Device-Level Power Electronic System Emulation for Hardware-in-the-Loop Applications

by

Wentao Wang

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science
in
Energy Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Hardware-in-the-loop (HIL) simulators are prevalent in many industries and are playing a significant role in the design and testing of new equipment. While detailed models of power system components are available for HIL simulators, there is limited knowledge in the area of power electronic converter modeling. Currently available HIL simulators employ simpler models for power electronic converters, based on ideal or averaged switch models. While such models are adequate for system-level performance evaluation and analysis, there are seldom sufficient for the analysis of device-level stresses, electromagnetic interference (EMI), and parasitics, which are especially important at high switching frequencies.

This thesis develops device-level models for power electronic converters for HIL simulation. Detailed device-level hardware models are developed for the insulated-gate bipolar transistor (IGBT) and the power diode on the field-programmable gate array (FPGA). The hardware design are fully paralleled using an IEEE 32-bit floating-point precision to achieve the lowest latencies and resource consumption. An efficient variable time-stepping algorithm is proposed for the solution of the nonlinear device equations. Case studies for DC-DC and DC-AC converters are emulated and validated, showing good agreement.

Acknowledgements

First, I would like to show my deepest gratitude to my supervisor, *Dr. Venkata Dinavahi*, who has provided me valuable guidance in every stage of my research. Without his insightful instruction, unlimited patience, and tremendous passion, this work could never have been completed.

I also want to extend my thanks to my M.Sc. committee members *Dr. Douglas Barlage*, *Dr. Hao Liang* for reviewing my thesis and providing valuable suggestions. Furthermore, I will thank all my lab-mates in RTX-Lab for their infinite kindness and generous help.

Specially, I want to thank my wife, *Mengmeng Han*. Without her moral support and love, I would never have completed this hard work.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AC	Alternating Current
A/D	Analog to Digital Converter
BJT	Bipolar Junction Transistor
CLB	Configurable Logic Block
CPU	Central Processing Unit
D/A	Digital to Analog Converter
DC	Direct Current
DSP	Digital Signal Processing
FF	Flip Flop
FIFO	First In First Out
FPGA	Field-Programmable Gate Array
GPU	Graphic Processing Unit
HIL	Hardware-in-the-Loop
HUT	Hardware under Test
IC	Integrated Circuit
I/O	Input/Output
IOB	Input/Output Block
IP	Intellectual Property
LAE	Linear Algebraic Equation
IGBT	Insulated-Gate Bipolar Transistor
LTE	Local Truncation Error
JTAG	Joint Test Action Group
LUT	Look-Up Table
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
NAE	Nonlinear Algebric Equation

ODE	Ordinary Differential Equation
PLL	Phase Locked Loop
PO	Potential Output
PWM	Pulse Width Modulation
(S)RAM	(Static) Random Access Memory
STF	State Transit Flag
(V)HDL	(Very-High-Speed Integrated Circuit) Hardware Description Language
VLA	Vivado® Logic Analyzer
VLSI	Very Large Scale Integration
VTCM	Variable Time-Step Control Module
VTOM	Variable Time-Step Output Module

1 Introduction

Hardware-in-the-loop (HIL) simulators are increasingly being used in many industrial applications. A HIL simulator is an embedded hardware system which connects to the external system under test via various I/O interfaces (e.g. A/D and D/A). Because of its unique advantages of low cost and of excluding the possibility to damage the physical devices under unexpected transient conditions, such as faults when performing system tests, it is gaining popularity in many industries, such as automotive, aerospace, and electric power systems.

Being a reconfigurable hardware which has abundant resources of logic gates, RAM blocks, a field-programmable gate array (FPGA) is ideal for the design of a HIL simulator. Nowadays, the modern generations of FPGAs tend to have much more hardware resources and faster execution speed. For example, the newly released Virtex® UltraScale™16nm devices have up to 5 million logic cells. Given this trend of FPGA technologies, modeling complex device physics in hardware is possible.

Traditional power electronic system hardware emulation used ideal switches or average models to represent the switching devices such as IGBT and power diode. These models are only suitable to perform system-level evaluation. Modern power converter circuit hardware emulation requires even more with its switching device models so as to look into the high-frequency effects including device stress, parasitic, and EMI noise. Potential areas of applications could also includes parametric, statistical, and sensitivity analysis. Therefore, it is the right time to introduce detailed physics-based power device models into the realm of power converter hardware emulation.

1.1 Literature Review

This section provides a brief introduction to HIL simulation, FPGA applications in HIL simulation, and device-level models for HIL simulation.

1.1.1 HIL simulation

HIL simulation is able to perform thorough and repeated testing and evaluation to observe the transient dynamics even before the actual apparatus has been produced. While the simulation should be in real-time in order to fully reflect the characteristics of emulated apparatus and test conditions, HIL simulation can also be used in an open-loop fashion to accelerate complex system models. Moreover, the time-step should be sufficiently small so as to contain the model's detailed transients. A typical HIL simulation setting usually contains a piece of hardware under test (HUT), a real-time simulator, and the interface between the hardware and simulator. Based on the different types of the interface, the HIL simulators are classified into controller HIL (CHIL) and power HIL (PHIL). The interface of CHIL typically contains D/A and A/D, which do not adjust the power flow through the interface. In contrast, the HUT for PHIL needs a certain degree of power flow. Therefore, besides the D/A and A/D, amplifiers are needed to increase the power flow between the real-time simulator and the HUT. [1]

Currently, the HIL simulation has found widespread applications. For example, [2] introduced an effective interfacing technique between power systems with digital controllers. The related real-time HIL simulator could be implemented on DSP. Reference [3] designed a real-time HIL simulator for D-STATCOM applying both DSP and FPGA, since a single FPGA chip was still limited in terms of hardware resources to contain the whole system at that time. In order to speed up the testing process of wind energy system control methods, [4] introduced a real-time HIL simulation platform based on custom software and DSP hardware, which tracked the maximum turbine power line using neural networks. Reference [5] developed a HIL test platform for power electronic control system design, since the off-line software-based simulation is limited to provide the actual operational conditions. However, the test platform was only capable of integrating models from various commercial or noncommercial software for co-simulations. Even if the high-speed FPGA was introduced to the platform, it is only used as a supplementary technique to improve the simulation performance. In [6], the HIL simulation concept was even been applied to education. Specially, the wind-energy-conversion system was divided into many subcomponents, which was represented by their mathematical relationships. Thus, the HIL simulation of the whole system was successfully performed. Reference [7] based their HIL power converter system on a novel adaptive discretization scheme, however the im-

plementation was mostly done on sequential micro-processes with only the controller part was done on FPGAs. Reference [8] developed a DSP-based HIL simulator for induction motor drive in order to assist the related design issues. Around this time, the advantages of FPGA were widely recognized and applications of FPGAs for HIL simulation started appearing in the literature, which is discussed in the next subsection.

1.1.2 FPGA Applications in HIL Simulation

FPGA is the ideal platform to embed HIL simulator, due to its intrinsic advantages of full hardware parallelism and user reconfigurability. Thanks to the advances in very large scale integration (VLSI) technology and in digital hardware design tools, FPGAs have been used in many applications. The power industry applications can be roughly classified into: (1) control applications and (2) system modeling applications. There are numerous applications among the first group. For example, [9] made a thorough survey of the FPGA technologies and its applications in industrial control systems. More specifically, the FPGA-based controllers were finding widespread applications in embedded industrial and robotic and power electronics and drive systems. These controllers could even be designed using sophisticated artificial intelligence, neural networks, and fuzzy logic, which proved that the FPGA is an ideal platform to embed complicated algorithms. Reference [10] based a senseless controller on FPGA for a kind of synchronous machine. This controller was wholly embedded into a single FPGA board and had high performance. Reference [11] presented FPGA-based fuzzy-logic systems. The inner digital signal processing systems were designed on FPGA using specific MATLAB/Simulink® tools. The generated files could be immediately translated to netlists and bitstream files agreeable to FPGAs, which largely reduced the hardware design efforts. Reference [12] designed a sensorless ac drive using FPGA. Specially, the control unit was realized in both software and hardware. Among the system modeling applications, a real-time electromagnetic transient simulator of large-scale transmission line system was embedded on FPGA [13], which achieved both calculation accuracy and speed. Reference [14] based a novel processor on FPGA to build the testing platform for power converter system design. Reference [15] built a real-time simulator for the solution of the nonlinear electromagnetic transient in transmission line systems on FPGAs rather than sequential GPPs or DSPs, so as to promote emulation performance. In [16], a three-level voltage source converter system and a squirrel-cage induction motor system were emulated on FPGA, and the switching devices with different degree of accuracy for the converter were applied and compared. In [17], complicated universal machine and universal line models were emulated on FPGAs, which had been challenging to be embedded on traditional hardware platforms. In [18], multi-FPGA implementation was even performed to make the emulation of large-scale power system with detailed models to be possible. This practice fully demonstrated that FPGAs, as one of the most popular HIL platforms, can handle large nonlinear systems. Reference [19] intro-

duced a unified modular concept to deal with detailed modeling of various power system components for modern power grid on FPGAs, which was featured by full parallelism and deep pipeline. In [20], power system relay models were emulated on FPGAs instead of traditional sequential DSPs. The upgraded paralleled and pipelined hardware structure made the fault detection much faster. Reference [21] emulated the nonlinear power transformer on FPGA, which overcame the challenges to achieve fast execution time for the emulation of large-scale power system with complex transformer models. In [22], models for different types of electrical machines were implemented on FPGAs, which were proved to be very effective. Today, newer generations of FPGAs have much larger capacity in terms of logic blocks, distributed memories, DSP slices, and other advanced features such as partial reconfiguration, which makes the detailed physics-based device-level power electronic system HIL simulation feasible.

1.1.3 Device-Level Models for HIL Simulation

An accurate physics-based power electronic device digital hardware emulation has yet not been reported in literature, since the greatest obstacle is their computational burden. To observe the switching transient of the devices also implies infinitesimal time-step hence high execution speed, which is another obstacle. One of the compromises is to use approximate device-level models to perform real-time simulation. For example, [23] used linearized device-level characteristics of the IGBT. The collector-emitter voltage of the IGBT was modeled as a constant threshold voltage in series with an on-state resistance. Both the values of the voltage and resistance were specified on the manufacturer's datasheet. The switching times were estimated using a analyzing circuit setting provided by the manufacturer. The resultant IGBT model was experimentally validated to have reasonable rise, fall, and delay times, which was a significant progress compared with ideal switch models of IGBT. A more accurate IGBT model was introduced in [24], which used look-up tables to model the nonlinear switching characteristics of the IGBT. The basic theory behind this method is that the turn-on and turn-off switching times and tailing current are proportional to the steady-state values of collector current and emitter-collector voltage. Therefore, those per-unit device characteristics could be obtained by hard switching the actual IGBT. This IGBT model took a big step towards modeling nonlinear characteristics of IGBT in real-time. However, its transient behaviors were intrinsically based on estimations. The accuracy of the models could only be assured within limited operating conditions.

IGBTs and power diodes are the fundamental switching elements in modern power electronic systems. They also contribute significantly to the circuit nonlinearity. Much literature has been devoted to device-level model development for both the IGBT and the power diode. Device-level models for these two types of switches can be mainly classified into: (1) physics-based analytical models, (2) behavioral models, and (3) numerical models.

As the name implies, the physics-based models describe the carrier dynamics in the

semiconductor material in terms of nonlinear ordinary or partial differential equations with respect to time. The numerical solution methodology normally includes implicit discretization followed by Newton or Katzenelson iterations. For the IGBT two of the most detailed and popular models in this category are the Hefner [25], [26], [27] and Kraus [28] models. Specifically, Hefner modeled n-channel IGBT as a p-n-p BJT with its base connected to an n-channel MOSFET. The inner MOSFET was modeled similar to the power Vertical double Diffused MOSFET. The inner BJT transients such as the carrier distribution were obtained by solving its ambipolar transport equations. Moreover, Hefner modeled faster buffer layer IGBT, with its different switching behaviour discussed. Kraus modeled a vertical IGBT with high charge carrier lifetime and low emitter efficiency. All these models were experimentally validated to have high accuracy. For the power diode, the lumped charge models proposed by Lauritzen and Ma *et al* [29], [30] are quite famous, which had all the main inner phenomena of power diode represented using charge transport equations. Those phenomena included the forward recovery, reverse recovery, emitter recombination, junction capacitance, and contact series resistance effects. Their power diode models were proved to be effective and easy to be utilized in hardware emulation.

The behavioral models (macromodels) approximate the device behavior from measured and fitted characteristics that are incorporated into the simulation using lookup tables and discrete circuit components. For example, [31] presented two behaviour IGBT model structures with different degree of accuracy. Both of the models utilized a nonlinear unit to represent the IGBT static characteristics, and a series linear unit to describe the IGBT dynamic characteristics. Most of the model parameters could be extracted from physical devices or more accurate models by least-squares methods. Other parameters could be found from the manufacturer's datasheet. The complexities of the models were not very high, and their accuracies were satisfactory. Reference [32] modeled the IGBT with antiparallel diode pair as ideal switches and customizable voltage and current sources to represent the actual transient behaviours. The actual switching transients were obtained by scaling the per-unit measurements of physical devices transient waveforms. The validation results for this model showed satisfactory accuracy. Bond graph methods to build averaged models for power converters that include device nonlinearities have been proposed in [33].

The numerical models are highly exact two-dimensional (2D) or three-dimensional (3D) models [34] based on nonlinear partial differential equations in space-time that describe physical phenomena in the semiconductor material such as carrier generation, recombination, drift, and diffusion. The numerical solution methodology typically involves the finite element or the finite difference methods. To reduce high computational burden of such methods, hybrid models [35] have also been proposed which combine the analytic and numerical models.

This thesis does not select the most accurate but complex numerical finite element models to build power converter hardware model, since their computational burden is too high

to gain satisfactory execution speed. Instead, the analytical device models are found adequate to observe the device transient behaviors. Specifically, the IGBT and the power diode physics-based device-level digital hardware models were emulated on the FPGA. The device nonlinear equations are solved using a fully iterative Newton-Raphson method with a variable time-step. The IGBT hardware emulation utilizes the physics-based Hefner's model, and the power diode hardware emulation utilizes the Lauritzen's model. The proposed FPGA-based emulation also includes other power electronic system elements such as independent/dependent supply sources, linear/nonlinear lumped R-L-C elements, and pulse-width modulation, which makes it a complete power electronic circuit emulator. A variable time-step implicit discretization of the device equations is employed to gain computational advantage, and result comparison is presented with respect to Saber®. The entire hardware design is fully paralleled using IEEE 32-bit floating-point number precision to achieve high speed and accuracy requirements.

1.2 Challenges and Objectives of the Work

When power electronic circuit hardware emulation is based on physics-based device-level analytical models, the difficulties will surely be escalating compared with the traditional system-level models. The main challenges are listed as follows:

- Physics-based device-level analytical models typically comprise a great many highly coupled nonlinear differential equations. Solution of these equations requires rather complex processes of discretization and linearization. The resultant discrete-time linearized equivalent circuit equations must be tested with care, since small errors may lead to failure of this process.

- The discrete-time linearized power converter hardware model includes numerous hardware modules, units, and even sub-units. There lot of efforts in this work was focused on how to organize these hardware components and exploit the parallelization characteristics of FPGA. For example, the calculation sequence must be reasonably selected to reduce the latency. The timing sequences of inner signals must be made clear beforehand.

- For a circuit system with less nonlinearity, a large part of the system matrix could be diagonalized [36], which makes it possible for a very big matrix (e.g. 100×100) to be divide into many smaller matrices and solved in parallel. However, the system matrix of the power electronic circuit based on detailed physics-based analytical device model is obviously highly coupled and larger in system dimension (because of the inner nodes inside the devices), which means it is very difficult to be diagonalized and solved in an efficient way. Note that the latency for any existing matrix solver will increase exponentially with respect to the dimension of the matrix. Therefore, if the

system matrix cannot be solved in a more efficient way or the larger converter system cannot be divided in some way, the hardware emulation for the physics-based device-level power electronic circuit will fail.

- Another challenge for the work is that all of the previous HIL simulators are based on fixed time-step strategy, and the output waveforms of the hardware is naturally in shape. However, if the HIL simulators for the power electronic circuit based on complicated analytical model, the fixed time-step set-up will prolong the simulation time by times since the time-step had to be set very small so as to capture the transients during very tiny range of time (could be reaching to the level of nanoseconds). Obviously, the previous fixed-time strategy loses its value. By contrast, most off-line software-based circuit simulators support variable time-step strategies, and there is no problem to draw their resulting data with respective to the timeline (usually the x-axis). But for the hardware emulation, if the variable time-step strategy is imitated indiscriminately, the output waveforms will probably be out of shape and lose value.

- When physics-based device-level analytical switching element models are applied, the complexities of the whole power electronic systems are escalating. As a result, the hardware model of the power converter tends to be not converging, which needs to be paid much attention to. To prevent non-convergence, either some circuit simulation adjusting parameters are pretested, or some effective fail-safe strategies are applied to the hardware models.

- Finally, Newton-Raphson method has been widely used in both off-line software-based circuit simulators and hardware emulations. The numerical methods used in transient analysis begin by the system matrix formulation to represent the system as a group of nonlinear ordinary differential equations (ODEs). These ODEs are in turn transformed to nonlinear algebric equations (NAEs) before being linearized to linear algebraic equations (LAEs) using the classical Newton-Raphson method or others. During the whole transformation, the nonlinear part of the system (e.g. IGBT and Diode) and linear part of the system (e.g. load resistors and inductors) mingle with each other in the system representations. In other words, for different topologies of the power converter circuit, repetitive discretization and linearization work had to be done. This is cumbersome since the physics-based device-level power electronic circuit system equations are extremely complicated because of the complexities of the device inner nonlinear representations. As the kernel of the standard numerical methods, the classical Newton-Raphson method should be adjusted in the power electronic circuit hardware emulations in this thesis so as to be agreeable to the modularization of the nonlinear device models, which could in turn be combined to form different topologies of power converters.

In order to build physics-based device-level electronic circuit HIL simulators on FPGAs, the main research objectives of this work are as follows:

- When building electronic circuits based on physics-based analytical device models, the complexities of the circuits can mostly be attributed to the nonlinearities inside each devices, e.g. IGBTs and Diodes. Moreover, most of the power electronic circuits contain more than one devices, e.g. for a standard two-level voltage source DC-AC converter, 6 pairs of IGBT and Diode are included, so the nonlinearities of the whole converter circuit are escalating. Those off-line device-level circuit simulators (e.g. PSpice and Saber) have built up a group of systematic methods to tackle these difficulties. For example, SPICE-like simulators apply a modified nodal approach to formulate the nonlinear circuit system equations. Also, most of the off-line simulators support a variable time-step simulation strategy, which is essential when performing transient simulation based on detailed analytical device models. Although the solution process of the off-line circuit simulators are intrinsically different from FPGA, e.g. circuit simulators are mostly based on high-level programming languages which work in a sequential way, while FPGA can function parallel as integrated circuit hardware, it is still worthwhile to study. Therefore, a survey of the existing device-level software-based circuit simulators was made before the hardware design.

- In order to emulate the physics-based device-level power converter circuit, the discrete-time linearized physics-based analytical models of IGBT and power diode are essential before hardware emulations. Thankfully, G. T. Oziemkiewicz has done this work for the IGBT before [37]. However, the work for the power diode has to be done by ourselves. Both the hardware module designs of IGBT and power diode should be based on their discrete-time linearized models. Their inner structures should be designed to be in fully parallel.

- As is mentioned before, a highly coupled physics-based device-level power converter circuit system usually has a larger dimension of system matrix, which is an obstacle for its hardware emulation. Therefore, the existing linear solver may not be satisfactory. A novel parallel Gauss-Jordan linear solver, which is so fast as to tackle a matrix with a dimension larger than 7, is required to fulfill the task.

- The large discrete-time linearized system of the power converter circuit needs to be fully parallelized on FPGA. Hence, an improved unified numerical framework should be built up to avail the parallelization of those hardware modules. In other words, a flexible modularization methodology needs to be introduced, which could be referred by future practice of power electronic HIL simulator design. The aim of this methodology is to model different topologies of power converter with less programming efforts. The hardware modules should be reusable as much as possible.

- For traditional power converter hardware emulation, which based the switching devices on ideal switch or averaged model, a fixed time-step strategy is typical and sufficient. However, when the detailed device-level models are applied to observe a few nanoseconds' transient contents, the traditional fixed time-step strategy would be ineffective. Obviously, if the time-step is set too small, the execution speed could not be assured. Therefore, a variable time-step strategy is mandatary to the hardware emulation with device-level models and novel hardware modules should be designed and tested in the new practice since a variable time-step strategy has not be utilized in the realm of hardware emulation before.

1.3 Thesis Outline

This thesis contains 6 chapters. The other chapters are organized as follows:

- Chapter 2 reviews the main technologies of FPGAs. Specially, the overall architecture and sub-components of Xilinx® 7 series FPGA are described in details. The newly Xilinx® Vivado® design tool for FPGA and its design flow are presented.

- Chapter 3 make surveys off-line device-level circuit simulators, which have systematic methods to deal with physics-based device-level power electronic systems. Specially, solution approaches of circuit simulators are introduced and compared. Also, the co-simulation interfaces between different simulators are listed and discussed. Many of these simulation and interface techniques of circuit simulators could be applied in HIL simulators.

- Chapter 4 describes the hardware emulation on the FPGA of physics-based power diode and IGBT models, respectively. The formulations of both the two devices are presented as well as their model discretization and linearization. The overall structures of their hardware emulation modules are detailed. Also, this chapter describes the methodology the whole structure of the power electronic circuit hardware emulation, which flexibly combines both the two device hardware modules. A highly efficient linear solver is also presented as well as sub-modules which makes the variable time-step strategy possible in hardware emulation.

- Chapter 5 presents case studies of a DC-DC buck converter and a 2-level DC-AC converter. All of their steady-state and transient validation results are compared with Saber®. Furthermore, their power dissipation analyses are performed and compared.

- Chapter 6 gives conclusions and contributions of this thesis. A brief summary of the work and some suggestions for the future work are made.

2 FPGA Overview

Although custom integrated circuit has the advantages of good performance, wasting no hardware, it has many disadvantages such as great design and testing costs. More importantly, if the designs are found errors, the total costs to correct them will be escalating. FPGAs, on the other hand, have fast time-to-market. Even if the mask is changed, it can be reconfigured with ease. Compared with microprocessors, although modern CPUs and GPUs operate at a higher clock frequency than FPGAs, they could hardly reach the performance of FPGAs because of their overhead of operating system and other abstraction layers. Most importantly, microprocessors intrinsically work in a sequential way, which could be nowhere to compare with FPGAs featuring by hardware parallelism in design architecture. Obviously, multi-core CPUs will be much slower than FPGAs, which implement the hardware resources directly not to mention their high prices compared to FPGAs.

The modern FPGAs has the trend to grow in hardware resources as well as operating speed. The Virtex® UltraScale™16nm devices, for example, have as many as 5 million logic cells with the operating speed around 100 to 500 MHz. Given this trend of growth in FPGA size and speed, modeling complex device physics in hardware becomes possible on FPGAs.

This chapter will give a brief introduction to the FPGA technology. Its whole architecture as well as its subcomponents will be discussed first. Then the design flow of FPGA design tools will be described too.

2.1 FPGA Architecture

A FPGA is a programmable integrated circuit which contain a two dimensional array of logic blocks, hierarchy of reconfigurable routing channels interconnecting the blocks, and

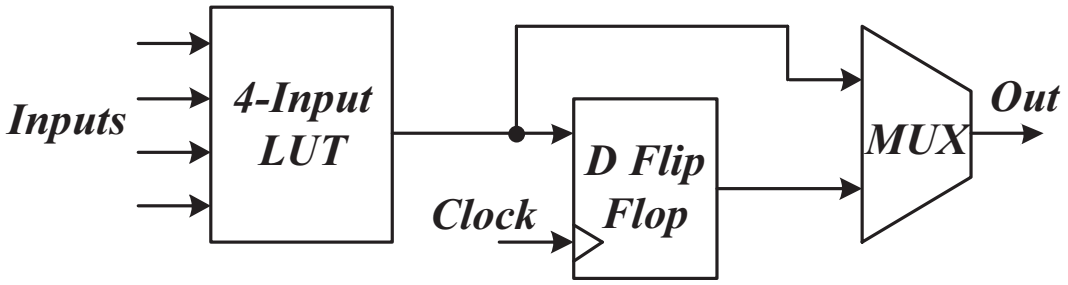

Figure 2.1: Conceptual structure of a logic cell [13].

input/output (I/O) pads fitting into the array. For most FPGAs, a logic block (Fig. 2.1) is simply built up by one 4-input look-up table (LUT), and one D-type flip-flop, as shown below. The only output is either a registered or unregistered LUT output. Despite its simple structure, it can implement different combinational and sequential logic functions with different degrees of complexities. [13]

The routing channels are basically the electrically programmable interconnections of wiring and switches, which are physical transistors controlled by fuses, antifuses or memory cells. Engineers are given the opportunities to configure the function of those logic blocks the interconnection between them. These configurations are realized by manipulating a switch box, which lies in every intersections of the horizontal and vertical routing channels. Among the intersected four wire segments, every two of them are connected by a programmable switch, which is controlled using different programming technologies based on SRAM (Static RAM), flash, and antifuse. SRAM-based FPGAs are the most popular group of FPGAs, which include most chips of Xilinx® Virtex® (used in our designs) and Spartan® families. Specifically, they store the configuration file in the volatile static memory, which needs to be reconfigured after the power source turned off. The I/O pads is located at the periphery of an FPGA connecting to its adjacent wire segments, which are used to send out the output signals. [13]

Since one of the Xilinx® 7 series FPGAs, Xilinx® Virtex®-7 FPGA, is applied to the design of the physics-based device-level power electronic circuit hardware emulation, their architectures will be specifically detailed in this thesis. The architecture of traditional Xilinx® FPGA was typically a two-dimensional symmetrical array structure, which contained a large number of logic blocks, named Configurable Logic Blocks (CLBs) , organized into a matrix. The inside part of the CLB matrix was interconnected by wire segments and programmable switches, while the periphery of the matrix was surrounded by a layer of Input/output blocks (IOBs). The connection blocks link the input and output pins of CLBs to their adjacent routing channels from all four directions. Specifically, they used route switches to connect those CLB pins to their surrounding wire segments. However, Xilinx® 7 series FPGAs begin to utilize upgraded structures, which organize their different resources including CLBs, IOBs, Block RAMs (BRAMs) and DSP slices in columns (see

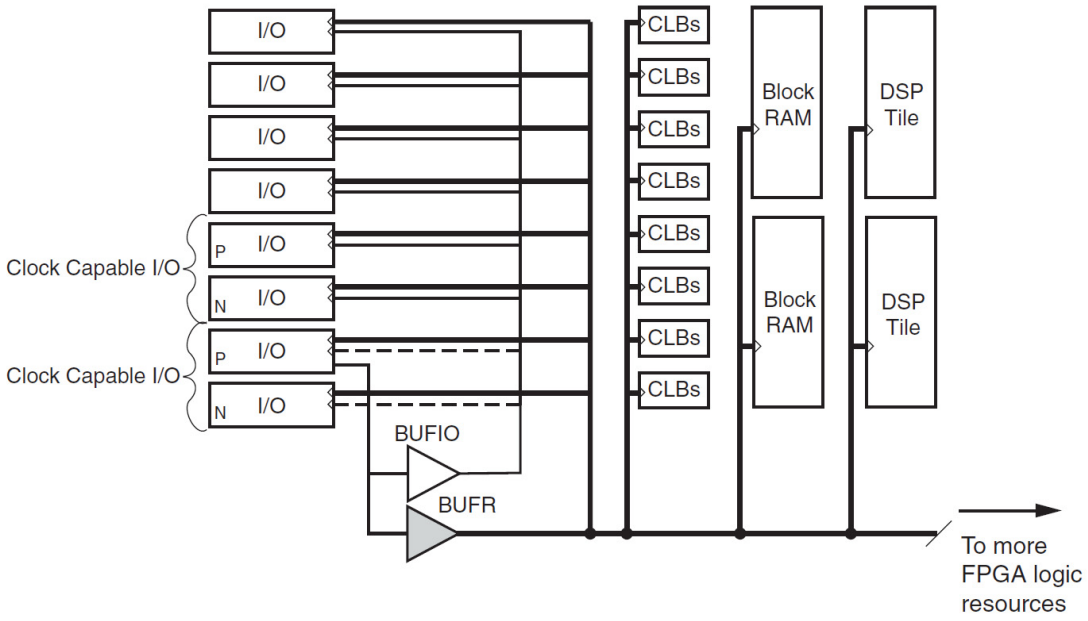

Figure 2.2: Common structure of 7 series FPGAs [38].

Fig. 2.2). The function of the modified structure is basically the same as the old one, but it manages the hardware resources in a more flexible and efficient way. [38]

2.1.1 Configurable Logic Block (CLB)

A 2-slice CLB contains two separated logic cells (Fig. 2.3). Each slice (logic cell) includes two four-input LUTs utilized as function generators directly connected to input pins, two D-type flip-flops, and two registers near the output side. The logic implemented in the LUTs is stored in static RAM (SRAM). The multiplexers inside the logic cell are also controlled by SRAM. [39]

2.1.2 Block RAM (BRAM)

BRAMs are located on left and right columns of CLB array. As a fully synchronous dual-ported 4096-bit RAM, BRAMs are responsible for storing addresses, data and write-control signals for inputs. A functional diagram of a BRAM is shown in Fig. 2.4. [40]

2.1.3 DSP48E1 Slice

The 7 series FPGAs also contain many low-power digital signal processing (DSP) slices, which can implement a lot of math functions including multiplication, accumulation, addition, and abstraction in a fully custom and parallel way. As can been seen in Fig. 2.5, since a DSP slice uses binary multipliers and accumulators to performing the aforementioned algorithms, it has the advantages of fast computational speed and capacity as well

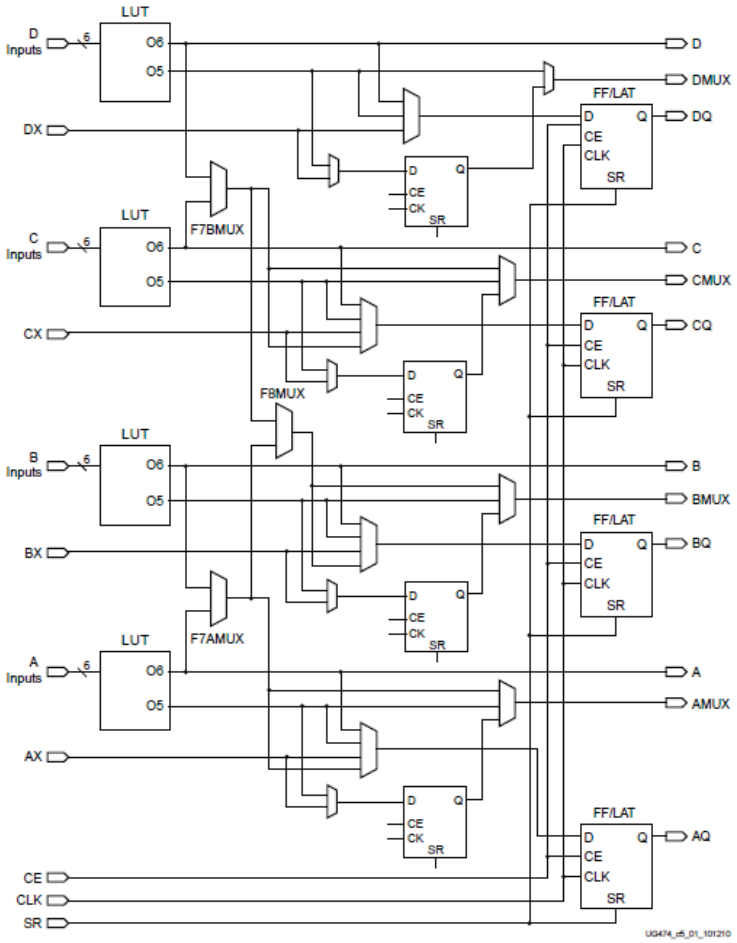

Figure 2.3: Structure of a logic cell in 7 series FPGAs [39].

