8.8. If we look deeper in the model, we can see that the chattering is actually caused by the incorrect identification of the diode switching status based on the current in the last simulation time step, since the diode changes actually its conduction status between two discrete simulation time steps.

As a consequence, as observed in Fig. 8.8 the current i_{L_s} has oscillation when i_{s1} is around zero. During the chattering, the numerical results of i_{L_s} are always near

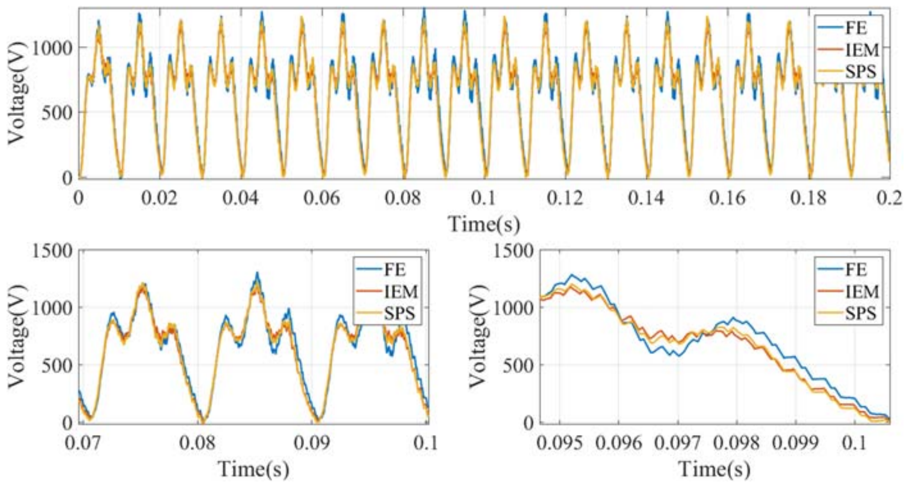


Figure 8.9 The simulation results of voltage U_{C_d}.

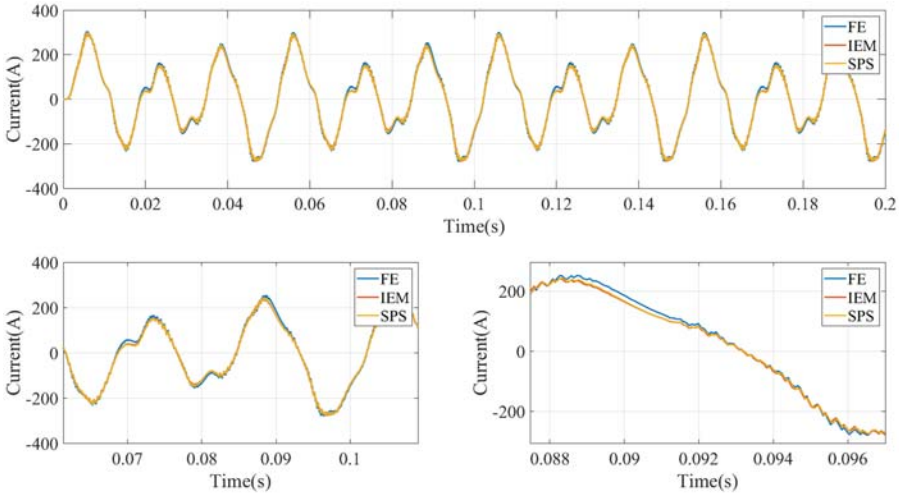


Figure 8.10 The simulation results of current i_{L_a}.

but never equal to zero. The reason causing chattering can be explained by the calculation of i_{s1} using (8.13).

$$L_s \frac{di_{L_s}}{dt} \approx \begin{cases} f_1\left(i_{L_s}\right) = U_s - U_{C_d} - R_s \times i_{L_s} & if \;\; i_{L_s} > 0 \\ f_2\left(i_{L_s}\right) = 0 & if \;\; i_{L_s} = 0 \\ f_3\left(i_{L_s}\right) = U_s + U_{C_d} - R_s \times i_{L_s} & if \;\; i_{L_s} < 0 \end{cases} \tag{8.13}$$

Due to the change in the topology, the calculation of i_{L_s} is made up of three modes: two conduction modes ($i_{L_s} > 0$ or $i_{L_s} < 0$) and one block mode ($i_{L_s} = 0$). At time step t_{n+1}, if no iteration process is used, the mode selection is decided by the value of i_{s1} (t_n) from the last time step t_n.

Assume that the diode passes from the conduction mode f_1 to the blocked mode f_2 in the interval of (t_n, t_{n+1}). If the diode status is identified by $i_{L_s}(t_n) > 0$, the value of i_{L_s} (t_{n+1}) is calculated based on the conduction mode. Under a fixed time step solver, it is impossible to get the exact time when i_{L_s} is reduced to "0." As a result, i_{L_s} (t_{n+1}) keeps decreasing and results in a negative value rather than 0. In the next step (t_{n+1}, t_{n+2}) the negative value of $i_{L_s}(t_{n+1}) < 0$ will cause the diode to enter conduction mode f_3, which will further bring a sudden rise of i_{L_s} (t_{n+2}). This is repeated for each chattering shown in Fig. 8.8, where the current is supposed to be zero in real case.

The major factor causing the chattering is the fixed-step solver used in real-time simulations when encountering the blocking state, and the effect will be expanded if a larger simulation time step is adopted. In the offline simulation, the variable-step solver can effectively reduce the chattering by reducing the time step until the state variable lies within the band defined by the error tolerence threshold. In real-time simulation, it is an undesirable phenomenon and it is usually characterized by a number of zero-crossing events occurring in a finite time interval. Since the chattering relates to the change of the circuit status, it could occur for both explicit and implicit solvers.

Iteration is a good way to relieve this problem by reducing the chattering magnitude, but for real-time simulation the time cost is too heavy. We can introduce an applicable method to ease the chattering phenomena. As shown in Fig. 8.11, we can simply force the switches $[S_a^+, S_a^-, S_b^+, S_b^-]$ to enter the blocking status once the zero-crossing event of the current i_L is detected. Assuming that the zero crossing occurs between t_n and t_{n+1}, $i_{L_s}(t_{n+1}) < 0$ indicates that the zero crossing happens, the value of i_{L_s} (t_{n+1}) can then be forced to zero as in blocking mode. When the absolute value of the input voltage $|U_s|$ is larger or smaller than U_{C_d} the blocking mode ends.

By applying this method, we rebuild the model, and the simulation results are shown in Figs. 8.12–8.14.

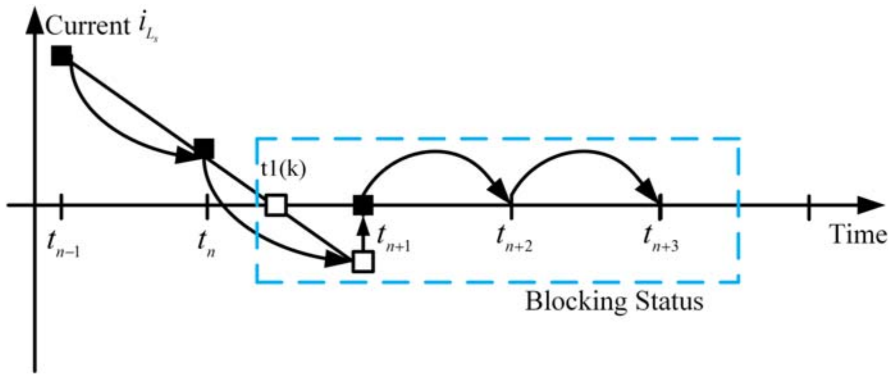


Figure 8.11 The high impedance status introduced to the modeling.

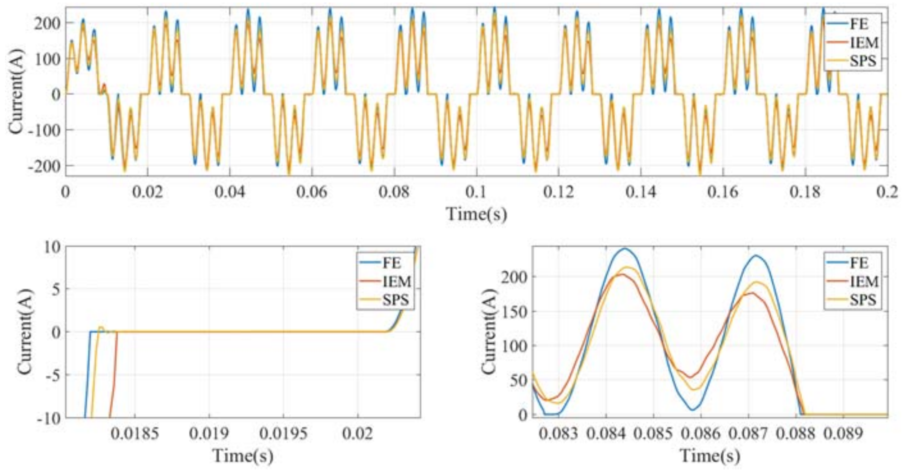


Figure 8.12 The simulation results of current i_{L_s} after adding the blocking mode.

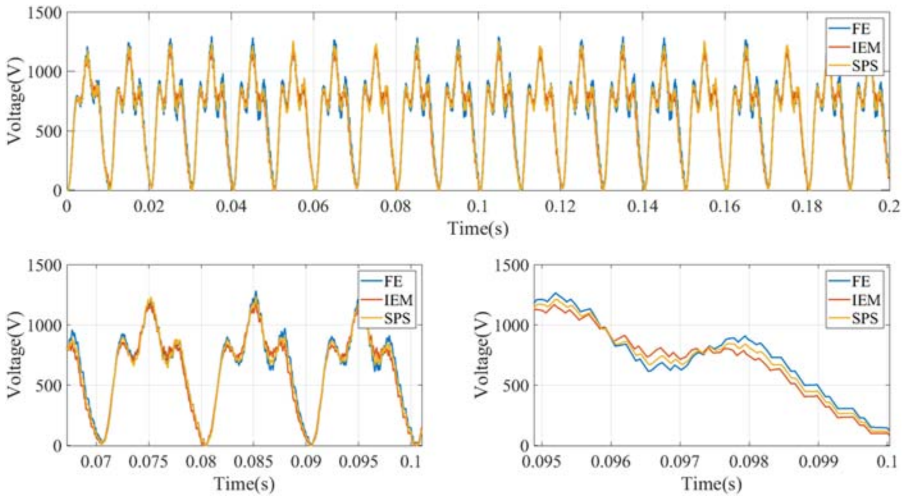


Figure 8.13 The simulation results of voltage U_{C_d} after adding the blocking mode.

Comparing Fig. 8.12 with Fig. 8.8, we can see that the chattering is significantly reduced after adding the forced blocking mode. Comparing Figs. 8.13 and 8.14 with Figs. 8.9 and 8.10 the accuracy of U_{C_d} and i_{L_a} is also improved, and the results of IEM are closer to the SPS reference than FE method.

8.4.2.4 Code compilation

The code implementation of the model is through the Embedded Coder Support Package for Texas Instrument C2000 in Matlab/Simulink Environment. This package

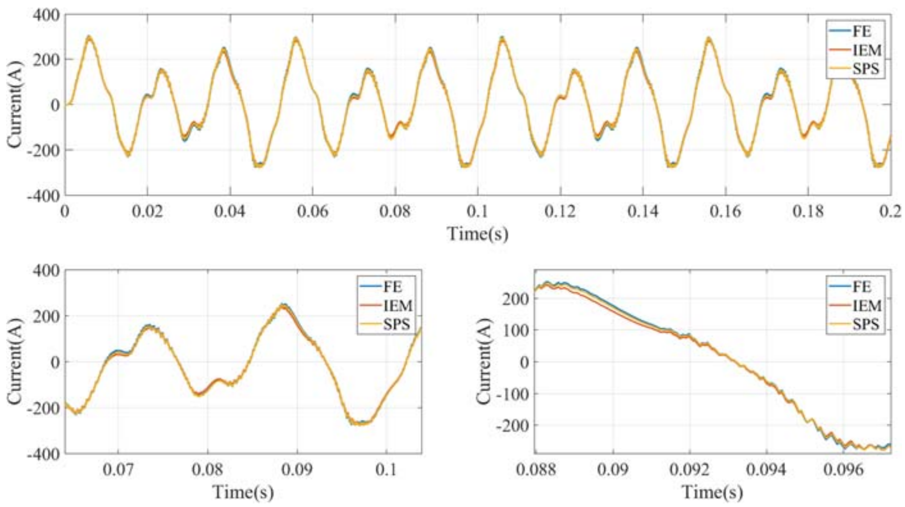


Figure 8.14 The simulation results of current i_{L_a} after adding the blocking mode.

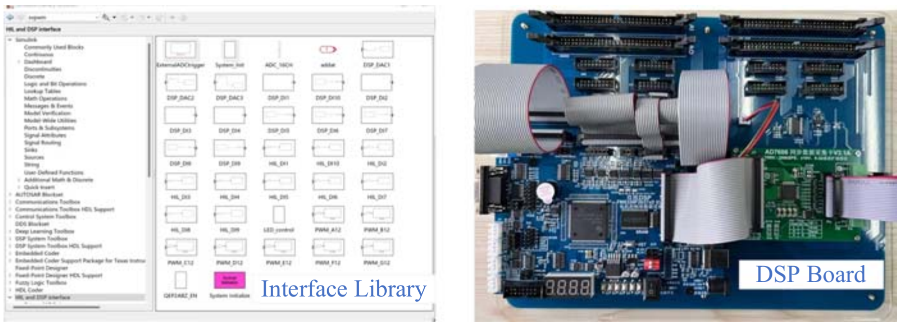


Figure 8.15 Interface library and DSP 28377D.

enables generating a real-time executable code and downloading it to the TI development board (shown in Fig. 8.15). The Embedded Coder automatically generates C code from the model and inserts the I/O device drivers in the block diagram.

In Simulink, we can also use real-time execution profiling to check whether the generated code meets real-time performance requirements. Execution profiling results can also be used to take actions to enhance the design of our system. Sample time specified in a Simulink model determines the schedule for running the generated code on target hardware. If the target hardware has sufficient computing power, the code runs according to specified sample time in real time.

After the compilation, the time step of 20 μs can be met for the developed model. Therefore in the real-time simulation, we choose 20 μs as the simulation time step. In the next section, we will further evaluate the performance of the compiled code.

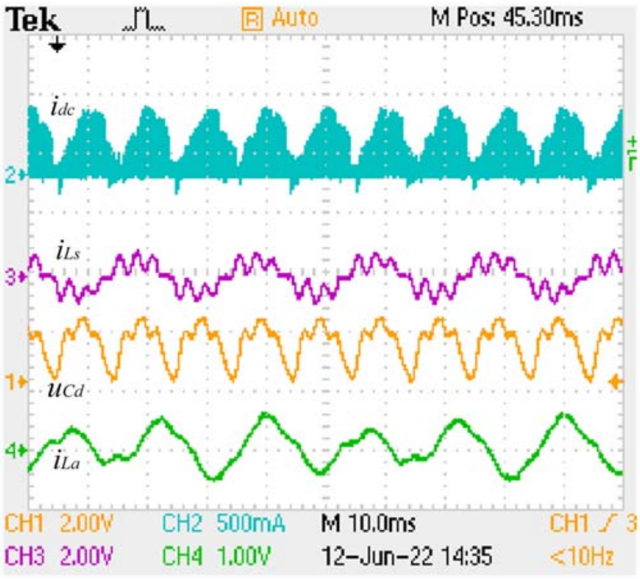


Figure 8.16 Real-time simulation results on digital signal processor.

8.4.2.5 Validate the real-time simulation in the target hardware

The code is downloaded and running in the DSP with a 20 μ*s* time step. The real-time simulation results can thus be produced by using the DAC ports of DSP. Fig. 8.16 shows real-time simulation results captured in the oscilloscope, where the waveforms of I_{dc}, U_{C_d}, and i_{L_s} are displayed. We can see that the results are consistent with the offline simulation results.

It should be noted that this case study is an open-loop test without I/O connected to controller hardware. The purpose is to simply demonstrate the model implementation steps in the real-time processor. The case studies related to close-loop controller HIL will be further introduced in Chapter 10.

References

[1] M.D. Omar Faruque, et al., Real-time simulation technologies for power systems design, testing, and analysis, IEEE Power and Energy Technology Systems Journal 2 (2) (2015) 63–73. Available from: https://doi.org/10.1109/JPETS.2015.2427370.

[2] RTDS Technologies Inc., “PB5 Card”, 2022. [Online]. Available: https://www.rtds.com/technology/simulation-hardware/

[3] OPAL-RT Technologies, Inc., “Flagship Real-Time Digital Simulator”, 2022. [Online]. Available: https://www.opal-rt.com/simulator-platform-op5707/.

Field programmable gate array-based real-time simulation of power electronic systems

9

9.1 Introduction

Since the power electronic converters are characterized by a high switching frequency, the simulation time step of a real-time simulator should be as low as possible to improve accuracy. The time step of the processor-based real-time simulator is at the microsecond level; a simulation time step below 1 microsecond is in general beyond the capacity of the processor due to the inherent latencies and lack of determinism.

Field programmable gate array (FPGA)-based real-time simulators can achieve significantly lower time steps compared to processor-based systems. First, the FPGA-based approach allows tight integration between the I/O and the model computation, resulting in significantly reduced communication latency. Second, FPGA configurable logic blocks and memory resources can be fully tailored to the given task, resulting in an extremely high computational efficiency. Third, the FPGA structure is parallel in its nature, so different processing operations do not have to compete for the same computational resources. Each independent processing task is assigned to a dedicated section of the chip and can function autonomously without any influence from the other logical blocks. As a result the performance of one part of the application is not affected when the designer adds more processing. Therefore, all the mentioned FPGA advantages result in a significant reduction of the overall execution time of implemented algorithms compared to the processor-based simulation approach.

From the literature, the time step of the FPGA-based real-time simulation ranges from tens of nanoseconds to several microseconds based on different customized designs. With the small simulation time step, the fast-transient processes at both the system level and device level can be simulated. In the system-level real-time simulation, FPGA enables the simulation of the high-switching-frequency converters, such as the soft-switching circuit and the resonant converter. In the device-level simulation the nanosecond-level time step permits the description of the detailed switching transients. Nevertheless, FPGA also has weaknesses in implementing real-time simulation, such as limited hardware resources, suboptimal floating-point arithmetic support, and lower achievable clock frequencies.

This chapter focuses on the discussion of FPGA-based real-time simulation of power electronics. Since modeling methodologies are already discussed in the previous chapters, the architecture of FPGA-based simulation at both the system level and device level is directly introduced. Then the issues about FPGA hardware design, implementation, and optimization for the real-time simulation are presented. Additionally the generic programmable FPGA solver approach is presented. An AC-DC-AC power electronic system is chosen as the case study of the FPGA-based real-time simulation for both the system-level and device-level simulations. At last the choices between the processor-based and FPGA-based real-time simulations are discussed.

9.2 The architecture of FPGA-based real-time simulation

In Chapters 5 and 6, we saw that the simulation methodology of power electronics mainly comprises three parts: the modeling of individual components, the network modeling for describing the connection, and the numerical method for the numerical solutions.

We consider a network that is modeled using the nodal analysis method for the illustration. In the time-domain simulation, components are first discretized using the numerical methods such as the Euler method and trapezoidal (TR) method and represented by the companion circuit model. Then the network will be represented by a resistive network containing the conductance and the power sources and formulated by the nodal voltage equations given by Eq. (9.1).

$$\mathbf{Ax} = \mathbf{b} \tag{9.1}$$

where $\mathbf{x} = \left[\mathbf{v_1} \ldots \mathbf{v_N}\right]^{\mathbf{T}}$ is the vector of nodal potentials, $\mathbf{A}$ is the nodal admittance matrix, and $\mathbf{b} = \left[\mathbf{i_1} \ldots \mathbf{i_N}\right]^{\mathbf{T}}$ is the vector of the current source. The detailed information of Eq. (9.1) can be seen in Chapter 5.

After getting the nodal voltages by solving Eq. (9.1), the branch voltages can be obtained based on the topology, and the branch currents can be solved by the voltage—current relationship. Therefore the output variables can be merged into vector y and solved using the output equation shown in Eq. (9.2).

$$\mathbf{y} = \mathbf{Wx} \tag{9.2}$$

9.2.1 The solving structure of the system-level simulation model

In the system-level simulation, the switch models are usually represented by the static model presented in Chapter 3 (like the binary resistor model and the ADC model). $\mathbf{Ax} = \mathbf{b}$ is thus a system of linear equations once the switch status is determined (neglecting the nonlinearities in the passive components). Thus the nodal voltage can be solved by Eq. (9.3).

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{9.3}$$

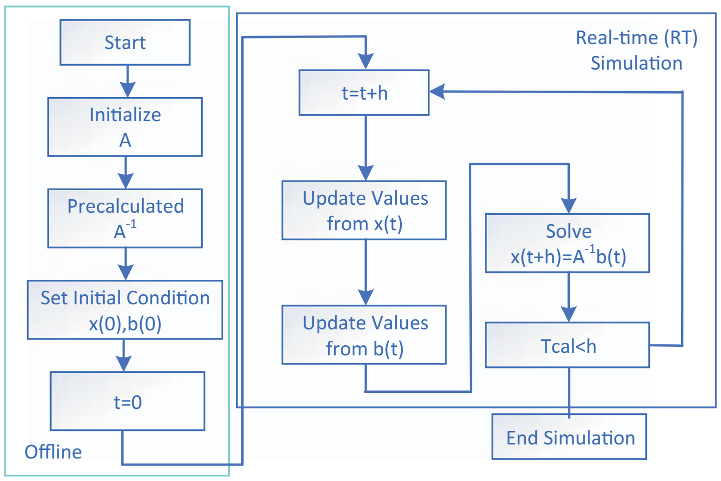


Figure 9.1 Working flow for FPGA-based real-time simulation.

For the model implementation of Eq. (9.3) on FPGA, the characteristics of FPGA should be fully considered to meet the time constraint in real-time simulation. Traditionally, as shown in Fig. 9.1, both the offline process and the online process of the model implementation on FPGA are involved, where the precomputing method presented in Chapter 6 is used.

1. *Offline process: the computation before real-time implementation*

To compute $\mathbf{x}$ using Eq. (9.3), we need to know the value of the inversed matrix $\mathbf{A}^{-1}$. However, matrix inversion is a very time-consuming process no matter whether the direct method or the iterative method is used as indicated in Chapter 6. Although parallel computing processes can be realized in solving $\mathbf{A}^{-1}$, the resulting hardware resources and computing latency can also be costly to FPGA. For the FPGA-based real-time simulation the matrix inversion to get $\mathbf{A}^{-1}$ should be avoided so that the time step can be minimized and hardware resources can be saved. Thus the precomputing method is commonly used to store the value of inversed matrix $\mathbf{A}^{-1}$ for all the possible combinations of switch status.

For the power electronic system the number of topology variants is determined by the number of switches. For example, if the system has 10 switches, the possible combinations of switch status are theoretically 1024 (2^{10}). The corresponding matrix A^{-1} could reach 1024. This is a heavy burden for the storage unit. Although the precomputing method is easy to use and can authentically reflect the topology change due to the switches, huge memory resources may be needed to store the different variants of A^{-1}. These are the advantages and disadvantages of the ideal switch model or the binary resistor switch model.

However, things are different for the ADC method. If the ADC switch model is used, the topology will be invariant when the switch status changes. The ADC model is composed of conductance and a current source. The value of the current source is updated at each step based on the historical value of the switch current/voltage and the switch status, while the conductance is of constant value regardless of the switch status. Therefore the

ADC method reflects the change of the topology only through updating **b**, and the value of $\mathbf{A}^{-1}$ remains constant. Therefore only one possibility of $\mathbf{A}^{-1}$ needs to be stored. However, the ADC method introduces fictive inductance and capacitance to the power electronic system, which may cause numerical oscillation. Additionally, this method is known to introduce virtual power losses in the simulated circuit.

2. *Online process: FPGA structure for real-time implementation*

For both the invariant topology when using the ADC switch model and the variant topology when using the ideal switch model or binary resistor model, the value of $\mathbf{A}^{-1}$ should be stored so that the online matrix inversion can be avoided, which is the key task in the offline process. The online process involves the hardware design to implement the simulation of the power electronic models. For each iteration, vector **b** is updated and then the vector **x** is computed. Fig. 9.2 presents the architecture of a custom design, which consists of five fundamental modules [1]:

Updater module: The updater is composed of two logic blocks that analyze gate signals and variables from the previous simulation step called history terms. In this module, the switch status is determined based on the gate signals and the history terms. If the switch is of the fully controlled type, the status is directly determined by the gate signals. However, if the switch is of the uncontrolled type, such as the diode, the status in the natural commutation process must be determined by the sign of the current or voltage at the switch terminals. Moreover the results of the last simulation step are treated as the initial point of this simulation time step.

Keeper module: The keeper contains the vector registers representing **b** and the matrix circuit representing $\mathbf{A}^{-1}$. The keeper receives the results of the switch status and the system input variables. Based on the status of the switch the corresponding inverse matrix $\mathbf{A}^{-1}$ and vector **b** are updated. In the invariant topology modeling approach, $\mathbf{A}^{-1}$ remains unchanged during switching operations. However, for the variant modeling approach, both the value of $\mathbf{A}^{-1}$ and **b** need to be updated at each simulation step.

Multiplier module: The matrix-vector multiplier is formed by several dot product (DP) units working in parallel. In this stage, each element of **x** in Eq. (9.3) is computed as the sum of dot products of elements in the corresponding row of matrix $\mathbf{A}^{-1}$ and vector **b**. One possible implementation of a DP unit is shown in Fig. 9.3.

Output selector: The output selector routes matrix–vector multiplication results. The variables are fed back to the updater module as the history terms for the next computing cycle, while the variables of interest are computed and sent to the output.

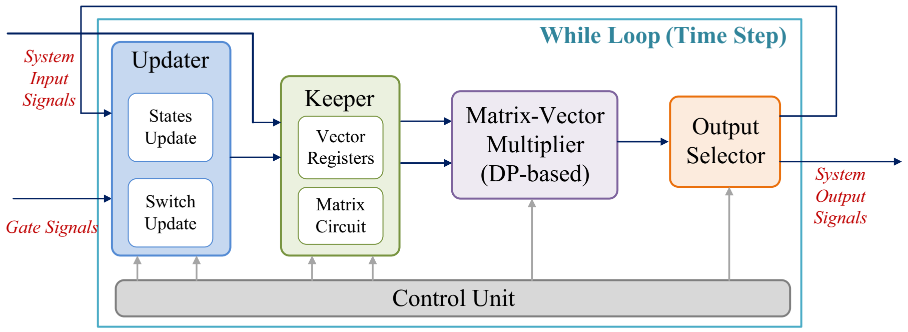


Figure 9.2 Custom FPGA design for power electronic system simulation [1].

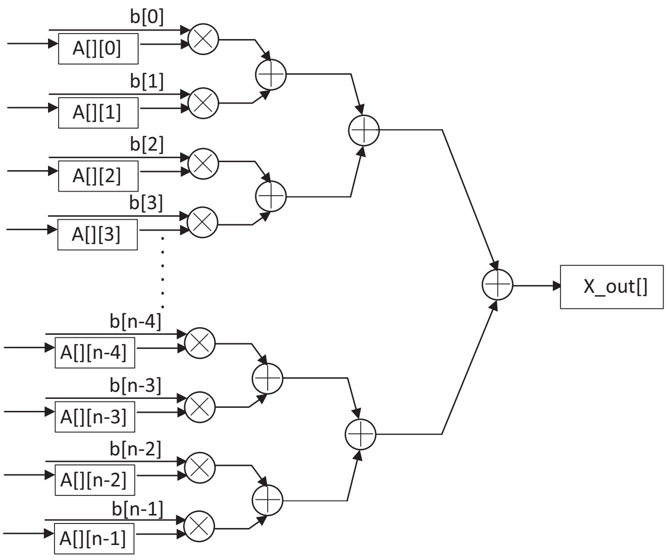


Figure 9.3 Dot product operator structure.

Control module: The control unit schedules all operations for vector updating, memory data reading, DP operation, data feedback, and output selection. It is a state machine formed by a counter and memory blocks.

To summarize, in the system-level simulation, the model of the power electronic system is represented by the dot product of Eqs. (9.3) and (9.2) and executed step by step to get the real-time solution in the discrete time.

9.2.2 The solving structure of the device-level simulation model

In the device-level real-time simulation, the main task is to integrate the switch's transient models into the real-time simulation. There are two ways to combine the switch transient model with the system-level models. The first one is to model the switch using the equivalent circuit and integrate it as a part of the network just as many offline simulation tools do. In this case the network scale is significantly enlarged since additional branches and nodes are added compared to the system-level simulation. Moreover, nonlinearities are introduced and an iterative solving process such N-R algorithm may be inevitable, which makes the system-level simulation solving structure presented above unsuitable. For example, if the IGBT nonlinear equivalent circuit model presented in Chapter 3 is used, both $\mathbf{A}$ and $\mathbf{b}$ will vary at each iteration, and $\mathbf{A}^{-1}$ has to be computed at each iteration in a simulation time step. The computing time will be too large to be accepted in the device-level real-time simulation where a smaller time step is usually desired due to model dynamics. To avoid the significant increase in computing time, in particular applications of device-level real-time simulation, the network partitioning method or decoupling method is applied to separate the subcircuits containing detailed device-level models of switches. Those subcircuits provide the condition to apply other kinds of the simplified transient switch model such as the piecewise linear transient model and the curve-fitting model in the simulation. The switching unit, such as an IGBT/diode switching pair, or a half-bridge (HB) is usually selected as the subcircuit to decouple. In this case the FPGA hardware design can be summarized into the following structure (Fig. 9.4).

It can be seen that the switch model is solved together with the network in each time step. In fact the network may not need such a shorter simulation time step as the switching transient does. Therefore there is another way to integrate switching transient model in the real-time simulation, which is using the two-level simulation framework, as presented in section 3.3.4 of Chapter 3. The network is modeled using the static switch model and implemented in the system-level simulation with a large time step, while the switching transient is generated by the transient model with a much smaller time step to connect with the steady-state turn-on and turn-off values. Therefore the transient switch model does not participate in the network solution, which greatly reduces the computing burden of the network. Moreover the transient switch model is solved without considering the surrounding circuit, which also reduces the computing burden of the device-level simulation. Let us take the high-resolution quasitransient (HRQT) model as an example to show its FPGA hardware structure.

The basic structure of its FPGA implementation is shown in Fig. 9.5, providing the top-level hardware diagrams. The top-level diagram consists of four basic modules, including the network solver, the timing sequence control unit, the transient parameter computation (TPC) module, and the transient waveform generation (TWG) module.

The network solver uses IGBT/diode static models and computes the branches' currents and voltages in the network model in each time step. After obtaining the network results, the sequence control unit selects the states of each HB subcircuit, including IGBT voltage and current (v_{ce} and i_c), gate signals, and the input current and voltage, and assembles them into groups according to the HB index. Then these values are sent to the TPC module group by group.

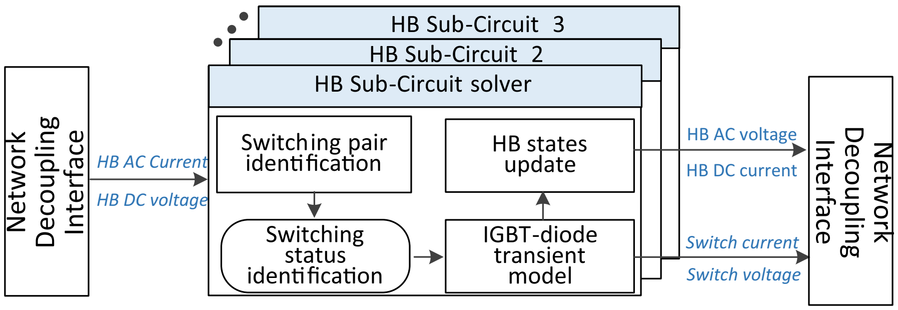


Figure 9.4 FPGA hardware structure of the device-level real-time simulation.

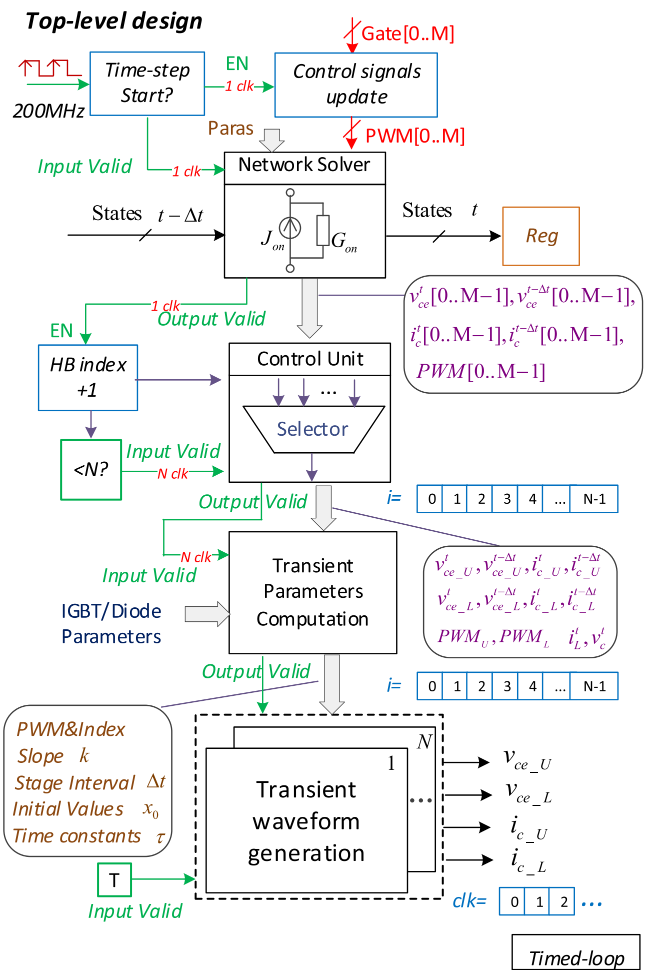


Figure 9.5 The diagram of the top-level hardware design.

It needs to be mentioned that the HRQT model is solved in the context of the half-bridge topology, in which only one pair of IGBT and diode is switching at one time. In Fig. 9.5 the number of IGBTs in the topology under test is denoted as M, and the number of HBs is denoted as N. The values in the upper side switches are denoted by subscript U, and the values in the lower side switches are denoted by subscript L.

The transient parameters of the switch model are then computed accordingly, including the stage intervals, slopes, initial values, and time constants of the transient waveforms modeled in section 3.3.4. The IGBT transient waveforms can thus be generated based on the computed parameters. In addition the power losses p_L are computed at the same time based on the generated waveforms.

The top-level computation sequence of this approach is depicted in Fig. 9.6. The 200 MHz clock is employed in the entire hardware. Each module is designed into

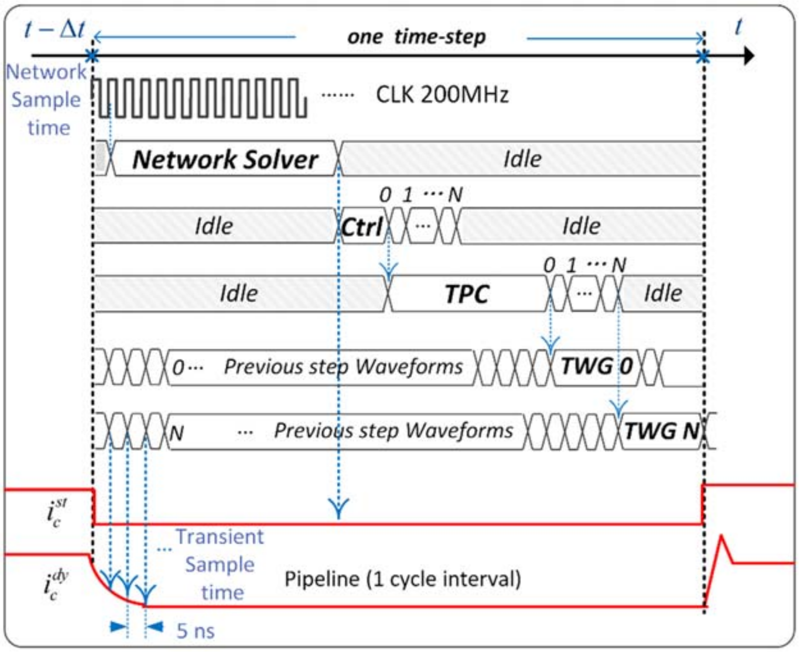


Figure 9.6 Computing sequence of the top-level FPGA implementation.

an IP core. The generated IP cores are placed into the timed-loop structure, and the sequences are guaranteed by the handshake signals, which are denoted as the "input valid" and "output valid" in Fig. 9.5.

By using the high-level synthesis (HLS) tool, the control unit, the TPC module, and the IGBT waveform generation module are designed into highly pipelined structures with one clock cycle interval. There are N HB subcircuits in all, and the control unit sends the N groups of HB static values in N successive clock cycles. Then the TPC module is enabled for N producing N groups of transient parameters after the specific computation time. The transient parameters of different HBs can thus be processed by a single TPC module. This pipelined structure is favorable since the multiple switching events can occur in one time step, and the duplicate TPC modules increase the occupation of FPGA hardware resources (the explanation of pipeline structure can be referred to in Section 9.3.2).

The IGBT waveform generation module is also highly pipelined with 1 cycle interval. Different from the TPC module, the motivation for using pipelined structure here is to generate high-resolution transient waveforms rather than economize the device resources. Therefore transient waveforms can be produced in each clock cycle, and a 5 ns resolution can be realized with a 200 MHz clock rate. In Fig. 9.6 the IGBT collector's current i_c is used as an example to illustrate the computing sequence. i_c^{st} is the network simulation result, and i_c^{dy} is the generated transient waveform.

9.3 FPGA hardware design

FPGAs are predominantly programmed using HDLs (hardware description languages) such as Verilog and VHDL. VHDL and Verilog are well suited for describing a design at the gate level and the register transfer level (RTL) of abstraction. However, using a higher level of abstraction than RTL can increase designer's productivity and reduces the likelihood of bugs, resulting in a faster time for development.

9.3.1 High-level design tools for FPGA hardware

1. *C-based design with high-level synthesis tools*

High-level synthesis (HLS) is an increasingly popular approach in electronic design automation (EDA) that raises the abstraction level for designing digital circuits [2]. Fig. 9.7 shows the standard implementation procedure from C code to VHDL code for real-time implementation on FPGAs. This approach enables users to further explore the design space, optimize hardware–software co-designs, and easily compare different final hardware implementations of the same algorithm in terms of performance, digital resources, and estimated power.

For example, using Vivado HLS, we can apply different optimization directives to the design, including the following:

a. Instruct a task to execute in a pipeline, allowing the next execution of the task to begin before the current execution is complete.

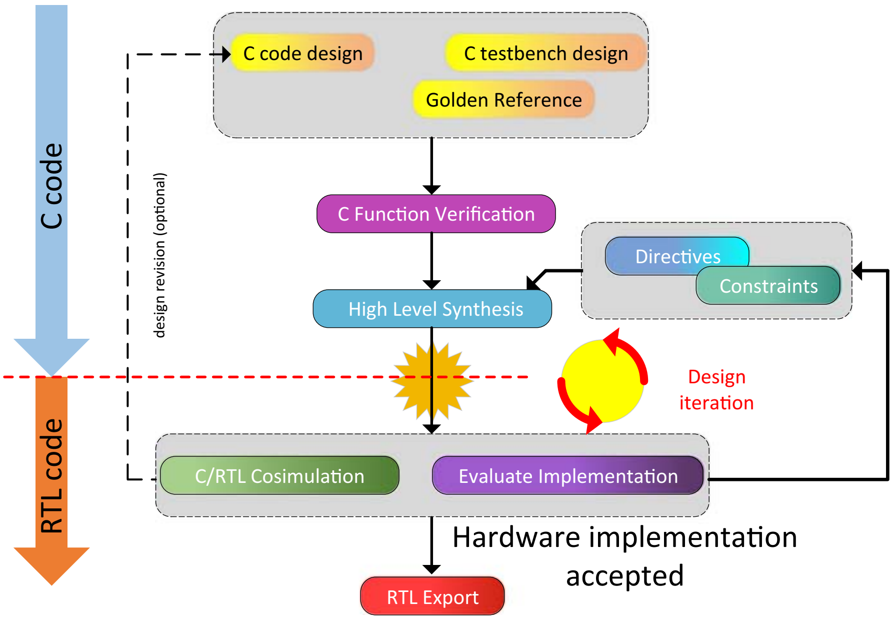


Figure 9.7 FPGA implementation procedure using high-level synthesis tools.

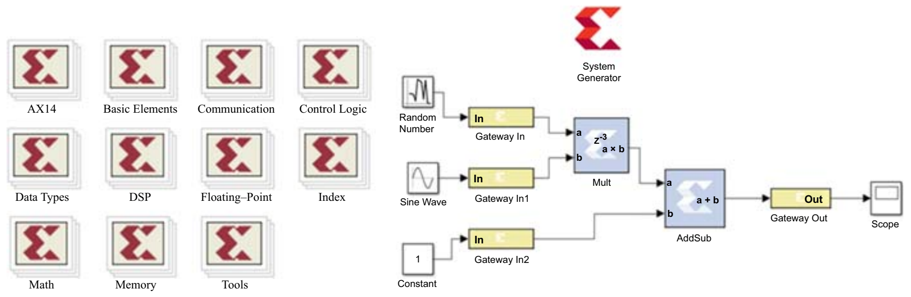


Figure 9.8 System generator in MATLAB®/Simulink.

b. Specify a latency for the completion of functions, loops, and regions.
c. Specify a limit on the number of resources used.
d. Override the inherent or implied dependencies in the code and permit specified operations. For example, if it is acceptable to discard or ignore the initial data values, allow a memory read before writing if it results in better performance.
e. Select the I/O protocol to ensure that the final design can be connected to other hardware blocks with the same I/O protocol.

2. *Model-based design with system generator*

   System generator is an FPGA design tool from Xilinx that enables the use of the MathWorks' Simulink design environment to design FPGA algorithms. Experience with Xilinx FPGAs or RTL design methodologies is not obligatory when using System Generator [3]. Algorithms are designed in the Simulink modeling environment using a Xilinx-specific block set, as shown in Fig. 9.8. The System Generator design can then be imported into a Vivado IDE project using the IP Catalog.

   The tool provides high-level abstractions that are automatically compiled into an FPGA. The tool also provides access to underlying FPGA resources through low-level abstractions, allowing the construction of highly efficient FPGA designs. Over 90 DSP building blocks are provided in the Xilinx DSP block set for Simulink. These blocks include the common DSP building blocks such as adders, multipliers, and registers. These blocks leverage the Xilinx IP core generators to deliver optimized results for the selected device [4].

9.3.2 Factors affecting the FPGA execution

Programming an FPGA is a process to customize its resources to implement a definite logical function. This involves the FPGA's basic building blocks like configurable logic blocks (CLBs), dedicated multiplexers, and others to meet the requirements of the digital design. On the FPGA the major delay is due to the computation of the mathematical model and the time required for the I/O signals to propagate to model inputs and model outputs. Nevertheless, the algorithm can be executed very fast because it lies in the hardware fabric of the FPGA instead of a software-controlled execution environment [5]. It has the flexibility of software since there are no physical changes to the device to change its behavior and has the speed of hardware, executing at clock rates up to the hundreds of megahertz range.

To achieve the full potential of the FPGA platform, we should get familiar with the following concepts: *latency, throughput, FPGA resource, and data representation.*

1. *Latency*

   Latency is the time it takes for an operation to complete [5]. It is measured in units of time, usually milliseconds, microseconds, and nanoseconds. In many applications, such as control applications, the round-trip latency, that is, the time from input to corresponding output, is very important in the design process.

2. *Throughput*

   Throughput is the rate at which data are processed by the system. It is the number of operations per given time period. Throughput is usually expressed in operations per second, such as MIPS (million instructions per second), MACS (multiply-accumulate operations per second), or FLOPS (floating-point operations per second) [5].

   Pipelining is a technique that enables the parallel execution of program instructions. Fig. 9.9 shows an example of a high-throughput structure after being optimized by HLS tools. Suppose a system contains four processes: A, B, C, and D. T_s is the total computing time and the period of the time loop clock C1.

   Without a pipeline structure the whole system reads the input signal $u(t)$ and outputs only one numerical result $x_{(t+Ts)}$ after a time step of T_s, in this case, by adding a pipeline structure using clock C2, which is four times faster than clock C1. Thus the execution rate per iteration will become four times. After a time step of T_s the whole system will output four numerical results $x_{(t+Ts)}$, $x_{(t+Ts+h)}$, $x_{(t+Ts+2h)}$, and $x_{(t+Ts+3h)}$ in four successive cycles of clock C2. Thus the output is updated with a simulation step h rather than Ts. The advantage of this pipeline process is that the throughput of the FPGA algorithm can be significantly improved. However, this causes more FPGA resource occupation. Also, it should be noted that pipelining in theory does not affect system latency, but in practice, it typically results in slightly increased latency due to suboptimal splitting of the pipeline stages.

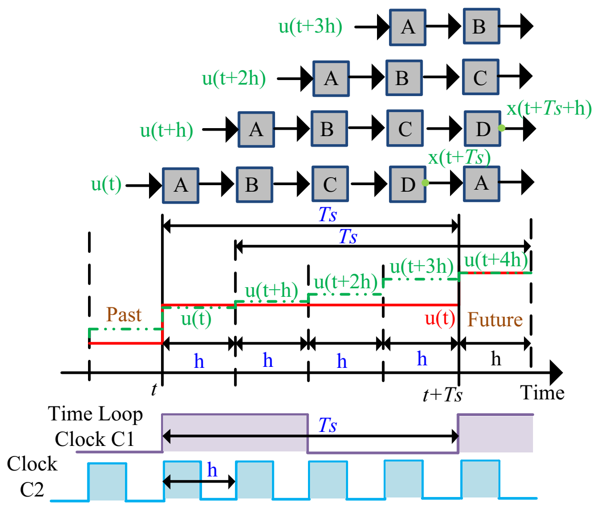


Figure 9.9 Performance improvement with pipelining.

3. *FPGA resource*

   Every FPGA chip is made up of a finite number of predefined resources with programmable interconnects to implement a reconfigurable digital circuit and I/O blocks [6]. FPGA resource specifications include the number of configurable logic blocks (CLBs) (made up of flip-flops and lookup tables), the number of fixed-function logic blocks such as multipliers, and the size of memory resources such as embedded block RAM. They are the most important parameters when selecting and comparing FPGAs for an application. Let us see a brief introduction of these parameters; for readers who are interested in more comprehensive explanations, please refer to reference [6].

   Flip-flops: There are three most important pins on the flip-flop: data input, data output, and clock input. Between clock cycles within an FPGA circuit, flip-flops save logical states from the input data and hold that value constant until the next clock edge.

   Lookup tables (LUTs): LUTs are the method of FPGA executing combinatorial logic. By predefining a truth table in the LUT memory the computation of all combinatorial logic (ANDs, ORs, NANDs, XORs, etc.) is implemented.

   Multipliers and DSP slices: To process the numerical multiplication operations, multiplier circuitry is prebuilt in FPGA and saves on LUT and flip-flop. Moreover the multiplier-accumulate circuitry has also been prebuilt in FPGA. The DSP48 slices are a multiplier with adder circuitry to process 25-bit multiplying 18-bit [6].

   Block RAM: Block RAMs are used for storing large amounts of data inside FPGA. Block RAMs have a customizable width and depth. One of the main applications is the first-in-first-out (FIFO) memory buffers. With FIFO, we can buffer data among different hardware logics running at different rates [6].

4. *Numerical* representation

   The numerical representation is an important point to consider for any FPGA application. In comparison to the fixed-point data type, the floating-point provides higher numerical precision. However, due to its complexity and the lack of native support by FPGA DSP blocks, it typically results in longer computation times and increased resource usage compared to the fixed-point data type.

   For that reason, fixed-point representation is commonly used in FPGA-based real-time simulations. The fixed-point representation simplifies implementation and provides size and speed advantages for math operations. The critical challenge in using the fixed-point representation is to derive the optimal values for the word length, number range, and resolution in one application. Considering that our application lies in the power electronic system, the optimization goal is to define a generic fixed-point representation method suitable for different power ranges.

   In some applications, it proves useful to combine fixed-point with floating-point operators. That is, if the inputs are already fixed-point numbers, it becomes natural to perform basic operations in this format. However, when more advanced operations appear in the algorithm, it is advisable to switch to a floating-point implementation due to its better dynamic range and accuracy.

9.4 Optimized methods for FPGA-based computation

9.4.1 Improved numerical accuracy by using the per-unit system

In typical electrical circuits, there are four basic electrical quantities: voltage $V(V)$, current $I(A)$, impedance $Z(\Omega)$, and power $S(\mathrm{VA})$.

For single-phase circuits,

$$\begin{aligned} V(\mathrm{V}) &= Z(\Omega)\cdot I(\mathrm{A}) \\ S(\mathrm{VA}) &= V(\mathrm{V})\cdot I(\mathrm{A}) \end{aligned} \tag{9.4}$$

In the per-unit (pu) system, the physical quantity is expressed as a fraction of the reference value.

$$\text{Per unit value} = \text{actual value}/\text{base value} \tag{9.5}$$

Similar apparatus (generators, transformers, lines) will have similar per-unit impedances and losses expressed on their rating, regardless of their absolute size. Thus different power ranges can be represented in a similar way as measured representations and ranges. Because of this, the number of bits used in the integer portion of the fixed-point data could be reduced.

For example,

$$V(\mathrm{pu}) = V(\mathrm{kV})/V\ \text{base (kV)} \tag{9.6}$$

where the base value is a reference value for magnitude.

It is a common practice to specify base power (S_B) and base voltage (V_B). Then other base values can be obtained,

$$\begin{cases} \text{Base current } I_B = S_B/V_B \\ \text{Base impedance } Z_B = V_B^2/S_B \end{cases} \tag{9.7}$$

9.4.2 Reduce computing latency by paralleled network modeling

FPGA support natively paralleled computing resources. Therefore if a power electronic system could be modeled in a parallel manner, parallel computing can be achieved in FPGA to speed up the simulation. As presented in Chapter 6, the paralleled network modeling includes the equivalent network partitioning methods and the delay-based network decoupling methods. No matter which method is adopted, one important procedure is to properly compute the partitioned/decoupled subcircuits in parallel. A large topology can be split into several smaller-scale pieces that are much easier and faster to solve, and the parallel computing structure takes advantage of the FPGA's inherent parallelism.

In addition to the benefits of faster simulation, parallel network modeling can also improve the efficiency in implementing the FPGA-based simulation. It is well known that different power electronic converters may share the same subcircuits, such as the bridge, the filter, and the transformers. These commonly used subcircuits can be built as IP cores and reused for different applications. Moreover, complex power electronic systems are usually composed of replicative simple topologies such as DC-DC converters, full-bridge converters, rectifiers, and so on. They can also be built as IP cores, and the large-scale system can be formed conveniently by merging them using parallel network modeling methods. However, in some cases where FPGA resources are limited, the same IPs can be executed in a pipelined way to save hardware resources.

However, it should be emphasized that, when using parallel network modeling, especially the delay-based network decoupling method, the numerical stability should be carefully considered. The stability is related to the dynamics of each subcircuit and between the subcircuits, being strongly affected by the simulation time step.

9.5 Generic programmable FPGA solver approach

The approach presented so far in the chapter relies on the implementation of a new digital design for each model to be simulated. The main advantage is that the digital design can be fully tailored for the specific model, thus providing the potential for an extremely high efficiency in terms of both performance and FPGA resource utilization. Another advantage of this approach is high flexibility since the user has full control over the model implementation.

However, the direct digital design implementation approach also has its downsides. First, to fully exploit the advantages of this approach, it is required from the user to have a particularly good understanding of the parallel computing concepts and implications of the different numerical approaches to the simulation performance and accuracy. Second, it requires a certain level of familiarity with the FPGA design, verification, and debugging techniques and toolchains, a skillset which is not that common in the power electronic community. Third, every change to the model usually requires reimplementation and reverification of the whole design, which is a lengthy process. Similarly, the digital design developed for a specific model can only be partly reused for another model, resulting in high incremental modeling effort.

For those reasons, commercial FPGA-based power electronic real-time simulation solutions typically use a generic programmable FPGA solver approach to simplify user experience and speed up model development time. With this approach a simulation platform provider creates a fixed solver digital design which is made to be general enough to cover a wide range of power electronic circuits. The solver configuration for a specific model is done by means of memory initialization. This way, there is no need to make modifications to the digital design for new models (or modifications to the existing ones), thus saving a lot of implementation and verification efforts on the user side and fully eliminating the need for FPGA toolchain expertise. Such generic programmable FPGA solvers are also typically accompanied by the matching software toolchain, which translates model description in the form of a circuit diagram into a configuration set for the solver using the principles presented in the previous chapters. This way, end users are guarded from the underlying complexity of such systems and can harness the power of FPGA-based real-time simulation using an interface that closely resembles the typical offline PC-based simulators [7].

However, the generic programmable FPGA solver approach also has its limitations. The generic nature of the solver means that it is not optimized for one given model, which in practice sometimes results in longer simulation steps and regularly in higher resource utilization compared to the direct implementation approach. Another limitation is the lack of flexibility when it comes to different model implementation options. In some cases, it is even possible that the generic solver cannot implement the required model due to its limitations in terms of, for example, model size, total number of switches, or lack of support for the used switch type or switching topology.

In the remainder of the section, we are going to describe the structure of the commercial FPGA-based real-time simulation platform from Typhoon HIL, which is based on the described generic programmable FPGA solver approach.

Typhoon HIL simulators are based on a hybrid CPU-FPGA multiprocessing platform, similar to Opal-RT or RTDS platforms presented in the previous chapter. The main difference is that the Typhoon HIL simulators fully rely on the FPGA for all electric system simulation-related tasks, while CPUs mainly have a supporting role. Another unique feature is that the power electronic simulation is solely based on the FPGA generic solver approach presented earlier in the chapter. As described, the main advantage of using this approach is a significantly simplified user experience. The hardware structure of the Typhoon-HIL platform is illustrated in Fig. 9.10.

The Typhoon HIL FPGA solver comprises a number of functional elements to fulfill modeling requirements of most typical power electronic models, namely,

1. linear passive elements—both constant and time-varying,
2. nonlinear passive elements,
3. converter blocks consisting of ideal switches,
4. contactors based on ideal and nonideal switches,
5. electrical machines.

The core of the solver is an array of standard processing cells (SPCs) [8]. Each SPC is fully programmable and solves a state space representation of a circuit up to a parameterizable level of complexity in terms of the number of switches and the number of energy storage elements. The solver uses the ideal switch model presented in Chapter 3. The models are typically split into one or more submodels running in parallel on different SPCs to address the issue of exponentially growing memory requirements and to shorten the computational time, as described in Chapters 3 and 6. SPC units operate synchronously and in parallel, but unlike multicore CPUs, they share no resources (e.g., memory), which improves time determinism. They are interconnected using dedicated communication lines which allow them to exchange variables with a single simulation time-step delay.

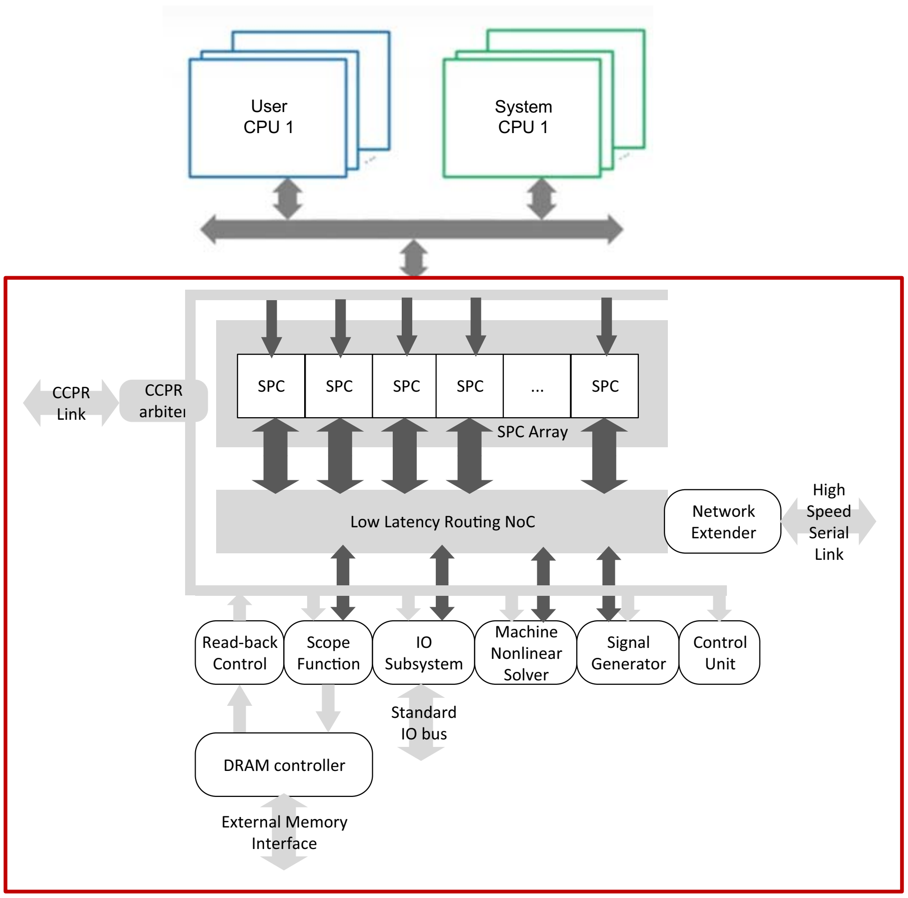


Figure 9.10 Hardware structure of Typhoon-HIL real-time simulator.

The SPC block uses predominantly floating-point number representation to achieve high accuracy and a dynamic range combined with a highly pipelined design to maximize computational speed. SPC blocks also use a parallel internal computation architecture to optimally leverage FPGA inherent parallelism and further increase computational speed. For example,

Fig. 9.11 shows the block diagram of the linear state space solver, which uses both mixed arithmetic and massive parallelism to make the best use of FPGA platform flexibility and available resources.

This presented FPGA solver can achieve simulation time steps down to 200 ns. The overall loop-back latency (digital input to analog output) can be as low as 700 ns thanks to the tight integration between the computation and the I/O modules in FPGA. These performance numbers clearly illustrate that the FPGA-based approach to power electronics real-time simulation brings significant performance benefits even in its generic, nonmodel-optimized form.

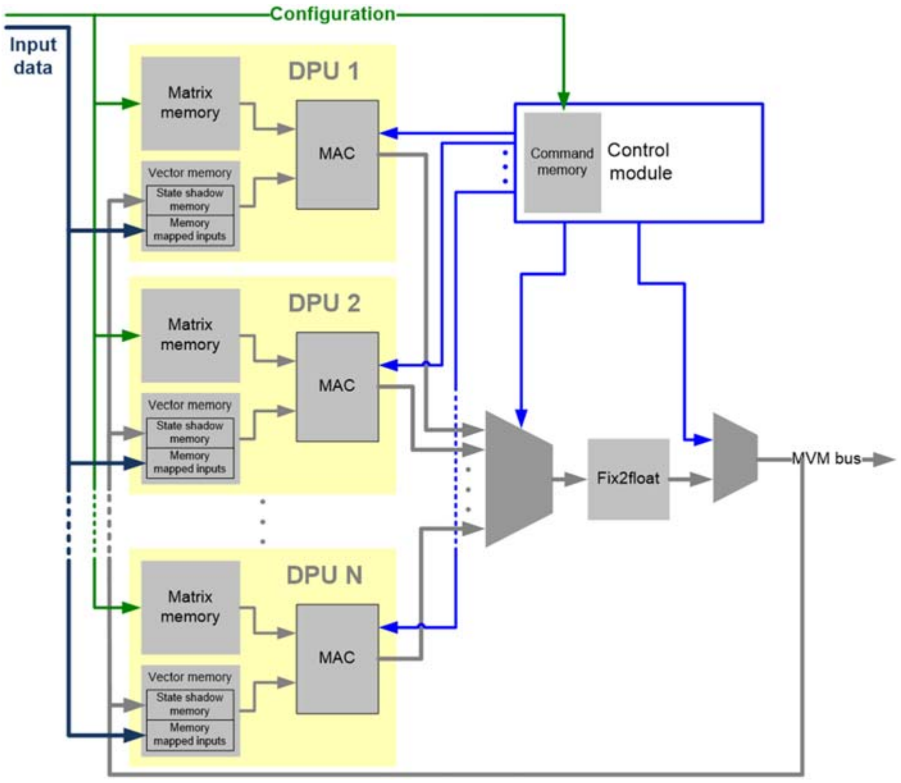


Figure 9.11 Block diagram of the linear state space solver of Typhoon-HIL real-time simulator.