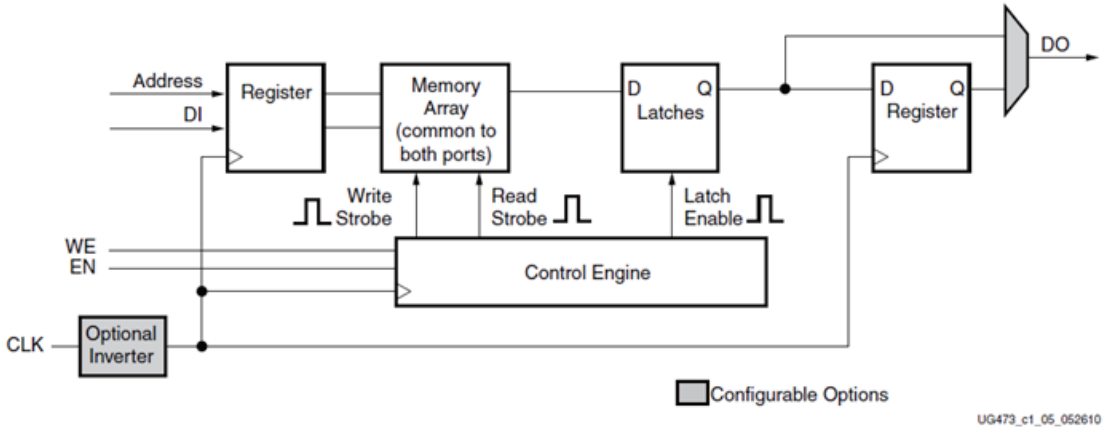

Figure 2.4: Functional structure of a BRAM in 7 series FPGAs [40].

as high flexibility and efficiency. [41]

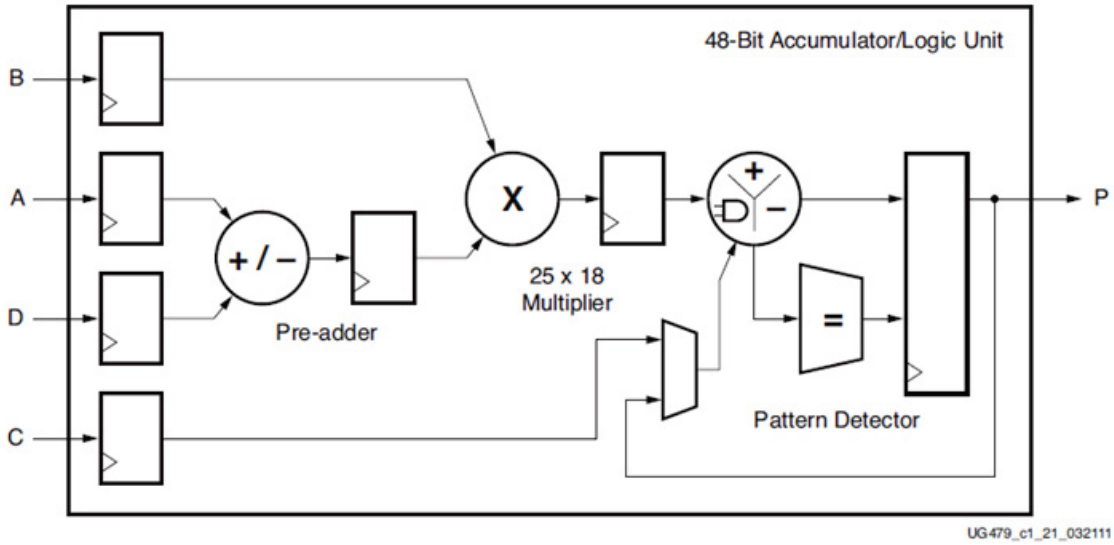

Figure 2.5: Functional structure of a DSP48 slice [41].

2.1.4 Input/Output Block (IOB)

IOBs (Fig. 2.6) reside at the periphery of FPGA chips responsible for interchanging signals with outside world. Similar to CLBs, IOBs is directly connected to the wire segments adjacent to them. Each IOB contains separated input and output buffer as well as three registers. Both the input and output support a wide range of signaling standards. The time delay between the input buffer and the register is programmable. A weak keeper circuit is specially design near the output pad. [42]

2.2 FPGA Design Tools and Design Flow

Modern FPGA is featured by ultra-fast speed and abundant hardware resources. The traditional FPGA design flow which start from the register-transfer level (RTL) shows increasingly ineffective and unreliable. System-level integration design flow, which is centered around intellectual property (IP) cores, shows increasing strength in large scale hardware design. The Xilinx® Vivado® Design Suite, for example, includes powerful and fast-integrated IP integrator environment, IP catalog, which contains abundant IP functions (e.g. math functions) and makes it possible for engineers to configure and package custom IP functions. [43]

At different stages of the design flow, the hardware design verification, including logic simulation, I/O and clock planning, power and timing analysis, design rule checks, visualization of design logic, modification of implementation results, and debugging, can

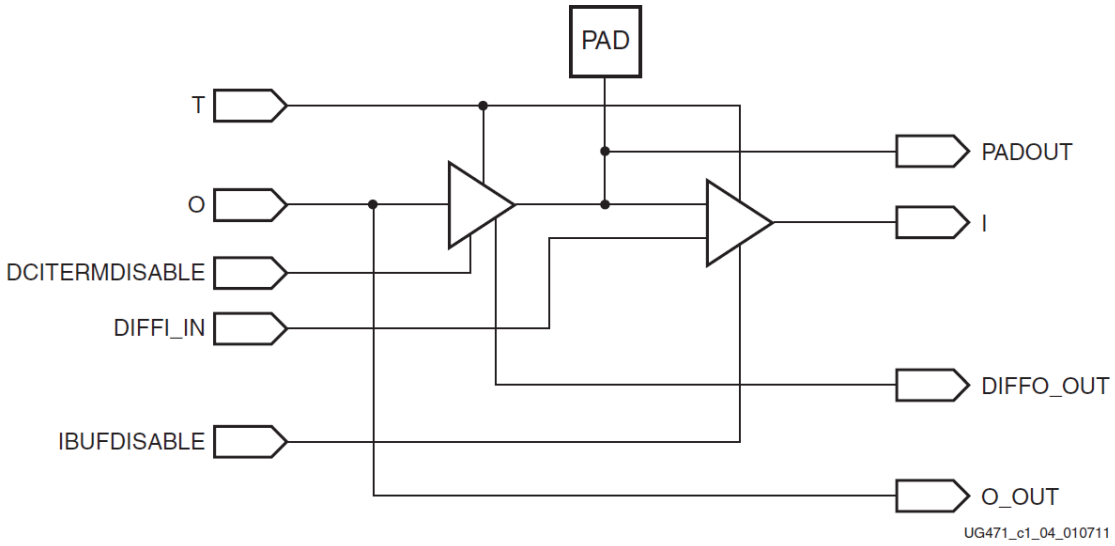

Figure 2.6: Functional structure of the input/output block [42].

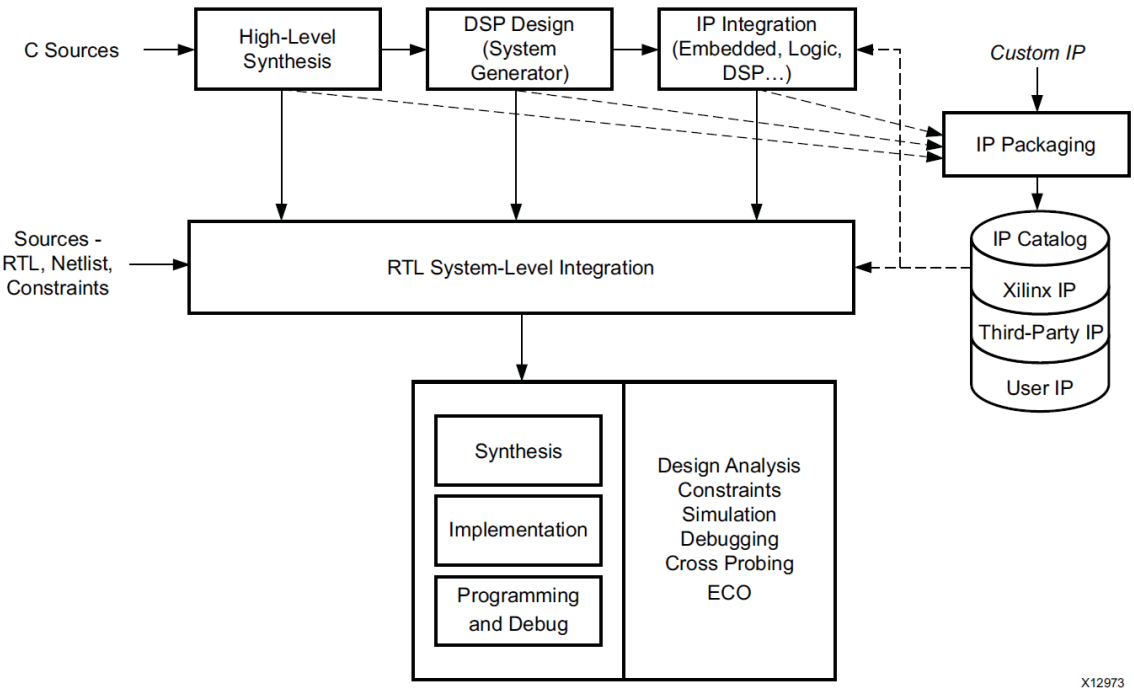

Figure 2.7: Design flow of Vivado® software [43].

be easily performed. The whole FPGA design using Xilinx® Vivado® software contains hardware description language (HDL) entry, synthesis, implementation, programming and debugging. [43]

2.2.1 HDL Design Entry

Vivado® Design Suite supports powerful hardware description language (HDL) to represent the behavior and structure of a digital integrated circuit system. This schematic design entry is capable of describing the system from processor level to gate level. As two most popular HDLs, Very-High-Speed Integrated Circuit Hardware Description Language (VHDL) and Verilog HDL are both supported by Vivado® Design Suite. Both the two HDLs support top-down or bottom-up approach to design. However, modern Vivado® software does not recommend the traditional design flow which start from register-transfer level or gate level. Numerous system-level IP cores with various functions source from all Xilinx® IP, third-party IP, and user-designed IP, have been presented in its IP catalog, which help the engineers free from arduous low-level designs, and focus on the high-level algorithms. [43]

2.2.2 Behavioral Simulation

Behavioral simulation is run to verify the syntax and function of input HDL codes. It only considers high-level functionality instead of the timing constraint of the designed digital system. Most design errors could be found in this type of simulation, while those low-level functions of the design circuits concerning timing information should be verified after synthesis or implementation using different types of simulations. [43]

2.2.3 Synthesis

Synthesis is a translation process for the original behavioral and structural description, entered as HDL codes. The resultant digital system description is in a structural netlist format, which represent the system in a lower-level of abstraction, gate-level representation. The whole synthesis process mainly comprises four stages, which include high-level synthesis, RT-level synthesis, gate-level synthesis, and technology mapping. [43]

2.2.4 Post-Synthesis Functional and Timing Simulations

These two kinds of post-synthesis simulations provide the designers to have better perspectives of the designed system. After running these two simulations, the resultant functional or timing netlists will be generated to provide the designers more accurate information whether the designed system meets the requirements or not. [43]

2.2.5 Implementation

The implementation process is actually a process of place and route, which means to interpret the designed logic to physical units in the hardware and make them properly connected. [43]

2.2.6 Post-Implementation Functional and Timing Simulations

After implementation, Vivado® Design Suite also provide opportunities for the designers to simulate the designed system. At this stage of design process, the generated system is more close to final system to be designed. The designers will given better insights to the functionality and timing issues of the whole system. [43]

2.2.7 Programming and Debugging

After all the aforementioned processes done, the whole design can be converted to a binary bitstream files and downloaded to the FPGA board using JTAG. Even at this stage, the designed system can be debugged using the Vivado® Logic Analyzer (VLA), which is fully integrated into the Vivado® Design Suite. If the designed system is proved meeting the requirements, it whole design processes are completed. [43]

2.3 Summary

With the technological advances in digital hardware design tools, FPGAs have became a ideal platform to develop HIL simulators. Current generation FPGAs endeavor to transcend their previous physical limits and achieve both ultra high running-speed and capacity in terms of hardware resources. Their boosted performance has made it possible to emulate the complex physics-based device-level power electronic circuit. Also, the new FPGA design tools provide abundant well-designed IP cores to upgrade the FPGA design to system-level. Designer today could focus more on the overall algorithms. In this chapter, the basic theory of FPGAs is introduced, the inner architecture of Xilinx® 7 series FPGAs is described, and the design flow of Xilinx® Vivado® software is discussed.

3

Simulation and Interface Techniques of Device-Level Circuit Simulators

Power electronic system modeling in hardware-in-the-loop simulators with simple system-level models such as ideal switching model or averaged model have been performed in recent years. These models are sufficient for observing system-level waveforms and harmonics. However, high-accurate physics-based device-level power electronic system modeling is especially necessary in those applications whose performance affected by high-frequency switching devices. The reason why physics-based power electronic circuit hardware emulation has not performed before is due to the following several obstacles. Firstly, due to the complexity of device nonlinear inner physics, the hardware emulation would encounter high computational burden. Secondly, to modelling the high-fidelity device switching behavior, the time-step must be reduced to a few nanoseconds to capture the high-frequency transients. Thirdly, actual power electronic circuit may involve a great many devices. With the increase numbers of those nonlinear switches, the difficulties to models to whole circuit become escalating.

Device-level circuit simulators have built up a systematically sophisticated group of methods to deal with those obstacles to simulating those physics-based device-level power electronic circuit. They have many useful methods (e.g. variable time-step strategy, various system matrix formulation methods, and alternative nonlinear solution methods), which can surely be referential to HIL simulator designs.

In this chapter, an survey to the techniques of the device-level circuit simulators [1] is presented. Firstly, the solution approaches and their salient features of a list of popular simulators is described. Then the co-simulation interface techniques of selected simulators

[1] This material will be submitted for publication: W. Wang and V. Dinavahi, "Co-simulation interfaces in device-level circuit simulators".

are discussed successively.

Table 3.1: List of Device-Level Circuit Simulators

Simulators	Salient Features
Saber®	A comprehensive mixed-signal simulator, provides a versatile modeling language named MAST, which makes it possible to divide specific models from simulation algorithms; applied to electrical, optical, thermal, mechanical systems [44], [45].
SPICE (Spice2, Spice3)	Simulation Program with Integrated Circuit Emphasis (SPICE), the most popular general-purpose and open source analog circuit simulator [46].
Orcad®	A strong desktop electronic design automation (EDA) software owned by Cadence Design Systems, Inc., which has integrated the widely-used PSpice® as its main subset, as well as a schematic capture package and other tools [47]. As a PC version of SPICE, PSpice® is a dominating industrial standard for circuit and system analysis, works in analog and mixed signal environments, supports the functions for analog behavioral modeling [48]
HSIM®	Designed to meet the requirement of nanometer circuit analysis, able to perform hierarchical simulation, commercial product from Synopsys® [49].
JSPICE	Based on Spice2, designed for superconductor and semiconductor circuits, incorporates the Josephson junction model [50].
MacSpice	A Mac version of SPICE, open source and free for non-commercial use only [51].
XSpice	Based on SPICE, but in further incorporating arbitrary user models [52].
Ngspice	Based on SPICE, CIDER and XSpice; is a mixed level and signal simulator, works on Linux and FreeBSD systems [53].
HSpice®	An analog circuit simulator similar to Spice3 but having better convergence, commercial product from Synopsys® [54].
HomSPICE	A member of SPICE-family circuit simulators, uses three homotopy algorithms: FIXPDF, FIXPNF, and FIXPQF, which is in favor of calculating a circuit's dc operating points and periodic steady-state response [55].
PSIM®	A strong simulation platform for power electronics and motor drive control [56].
PLECS®	A Piece-wise Linear Electrical Circuit Simulator (PLECS®) based on the state-variable formulation working within MATLAB/Simulink® environment and integrating circuits entered into Simulink® as S-functions [57].
XSIM	An efficient crosstalk simulator, based on indigenous methodology in Visual C++, provides user-friendly graphical user interface [58].

Simulators	Salient Features
Multisim™	A updated version of SPICE; software simulation kit provides dynamic simulation models, with powerful interactivity; has powerful Design-Rules-Check and Connectivity Check with the breadboard tool [46], [59].
QUCS	Quite Universal Circuit Simulator (QUCS), an integrated circuit simulator with a graphical user interface [60].
PECS	Power Electronics Circuit Simulator (PECS), excelling in time-domain simulation of switched networks with non-linearity; achieves both high speed and accuracy as a result of several new algorithms including a state space method for nonlinear analysis of switched networks [61].
TITAN	A complete customizable simulator, which provides the freedom to divide the whole circuit into arbitrary sub-circuits; circuit equations are solved by its special vectorized solver [62].
PETS	Power Electronics Transient Simulator (PETS), used for time-domain analysis, provides freedom to choose different degree of complexity for piecewise-linear models, supports continuously differentiable nonlinear models by applying a delay approximation method with Newton-Raphson iteration [63].
Spectre®	An improved SPICE-like analog simulator; commercial product from Cadence® [64].
MEDUSA	A user-oriented simulator for modular circuits, satisfies the needs for device and circuit simulations at the same time [65].
CODECS	A mixed-level circuit simulator, based on Spice3, while incorporates a set of numerical models under its main structure without changing its Spice3 core [66].
AWEswit	A mixed analog and digital circuit simulator, special for the switched capacitor circuits, unique in its asymptotic waveform evaluation (AWE) technique in modeling and simulation, provides flexibility in circuit formulations [67].
DesignLab	A Windows version of PSpice®, developed in the form of web pages with multimedia effect for analysis and design of circuits and electronics [68].
Eldo™	An analog, digital and mixed circuit simulator with a VHDL-based Analog Hardware Language [69].
IsSPICE4	An improved version from Spice3f5 and XSpice, adding some strong interactive features and extensions [70].
Gnucap	A general purpose mixed analog and digital circuit simulator, fully interactive, compatible to SPICE, containing a simple behavioral modeling language [71].
LTSpice	A high performance simulator, making many enhancements based on traditional SPICE simulator [72].
TINA-TI™	A user-friendly, powerful circuit simulator based on a version of SPICE [73].

3.1 Solution Approach in Device-Level Circuit Simulators

The enumeration of device-level circuit simulators is given in Table 3.1 with their salient features. This will be helpful to compare their particular capabilities. Notably, these simulators can be categorized into free and/or open source (e.g. SPICE, Ngspice and QUCS), commercial (e.g. PSpice®, Saber® and PSIM®), and research-oriented simulators (e.g. MEDUSA and CODECS). Furthermore, a group of simulators, including SPICE, Spice2, Spice3, Ngspice, PSpice®, HomSPICE, MultiSim™and so on can be categorized as SPICE-like simulators, which are essentially based on the kernel of SPICE [55], [46], [74], and their transient analysis operations are basically the same.

In this section, the solution approach for transient analysis used by a majority of device-level simulators is described.

The first step is the system matrix formulation. The modified nodal analysis (MNA) is taken by Saber® and SPICE-like simulators, where the system is represented by a group of nonlinear first-order differential algebraic equations.

$$N(\mathbf{x}(t), \frac{d\mathbf{x}(t)}{dt}, t) = \mathbf{0} \tag{3.1}$$

where $\mathbf{x}(t)$ is the vector of unknown circuit variables, and $N(\cdot)$ is a nonlinear vector function [75], [76].

In contrast, other circuit simulators such as PECS, utilize a state space approach to undertake the system matrix formulation step [61].

The aforementioned nonlinear equation group contain nonlinear ordinary differential equations (ODEs) of the form [77]

$$\frac{d\mathbf{x}(t)}{dt} = o(\mathbf{x}(t), t) \tag{3.2}$$

To solve (2) at the next time step t_{n+1}, numerical integration is applied. For instance, SPICE-like simulators and Saber® use Gear or Trapezoidal methods.

For the Trapezoidal method,

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}[o(\mathbf{x}_{n+1}, t_{n+1}) + o(\mathbf{x}_n, t_n)] \tag{3.3}$$

where $\mathbf{x}_{n+1}$ is the solution of next time step t_{n+1}, $\mathbf{x}_n$ is the solution of current time step t_n [77].

For the second-order Gear method (the default method for Saber®),

$$\mathbf{x}_{n+1} = \frac{4}{3}\mathbf{x}_n - \frac{1}{3}\mathbf{x}_{n-1} + \frac{2}{3}h \cdot o(\mathbf{x}_{n+1}, t_{n+1}) \tag{3.4}$$

where $\mathbf{x}_{n+1}$ is the solution of next time step t_{n+1}, $\mathbf{x}_n$ is the solution of current time step t_n. Thus (2) is successfully transformed into nonlinear algebraic equations (NAEs), which have the general form of

$$F(\mathbf{x}) = \mathbf{0} \tag{3.5}$$

where $\mathbf{x} \equiv \mathbf{x}_{n+1}$, $F(\cdot)$ is the general nonlinear operator [76], [75].

To solve these NAEs, (5) needs to be linearized in this step, SPICE-like simulators and Saber® use Newton-Raphson or Katzenelson algorithms. Equations (5) can be further represented as

$$F(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^T \tag{3.6}$$

where $f_1(\cdot), f_2(\cdot), \ldots, f_m(\cdot)$ are all nonlinear operators [75].

For Newton-Raphson algorithm, the Jacobian matrix must be formulated, which is given as

$$J(\mathbf{x}_k) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}\mid_{\mathbf{x}_k} & \frac{\partial f_1}{\partial x_2}\mid_{\mathbf{x}_k} & \cdots & \frac{\partial f_1}{\partial x_m}\mid_{\mathbf{x}_k} \\ \vdots & & & \vdots \\ \frac{\partial f_m}{\partial x_1}\mid_{\mathbf{x}_k} & \frac{\partial f_m}{\partial x_2}\mid_{\mathbf{x}_k} & \cdots & \frac{\partial f_m}{\partial x_m}\mid_{\mathbf{x}_k} \end{pmatrix} \tag{3.7}$$

where $\mathbf{x}_k$ is the solution at the kth iterate [75]. Then, a linearized system of equations is obtained as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - J(\mathbf{x}_k)^{-1} F(\mathbf{x}_k) \tag{3.8}$$

where $\mathbf{x}_{k+1}$ is the solution at the $(k+1)$th iteration [75].

Notably, when evaluating the Jacobian matrix, Saber® applies a simplicial subdivision technique to calculate the first derivatives thereby reducing the computational burden at every iteration, which is different from SPICE [44].

When applying the Newton-Raphson method, the SPICE-like simulators apply prespecified tolerances (e.g. *ABSTOL*, *RELTOL*, and *CHGTOL*) to determine convergence to a valid solution. In contrast, Saber® uses no tolerance to determine convergence, since the system of equations are evaluated piecewise linearly and solved exactly [44]. The CODECS simulator, however, use a modified two-level Newton algorithm [66] in this step. Different from Newton-Raphson, the Katzenelson algorithm is based on piecewise linear systems. Sample points are used to find the linear regions for every nonlinear device [77]. Finally, in order to solve the linear algebraic equations (8), SPICE-like simulators and Saber® use the methods of Gaussian elimination or LU decomposition techniques with forward and backward substitutions [75], [76], whilst the TITAN simulator uses a vectorized solver method [62].

The aforementioned transient analysis operations (taking Saber® as an example) can be illustrated by the flowchart in Fig. 3.1.

Other circuit simulators for formulating the Jacobian matrix mostly utilize the standard techniques mentioned above, but they also use new methods. For example, PETS uses a novel algorithm to decide the accurate element states of its piecewise-linear networks as

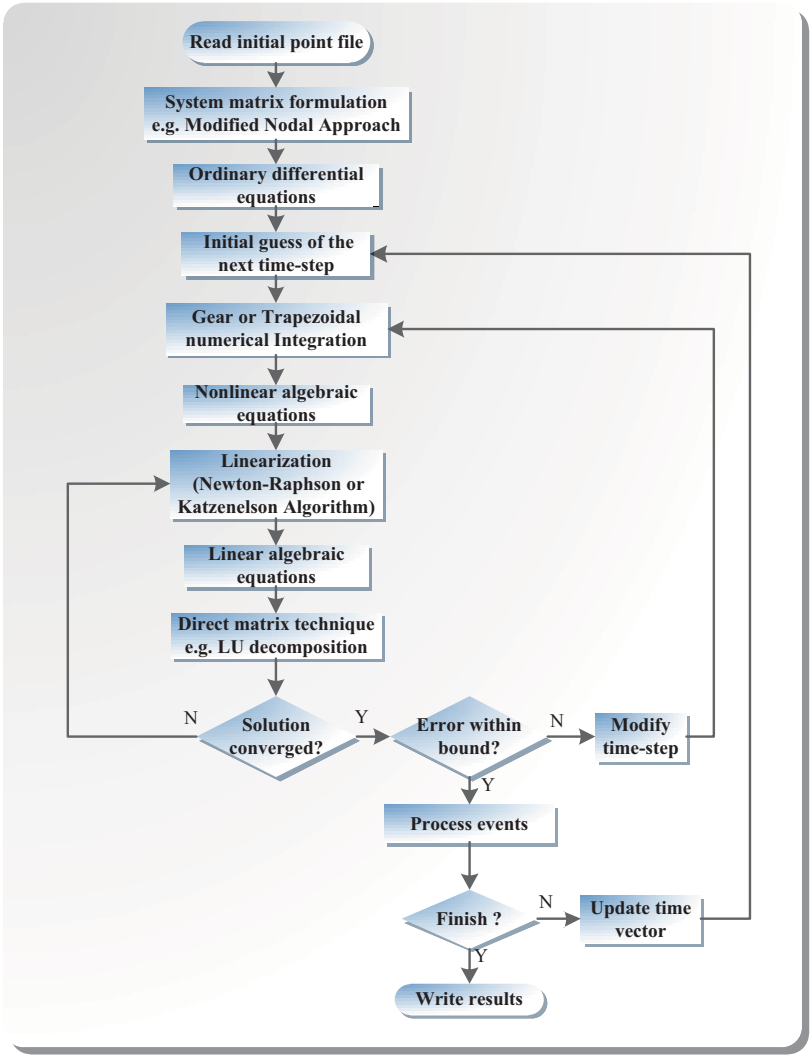

Figure 3.1: Flowchart of transient analysis operation [77].

well as an efficient way to avoid its piecewise-linear and reactive elements from changing values with each time step, thereby keeping the system matrix to be constant. [63].

3.2 Co-Simulation of Device-Level Circuit Simulators and System-Level Simulators

In this section, co-simulation examples for device-level circuit simulators are discussed.

3.2.1 Saber® and MATLAB/Simulink®

Saber® is a device-level circuit simulator which specializes in power electronic simulation, while as is mentioned above, Simulink® is versatile in building control systems. Like other multi-domain designs, the co-simulation between Saber® and MATLAB/Simulink® can

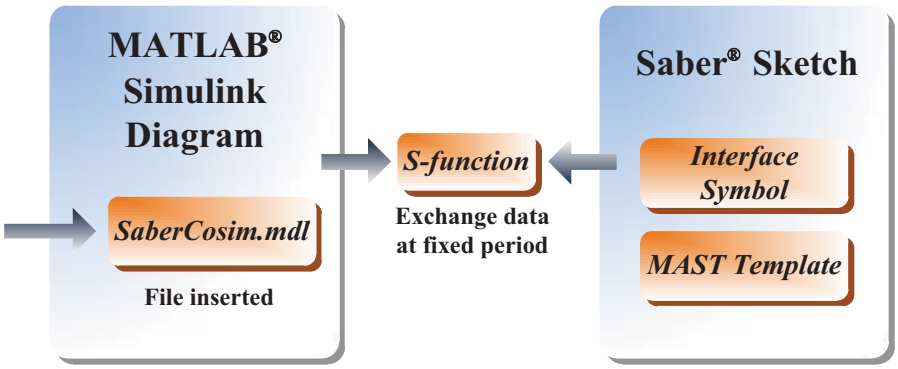

Figure 3.2: Illustration of Saber® Simulink® co-simulation interface [78].

be very effective in many circumstances. The procedure of using the Saber®-Simulink® co-simulation tool is available in [78]. The principle of Saber®-Simulink® interfacing is illustrated in Fig. 3.2.

Notably, the running processes of the two simulators are fully independent except that they need to exchange data at fixed period of intervals. This communication mechanism is realized by using an S-function. Additionally, a Saber® Co-simulation block (SaberCosim.mdl) should be integrated into the Simulink® model, which is then imported in to Saber® interface using Saber®-Simulink® co-simulation tool. This tool is responsible for producing the required co-simulation interface symbol and the MAST template. This expressive user interface makes it possible for the co-simulation to run totally in Saber® interface.

Reference [79] presented a method to construct a high-voltage source circuit system by the hybrid modeling of Saber® and Simulink®. In their designs, the Saber® software was responsible for the power electronic circuit, while Simulink® was in charge of building a fuzzy PID controller. The results showed that the co-simulation of Saber® and Simulink® was a highly efficient way to analyze the whole system.

Similarly, [80] applied the Saber® and Simulink® co-simulation for the modeling of a high pulse power circuit. In their studies, the physical model of Reversely Switched Dynistor (RSD) was constructed using Simulink®, and Saber® was used to model the pulse power circuit and the magnetic switch.

3.2.2 PSpice® and MATLAB/Simulink®

PSpice® is a simulation tool which can tackle models in analog and mixed-signal environments, whereas MATLAB/Simulink®, as a platform for multi-domain simulation, is mainly based on approximate continuous-time and discrete-time models of dynamic systems. Obviously, they both have advantages in a simulation project, since the former can make it possible for designers to perform simulation which includes realistic electrical models, while the latter is mainly focused on building the whole system.

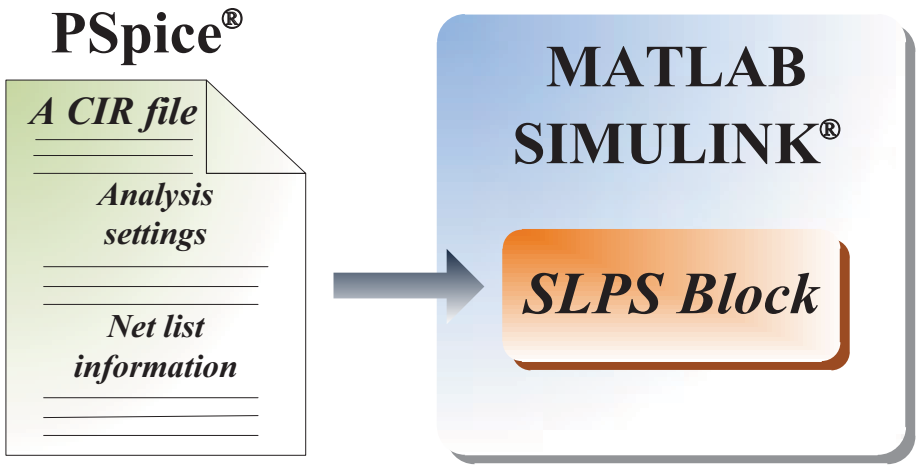

Figure 3.3: Illustration of PSpice® Simulink® co-simulation interface [81].

The co-simulation between MATLAB/Simulink® and PSpice® is realized by an interface tool named the PSpice® SLPS Interface, which enables electro-mechanical system designers to perform system-level simulation which include specific device-level circuit simulation. More specifically, the PSpice® SLPS Interface makes it possible for the designer to include realistic electrical PSpice® models of actual components when performing system-level simulation. The detailed procedure to use the SLPS interface is available in [81] and the co-simulation is illustrated in Fig. 3.3.

Notably, the co-simulation of PSpice® and Simulink® is initialized by creating a CIR file, which specifies the PSpice® analysis settings (e.g. the analysis time) and the net list information, thereby assigning the circuit built in PSpice® to the SLPS block, which, in turn, is integrated into Simulink® models [81].

The co-simulation is dominated by Simulink®, which exchanges data with PSpice® at its own time step. As a result, if the simulation step-size of Simulink® is set too large, then the results of the simulation will be incorrect.