9.6 Case study

In this illustrative example, we continue to use the same AC-DC-AC topology as in Chapter 8. The topology is redrawn in Fig. 9.12. The model will be built using the Xilinx System Generator (SysGen) in the MATLAB Simulink environment. With SysGen, we can use the graphical block programming to build the model and check the FPGA architecture-level design. Besides, we can also validate the same model through simulation in Simulink. After the model verification, the design can be packaged into a Vivado IP and imported into a Vivado project.

9.6.1 FPGA-based system-level simulation

The FPGA implementation structure using the System Generator is shown in Fig. 9.14 with the equations from the same example in Chapter 8. From the simulation results in Chapter 8, we can find that with the time step of 20 μs the FE method could provide acceptable results. Chapter 5 shows that the solving accuracy

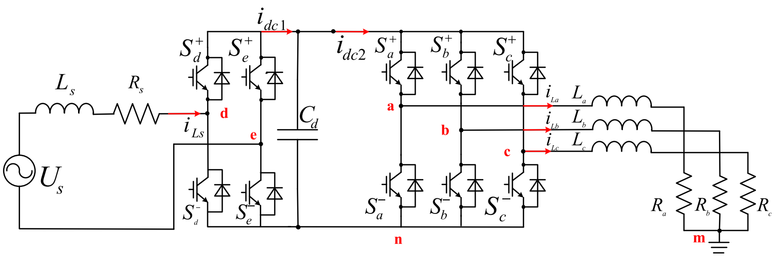


Figure 9.12 AC-DC-AC structure.

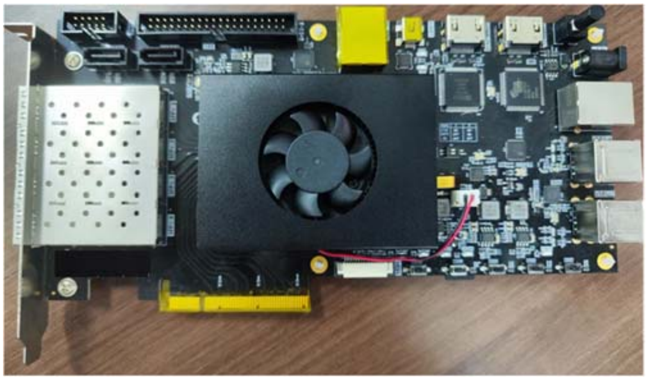


Figure 9.13 Kintex 7 FPGA Board.

increases with the time-step decrease. If the simulation step further reduces less than 1 μs, the error of FE discretization method can be further reduced. Moreover the FE method has the simplest solving structure. Therefore we choose to use the FE method in this case study.

A Xilinx Kintex-7 XC7K325T FPGA is used, and the associated FPGA board is shown in Fig. 9.13. The FPGA clock period is 10 ns (clock frequency at 100 MHz). In the design, the longest route contains 10 latencies, which means that the maximum computing time in the model solving is 100 ns. All the model variables are chosen as signed fixed-point numbers, except that the gate signals are Boolean type. The number format is 40 bits, where 1 bit is used to represent the sign, 4 bits are used to represent the integer part, and 35 bits are used to represent the fractional part.

The simulation results using the FPGA are shown in Figs. 9.15–9.18. It can be seen that the Simulink results are very consistent with the results obtained from the model generated by the Xilinx System Generator. After building and compilation, Fig. 9.19 shows the resource occupation of the FPGA-based simulation, which provides useful design directives to adjust and optimize the models and their designs.

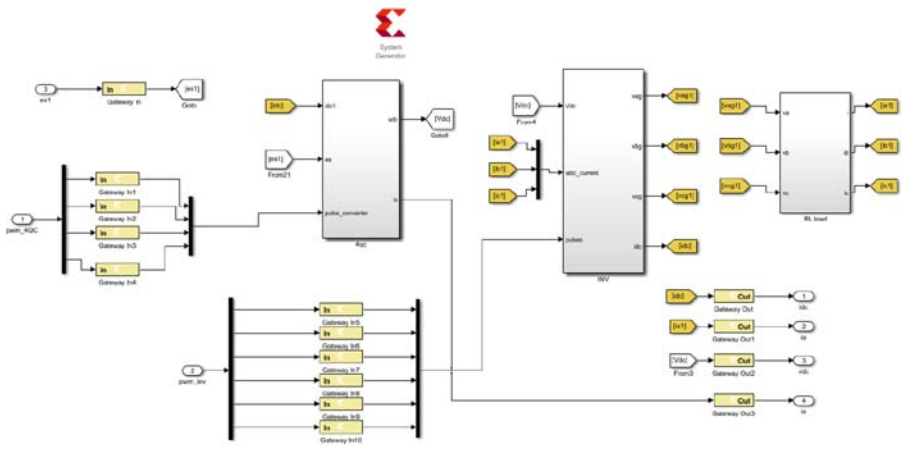


Figure 9.14 Simulink model built using System Generator from Xilinx.

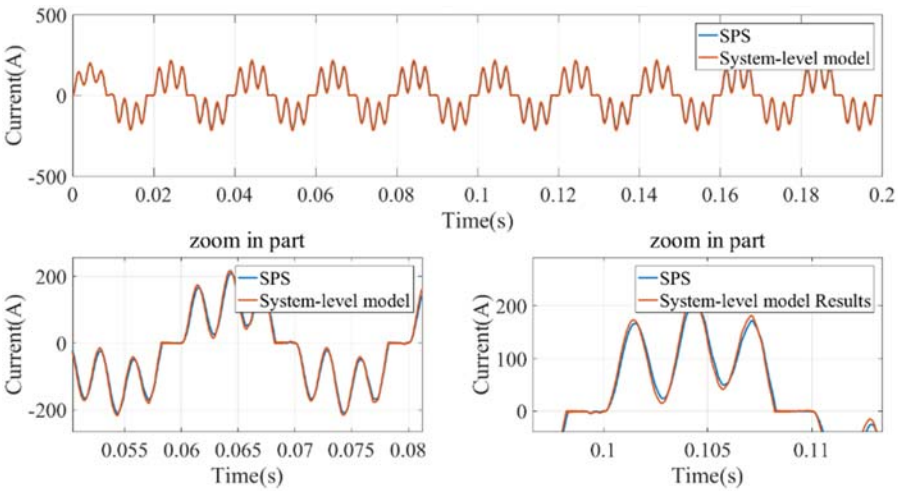


Figure 9.15 The simulation results of the current i_{L_s}.

9.6.2 FPGA-based device-level simulation

Using the RAM block in the library, we can further store the IGBT's transient data in per unit with the function of time. The FPGA structure is shown in Figs. 9.20 and 9.21. The half-bridge circuit is selected as the modeling unit. It receives two driving signals from both the upper IGBT and the lower IGBT. At each time step, based on the load current and driving signals, the model implements a state machine with three valid states: dead-time state, turn-on state, and

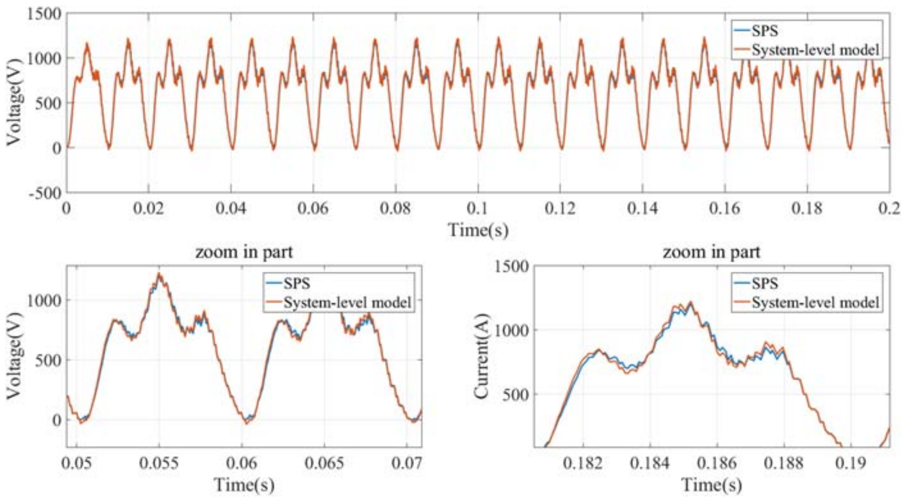


Figure 9.16 The simulation results of voltage U_{C_d}.

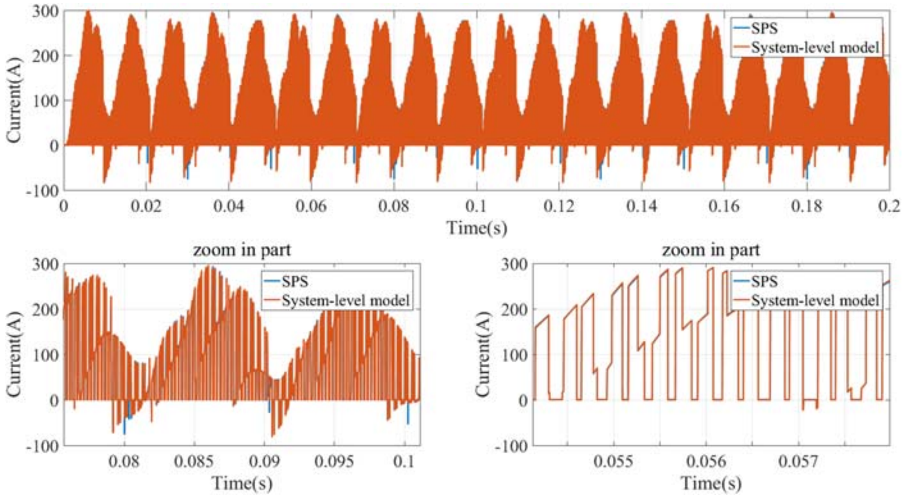


Figure 9.17 The simulation results of current i_{dc}.

turn-off state. With an interval of 10 ns, the appropriate IGBT voltage and current are read from the RAM block. By combining the values of DC link voltage and AC current, the AC output voltage and the DC input current can be computed for each half-bridge. The half-bridge model is duplicated to form the inverter model. The inputs to the inverter model are the three-phase currents, the DC voltages, and six gate signals.

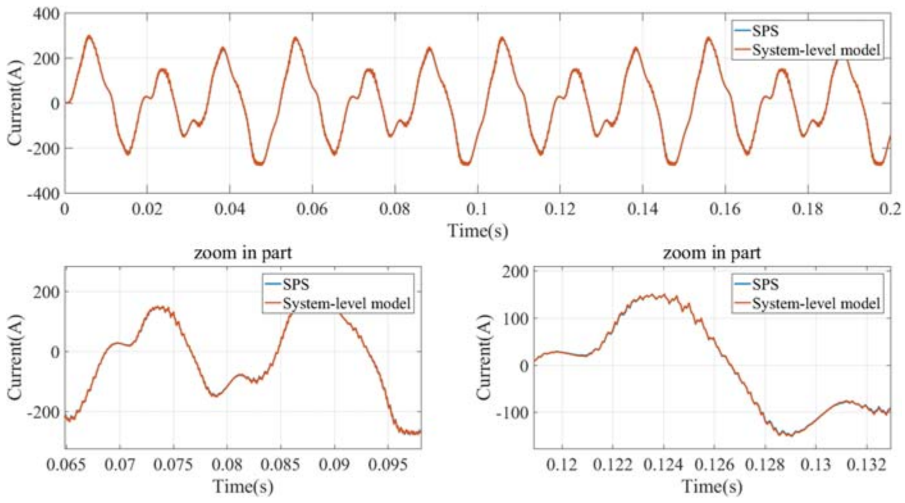


Figure 9.18 The simulation results of current i_{L_a}.

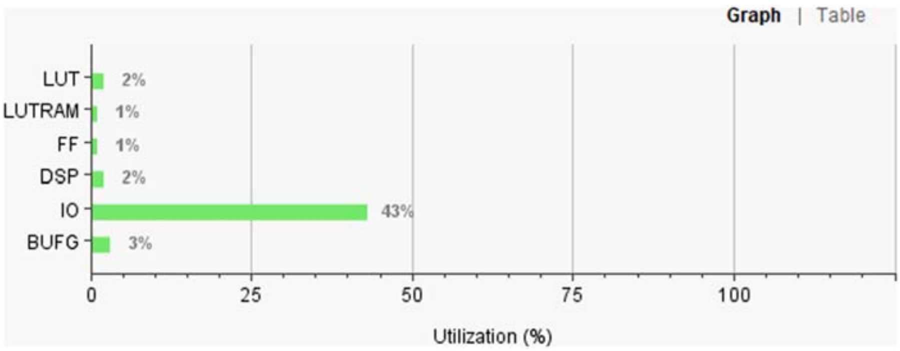


Figure 9.19 Resource utilization on Kintex-7 XC7K325T FPGA.

The overall device-level simulation is implemented on the Kintex-7 XC7K325T FPGA as well. The final device utilization is shown in Fig. 9.22. The FPGA-based device-level simulation results with a comparison to the system-level simulation results are shown in Figs. 9.23–9.28. In Figs. 9.27 and 9.28, it can be observed that in addition to steady-state values of voltage, the switching transient can also be simulated in real time. Regarding the switch performance, the presence of a transient waveform could provide further information about power loss and device stresses. This could further be used to study the multiphysical simulation and the possible effects of EMI.

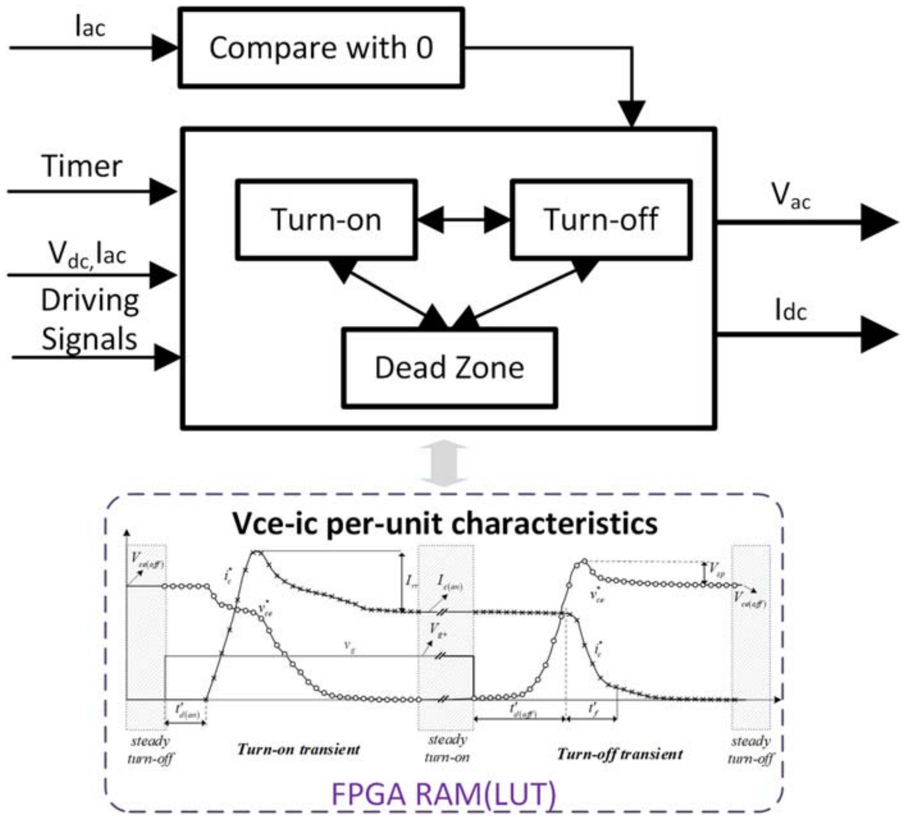


Figure 9.20 Functional blocks of FPGA implementation for one half-bridge of the inverter.

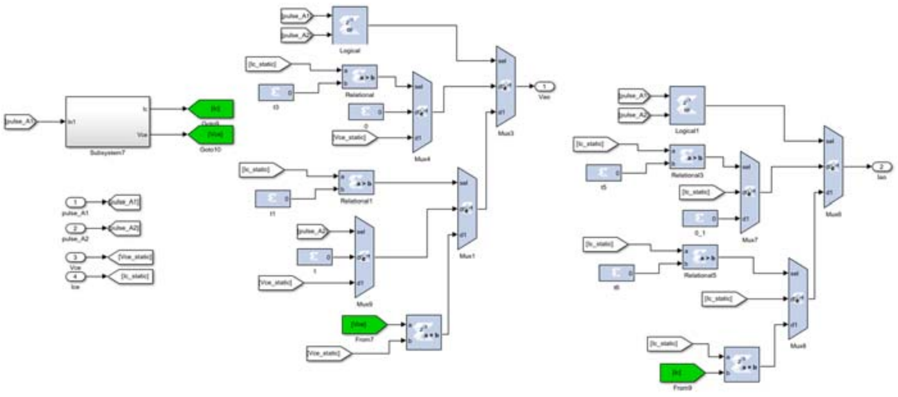


Figure 9.21 One half-bridge model using System Generator.

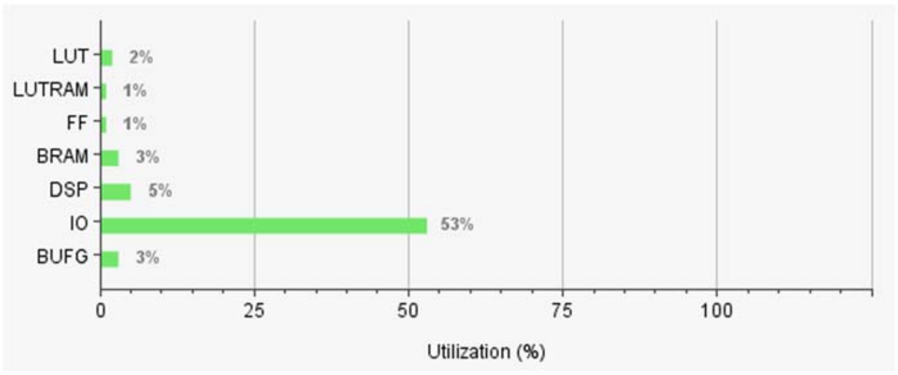


Figure 9.22 Resource utilization on Kintex-7 XC7K325T FPGA.

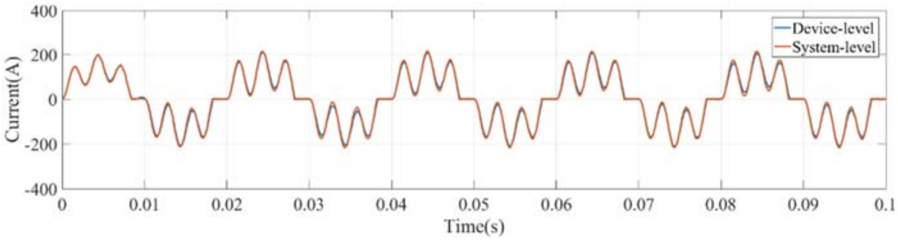


Figure 9.23 The simulation results of current i_{L_s}.

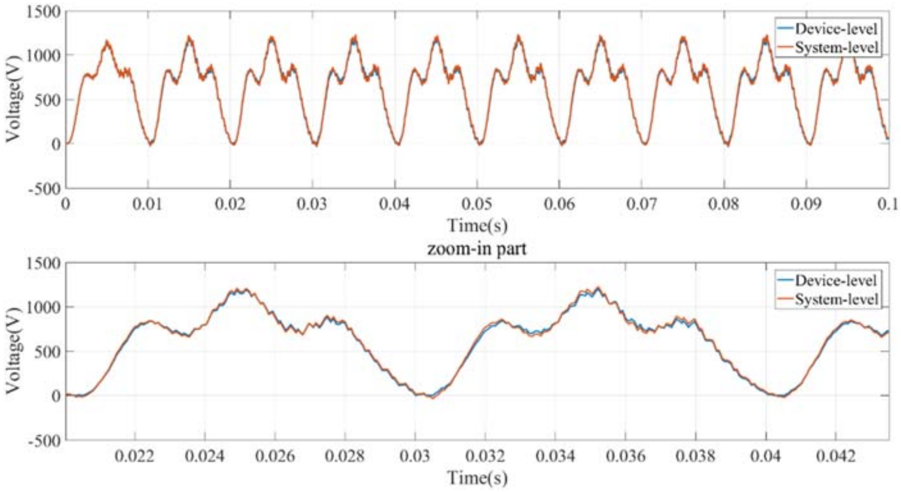


Figure 9.24 The simulation results of voltage U_{C_d}.

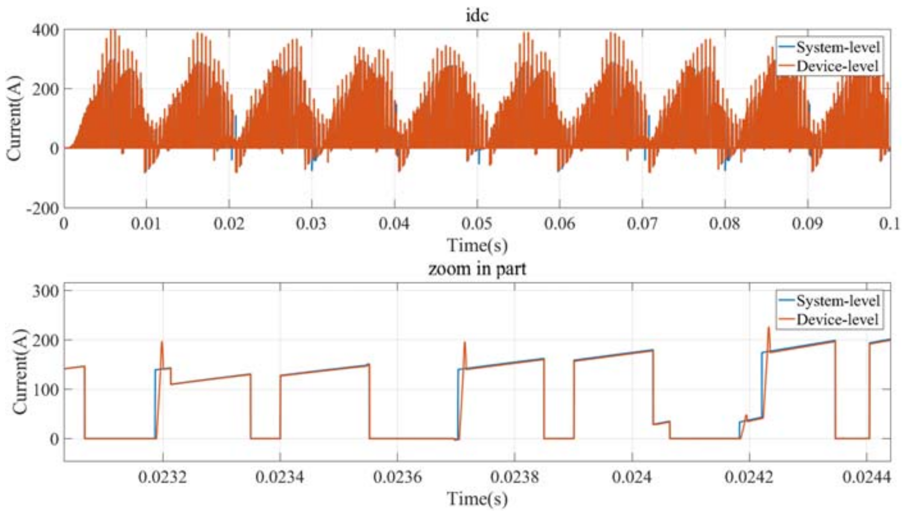


Figure 9.25 The simulation results of current i_{dc}.

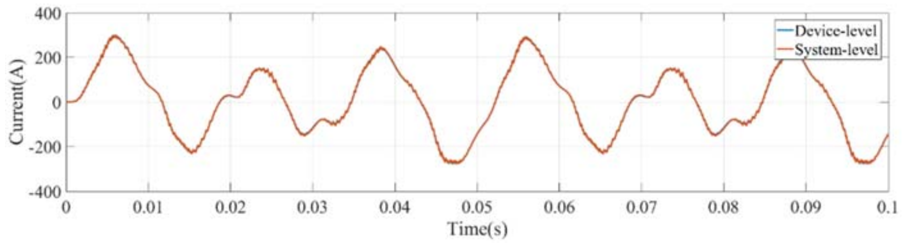


Figure 9.26 The simulation results of current i_{L_a}.

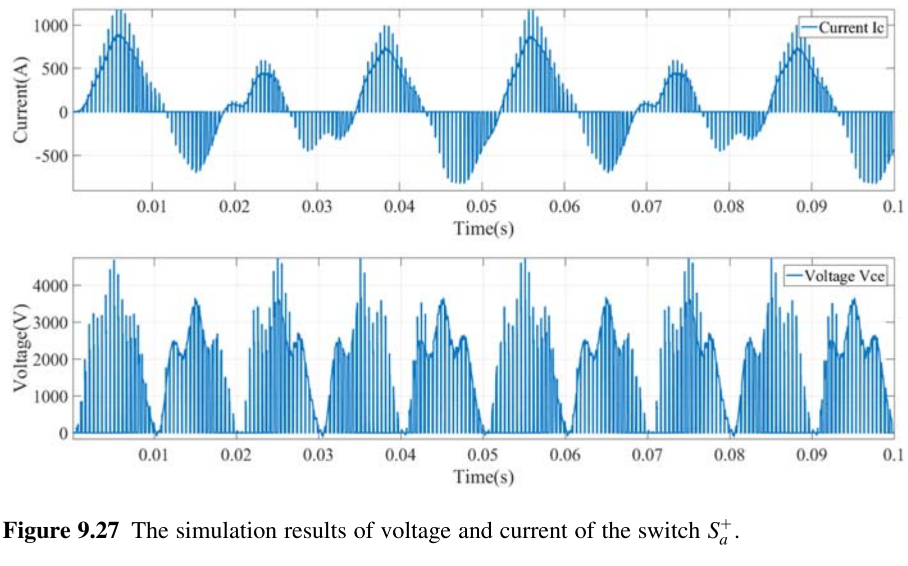


Figure 9.27 The simulation results of voltage and current of the switch S_a^+.

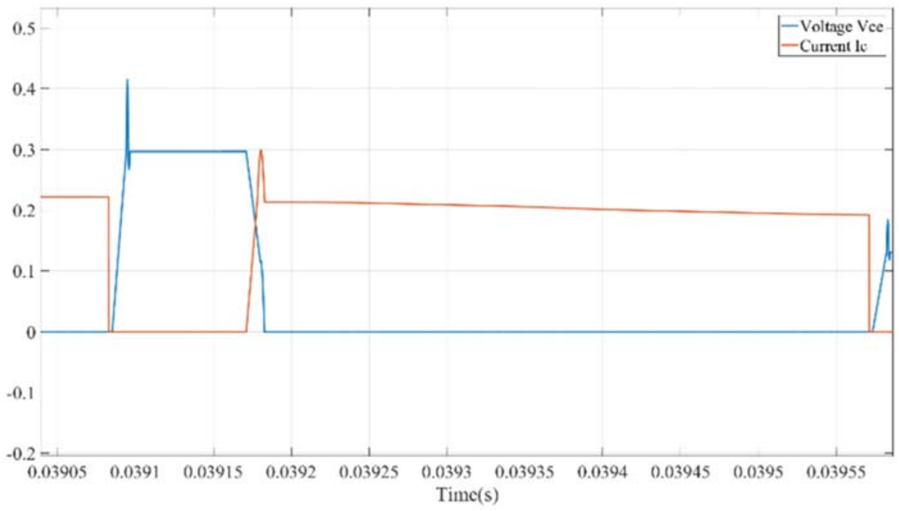


Figure 9.28 The simulation results of switching transient of switch S_a^+ (in per unit, $I_{\text{base}} = 3000$ A, $V_{\text{base}} = 3000$ V).

9.7 Choice of the simulator for the application

As described in Chapters 8 and 9, processor-based and FPGA-based solutions are quite different in time step, implementation process, and so on. Both of them can be applied to the real-time simulation of power electronic systems. For example the processor-based simulator can handle power electronic systems of large scale and/or slow dynamics. In contrast, the FPGA-based simulator is fit for the simulation of a power electronic system with strict requirements on the small time step and fast dynamics. In this section, we will make a brief comparison between the processor-based and FPGA-based simulations.

9.7.1 Comparison between processors and FPGAs

Many power electronic system models inlcude different physical parts that have very different characteristics. Some of them are best to be simulated in FPGA, and others are best to be simulated in processors. The processor is suitable for the parts with low dynamics and high mathematical complexity, while the FPGA is suitable for the parts with fast dynamics and relatively low complexity in the model. Here we provide a rough guideline for the simulator choice:

1. What is the dynamics of the system that needs to be simulated in real time?

   The dynamics defines the required minimum time step of the system. If the minimum time step is lower than 1μs, FPGA is a natural choice.
2. What is the coding language?

   If the system is already modeled in C, a processor may be more direct to implement it in real time. It may not be the solution with the highest performance, but it will be quicker to develop.

Table 9.1 Solver comparisons between the processor-based simulator and the FPGA-based simulator.

	Processor-based simulator	**FPGA-based simulator**
Solver implementation	Flexible in algorithm complexity	Limitation for complex algorithm
System scale for simulation	Suitable for larger system simulations with a large time step	Suitable for smaller system simulations with a small time step
Time-step range	more than 10 μs	25 ns to less than 10 μs
Coding issue	Ease of coding in C or other common programming languages	Relatively easy coding in HLS or System Generator; Tough tasks in HDL for general users
Data representation	Mainly floating-point	Mainly fixed-point

3. What is the required accuracy in data representation?

   If the application needs floating-point representations, the processor may be a good choice. Nevertheless, FPGA is also capable of dealing with the floating-point number if the scale and required simulation time step of target topology are suitable to do so.
4. What kinds of modeling libraries the real-time simulator has?

   Both commercial processor-based and FPGA-based simulators offer libraries for basic building blocks. However, more complex components may not be available, and this could sway the decision from one approach to the other. In Table 9.1, we have summarized the solver's characteristics of the processor-based simulator and the FPGA-based simulator.

9.7.2 Choice of the simulator based on the time-step requirements

In Section 9.7.1, we can see that the dynamics of the system plays a crucial role in the choice of the simulation time step. Therefore in Fig. 9.29, typical time-step requirements for a variety of applications of real-time simulation related to power electronics are outlined. The top side of the chart illustrates fuel cell or Li-ion batteries with slow dynamics that generally require a simulation time step from 300 μs to 10 ms.

A relatively smaller time step is required to maintain numerical stability in power system models and electrical machine models. Especially, when the effect of the third or fifth harmonic in the power system is considered, simulation time steps as low as 200–50 μs may be required to provide acceptable results for transients up to 1 kHz. Furthermore, with an FPGA platform, the simulation time step of electrical machine models can reach as low as hundreds of nanoseconds. In the high-power drive system simulation, the control frequency is usually not high. Thus a larger simulation time step can also provide acceptable accuracy.

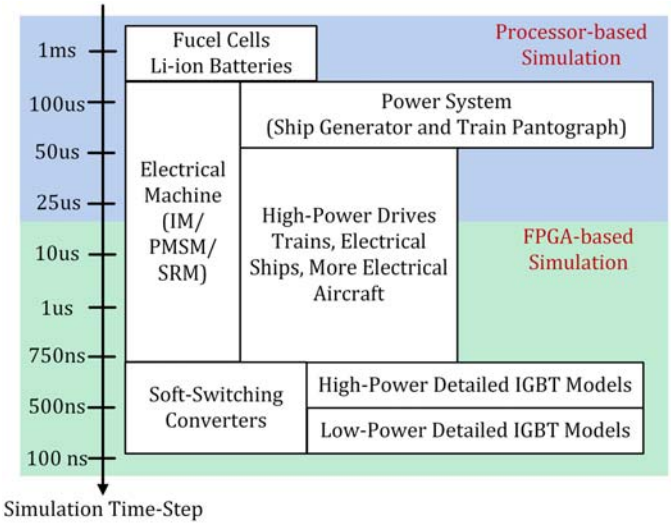

1ms
100us
50us
25us
10us
1us
750ns
500ns
100 ns
Simulation Time-Step
Fucel Cells
Li-ion Batteries
Processor-based Simulation
Power System
(Ship Generator and Train Pantograph)
Electrical Machine (IM/ PMSM/ SRM)
High-Power Drives Trains, Electrical Ships, More Electrical Aircraft
FPGA-based Simulation
Soft-Switching Converters
High-Power Detailed IGBT Models
Low-Power Detailed IGBT Models

Figure 9.29 Simulation time steps in different applications.