Because of the obvious merits of the co-simulation between PSpice® and Simulink®, the PSpice® SLPS Interface has been used in many electro-mechanical designs. These applications include the design presented in [82], which successfully simulates pipeline ADC circuits and obtains satisfactory results. Also, [83] presents a method for the simulation of solar photovoltaic (PV) cell by applying the PSpice® and Simulink® co-simulation interface; it builds a hybrid model of the PV module using PSpice® and Simulink®. Additionally, Ref. [84] presents a co-simulation solution for a high efficiency full-bridge DC-DC converter for fuel cell; because the Simulink® has merits in building the feedback controller of the converter while PSpice® excels in modeling the electronic circuits.

3.2.3 PSIM® and MATLAB/Simulink®

Reference [86] introduced the SimCoupler Module by which the co-simulation between PSIM® and MATLAB/Simulink® was made possible and provided detailed procedures to use this module.

As a device-level circuit simulator which has special advantages in power electronics simulation, PSIM® has disadvantages to build a control subsystem at the same time. However, MATLAB/Simulink® is good at constructing control circuits. Therefore, the co-simulation between PSIM® and Simulink® has its unique value to be studied and applied in many circumstances. For example, [87] built a simulation platform for a three-level adjustable speed drive. In view of the merits of PSIM®, the designer built the main circuit of the three-level adjustable speed drive using PSIM®, whereas, they constructed the control system, observed the output voltage and performed Fourier analysis in MATLAB/Simulink®.

Another case for the application of the co-simulation between PSIM® and Simulink® was presented in [88], where the designer first compared PSIM® and Simulink® separately in simulating single-phase uncontrolled rectifier and three-phase controlled rectifier. After discussing of the pros and cons of both simulators, they concluded that the co-simulation between them was better solution.

3.2.4 SPICE and MATLAB/Simulink®

Reference [85] introduces a co-simulation interface between Simulink® and SPICE, which is realized by a new Simulink® block SLSP. The SLSP block is written in C MEX S-function, which is responsible for reading in the circuit file, initializing the simulation, performing the time-domain numerical integration and manipulating the SPICE-Simulink® communication.

The co-simulation setting is simple to build up as an SLSP block in the MATLAB/Simulink® environment with a parameter for the name of a SPICE circuit file. This mechanism is described in Fig. 3.4.

Reference [85] also elaborates an application to use Simulink®-SPICE Interface to simulate a speed control system of a dc motor. Specifically, the whole system is modeled in the Simulink® environment, except for the PI controller, the machine is modeled in SPICE.

3.2.5 PLECS® and MATLAB/Simulink®

PLECS®, is a piece-wise linear electrical simulator that enters the circuit information as netlists, which are in turn integrated into MATLAB/Simulink® using S-functions. Compared to the Power System Blockset models of Simulink®, the PLECS® improves the performance greatly using the ideal switch models [57].

Reference [89] applies Simulink® and PLECS® co-simulation in a photovoltaic energy

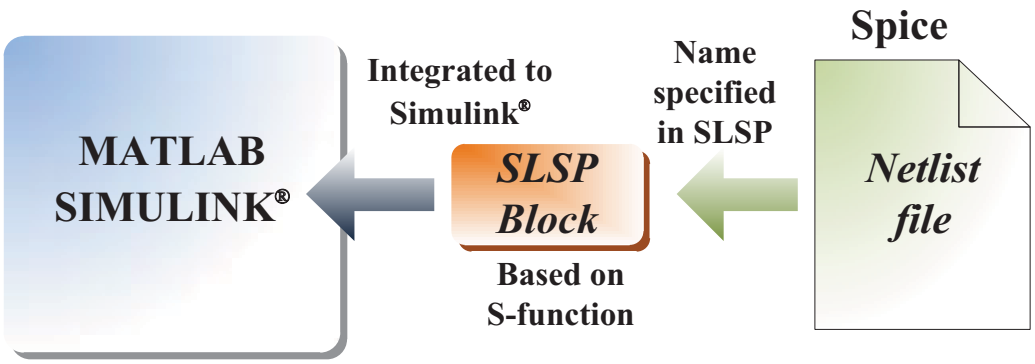

Figure 3.4: Illustration of Simulink®-SPICE Interface [85].

conversion system. The control subsystem is modeled using Simulink®, whilst the plant subsystem, including DC supply, inverter, LCL filter and utility grid, is modeled using PLECS®. The simulation results show that the co-simulation between Simulink® and PLECS® is much faster than using Simulink® transfer functions.

3.3 Analog and Digital Co-Simulation of Circuit Simulators

Although co-simulation with some system-level simulators may be effective in handling large and complex simulation tasks, for those applications such as power and mechatronic system simulations, which may include a great many analog and digital subsystems, novel co-simulation platforms have become necessary [90]. For example, simulators such as Saber® work in analog and mixed-signal environments, whereas many of the control subsystems are performed in digital logic. Therefore, co-simulations with other digital simulators such as ModelSim® become good solutions [91]. In this section, several interfacing instances for analog and digital co-simulation of circuit simulators are discussed.

3.3.1 Saber® and ModelSim®

According to [91], the Saber®/ModelSim® co-simulation interface enables Saber®, an analog and mixed-signal simulator, to support VHDL modeling. More specifically, a designer can model the analog part by using Saber® MAST and the digital part in VHDL using ModelSim® VHDL or in VHDL/Verilog using ModelSim Plus®. The mechanism behind the Saber®/ModelSim® co-simulation is illustrated in Fig. 3.5.

The hypermodels shown in Fig. 3.5 that interface the digital pins from the models built in ModelSim® with the analog device models in Saber® are included in a library of more than 3,500 models, which guarantee the co-simulation to be accurate [91].

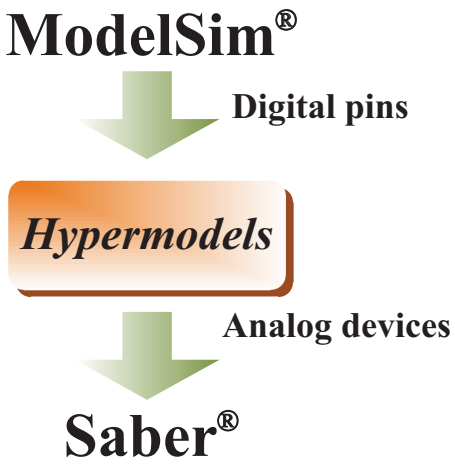

Figure 3.5: Illustration of Saber®/ModelSim® co-simulation interface [91].

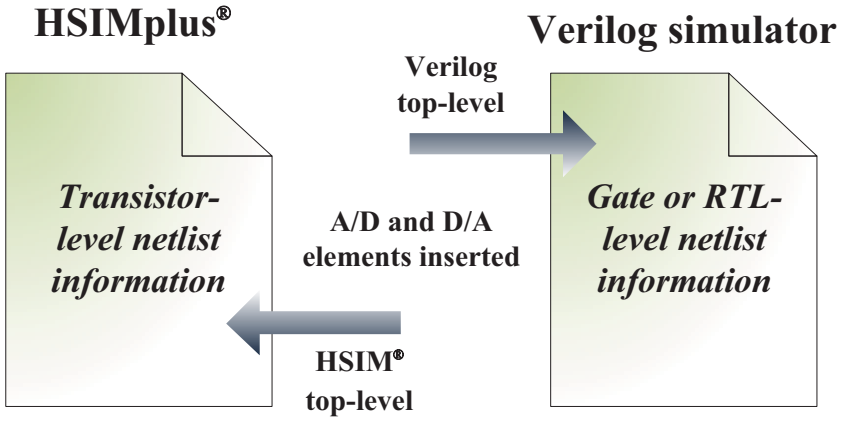

Figure 3.6: Illustration of HSIMplus® HDL co-simulation interface [90].

3.3.2 HSIMplus® and HDL

Reference [90] introduces the co-simulation between HSIMplus® and HDL, which connects digital and analog subsystems. Specifically, the HSIMplus® supports interface to several different Verilog simulators such as Synopsys® VCS, Cadence® NC-Verilog, and Mentor Graphics® ModelSim®. The co-simulation of HSIMplus® and ModelSim® can be described as in Fig. 3.6.

The co-simulation can facilitate full integration of circuit information using different formats. Furthermore, the designers can decide which simulator is the dominant one for specific cases. [90]

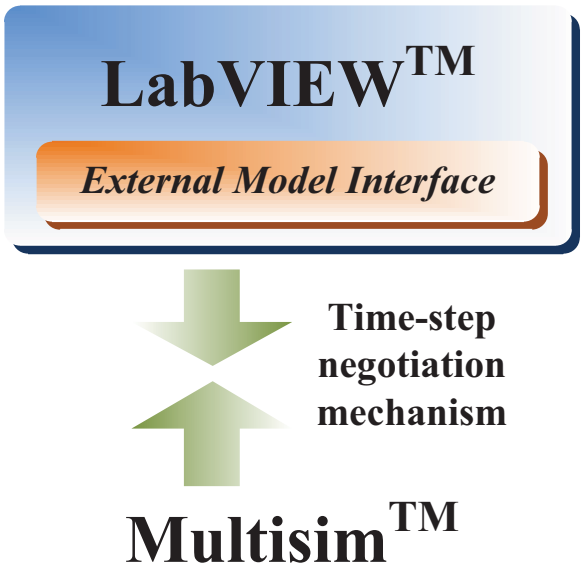

Figure 3.7: Illustration of Multisim™LabVIEW™co-simulation interface [92].

3.3.3 Multisim™and LabVIEW™

Reference [92] introduces the co-simulation between Multisim™and LabVIEW™. The advantage of this co-simulation is that it integrates the analog and digital system in an effective way, because the Multisim™is excellent in modeling analog and mixed-signal circuitry whilst the LabVIEW™is prominent in implementing the control logic.

As illustrated in Fig. 3.7, the data exchange between the two simulators uses a variable time-step mechanism, which is decided by the conditions of both simulators. An External Model Interface in LabVIEW™is responsible for all the data exchanged. [92] Also, [92] describes the detailed procedures on how to initiate and perform the co-simulation with an example.

3.4 Co-Simulation of Circuit Simulators with Programming Languages

The limitations of circuit simulators may sometimes require them to perform co-simulation with programming languages. For example, SystemC™(essentially C++) is a good option for multi-domain applications because of its advantages to undertake high levels of abstraction when modeling systems. As a programming language, it can act as a simulator as well as connect and coordinate element between other simulators [93], [94]. A discussion of the co-simulation between circuit simulators and programming languages is of significance because traditional circuit simulators have several modeling limitations.

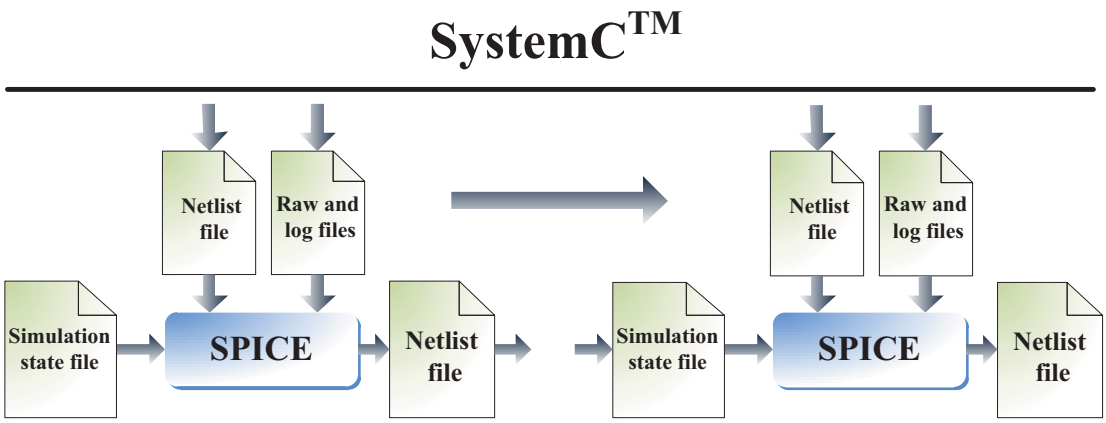

Figure 3.8: Illustration of SPICE and SystemC™co-simulation interface [93].

3.4.1 PSpice® and C++

Although the co-simulation of PSpice® and Simulink® can be very effective to tackle many demanding designs, high level languages such as C++ sometimes can be applied to compile and store the external data for PSpice® in order to simplify the modeling and reduce the iteration times.

For example, [95] applies the co-simulation technique of PSpice® with C++ in the application of a three-phase PWM power converter. Specifically, the designers produce the PWM waveforms using C++ programming by interfacing the micro-controller with PSpice®. The output data of PWM pulses is stored in memory, which is in further integrated into PSpice® listing programs for the whole circuits. At last, PSpice®'s graphic processor, named Probe, works as oscilloscope to display the output waveforms.

3.4.2 SPICE and SystemC™

Reference [93] introduced a SPICE and SystemC™co-simulation method which is realized on the concept of loose simulator coupling. In practice, SystemC™acts as the master simulator, keeping in charge of the whole simulation period, while SPICE acts as an integral part of SystemC™, exchanging information with the master simulator in a loosely synchronized manner.

SystemC™is active and dominant throughout the cycle of simulation, whilst SPICE, as a slave simulator, gets called by SystemC™until exiting, when its current state is saved to a file that is used to resume the simulation when it is called next time. The mechanism of SPICE and SystemC™co-simulation can be illustrated in Fig. 3.8.

As can be seen from Fig. 3.8, the information exchange between SystemC™and SPICE is realized by several simulation source files. A specially designed wrapper program is responsible for the interfacing with SPICE, turning SystemC™commands into the netlist file before sending to SPICE, and scanning the events and errors from SPICE.

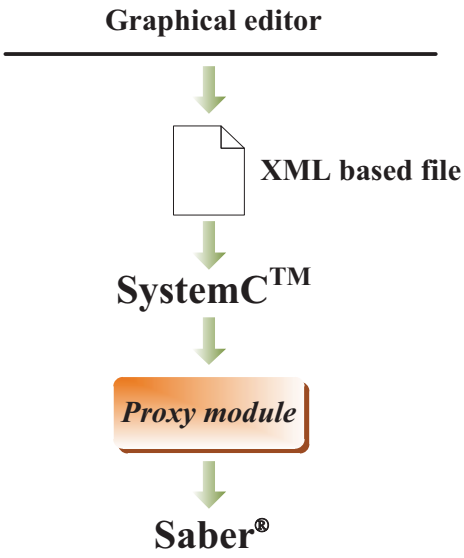

Figure 3.9: Illustration of Saber® and SystemC™co-simulation interface [94].

3.4.3 Saber® and SystemC™

A successful example for the co-simulation of Saber® and SystemC™has been mentioned in [94], where the co-simulation was implemented by a proxy module responsible for transferring SystemC™signals to Saber®. A special synchronization method was applied to tackle the difference between analog and digital simulators. Furthermore, a graphical editor had been made it easy for the mixed signal design by Saber® and SystemC™ [94]. The co-simulation interface of Saber® and SystemC™is shown in Fig. 3.9. Specifically, the graphical editor is capable of exporting the design to SystemC™via a XML based file, which contains the structural information (e.g. components, interfaces and interconnections). [94]

3.5 Summary

In this chapter, a group of popular device-level circuit simulators are listed and compared with their specialties. It is worth mentioning that many of these techniques of circuit simulators including the time-step control algorithms and solution approaches are also adaptable to physics-based device-level power electronic circuit hardware emulation. Also, many of the circuit simulators have been developed with versatile co-simulation interfaces to meet the higher demands from the simulation tasks today. With these co-simulation interfaces, the researchers today can perform very large simulation tasks which can achieve desirable accuracy with satisfactory speed. The transient information of specific part of the sub-circuits can also be collected and analyzed. It is possible that some of the aforementioned co-simulation algorithms could be utilized by HIL simulators in the near future.

4

Power Electronic Circuit Hardware Emulation

Accurate models of power electronic devices are necessary for HIL simulators. In this chapter, a digital hardware emulation of device-level models for the IGBT and the power diode on FPGA [1] is presented. The hardware emulation utilizes detailed physics-based nonlinear models for these devices, and features a fully paralleled implementation using an accurate floating-point data representation in VHDL language. First, the discrete-time linearized hardware modules of the main switching devices, power diode and IGBT, are detailed for their design techniques. Then the whole power converter circuit hardware emulation is described.

4.1 Power Diode Module

This section provides a brief introduction to the physical power diode model and its hardware emulation in the FPGA. Two of the most important power diode semiconductor behaviour of power diode are its current reverse recovery and voltage forward recovery phenomena, which are essential in transient analysis and switching power loss estimation. In [29], [30], two kinds of simple power diode models are introduced, which are both modelled using lumped-charge approach. Also, their parameter set are very similar to that of Saber®.

[1] Material of this chapter has been published: W. Wang, Z. Shen, and V. Dinavahi, "Physics-based device-level power electronic circuit hardware emulation on FPGA", *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2166-2179, Nov. 2014.

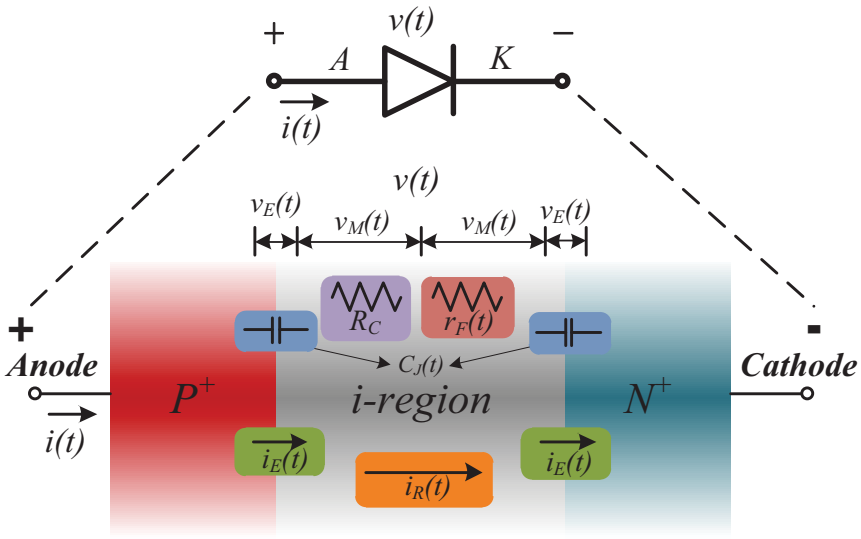

Figure 4.1: Physical structure of power p-i-n diode.

4.1.1 Model formulation

In contrast to the normal p-n diode, the p-i-n structure has a wide intrinsic region between the p-layer and n-layer, which enables it to work in fast switching and high voltage operations, standard for most power diodes [96]. Other type of diodes such as the Schottky diode only work in specific scenarios [97]. Therefore, for power converter circuit hardware emulation, a detailed model for p-i-n power diode is desirable. The physics-based p-i-n diode model contains the following five major phenomenological characteristics: a) reverse recovery, b) forward recovery, c) emitter recombination, d) junction capacitance, and e) contact resistance (see Fig. 4.1). This model is based on the lumped charge concept [29], [30], utilizes transport equations for semiconductor to represent the diode physics. The reverse recovery phenomenon is represented by the following equations:

$$i_R(t) = \frac{q_E(t) - q_M(t)}{T_M}, \tag{4.1}$$

$$0 = \frac{dq_M(t)}{dt} + \frac{q_M(t)}{\tau} - \frac{q_E(t) - q_M(t)}{T_M}, \tag{4.2}$$

$$q_E(t) = I_S \tau [e^{\frac{v_E(t)}{V_T}} - 1], \tag{4.3}$$

where $i_R(t)$ is the diffusion current from the center of the intrinsic region; $v_E(t)$ is the junction voltage; $q_E(t)$ is the junction charge variable; $q_M(t)$ is the charge in the middle of intrinsic region; T_M is the diffusion transit time; τ is the carrier lifetime; I_S is the diode saturation current constant; and the constant V_T is the thermal voltage.
The forward recovery phenomenon is described by

$$v_M(t) = \frac{V_T T_M R_{M0} i(t)}{q_M(t) R_{M0} + V_T T_M}, \tag{4.4}$$

where $i(t)$ represents the whole diode current, $v_M(t)$ is voltage across half of the intrinsic region, and R_{M0} is the initial resistance in the intrinsic region.
The emitter recombination phenomenon is formulated as

$$i_E(t) = I_{SE}[e^{\frac{2v_E(t)}{V_T}} - 1], \tag{4.5}$$

where $i_E(t)$ is the end region recombination current, and I_{SE} is the emitter saturation current constant.
The contact resistance R_C is modeled by

$$v(t) = 2v_M(t) + 2v_E(t) + R_C \cdot i(t), \tag{4.6}$$

where $v(t)$ is the voltage across the diode.
Finally, the charge stored in the junction capacitance $q_J(t)$ which contributes to the whole diode current $i(t)$ is described by

$$i(t) = i_R(t) + i_E(t) + \frac{dq_J(t)}{dt}. \tag{4.7}$$

The junction capacitance $C_J(t)$ and its charge $q_J(t)$ are determined by

$$C_J(t) = \begin{cases} \frac{C_{J0}}{[1-\frac{2v_E(t)}{V_J}]^m} & v_E < \frac{V_J}{4} \\ \frac{m \cdot C_{J0} \cdot 2v_E(t)}{V_J \cdot 0.5^{m+1}} - (m-1)\frac{C_{J0}}{0.5^m} & v_E \geq \frac{V_J}{4} \end{cases} \tag{4.8}$$

$$q_J(t) = \int C_J(t) d[2v_E(t)], \tag{4.9}$$

where C_{J0} is the zero-biased junction capacitance, V_J is the junction built-in potential, and m is the junction grading coefficient.

4.1.2 Model Discretization and Linearization

As seen from above, the power diode model described by (4.1)-(4.9) is nonlinear and time-varying. Here we discretize and linearize the model to obtain a discrete-time equivalent circuit for time-domain simulation. The charge $q_M(t)$ differential equation (4.2) is first discretized according to the Trapezoidal rule:

$$q_M(t) = \frac{\Delta t}{2T_M(1 + \frac{k_1 \Delta t}{2})} \cdot q_E(t) + \frac{q_{hist}(t - \Delta t)}{1 + \frac{k_1 \Delta t}{2}}, \tag{4.10}$$

where

$$q_{hist}(t) = q_M(t)(1 - \frac{\Delta t}{2} \cdot k_1) + \frac{\Delta t}{2T_M} \cdot q_E(t), \tag{4.11}$$

and

$$k_1 = \frac{1}{\tau} + \frac{1}{T_M}. \tag{4.12}$$

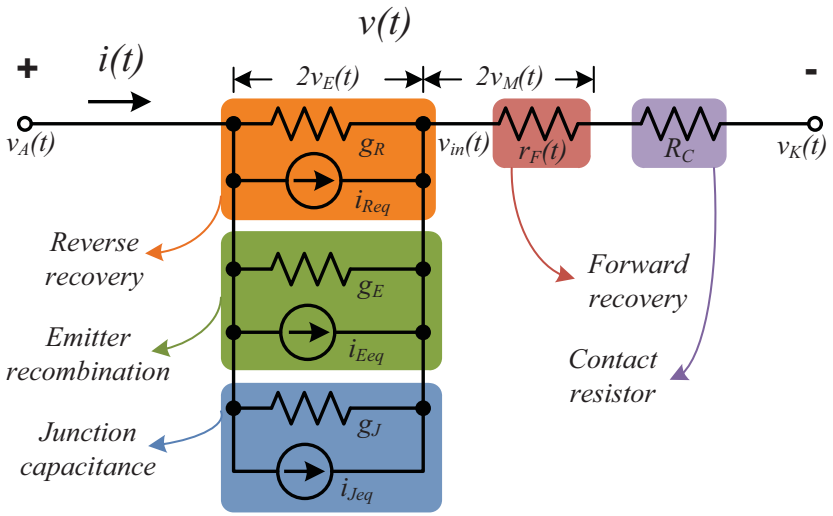

Figure 4.2: Discrete-time linearized equivalent circuit for the power diode.

Similarly, the differential term $\frac{dq_J(t)}{dt}$ (represented by $i_J(t)$, the current contributed by junction capacitor) in (4.7) is discretized as:

$$i_J(t) = \frac{2}{\Delta t} \cdot q_J(t) - \frac{2}{\Delta t} \cdot q_J(t - \Delta t) - i_J(t - \Delta t), \tag{4.13}$$

where the junction capacitor charge $q_J(t)$ can be deduced from

$$q_J(t) = \begin{cases} \frac{-V_J C_{J0}}{1-m} \cdot [1 - \frac{2v_E(t)}{V_J}]^{(1-m)} & v_E(t) < \frac{V_J}{4} \\ \frac{2^{m+2} m C_{J0} v_E^2(t)}{V_J} - & \\ \quad 2^{m+1}(m-1) C_{J0} v_E(t) & v_E(t) \geq \frac{V_J}{4} \end{cases} \tag{4.14}$$

Meanwhile, the nonlinearity of the power diode model is obvious from (4.3), (4.4), (4.5) and (4.8). The nonlinearity coming from the reverse recovery characteristics (4.3) is linearized by a dynamic conductance g_R and a parallel current source i_{Req} given as:

$$g_R = \frac{\partial i_R}{\partial(2v_E)} = k_2 I_S \tau \cdot \frac{1}{2V_T} \cdot e^{\frac{v_E(t)}{V_T}}, \tag{4.15}$$

$$i_{Req} = k_2 I_S \tau [e^{\frac{v_E(t)}{V_T}} - 1] - k_3 q_{hist}(t - \Delta t) - g_R \cdot 2v_E(t), \tag{4.16}$$

where

$$k_2 = \frac{1}{T_M} - \frac{\Delta t}{2T_M^2(1 + \frac{\Delta t k_1}{2})}, \tag{4.17}$$

$$k_3 = \frac{1}{T_M(1 + \frac{\Delta t k_1}{2})}. \tag{4.18}$$

Similarly, the emitter recombination (4.5) and the junction capacitance (4.8) nonlinearities are linearized to obtain the corresponding conductance and current source pairs (g_E, i_{Eeq}) and (g_J, i_{Jeq}) respectively:

$$g_E = \frac{\partial i_E}{\partial(2v_E)} = \frac{I_{SE}}{V_T} e^{\frac{2v_E(t)}{V_T}}, \tag{4.19}$$

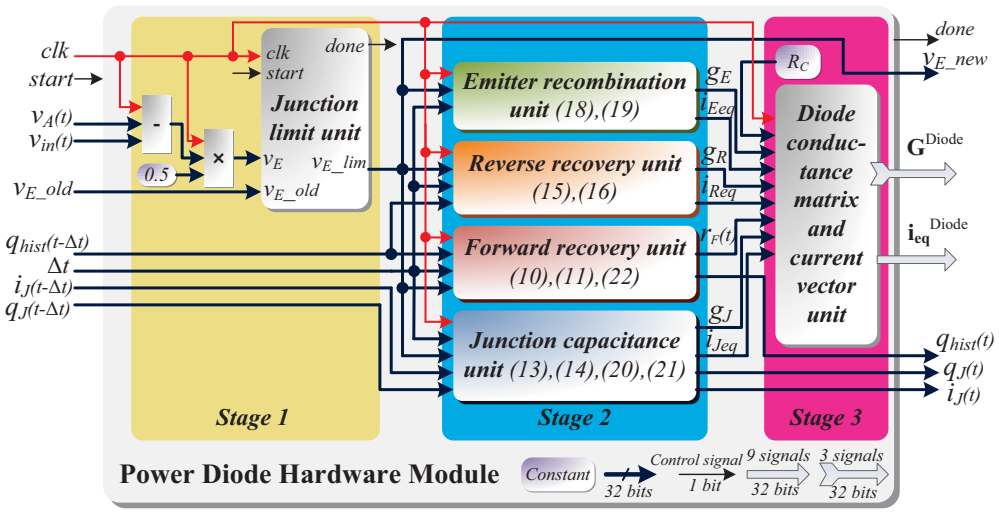

Figure 4.3: Architecture of power diode hardware module in FPGA.

$$i_{Eeq} = I_{SE}[e^{\frac{2v_E(t)}{V_T}} - 1] - g_E \cdot 2v_E(t), \tag{4.20}$$

$$g_J = \frac{\partial i_J}{\partial(2v_E)} = \frac{2}{\Delta t} C_J(t), \tag{4.21}$$

and

$$i_{Jeq} = i_J(t) - g_J \cdot 2v_E(t). \tag{4.22}$$

The forward recovery effect (4.4) is represented as a time-varying resistor:

$$r_F(t) = \frac{2v_M(t)}{i(t)} = \frac{2V_T T_M R_{M0}}{q_M(t)R_{M0} + V_T T_M}. \tag{4.23}$$

Finally, the complete discretized and linearized power diode (see Fig. 4.2) can be written in the form:

$$\mathbf{G}^{Diode} \cdot \mathbf{v}^{Diode} = \mathbf{i}_{eq}^{Diode}, \tag{4.24}$$

where $\mathbf{G}^{Diode}$ is a 3×3 conductance matrix given as:

$$\begin{bmatrix} g_R + g_E + g_J & -g_R - g_E - g_J & 0 \\ -g_R - g_E - g_J & g_R + g_E + g_J + \frac{1}{r_F(t)+R_C} & -\frac{1}{r_F(t)+R_C} \\ 0 & -\frac{1}{r_F(t)+R_C} & \frac{1}{r_F(t)+R_C} \end{bmatrix}, \tag{4.25}$$

and $\mathbf{v}^{Diode} = [v_A \;\; v_{in} \;\; v_K]^T$ is the node voltage vector, and the equivalent current source vector $\mathbf{i}_{eq}^{Diode}$ is

$$[-i_{Req} - i_{Eeq} - i_{Jeq} \quad i_{Req} + i_{Eeq} + i_{Jeq} \quad 0]^T. \tag{4.26}$$

4.1.3 Hardware Emulation on FPGA

The power diode hardware module includes 6 units, which execute in 3 stages (Fig. 4.3). The Junction Limit unit runs in Stage 1, which is used to limit the p-n junction voltage value

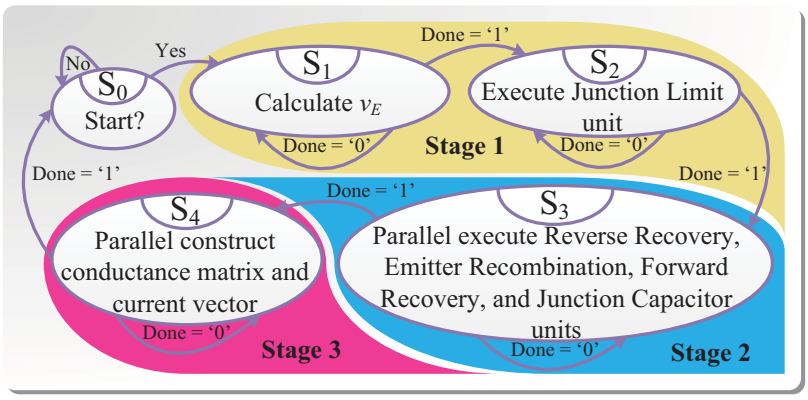

Figure 4.4: The finite state machine of the power diode module.

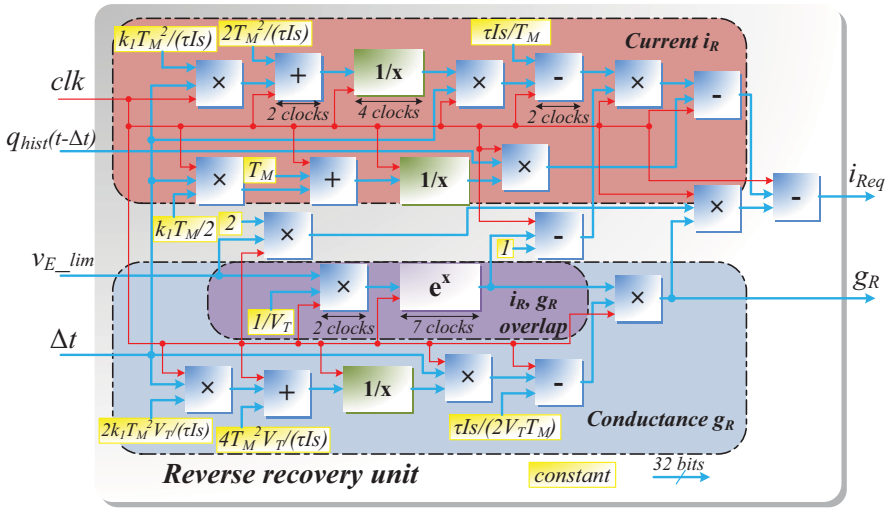

Figure 4.5: Structure of the Reverse Recovery Unit in FPGA.

during the Newton iterations. Specifically, the input signal v_E_old (history v_E value from the previous Newton iteration) is updated by the output signal v_E_new within the current iteration for this unit. This limit models a bounded p-n junction voltage for the diode exponential characteristic. Otherwise, the values related to the diode module will be so high that the system equations may become singular and could not be solved, and sometimes they could even hardly be represented as 32-bit floating point numbers. Stage 2 comprises of 4 units executed in parallel: the Emitter Recombination Unit, the Reverse Recovery Unit, the Forward Recovery Unit, and the Junction Capacitance Unit. Corresponding dynamic conductances and equivalent current sources calculated by these 4 units are used to form the conductance matrix and the equivalent current source vector in Stage 3. The sequence of operations for this module is also reflected in the finite state machine (Fig. 4.4), where all the 4 aforementioned units execute simultaneously in State S_3, before the formation of $\mathbf{G}^{Diode}$ and $\mathbf{i}_{eq}^{Diode}$ within the next state S_4.