In high-frequency transient applications, like soft-switching converters in rail auxiliary power systems, the stiff problem exists due to the presence of different time constants in the whole system. The time step is usually less than 750 ns to ensure the representation of the transient behavior and the numerical stability of the whole system model simulation.

Accurately simulating the transient behavior of the IGBT device requires the use of much smaller time steps. Since the duration of the transient behavior is usually in the range of microseconds, the simulation of a detailed high-power IGBT model is usually in hundreds of nanoseconds. For the low-power IGBT model, the simulation time step could be less than 500 ns or even lower.

Finally, with the development and application of wide-bandgap (WBG) power semiconductor devices, new opportunities are emerging in the powertrain, which in turn requires a much smaller simulation time step of the corresponding models. Many new issues are raised and need to be addressed to realize the real-time simulation of this new trend in electrified transportation. WBG operates with a switching frequency up to MHz level, which requires a simulation time step below 100 ns. Since state-of-the-art methods and platforms are hardly adapted to time steps below 100 ns, new technologies and exclusive methodologies must be developed to meet these timing requirements.

9.7.3 Introduction to processor-FPGA cosimulation framework

The processor and FPGA have their own merits in implementing real-time simulation. To combine their advantages, the processor-FPGA co-simulation framework can be used. Different from the cases in most of the processor-based simulators where FPGA is only used for the I/O sampling and the processor takes all the computation tasks as do, in the co-simulation framework, the FPGA participates in model computing as well.

In the literature, one idea of co-simulation is to use the FPGA to handle the electromagnetic transient with fast dynamics and to use the processor to solve the large-scale system with slow dynamics. Therefore the fast dynamics and the slow dynamics are simulated in FPGA and the processor, respectively, with very different time steps. In such a setup, communication delays are inevitably introduced between the two processing elements.

For example, in the HVDC transmission, there are two levels of decoupling. As shown in Fig. 9.30A, two MMCs are first decoupled using the transmission line model presented in Chapter 6. Moreover, for each MMC the transient of the submodule (SM) is simulated in FPGA, while the rest of the MMC and the grid are simulated in CPU, as shown in Fig. 9.30B. By applying this decoupling method, a short transmission line is introduced to decouple the SMs from the rest of the MMC. This approximation has thus introduced fictive line capacitances that are not present

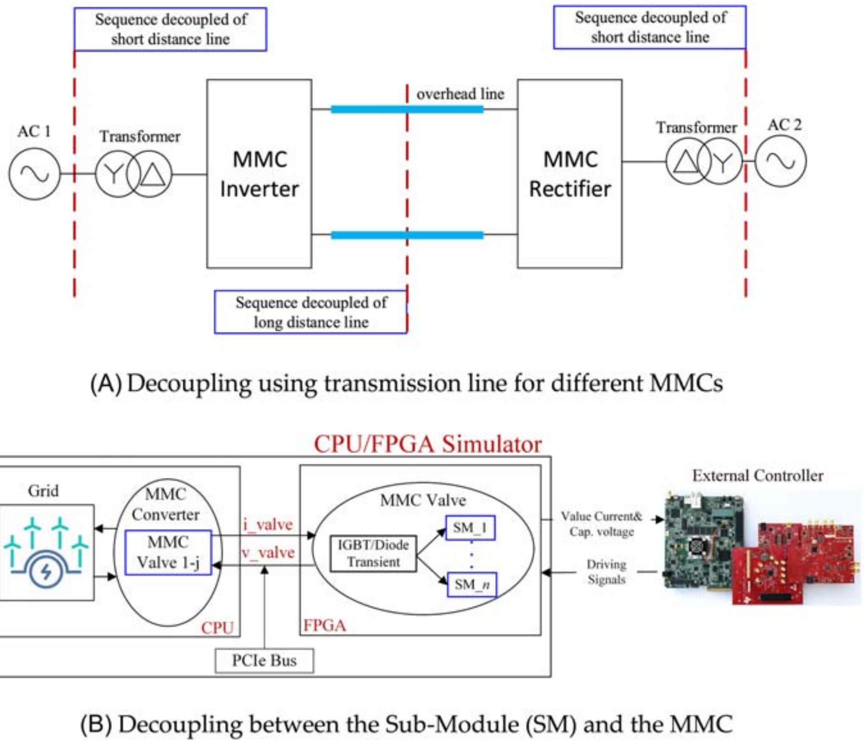

Sequence decoupled of short distance line
Sequence decoupled of short distance line
overhead line
AC 1
Transformer
MMC Inverter
MMC Rectifier
Transformer
AC 2
Sequence decoupled of long distance line
(A) Decoupling using transmission line for different MMCs
CPU/FPGA Simulator
Grid
MMC Converter
MMC Valve 1-j
i_valve
v_valve
MMC Valve
IGBT/Diode Transient
SM_1
SM_n
CPU
FPGA
PCIe Bus
External Controller
Value Current& Cap. voltage
Driving Signals
(B) Decoupling between the Sub-Module (SM) and the MMC

Figure 9.30 Decoupling of HVDC transmission system under cosimulation framework. (A) Decoupling using transmission lines for different MMCs. (B) Decoupling between the submodule and the MMC.

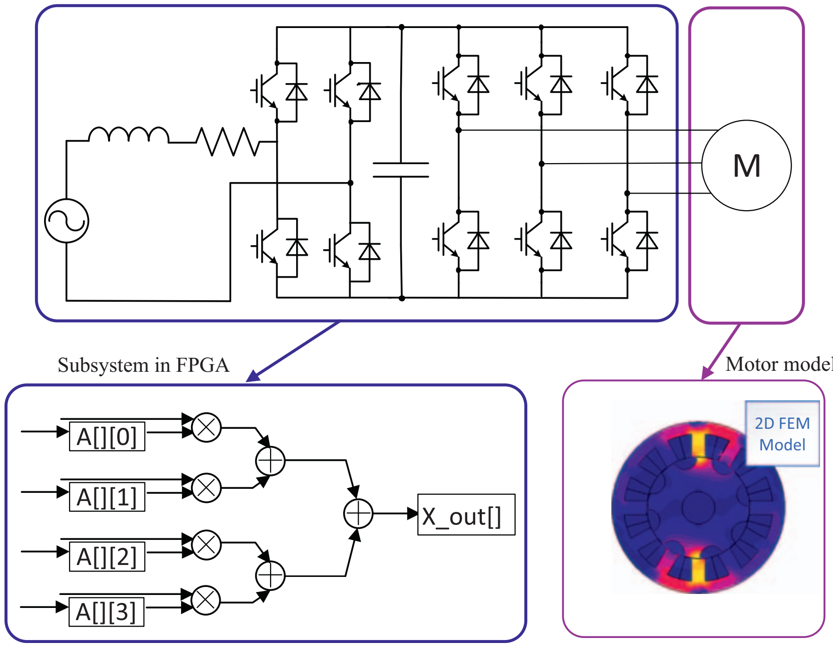


Figure 9.31 Structure of CPU/FPGA cosimulation using detailed motor model.

in the real circuit. However, if the time step is small enough, the influcence of fictive capacitances are also small, and the error can thus be neglected.

Another case suited for co-simulation is motor drive, as shown in Fig. 9.31. In the HIL application associated with motor control, it is a good choice to implement the motor model and power electronic converter model in CPU and FPGA, respectively. On one hand the motor has much slower dynamics than the power electronics. On the other hand the motor involves the complex conversion between electrical and mechanical domains through the magnetic field. It needs the high floating-point computing power of the CPU for accurate representation of motor model. The distinct different time scales between the motor and converter make it possible to do the multirate simulation in a co-simulation framework.

References

[1] F. Montano, T. Ould-Bachir, J.P. David, An evaluation of a high-level synthesis approach to the FPGA-based submicrosecond real-time simulation of power converters, IEEE Transactions on Industrial Electronics 65 (1) (2017) 636–644.
[2] R. Nane, V.M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, et al., A survey and evaluation of FPGA high-level synthesis tools, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 35 (10) (2015) 1591–1604.

[3] Vivado Design Suite User Guide: Model-Based DSP Design Using System Generator, [Online]. 2019. Available: https://china.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2019_2/ug897-vivado-sysgen-user.pdf.
[4] Xilinx, System Generator for DSP, Getting Started Guide, UG639, 2012. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx14_7/sysgen_gs.pdf.
[5] National Instruments Corp., More Throughput vs. Less Latency: Understand the Difference, 2022. Available: https://www.ni.com/zh-cn/innovations/white-papers/13/make-it-faster--more-throughput-or-less-latency-.html.
[6] National Instruments Corp., FPGA Fundamentals: Basics of Field-Programmable Gate Arrays, 2022. Available: https://www.ni.com/zh-cn/innovations/white-papers/08/fpga-fundamentals.html.
[7] D. Majstorovic, I. Celanovic, N.D. Teslic, N. Celanovic, V.A. Katic, Ultralow-latency hardware-in-the-loop platform for rapid validation of power electronics designs, IEEE Transactions on Industrial Electronics 58 (10) (2011) 4708–4716.
[8] Majstorovic, D., Pele, Z., Kovacevic, A., & Celanovic, N. Computer based emulation of power electronics hardware. In Proceedings of the First IEEE Eastern European Conference on the Engineering of Computer Based Systems,September 2009, pp. 56–64. IEEE.

Case studies of power electronics real-time simulations in industrial applications

10

10.1 Introduction

In this chapter, three case studies of real-time simulation for industrial applications are presented: the drivetrain of an electric vehicle, a microgrid with distributed renewable energy resources, and a controller-HIL testing setup for battery energy storage systems.

A real-time simulator with a hybrid architecture containing both central processing units (CPUs) and a multicore field programmable gate array (FPGA) is used. According to the purpose of each simulation and the respective testing requirements, the models are built considering different approaches presented throughout the book, with different utilizations of the CPU and FPGA resources.

10.2 Real-time simulation in electric vehicle applications

In this example, a generic battery electric vehicle drivetrain is simulated in real time. A simplified schematic of the model is illustrated in Fig. 10.1, encompassing

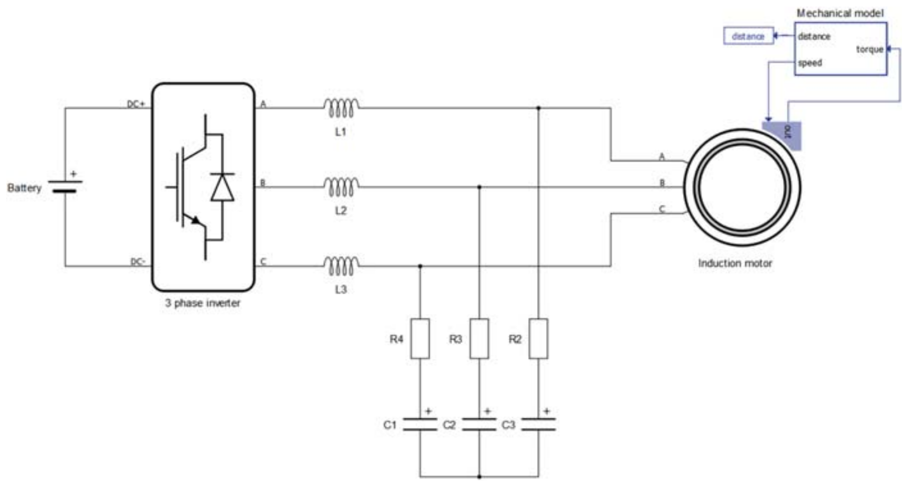


Figure 10.1 Electric vehicle powertrain.

Real-Time Simulation Technology for Modern Power Electronics. DOI: https://doi.org/10.1016/B978-0-323-99541-2.00010-7

a 400 V, 250 Ah Li-ion battery, a two-level three-phase inverter with a switching frequency of 20 kHz, an LC filter, and a 270 kW squirrel cage induction motor. The main purpose of this example is to demonstrate the use of the models described throughout the book on a system level.

The passive elements that comprise the LC filter are modeled as presented in Chapter 4.2, with $L = 1\mu H$, $C = 1000\mu F$, and $R = 5m\Omega$

Based on Chapter 4.3.3 the battery is implemented with a simplified model comprising a controlled voltage source with a series resistance ($R_{int} = 0.04\Omega$). The battery voltage is defined as a function of the state of charge according to the equations provided in the respective chapter. The parameters are given in the Table 10.1. Higher-order effects such as chemical diffusion processes, voltage hysteresis, and temperature dependance are neglected.

Since in this example the objective is to validate the complete motor drive control scheme, it is important to accurately emulate the switching behavior of the inverter. Taking that into account and also the computing time constraints of real-time simulations, power electronic switching devices are modeled here as ideal switches, as described in Chapter 3.2.3, and illustrated in Fig. 10.2.

A piecewise linear approach is used to model the plant on a circuit level. For every reachable permutation of switching states (possible states of the semiconductor switches), there is an associated circuit topology, which is described using the

Table 10.1 Parameters of the simplified battery model.

Parameter	Value	Parameter	Value
V_{rated}	400 V	Q_{rated}	250 Ah
V_{full}	116%	Q_{nom}	93.5%
V_{exp}	103%	Q_{exp}	85%

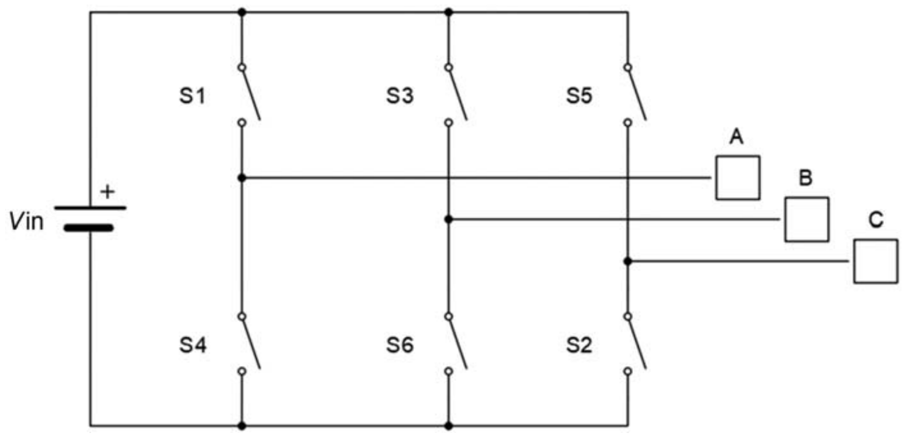


Figure 10.2 Illustrative example of a three-phase inverter modeled with ideal switches.

state-space equations, as shown in Chapter 5. The trapezoidal method is used to obtain the discrete-time representation of each topology. The representations are calculated offline as part of the model compilation process. During the simulation runtime, the solver dynamically switches among the precomputed topologies according to the current states of the switching devices.

The three-phase squirrel cage induction motor model uses the current source interface, where the classic squirrel cage induction machine model equations are formulated such that the stator voltages are inputs, while stator currents are outputs of the model [1]. Nonlinearities such as magnetic saturation effects are neglected in this example but could be considered and specified as lookup tables (flux vs current, for instance).

The current source interface and some of the machine solver features are shown in Fig. 10.3. Although the machine modeling is not covered in this book, the machine parameters are provided in Table 10.2, for reference.

The motor drive control scheme consists of a simple indirect field-oriented control strategy, in which the three-phase currents are transformed to the synchronous reference frame (Park transform) and then controlled using PI regulators. The machine flux is related to the *d*-axis current, while the torque is related to *q*-axis current. The rotor flux angle, needed for the reference frame transforms, is calculated from the machine electrical parameters, rotor speed, and measured currents. Additional control logic is used to limit the power and torque according to the motor operation points.

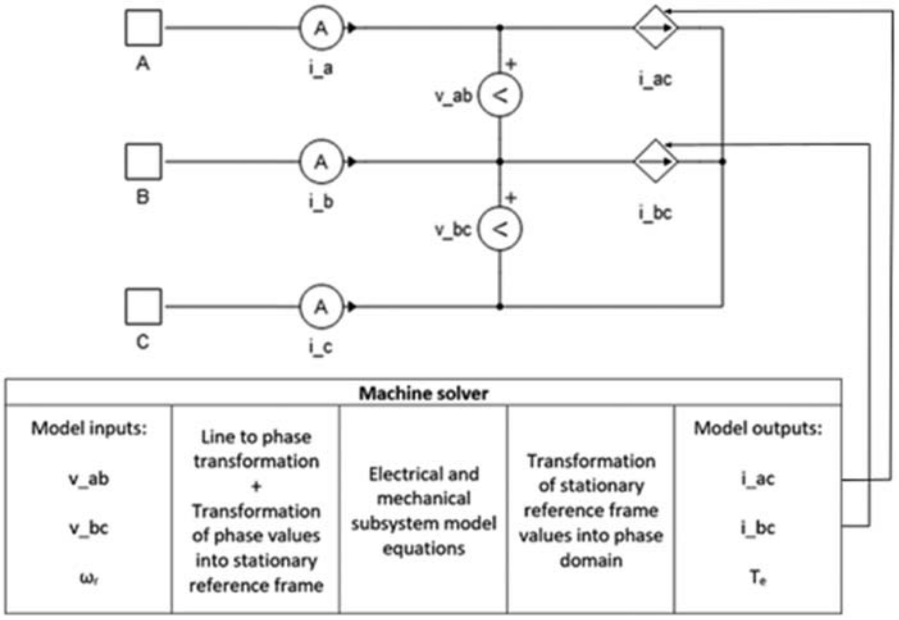

A
B
C
A
i_a
A
i_b
A
i_c
v_ab
v_bc
+
+
i_ac
i_bc
Machine solver
Model inputs:
v_ab
v_bc
ω_r
Line to phase transformation + Transformation of phase values into stationary reference frame
Electrical and mechanical subsystem model equations
Transformation of stationary reference frame values into phase domain
Model outputs:
i_ac
i_bc
T_e

Figure 10.3 Machine circuit interface and FPGA-based solver.

Table 10.2 Parameters of the induction motor.

Parameter	**Value**	**Parameter**	**Value**
Stator phase resistance	3mΩ	Magnetizing inductance	500μH
Rotor phase resistance	2.4mΩ	Number of pole pairs	2
Stator leakage inductance	19μH	Moment of inertia	1.0kgm^2
Rotor leakage inductance	19μH	Viscous friction coefficient	0Nms

Due to the high switching frequency used in this example, the electrical part of the model (battery, passive elements, and inverter) is simulated using an FPGA-based solver, which allows to minimize the simulation time step. A dedicated FPGA module is responsible for simulating the machine model, allowing it to run with the same simulation time step as the rest of the electrical circuit. The PWM modulator is implemented in a separate FPGA module operating at the FPGA fundamental clock frequency, which allows for high resolution of the generated gate drive signals including the ability to generate subsimulation step deadtime periods. The worst case model calculation time on the used platform, running at the clock frequency of 280 MHz, is around 298 ns. This time defines the minimum achievable simulation step for the model with no risk of overruns due to the deterministic nature of the FPGA solver. The actual simulation time-step used in this example is set to 500 ns, which means that the worst case time slot utilization is around 58%.

The control scheme is implemented within the model using standard functional blocks (e.g., gain and integrator) and C-code-based components, running on the CPU with the computation cycle of 50 μs, which corresponds to the PWM period. The worst case measured CPU time slot utilization in this example is around 10.3%, which ensures stable real-time operation and provides significant headroom for potential model expansion.

10.2.1 Real-time simulation results

In this simulation, the input is given by the throttle position (from 0% to 100%), which defines the torque reference for the induction motor and, therefore, the q-axis current reference to be synthesized by the three-phase inverter. Additional logics are included within the controller in order to

1. Limit the acceleration power to 270 kW.
2. Limit the acceleration torque to 860 Nm.

Assuming that the electric vehicle is initially at a standstill, a step in the throttle input from 0% to 100% is applied in the simulation. As mentioned before the objective here is to validate the inverter-based motor drive control. Therefore it is important to have real-time simulation results that faithfully represent the inverter switching dynamics.

The transient three-phase currents and voltages at the output of the LC filter are shown in Fig. 10.4, with zoomed waveforms shown in Fig. 10.5A and B. The respective PWM output voltages are shown in Fig. 10.5C and D.

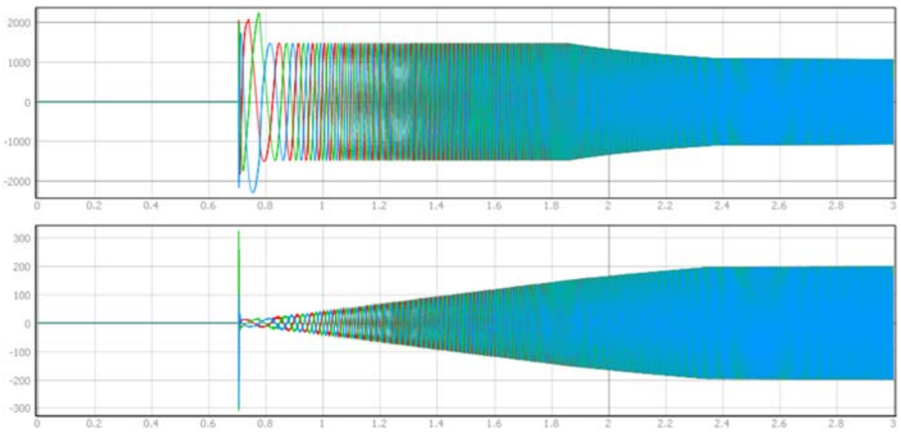


Figure 10.4 Transient responses at the LC filter output under acceleration with 100% throttle: top, three-phase currents (A); bottom, three-phase voltages. (Horizontal scale = 200 ms/div).

Fig. 10.6 shows the current at the battery, the inverter output voltages, and the currents through the filter inductor with a horizontal scale of 20 μs/div. One can highlight the fast dynamics caused by the inverter switching as well as the fluctuations in the battery voltage (DC link), which is according to what is expected from practical applications.

To verify the overall system performance, Fig. 10.7 shows the torque versus speed and power versus speed responses for a step change in the throttle input from 0 to 100%. From the voltages and currents at the machine armature (output variables of the LF filter), power, torque, and speed are calculated using classic electrical and mechanical motor model equations, implemented using the machine solver shown in Fig. 10.3.

Notice that the maximum torque (860 Nm) is applied up to a speed about 40 km/h, which is the base speed for the maximum torque and maximum power (270 kW). Following the implemented control algorithm, above this speed, the torque reference is reduced to maintain a constant power region, which goes up to 250 km/h (electronically limited speed).

Fig. 10.8 shows the speed response of the EV over time under acceleration with 100% throttle input. It is possible to see that the EV accelerates from 0 to 100 km/h in about 4 s, reaching the final speed of 250 km/h in 30 s, with a small overshoot before it reaches the steady state.

For comparison purpose, the results under acceleration are shown in Fig. 10.9 for throttle positions of 100%, 80%, 50%, and 30%. The control logic implemented in this example leads to torque and power values that are proportional to the throttle input, while the speed up to which the torque is held constant is the same for all cases. Similar curves can be obtained for the deceleration responses.

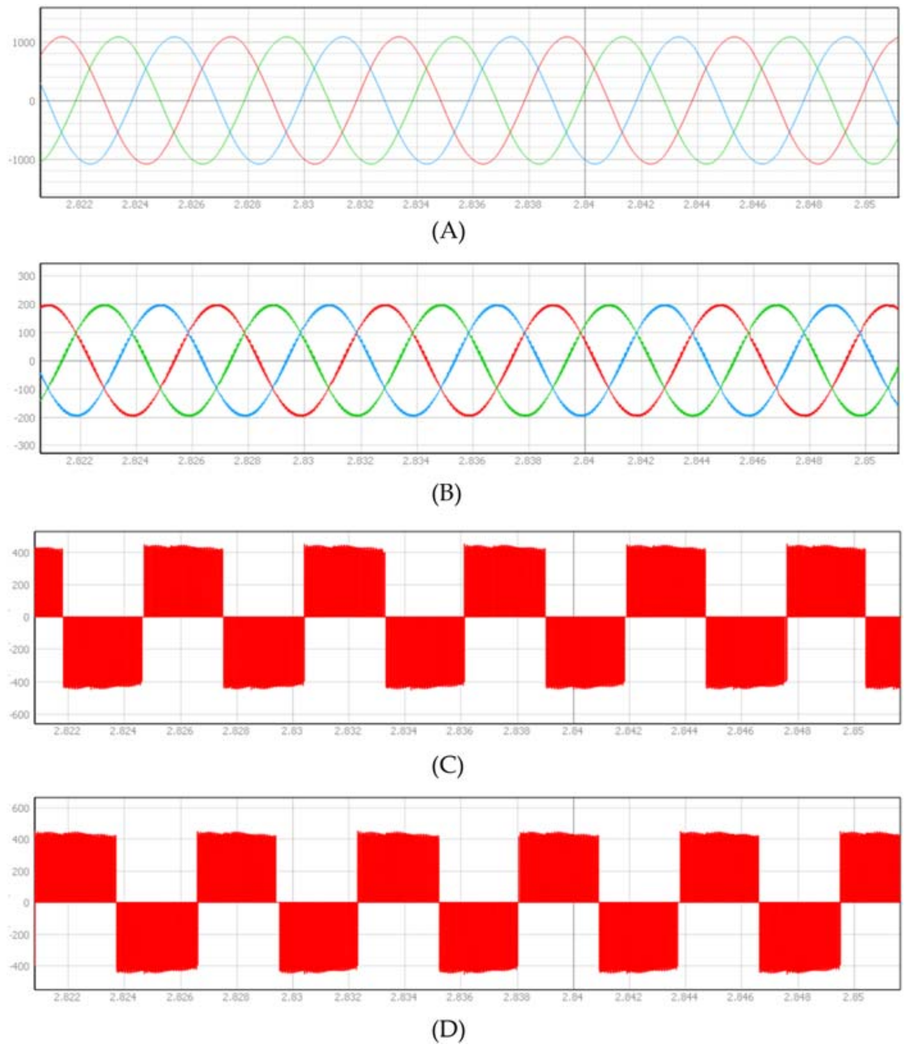


Figure 10.5 (A) Three-phase currents (A) and (B) voltages at the LC filter output; (C) inverter output voltage V_{ab} (V); (D) inverter output voltage V_{bc} (V). (Horizontal scale = 2 ms/div).

10.3 Terrestrial microgrid

In this second example, a three-phase terrestrial microgrid is simulated in real time. Different from Example 10.2, the objective here is to validate the microgrid central controller (energy management system), and the switching dynamics of the power electronics devices can be neglected. Therefore average converter models are used as they require significantly less computation resources.

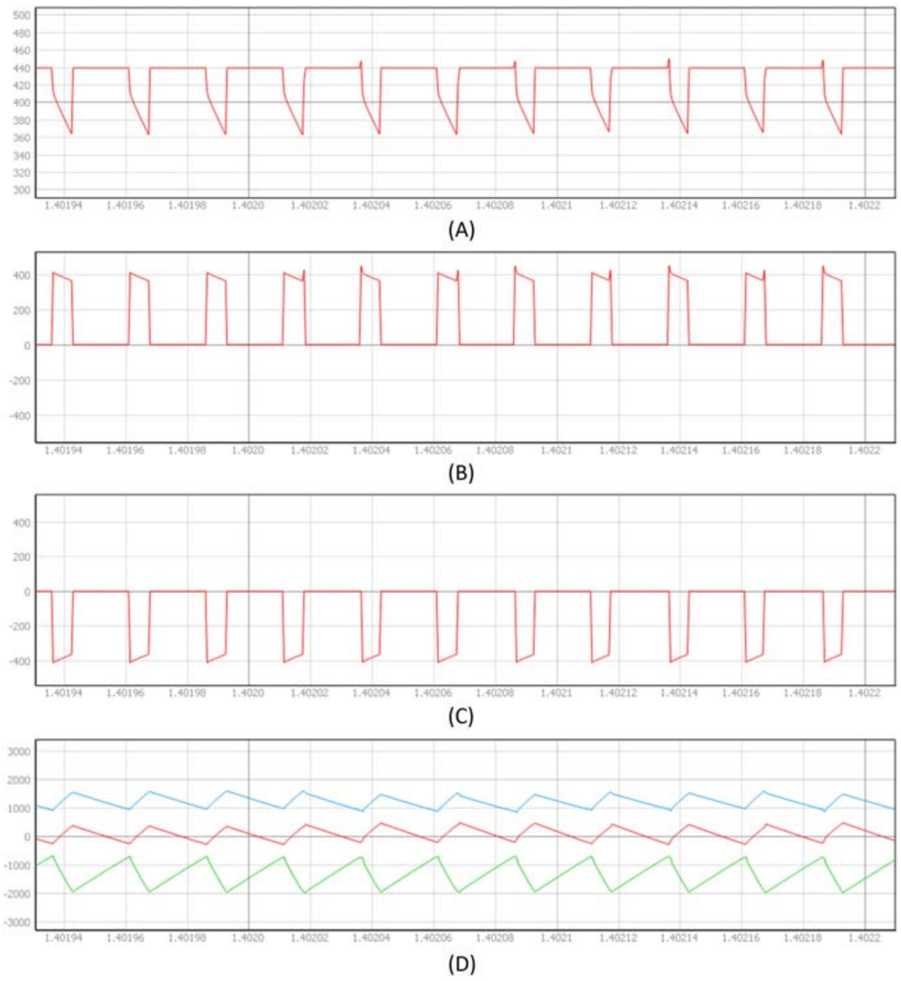


Figure 10.6 Detailed waveforms to highlight the switching dynamics: (A) Voltage at the battery terminals; (B) inverter output voltage V_{ab} (V); (C) inverter output voltage V_{bc} (V); (D) currents through the filter inductor (A). [Horizontal scale = 20 μs/div].

The main purpose of this example is to illustrate how larger systems can be simulated in real time with high fidelity by adapting the modeling approach and optimizing the use of the HIL simulator resources according to the objectives and requirements of the simulation.

The single line diagram of the microgrid is depicted in Fig. 10.10. The microgrid can operate in both grid connected and islanded modes, and it is designed to provide seamless transition between the operation modes.