Furthermore, the hardware structure of each of the 6 units is highly paralleled. For

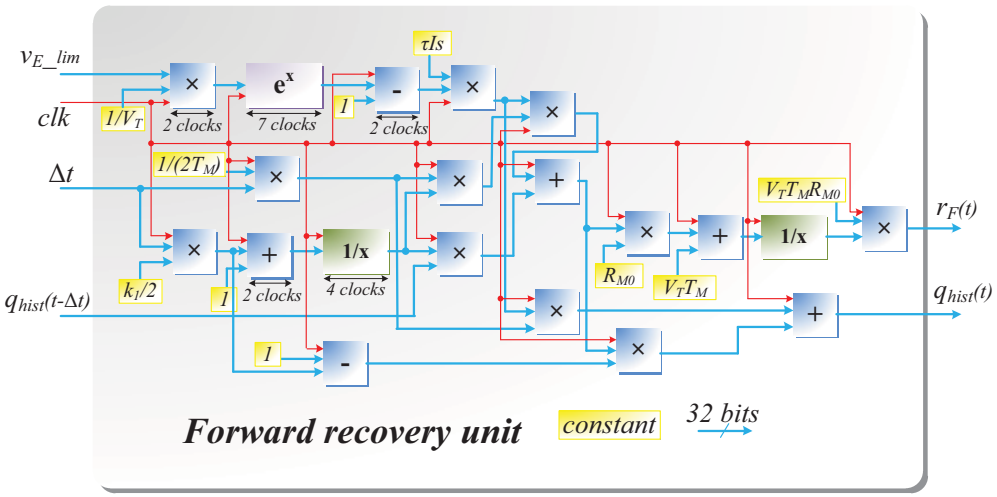

Figure 4.6: Structure of the Forward Recovery Unit in FPGA.

example, within the Reverse Recovery Unit (Fig. 4.5), the calculations of the current i_R and conductance g_R are fully paralleled. To save hardware resources, as can be seen part of the resources are shared and contribute to both calculations. The result is a highly efficient and fast hardware architecture. Terms such as $\frac{k_1 T_M^2}{\tau I_S}$ and $\frac{1}{V_T}$ are precalculated and used as constants in the hardware design, which reduce a great deal of computational complexity and latency. All the basic computation components such as adder, multiplier, and nonlinear operators such as e^x and $1/x$ were generated using the IP Catalog in Xilinx Vivado® Design Suite [98].

Similar structure can be seen in the Forward Recovery Unit (see Fig. 4.6), which has the constant terms such as $V_T T_M$ and $V_T T_M R_{M0}$ and parallelised lay-out of computational subunits. It is worth mentioning that the output signal $q_{his}(t)$ is the history term for the next time-step calculation. However, the calculations for this kind of terms are deliberated put inside Newton iterations to reduce latency of the whole hardware emulation. Therefore, only the last Newton iteration results for the terms like $q_{his}(t)$ are valid. The inner structure of the Junction Limit Unit is shown in Fig. 4.7. This unit includes 4 state transit flag (STF) signals and 4 potential output (PO) signals. The STF signals decide under what circumstances the PO signals are sent to the output v_{lim} (see Fig. 4.8).

4.2 IGBT Module

The Insulated Gate Bipolar Transistor (IGBT) today has become the preferred switching device in a great many power electronics circuits, because it has both the advantages of fast switching speed and low conduction losses. In order to analyse and estimate a range of device and circuit behaviours (e.g. transient and power loss), an accurate IGBT model has become essential. As one of the best mathematical/analytical IGBT models, Hefner's physics-based IGBT model [25], [26], [27], has been adopted [37] by popular device-level

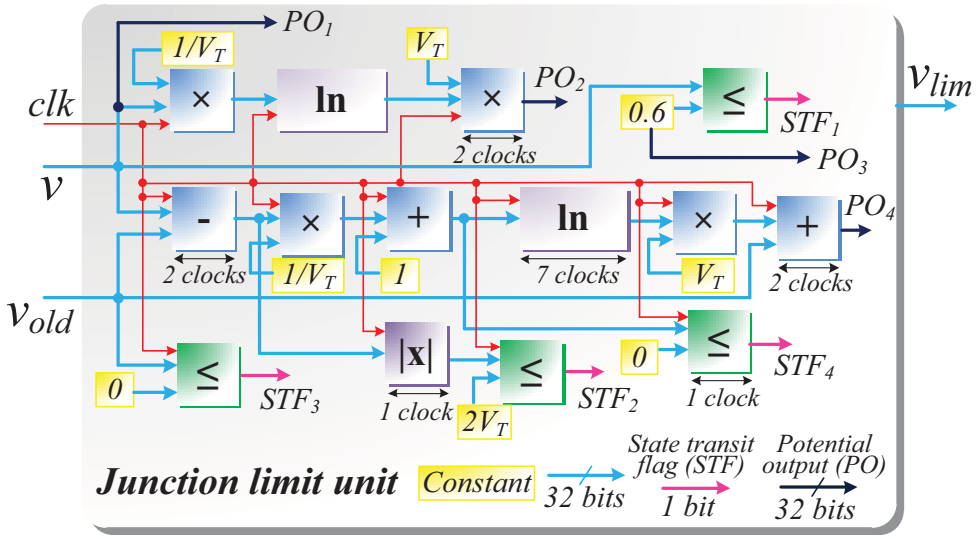

Figure 4.7: Structure of the Junction Limit Unit in FPGA.

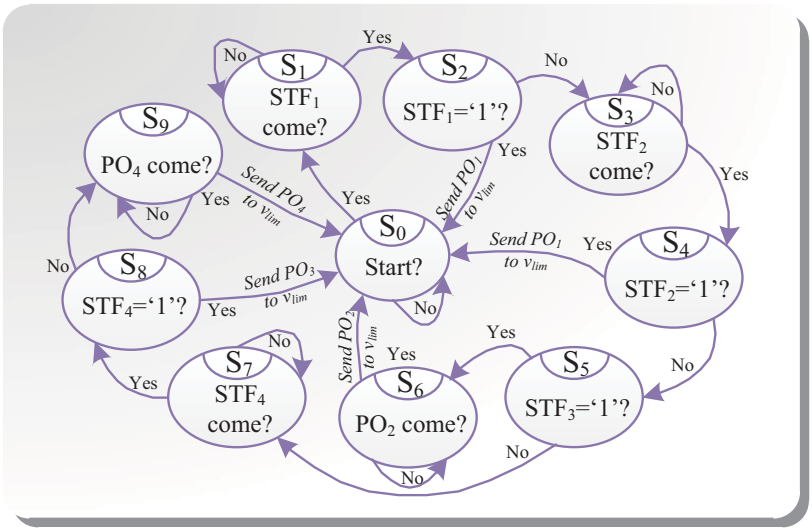

Figure 4.8: Finite state machine of the Junction Limit Unit in FPGA.

circuit simulators such as Saber® and PSpice®.

4.2.1 Model Formulation

According to [26], the IGBT contains the following phenomenological characteristics including: a) internal MOSFET phenomenon, b) internal BJT phenomenon. The base current of the is fed by the MOSFET, thereby combining the advantages of low on-state resistance plus high current capacity of power BJT and excellent gate drive control of power MOSFET. However, both of the internal conceptual BJT and MOSFET function differently with the well-designed microelectronic ones, because their structures differ greatly based on their design goals. Specifically, Hefner interpret the intrinsic nonlinear physical phenom-

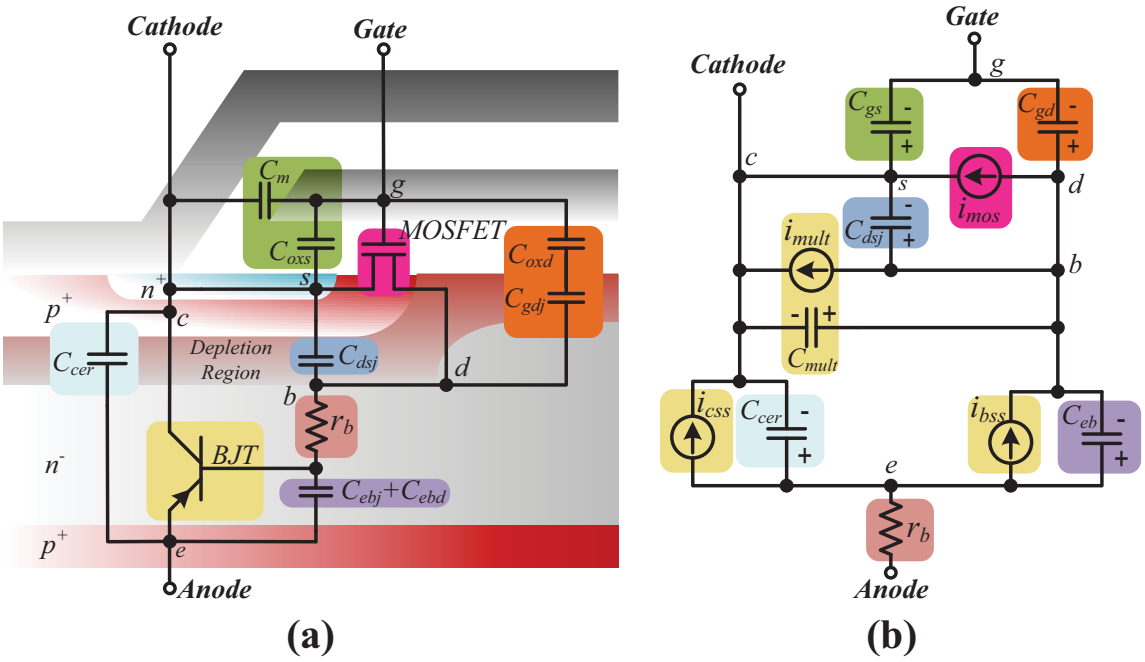

Figure 4.9: Phenomenological (a) and analog (b) equivalent circuits of Hefner's IGBT model [26].

ena into circuit elements (nonlinear capacitors and dependent current sources, Fig. 4.9), which make it possible for device-level circuit simulators to implement the dynamic phenomenological model of IGBT.

The MOSFET channel current is modelled as a current source i_{mos}, which is determined by

$$i_{mos} = \begin{cases} 0 & v_{gs} < V_{th} \\ \frac{K_p K_f[(v_{gs}-V_{th})v_{ds} - \frac{K_f|v_{ds}|v_{ds}}{2}]}{1+\theta(v_{gs}-V_{th})} & |v_{ds}| \leq \frac{v_{gs}-V_{th}}{K_f} \\ \frac{K_p(v_{gs}-V_{th})^2}{2[1+\theta(v_{gs}-V_{th})]} & v_{ds} > 0 \\ \frac{-K_p(v_{gs}-V_{th})^2}{2[1+\theta(v_{gs}-V_{th})]} & v_{ds} < 0 \end{cases} \tag{4.27}$$

where v_{gs} is gate-source voltage, v_{ds} drain-source voltage, K_p is MOSFET transconductance, K_f is MOSFET transconductance factor, V_{th} is MOSFET channel threshold voltage, and θ the transverse field transconductance factor [37]. Its gate-source capacitance C_{gs} equals to the combination of the source metallization capacitance C_m and gate-source overlap oxide capacitance C_{oxs}. The gate-drain capacitance C_{gd} is the sum of gate-drain overlap oxide capacitance C_{oxd} and gate-drain junction depletion capacitance C_{gdj}. C_{dsj} is the drain-source junction depletion capacitance [26].

The BJT is modeled by the steady-state base current source i_{bss} and collector current i_{css}.

The former is decided by

$$i_{bss} = \begin{cases} \frac{Q}{\tau_{HL}} + \frac{4Q^2 N_B^2}{Q_B^2 n_i^2} I_{sne} & v_{eb} > 0 \\ 0 & v_{eb} \leq 0 \end{cases} \tag{4.28}$$

where τ_{HL} stands for the base high-level lifetime, v_{eb} emitter-base voltage, Q_B background mobile carrier base charge, N_B base doping concentration, n_i intrinsic carrier concentration, and I_{sne} emitter electron saturation current. Q is the instantaneous excess carrier base charge, which is decided by:

$$Q = p_0 qAL \; tanh(\frac{W}{2L}), \tag{4.29}$$

where q is the electron charge, A is the active device area, L is the ambipolar diffusion length (the ambipolar diffusion here is defined as the diffusion of electrons and holes with opposite electrical charges under the electrical field), p_0 is the carrier concentration at emitter end of base, which is decided by a nonlinear equation:

$$(\frac{p_0}{n_i^2} + \frac{1}{N_B})(N_B + p_0)^{1-\beta} N_B^{\beta} = e^{\frac{qv_{eb}}{kT}}, \tag{4.30}$$

where β is order of the nonlinearity, k the Boltzmann constant, T is the room temperature, and W is quasi-neutral base width given as:

$$W = W_B - \sqrt{\frac{2\epsilon_{si}(v_{bc} + 0.6)}{qN_B}}, \tag{4.31}$$

where W_B is the metallurgical base width, ϵ_{si} is the silicon dielectric constant, v_{bc} is the base-collector voltage, equal to v_{ds} [26].

The BJT collector current is decided by

$$i_{css} = \begin{cases} \frac{i_T}{1+b} + \frac{4bD_p}{W^2(1+b)} Q & v_{eb} > 0 \\ 0 & v_{eb} \leq 0 \end{cases} \tag{4.32}$$

where i_T represents the anode current, b ambipolar mobility ratio, D_p hole diffusivity [26]. Its emitter-base capacitance C_{eb} is coming from the joint effect of the emitter-base junction depletion capacitance C_{ebj} and the emitter-base diffusion capacitance C_{ebd}. C_{cer} is the collector-emitter redistribution capacitance. r_b is the conductivity-modulated base resistance, through which and the anode current i_T conforms to $i_T = v_{ae}/r_b$, where v_{ae} stands for anode-emitter voltage. The avalanche multiplication current source i_{mult}, given by

$$i_{mult} = (M-1)(i_{mos} + i_{css}) + M \cdot i_{gen}, \tag{4.33}$$

where M is the avalanche multiplication factor defined as:

$$M = \frac{1}{1 - (\frac{v_{ds}}{BV_{cb0}})^{BV_n}}, \tag{4.34}$$

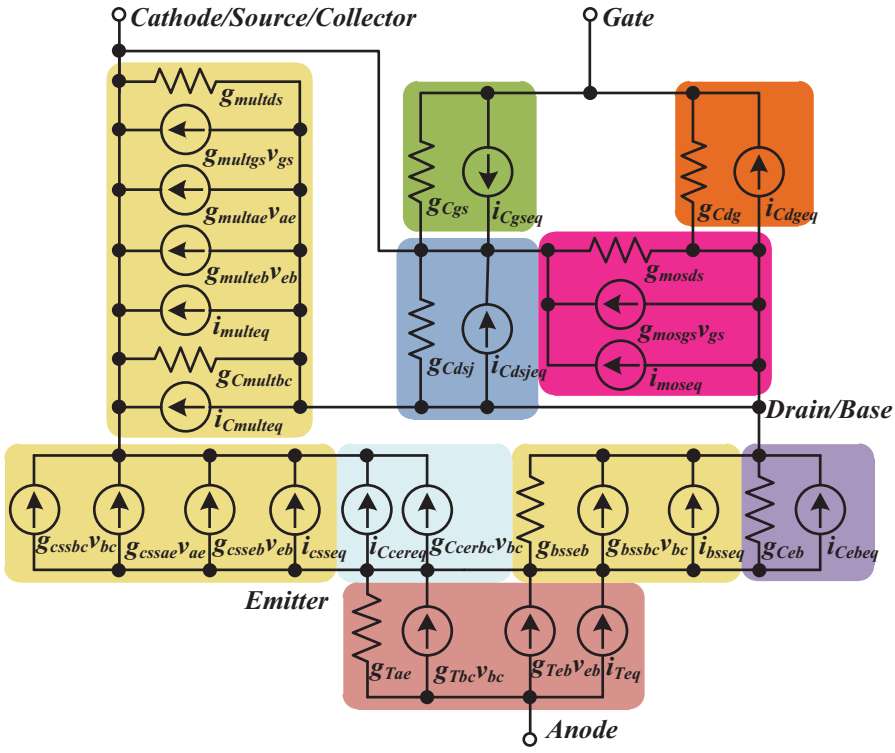

Figure 4.10: Discrete-time linearized equivalent circuit for the IGBT.

where BV_n is the avalanche multiplication exponent, BV_{cb0} is collector-base breakdown voltage (emitter open), and i_{gen} is collector-base thermally generated current given by:

$$i_{gen} = \frac{qn_iA}{\tau_{HL}}\sqrt{\frac{2\epsilon_{si}v_{bc}}{qN_B}}, \tag{4.35}$$

and capacitor C_{mult} supplement the base to collector current for the BJT [26].

Notably, all the aforementioned capacitances (except for constant C_{gs}) are charge dependent, whose values are directly connected to the depletion region width of the p-n junctions inside the IGBT and they all conform to the formulations like

$$C_x = \frac{A_x \cdot \epsilon_{si}}{W_x}, \tag{4.36}$$

where A_x is the active area related to the capacitance, and W_x is the depletion width related to the capacitance [26].

4.2.2 Model Discretization and Linearization

The discrete-time linearized IGBT model (Fig. 4.10) can be obtained using procedures similar to the power diode module in Sec. II (A), which contains 11 conductances (e.g. g_{bsseb}), 11 equivalent current sources (e.g. i_{bsseq}), and 10 voltage-controlled current sources (e.g. $g_{bssbc} \cdot v_{bc}$). All the capacitances (except for C_{cer}) are modeled as pairs of conductance and

equivalent source (e.g. g_{Cgs} and i_{Cgseq}). For example, the drain-source capacitance C_{dsj} has its conductance g_{Cdsj} formulated as:

$$g_{Cdsj} = \begin{cases} 0 & W = W_B \\ \frac{2\epsilon_{si}(A-A_{gd})}{\Delta t \cdot (W_B - W)} & W \neq W_B \end{cases} \tag{4.37}$$

where A_{gd} is the gate-drain overlap area, and the equivalent current source i_{Cdsjeq} is formulated as:

$$i_{Cdsjeq} = i_{Cdsj} - g_{Cdsj} v_{ds} \tag{4.38}$$

where

$$i_{Cdsj} = \frac{2}{\Delta t}[q_{Cdsj} - q_{Cdsj}(t - \Delta t)] - i_{Cdsj}(t - \Delta t) \tag{4.39}$$

and

$$q_{Cdsj} = qN_B(A - A_{gd})(W_B - W). \tag{4.40}$$

In contrast, the current sources (e.g. i_{mos}) are modeled as not only pairs of conductance and equivalent source (except for i_{css}), but also as voltage-controlled current sources (e.g. $g_{mosgs} \cdot v_{gs}$). The MOSFET channel current i_{mos}, for example, has its conductance g_{mosgs} ($\frac{\partial i_{mos}}{\partial v_{gs}}$) formulated as:

$$g_{mosgs} = \begin{cases} \frac{K_p K_f v_{ds} - i_{mos}\theta}{1+\theta(v_{gs}-V_{th})} & |v_{ds}| \leq \frac{v_{gs}-V_{th}}{K_f} \\ \frac{K_p(v_{gs}-V_{th}) - i_{mos}\theta}{1+\theta(v_{gs}-V_{th})} & v_{ds} > 0 \\ \frac{-K_p(v_{gs}-V_{th}) - i_{mos}\theta}{1+\theta(v_{gs}-V_{th})} & v_{ds} < 0 \\ G_{min} & v_{gs} < V_{th} \end{cases} \tag{4.41}$$

where G_{min} is the MOSFET minimum conductance. The drain-source conductance g_{mosds} ($\frac{\partial i_{mos}}{\partial v_{ds}}$) is formulated as:

$$g_{mosds} = \begin{cases} \frac{K_p K_f (v_{gs}-V_{th}-K_f|v_{ds}|)}{1+\theta(v_{gs}-V_{th})} & |v_{ds}| \leq \frac{v_{gs}-V_{th}}{K_f} \\ 0 & v_{ds} > 0 \\ 0 & v_{ds} < 0 \\ G_{min} & v_{gs} < V_{th} \end{cases} \tag{4.42}$$

and its equivalent current i_{moseq} can be obtained as:

$$i_{moseq} = i_{mos} - g_{mosds} v_{ds} - g_{mosgs} v_{gs}. \tag{4.43}$$

All aforementioned conductances (including those formulated in the voltage-controlled current sources) and equivalent current sources are finally combined to form a 5×5 conductance matrix $\mathbf{G}^{IGBT}$ and current source vector $\mathbf{i}_{eq}^{IGBT}$, which satisfy the equation:

$$\mathbf{G}^{IGBT} \cdot \mathbf{v}^{IGBT} = \mathbf{i}_{eq}^{IGBT}, \tag{4.44}$$

where $\mathbf{v}^{IGBT} = [v_c \;\; v_g \;\; v_a \;\; v_d \;\; v_e]^T$ is the IGBT node voltage vector, and $\mathbf{G}^{IGBT}$ and $\mathbf{i}_{eq}^{IGBT}$ are given as (4.45) and (4.46) respectively.

$$\mathbf{G}^{IGBT} = \begin{bmatrix} g_{Cgs} + g_{Cdsj} + g_{Ccerbc} + & -g_{Cgs} - g_{mosgs} & -g_{cssae} & -g_{Cdsj} - g_{mosds} - g_{Ccerbc} & g_{cssae} - g_{csseb} + \\ g_{mosds} + g_{mosgs} + g_{multgs} + & -g_{multgs} & -g_{multae} & +g_{csseb} - g_{multds} + g_{multeb} & g_{multae} - g_{multeb} \\ g_{multds} + g_{Cmultbc} + g_{cssbc} & & & -g_{Cmultbc} - g_{cssbc} & \\ -g_{Cgs} & g_{Cgs} + g_{Cdg} & 0 & -g_{Cdg} & 0 \\ -g_{Tbc} & 0 & g_{Tae} & -g_{Teb} + g_{Tbc} & g_{Teb} - g_{Tae} \\ g_{bssbc} - g_{Cdsj} - g_{mosds} & -g_{Cdg} + g_{mosgs} & & g_{Cdg} + g_{Cdsj} + g_{Ceb} & -g_{Ceb} - g_{bsseb} \\ -g_{mosgs} - g_{multgs} & +g_{multgs} & g_{multae} & +g_{mosds} + g_{bsseb} & -g_{multae} + g_{multeb} \\ -g_{multds} - g_{Cmultbc} & & & +g_{multds} - g_{multeb} & \\ -g_{Ccerbc} - g_{bssbc} & & g_{cssae} - & -g_{Ceb} - g_{bsseb} + g_{Ccerbc} & g_{Ceb} + g_{bsseb} \\ -g_{cssbc} + g_{Tbc} & 0 & g_{multae} & -g_{csseb} + g_{Teb} + & -g_{cssae} + g_{csseb} \\ & & & g_{bssbc} + g_{cssbc} - g_{Tbc} & -g_{Teb} + g_{Tae} \end{bmatrix} \tag{4.45}$$

$$\mathbf{i}_{eq}^{IGBT} = \begin{bmatrix} i_{moseq} + i_{csseq} + i_{multeq} & & & -i_{moseq} + i_{bsseq} - i_{multeq} & -i_{bsseq} - i_{csseq} \\ +i_{Cgseq} + i_{Cdsjeq} & - i_{Cgseq} + i_{Cdgeq} & - i_{Teq} & -i_{Cdsjeq} - i_{Cdgeq} & - i_{Cebeq} - i_{Ccereq} \\ +i_{Ccereq} + i_{Cmulteq} & & & +i_{Cebeq} - i_{Cmulteq} & +i_{Teq} \end{bmatrix}^T \tag{4.46}$$

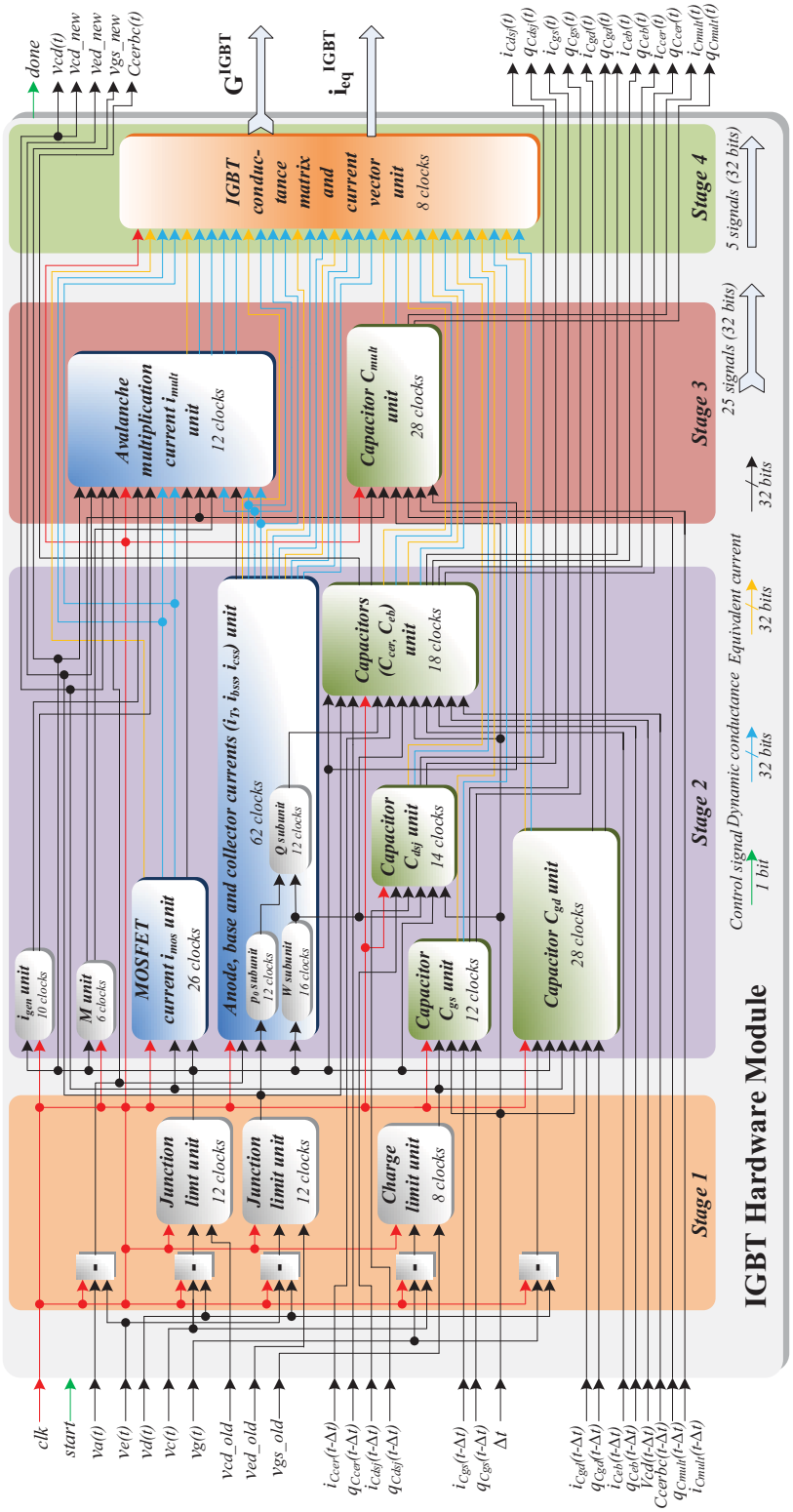

Figure 4.11: Architecture of the IGBT hardware module with all units horizontally scaled with respect to latency.

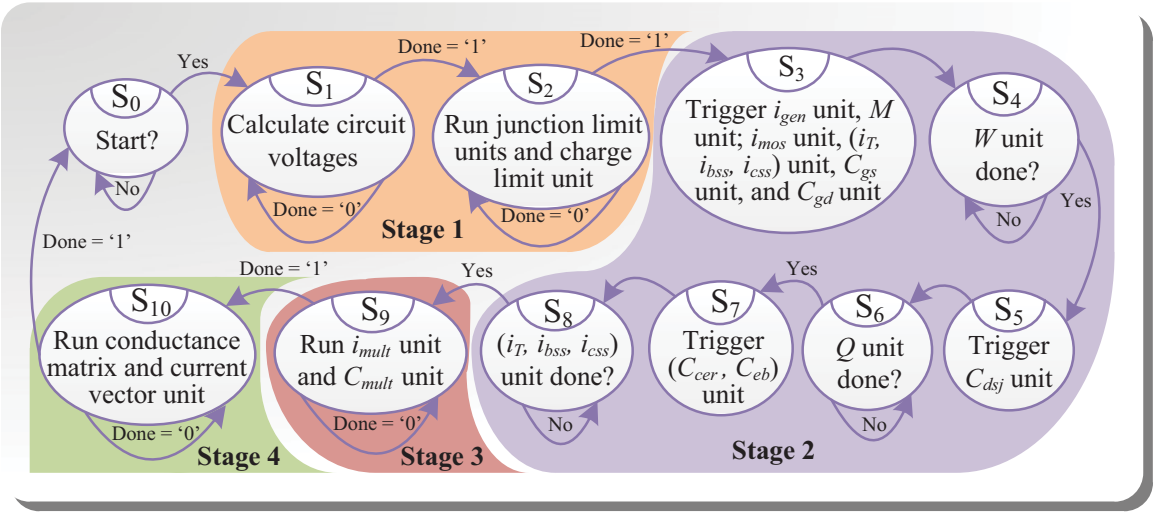

Figure 4.12: The finite state machine of the IGBT hardware module.

4.2.3 Hardware Emulation on FPGA

The basic strategy of the IGBT model hardware design is to turn all the components of the analog equivalent circuit (Fig. 4.9(b)) into 14 hardware units (see Fig. 4.11), to fully exploit the possibility of hardware parallelism. Also, it is highly efficient to code and debug all the hardware units individually. The customizable nature of the IGBT hardware module makes it extremely flexible to fit into different power converter topologies with varying module complexities. All the current units including i_{mos}, (i_T, i_{bss}, i_{css}), and i_{mult} are responsible for the calculations of the corresponding currents, dynamic conductances, and equivalent current sources. All the capacitance units including the C_{gs}, C_{gd}, C_{dsj}, (C_{cer}, C_{eb}), and C_{mult} are responsible for the calculations of not only the currents, dynamic conductances and equivalent currents, but also the charges. The Junction Limit Units and the conductance matrix and current vector unit have similar functions to those used in the power diode hardware module. Other units like the Charge Limit Unit is used to limit the charge variation range between successive Newton iterations, while the avalanche factor unit calculates M, and the i_{gen} unit calculates i_{gen} and its base-collector conductance, which are essential for the calculations in the i_{mult} unit and C_{mult} unit. The input signals to the IGBT hardware module includes the circuit node voltages such as v_c (cathode/source/collector), v_g (gate), v_a (anode), v_d (drain/base) and v_e (emitter). The 3 input signals v_{cd}_old, v_{ed}_old, v_{gs}_old and their 3 output counterparts v_{cd}_new, v_{ed}_new, v_{gs}_new are used to transfer the necessary history values between successive Newton iterations.

As shown in Fig. 4.11, the IGBT hardware module executes in 4 stages. In Stage 1, the 2 junction limit units and the charge limit unit run in parallel. In Stage 2, eight other units run in parallel, which include the i_{gen}, M, i_{mos}, (i_T, i_{bss}, i_{css}), C_{gs}, C_{dg}, C_{dsj} unit, and the (C_{cer}, C_{eb}). In Stage 3, the i_{mult} unit and C_{mult} unit run in parallel. In Stage 4,

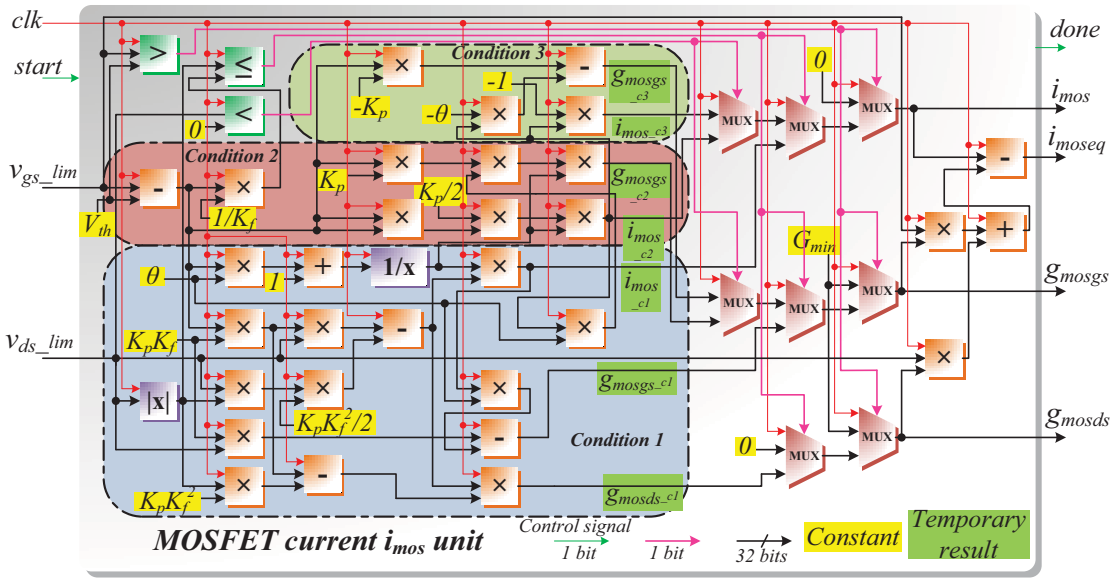

Figure 4.13: Architecture of the MOSFET current i_{mos} unit.

all the dynamic conductances and equivalent currents calculated by the above units are combined to the output conductance matrix $\mathbf{G}^{IGBT}$ and equivalent current source vector $\mathbf{i}_{eq}^{IGBT}$. Moreover, the internal structure of all the aforementioned units or subunits are paralleled. The operation sequence of the IGBT hardware module can be seen in Fig. 4.12, where 6 different states (S_3 to S_8) contribute to Stage 2. The reason is that W subunit and Q subunit of the (i_T, i_{bss}, i_{css}) unit interconnect to the C_{dsj} unit and the (C_{cer}, C_{eb}) unit, respectively. So for synchronization within the (i_T, i_{bss}, i_{css}) unit, the 2 subunits responsible for W, p_0 computations are also executed in parallel, these units need to be triggered at the right time.

4.2.3.1 Hardware Designs of the current units

The internal hardware structure of one of the units, the MOSFET current i_{mos} unit, is shown in Fig. 4.13, to illustrates the parallelism. There are 3 comparators and 5 multiplexers which are responsible for selecting 4 groups of results among 3 nested conditions, see (4.41) and (4.42). Since one group of results (when $v_{gs} < V_{th}$) is obtained directly, the remaining 3 groups of temporary results for i_{mos}, g_{mosgs}, g_{mosds} are all calculated before being selected by the 3 layers of multiplexers under different conditions as the final results. Only after the results for i_{mos}, g_{mosgs}, g_{mosds} are obtained, the value of i_{moseq} can be computed. In Fig. 4.13 it is highlighted that the hardware structures to calculate the results under 3 conditions are fully in parallel. All basic function and nonlinear operator IP cores were generated using the IP Catalog in Xilinx Vivado® Design Suite [98].

The i_{gen} Unit is a relatively small hardware unit, which calculate i_{gen} and its conductance g_{genbc} ($\frac{\partial i_{gen}}{\partial v_{bc}}$). Its structure and finite state machine are shown in Fig. 4.14. The

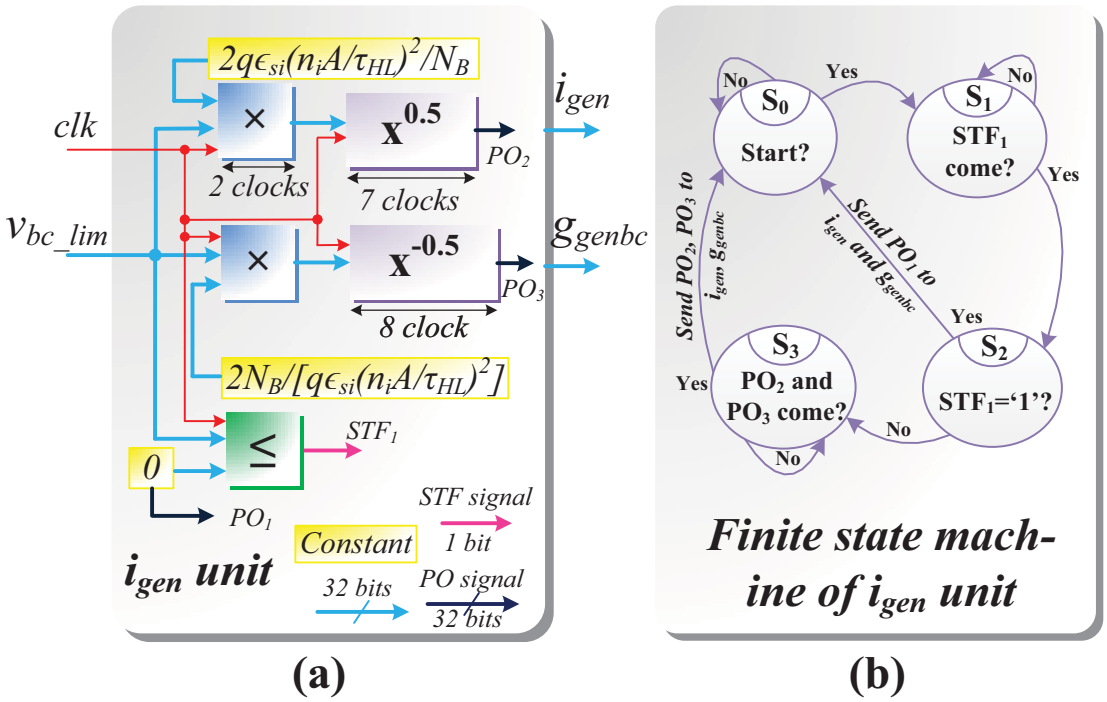

Figure 4.14: Architecture (a) and finite state machine (b) of the i_{gen} Unit.

Avalanche Multiplication Current i_{mult} Unit (Fig. 4.15) conforms to a combinational logic circuit structure, which receives the output signals from i_{gen} Unit, M Unit, i_{mos} unit, and (i_T,i_{bss},,i_{css}) Unit to generate the 3 related conductances and 1 equivalent source.

Specially, inside the Anode, Base and Collector Current (i_T, i_{bss}, i_{css}) Unit, there is a p_0 Subunit, which is used to calculate a single parameter p_0. However, from equation (4.30), it can been seen that there is a complex nonlinear relationship between p_0 and the emitter-base voltage v_{eb}. Temporally, we can represent their relationship as $p_0 = f(v_{eb})$ (v_{eb} is ranging from 0 to nearly 0.9 in this equation), if this hardware subunit is designed based on using Newton-Raphson method (for most of the time, it takes 5 to 7 iterations to converge and get the result), it is obviously very time-consuming to run the subunit.
To overcome such potential shortcoming, the look-up table (LUT) method (see Fig. 4.16) is applied due to its strong advantages to tackle the nonlinear equations. In practice, a list of values of output p_0 are precalculated offline using Newton method. More specifically, 90000 different values of v_{eb} $(0.00000, 0.00001, 0.00002, \cdots, 0.89998, 0.89999)$ input into the nonlinear equation to get corresponding output values. After that, the 90000 values of p_0 was simultaneously fed to a dual-port ROM, thereby getting the actual p_0 value by linear interpolation method. The ROM contributes to find the interpolation high bound fx_h and low bound fx_l. The addresses of the ROM are generated by multiplying the input v_{eb} by 10^5 to a larger value x before add (for high bound ROM) or subtract (for low bound ROM) 0.5, thereby making sure that the float to fixed units (turning 32-bit float point value to

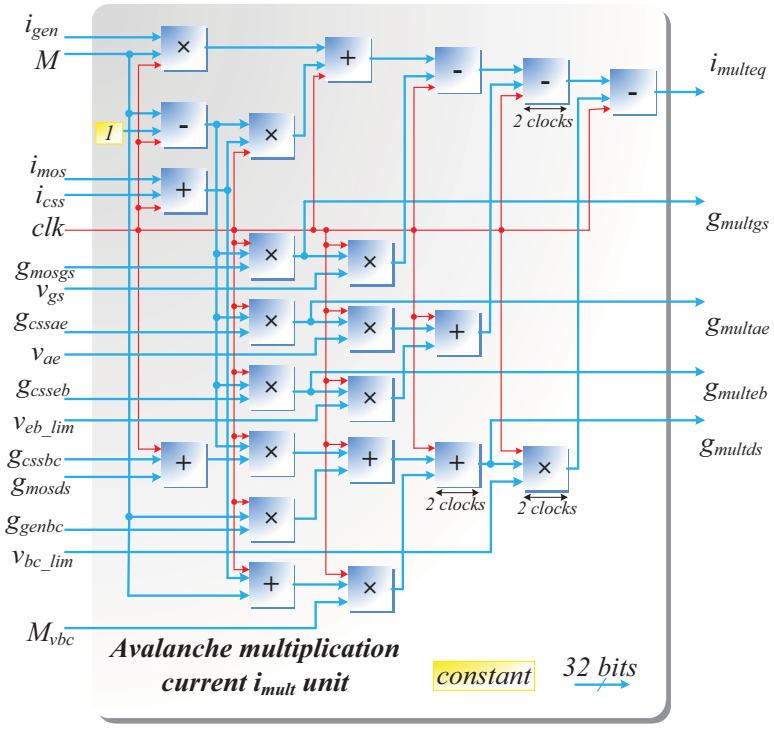

Figure 4.15: Architecture of the Avalanche Multiplication Current i_{mult} Unit.

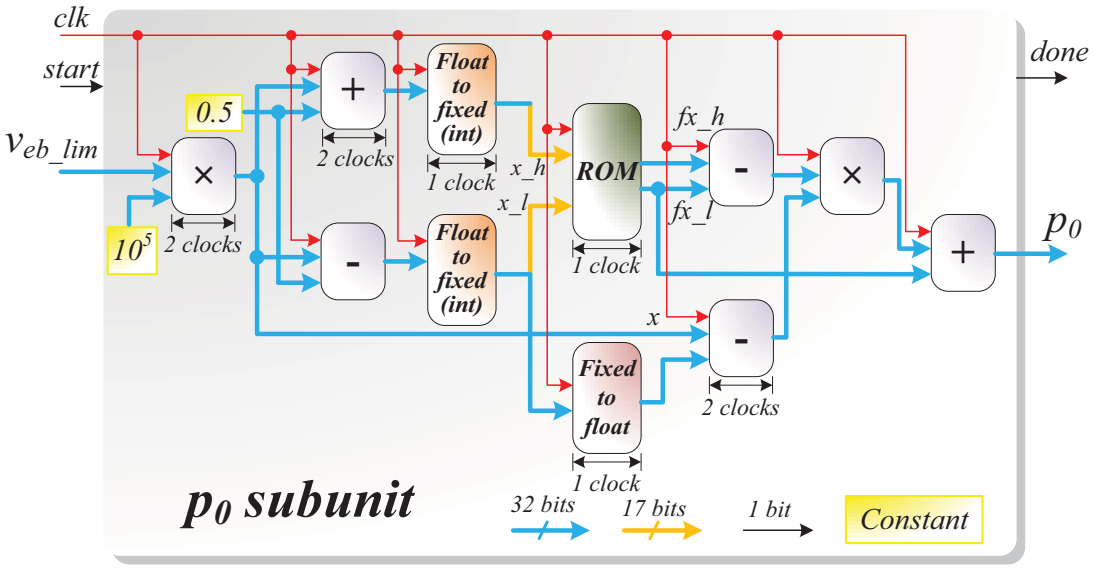

Figure 4.16: Architecture of the p_0 subunit.

32-bit integer value) could find their accurate corresponding low bound address x_l and high bound address x_h, after extracting the lower 14-bit value.

For example, if the input v_{eb} is 0.456789. Obviously, x will be 45678.9, x_h be 45679, and x_l be 45678 after some calculations. The final solution for p_0 after the linear interpolation

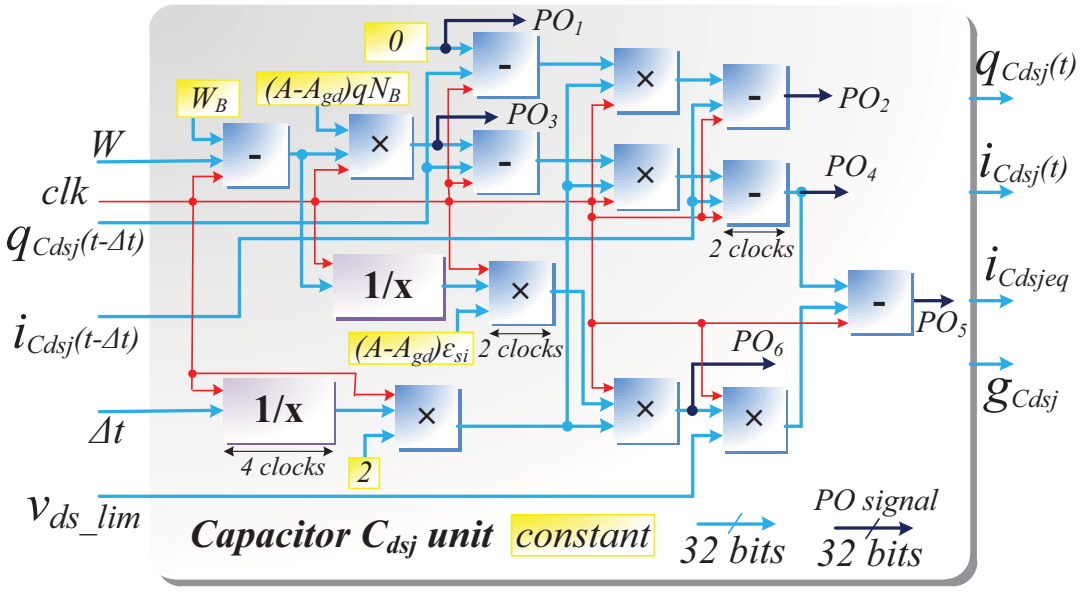

Figure 4.17: Architecture of the Capacitor C_{dsj} Unit.

can be get by

$$f(45678.9) = f(45678) + [f(45679) - f(45678)] \times (45678.9 - 45678) \tag{4.47}$$

where $f(45679)$ and $f(45678)$ are exactly the high bound value fx_h and low bound value fx_l respectively.
The total latency for the optimized module is as short as around 12, much less than the previous 130.

4.2.3.2 Hardware Designs of the capacitor units

The IGBT hardware module has 5 capacitor(s) units. Sine capacitor C_{cer} and C_{eb} are closely related, they are combined as only one unit. Different from current units, the calculations of capacitor(s) units usually include the history terms from the last time-step. For example, the Capacitor C_{dsj} Unit (Fig. 4.17) receives the history terms from the last time-step such as $q_{Cdsj}(t - \Delta t)$ and $i_{Cdsj}(t - \Delta t)$. Meanwhile, it generates the output signals, $q_{Cdsj}(t)$ and $i_{Cdsj}(t)$, for the next time-step calculations. Since this module runs inside Newton iterations, these output signals are only valid during the every last iteration and transferred to the next Δt. As can be seen from Fig. 4.18, the finite state machine of this unit includes 4 different states, which generates two group of results and sends them to the output. As the only linear capacitor in the IGBT module, the Capacitor C_{gs} Unit has a simple combinational circuit structure, which is shown in Fig. 4.19.

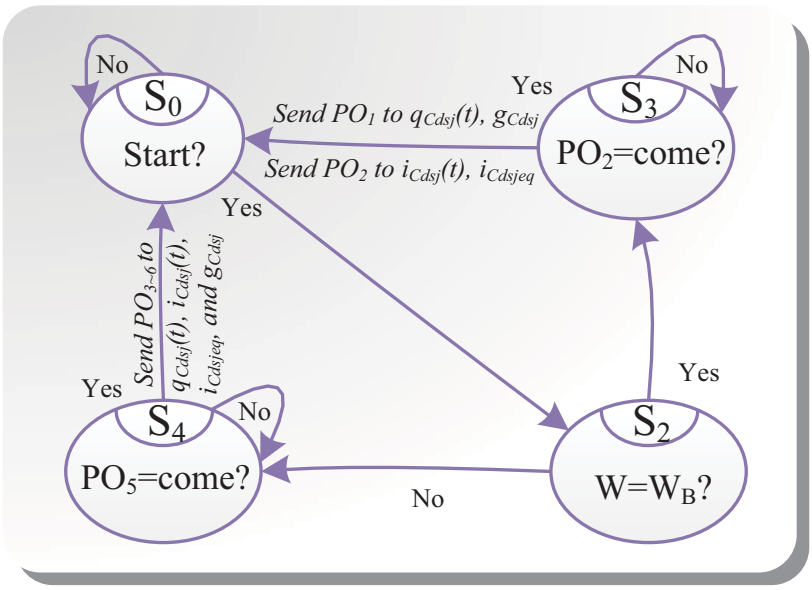

Figure 4.18: Finite state machine of the Capacitor C_{dsj} Unit.

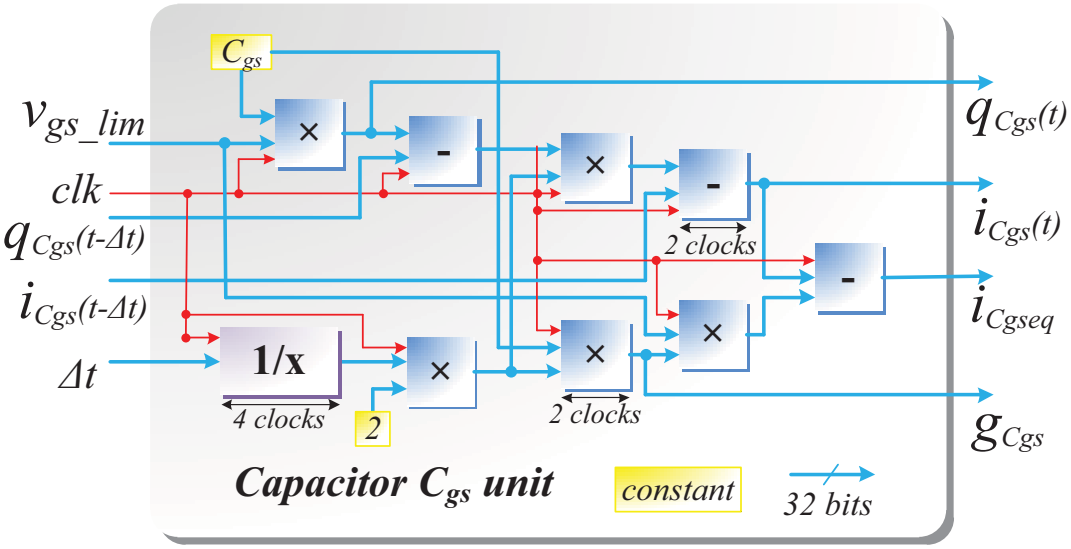

Figure 4.19: Architecture of the Capacitor C_{gs} Unit.

4.3 Power Converter Hardware Emulation

With detailed hardware modules for the power diode, the IGBT, and the linear circuit components, the emulation of a complete power converter can be realized. As shown in Fig. 4.20, it is composed of 3 intervals within a simulation time-step Δt. Interval 1 is responsible for selecting the proper time-step Δt utilizing the Variable Time-Step Control Module (VTCM) as well as the input voltage of the converter circuit. Interval 2 is responsible for performing the Newton iterations, which includes the calculations of the conductance matrix and equivalent current source vector for the whole converter circuit and

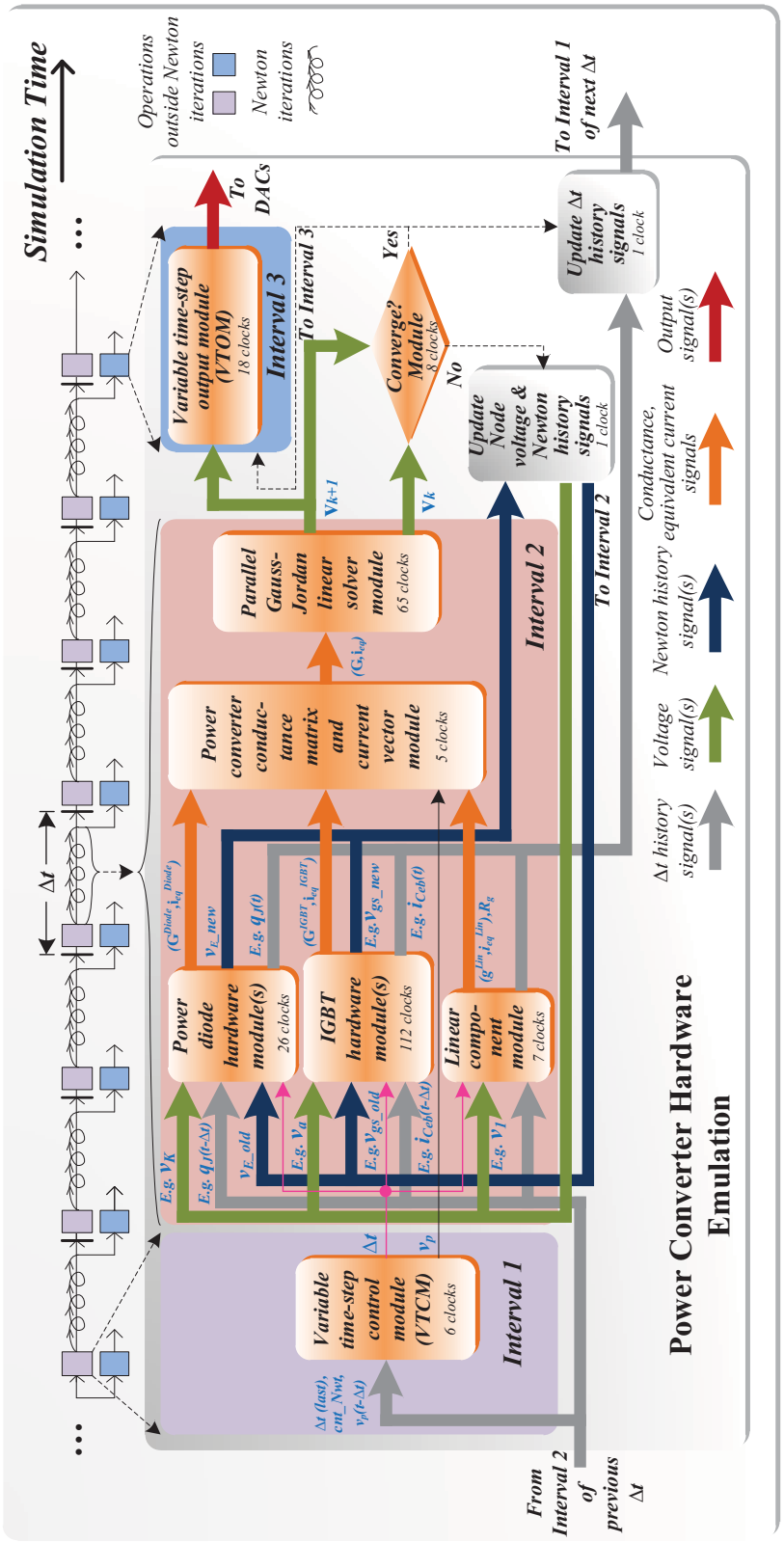

Figure 4.20: Architecture of the power converter hardware emulation.

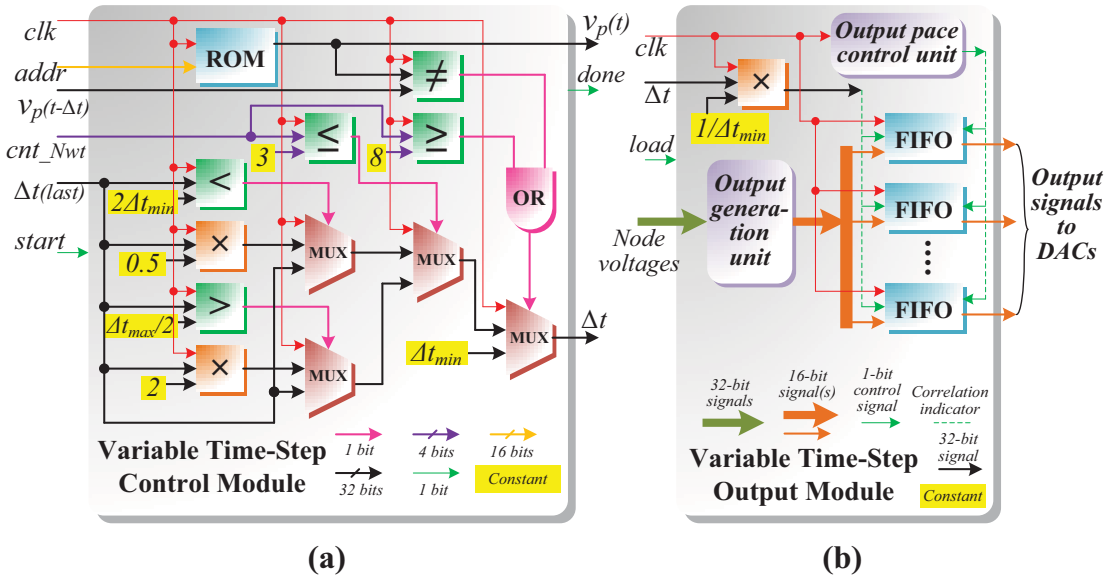

Figure 4.21: Hardware structures of (a) VTCM, and (b) VTOM modules.

obtaining the solution for circuit node voltages using a parallel linear solver. Although the linear component module is triggered simultaneously with the power diode and IGBT modules, their input node voltages are deliberately kept constant unlike those of the nonlinear ones. The reason to put them inside the Newton iteration loop is to increase the parallelism of the whole structure so as to reduce latency. Most of the Δt history terms and Newton history terms are related to the modules in this interval, which are updated during the calculations between successive time-steps and Newton iterations, respectively.

4.3.1 Variable Time-Step Control and Output Modules (VTCM and VTOM)

Accuracy and speed are both essential for the power converter hardware emulation. The time-step Δt could be large when the circuit is operating under steady-state, thereby gaining speed; whereas Δt should be small during transients for higher accuracy. The VTCM in Interval 1 adjusts the time-step Δt based on the number of Newton iterations cnt_Nwt during the calculations of the last time-step. When $cnt\_Nwt \leq 3$, the time-step is increased by 2, and when $cnt\_Nwt > 3$, the time-step is decreased by 2. Similar time-step control strategy has also been applied by Spice [99]. The main function of the VTOM is to ensure that the required output signals are calculated from the converged node voltages, and they are output at a fixed sample rate to guarantee that the waveforms are not out of shape. It receives uneven calculation data from the Newton iterations and sends the outputs at equidistant intervals of time. Fig. 4.21 shows the hardware architectures of the VTCM and VTOM modules.

The input switching signal of the IGBT, v_p, was precalculated and put into a ROM. At the beginning of an emulation step it is compared with its history value from the last

time-step, $v_p(t - \Delta t)$, to decide whether to set the current time-step to minimum. Then, a group of nested comparators and three layers of multiplexers adjust the current time-step for efficiency. In the VTOM module the output pace of the results is based on the current time-step Δt by multiplying a coefficient $\frac{1}{\Delta t_{min}}$ to get an integer value, which in turn controls the inputs of a group of FIFO registers. The larger the current time-step, the more duplicated output results being stored in those FIFOs. The Output Pace Control Unit is designed to adjust the refresh rate of the FIFOs which can be specified by the user. Consequently, the results are sent out evenly with respect to time.

4.3.2 Newton Iterations

Given an n-dimensional nonlinear power converter circuit represented as:

$$\mathbf{i} = F(\mathbf{v}), \tag{4.48}$$

where $\mathbf{v} = (v_1, v_2, \ldots, v_n)^T$, $\mathbf{i} = (i_1, i_2, \ldots, i_n)^T$ are node voltages and current injection vectors respectively, and $F(\cdot)$ is the general nonlinear operator. Using classical Newton-Raphson, the node voltage vector at $(k+1)^{th}$ iteration are calculated as:

$$\mathbf{v}_{k+1} = \mathbf{G}^{-1}(\mathbf{v}_k) \cdot \mathbf{i}_{eq}(\mathbf{v}_k), \tag{4.49}$$

where $\mathbf{i}_{eq}(\mathbf{v}_k)$ is the vector of equivalent current sources, determined as:

$$\mathbf{i}_{eq}(\mathbf{v}_k) = \mathbf{i}(\mathbf{v}_k) - \mathbf{G}(\mathbf{v}_k) \cdot \mathbf{v}_k, \tag{4.50}$$

and $\mathbf{G}(\mathbf{v}_k)$ is the general linearized conductance matrix formulated as:

$$\mathbf{G}(\mathbf{v}_k) = \begin{pmatrix} \frac{\partial i_1}{\partial v_1}|_{\mathbf{v}_k} & \frac{\partial i_1}{\partial v_2}|_{\mathbf{v}_k} & \cdots & \frac{\partial i_1}{\partial v_n}|_{\mathbf{v}_k} \\ \vdots & & & \vdots \\ \frac{\partial i_n}{\partial v_1}|_{\mathbf{v}_k} & \frac{\partial i_n}{\partial v_2}|_{\mathbf{v}_k} & \cdots & \frac{\partial i_n}{\partial v_n}|_{\mathbf{v}_k} \end{pmatrix}. \tag{4.51}$$

The generalized $(\mathbf{G}(v_k), \mathbf{i}_{eq}(v_k))$ pair for the whole converter circuit can be directly obtained by superimpositions from the those of its separate linear and nonlinear components, such as ($\mathbf{G}^{Diode}$, $\mathbf{i}_{eq}^{Diode}$) and ($\mathbf{G}^{IGBT}$, $\mathbf{i}_{eq}^{IGBT}$), which will be illustrated in the case study. The iteration convergence criteria is given as:

$$|\frac{v_{i(k+1)} - v_{i(k)}}{v_{i(k)}}| \leq \epsilon \qquad (i = 1, 2, \cdots, n), \tag{4.52}$$

ϵ is chosen to be 10^{-3}.

4.3.3 Parallel Gauss-Jordan Linear Solver

As seen in Fig. 4.20, in Interval 2, inside the k^{th} Newton iteration, a set of linear equations need to be solved to obtain the node voltages (4.49). The Parallel Gauss-Jordan Linear

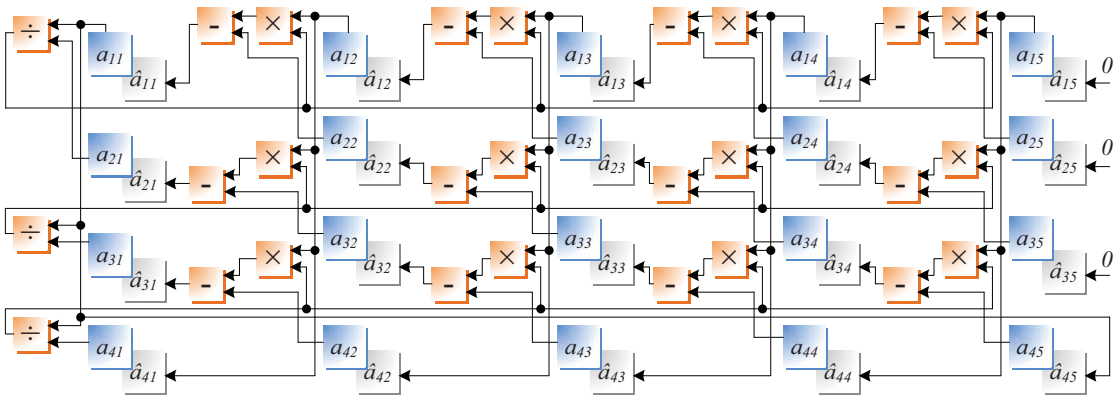

Figure 4.22: Structure of 4-dimensional parallel Gaussian elimination solver.