The microgrid comprises three distributed energy resources (DERs): a 500 kVA wind power plant, a 250 kVA PV power plant, and a 1.6 MVA battery energy

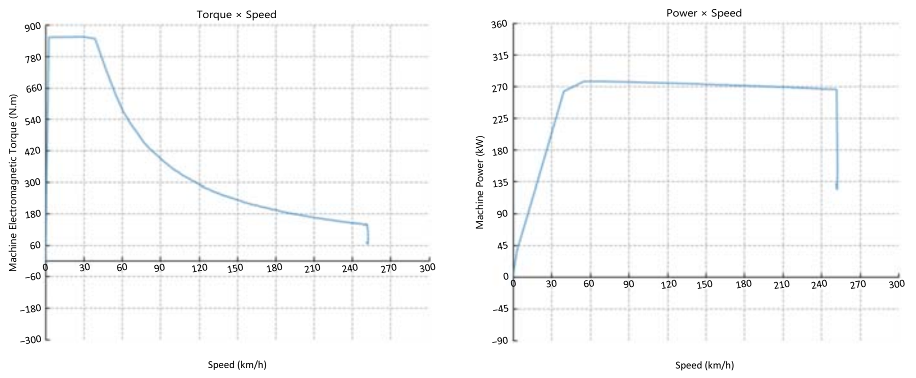


Figure 10.7 Torque and power characteristics under acceleration with 100% throttle.

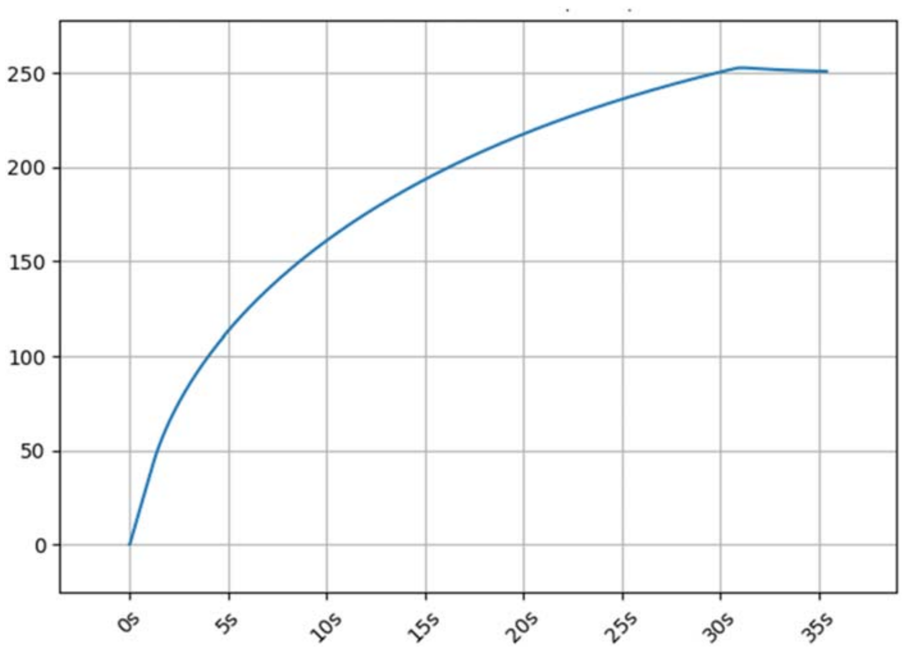


Figure 10.8 EV response under 100% throttle: speed vs time.

storage system (BESS). The BESS has a nominal voltage of 1000 V and a capacity of 500 Ah, being engaged in the grid-forming mode in the case of electricity shortage from the utility grid or intentional islanding. $Z_{\mathrm{line}1}$ and $Z_{\mathrm{line}2}$ represent medium voltage distribution lines, modeled as RL impedances. The rated load is equal to 1.6 MVA.

The modeling approaches for the utility grid, DERs, and load are described in sequence hereafter.

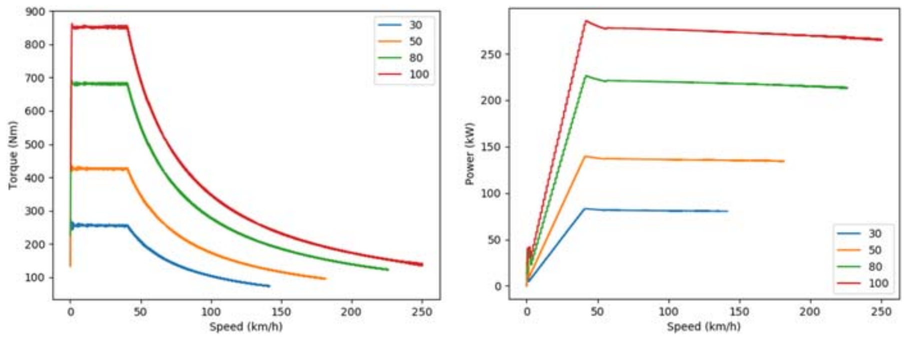


Figure 10.9 Torque and power characteristics under acceleration for different throttle inputs. The total simulation time is 30 s.

10.3.1 Utility grid

The grid model consists of a three-phase voltage source in series with an inductive-resistive impedance and a contactor, as shown in Fig. 10.11. The passive elements are modeled as presented in Chapter 4.2.

10.3.2 Distributed energy resources average model

All three DER components (wind, PV, and BESS) used in this example are based on the same average model building block. It consists of the following: the power stage, the low-level control subsystem, and the high-level control subsystem. The block diagram with the generic structure is shown in Fig. 10.12.

The power stage part of the model is simulated mainly using the FPGA-based solver, while the control subsystems (low-level and high-level) are implemented using standard functional blocks and C-code-based components, running on the CPU.

The input and output signals can be defined according to the DER being modeled, serving as the interface to the microgrid control logic. For instance, the inputs encompass signals such as the enable command and power references as well as the parameter settings. In case of modeling a DER capable of engaging in grid-forming mode, the inputs can also include the converter mode of operation, voltage references, frequency references, and so on. The outputs encompass the feedback variables and alarm messages.

The power stage for a generic DER average model is shown in Fig. 10.13. It comprises the controlled voltage sources (for which the voltage value is provided by the low-level control), the input L filter, the transformer, the contactor that connects the DER to the grid, and appropriate current and voltage measurements.

The low-level control includes the regulators for the inner current control loop (when operating in grid-following mode). The inputs of the current control are the reference currents, provided by the high-level control, and the current measurements

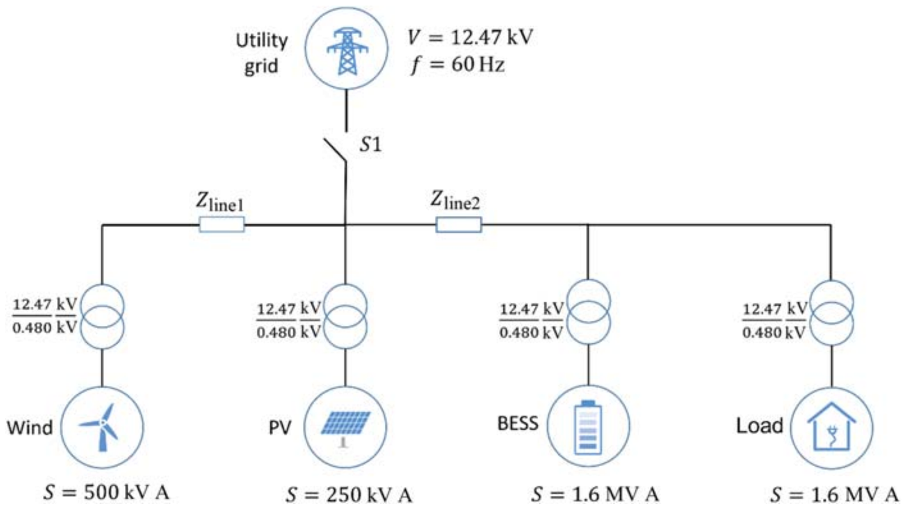


Figure 10.10 Microgrid single line diagram.

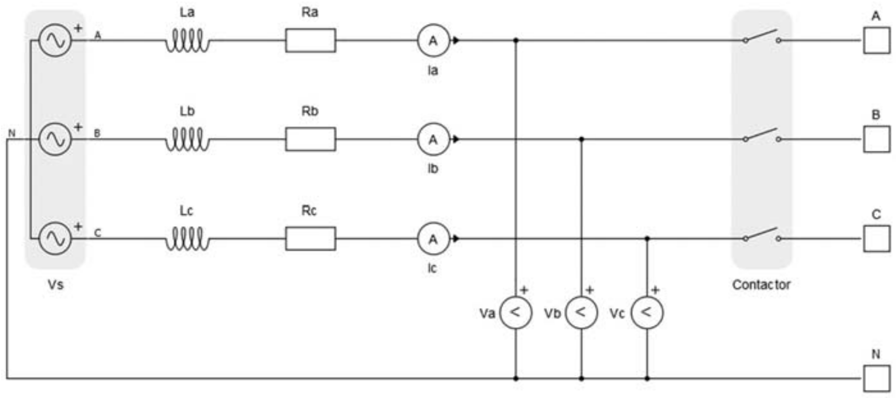


Figure 10.11 Model of the utility grid.

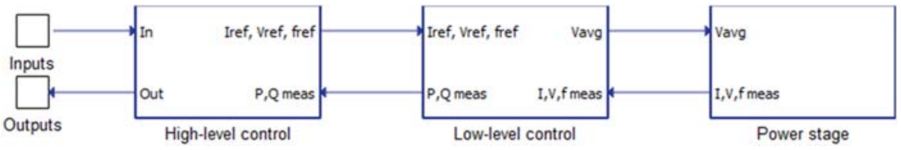


Figure 10.12 Block diagram of the average distributed energy resources model.

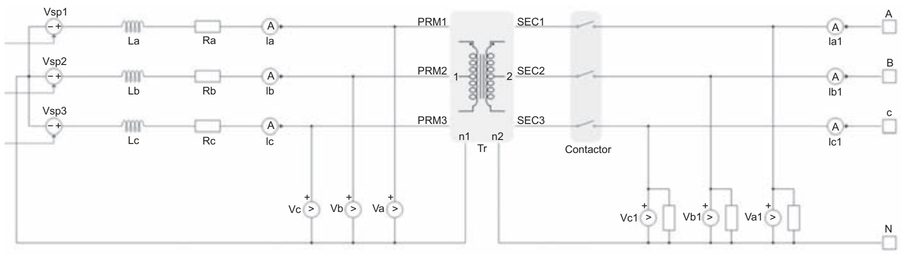


Figure 10.13 Generic power stage model for distributed energy resources.

before the transformer, provided by the power stage. The outputs of the current controller are the average values of the voltage sources in the power stage.

In case the DER can also be engaged in grid-forming mode, the low-level control also includes the voltage/frequency controllers, for which the references are provided by the droop functionalities implemented on the high-level control, and the measurements are provided by the power stage.

The low-level control also includes a state machine for fault detection, which checks the measurements from the power stage and signals faults such as overcurrent protection, overpower protection, and state of charge (SOC) at the critical level.

The high-level control contains mainly the regulators for active and reactive power, which provide the reference output currents based on the power references and the power measurements from the low-level control. It also includes the main state machine, responsible for checking and controlling the overall functionality of the plant. The states of the main state machine are described in Table 10.3.

Additional elements can be included in the high-level control to encompass different functionalities required by specific DERs. For instance, frequency droop and voltage droop functionalities are considered when the DER can engage in grid-forming model. These functionalities will be better detailed later for the BESS.

Although all DERs use the same average model building blocks previously described, in the sequence, additional features in the modeling of the different DERs are detailed.

10.3.2.1 Battery energy storage system

For the BESS model, the SOC is an additional output of the model, calculated according to Chapter 4.3.3, where the battery current is determined according to the reference power inputs. Based on the SOC the battery open-circuit voltage is determined using lookup tables and also provided as output of the model.

In this example, the BESS is the only DER capable of operating in grid forming mode. For that purpose the high-level control includes the frequency and voltage droop functionalities, and the low-level control includes the frequency and voltage controllers.

Table 10.3 Main state machine of the distributed energy resources.

State	Description
Starting up	State of the converter between the signal activation and the start of converter operation. It includes, for instance, system checking and the synchronization with the grid.
Running	Operational state of the converter.
Disabled	Reports if the converter is not enabled or it is not in operation.
Fault	State of the converter if a fault is identified.

When the droop mode is activated, the frequency droop algorithm is responsible for calculating the frequency reference based on the active power error, while the voltage droop is responsible for calculating the voltage reference based on the reactive power error. The specific droop parameters (nominal frequency/voltage, offset, and coefficient) are provided by the user as inputs to the model.

The maximum and minimum allowed SOC of the battery are also additional inputs of the BESS model, used to constrain the power reference if necessary.

10.3.2.2 PV power plant

For the PV power plant model, the high-level control includes as an additional feature a simplified maximum power point tracking (MPPT) algorithm, which changes the operating voltage as a function of the solar irradiance and temperature (external inputs) using an I-V lookup table, as shown in Chapter 4.3.5. This strategy ensures extraction of the maximum available power as the operating point changes.

Additional inputs limit the maximum output power (curtailment) and set the rate of change of the active power, emulating the dynamics to reach the maximum power point when the operating point changes.

10.3.2.3 Wind power plant

For the wind power plant model, the high-level control includes as an additional feature a lookup table which defines the active power production as a function of the wind speed (external input). This dependency is defined based on the equations and parameters detailed in Chapter 4.3.6. Additionally, from the voltages and currents measured at the power stage the actual generated power is calculated and, using lookup tables, the angular speed of the rotor is obtained.

10.3.3 Variable load

The variable load is modeled using a structure similar to the one presented in Fig. 10.12, with power modeled using the same circuit shown in Fig. 10.13. In this case the average value of the voltage sources is defined such that the load meets the chosen active and reactive power setpoints (external inputs).

10.3.4 Microgrid controller

The microgrid controller (MC) is responsible to monitor and issue commands to all the DER units as well as to command the synchronization relay S1 according to the operation mode requested by the microgrid operator or if a grid fault is detected. In case of intentional islanding, the microgrid controller handles the power flow regulation through the PCC by sending power references to the BESS, such that the active and reactive power flowing through the PCC can be zeroed, allowing for a smooth transition to from the grid connected to the islanding mode.

10.3.5 Real-time simulation results

The microgrid model in this example is simulated in real time using the same platform from Example 10.2, with a clock frequency of 280 MHz. On the other hand, now, to take advantage of the multicore FPGA solver and the parallel computing, the model is divided into three cores, where the subcircuits are solved simultaneously: The utility grid and the power stage of the PV power plant are simulated in core 1, the power stage of the wind power plant is simulated in core 2, and the power stage of the BESS and the variable load are simulated in core 3. The subcircuits are partitioned using the transmission line modeling (TLM) applied to the line impedances Z_{line1} and Z_{line2}, as shown in Chapter 6.4.1.

The simulation time step in this example is set to 4 μs. In the worst case the time slot utilization is around 26% for core 1, 10% for core 2, and 27% for core 3, which ensures stable real-time operation.

The control subsystems (low-level and high-level) of the DER average models run on the CPU with a computation cycle of 200 μs, for which the worst case measured for the CPU time slot utilization is around 16%. The MC is fully implemented within the model, also running in real time on the CPU but with a computation cycle set to 1 ms, resulting in a worst case time slot utilization of 6%.

Notice that given the choice to model the system using average model building blocks, although the simulation encompasses multiple DERs and the entire microgrid control logic, the time slot utilization of the FPGA cores and the CPU is low, providing significant headroom for potential model expansion.

10.3.5.1 Case 1

In this first case, the microgrid operates connected to the utility grid. Fig. 10.14 shows the active and reactive power for the DERs and the utility grid. In the

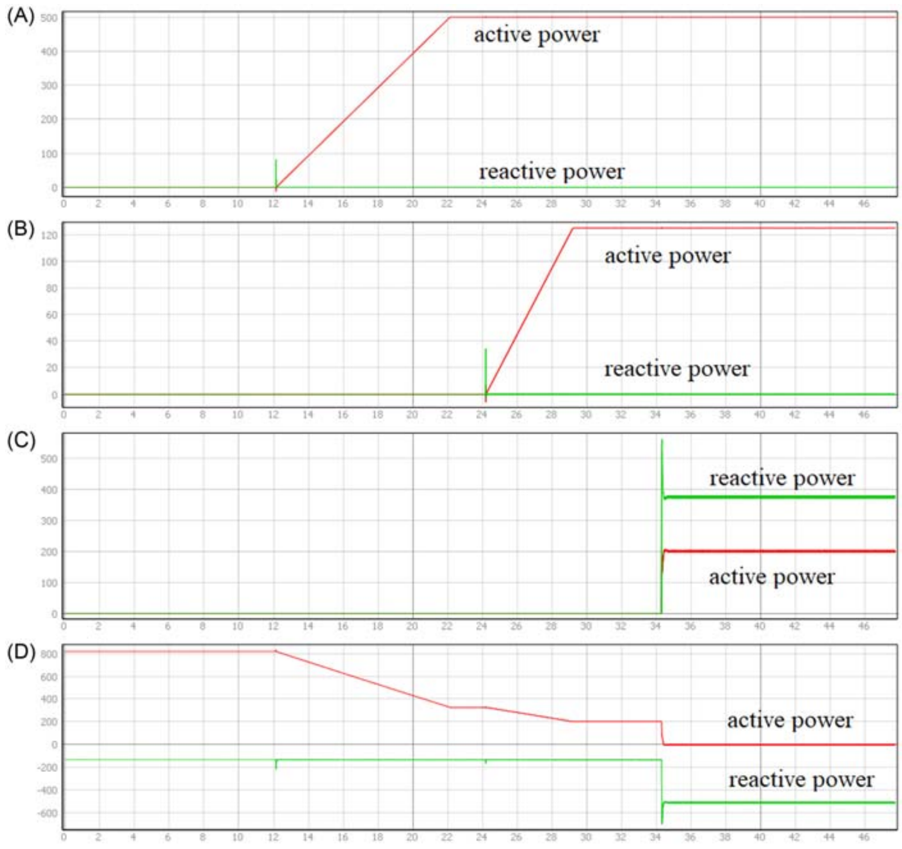


Figure 10.14 Active and reactive power: (A) wind power plant, (B) PV plant, (C) BESS, and (D) utility grid.

beginning the utility grid supplies the load, set to 800 kW with a unity power factor. At the time instant around 12 s the wind power plant is turned on, considering an average wind speed equal to 16 m/s. The rate of change of the active power reference is limited to 10 pu per second until reaching the nominal power (500 kW). In sequence the PV power plant is turned on. The active power reference is set to 125 kW (curtailment of 50%), with the rate of change limited to 0.5 pu/s. The irradiance is set to 1000 W/m^2, and the ambient temperature to 25°C.

Finally the BESS is turned on with an active power reference of 200 kW. The voltage droop is enabled, and the battery provides around 375 kVAr as reactive power support to the grid (ancillary service) to compensate the voltage drop.

Once all the DERs are turned on, one can see that the active power flowing through the point of common coupling with the utility grid is close to zero.

10.3.5.2 Case 2

The results for this second case study are shown in Fig. 10.15. Around 3 s after the simulation starts the load is increased gradually until reaching 1.6 MW, with a unitary power factor. Since the active power references of the DERs do not change at this moment, one can see in Fig. 10.15C that the active power flowing from the grid increases to compensate the load.

At 15 s the intentional islanding of the microgrid is requested, changing the operation mode of the controller from grid-connected (0) to islanded (1), as can be seen in Fig. 10.15A. The microgrid controller then regulates the power flow through the PCC by increasing the active power reference of the BESS, as can be seen in Fig. 10.15B, such that the active power flowing from the grid can be zeroed. Once this is achieved (around 18 s), one can see from Fig. 10.15A that the contactor connecting the microgrid to the utility grid is turned off, with the status going from 1 to 0. The currents of the DERs and of the grid for one of the phases during these transients are shown in Fig. 10.16.

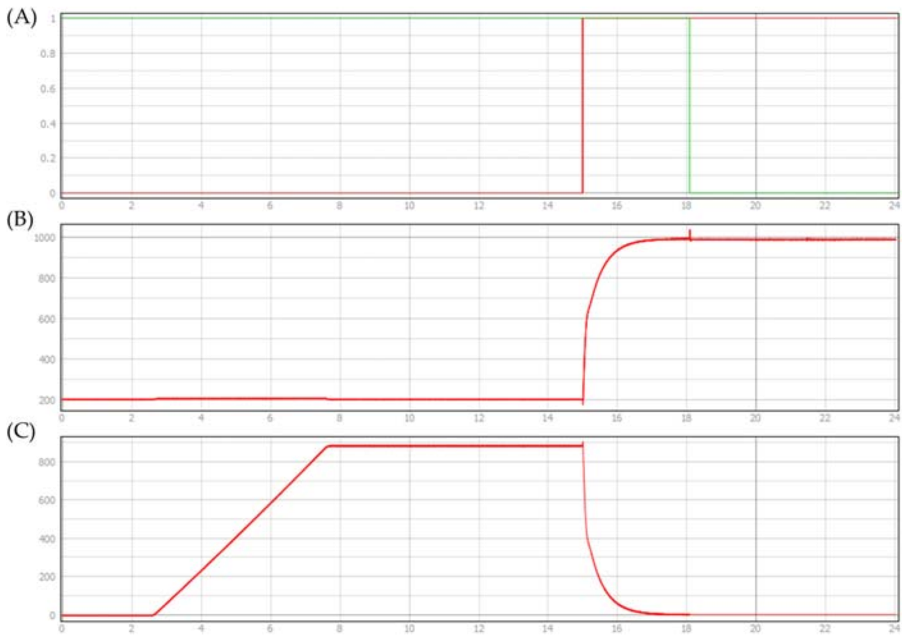


Figure 10.15 Islanding test: (A) operation mode and status of the contactor connecting the utility grid, (B) BESS active power, and (C) utility grid active power.

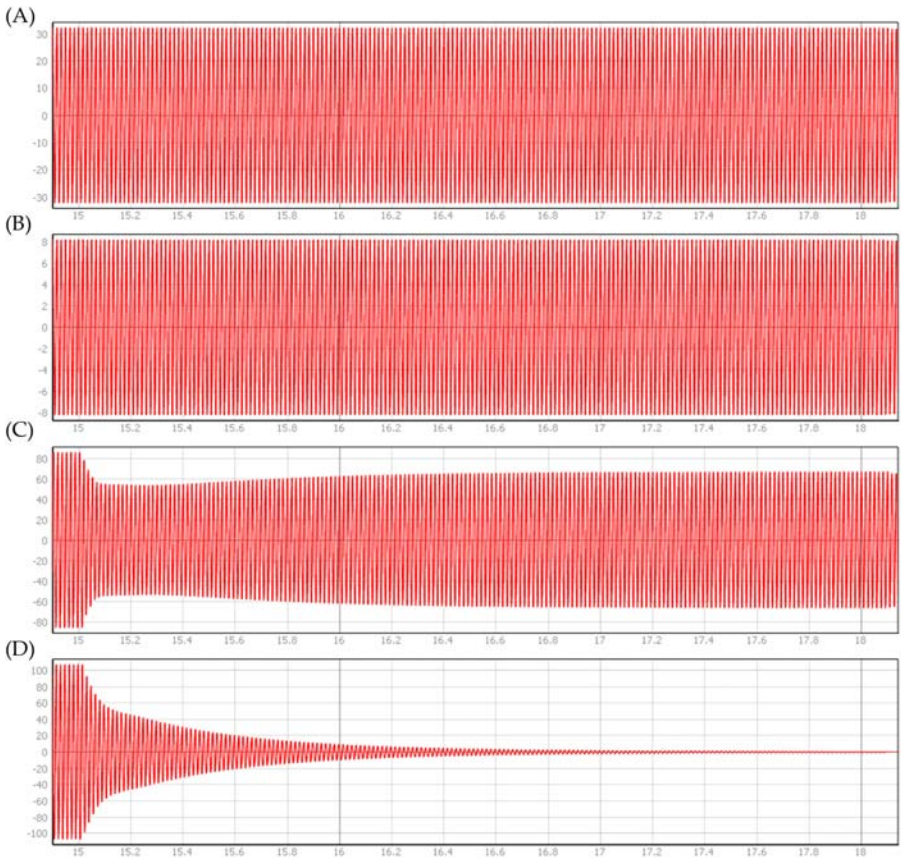


Figure 10.16 Islanding test. Currents on phase a for (A) wind power plant, (B) PV plant, (C) BESS, and (D) utility grid.

10.4 Controllerhardware-in-the-loop testing for battery energy storage systems

In this example, a battery energy storage system connected to the utility grid though an inverter is simulated in real time. The objective is to validate the controller under different operation conditions, including grid following and grid forming modes. Different from the previous examples in this chapter, where both plants and controllers were implemented within the model and run in the real-time simulator, now a controller hardware-in-the-loop (C-HIL) setup is considered.

This means that the real controller (i.e., the device under test (DUT)) is part of the loop, being interfaced with the real-time simulator by means of appropriate signal conditioning of the analog and digital inputs/outputs. Therefore it is possible to test and validate not only the control logic but also the real controller with its own hardware, software, and firmware. Fig. 10.17 shows an example of a generic controller hardware-in-the-loop testing setup. As mentioned in Chapter 7, C-HIL testing is recognized as an effective approach for testing controllers, combining high fidelity with flexibility, safety, and reduced cost.

The simplified schematic of the plant to be simulated in real time is presented in Fig. 10.18. A battery is connected to the utility grid by means of a smart grid converter. A constant impedance load is connected in parallel to the point of common coupling, and a contactor is included to enable testing the external controller in both grid-connected and islanding conditions.

The battery is implemented according to Chapter 4.3.3, with a simplified model comprising a controlled voltage source with a series resistance ($R_{\text{int}} = 0.02\Omega$), neglecting the higher-order effects. The battery type is lithium-ion, and the parameters are given in Table 10.4. The initial SOC is set to 50%.

The load is modeled as a constant resistive load of 20 kW (at a nominal voltage of 400 V), connected to the point of common coupling in a Y connection. The grid model consists of a three-phase voltage source in series with an inductive-resistive impedance, with parameters described in Table 10.5. The passive elements are modeled as presented in Chapter 4.2.

The smart grid converter is detailed in sequence.

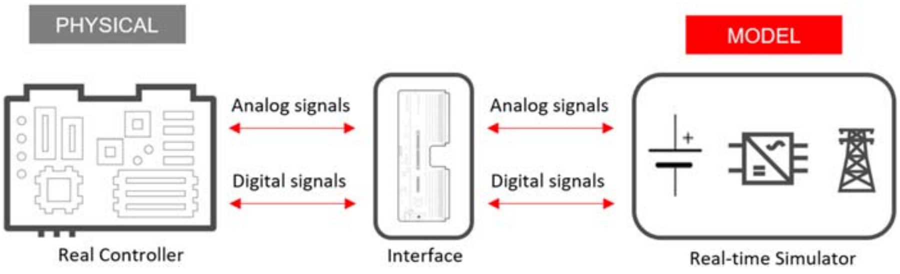


Figure 10.17 Controller hardware-in-the-loop testing setup.

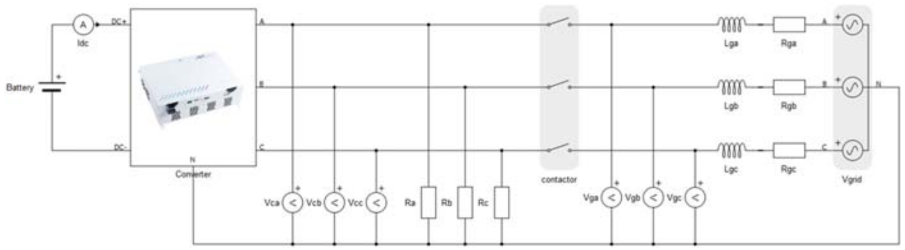


Figure 10.18 Simplified schematic of the model.

Table 10.4 Parameters of the simplified battery model.

Parameter	Value	Parameter	Value
V_{rated}	800 V	Q_{rated}	100 Ah
V_{full}	116%	Q_{nom}	93.5%
V_{exp}	103%	Q_{exp}	85%

Table 10.5 Three-phase grid parameters.

Parameter	Value
Nominal line voltage	400V
Nominal frequency	50Hz
Resistance per phase	0.14546Ω
Inductance per phase	0.00138H

10.4.1 Smart grid converter

The converter topology is shown in Fig. 10.19 based on a four-leg three-level T-type voltage source converter with an LC filter [2]. The rated parameters of the smart grid converter are presented in Table 10.6.

Different from typical three-phase four-wire converters, the combination of a DC-link neutral clamp and a fourth-phase leg allows to mitigate the DC-link currents and reduce capacitor voltage ripple. That is possible once the neutral current does not flow into the middle point of the DC bus and can be controlled by the fourth leg to compensate the neutral current or to provide a path for unbalanced components.

Similar to what was shown in Example 10.2, the objective here is to validate the inverter control, for which the high-frequency dynamics is important. Therefore the T-type converters are modeled as ideal switches, taking into account the switching behavior as described in Chapter 3.2.3. The topology for the three-phase T-type converter is illustrated in Fig. 10.20. The same reasoning is valid for the single-phase T-type.

Given that a controller hardware-in-the-loop approach is in place, the converter switches are driven by gate-drive signals from the external controller, which are interfaced to the digital inputs of the simulator. The controller calculates these signals based on the variables measured in the model, which are fed back to it through the analog outputs of the simulator.

Details about the C-HIL setup are presented in the next section.

10.4.2 Controller-hardware-in-the-loop setup

In this section, details about the C-HIL setup in this example are presented, including details about the controller and the signal interface.

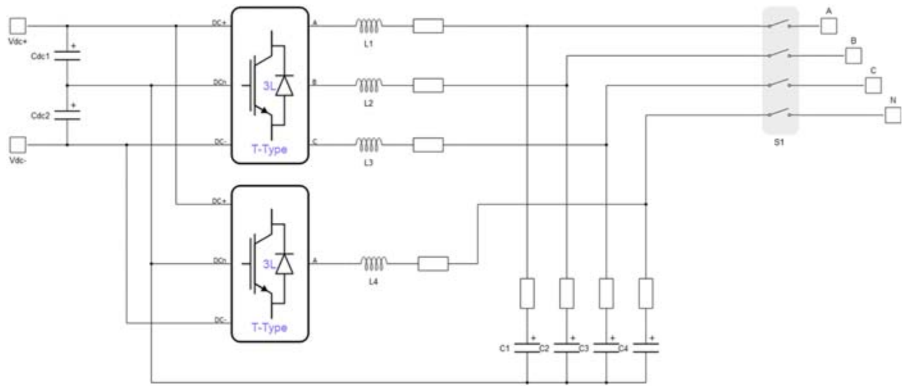


Figure 10.19 Four-leg three-level T-type voltage source converter.