Solver Module is used in the work for the linear solver. This module consists of two main stages: forward elimination and backward substitution, and its hardware implementation is shown in [15]. To improve its latency, the upgraded structure is featured by deep parallelism in both elimination and backward substitution at the cost of using extra hardware resources, and the pivoting is sped-up to as fast as two clock cycles each step. The advantages of this approach is reduced latency and high efficiency for handling larger matrices. Specially, for a 4-dimensional linear system of equations formulated as

$$\begin{cases} \tilde{a}_{11}x_1 + \tilde{a}_{12}x_2 + \tilde{a}_{13}x_3 + \tilde{a}_{14}x_4 = \tilde{a}_{15} \\ \tilde{a}_{21}x_1 + \tilde{a}_{22}x_2 + \tilde{a}_{23}x_3 + \tilde{a}_{24}x_4 = \tilde{a}_{25} \\ \tilde{a}_{31}x_1 + \tilde{a}_{32}x_2 + \tilde{a}_{33}x_3 + \tilde{a}_{34}x_4 = \tilde{a}_{35} \\ \tilde{a}_{41}x_1 + \tilde{a}_{42}x_2 + \tilde{a}_{43}x_3 + \tilde{a}_{44}x_4 = \tilde{a}_{45} \end{cases} \tag{4.53}$$

where $\tilde{a}_{11}$, $\tilde{a}_{21}$,$\cdots$, $\tilde{a}_{34}$, $\tilde{a}_{44}$ are the coefficients, $\tilde{a}_{15}$, $\tilde{a}_{25}$, $\tilde{a}_{35}$, $\tilde{a}_{45}$ is the right hand side elements, and x_1, x_2, x_3, x_4 are the variables, the detailed structure of the upgraded parallel Gaussian elimination solver is shown in Fig. 4.22. Before transferring the equation elements to their counterparts in the solver, the first time of pivoting should be performed, which could only completed by connecting every two elements between $\tilde{a}_{11}$, $\tilde{a}_{21}$, $\tilde{a}_{31}$, $\tilde{a}_{41}$ to 6 float-point comparators within one clock cycle, since all the comparators run in parallel and their latency is as short as less than one. The outputs of the 6 comparators could combine a vector which is formulated as

$$\mathbf{c} = [c_{12}, c_{13}, c_{14}, c_{23}, c_{24}, c_{34}] \tag{4.54}$$

where c_{12} is 1 if $\tilde{a}_{11} \geq \tilde{a}_{21}$ and 0 if $\tilde{a}_{11} < \tilde{a}_{21}$. Other 5 elements in the vector have similar definition for their values. Therefore, by judging from the value of $\mathbf{c}$, the pivot element can be found. For example, if $\mathbf{c} = [1, 1, 1, x, x, x]$ (x could be either 1 or 0), the pivot element is $\tilde{a}_{11}$. Thereafter, exchange the first row with the row headed with the pivot element and assign the element values to their counterpart positions in the solver.

After one cycle of elimination, the element matrix

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \end{bmatrix} \tag{4.55}$$

shifts left and up for once, changing to

$$\begin{bmatrix} a_{22} - \frac{a_{21}a_{12}}{a_{11}} & a_{23} - \frac{a_{21}a_{13}}{a_{11}} & a_{24} - \frac{a_{21}a_{14}}{a_{11}} & a_{25} - \frac{a_{21}a_{15}}{a_{11}} & 0 \\ a_{32} - \frac{a_{31}a_{12}}{a_{11}} & a_{33} - \frac{a_{31}a_{13}}{a_{11}} & a_{34} - \frac{a_{31}a_{14}}{a_{11}} & a_{35} - \frac{a_{31}a_{15}}{a_{11}} & 0 \\ a_{42} - \frac{a_{41}a_{12}}{a_{11}} & a_{43} - \frac{a_{41}a_{13}}{a_{11}} & a_{44} - \frac{a_{41}a_{14}}{a_{11}} & a_{45} - \frac{a_{41}a_{15}}{a_{11}} & 0 \\ a_{12} & a_{13} & a_{14} & a_{15} & a_{11} \end{bmatrix} \tag{4.56}$$

which are temporarily fed in the the temporary register of every element (shown as the shadow block behind every element in Fig. 4.22), waiting to update the element values for the next cycle of elimination before pivoting again.

The second time of pivoting is similar to the first time, except that it only compares the first elements of the first three rows. After comparing, the pivot element is found and the element values in the temporary registers are mapped and assigned to their correct positions in the element matrix in the solver, preparing for the next cycle of elimination.

For a 4-dimensional matrix, the whole times for the elimination is 4 and the whole latency is not more than 50, which is very fast.
Obviously, this structure can be applied to any dimensional linear system of equations. Actually, the larger the dimension of matrix, the greater the advantages of the parallel linear solver. Since the latency increase conforms to a linear trend instead of an exponential one like most of other linear solvers.

4.4 Summary

This chapter proposed a digital hardware emulation of detailed physics-based device-level IGBT and diode models. Both these models are fully paralleled on the FPGA. The hardware emulation of both devices is based on a unified numerical framework, and can be extended in a straightforward fashion to model complete power electronics circuits. The Newton-Raphson linearization is very advantageous in power converter hardware emulations, which may involve all kinds of different topologies and this method is quite useful due to its strong flexibility and convenience in the linearized model formulation. The variable time-step strategy makes the hardware emulations with power converter based on physics-based mathematical model become possible, which not only ensures the speed of the emulation but also the detailed switching behaviours.

5 Case Studies and Experimental Results

In the case studies, a DC-DC buck converter circuit and a 2-level DC-AC converter are emulated to validate the hardware IGBT and diode models as well as the circuit emulation methodology. The captured oscilloscope results demonstrate high accuracy of the emulator in comparison to the off-line simulation of the original systems using Saber® software.

5.1 DC-DC Converter

The test circuit used for the power converter hardware emulation is a DC-DC buck converter shown in Fig. 5.1. There are 2 nonlinear components (IGBT and Diode), 4 linear components and a voltage source in the circuit. The parameters of the circuit are given in Table 5.1. The duty ratio of the square wave PWM to trigger the IGBT is set to 0.5. The IGBT node voltages v_c, v_g, v_a, v_d, v_e are also the circuit node voltages v_1, v_2, v_7, v_3, v_4, re-

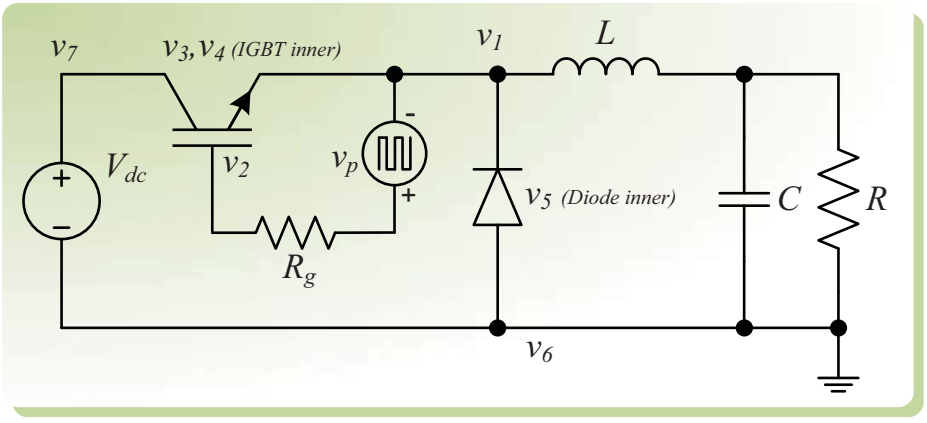

Figure 5.1: The DC-DC buck converter circuit.

Table 5.1: DC-DC Test Circuit and Device Parameters

Test Circuit Parameters
$V_{DC} = 100V$, $R = 5\Omega$, $L = 700\mu H$, $C = 70\mu F$, $R_g = 100\Omega$
IGBT Parameters
Out of 32 parameters, the ones used in this paper: $K_p = 1A/V$, $K_f = 2$, $\theta = 0.01V^{-1}$, $V_{th} = 5V$, $N_B = 2 \times 10^{14}cm^{-3}$, $I_{sne} = 10^{-14}A$, $n_i = 1.45 \times 10^{10}cm^{-3}$, $\tau_{HL} = 4 \times 10^{-7}s$, $A = 0.1cm^2$, $q = 1.6 \times 10^{-19}C$, $L = 2.7 \times 10^{-3}cm$, $T = 296K$, $k = 8.617 \times 10^{-5}eV/K$, $\beta = 0.4615$, $\epsilon_{si} = 1.05 \times 10^{-12}F/cm$, $W_B = 0.01cm$, $b = 3.33$, $D_p = 11.48cm^2/s$, $BV_n = 4$, $BV_{cb0} = 3.18 \times 10^7V$, $G_{min} = 10^{-12}S$, $A_{gd} = 0.05cm^2$. The rest can be obtained from Saber®.
Diode Parameters
$I_S = 10^{-14}A$, $\tau = 5\mu s$, $T_M = 5\mu s$, $V_T = 0.0259V$, $m = 0.5$, $C_{J0} = 1nF$, $V_J = 0.7V$, $I_{SE} = 10^{-22}A$, $R_{M0} = 50\Omega$, $R_C = 10^{-3}\Omega$

spectively. The diode node voltages v_A, v_{in}, and v_K correspond to the circuit node voltages v_6, v_5, and v_1. The gate voltage source v_p with a serial resistor R_g can be equally changed to a current source $\frac{v_p}{R_g}$ and a parallel resistor R_g in order to reduce the dimension of the whole system [100]. The linear components, including the inductor L, load resistor R, and load capacitor C, can be collectively discretized as a conductance g^{Lin} and a parallel equivalent current source i_{eq}^{Lin}. Therefore, the whole discrete-time linearized system can be represented as equation (5.1). Since the circuit node voltages $v_6 = 0$ (ground) and $v_7 = V_{dc}$, it can be reduced to 5, given as equation (5.2).

This DC-DC converter was emulated on the Xilinx® Virtex®-7 XC7VX485T FPGA board, which was connected to a 16-bit 4 channel DACs. This board has 607200 FFs, 303600 LUTs, 130800 Memory LUTs, 700 I/Os, 2060 BRAMs, 2800 DSP48s, 32 BUFGs. The emulation results were captured by a 4-channel oscilloscope. The off-line simulation for the DC-DC converter was executed on a PC with Intel® Core™i7-2600K 3.4GHz CPU, 8GB RAM, running Windows® 7 operating system. The execution time for the off-line simulation of the converter under 2.5kHz switching frequency using Saber® for 100ms using variable time-step strategy (with an initial time-step of 10ns and maximum time-step of 1μs) was 17.5s, while the hardware emulation time for 100ms of simulation time was 0.58s. Under the switching frequency of 40kHz, the running time for Saber® for 100ms was 114s, while the hardware emulation time was 3.68s. Therefore, the speed-up is more than 30 times. It is conceivable that off-line simulation of converter circuits with more IGBTs and diodes employing detailed device-level models would be much slower than hardware emulation.

$$\begin{bmatrix} \mathbf{G}^{IGBT}(1,1)+\mathbf{G}^{Diode}(3,3)+g^{Lin}+1/R_g & \mathbf{G}^{IGBT}(1,2)-1/R_g & \mathbf{G}^{IGBT}(1,4) & \mathbf{G}^{IGBT}(1,5) & \mathbf{G}^{Diode}(3,2) & \mathbf{G}^{Diode}(3,1)-g^{Lin} & \mathbf{G}^{IGBT}(1,3) \\ \mathbf{G}^{IGBT}(2,1)-1/R_g & \mathbf{G}^{IGBT}(2,2)+1/R_g & \mathbf{G}^{IGBT}(2,4) & \mathbf{G}^{IGBT}(2,5) & 0 & 0 & \mathbf{G}^{IGBT}(2,3) \\ \mathbf{G}^{IGBT}(4,1) & \mathbf{G}^{IGBT}(4,2) & \mathbf{G}^{IGBT}(4,4) & \mathbf{G}^{IGBT}(4,5) & 0 & 0 & \mathbf{G}^{IGBT}(4,3) \\ \mathbf{G}^{IGBT}(5,1) & \mathbf{G}^{IGBT}(5,2) & \mathbf{G}^{IGBT}(5,4) & \mathbf{G}^{IGBT}(5,5) & 0 & 0 & \mathbf{G}^{IGBT}(5,3) \\ \mathbf{G}^{Diode}(2,3) & 0 & 0 & 0 & \mathbf{G}^{Diode}(2,2) & \mathbf{G}^{Diode}(2,1) & 0 \\ \mathbf{G}^{Diode}(1,3)-g^{Lin} & 0 & 0 & 0 & \mathbf{G}^{Diode}(1,2) & \mathbf{G}^{Diode}(1,1)+g^{Lin} & 0 \\ \mathbf{G}^{IGBT}(3,1) & \mathbf{G}^{IGBT}(3,2) & \mathbf{G}^{IGBT}(3,4) & \mathbf{G}^{IGBT}(3,5) & 0 & 0 & \mathbf{G}^{IGBT}(3,3) \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix} = \begin{bmatrix} \mathbf{i}_{eq}^{IGBT}(1)+\mathbf{i}_{eq}^{Diode}(3) \\ -i_{eq}^{Lin}-v_p/R_g \\ \mathbf{i}_{eq}^{IGBT}(2)+v_p/R_g \\ \mathbf{i}_{eq}^{IGBT}(4) \\ \mathbf{i}_{eq}^{IGBT}(5) \\ \mathbf{i}_{eq}^{Diode}(2) \\ \mathbf{i}_{eq}^{Diode}(1)+i_{eq}^{Lin} \\ \mathbf{i}_{eq}^{IGBT}(3) \end{bmatrix} \tag{5.1}$$

$$\begin{bmatrix} \mathbf{G}^{IGBT}(1,1)+\mathbf{G}^{Diode}(3,3)+g^{Lin}+1/R_g & \mathbf{G}^{IGBT}(1,2)-1/R_g & \mathbf{G}^{IGBT}(1,4) & \mathbf{G}^{IGBT}(1,5) & \mathbf{G}^{Diode}(3,2) \\ \mathbf{G}^{IGBT}(2,1)-1/R_g & \mathbf{G}^{IGBT}(2,2)+1/R_g & \mathbf{G}^{IGBT}(2,4) & \mathbf{G}^{IGBT}(2,5) & 0 \\ \mathbf{G}^{IGBT}(4,1) & \mathbf{G}^{IGBT}(4,2) & \mathbf{G}^{IGBT}(4,4) & \mathbf{G}^{IGBT}(4,5) & 0 \\ \mathbf{G}^{IGBT}(5,1) & \mathbf{G}^{IGBT}(5,2) & \mathbf{G}^{IGBT}(5,4) & \mathbf{G}^{IGBT}(5,5) & 0 \\ \mathbf{G}^{Diode}(2,3) & 0 & 0 & 0 & \mathbf{G}^{Diode}(2,2) \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix} = \begin{bmatrix} \mathbf{i}_{eq}^{IGBT}(1)+\mathbf{i}_{eq}^{Diode}(3)-i_{eq}^{Lin} \\ -v_p/R_g-\mathbf{G}^{IGBT}(1,3)\cdot V_{dc} \\ \mathbf{i}_{eq}^{IGBT}(2)+v_p/R_g \\ -\mathbf{G}^{IGBT}(2,3)\cdot V_{dc} \\ \mathbf{i}_{eq}^{IGBT}(4)-\mathbf{G}^{IGBT}(4,3)\cdot V_{dc} \\ \mathbf{i}_{eq}^{IGBT}(5)-\mathbf{G}^{IGBT}(5,3)\cdot V_{dc} \\ \mathbf{i}_{eq}^{Diode}(2) \end{bmatrix} \tag{5.2}$$

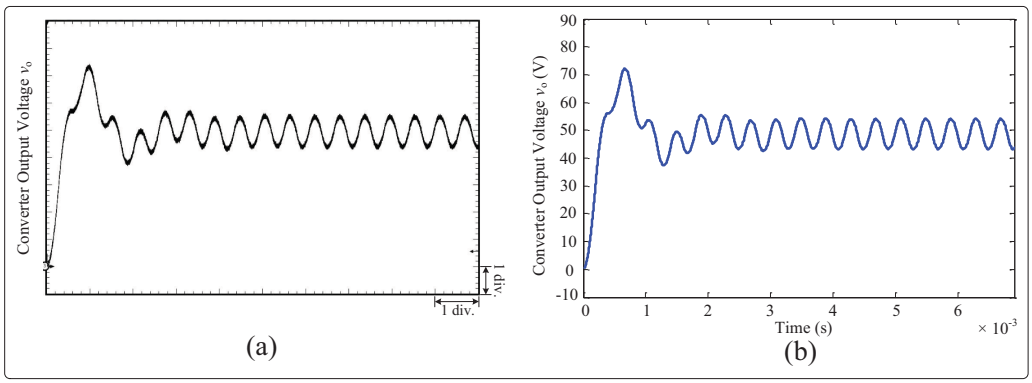

Figure 5.2: Steady-state results for the output voltage of DC-DC converter from hardware emulation (oscilloscope) and off-line simulation (Saber® software). Scale: y-axis: 10 V/div., x-axis: 4.0 ms.

5.1.1 Hardware Resources

The latency of one time-step calculation is 579 clock cycles, the highest frequency is 115MHz, and the resource utilization is listed in Table 5.2.

Table 5.2: Hardware Resources Utilized by Nonlinear Components

Resources	Diode	IGBT	Overall Circuit
FF	7763 (1.3%)	61384 (10%)	72181 (12%)
LUT	10741 (3.5%)	88061 (29%)	107264 (35%)
$Memory\ LUT$	86 (0.066%)	603 (0.46%)	729 (0.56%)
I/O	32 (4.6%)	32 (4.6%)	91 (13%)
$BRAM$	1 (0.049%)	34 (1.7%)	38 (1.8%)
$DSP48$	158 (5.6%)	1419 (51%)	1657 (59%)
$BUFG$	0 (0.0%)	0 (0.0%)	1 (3.1%)

5.1.2 Results and Comparisons

5.1.2.1 Time-Domain Results

The steady-state results for the converter output voltage v_o, the IGBT collector-emitter voltage $v_{ce}(= v_7 - v_1)$, the IGBT collector current i_c, the diode voltage $v_d(= v_6 - v_1)$, and the diode current i_d of the DC-DC converter hardware emulation under the switching frequency of 2.5kHz are shown in Figs. 5.2, 5.3. The circled areas in Fig. 5.3 (a) and (c) are the bad convergence points of Saber®. Compared with the Saber®, the steady-state results of the hardware emulation are proved to be accurate. Since Saber® did not converge when R_{M0} is set to 50Ω, the waveforms shown in Figs. 5.3, 5.4 (a)-(c) were obtained by setting R_{M0} to 0Ω (omitting forward recovery).

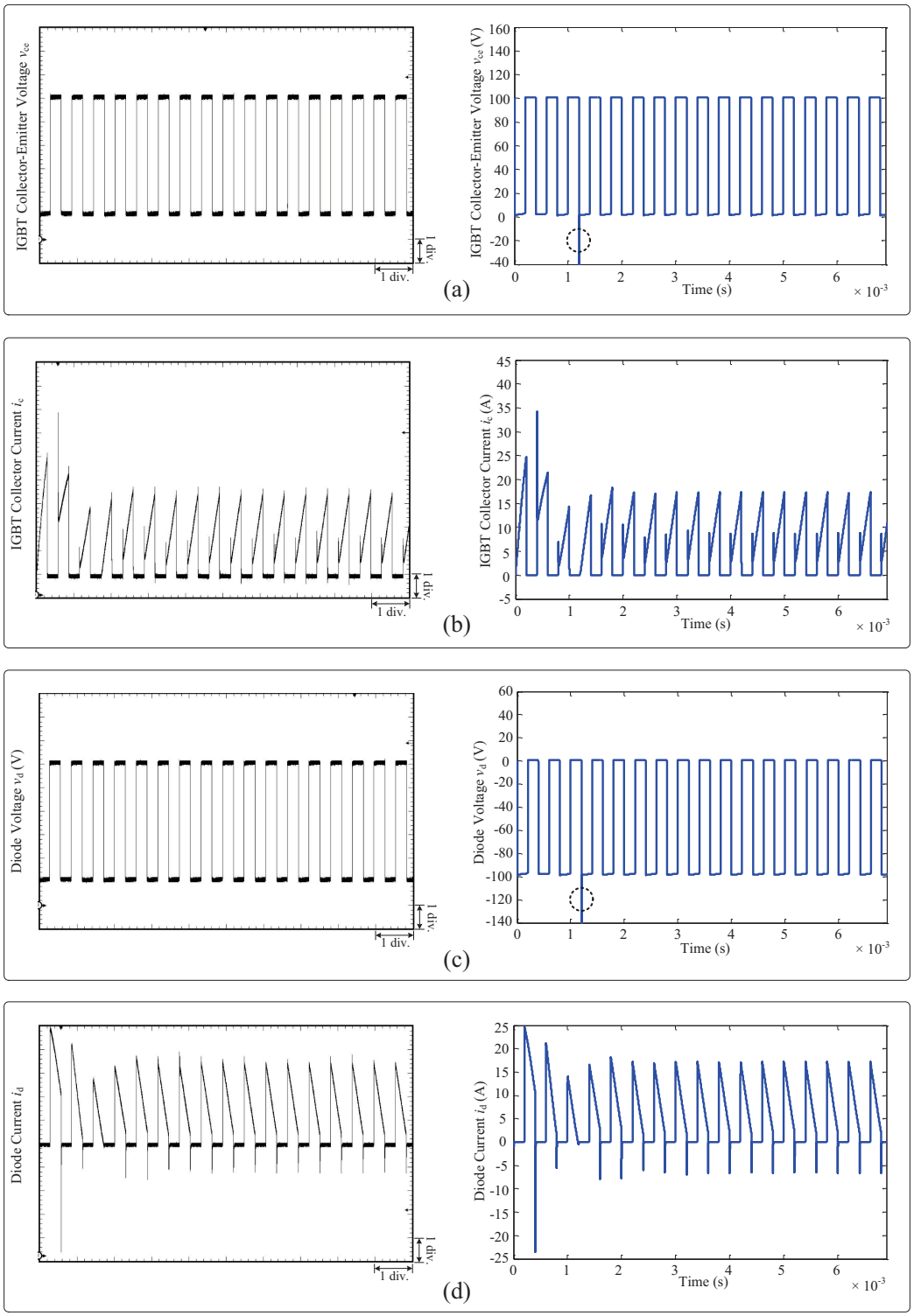

Figure 5.3: Steady-state results for the devices of DC-DC converter from hardware emulation (oscilloscope) and off-line simulation (Saber® software). Scale: (a) y-axis: 20 V/div., (b) y-axis: 5 A/div., (c) y-axis: 20 V/div., (d) y-axis: 5 A/div.; (a)-(d) x-axis: 4.0 ms.

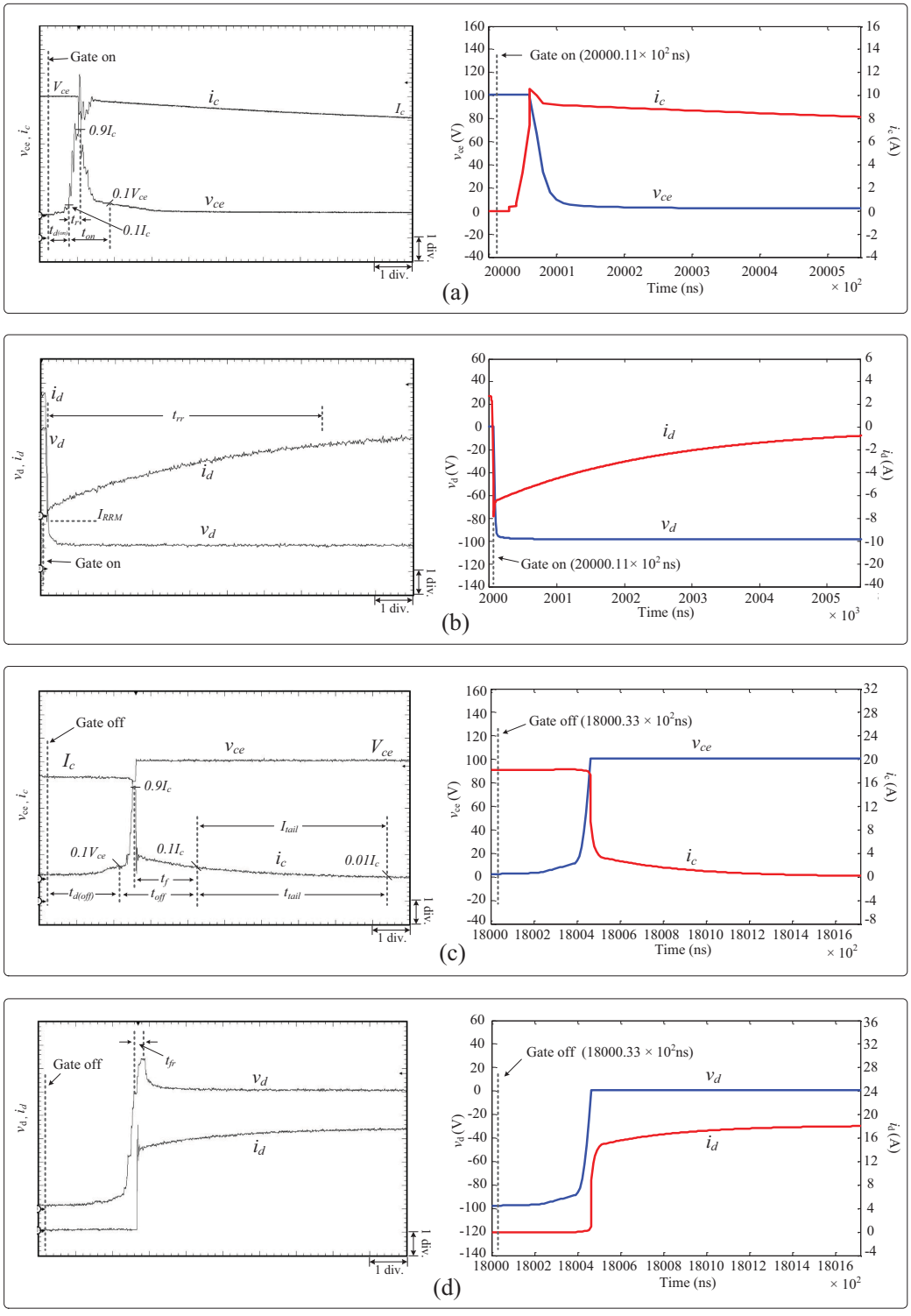

Figure 5.4: Transient results for DC-DC converter from hardware emulation (oscilloscope) and off-line simulation (Saber® software). Scale: (a) y-axis: 20 V/div. (v_{ce}), 2 A/div. (i_c), (b) 20 V/div. (v_d), 2 A/div. (i_d), (c) y-axis: 20 V/div. (v_{ce}), 4 A/div. (i_c), (d) y-axis: 20 V/div. (v_d), 4 A/div. (i_d); (a) x-axis: 320ns, (b) x-axis: 3.2μs, (c)-(d) x-axis: 1μs.

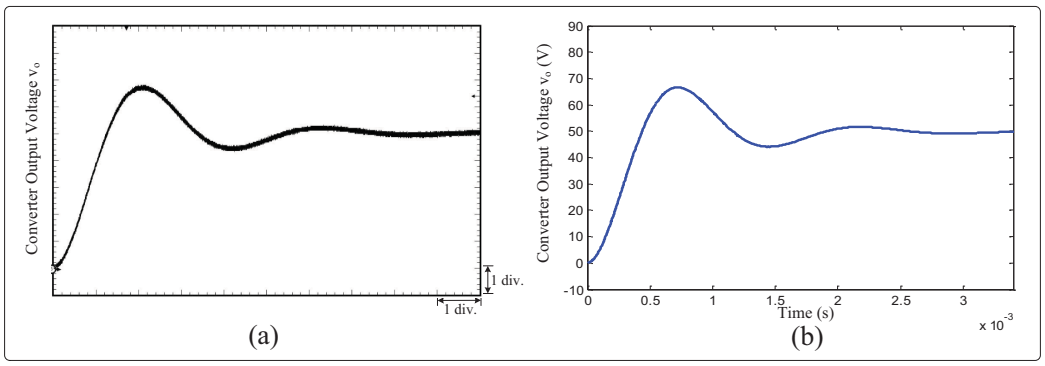

Figure 5.5: Steady-state results for the output voltage of DC-DC converter from hardware emulation (oscilloscope) and off-line simulation (Saber® software) under high switching frequency. Scale: y-axis: 10 V/div., x-axis: 12.5 ms.

The device-level transient results for (v_{ce}, i_c), and (v_d, i_d) during turn-on and turn-off switching of the DC-DC converter hardware emulation under the switching frequency of 2.5kHz are shown in Fig. 5.4. Compared with the Saber®, the device-level transient results of the hardware emulation show good agreement. Specifically, the transient waveform of diode voltage from oscilloscope in Fig. 5.4 (d) is got by setting R_{M0} to 50Ω, which showed the forward recovery clearly. The comparison of IGBT and diode switching times from Saber® and hardware emulation is shown in Table 5.3. The high-frequency results of the DC-DC converter hardware emulation under the switching frequency of 40kHz are shown in Figs. 5.5, 5.6. The maximum frequency could reach as high as around 100kHz.