Table 10.6 Parameters of the smart grid converter.

Parameter	**Value**
Nominal line voltage	400V
Nominal line frequency	50Hz
Nominal 3 ph power	34.5kW
Switching frequency	20 kHz

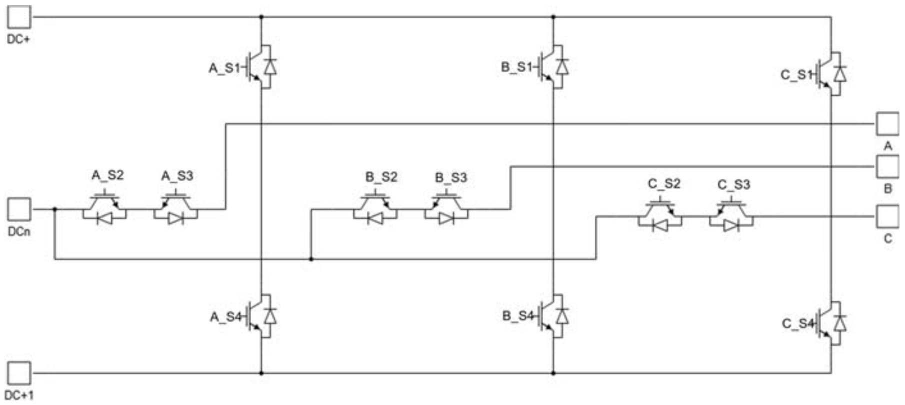


Figure 10.20 Topology of the three-phase T-type converter.

10.4.2.1 Controller

The controller used in this example is the DSP-based AIT General Purpose Inverter Controller kit, developed for rapid control prototyping in power electronics applications [3]. The advantage of using this controller is that it is easily deployable for hardware-in-the-loop simulations. As key features, it includes support for multiple converter topologies, seamless transition between grid forming and off-grid modes, simplified integration, widget-based user interface, and reference designs and examples.

10.4.2.2 Signal interface

As mentioned before, in the C-HIL framework the real external controller is connected to the model running on the real-time simulator. To properly establish this setup, all interface points between the model and the controller must be identified and mapped to the analog and digital inputs/outputs (IOs) of the real-time simulator. These signals are then routed to the IOs of the controller via an interface board, which conditions the signals to ensure compatibility of the voltage and current levels.

For this example, the digital inputs of the simulation (digital outputs of the controller) are the gate-drive signals for all converter switches (Fig. 10.20), the command signals of the smart grid contactors (Fig. 10.19), and the command signals of the grid contactor (Fig. 10.18). For these signals the interface board ensures compliance with the required input voltages of the real-time simulator.

Table 10.7 shows the voltage and current measurements fed back to the controller using the analog outputs of the real-time simulator. It is important to note that the outputs of the real-time simulator have a rated voltage range, which in this case

Table 10.7 Analog output specifications.

Signal	**Description**
V_{ca}	Phase A voltage at the output of the converter
V_{cb}	Phase B voltage at the output of the converter
V_{cc}	Phase C voltage at the output of the converter
V_{ga}	Phase A voltage at the grid side
V_{gb}	Phase B voltage at the grid side
V_{gc}	Phase C voltage at the grid side
V_{dc+}	Upper DC voltage of the DC link
V_{dc-}	Lower DC voltage of the DC link
I_{dc}	DC side current
I_{L1}	L1 filter inductor current
I_{L2}	L2 filter inductor current
I_{L3}	L3 filter inductor current
I_{L4}	L4 filter inductor current
I_{Lga}	Phase A current on the grid side
I_{Lgb}	Phase B current on the grid side
I_{Lgc}	Phase C current on the grid side

is [− 10 V, + 10 V]. Therefore the measurements in the simulation must be scaled to represent the variables within this range and with suitable resolution.

10.4.2.3 Real-time simulation results

The grid-connected battery energy storage system shown in Fig. 10.17 is simulated in real time using the same platform described in the previous examples in this chapter, a hybrid architecture containing both CPUs, and a multicore FPGA with a clock frequency of 280 MHz.

Similar to Example 10.3, to take advantage of the multicore FPGA solver and the parallel computing the model is divided into four cores: The DC side, including the battery, is simulated in core 1; the three-phase T-type converter and the respective filter inductors are simulated in core 2, the fourth-leg T-type converter and the respective filter inductor are simulated in core 3; and the AC side, from the filter capacitor to the grid, including the contactors, is simulated in core 4. The subcircuits are partitioned using the equivalent power source method, as presented in Chapter 6.5.2.

The simulation time step in this example is set to 500 ns. In the worst case the time slot utilization is around 35% for core 1, 60% for core 2, 43% for core 3, and 96% for core 4, which ensures stable real-time operation.

10.4.2.4 Overall loop-back latency

In terms of controller hardware-in-the-loop simulation for power electronics applications, one very strict requirement to achieve high-fidelity results is to have a simulator with ultralow latency computation, which is possible thanks to the FPGA technology [4].

To evaluate the overall latency, the most relevant measure is the delay between the change in a digital input signal (e.g., the GDS from an external controller) and its effects in the model running in real time (e.g., the voltage measured in the analog outputs of the simulator).

To recall, this overall loop-back latency mainly consists of the following parts:

1. Sampling delay: A change in the digital input signals can occur at any time between two consecutive simulation time steps. Assuming that the digital inputs are sampled exactly at the beginning of the simulation cycle the sampling delay can be anywhere between 0 and the simulation time step, depending on when the digital input event occurs relative to the simulation cycle boundaries. For instance a larger sampling delay happens when a change on the digital inputs occurs closer to the beginning of a simulation cycle, while lower latency is observed when the change occurs closer to the end. In general the sampling delay is a function of the simulation time step. Additional oversampling strategies can be implemented to reduce the sampling error and increase effective resolution, but they are not covered in the latency measurements in this section.
2. Computational time: Time needed to calculate the model response. The computational time is typically an integer multiple of the simulation step, ranging from 1 to several simulation steps depending on the solver configuration.

3. Hardware output stage latency: Time needed to propagate sampled and/or calculated values to the outputs of the simulator. This latency is device-specific and independent of the selected simulation step.

In the sequence, experimental results are shown, considering the loop-back latency measured from the event on a digital input to the response on the analog output. Specifically, according to Fig. 10.20 the digital input considered is the GDS of the switch A_S1, while the analog output considered is the one that represents the phase-to-neutral voltage on phase A of the respective T-type converter.

The experimental results are shown in Fig. 10.21, for which the persistence mode of the oscilloscope is used to superimpose multiple captures during the simulation runtime. From this figure, it is possible to see that the overall loop-back latency is composed of a fixed delay of approximately 925 ns (mainly consisting of the computational and hardware output stage latency) plus a variable sampling delay, ranging from 0 to 500 ns (one simulation step).

10.4.2.5 Controller-hardware-in-the-loop simulation results

In this section, the system is simulated considering different modes of the battery energy system operation to validate the external controller and the C-HIL setup. The first two scenarios cover the grid following operating mode, where the converter is connected to the utility grid and the controller regulates the active and reactive power. The third scenario covers the transition from the grid following to grid forming mode, where the battery energy storage system is disconnected from the grid, supplying the load in islanding operation.

10.4.2.6 Case 1

For this first scenario the objective is to verify the transient responses against a variation in the active power reference. Fig. 10.22 shows the converter output currents

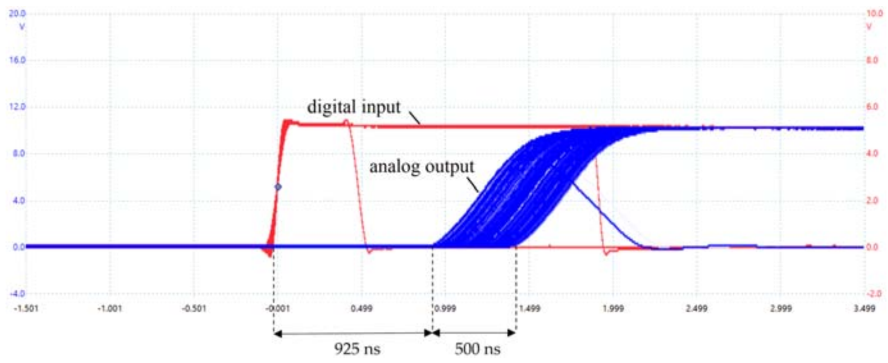


Figure 10.21 Experimental results for latency measurements: digital input and analog output of the real-time simulator.

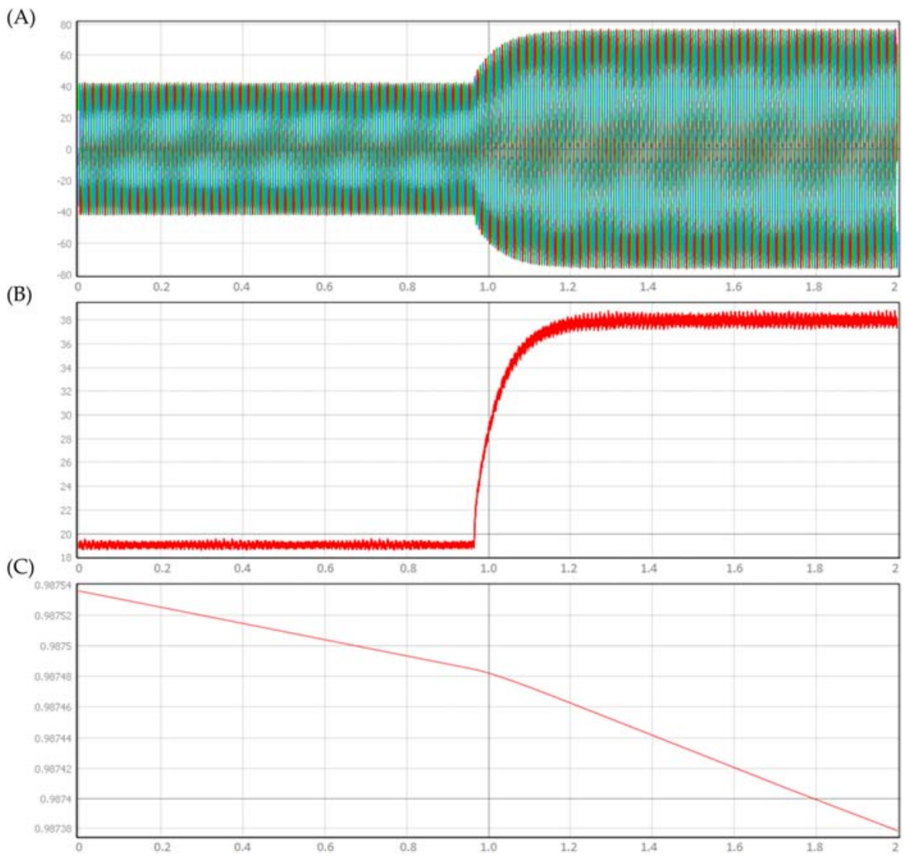


Figure 10.22 Transient response of the battery energy storage system under an increase in the active power reference: (A) three-phase inverter output currents, (B) DC current from the battery, and (C) battery state of charge (horizontal scale = 200 ms/div).

(currents through the filter inductors), the DC current from the battery, and the battery state of charge.

Initially the active power reference of the battery inverter is set at 0.5 p.u. (injecting power intro the grid), while the reactive power is set to zero. Then the active power reference is increased to 1 p.u, which leads to an increase in the amplitude of the grid currents as well as an increase in the current drained from the battery and therefore a faster discharge of the battery.

10.4.2.7 Case 2

In this second scenario the objective is to verify the transient responses of the battery energy storage system when switching from discharging to charging mode. The battery energy storage system initially operates injecting 0.5 p.u. of active power into the grid (reactive power reference set to zero). The results are presented in Fig. 10.23. Like in the previous case, one can see a constant current drained from the battery, around 20 A, and the decrease of the state of charge following a certain decay rate.

Then the active power reference is suddenly changed from 0.5 p.u. to -0.5 p.u., which should result in the converter charging the battery at half of the rated power. Indeed, as can be seen in Fig. 10.7, the inverter output currents have their phase changed, the DC current changes its direction, and the SOC of the battery starts to increase.

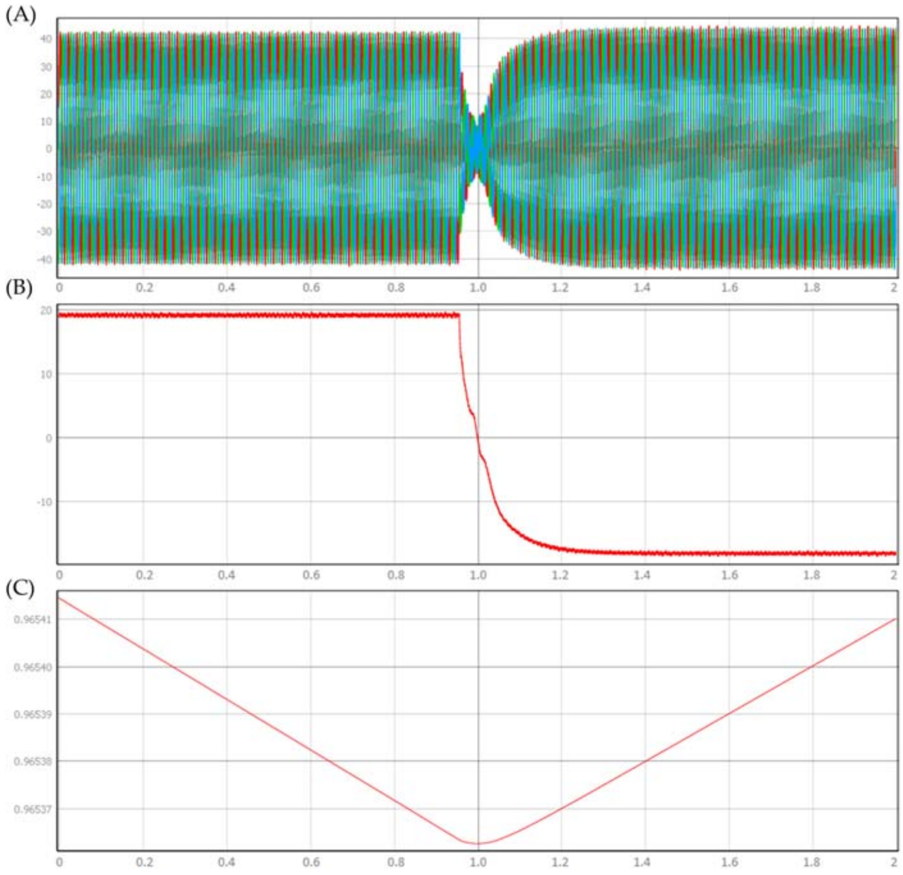


Figure 10.23 Transient response of the battery energy storage system discharging to charging mode: (A) three-phase inverter output currents; (B) DC current from the battery; (C) battery state of charge (horizontal scale = 200 ms/div).

As mentioned before, the objective is to validate the inverter-based battery energy storage controller. Therefore it is also important to have real-time simulation results detailed enough to allow analysis of inverter switching dynamics. For this purpose, considering the steady-state operation before the transient, Fig. 10.24 shows the inverter output current through the filter inductors as well as DC side current. The results are now zoomed in with a horizontal scale of 100 μs/div to show how the fast-switching dynamics introduce fluctuations in the current waveforms.

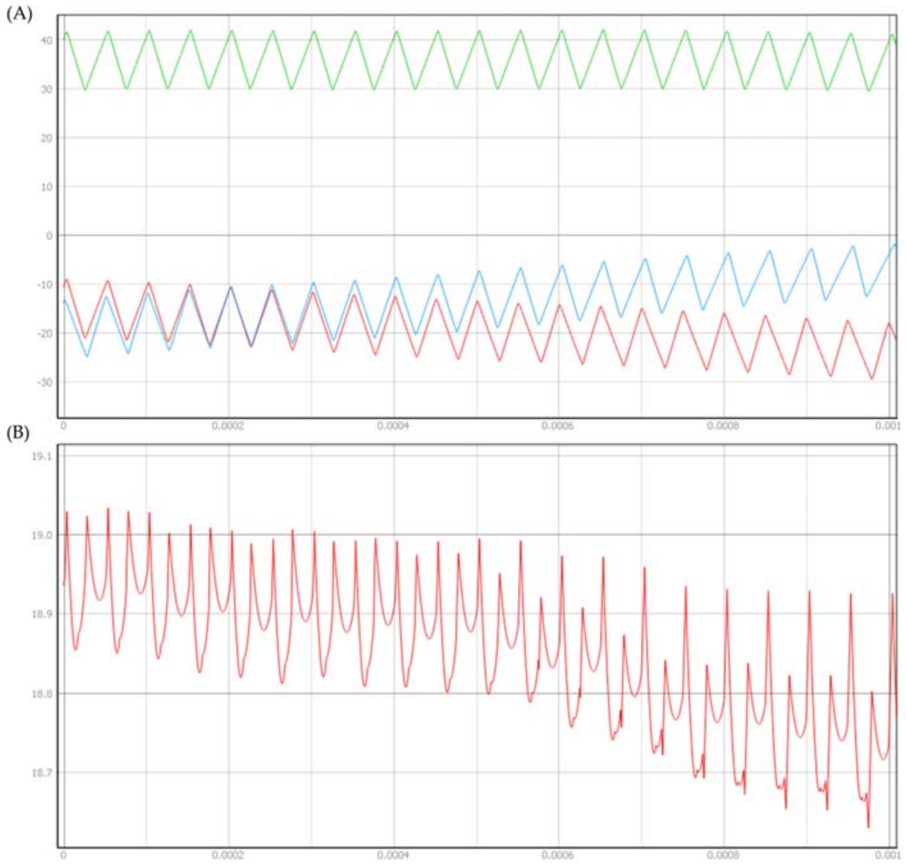


Figure 10.24 (A) Inverter output currents [A] and the (B) DC side current [A] (horizontal scale = 100 μs/div).

10.4.2.8 Case 3

In this third scenario, the objective is to verify the battery energy storage system responses in the transition from the grid following to grid forming mode (off-grid). The results are presented in Fig. 10.25.

As the initial condition the BESS operates connected to the grid, with the active power reference set to -0.5 p.u., charging the battery. In Fig. 10.25B and C, that can be confirmed by the negative DC battery current and the increasing state of charge. The grid contactor is opened, and the system is disconnected from the grid. The BESS automatically switches to the grid forming mode (voltage droop and frequency droop functionalities), providing uninterrupted power supply to the load. The change in active power and frequency (measured from the load side) can be observed in Fig. 10.26.

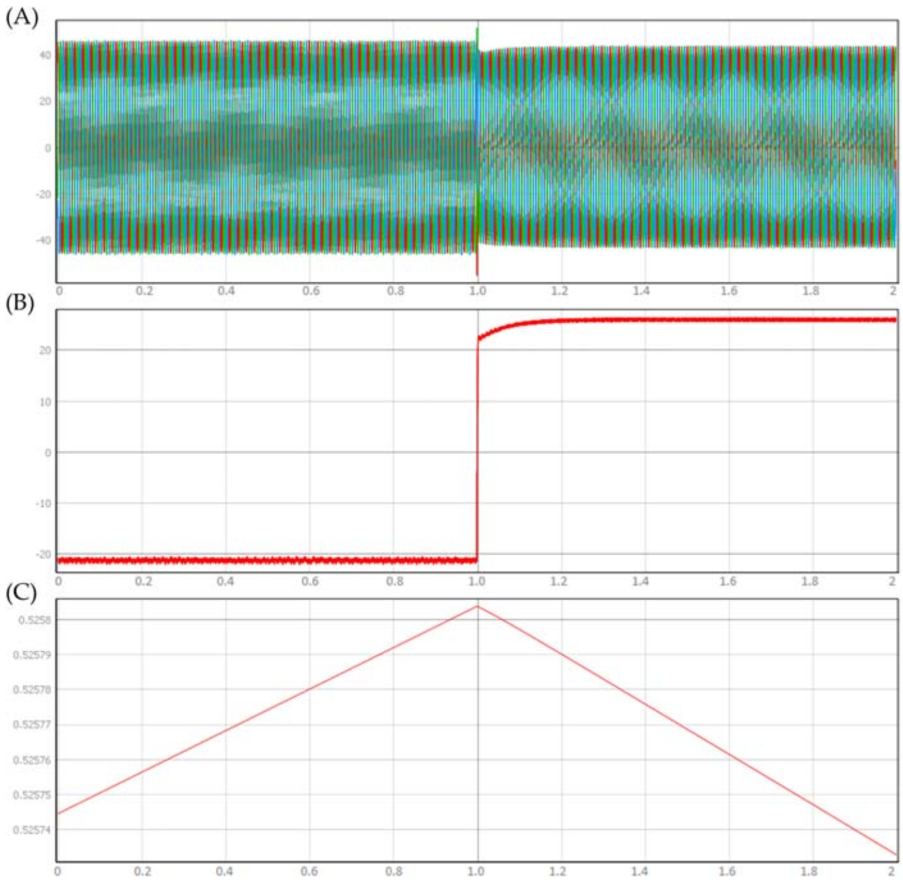


Figure 10.25 Transient response of the battery energy storage system in the transition from the grid following to grid forming mode: (A) three-phase inverter output currents, (B) DC current from the battery, and (C) battery state of charge (horizontal scale = 200 ms/div).

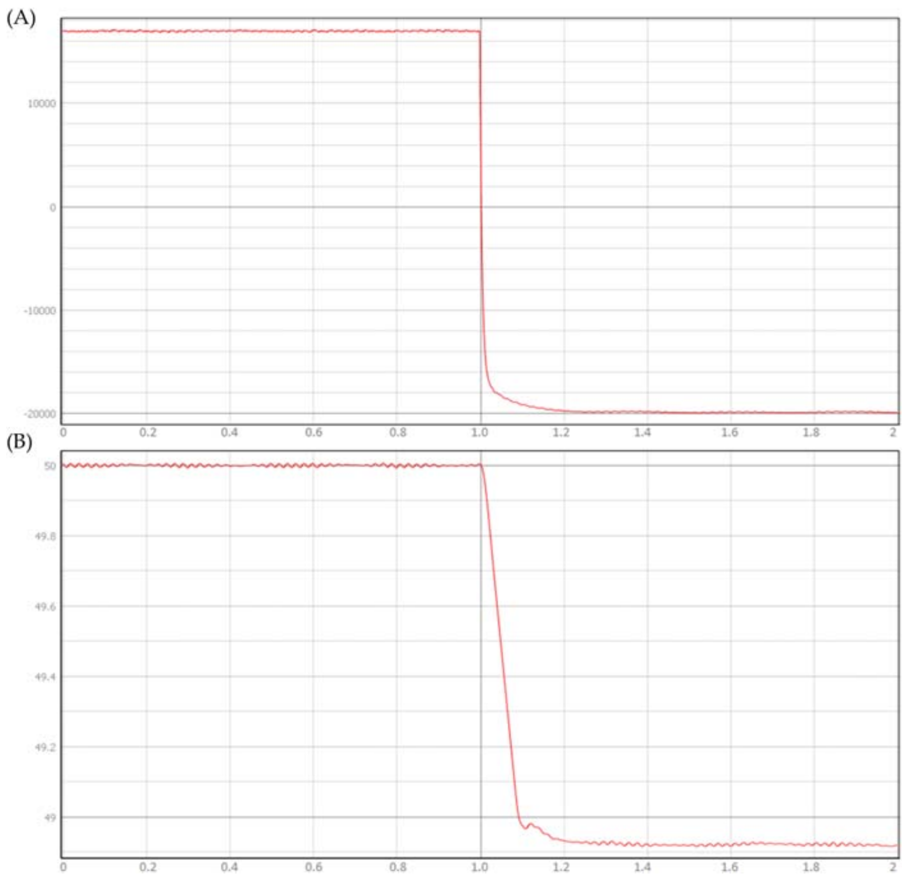


Figure 10.26 Transient response of the battery energy storage system to a disconnection from the grid: (A) active power [W] and (B) frequency [Hz] (horizontal scale = 200 ms/div).

References

[1] P.C. Krause, O. Wasynczuk, S.D. Sudhoff, Analysis of Electric Machinery, IEEE Press, 2002.

[2] Miletic, Z., Tremmel, W., Bründlinger, R., & Stöckl, J. Analysis of the RMS current stress on the DC link capacitors of the four phase 3-level T-type voltage source converter, in: Proceedings of the Twenty-second European Conference on Power Electronics and Applications (EPE'20 ECCE Europe), 2020.

[3] Austrian Institute of Technology. AIT Smart Grid Converter, 2022. https://www.ait.ac.at/en/solutions/power-system-technologies-development-validation/power-electronics-solutions.

[4] D. Majstorovic, I. Celanovic, Ultralow-latency hardware-in-the-loop platform for rapid validation of power electronics designs, IEEE Transactions on Industrial Electronics 58 (10) (2011) 4708−4716.

Advances and trends in power electronics real-time simulation

11

11.1 Introduction

The real-time simulation of power electronics faces challenges in finding the best tradeoff between the simulation speed and the modeling accuracy. The requirement of the fast simulation speed is driven by the high switching frequency of the power semiconductors, and it is the key to successfully implementing the simulation in real time. Meanwhile the required modeling accuracy depends on the device under test and the responses of interest. The closer the simulation is to the physical system, the more trustworthy the real-time simulation results will be, and the users can achieve a higher technology readiness level by using real-time simulation-based testing.

In recent years, power electronics real-time simulation technologies have matured and many real-time simulators are available off-the-shelf. However, the development of power electronics technology brought both challenges and opportunities for real-time simulation. Power electronic converters are evolving toward higher switching frequencies, lower switching losses, high voltage withstand, high operating temperature, and higher reliability. In the meantime, the control algorithms are becoming increasingly complex, taking into account advanced techniques regarding thermal management, prognosis and health management, stress-reduction strategies, fault-tolerant capability, and so on. The real-time simulation should also follow those trends, providing a suitable, safe, and high-fidelity environment for model-based design and validation of such advanced controllers. In this chapter, we focus on the recent advances and trends developing in response to the evolution of power electronic converters and their controllers. We first discuss the challenges induced by the integration of the wide-bandgap devices in the real-time simulation, and discuss the possible solutions to embrace this revolution in power electronics. Then, we take a look at the possible combinations of artificial intelligence and real-time simulation to improve the tradeoff between the modeling accuracy and simulation speed. Moreover, we explore the prognostics and health management implemented and tested by real-time simulation. At last, we will focus on the derivatives of real-time simulation, that is, the faster-than-real-time simulation and the slower-than-real-time simulation, and their applications.

11.2 Enabling the real-time simulation of wide-bandgap device-based converters

In the former chapters, regarding the fully controlled power semiconductors, we focused on Si-based devices. With the nanosecond-level simulation time step, we can simulate

Real-Time Simulation Technology for Modern Power Electronics. DOI: https://doi.org/10.1016/B978-0-323-99541-2.00006-5

the network behaviors as well as some device-level transient behaviors. Nowadays, wide-bandgap (WBG) devices, like the silicon carbide (SiC)-based and gallium nitride (GaN)-based power semiconductors, have shown great potential and advantages in terms of high efficiency and high operating temperature. Due to the high switching frequencies, the real-time simulation of WBG device-based power converters will pose even stricter time constraints than the conventional ones. Therefore in this section, we will discuss the challenges imposed by WBG devices.

11.2.1 Brief introduction of wide bandgap devices

The superior performance of the WBG devices is determined by the characteristics of their materials, that is, SiC and GaN. A radar plot is given in Fig. 11.1 to show the comparisons between Si, SiC, and GaN in terms of the energy gap, breakdown electric field, electron mobility, electron saturation velocity, thermal conductivity, and melting point.

The energy gap, also called the band gap, is one of the key properties of semiconductors. SiC and GaN have an energy gap almost three times larger than that of Si. The high energy gap will induce a higher breakdown electric field, and the power semiconductor will have a higher breakdown voltage. Therefore the doping levels can be increased, and the drift region can be thinner, leading to a reduced

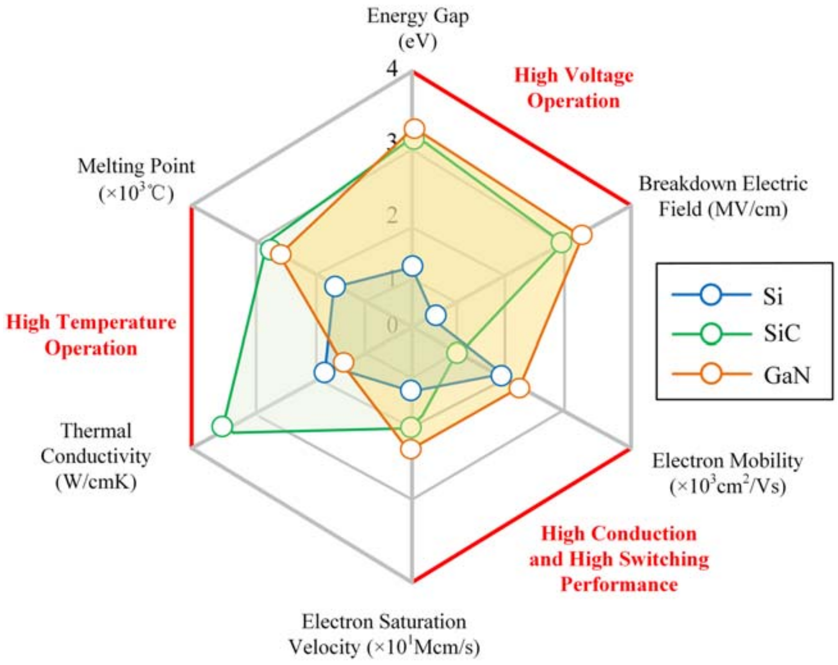


Figure 11.1 Radar chart of the key material properties of Si, 4H-SiC, and GaN at 25°C and atmospheric pressure [1].

onstate resistance and a faster switching speed. The wide bandgap allows also higher operating temperatures since the electrons need more energy to jump between bands. Based on Fig. 11.1, it can also be observed that the thermal conductivity of SiC is larger than that of Si, but the thermal conductivity of GaN is lower than that of SiC. The high thermal conductivity enables better heat dissipation from the junction to the outside and thus allows a higher power density. The higher values of electron saturation velocity and electron mobility of WBG materials lead to faster switching and improved conduction properties. The higher electron saturation velocity leads also to faster removal of the charge stored in the depletion region of a diode, thus weakening the reverse-recovery effects [1].