5.1.2.2 Power Dissipation Analysis

The power dissipation during a switching cycle of IGBT mainly comes form the switching loss and conduction loss. The instantaneous power dissipation is calculated by multiplying the IGBT collector-emitter voltage (v_{ce}) and collector current (i_c). By choosing corresponding time range from t_0 to t_1, the energy losses in one switching cycle are obtained by integrating the instantaneous power dissipation during turn-on, turn-off and conduction period, respectively. The power loss P_{loss} is quotient of the energy loss and switching period T_s, expressed as:

$$P_{loss} = \frac{\int_{t_0}^{t_1} v_{ce}(t) \cdot i_c(t)\, dt}{T_s}. \tag{5.3}$$

The power dissipation of reverse recovery and conduction period for the diode is obtained with the same procedures. Table 5.3 shows the power dissipation results under the switching frequency of 2.5kHz from both Saber® and hardware emulation. Also, the variation of device power dissipation with switching frequency is shown in Fig. 5.7, which shows close agreement between the emulation and Saber® both at low as well as high switching frequencies proving the high-fidelity of the detailed hardware emulation. As the switching frequency increased, as expected, the IGBT switching losses $(P_{on(IGBT)}, P_{off(IGBT)})$ and

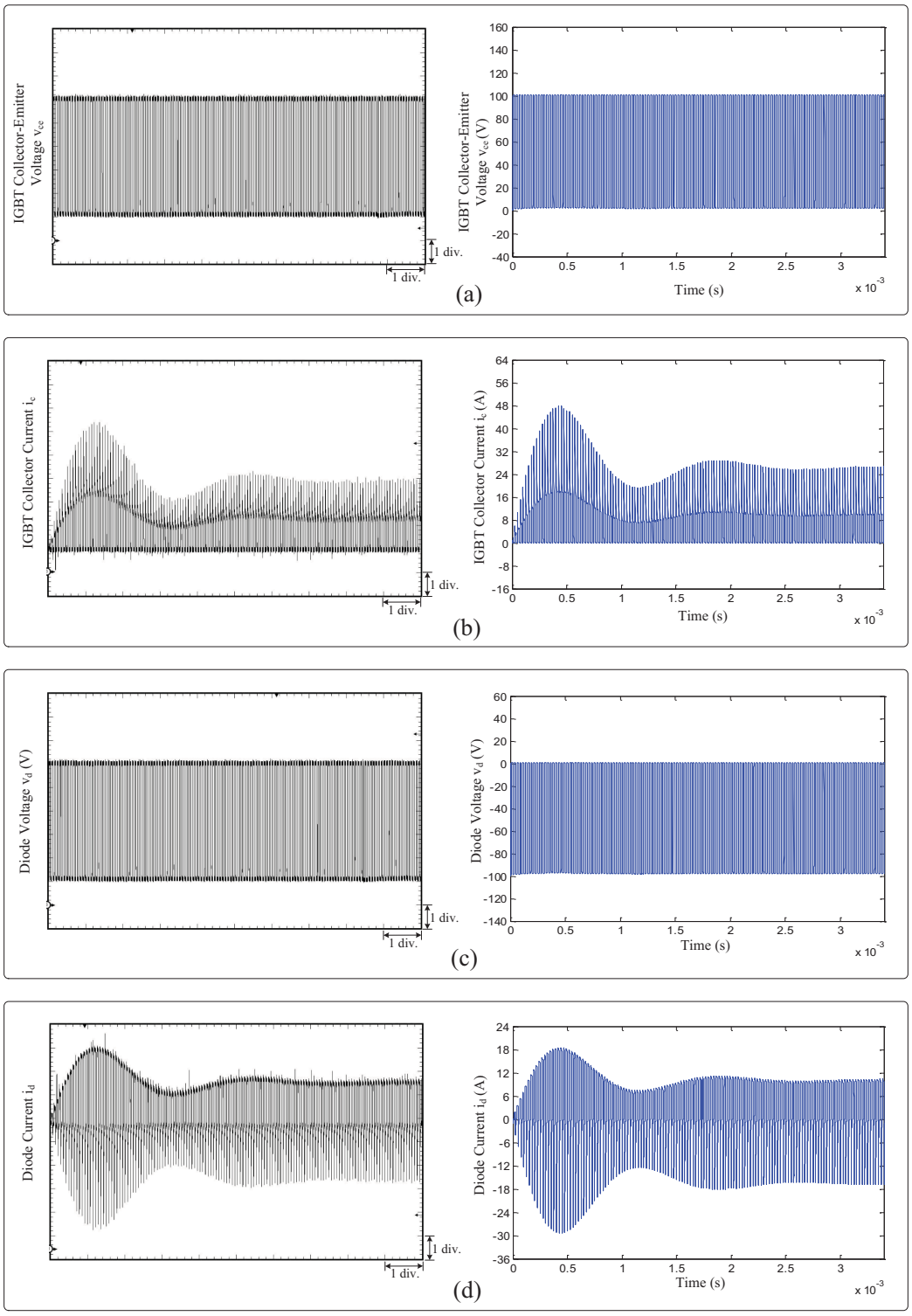

Figure 5.6: Steady-state results for devices of DC-DC converter from hardware emulation (oscilloscope) and off-line simulation (Saber® software) under high switching frequency. Scale: (a) y-axis: 20 V/div., (b) y-axis: 8 A/div., (c) y-axis: 20 V/div., (d) y-axis: 6 A/div.; (a)-(d) x-axis: 12.5 ms.

Table 5.3: Comparison of Switching Times and Power Dissipation of IGBT and Diode under a Switching Frequency of 2.5kHz

	Saber®	**FPGA**	**Error**
Switching times			
$t_{d(on)}(IGBT)$	28.8ns	29.5ns	2.43%
$t_{on}(IGBT)$	60.0ns	58.7ns	2.17%
$t_r(IGBT)$	18.6ns	17.8ns	4.30%
$t_{d(off)}(IGBT)$	327ns	334ns	2.14%
$t_{off}(IGBT)$	380ns	368ns	3.16%
$t_{tail}(IGBT)$	860ns	879ns	2.21%
$t_f(IGBT)$	285ns	293ns	2.81%
$t_{rr}(Diode)$	4009ns	4020ns	0.27%
$t_{fr}(Diode)$	n/a	42.1ns	n/a
Dissipated Power			
$P_{on}(IGBT)$	96.05mW	93.87mW	2.27%
$P_{off}(IGBT)$	539.6mW	522.5mW	3.17%
$P_{cond}(IGBT)$	11.46W	11.05W	3.58%
$P_{rr}(Diode)$	5.434W	5.507W	1.34%
$P_{cond}(Diode)$	4.768W	4.782W	0.29%

diode reverse recovery loss ($P_{rr(Diode)}$) increased significantly, however, the IGBT conduction loss ($P_{cond(IGBT)}$) increased much slowly, and the diode conduction loss ($P_{cond(Diode)}$) remained almost constant.

5.2 DC-AC Converter

For the hardware emulation of a more complicated topology of power converter such as a two-level DC-AC converter (see Fig. 5.8), which comprises 12 nonlinear components (6 IGBTs with 6 antiparallel diodes), 6 linear components (3 RL loads) and a voltage source (parameters of the circuit are also given in Table 5.4. Sinusoidal PWM strategy is applied for the switching of IGBTs), the strategy for the hardware emulation should be more sophisticated. As can be seen, the whole nonlinear circuit contain around 22 nodes (when the diode model only includes the most important reverse recovery effect to reduce circuit nodes). Admittedly, the 22 node system can be converted to 22 dimensional discrete-time linearized system, but it would take infinite time to solve the equations when implemented on the FPGA. Therefore, more efficient method should be applied to reduce the system dimension. Specifically, we divide the three-phase circuit into 3 separated legs with 3 RL loads (The loads are temporally treated as current sources, since the load current is contin-

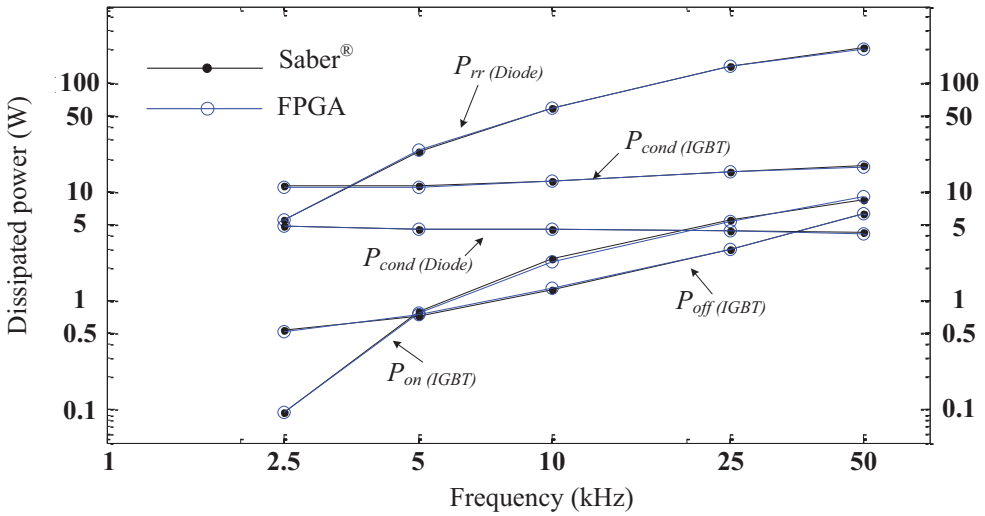

Figure 5.7: Variation of device power dissipation with switching frequency from Saber® and hardware emulation.

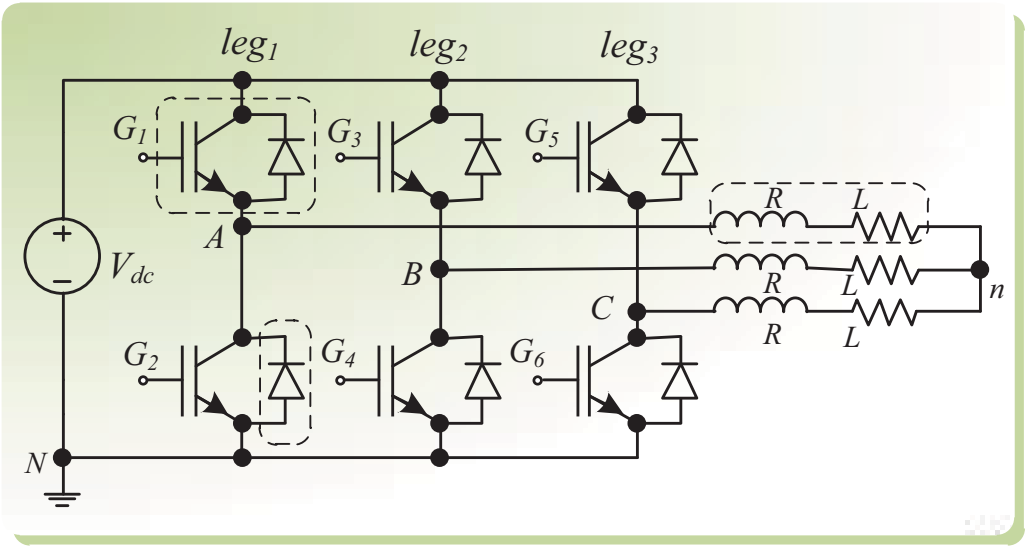

Figure 5.8: Two-level DC-AC converter circuit topology.

uous which doesn't change too much between two time-steps, and the current is a history term which can be obtained from last time-step calculation). Such division make it possible for all of the three sub-circuits to be solved in parallel. As a result, all of the 3 leg-load sub-circuit can be reduced to 3 identical 7×7 discrete-time linearized systems. The only differences of these systems are their input PWM switching signals of their IGBTs (Phases of their sinusoidal reference wave differ for $120°$). Specifically, the 7×7 discrete-time linearized system can be given as equation (5.4).

$$
\begin{bmatrix}
\mathbf{G}^{IGBT2}(2,2) & \mathbf{G}^{IGBT2}(2,3) & \mathbf{G}^{IGBT2}(2,4) & \mathbf{G}^{IGBT2}(2,5) & 0 & 0 & 0 \\
+1/R_{g2} & & & & & & \\
\mathbf{G}^{IGBT2}(3,2) & \mathbf{G}^{IGBT2}(3,3)+ \\ \mathbf{G}^{IGBT1}(1,1)+ \\ g_R^{Diode2}+ \\ g_R^{Diode1}+1/R_{g1} & \mathbf{G}^{IGBT2}(3,4) & \mathbf{G}^{IGBT2}(3,5) & \mathbf{G}^{IGBT1}(1,2) \\ -1/R_{g1} & \mathbf{G}^{IGBT1}(1,4) & \mathbf{G}^{IGBT1}(1,5) \\
\mathbf{G}^{IGBT2}(4,2) & \mathbf{G}^{IGBT2}(4,3) & \mathbf{G}^{IGBT2}(4,4) & \mathbf{G}^{IGBT2}(4,5) & 0 & 0 & 0 \\
\mathbf{G}^{IGBT2}(5,2) & \mathbf{G}^{IGBT2}(5,3) & \mathbf{G}^{IGBT2}(5,4) & \mathbf{G}^{IGBT2}(5,5) & 0 & 0 & 0 \\
0 & \mathbf{G}^{IGBT1}(2,1) & 0 & 0 & \mathbf{G}^{IGBT1}(2,2) & \mathbf{G}^{IGBT1}(2,4) & \mathbf{G}^{IGBT1}(2,5) \\
 & -1/R_{g1} & & & +1/R_{g1} & & \\
0 & \mathbf{G}^{IGBT1}(4,1) & 0 & 0 & \mathbf{G}^{IGBT1}(4,2) & \mathbf{G}^{IGBT1}(4,4) & \mathbf{G}^{IGBT1}(4,5) \\
0 & \mathbf{G}^{IGBT1}(5,1) & 0 & 0 & \mathbf{G}^{IGBT1}(5,2) & \mathbf{G}^{IGBT1}(5,4) & \mathbf{G}^{IGBT1}(5,5)
\end{bmatrix}
\cdot
\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix}
=
\begin{bmatrix}
\mathbf{i}_{eq}^{IGBT2}(2) + v_{p2}/R_{g2} \\
\mathbf{i}_{eq}^{IGBT2}(3) + i_{Req}^{Diode2} + \\
\mathbf{i}_{eq}^{IGBT1}(1) - i_{Req}^{Diode1} - \\
(\mathbf{G}^{IGBT1}(1,3) - g_R^{Diode1})V_{dc} \\
-v_{p1}/R_{g1} + i_{LR}(t-\Delta t) \\
\mathbf{i}_{eq}^{IGBT2}(4) \\
\mathbf{i}_{eq}^{IGBT2}(5) \\
\mathbf{i}_{eq}^{IGBT1}(2) + v_{p1}/R_{g1} \\
-\mathbf{G}^{IGBT1}(2,3) \times V_{dc} \\
\mathbf{i}_{eq}^{IGBT1}(4) \\
-\mathbf{G}^{IGBT1}(4,3) \times V_{dc} \\
\mathbf{i}_{eq}^{IGBT1}(5) \\
-\mathbf{G}^{IGBT1}(5,3) \times V_{dc}
\end{bmatrix}
\tag{5.4}
$$

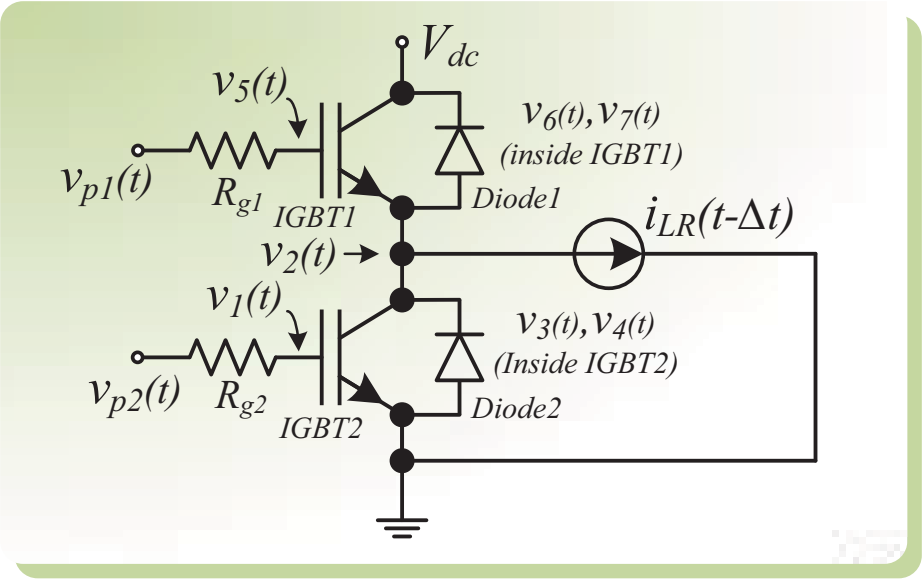

Figure 5.9: Separated single-phase sub-circuit of the two-level DC-AC converter.

Table 5.4: DC-AC Test Circuit and Device Parameters

Test Circuit Parameters
$V_{DC} = 300V$, $R = 1\Omega$, $L = 20mH$, $R_g = 100\Omega$
IGBT Parameters
Out of 32 parameters, the ones used in this paper: $K_p = 1A/V$, $K_f = 2$, $\theta = 0.01V^{-1}$, $V_{th} = 5V$, $N_B = 2 \times 10^{14}cm^{-3}$, $I_{sne} = 10^{-14}A$, $n_i = 1.45 \times 10^{10}cm^{-3}$, $\tau_{HL} = 4 \times 10^{-7}s$, $A = 0.1cm^2$, $q = 1.6 \times 10^{-19}C$, $L = 2.7 \times 10^{-3}cm$, $T = 296K$, $k = 8.617 \times 10^{-5}eV/K$, $\beta = 0.4615$, $\epsilon_{si} = 1.05 \times 10^{-12}F/cm$, $W_B = 0.01cm$, $b = 3.33$, $D_p = 11.48cm^2/s$, $BV_n = 4$, $BV_{cb0} = 3.18 \times 10^7V$, $G_{min} = 10^{-12}S$, $A_{gd} = 0.05cm^2$. The rest can be obtained from Saber®.
Diode Parameters
$I_S = 10^{-14}A$, $\tau = 50ns$, $T_M = 50ns$, $V_T = 0.0259V$

Now the whole DC-AC circuit is reduced to three identical single-phase nonlinear sub-circuits (Fig. 5.9) in parallel and one linear sub-circuit (Fig. 5.10). Since the three RL loads are identical, their conductances are the same too, given as:

$$g_{LR}^{leg1}(t) = g_{LR}^{leg2}(t) = g_{LR}^{leg3}(t) = \frac{\Delta t}{2L + \Delta t \cdot R}, \tag{5.5}$$

and their equivalent currents are given as:

$$i_{LReq}^{leg1}(t) = \frac{[v_2^{leg1}(t - \Delta t) - v_n(t - \Delta t)]\Delta t}{2L + \Delta t \cdot R} + \frac{i_{LR}^{leg1}(t - \Delta t) \cdot (2L - \Delta t \cdot R)}{2L + \Delta t \cdot R}, \tag{5.6}$$

$$i_{LReq}^{leg2}(t) = \frac{[v_2^{leg2}(t - \Delta t) - v_n(t - \Delta t)]\Delta t}{2L + \Delta t \cdot R} + \frac{i_{LR}^{leg2}(t - \Delta t) \cdot (2L - \Delta t \cdot R)}{2L + \Delta t \cdot R}, \tag{5.7}$$

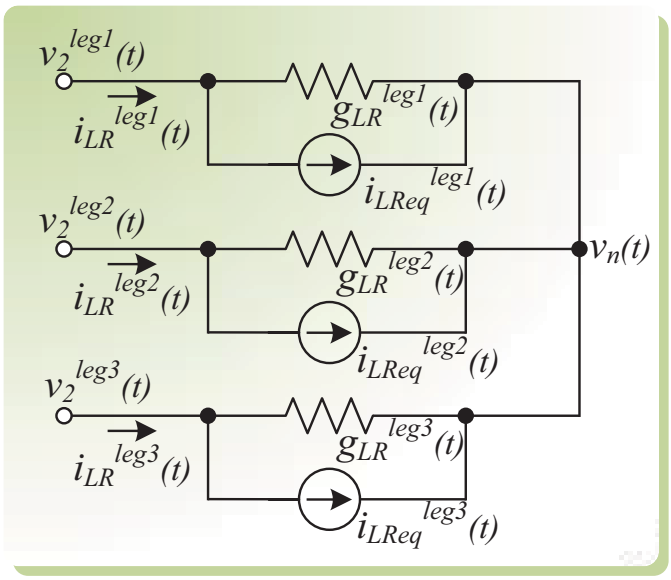

Figure 5.10: Separated linear sub-circuit of the two-level DC-AC converter.

and

$$i_{LReq}^{leg3}(t) = \frac{[v_2^{leg3}(t-\Delta t) - v_n(t-\Delta t)]\Delta t}{2L + \Delta t \cdot R} + \frac{i_{LR}^{leg3}(t-\Delta t) \cdot (2L - \Delta t \cdot R)}{2L + \Delta t \cdot R}, \tag{5.8}$$

respectively. The linear sub-circuit is only a one-dimension system and the neural point voltage v_n can be directly solved as:

$$v_n(t) = \frac{i_{LReq}^{leg1}(t) + i_{LReq}^{leg2}(t) + i_{LReq}^{leg3}(t)}{g_{LR}^{leg1}(t) + g_{LR}^{leg2}(t) + g_{LR}^{leg3}(t)} + \frac{v_2^{leg1}(t) + v_2^{leg2}(t) + v_2^{leg3}(t)}{3}. \tag{5.9}$$

The currents of the three legs are represented as:

$$i_{LR}^{leg1}(t) = \frac{[v_2^{leg1}(t) - v_n(t)]\Delta t}{2L + \Delta t \cdot R} + i_{LReq}^{leg1}(t), \tag{5.10}$$

$$i_{LR}^{leg2}(t) = \frac{[v_2^{leg2}(t) - v_n(t)]\Delta t}{2L + \Delta t \cdot R} + i_{LReq}^{leg2}(t), \tag{5.11}$$

and

$$i_{LR}^{leg3}(t) = \frac{[v_2^{leg3}(t) - v_n(t)]\Delta t}{2L + \Delta t \cdot R} + i_{LReq}^{leg3}(t), \tag{5.12}$$

respectively.

As can be seen from the resource utilization of the DC-DC converter, which contains only one IGBT and diode, the used DSP48s made up more than half of its total number of the whole FPGA board while the unused other resources were abundant. The potential utilization of DSP48 usage for a fully parallelized 2-level DC-AC converter hardware model will surely overpass the limit of the FPGA board. In this work, the previously parallelized three legs of the DC-AC converter are implemented in a sequential way to save hardware resources, especially DSP48s. More specifically, a single leg hardware module

is repetitively used to generate the results of all three legs. However, even the potential utilization of only one leg, which contains 2 IGBTs and 2 diodes, would exceed the board limit. Therefore, special design strategy to save DSP48 resource usages must be applied to fit the one leg to the Xilinx® Virtex®-7 XC7VX485T FPGA board. Thankfully, the previous hardware design of the DC-DC converter was emphasizing on reducing the latency of all hardware components in disregard of the hardware usages, which means many hardware resources could be saved at the cost of longer latencies of some inner components of the hardware model. Actually, many hardware modules inside the whole converter model do not contribute to its whole latency. These modules include the Power Diode Module and the Linear Component Module as well as some inner hardware units of the IGBT Module. In practice, all these inner components of converter hardware model are redesigned with less DSP slices. Although the usages of some other hardware resources (e.g. FFs and LUTs) will increase correspondingly, they are deliberately restricted to within the limit of the FPGA board. As a result, the whole DC-AC converter hardware model is successfully fitted into the FPGA board.

The execution time for the off-line simulation of the converter under 1kHz switching frequency using Saber® for 100ms using variable time-step strategy (with an initial time-step of 10ns and maximum time-step of 1μs) was 80.6s, while the hardware emulation time for 100ms of simulation time was around 2.3s. Therefore, the speed-up is more than 35 times. It is predicable that the speed-up could overpass 100 times if the fully parallelism strategy of all three legs of the DC-AC converter are applied in this work with larger FPGA board. Compared with 30-time speed-up for the DC-DC converter, it now proves that off-line simulation of converter circuits with more IGBTs and diodes employing detailed device-level models would be much slower than hardware emulation.

The latency of one time-step calculation is 1982 clock cycles, the highest frequency is 115MHz, and the resource utilization is listed in Table 5.5.

Table 5.5: Hardware Resources Utilized by Nonlinear Components

Resources	**Diode**	**IGBT**	**Overall Circuit**
FF	56718 (9.3%)	227408 (37%)	300221 (49%)
LUT	45131 (15%)	214763 (71%)	278051 (92%)
$Memory\ LUT$	459 (0.70%)	2156 (1.6%)	2307 (1.8%)
I/O	64 (9.1%)	64 (9.1%)	155 (22%)
$BRAM$	2 (0.097%)	68 (3.3%)	81 (3.9%)
$DSP48$	169 (6.0%)	2107 (75%)	2336 (83%)
$BUFG$	0 (0.0%)	0 (0.0%)	1 (3.1%)

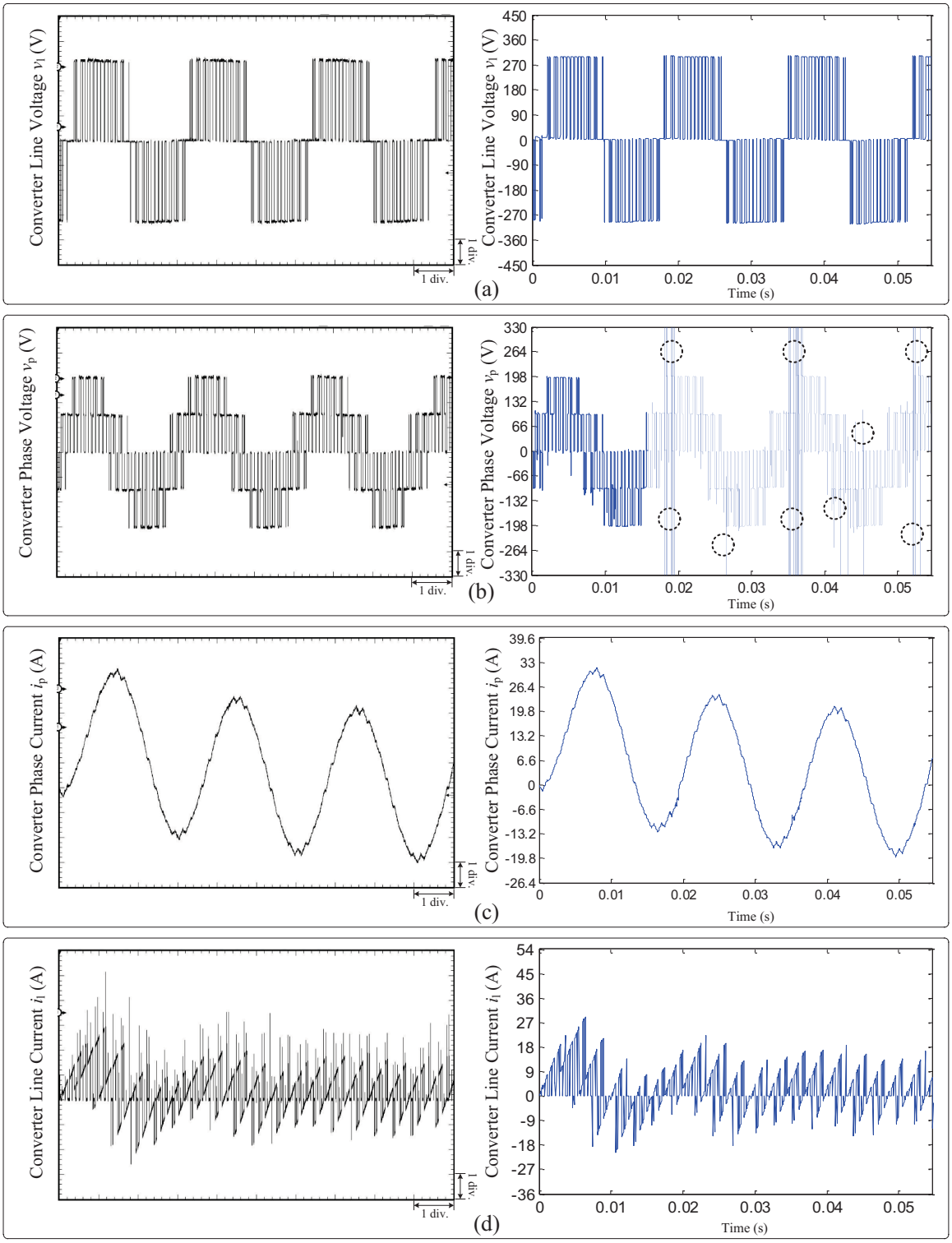

Figure 5.11: Steady-state results for the load voltage and current of DC-AC converter from hardware emulation (oscilloscope) and off-line simulation (Saber® or Simulink® software). Scale: (a) y-axis: 90 V/div., (b) y-axis: 66 V/div., (c) y-axis: 6.6 A/div., (d) y-axis: 9 A/div.; (a)-(d) x-axis: 125 ms.

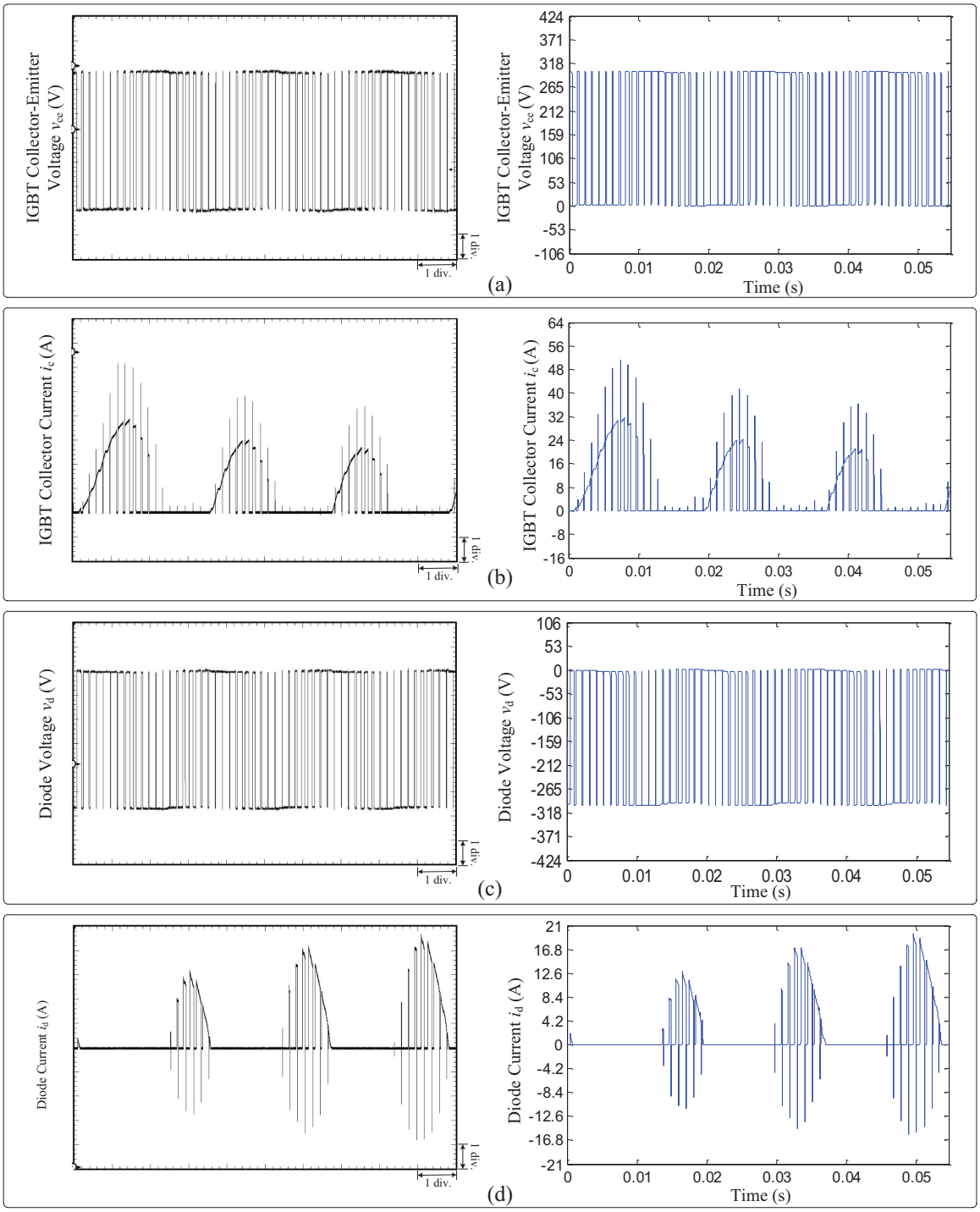

Figure 5.12: Steady-state results for the devices of DC-AC converter from hardware emulation (oscilloscope) and off-line simulation (Saber® software). Scale: (a) y-axis: 53 V/div., (b) y-axis: 8 A/div., (c) y-axis: 53 V/div., (d) y-axis: 4.2 A/div.; (a)-(d) x-axis: 125 ms.

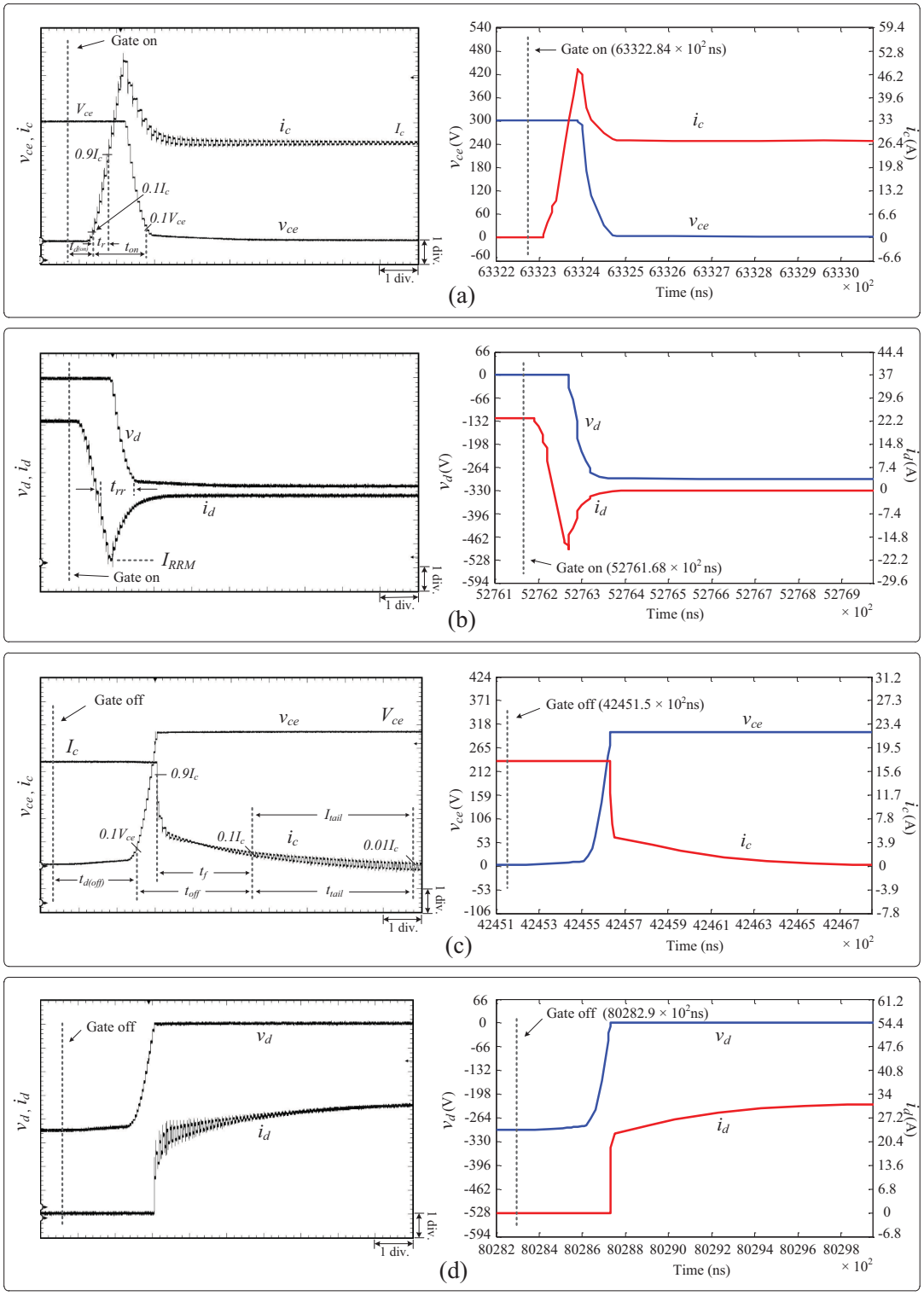

Figure 5.13: Transient results for DC-AC converter from hardware emulation (oscilloscope) and off-line simulation (Saber® software). Scale: (a) y-axis: 60 V/div. (v_{ce}), 6.6 A/div. (i_c), (b) 66 V/div. (v_d), 7.4 A/div. (i_d), (c) y-axis: 53 V/div. (v_{ce}), 3.9 A/div. (i_c), (d) y-axis: 66 V/div. (v_d), 6.8 A/div. (i_d); (a)-(b) x-axis: 2μs, (c)-(d) x-axis: 4μs.

5.2.1 Results and Comparisons

5.2.1.1 Time-Domain Results

Selected load and devices (see the dashed boxes in Fig.5.8) of the DC-AC converter will have their test results in this section to be compared with Saber®. More specifically, the steady-state results for the phase voltage $v_p(= v_2^{leg1} - v_n)$ and current i_p, the line voltage $v_l(= v_2^{leg1} - v_2^{leg2})$ and current i_l of the DC-AC converter hardware emulation under the switching frequency of 1kHz are shown in Fig. 5.11. The dash circled areas in Fig. 5.11(b) are bad convergence points of Saber®. Obviously, even a sophisticated circuit simulator as Saber® cannot handle the DC-AC converter very well. Although Saber® has mature time-stepping and transient simulation techniques, its solution approaches are based on general (but inflexible) methodologies. In contrast, the hardware emulation applies parallel strategies to reduce the system matrix size wherever possible, so its results could even better Saber®. The off-line software used in Fig. 5.11(d) is Simulink® rather than Saber® because the latter has implicit method to select initial conditions for the second and third legs of the converter and shows different results for the line current. Since Simulink® utilizes simpler system-level models for the switching devices, the differences in the transient dynamics of the results between Simulink® and hardware emulation are unavoidable. In Fig. 5.12, the upper IGBT collector-emitter voltage $v_{ce}(= V_{dc} - v_2^{leg1})$ and collector current i_c, the upper diode voltage $v_d(= v_2^{leg1} - V_{dc})$ and current i_d of the DC-AC converter hardware emulation under the switching frequency of 1kHz are shown too. Compared with Saber® or Simulink®, the steady-state results of the hardware emulation are proved accurate.

The device-level transient results for (v_{ce}, i_c) of upper IGBT, and (v_d, i_d) of lower diode during turn-on and turn-off switching of the DC-AC converter hardware emulation under the switching frequency of 1kHz are shown in Fig. 5.13. Compared with Saber®, the device-level transient results of the hardware emulation show good agreement. The variation of device power dissipation with switching frequency is shown in Fig. 5.14, which shows close agreement between the emulation and Saber® both at low as well as high switching frequencies proving the high-fidelity of the detailed hardware emulation. As the switching frequency increased, as expected, the IGBT switching losses $(P_{on(IGBT)}, P_{off(IGBT)})$ and diode reverse recovery loss $(P_{rr(Diode)})$ increased significantly, however, the IGBT conduction loss $(P_{cond(IGBT)})$ increased much slowly, and the diode conduction loss $(P_{cond(Diode)})$ remained almost constant.

5.3 Summary

This chapter provided the hardware emulation of 2 case studies: DC-DC converter and DC-AC converter. These two case studies utilized the IGBT and diode physics-based hardware models developed in the previous chapter. The Newton-Raphson linearization is very advantageous in power converter hardware emulations, which may involve all kind-

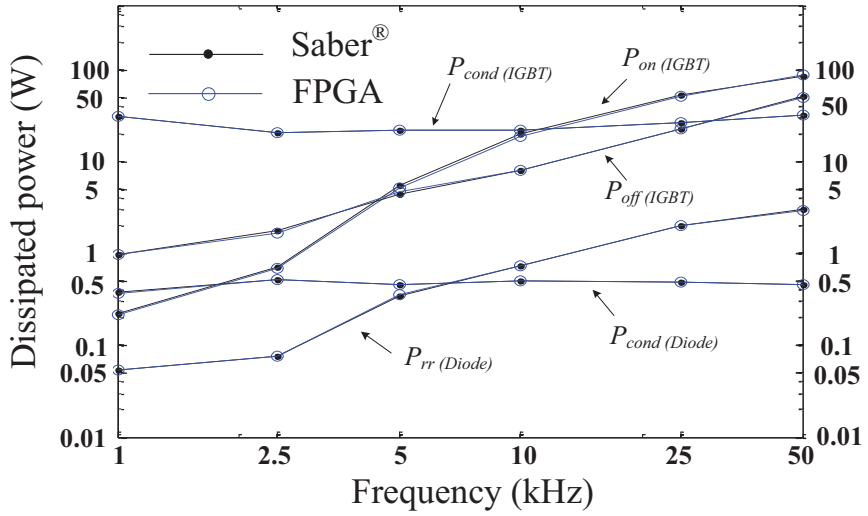

Figure 5.14: Variation of device power dissipation with switching frequency for DC-AC converter from Saber® and hardware emulation.

s of different topologies and this method is quite useful due to its strong flexibility and convenience in the linearized model formulation. The variable time-step strategy makes the hardware emulations with power converter based on physics-based analytical model become possible, which not only ensures the speed of the emulation but also the detailed switching behaviours. Therefore, the emulated models were able to predict the dissipated energy in the power electronic circuit quite accurately. Comparison with Saber® both of low and high switching frequencies proves that the emulated models are accurate and efficient in terms of speed of execution. While the hardware emulation is still slower than real-time execution to be applied for closed-loop HIL simulation, with newer FPGA generations, and further efficiency refinements in the numerical solution, that goal appears feasible. Presently, the proposed hardware models can be readily used for open-loop HIL applications and for circuit design acceleration that involve statistical, parametric, and sensitivity analyses.

The relative errors between the hardware emulation and Saber® have several reasons. The first one can be attributed to the modeling error. Specifically, the power diode model used by Saber® is more sophisticated than Lauritzen's diode model used in this work. For the IGBT model, although both Saber® and the hardware emulation use Hefner's model, Saber®'s model is more refined judging from its larger parameter set. Another reason for the error is the limit of the 32-bit floating-point number used in the hardware emulation, which is very accurate under ordinary scenarios but still restrictive when used to emulate the highly complicated physics-based IGBT and power diode model. Also, some specific methods used in the hardware emulation to deal with the nonlinearity of the model (e.g. look-up tables) although performed very fast, but at the cost of certain level of accuracy.

6 Conclusions and Future Work

The focus of this thesis is to develop detailed device-level real-time models of power electronic systems for hardware-in-the-loop simulation. This work proposed a digital hardware emulation of physics-based device-level IGBT and diode models. Both these models are fully paralleled on the FPGA. The hardware emulation of both devices is based on a unified numerical framework, and can be extended in a straightforward fashion to model complete power electronic converter circuits. The main contributions of this research are given below.

6.1 Contributions of this Thesis

- The concept of modularization is introduced to the physics-based device-level power electronic circuit hardware emulation. Therefore, the hardware designs of the power electronic devices (e.g. IGBT and diode) are separated from the power converter circuit. Moreover, the device hardware modules can be fully reused and fit into different topologies of power converter circuits, which reduce a great deal of repetitive work in the future.

- A complete description of the paralleled IGBT and power diode hardware designs is presented, which is beneficial for the hardware emulation of power electronic converter circuits with device-level models. For example, this thesis described many practical methods (e.g. LUTs) to tackle the complicated nonlinearity inside the device physics. Also, the inner hardware structures of the device modules exploited the potential of parallelization as much as possible, which significantly reduced the latencies of the whole converter circuit.

- The concept of variable time-step strategy is introduced to the field of HIL simulation, and the related hardware modules of the Variable Time-Step Control Module (VTCM) and the Variable Time-Step Output Module (VTOM) have been developed.
- A fast parallel Gauss-Jordan Linear Solver is introduced in this thesis, which make it possible to solve a large matrix with a low latency. This is especially beneficial to device-level power converter hardware emulations, since they usually have larger system matrices that need solution.
- Validation is made using Saber®, a device-level commercial simulation software. Steady-state and transient results are observed as well as power analysis. All of them show excellent agreement.

6.2 Recommendations for Future Work

- Although Newton-Raphson method is one of the most classical and effective methods to find the solution of nonlinear circuit equations, it does have drawbacks. For example, its convergence is sensitive to the initial conditions. This would largely limit the size of the selected circuit emulation time-step, since the initial guess is directly coming from the solution of the last time-step. In order to keep the Newton's method from diverging, the selected time-step could not be chosen to be too large. As a result, the emulation time of the whole power converter circuit has been prolonged. Therefore, future work in this field can be directed to find ways to improve the Newton-Raphson's convergence.
- Although this thesis has learnt a lot from popular off-line software-based circuit simulators, there are still more things to study. For example, this thesis has made a successful first attempt to applied the variable time-step strategy into the hardware emulation, which largely sped up the power electronic emulation time. However, the inner mechanism of VTCM to adjust the size of the time-step is based on the Newton iteration time of the last time-step, which is not very reliable if the power electronic circuit dynamics at some point changes too fast between successive time-steps. In contrast, Saber® adjusts the size of its time-step according to local truncation error (LTE) of the current time-step calculation. If the LTE exceeds the limit, it will reduce the time-step and the LTE is revaluated . Moreover, in order to overcome the shortcoming of Newton-Raphson method, Saber® makes the initial guess of the solution using an implicit method. If the Newton's method does fail to converge, Saber® would restart the calculation at the non-convergence point by finding the DC initial values. All these smart strategies can be introduced into the field of hardware emulation, if proper hardware modules are designed in the future.

- Regarding the parallel Gauss-Jordan linear solver, almost half of its latency came from the operation of divisions, since the 32-bit floating-point dividers are always several times slower than adders, subtractors, and multipliers. If the dividers are evaded or their implementation improved in the future, then the linear solver can improve its speed as well.
- The Hefner's IGBT model applied in this thesis is based on the non-buffer layer IGBT model. Actually, there are other types of physics-based IGBT models (e.g. IGBT with thermal model) as well as for the power diode, which can be applied into the realm of power electronic circuit hardware emulation.

Bibliography

[1] W. Ren, M. Sloderbeck, M. Steurer, V. Dinavahi, T. Noda, S. Filizadeh, A. R. Chevrefils, M. Matar, R. Iravani, C. Dufour, J. Belanger, M. O. Faruque, K. Strunz, and J. A. Martinez "Interfacing issues in real-time digital simulators", *IEEE Trans. Power Delivery,* vol. 26, no. 2, pp. 1221-1230, Apr. 2011.

[2] V. Dinavahi, R. Iravani, and R. Bonert, "Real-time digital simulation of power electronic apparatus interfaced with digital controllers", *IEEE Trans. Power Delivery,* vol. 16, no. 4, pp. 775-781, Oct. 2001.

[3] V. Dinavahi, R. Iravani, and R. Bonert, "Design of a real-time digital simulator for a D-STATCOM system", *IEEE Trans. Ind. Electron.,* vol. 51, no. 5, pp. 1001-1008, Oct. 2004.

[4] H. Li, M. Steurer, S. Woodruff, K. L. Shi, and D. Zhang, "Development of a unified design, test, and research platform for wind energy systems based on hardware-in-the-loop real time simulation", *IEEE Trans. Ind. Electron.,* vol. 53, no. 4, pp. 1144-1151, Jun. 2006.

[5] B. Lu, X. Wu, H. Figueroa, and A. Monti, "A low-cost real-time hardwarein-the-loop testing approach of power electronics controls", *IEEE Trans. Ind. Electron.,* vol. 54, no. 2, pp. 919-931, Apr. 2007.

[6] A. Bouscayrol, X. Guillaud, R. Teodorescu, P. Delarue, and B. Lemaire-Semail, "Energetic macroscopic representation and inversion-based control illustrated on a wind-energy-conversion system using hardware-in-the-loop simulation", *IEEE Trans. Ind. Electron.,* vol. 56, no. 12, pp. 4826-4835, Dec. 2009.

[7] M. O. Faruque and V. Dinavahi, "Hardware-in-the-loop simulation of power electronic systems using adaptive discretization", *IEEE Trans. Ind. Electron.,* vol. 57, no. 4, pp. 1146-1158, Apr. 2010.

[8] E. Adzic, M. Adzic, V. Katic, D. Marcetic, and N. Celanovic, "Development of high-reliability EV and HEV IM propulsion drive with ultra-low latency HIL environment", *IEEE Trans. Ind. Informat.,* vol. 9, no. 2, pp. 630-639, May 2013.

[9] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications", *IEEE Trans. Ind. Informat.,* vol. 7, no. 2, pp. 224-243, May 2011.

[10] Z. Ma, J. Gao, R. Kennel, "FPGA implementation of a hybrid sensorless control of SMPMSM in the whole speed range", *IEEE Trans. Ind. Informat.,* vol. 9, no. 3, pp. 1253-1261, Aug. 2013.

[11] M. Brox, S. Sanchez-Solano, E. del Toro, et al, "CAD tools for hardware implementation of embedded fuzzy systems on FPGAs", *IEEE Trans. Ind. Informat.,* vol. 9, no. 3, pp. 1635-1644, Aug. 2013.

[12] I. Bahri, L. Idkhajine, E. Monmasson, M. El Amine Benkhelifa, "Hardware/software codesign guidelines for system on chip FPGA-based sensorless AC drive applications", *IEEE Trans. Ind. Informat.,* vol. 9, no. 4, pp. 2165-2176, Nov. 2013.

[13] Y. Chen and V. Dinavahi, "FPGA-based real-time EMTP", *IEEE Trans. Power Delivery,* vol. 24, no. 2, pp. 892-902, Apr. 2009.

[14] D. Majstorovic, I. Celanovic, N. D. Teslic, N. Celanovic, and V. A. Katic, "Ultralow-latency hardware-in-the-loop platform for rapid validation of power electronics designs", *IEEE Trans. Ind. Electron.,* vol. 58, no. 10, pp. 4708-4816, Oct. 2011.

[15] Y. Chen and V. Dinavahi, "An iterative real-time nonlinear electromagnetic transient solver on FPGA", *IEEE Trans. Ind. Electron.,* vol. 58, no. 6, pp. 2547-2555, June 2011.

[16] A. Myaing, M.O. Faruque, V. Dinavahi, and C. Dufour, "Comparison of insulated gate bipolar transistor models for FPGA-based real-time simulation of electric drives and application guideline", *IET Power Electron.,* vol. 5, iss. 3, pp. 293-303, 2012.

[17] Y. Chen and V. Dinavahi, "Digital hardware emulation of universal machine and universal line models for real-time electromagnetic transient simulation", *IEEE Trans. Ind. Electron.,* vol. 59, no. 2, pp. 1300-1309, Feb. 2012.

[18] Y. Chen and V. Dinavahi, "Multi-FPGA digital hardware design for detailed large-scale real-time electromagnetic transient simulation of power systems", *IET Gener. Transm. Distrib.,* vol. 7, iss. 5, pp. 451-463, 2013.

[19] Y. Chen and V. Dinavahi, "Hardware emulation building blocks for real-time simulation of large-scale power grids", *IEEE Trans. Ind. Informat.,* vol. 10, no. 1, pp. 373-381, Feb. 2014.

[20] Y. Wang and V. Dinavahi, "Low-latency distance protective relay on FPGA", *IEEE Trans. on Smart Grid*, vol. 5, no. 2, pp. 896-905, Mar. 2014.

[21] J. Liu and V. Dinavahi, "A real-time nonlinear hysteretic power transformer transient model on FPGA", *IEEE Trans. Ind. Electron.,* vol. 61, no. 7, pp. 3587-3597, Jul. 2014.

[22] N. R. Tavana and V. Dinavahi, "A general framework for FPGA-based real-time emulation of electrical machines for HIL applications", *IEEE Trans. Ind. Electron.*, pp. 1-12, 2014.

[23] G. G. Parma, V. Dinavahi, "Real-time digital hardware simulation of power electronics and drives", *IEEE Trans. Power Delivery,* vol. 22, no. 2, pp. 1235-1246, Apr. 2007.

[24] A. Myaing and V. Dinavahi, "FPGA-based real-time emulation of power electronic systems with detailed representation of device characteristics", *IEEE Trans. Ind. Electron.,* vol. 58, no. 1, pp. 358-368, Jan. 2011.

[25] A. R. Hefner and D. L. Blackburn, "An analytical model for the steady state and transient characteristics of the power insulated-gate bipolar transistor", *Solid State Electronics,* vol. 31, no. 10, pp. 1513-1532, May 1988.

[26] A. R. Hefner and D. M. Diebolt, "An experimentally verifed IGBT model implemented in the Saber circuit simulator", *IEEE Trans. Power Electron.,* vol. 9, no. 5, pp. 532-542, September 1994.

[27] A. R. Hefner, "Modeling buffer layer IGBTs for circuit simulation", *IEEE Trans. Power Electron.,* vol. 10, no. 2, pp. 111-123, Mar. 1995.

[28] R. Kraus and K. Hoffmann, "An analytical model of IGBT's with low emitter efficiency", in *Proc. ISPSD,* Monterey, CA, 1993, vol. 3, pp. 30-34.

[29] P. O. Lauritzen and C. L. Ma, "A simple diode model with reverse recovery", *IEEE Trans. Power Electron.,* vol. 6, pp. 188-191, Apr. 1991.

[30] C. L. Ma and P. O. Lauritzen, "A simple power diode model with forward and reverse recovery", *IEEE Trans. Power Electron.,* vol. 8, no. 4, pp. 342-346, 1993.

[31] J. T. Hsu and K. D. Ngo, "Behavioral modeling of the IGBT using the Hammerstein configuration", *IEEE Trans. Power Electron.,* vol. 11, no. 6, pp. 746-754, Nov. 1996.

[32] C. Wong, "EMTP modeling of IGBT dynamic performance for power dissipation estimation", *IEEE Trans. Ind. Appl.,* vol. 33, no. 1, pp. 6471, Jan./Feb. 1997.

[33] B. Allard, H. Morel, S. Ghedira, and A. Ammous, "Building advanced averaged models of power converters from switched bond graph representation", *Soc. Comput. Simul., Simulation Ser.,* vol. 31, no. 1, pp. 331-338, 1999.

[34] B. J. Baliga, "Analytical Modeling of IGBTs: Challenges and Solutions", *IEEE Trans. Electron. Devices,* vol. 60, no. 2, pp. 535-543, Feb. 2013.

[35] K. Sheng, B. W. Williams and S. J. Finney, "A review of IGBT models", *IEEE Trans. Power Electron.,* vol. 15, pp. 1250-1266, Nov. 2004.

[36] Z. Zhou and V. Dinavahi, "Parallel massive-thread electromagnetic transient simulation on GPU", *IEEE Trans. Power Delivery,* vol. 29, no. 3, pp. 1045-1053, Jun. 2014.

[37] G. T. Oziemkiewicz, "Implementation and Development of the NIST IGBT Model in a SPICE-Based Commercial Circuit Simulator", Master of Science Thesis at University of Florida, 1995.

[38] "*7 series FPGAs clocking resources user guide*," Xilinx, Inc., USA, May. 2014.

[39] "*7 series FPGAs configurable logic block user guide*," Xilinx, Inc., USA, Aug. 2014.

[40] "*7 series FPGAs memory resources user guide*," Xilinx, Inc., USA, May. 2014.

[41] "*7 series DSP48E1 slice user guide*," Xilinx, Inc., USA, May. 2014.

[42] "*7 series FPGAs SelectIO resources user guide*," Xilinx, Inc., USA, May. 2014.

[43] "*Vivado design suite user guide design flows overview*," Xilinx, Inc., USA, Apr. 2014.

[44] H. Mantooth and M. Vlach, "Beyond SPICE with Saber and MAST," *Proc. IEEE Int. Symp. Circuits Syst.*, May 1992, vol. 1, pp. 77-80.

[45] C. K. Chuang and C. G. Harrison, "Analogue behavioural modelling and simulation using VHDL and Saber-MAST," *Proc. IEE Colloq. Mixed Mode Modelling and Simul.*, Nov. 1994, pp. 1/1-1/5.

[46] "*SPICE/MultiSim Tutorial*," EECS Department, UC Berkeley, USA, 2007.

[47] Introduction to OrCAD Capture and PSpice, Sep 2008. [Online]. Available: http://userweb.eng.gla.ac.uk/john.davies/orcad/spiceintro160.pdf

[48] I. M. Wilson, "Analog behavioral modeling using PSpice," *Proceedings of the 32rd Midwest Symposium on Circuits and Systems,* , vol. 2, pp. 981-984, Aug. 1989.

[49] Hierarchical Full-chip Circuit Simulation and Analysis, 2012. [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification/ CircuitSimulation/HSIM/Pages/default.aspx

[50] JSPICE, 2008. [Online]. Available: http://www.eecs.berkeley.edu/IPRO/Software/Description/jspice.html

[51] MacSpice User's Guide. [Online]. Available: http://www. macspice.com/ug/

[52] "*XSpice Software User's Manual*," Georgia Tech Research Corporation, USA, Dec. 1992.

[53] "*NGSPICE User Manual*," University of California, USA, 1996.

[54] HSPICE, 1999. [Online]. Available: http://www.stanford.edu/class/ee133/spice/HSpice.pdf

[55] L. Trajkovic, E. fung, and S. Sanders, "HomSPICE: simulator with homotopy algorithms for finding dc and steady-state solutions of nonlinear circuits," *Proc. IEEE Int. Symp. Circuits Syst.*, 1998, vol. 6, pp. 227-231.

[56] "*PSIM user guide*," Powersim Inc., Jun. 2003.

[57] J. H. Allmeling and W. P. Hammer, "PLECS-piece-wise linear electrical circuit simulation for Simulink," *Proc. IEEE Int. Conf. Power Electron. Drive Syst.*, 1999, vol. 1, pp. 355-360.

[58] A. K. Palit, K. K. Duganapalli, and W. Anheier, "XSIM: an effecient crosstalk simulator for analysis and modeling of signal integrity faults in both defective and defect-free interconnects," *IEEE Design Diagnostics Electro. Circuit Syst.*, Apr. 2007, pp. 1-4.

[59] Z. Cheng, "The application of Multisim in power electronic technology," *Int. Conf. Multimedia Technol.*, Jul. 2011, pp. 813-816.

[60] Qucs project. [Online]. Available: http://qucs.sourceforge.net/

[61] D. Li, R. Tymerski, and T. Ninomiya, "PECS-power electronics circuit simulator," *7th Workshop Computer Power Electron.*, 2000, pp. 159-165.

[62] U. Feldmann and R. Schultz, "TITAN: an universal circuit simulator with event control for latency exploitation," *4th European Solid-State Circuits Conf.*, Sep. 1988, pp. 183-185.

[63] P. Pejovic and D. Maksimovic, "PETS-a simulation tool for power electronics," *IEEE Workshop on Computer in Power Electron.*, Aug. 1996, pp. 1-8.

[64] "*Spectre circuit simulator reference*," Cadence Design Systems, Inc., Sep. 2003.

[65] W. L. Engl, R. Laur, and H. K. Dirks, "MEDUSA-a simulator for modular circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Syst.*, vol. 1, no. 2, pp. 85-93, Apr. 1982.

[66] K. Mayaram and D. O. Pederson, "CODECS: a mixed-level device and circuit simulator," *Proc. Int. Conf. Computer-Aided Design*, Santa Clara, CA, Nov. 1988, pp. 112-115.

[67] R. J. Trihy and R. A. Rohrer, "A switched capacitor circuit simulator: AWEswit," *IEEE J. Solid-State Circuits,*, vol. 29, pp. 217-225, March 1994.

[68] F. Ye and J. Cheng, "A multimedia software for the analysis and design of circuits and electronics," *Proc. Int. Symp. multimedia software eng.*, Taipei, Dec. 2000, pp. 378-381.

[69] "*Eldo users manual*," Mentor Graphics Corporation, USA, 2005.

[70] "*IsSPICE4 users guide*," intusoft, USA, 2007.

[71] "*Gnucap the gnu circuit analysis package users manual*," Albert Davis, Sep. 2006.

[72] "*SwitcherCAD III/LTspice getting started guide*," Linear Technology, 2008.

[73] "*Getting started with TINA-TI*," Texas Instruments and DesignSoft, Inc., USA, 2008.

[74] I. M. Wilson, "Analog behavioral modeling using PSpice," *Proceedings of the 32rd Midwest Symposium on Circuits and Systems*, Aug. 1989, vol. 2, pp. 981-984.

[75] "*SPICE2: a computer program to simulate semiconductor circuits*," Electronics research laboratory report UCB/ERL M520, University of California, Berkeley, May 1975.

[76] K. G. Nichols, T. J. Kazmierski, M. Zwolinski, and A. D. Brown, "Overview of SPICE-like circuit simulation algorithms," *IEE Proc. Circuits, Devices Syst.*, Aug. 1994, vol. 141, no. 4, pp. 242-250.

[77] "*Saber user guide*," Synopsys, Inc., USA, Sept. 2011.

[78] "*Saber Simulink co-simulation interface user guide*," Synopsys, Inc., USA, Sept. 2004.

[79] J. Mo, S. he, and Y. Zou, "Fuzzy PID controller design for PWM-buck EBW stabilized high-voltage source using Saber-Matlab co-simulation," *IEEE Int. Conf. Control Automation*, 2007, pp. 2003-2007.

[80] Y. Peng, Y. Yu, L. Liang, C. Shang, J. Liu, and Y. Wang, "Simulation of high pulse power circuit based on semiconductor switch RSD in SABER-Simulink co-simulation environment and experiment," *Int. Conf. Electrical Machines Syst.*, Oct. 2008, pp. 3840-3844.

[81] "*SLPS user guide*," Cadence Design System, USA, 2007.

[82] Q. Guo, X. Jia, and H. Tang, "Co-simulation of pipeline ADC using Simulink and PSpice," *International Conference on Intelligent Computation Technology and Automation*, March 2011, vol. 2, pp. 487-490.

[83] Y. Jiang, J. A. A. Qahouq, and M. Orabi, "MATLAB/Pspice hybrid simulation modeling of solar PV cell/module," *26th Annu. IEEE Applied Power Electron. Conf. Exposition*, March 2011, pp. 1244-1250.

[84] O. A. Ahmed and J. A. M. Bleijs, "Pspice and Simulink co-simulation for high efficiency dc-dc converter using SLPS interface software," *5th IET Int. Conf. Power Electron. Machines Drives*, Apr. 2010, pp. 1-6.

[85] "*Simulink-SPICE-interface*," BAUSCH-GALL GmbH (LLC), Munich, Germany, 2010.

[86] "*Tutorial on how to use the SimCoupler Module*," Powersim Inc., Apr. 2009.

[87] Y. Zhang, Z. Zhao, Baihua, L. Yuan, and H. Zhang, "Pspice and Simulink co-simulation for high efficiency dc-dc converter using SLPS interface software," *CES/IEEE 5th Int. Power Electron. Motion Control Conf.*, Aug. 2006, vol. 1, pp. 1-5.

[88] S. Khader, A. Hadad, and A. A. Abu-aisheh, "The application of PSIM & MATLAB/SIMULINK in power electronics courses," *IEEE Global Eng. Education Conf.*, Apr. 2011, pp. 118-121.

[89] M. Ciobotaru, T. Kerekes, R. Teodorescu, and A. Bouscayrol, "PV inverter simulation using MATLAB/Simulink graphical environment and PLECS blockset," *32nd Annual Conf. IEEE Industrial Electron.*, Nov. 2006, pp. 5313-5318.

[90] HSIMplus HDL Co-Simulation, 2012. [Online]. Available: http://www.synopsys.com/Tools/Verification/AMSVerification/DesignAnalysis /Pages/HDLCo-Simulation.aspx

[91] Analogy Introduces VHDL Co-Simulation Product, 1998. [Online]. Available: http://www.eetimes.com/electronics-news/4149530/Analogy-Introduces-VHDL-Co-Simulation-Product

[92] Introduction to Digital and Analog Co-simulation Between NI LabVIEW and NI Multisim, 2012. [Online]. Available: http://www.ni.com/white-paper/13663/en

[93] T. Kirchner, N. Bannow, and C. Grimm, "Analogue mixed signal simulation using Spice and SystemC," *Design, Automation & Test Europe Conf. & Exhibition*, Apr. 2009, pp. 284-287.

[94] T. Kirchner, N. Bannow, C. Kerstan, and C. Grimm, "Mixed signal simulation with SystemC and Saber," *Forum Specification & Design Languages*, Sep. 2010, pp. 1-6.

[95] S. Mekhilef, N. A. Rahim, and Z. A. Karim, "Analysis of different type of PWM for three phase power converter," *Proc. TENCON*, 2000, vol. 1, pp. 414-418.

[96] H. Garrab, B. Allard, H. Morel, K. Ammous, S. Ghedira, A. Amimi, K. Besbes, and J. Guichon, "On the extraction of PiN diode design parameters for validation of integrated power converter design", *IEEE Trans. Power Electron.,* vol. 20, no. 3, pp. 660-670, May 2005.

[97] F. Mohammed, M. F. Bain, F. H. Ruddell, D. Linton, H. S. Gamble, and V. F. Fusco, "A novel silicon schottky diode for NLTL applications", *IEEE Trans. Electron Devices,* vol. 52, no. 7, pp. 1384-1391, Jul. 2005.

[98] *Ultra Fast Design Methodology Guide for the Vivado Design Suite.* Xilinx, Inc., April 2014.

[99] G. Massobrio and P. Antognetti, *Semiconductor Device Modeling with SPICE.* McGraw-Hill, Inc., USA, 1993.

[100] L. T. Pillage, R. A. Rohrer, and C. Visweswariah, *Fundamentals of Computer-Aided Circuit Simulation.* McGraw-Hill, Inc., USA, 1995, chapter 4.