Based on the material properties of SiC and GaN, it is inherent to consider that WBG-based power semiconductors have a higher voltage breakdown voltage, lower onstate resistance, a higher switching speed, a higher current density, and a higher temperature than Si-based ones. Nevertheless, among the WBG devices, the SiC devices are superior in current density and operating temperature, while the GaN devices are superior in breakdown voltage, onstate conductance, and switching speed.

Even though GaN features a superior theoretical performance with respect to SiC, both WBG technologies are currently competing in the market. The maturity level of GaN devices and processes is relatively lower. While GaN is already in the leading place in low-voltage applications (i.e., up to 200V), SiC dominates the high-voltage market (i.e., $\geq$ 1200V), and both are competing in the 600/650V class [1].

11.2.2 Challenges of real-time simulation in embracing the wide bandgap devices

11.2.2.1 Simulation speed

Due to the increased switching frequency of the WBG device, the dynamics in the power converter will be faster. According to the relationship between the real-time simulation time step and the switching frequency, to implement an accurate simulation of WBG-based converters, the required simulation time step may be much shorter than the one required for the simulation of Si-based converters. For example, a real-time simulation with a 200ns time step cannot describe a power converter operating at 1MHz due to insufficient simulation resolution (i.e., the discrete model is computed only five times within a switching cycle). Therefore the real-time simulation of WBG device-based power electronic converters should have a faster simulation speed. If it is hard to further reduce real time simulation time step, another way to improve simulation accuracy for higher switching frequencies is by employing oversampling strategies, which allows to improve the effective PWM resolution without necessarily having to reduce the simulation time step.

11.2.2.2 Modeling accuracy

The increased switching frequency rises the importance of improving the modeling accuracy. For instance the distributed parameters in the converter and the devices can no longer be neglected since the increased frequency makes them play a more important role in the circuit behaviors. The real-time simulation usually uses the lumped parameters, which is very convenient to use when the switching frequency is not high. However, for the passive components, the lines, and the switching devices, there are always distributed parameters, and their impedance will become larger with the higher operating frequency. For example the parasitic inductances and capacitances of the inductor, capacitor, and switches may need to be considered in modeling the WBG power converters to improve simulation accuracy.

11.2.2.3 Electromagnetic interference

The increase in switching frequency will lead to a bigger change rate of the output current, which makes electromagnetic interference (EMI) an important factor that influences the circuit behaviors. So far, very little attention has been drawn to simulating the EMI together with the electromagnetic transient behaviors of power electronics. However, with the spread of the WBG device, there will be an inevitable need in simulating the EMI effects in real-time simulation.

11.2.2.4 Multidomain simulation capacity

The WBG devices can operate under a higher temperature than the Si devices and they are subject to a wide operating range of temperatures. Therefore the electrothermal behaviors should be simulated to fully consider the electrothermal coupling effects, thus improving the simulation accuracy. Besides the overheating of the devices will reduce the lifespan and may induce failure. Thermal management and reliability are thus the tough challenges in applying WBG devices. To evaluate or implement the thermal management and prognosis and health management (PHM), the real-time simulators should be capable of performing multidomain simulations. Not only the electrical-thermal behaviors but also the mechanics coupled with materials may have to be considered for the comprehensive design and testing of the WBG-based power converters.

11.2.3 Possible ways to embrace the wide bandgap devices in real-time simulation

Based on the previous analysis, to embrace the characteristics of WBG devices in real-time simulations, the speed and the modeling accuracy should be improved, while the support of EMI simulations and multidomain simulation should also be considered.

The modeling accuracy can be improved by applying more accurate models that encompass parasitic parameters and nonlinear characteristics. For example, we can use more dedicated models to describe the circuit components and implement the device-level real-time simulation. However, more complex models will lead to higher computational burden.

The development of real-time simulation has fallen behind the application of WBG devices. In the state-of-art, it is very hard to develop a power electronics real time model with a simulation time step less than 100 ns, which means that the simulation of a power converter operating higher than 100kHz can be troublesome. As presented in Chapter 5, the computational burden comes mostly from the solution of $\mathbf{Ax} = \mathbf{b}$. The most efficient way is to avoid the inversion operation of A. One way is to use the ADC switch model to form a matrix A with constant entries. In this method the oscillation and virtual losses should however be compensated using special techniques. The other way is to use the precomputing method. This is a very fast way, but it is weak in simulating large topologies.

FPGA is an ideal computing unit for implementing low-latency simulations since parallelism can be fully exploited. Therefore to speed up the simulation, the network of the converter could also be modeled in parallel, as discussed in Chapter 6. Nevertheless, accuracy and stability issues that may arise because of parallelism should be carefully dealt with.

The device-level simulation can improve the modeling accuracy while providing the transient behaviors beneficial to the EMI simulation and the thermal simulation. However, the transient switch model increases the modeling complexity. In such a case, the two-level simulation framework presented in Chapters 3 and 5 can be very useful to reduce computational burden. Liang et al. [2] have implemented a real-time simulation of SiC-based energy conversion in a high-speed rail power system. By using the two-level framework, the power system network simulation speed is not affected, but the parallelism of FPGA is fully taken use of to simulate the device-level electrothermal behaviors. It is worth mentioning that the transient behaviors of the WBG devices have some differences from that of the Si devices. Therefore the transient models should be tailored to the WBG devices.

The thermal behaviors of the WBG devices can also be simulated by using the equivalent thermal network presented in Chapter 3. The WBG devices are usually used in high-power density applications where the compact mounting environment is involved. Moreover the WBG devices are often used in parallel to support higher currents. Therefore the thermal coupling between multiple heat sources should be considered. The multidimensional thermal models presented in Chapter 3 are favorable choices.

The electrical simulation together with the thermal simulation will constitute an electrothermal real-time simulation. Not only the multidomain behaviors but also their coupling effects can be simulated. In such a case the design and test of the thermal management of the WBG device-based converter can be effectively implemented through real-time simulation. Moreover, if the failure and remaining useful life are mapped to the electrothermal behaviors, the PHM can also be tested using real-time simulation.

11.3 Artificial intelligence-aided real-time simulation

As shown in the previous chapters, the real-time simulations are based on the numerical solution of the mathematical models of the target power converters.

More accurate models will introduce additional complexity in the numerical solver, increasing the computational burden. Although paralleled computing can relieve this tradeoff to some extent, the accuracy and speed are still hard to meet simultaneously when the scale of the topology and the number of switches are large.

In order to improve the modeling accuracy, the nonlinear behaviors may have to be included to model the behaviors of the power electronic converter as well as the power semiconductors. The solution of nonlinearities is one of the tough tasks since the iterative solving algorithm should be used, which is time-consuming. In recent years, the burst interest in artificial intelligence (AI) takes place in many fields, achieving significant methodological advances. Researchers are also trying to bring AI benefits in real-time simulation to improve the tradeoff between modeling accuracy and computing speed.

In an AI algorithm, the nonlinear characteristics can be established through a artificial neural network when the nonlinear mechanisms between input and output are not easy to describe analytically or they are very complex to solve. Meanwhile the artificial neural network has a parallel structure that is hightly compatible with the FPGA hardware, allowing to reduce the overall computing latency. Therefore it seems that there is potential for implementing artificial neural networks in real-time simulation to describe the nonlinear characteristics that are costly to solve numerically.

The neural network has already been applied to the offline simulation. Krishnamoorthy and Aayer [3] proposed a method to model the steady state and the transient of the power electronic converter based on the feed-forward neural network and the random forest algorithm, which is successfully used to simulate the Buck converter and the Boost converter. Qashqai et al. [4] proposed a power converter simulation method by using the long-short-term memory (LSTM) networks to improve the modeling accuracy of using the feed-forward neural network. Hari et al. [5] proposed a deep feed-forward neural network in simulating the GaN device's switching voltage and current. Deng et al. [6] proposed an estimation method of IGBT switching power losses based on neural networks. Based on previous studies, artificial neural networks can provide satisfying results as long as the training data are sufficient and the artificial neural network is well trained.

Regarding the applications of artificial neural networks in real-time simulation, it is an emerging topic and some explorative studies are reported. In the literature, AI has been used in improving the performance of device-level real-time simulation. It is known that the switch transient model contains discontinuities and nonlinearities, and they are simplified or piecewise linearized to reduce the complexity in the model-based device-level real-time simulation. Liang et al. [7] proposed a switching transient simulation method based on the hybrid neural network. The k-nearest neighbors (kNN) concept is adopted to classify the device operating conditions to different regions. In each region the recurrent neural network (RNN) is trained to describe the switching transient and then generates the switching waveforms in the real-time simulation with a 100 ns resolution. Similarly, Zhang et al. [8] and Li et al. [9] proposed methods to replace the switch transient model with the neural networks, as depicted in Fig. 11.2. In addition to modeling the switching

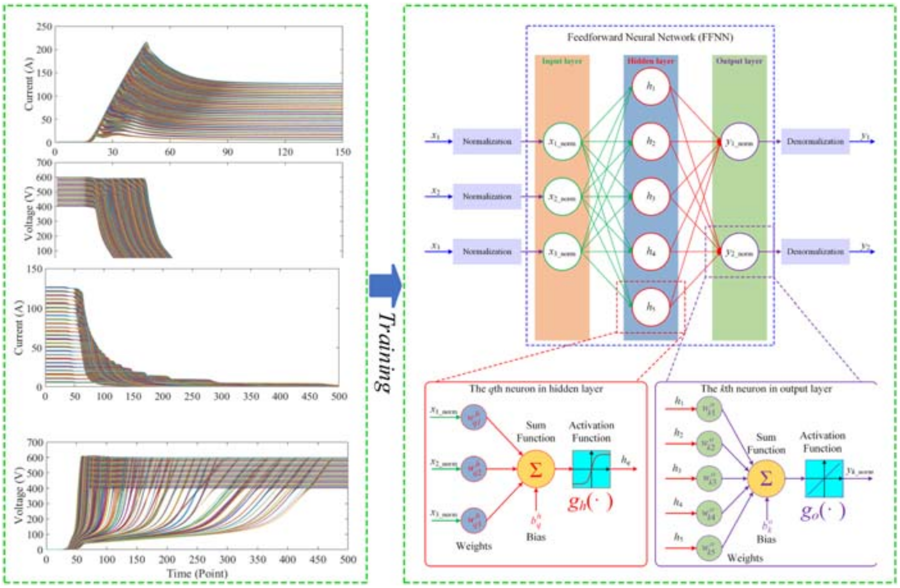


Figure 11.2 Training the feed-forward neural network by the switching transient [9].

transient, Bai et al. [10] proposed an electrothermal real-time simulation method by using the neural network to model the switching losses.

We can see that in those models, the target nonlinear characteristics are treated as the "black box", and the artificial neural network is used to approximate the complex relationship between input and output. There are still some limitations when using the artificial neural network to represent the nonlinearities. The generalization ability of the neural network can be insufficient to cover a wide operating range of power electronic devices. If a more complex artificial neural network is used to improve the accuracy of the model, such as the deep network, the computing cost will increase dramatically, which threatens the feasibility of using them in real-time simulation. Moreover, the computing time of neural networks may not be as short as we expected. The computing latency of AI in nonlinearity modeling, such as the switching transient modeling, will also vary with the network structures. Sometimes the computing latency is still too large to represent fast switching transient.

11.4 Prognostics and health management using real-time simulation

The prognostics and health management (PHM) processes use diagnostic and predictive algorithms to determine the extent of deviation or degradation of a product from its expected normal operating condition. PHM is an effective means of improving the lifetime of power electronic systems. Many solutions have been developed, but their adoption in industrial applications is still limited. While fault diagnostics is based on observed data and knowledge about the system's current operational and environmental data, PHM is based on historical data and profiles of future usage and environmental factors [11]. Sometimes the required data can be measured by sensors and directly used for detecting the potential fault or failure. However, the required data may not be easy to acquire. The inclusion of additional sensors may not be feasible, and it may not be possible to measure some of the variables directly. Therefore the model-based methods can be used to support the PHM. In this case, the real-time simulation is thus an eligible choice because it can perform the model simulation in a fast and accurate manner.

Peng et al. [12] proposed an estimation method of health indicators for the DC-DC converters by using real-time simulation. Different from the traditional real-time simulation, the parameters in the simulated model are updated through the optimization algorithm according to the sampled variables from the real converter. Actually, this kind of real-time simulation can also be regarded as a simple digital twin.

Taking the Buck converter as an example, the health indicators are monitored through the framework shown in Fig. 11.3. The Buck converter is modeled using the state-space equations. The inductor L, capacitor C, the ESR of inductor R_L, the ESR of capacitor R_C, the turn-on resistance of MOSFET R_{dson}, and the forward voltage of the diode are modeled. The RK-4 method is used to discretize the state-space equation, and the discretized state-space equations are computed in real time. Meanwhile the controller is also implemented. The model as well as the controller constitutes the digital world replica of the Buck converter.

In the physical world, a real Buck converter operates with the same parameters and the same operating point as in the digital world. The real inductor current and capacitor voltages are sampled and compared with the corresponding simulated

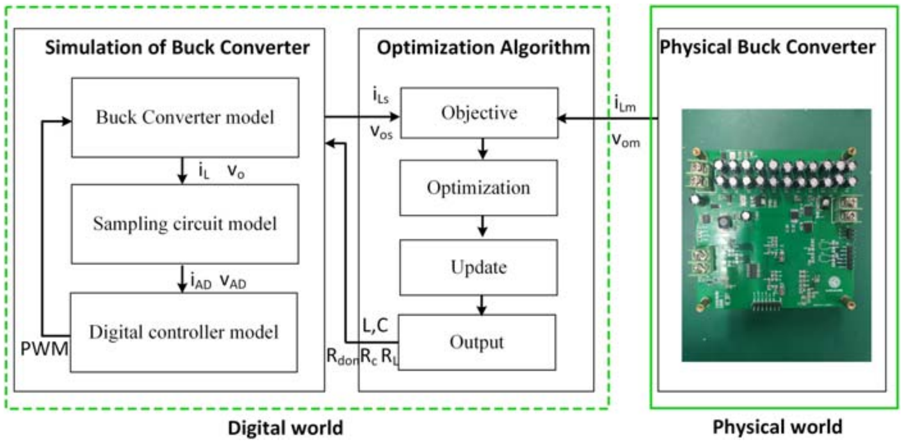


Figure 11.3 System structure of the Buck converter's health condition monitoring [12].

results. An optimization algorithm such as the particle swarm optimization (PSO) is adopted to find the optimal values of R, L, C, R_L, R_C, and R_{dson} to minimize the difference between the digital model and the physical system. By doing so, the real-time simulation model will be updated to follow the changes in the real world. Once the degradation of the switch and the passive components occurs, their parameters in the digital model will show an obvious trend deviating from the original values. Based on these indicators, the health states can be monitored in a noninvasive and calibration-free manner without any additional circuits.

In addition to the implementation of the PHM using real-time simulation, the real-time simulation can also be used to test the PHM strategies. PHM focuses on the long-term behaviors of the power electronics, such as the degradation of the power semiconductors. It is not convenient to test the PHM on the real prototype bench for such purposes. Instead, the real-time simulation can be used in the HIL environment to test PHM strategies and provide useful guidance in improving PHM effectiveness and performance. To test the PHM, the real-time model should be able to simulate the commonly used thermal-sensitive electrical parameters, damage-sensitive electrical parameters, and thermal behaviors.

11.5 Faster or slower than real-time simulation

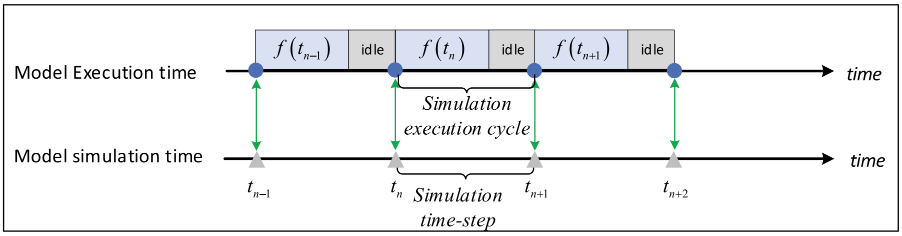
In the real-time simulation, the amount of real time needed to compute the system model $f(t)$ is shorter than the discretized time step of the simulation, as shown in Fig. 11.4A. Therefore the simulation execution cycle can be equal to the simulation time step.

If the model computation can be realized in a faster way, the simulation execution cycle can be set to be smaller than the simulation time step, as shown in Fig. 11.4B. Multiple computing steps can be implemented within the real-time period corresponding to one or less than one simulation time step. Therefore the faster-than-real-time (FTRT) simulation can be implemented, where the simulation model runs faster than the physical system. Here, we refer FTRT to those simulations where the simulation execution cycle has a fixed relationship with the simulation time step, which is different from the offline simulation with a random rate of acceleration regarding to real time.

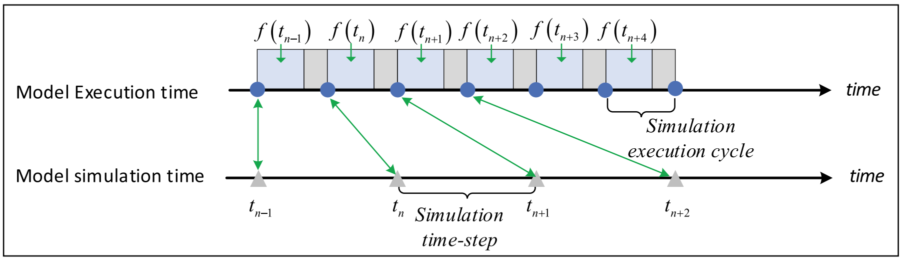
If the model computation cannot be finished within the real-time period that the simulation time step corresponds to, the model will run at a slower rate than the physical system, and the slower-than-real-time (STRT) simulation can be implemented as shown in Fig. 11.4C. In this case the simulation execution cycles will be an integral number of times larger than the simulation time step.

11.5.1 Faster-than-real-time simulation

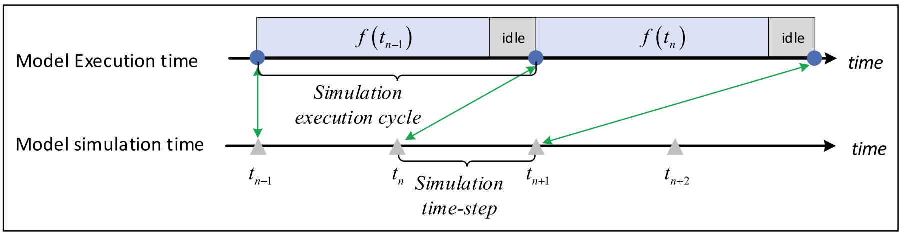
Real-time simulation enables the understanding of system operations by running the model in real-time clock and connecting it to the physical devices. However, for a



(A) Real-time simulation



(B) Faster-than-real-time simulation



(C) Slower-than-real-time simulation

Figure 11.4 Timing characteristics in real-time, faster-than-real-time, and slower-than-real-time simulations. (A) Timing characteristics in real-time simulation, (B) timing characteristics in faster-than-real-time simulation, and (C) timing characteristics in slower-than-real-time simulation.

system requiring long-term evaluation, the real-time simulation will take a long time to get a complete insight into the full-scale behavior. The FTRT simulation is more suitable for this task as it can shrink the total time to evaluate the long-term performance of the target system.

With the development of renewable energy sources, power electronics are highly integrated into the power system. Therefore the power system analysis should also focus on the behaviors of the power electronic systems. It is well-known that the power system needs effective simulation tools to predict the system behaviors and aid the decision-making in the system operation. An accurate simulation that is much faster than real-time simulation and fully considers the fast dynamics of power electronics is demanded to address the operation, design, and investigation of power system events, building emergency management plans, and effective controls for system stability and reliability.

To implement the FTRT simulation of power electronic systems, the selection of the simulation execution cycles and the simulation time step should be coordinated. On one hand the execution cycles should be reduced as much as possible. The paralleled computing technologies and optimization methods could be applied to reduce the total computing latency. On the other hand, a relatively larger simulation time step is welcomed in the context of ensuring simulation accuracy and stability. One should carefully consider the dynamics of the target systems and define the scope of the simulation, based on which the corresponding model comlexity must select. For example, for a system-level simulation more concerned with energy management, the transient switch models may be neglected. In addition, the ratio of the acceleration that the FTRT simulation is expected to achieve should also be considered in matching the simulation execution cycle and the simulation time step. With a reduced execution cycle operating with a larger time step, the FTRT simulation can be implemented to predict the power electronic system behaviors in short or even long term.

Cao et al. [13] proposed an FTRT dynamic simulation of integrated AC/DC grids on the FPGA by using a fine-grained relaxation algorithm to efficiently solve the nonlinear differential-algebraic equations of the integrated system model. 134 times faster than real time is reported for simulating the integrated large-scale AC/DC grid based on 2 IEEE 39-bus systems interconnected with a 4-terminal HVDC grid. Duan and Dinavahi [14] has proposed an adaptive time-stepping method for the FTRT simulation of the universal transmission line model and universal machine on the FPGA. The IEEE 39-bus system with ATS models is emulated on two interconnected FPGA boards in a 4 times FTRT manner. FPGA may be a good choice but not the only choice. The CPU can also be a very efficient way in FTRT simulation of large-scale power electronic systems. Wang et al. [15] implemented an FTRT simulation of modular multilevel converters using standard multicore CPU and FPGA chips. A back-to-back MMC HVDC system where an MMC system with as high as 3000 SMs is simulated in an FTRT in both CPU and FPGA. High-fidelity results are produced by both computing engines, while a larger number of SMs can be embraced in FPGA and the simulation latency can be shorter than the CPU.

11.5.2 Slower-than-real-time simulation

Power electronics usually have fast dynamic behaviors due to switching events. With the development of power semiconductors the switching frequency has increased significantly. Due to the high switching frequency characteristics, the real-time simulation time step needs to be far less than one switching period to get high discretization accuracy and reduce errors caused by the inner-step switching events [16]. Generally a time step 100 times smaller than the switching period is typically used in DC-DC simulation practice [17]. At present the minimum simulation time step of commercial real-time simulators reported so far is in the range from 100 to 200 ns, such as Typhoon HIL 4th generation devices (200 ns) [18] and Opal-RT OP5707 (145 ns) [19], which means that the accuracy can be weak to simulate the power converter operating above 100kHz. Aiming to further improve simulation accuracy for higher switching frequencies, Typhoon HIL recently introduced the concept of a programmable FPGA solver optimized for certain converter topologies, reducing the simulation step down to 50 ns for typical DC-DC topologies [20]. Moreover, oversampling strategies can be employed to increase the PWM time resolution. For instance, Typhoon HIL 4th generation devices can oversample the digital inputs at the device's clock frequency (280 MHz), applying compensation algorithms to minimize the effects of the sampling error. As a result, it is possible to achieve a PWM resolution of 10 bits for converters operating with switching frequencies around 300 kHz.

In academia, many efforts have been made to reduce the simulation time step. So far the lowest 25 ns power converter simulation time step is reported in [21] and [22] by using the predictive solver and the direct-mapped method, respectively. Although the simulation time step has been reduced by optimizing the real-time solver, the simulation speed can be still insufficient, especially when simulating the WBG-based converter operating at MHz level, which may also exceed the limitation of the real-time simulation capacity.

The STRT simulation provides an alternative way to simulate the high-switching frequency power converters. Since one of the most important uses of real-time simulation is to implement the CHIL testing, here we present an STRT simulation-based CHIL testing technique to test the controller of high-switching frequency power converters which are originally beyond the capacity of the current CHIL state-of-art. The basic idea is to slow down the execution rate of both the simulator and the controller simultaneously, with the same ratio. In the STRT simulation, the execution cycle of one simulation time step is an integer number of times larger than the simulation time step. Correspondingly the controller must be slowed down by the same ratio by changing the execution cycle of the control algorithm as well as the cycles of PWM signals. The CHIL environment can thus be established between the STRT and controllers since the control model in the controller is still running at the same "clock rate" (slower than real time clock) as the converter model in the simulator. The CHIL testing of a high-switching frequency converter can thus be achieved without worrying about the obstruction of the simulation speed.

Fig. 11.5 shows the timing performance of the normal real-time model-based CHIL test. It has strict time constraints; that is, the time in the controller and the simulator exactly correspond to the real physical time. Therefore the control model is running at the same rate as the converter model. For the controller, connecting to the real-time simulator has equivalent effects to connecting to the real converter in terms of functions and performances.

If we slow down the simulation execution cycle and the control execution cycle by the same ratio, we effectively apply the STRT simulation method. The timing performance is shown in Fig. 11.6. After slowing down the controller and simulator by a factor of two compared to the real-time case in the figure, the running rates of

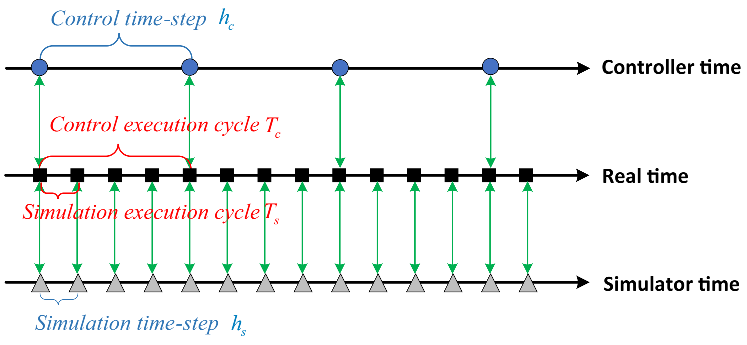

Control time-step h_c
Controller time
Control execution cycle T_c
Real time
Simulation execution cycle T_s
Simulator time
Simulation time-step h_s

Figure 11.5 Time constraint in the real-time simulation-based CHIL testing system.

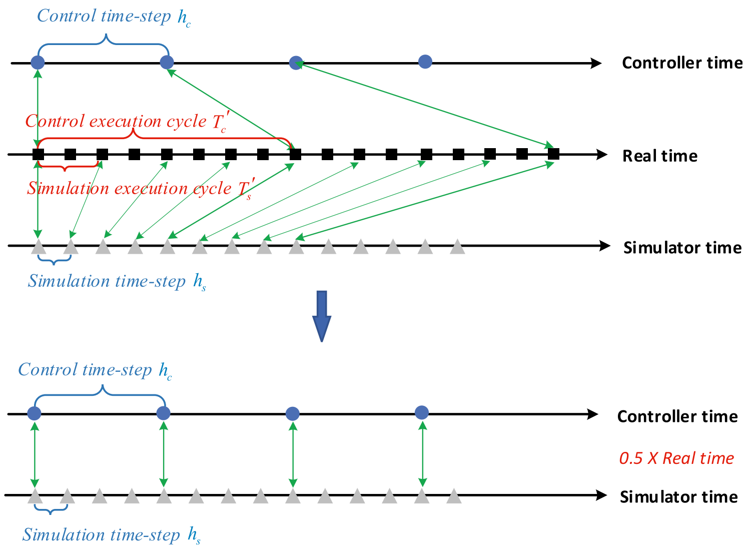


Figure 11.6 Timing relationship of STRTS-based CHIL system.

the controller and the converter model still remain in sync. The STRT-based CHIL relaxes the constraint of the model computation time and also keeps the controller and simulator running at the same rate to satisfy the condition of CHIL testing. Only some modifications in the controller's software are needed to establish the CHIL using the STRT simulation. In addition, due to the slowed-down dynamics in the STRT simulation, the effective simulator's analog output dynamics can be improved.

Here, we use a Buck converter as an example. The Buck converter is modeled and simulated in FPGA, and the Buck controller is tested in the real-time simulation-based CHIL and the STRT simulation-based CHIL, respectively. The Buck converter is operating at 10 kHz. In the real-time simulation, the control execution cycle is 100μs and the simulation execution cycle is 1μs; slowing down the simulation and the control by a ratio of 2 the control execution cycle becomes 200μs and the simulation execution cycle becomes 2μs in the STRT simulation.

The CHIL testing results are shown in Figs. 11.7 and 11.8. In Fig. 11.7 the inductor current is regulated at the steady state 4A, and the STRT simulation results are of the same shape as the real-time simulation except for the fact that the time interval is twice longer. Fig. 11.8 depicts the step response of inductor current. The STRT simulation can produce consistent results of the control transient with the real-time simulation, which means that the STRT simulation has equivalent effects to the real-time simulation in testing the control performance.

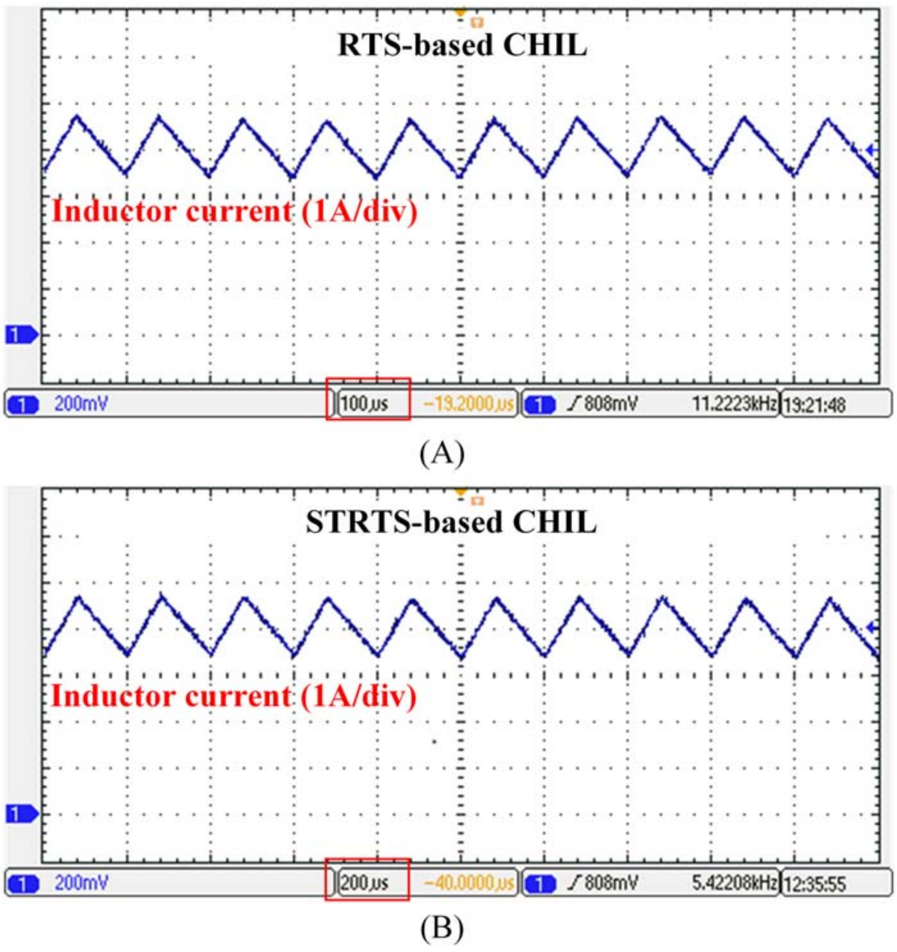
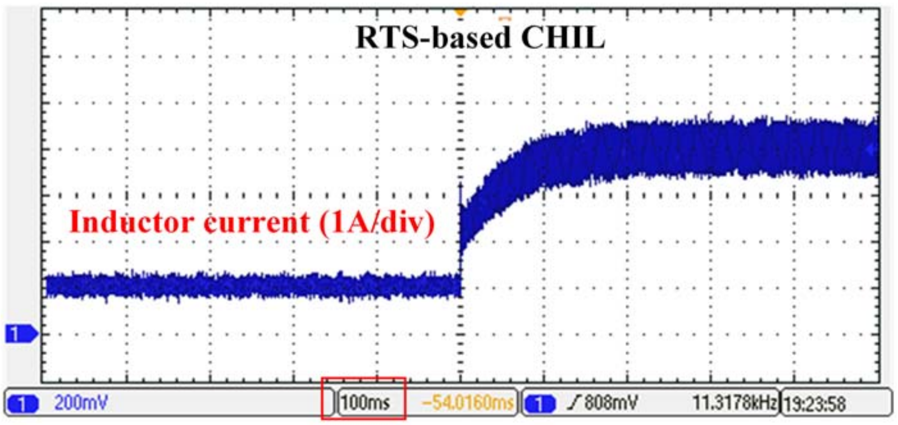
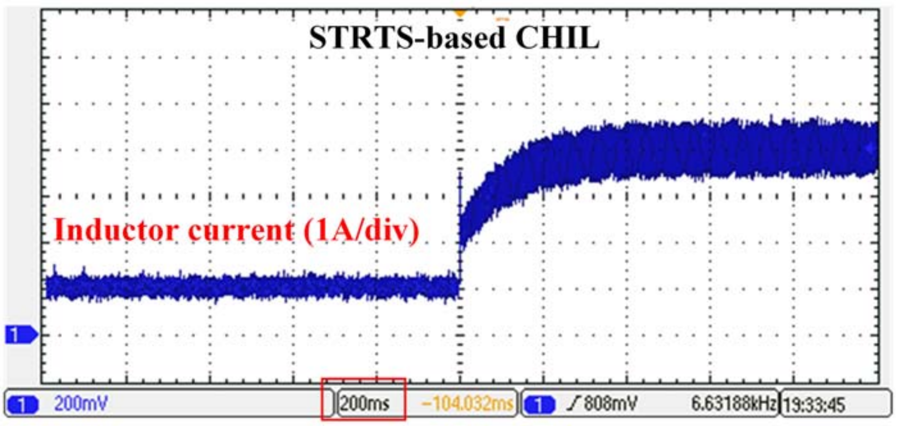


Figure 11.7 Comparisons of steady inductor current at 4A. (A) Waveforms of real-time simulation (time: 100 μs /div, amplitude: 1A/div); (B) waveforms of slower-than-real-time simulation (time: 200 μs /div, amplitude: 1A/div).



(A)



(B)

Figure 11.8 Comparisons of the step responses of inductor current between 1A and 4A. (A) Waveforms of real-time simulation (time: 100 ms /div, amplitude: 1A/div); (B) waveforms of slower-than-real-time simulation (time: 200 ms /div, amplitude: 1A/div).

References

[1] D. Cittanti, E. Vico, I.R. Bojoi, New FOM-based performance evaluation of 600/650. V SiC and GaN semiconductors for next-generation EV drives, IEEE Access vol. 10 (2022) 51693−51707. Available from: https://doi.org/10.1109/ACCESS.2022.3174777.

[2] T. Liang, Q. Liu, V.R. Dinavahi, Real-time hardware-in-the-loop emulation of high-speed rail power system with SiC-based energy conversion, IEEE Access vol. 8 (2020) 122348−122359. Available from: https://doi.org/10.1109/ACCESS.2020.3006904.

[3] Krishnamoorthy H.S., Aayer T.N. Machine learning based modeling of power electronic converters, in: Proceedings of the IEEE Energy Conversion Congress and Exposition (ECCE). IEEE, 2019, pp. 666-672.

[4] Qashqai P., Al-Haddad K., Zgheib R. Modeling power electronic converters using a method based on long-short term memory (LSTM) networks, in: Proceedings of the IECON 2020 Forty-sixth Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2020, pp. 4697-4702.

[5] Hari N., Chatterjee S., Iyer A. Gallium nitride power device modeling using deep feed forward neural networks, in: Proceedings of the First Workshop on Wide Bandgap Power Devices and Applications in Asia (WiPDA Asia). IEEE, 2018, pp. 164-168.

[6] Y. Deng, X. He, J. Zhao, et al., Application of artificial neural network for switching loss modeling in power IGBTs, Journal of Zhejiang University SCIENCE C 11 (6) (2010) 435−443.

[7] T. Liang, Z. Huang, V. Dinavahi, Adaptive real-time hybrid neural network-based device-level modeling for DC traction HIL application, IEEE Access 8 (2020) 69543−69556.

[8] S. Zhang, T. Liang, V. Dinavahi, Machine learning building blocks for real-time emulation of advanced transport power systems, IEEE Open Journal of Power Electronics 1 (2020) 488−498.

[9] Q. Li, H. Bai, E. Breaz, R. Roche and F. Gao, ANN-aided data-driven IGBT switching transient modeling approach for FPGA-based real-time simulation of power converters, IEEE Transactions on Transportation Electrification, 2022.

[10] H. Bai, C. Liu, E. Breaz, et al., Artificial neural network aided real-time simulation of electric traction system, Energy and AI 1 (2020) 100010.

[11] S.K.M. PECHT, Modeling approaches for prognostics and health management of electronics, International Journal of Performability Engineering 6 (5) (2010) 467.

[12] Y. Peng, S. Zhao, H. Wang, A digital twin based estimation method for health indicators of DC−DC converters, IEEE Transactions on Power Electronics 36 (2) (2021) 2105−2118. Available from: https://doi.org/10.1109/TPEL.2020.3009600.

[13] S. Cao, N. Lin, V. Dinavahi, Faster-than-real-time dynamic simulation of AC/DC grids on reconfigurable hardware, IEEE Transactions on Power Systems 35 (2) (2020) 1539−1548. Available from: https://doi.org/10.1109/TPWRS.2019.2944920.

[14] T. Duan, V. Dinavahi, Adaptive time-stepping universal line and machine models for real time and faster-than-real-time hardware emulation, IEEE Transactions on Industrial Electronics 67 (8) (2020) 6173−6182. Available from: https://doi.org/10.1109/TIE.2019.2935930.

[15] Can Wang, Wei Li and J. Belanger, Real-time and faster-than-real-time simulation of modular multilevel converters using standard multi-core CPU and FPGA chips, in: Proceedings of the IECON 2013 Thirty-ninth Annual Conference of the IEEE Industrial Electronics Society, 2013, pp. 5405-5411. Available from: https://doi.org/10.1109/IECON.2013.6700015.

[16] M. Matar, R. Iravani, FPGA implementation of the power electronic converter model for real-time simulation of electromagnetic transients, IEEE Transactions on Power Delivery 25 (2) (2010) 852−860. Available from: https://doi.org/10.1109/TPWRD.2009.2033603.

[17] M. Deter, Q. Ha, M. Plöger, F. Puschmann, FPGA-based real-time simulation of a DC/DC converter, ATZelektronikWorldwide 9 (2) (2014) 32−35. Available from: https://doi.org/10.1365/s38314-014-0236-8.

[18] Typhoon HIL, Inc., Typhoon-HIL404-Brochure, 2022. https://typhoon-hil.com/doc/brochures/Typhoon-HIL404-Brochure.pdf.

[19] OPAL-RT Technologies, Inc., Flagship Real-Time Digital Simulator, 2022. https://www.opal-rt.com/simulator-platform-op5707/.

[20] Typhoon HIL, Inc., DC-DC Converter Solver, 2022. https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/topics/dc_dc_converter_solver.html.

[21] C. Liu, H. Bai, S. Zhuo, X. Zhang, R. Ma, F. Gao, Real-time simulation of power electronic systems based on predictive behavior, IEEE Transactions on Industrial Electronics 67 (9) (2020) 8044−8053. Available from: https://doi.org/10.1109/TIE.2019.2941135.

[22] H. Chalangar, T. Ould-Bachir, K. Sheshyekani, J. Mahseredjian, A direct mapped method for accurate modeling and real-time simulation of high switching frequency resonant converters, IEEE Transactions on Industrial Electronics 68 (7) (2021) 6348−6357. Available from: https://doi.org/10.1109/TIE.2020.2998746.

Outlooks on power electronics real-time simulation

12

12.1 Introduction

Chapter 11 presents the recent advances and trends in the power electronics real-time simulation that are happening or could happen shortly. Since a bottleneck of the real-time simulation is the tradeoff between modeling complexity and the achievable simulation time step, the goal is always to realize accurate but computationally efficient models in real time while constantly improving the computing speed.

Synthesizing the multiphysics behaviors in the simulation will make the models even more accurate. In addition to the electrothermal models presented in previous chapters, the electromagnetic-thermal-mechanics interactions can be also considered in real-time simulation in the future. Although the physics-based model of power electronics is yet not fully feasible, it will become true after the improvements coming in both hardware and numerical algorithms. Even today, real-itme simulation using physics-based models are already successfully applied in some specific applications.

With the increase in model accuracy, real-time simulation applications will be broadened as well, covering, for instance, everything from semiconductor thermal management to electromagnetic compatibility in the system integration, from design validation to full-lifecycle maintenance. The digital twin (DT) is thus built and enabled by real-time simulation, communication, and data fusion technologies. The DT not only serves for the design and validation of power electronics but also provides useful information in system management and maintenance, which acts as the "twin" of the power electronic system.

With the development of the distributed energy system, collaborative real-time simulation will become an indispensable method in analyzing, testing, and predicting the system. Therefore the geographically distributed real-time simulation (GDRTS) will be a good choice to break the geographical constraints of the cosimulation framework. GDRTS is good for the integration and utilization of global simulation resources to constitute a worldwide real-time simulation or DT, which will innovate the ways we research, design, test, manufacture, operate, and maintain the power electronic system.

In this chapter, we provide the outlooks of power electronics real-time simulation from three aspects:

1. integrating the multiphysics power electronic model in the real-time simulation;
2. expanding real-time simulation to the full lifecycle (DT application);

3. worldwide collaborative simulation using GDRTS

In this chapter the future possible developments of power electronics real-time simulation are discussed. This chapter will serve as a modest spur to induce someone to come forward with his or her valuable contributions to the power electronics real-time simulation.

12.2 Integrating the multiphysics power electronics model in the real-time simulation

In Chapter 3, we have already presented the static model and the transient models of the power semiconductors, such as IGBT and the diode. The static models include the average model, the ideal switch model, the binary resistor model, and so on. Regarding the transient models, we have presented the nonlinear equivalent circuit model, the piecewise linear transient model, the curve-fitting model, and the two-level quasi-transient model, which should be treated as behavioral transient models since only the external voltage—current relationship rather than the semiconductors' physical mechanism are accounted for. The behavioral model is simplified and used for the fast-computing purpose. The minimum time step can be as low as tens of nanoseconds for the real-time simulation of power electronic converters. Unlike the behavioral model, the physical model uses semiconductor physics equations to describe carrier distributions and electrical behavior, which can be more accurate to reflect the intrinsic mechanism inside the device. However, the physical model usually involves partial differential equations that are highly computationally intensive. With such a model, it is tough to meet the time constraint of the real-time simulation. Besides the physics-based model should consider the physical structure of the devices, as shown in Fig. 12.1, which involves the interactions of multiple physical domains in terms of electricity, magnetism, heat, and mechanics [1]. It is challenging to accurately simulate the dynamic behaviors in an electric-magnetic-thermal-mechanical coupled manner. Thus the semiconductor model will not only focus on the integration of a physics-based model but also target the implementation of a multidomain model in real-time simulation. In addition to the power semiconductors, passive components such as inductors, capacitors, and

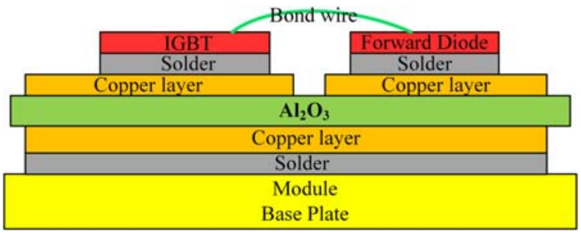


Figure 12.1 The structure of the IGBT module.

transformers can be also modeled using a multidomain approach to further improve the modeling accuracy. Table 12.1 summarizes the different switch models for a qualitative comparisons between them.

The conceptual structure of the multidomain power electronic model is shown in Fig. 12.2. With the multidomain model, the real-time simulation will be capable of mechanism revelation, packaging optimization, condition monitoring, failure analysis, and so on. Taking IGBTs as an example, in the design process of the chip, the electrothermal effect of the cell can be considered to avoid the occurrence of thermal breakdown; in the packaging process the devices can be thermally managed to ensure the performance of the power chip as well as the solder layer, leads, and so on; in the operating cycles the simulation of the thermal behaviors and the mechanic stress of the materials can support the thermal management and detecting the fatigue and failure problems such as the voids in the solder layer and falling off of bond wires; in the system integration process the electromagnetic interference (EMI) to the surrounding devices and circuits can be evaluated to improve the electromagnetic compatibility.

The difficulties of multidomain modeling for the real-time simulation are mainly from the collaborative computing between different physical fields, which is embodied by

1. matching different time scales under different physical domains;
2. the interactions between different physical domains;
3. convergence of iterative solutions algorithm.

Matching different time scales in different physical fields can be problematic. Different physical domains have different time scales, making solving the multidomain model a stiff problem. For instance the stable simulation time-step range of the electromagnetic transient is usually in the nanosecond level to the microsecond level, while for thermal behaviors the range is from the millisecond level to the second level. If we use the nanosecond-level time step to solve the thermal domain, computing resources will be spent unnecessarily. On the contrary, if we use the

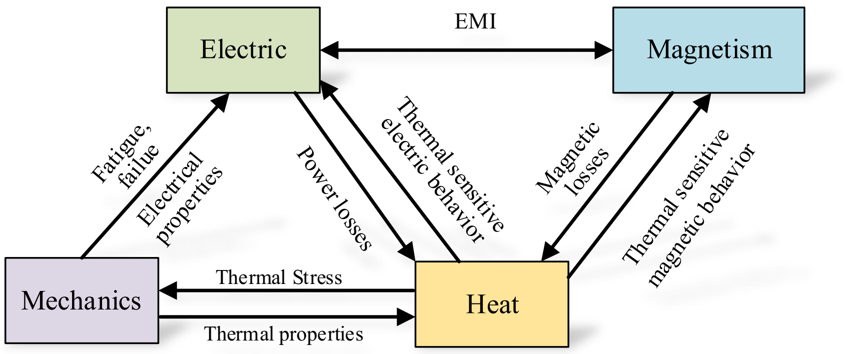


Figure 12.2 The structure of the multidomain power electronics model.

Table 12.1 Comparisons between different switch models.

	Simulation step	**Accuracy**	**Convergence**	**Application**
Static model	μs, ns	Low	High	Large-scale power electronic systems, fast switching applications
Behavioral transient model	μs, ns	Medium	Medium	Detailed power electronic simulation
Physical model	ns or ps	High	Low	Device study

second-level simulation step to simulate the electromagnetic domain, the solver stability and convergence will be lost. In such case, if the direct coupled method is used, where the equations of each physical domain are solved simultaneously with a unified time step, the required computing efforts will be significant, which is not suitable for real-time simulation. The indirect coupling between multiphysics domains is more friendly for real-time simulation. It uses the result of the other physical domains as the excitation of the targeted domains and solves them one by one according to the coupling sequence, which is easy to converge and has a higher computation efficiency. The different domains can use the most appropriate time step to represent their behaviors flexibly. However, the indirect method breaks the multidomain coupling artificially. Latencies will be introduced between domains, which may cause numerical instabilities in the simulation. Besides, due to the different solving strategies, the granularities in different domains are different in terms of discretization, linearization, and grid meshing, which may cause problems in data exchange between domains. The decoupling of domains relies on carefully analyzing their behaviors. The data communication between domains should be fast enough, and compensation algorithms may be needed to improve the simulation stability.

Nonlinearities should also be taken into account in the refined power electronic model, which usually requires iterative solvers and suffers from the convergence problem. As indicated in Chapter 6, the real-time simulation cannot set an error criterion alone to control the number of numerical iterations. Since the computing time is very limited, the maximum iteration numbers should be set. Thus the errors are prone to be accumulated and threaten the convergence of the iterative solver.

In previous chapters, we simulate the power electronic converters based on the numerical solutions of the ordinary differential equations (ODEs). The independent variable is only related to time. However, when dealing with multidomain power electronics models, variables may not only related to time but also related to spatial position. The formation of models is changing from ODE to a partial differential equation (PDE). Thus the numerical solution of PDE for real-time simulations might be an inevitable choice in the future. The three most widely used numerical methods for solving PDEs are the finite element method (FEM), finite volume method (FVM), and finite difference method (FDM). The FEM dominates, especially its efficient higher-order version—hp-FEM. Other versions of the FEM are the generalized finite element method (GFEM), extended finite element method (XFEM), meshfree finite element method, and discrete discontinuous Galerkin finite element method (DGFEM).

Implementing these methods is not easy in the real-time simulation of power electronics, where very limited computing time is available. With the development of more efficient numerical solving algorithms and more powerful computing units, in the future, methods such as FEM could be used to simulate the multiphysics models of the power electronics in real time, which will significantly improve the simulation accuracy and broaden its application range.

12.3 Expanding real-time simulation in the full lifecycle (digital twin application)

So far, real-time simulations applied to power electronics are mostly used in the development cycle, aiding the design and validation process of different DUTs, such as the converter controllers. Suppose we expand the vision to a longer term, including a wide range of operating scenarios over the full lifecycle of power electronics. By achieving that the simulation model can be like a "twin", where the details of the targets can be reflected in a safe simulation environment, while the simulation model can also predict the behaviors of the targets in the future. The real-time simulation in the full lifecycle vision is usually referred to as the concept of the DT.

The concept of the DT will help designers break through the limitations of traditional development tools to achieve new functions [2]. The DT is born in the virtual world and yields values in the physical world. We can look at its potential applications from two stages, that is, the virtual stage and the physical stage, as shown in Fig. 12.3. In the virtual stage the DT model plays almost the same roles as the simulation, including the offline simulation, real-time simulation, and faster-than-real-time simulation, which can be used to evaluate the performance of the target system. For example the offline DT aids the iteration of power electronics designs, the real-time DT is used in the model-in-the-loop (MIL) simulation and the software-in-the-loop (SIL) simulation to evaluate the power electronics control algorithm, and the faster-than-real-time DTs predicts the power electronic behaviors before it happened. We can see that since no real hardware components are integrated into the simulation, the simulation time is not necessarily to be the same as the real time, that is to say, a virtual time axis can be used in the simulation which can be slower, equal to, and faster than real time.

In the physical stage, the DT model can be used not only for the HIL simulation to replace a physical entity in a semiphysical testing environment, but also to

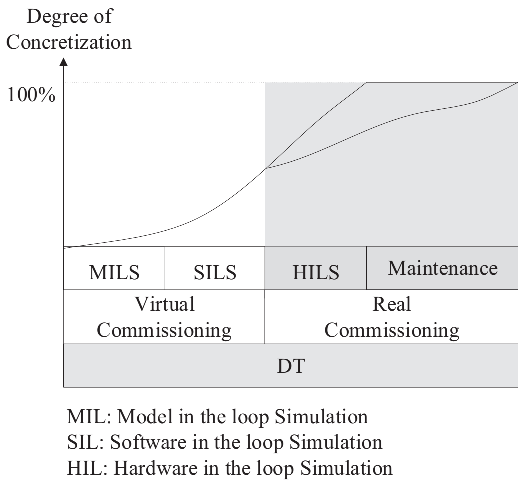

Degree of Concretization
100%
MILS
SILS
HILS
Maintenance
Virtual Commissioning
Real Commissioning
DT
MIL: Model in the loop Simulation
SIL: Software in the loop Simulation
HIL: Hardware in the loop Simulation

Figure 12.3 Digital twin applications in the full life cycle.

operate in parallel with the physical counterpart to indicate the status and prognosis failure, estimate the remaining useful life, and provide guidance for the device maintenance. Different from the virtual stages, the DT in the HIL simulation operates in real time to be able to interact with the real controllers or the subsystems. When testing the high-switching frequency power converters the DT can also operate in a slower than real-time manner as presented in Chapter 11.

In the physical stages, the DT can also be used for management and maintenance, where the simulated models also exchange data with the physical systems [3]. Benefiting from the data exchange from the physical system, model self-learning can be realized to adjust the model parameters, model forms, and simulation strategies to align the model's behaviors to the physical system. In these cases, the DT can operate in both real-time and faster-than-the-real-time simulations. In the real-time condition, the DT can monitor the operating status of the power electronic system by comparing the results of the simulation with the results sensed from the system, to provide a deeper and better insight into the systems and components' behaviors, detecting the abnormal occurrence, and prognosis of the potential failure. In the faster-than-real-time condition, the behaviors of the system can be predicted to support the decision-making of the system management, and the remaining useful life of the power electronic components can be predicted to support the reasonable schedule of the system maintenance. Moreover, by combining virtual reality technology, the DT helps to provide a "mirror" of the physical system that people can visualize.

Regarding the application in the renewable power system, we can expect the DT applications in different parts of the energy system, including energy production, transmission, distribution, storage, and consumption.

In power production, the DT model includes wind turbines, photovoltaic systems, hydrogen fuel cells, power plants, etc. By using the DTs model, we can perform the following functions:

1. monitoring the operating status and operating environment of the energy production units;
2. formulating the optimal operation strategy of each energy production unit;
3. predicting the power production data.

For the power transmission part, the DT model can help to control and optimize of the energy transmission process, such as:

1. reflect the status of the operation status of power transmission equipment and the load status of each node;
2. monitor the power grid in real time through big data and intelligent algorithms;
3. warn the possible risks if there is a significant difference between the real-time results and the physical data captured from the transmission nodes;
4. guide the design of the cables in view of the full life cycle to improve the performance and increase the lifetime.

For power distribution and consumption, the DT can enable:

1. monitor the operating status of the distributed networks and the apparatus;
2. predict the load behaviors to optimize the power distribution logic;
3. detect fault and redistribute the power for uninterrupted power supply;
4. guideline for users of the economic evaluation of electricity usage;

For energy storage, the DT can enable:

1. health management of the storage unit in the full life cycle;
2. size and optimize the storage configuration;
3. verifying the energy management strategies.

The future applications of DTs can be broad, not just limited to the previous outlooks. With the emergence of the "metaverse" concept, the DT will surely play an important role in the future industrial applications of power electronics.

12.4 Worldwide collaborative simulation using GDRTS

HIL simulation is an efficient way to predict how a device will interact with the real operating scenario (i.e., to determine its closed-loop response). The HIL simulation usually requires the DUT and the simulator to be located in the same place. However, a large network may include many devices that are situated in different laboratories. For example, when simulating the distributed energy system, the power generated by the wind farms in Germany may be consumed by the electrolytic tanks for the H_2 storage in France, where multiple DUTs are geographically distributed. In addition, not every location may have a HIL platform available for testing purpose. As a result, there is always a remote HIL test requirement for the DUTs through the shared simulator and interface.

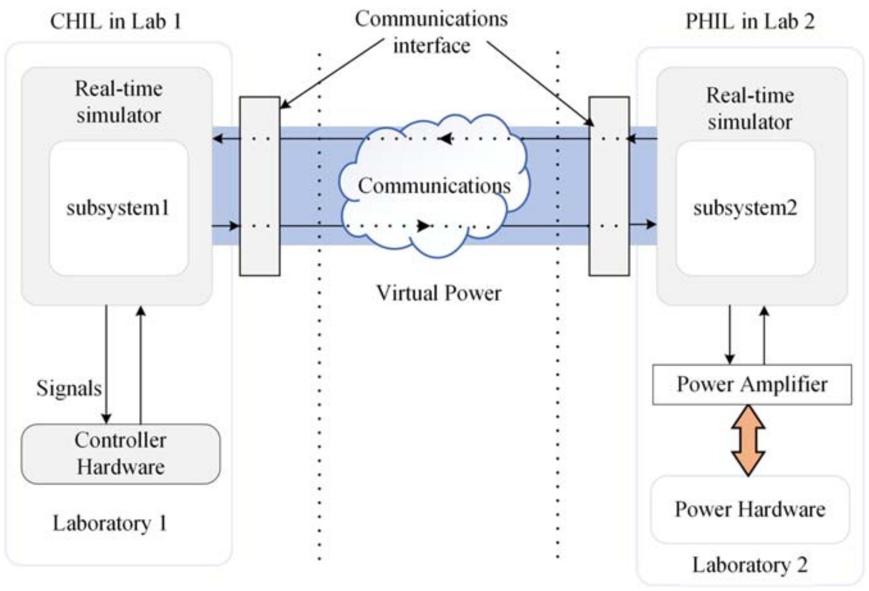


Figure 12.4 Geographically distributed real-time simulation [5].

GDRTS is one of the solutions. GDRTS represents a vendor-neutral distributed platform based on the virtual interconnection of digital real-time simulators (DRTSs) and HIL setups hosted at geographically distributed locations. Fig. 12.4 describes the efforts toward the realization of this large-scale virtual infrastructure and explains a demonstration of the multilab setup for the simulation and testing of next-generation global power grids [4].

From the discussion from Chapters 8 and 9, we know that bandwidth, accuracy, and stability are critical factors in real-time simulation, which can all be constrained by the long communication latencies in the GDRTS. As shown in Fig. 12.4, data are exchanged between the model and the hardware via the CHIL or PHIL interface, while data are also exchanged between GDRTS nodes via the communication interface. The total response time is much longer than the response time of the conventional standalone real-time simulation. To avoid the risks caused by the long response time, the communication latency can be reduced by using advanced 5G/6G communication technology for example. Meanwhile the network should also be carefully partitioned to minimize the effects of the latencies. Besides, the interface algorithms (IA) can also be used for compensation of the latency-related effects to improve the GDRTS stability and fidelity. With the help of GDRTS, a larger simulation networks can be constructed, and the simulation resources can be shared to save the repeated modeling and simulation efforts. Meanwhile, by adopting GDRTS, collaborative HIL real-time simulations can be further developed for the-power electronic systems worldwide.

References

[1] T. Grasser, T.W. Tang, H. Kosina, S. Selberherr, A review of hydrodynamic and energy-transport models for semiconductor device simulation, Proceedings of the IEEE 91 (2) (2003) 251−274.

[2] M. Zhou, J. Yan, D. Feng, Digital twin framework and its application to power grid online analysis, CSEE Journal of Power and Energy Systems 5 (3) (2019) 391−398. *Electronics*, *35*(9), 9850-9864.

[3] Ebrahimi, A. (2019, June). Challenges of developing a digital twin model of renewable energy generators, in: *Proceedings of the IEEE Twenty-Eighth International Symposium on Industrial Electronics (ISIE)* (pp. 1059-1066). IEEE.

[4] L. Pellegrino, D. Pala, E. Bionda, V.S. Rajkumar, R. Bhandia, M.H. Syed, V.H. Nguyen, Laboratory coupling approach, European Guide to Power System Testing, Springer, Cham, 2020, pp. 67−86.

[5] A. Monti, et al., A global real-time superlab: enabling high penetration of power electronics in the electric grid, IEEE Power Electronics Magazine 5 (3) (2018) 35−44. Available from: https://doi.org/10.1109/MPEL.2018.2850698.

Index

Note: Page numbers followed by "*f*" and "*t*" refer to figures and tables, respectively.

REAL-TIME SIMULATION TECHNOLOGY FOR MODERN POWER ELECTRONICS

Real-time Simulation Technology for Modern Power Electronics provides an invaluable foundation as well as a state-of-the-art review in advanced implementations of real-time simulation for power electronics. The work opens with a discussion of power electronics device modeling, component modeling, and power converter modeling before addressing numerical methods to solve converter model, emphasizing computational efficiency and accuracy. It discusses both CPU-based and FPGA-based real-time implementations and provides an extensive review of current applications, including hardware-in-the-loop (HIL) and its case studies in the electric vehicle and microgrid applications. The work closes with a review of the near and long-term outlooks for the evolving technology. Collectively, the work provides a systematic resource for students, researchers, and engineers in the electrical engineering and other closely related fields.

- Introduces the theoretical building blocks of real-time power electronic simulation through advanced model implementations
- Completes relevant case studies and implementations across diverse applications, including electric vehicle component testing and microgrid controller testing
- Discusses FPGA-based and CPU-based real-time simulation techniques, complete with illustrative examples, comparisons, computational performance analysis, and cosimulation architectures

Hao Bai is currently working as an associate professor at Northwestern Polytechnical University, China. His main research interests include real-time simulation of power electronics systems, multilevel multidomain modeling and simulation, and digital twin technology.

Chen Liu is currently an associate professor in electrical engineering in Zhengzhou University, Zhengzhou, China. His main research interests include real-time simulation of power electronic system and HIL test.

Dusan Majstorovic is currently working as the chief technical officer at Typhoon HIL and a member of the team that developed both the theoretical foundation and hardware platform for the world's first programmable FPGA solver–based HIL real-time emulator platform for power electronics. He leads both the technology and product development and is the chief architect of Typhoon HIL FPGA solver. His field of interest is focused on real-time high-performance computing.

Fei Gao is currently working as a professor at the University of Technology of Belfort-Montbeliard (UTBM) in France and the editor-in-chief of *IEEE Industrial Electronics Technology News*. His main research fields include fuel cells for transportation and digital twin technology for modern power electronics and energy systems. Prof. Gao is a Fellow of Institute of Electrical and Electronics Engineers (IEEE) and a Fellow of Institution of Engineering and Technology (IET).

ISBN 978-0-323-99541-2

9 780323 995412

ELSEVIER

ACADEMIC PRESS
An imprint of Elsevier
elsevier.com/books-and-journals