REAL-TIME FPGA-BASED CO-SIMULATION OF LARGE SCALE POWER SYSTEMS

by

CONGHUAN YANG

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

Department of Electronic, Electrical
and Systems Engineering
School of Engineering
University of Birmingham
September 2017

To my parents

ABSTRACT

Real-time Electromagnetic Transient (EMT) simulation plays an important role in the planning, operation and analysis of electrical power systems. However, with the rapid increase of size and complexity of modern electrical power systems, 1) the simulation accuracy and 2) the capability of simulating large power systems have become two conflicting objectives. Sacrificing accuracy can lead to inappropriate component designs (for example incorrect insulation level due to inaccurate simulation results), incorrect control & relay settings and less optimized planning (for example less optimized planning & design of harmonic filters due to inaccurate simulation of harmonics) [1, 2]. Sacrificing efficiency can discourage technology innovation, slow down concept validation and the speed of product prototyping. It is therefore highly desirable and advantageous that the accuracy and the efficiency of simulating large power systems can be achieved simultaneously. This thesis proposes a new co-simulation platform by integrating Field Programmable Gate Array (FPGA) with Real-Time Digital Simulator (RTDS) to achieve these two conflicting objectives.

As the basis of the co-simulation platform, a library of power system components using EMT models is developed in FPGA, including most of the common power system elements (synchronous machines, transmission lines, passive elements, voltage/current sources and circuit breakers) and control systems (excitation systems and governor systems). The

modelling and hardware implementation of each component are described and simulation results are compared with SIMULINK to verify the accuracy of the developed models.

By integrating FPGA with RTDS, the FPGA-RTDS co-simulator is developed which combines the advantages of 1) paralleled architecture and high clock speed from FPGA and 2) better modelling flexibility and user-friendly Graphical User Interface (GUI) from RTDS together. The use of detailed EMT models for all electrical components and control systems achieves a high level of accuracy of simulation, and the interface error is eliminated in the proposed method comparing with the conventional method of interfacing Transient Stability (TS) program with EMT program as only the EMT models are used on both RTDS and FPGA sides. Deeply pipelined and massively paralleled computation algorithms in FPGA are developed. Real-time EMT simulation of power systems with more than 400 nodes is presented to demonstrate the simulation capability. Comparisons are made with SIMULINK and RTDS to verify the accuracy of the proposed co-simulator. Modular design of the hardware components and the pipelined computations significantly facilitate the system expansion with minimized hardware requirement.

To further improve the simulation capability of the proposed co-simulator, multi-FPGA structure is introduced. In this structure, the large power system to be simulated is partitioned into subsystems and each FPGA simulates one subsystem. The partition of the system is carried out in such a way that each FPGA and RTDS are decoupled and run in parallel. Thus great expandability is achieved as the system size can be easily expanded by

adding additional FPGAs without affecting the existing ones. A case study utilizing 10 FPGAs to simulate a large scale power system with more than 4000 nodes is presented to demonstrate the capability of the proposed co-simulator.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my PhD supervisor Professor Xiao-Ping Zhang. It was his innovative ideas that motivated me to start my PhD topic. His encouragement and support helped me through the four years of study. The research would not be the same level as it is today without his help.

I also want to thank all my colleagues from the research group for the interesting discussions and memorable dinners. They made my research years more enjoyable.

I would like to extend my deepest gratitude to my boyfriend and colleague, Dr. Ying Xue. Without his encouragement, understanding and support, this thesis would not have been completed. I want to thank all my friends for their company and friendship over all these years.

Finally I would like to express my gratitude to my parents for their love, support and understanding since the first day of my life. Without you, I would not be the person I am today.

TABLE OF CONTENTS

Chapter 1 INTRODUCTION 1

1.1 Research Background 1

1.1.1 Evolution of Modern Electrical Power Systems 1

1.1.2 The Need for Accurate Simulation Technology 4

1.1.3 The Need for the Efficient Simulation of Larger Systems 6

1.1.4 Opportunities and Challenges 7

1.2 Literature Review 7

1.2.1 Improvement of Simulation Capability for Large Power Systems 8

1.2.2 Improvement of Simulation Accuracy 10

1.2.3 FPGA Technology 14

1.2.4 Application of FPGA in Real-Time Simulation 18

1.3 Project Objective, Contributions and Thesis Outline 20

1.3.1 Project Objectives 20

1.3.2 Contributions of the Thesis 21

1.3.3 Thesis Outline 21

1.3.4 Logical Relationships between Chapters 23

Chapter 2 MODELLING AND HARDWARE IMPLEMENTATION OF POWER SYSTEM ELEMENTS 24

2.1 Introduction 24

2.2 Power System Elements 24

2.2.1 Synchronous Machines 24

2.2.2 Transmission Lines 45

2.2.3 Passive Elements 56
2.2.4 Voltage and Current Sources 63
2.2.5 Circuit Breakers 66
2.2.6 Loads 67
2.3 Power System Controllers 67
2.3.1 Excitation System 73
2.3.2 Governor/Turbine System 80
2.4 Model Verification 84
2.4.1 Case 1: Synchronous Machine with Constant Excitation and Governor Inputs 84
2.4.2 Case 2: Synchronous Machine with Excitation and Governor Systems 87
2.4.3 Case 3: Model Verification of Transmission Lines, Passive Elements, Voltage Sources and Circuit Breakers 91
2.5 Summary 96
Chapter 3 FPGA BASED REAL-TIME EMT SIMULATOR 97
3.1 Introduction 97
3.2 Computational Architecture 97
3.2.1 Pre-Simulation Part 98
3.2.2 Real-Time Simulation Part 100
3.3 Simulator Global Controller 104
3.3.1 Module Coordination 104
3.3.2 Hardware Implementation 105
3.4 Power System Solver 106
3.4.1 Network Solution 106
3.4.2 Non-Linear Elements: Compensation Method 108
3.4.3 Hardware Implementation 109

3.5 Case Study 114
3.5.1 Case 1 114
3.5.2 Case 2 118
3.6 Summary 121
Chapter 4 REAL-TIME FPGA-RTDS CO-SIMULATOR 122
4.1 Introduction 122
4.2 FPGA vs RTDS 122
4.3 Architecture of the FPGA-RTDS Co-Simulator 123
4.3.1 Study System & External System 124
4.3.2 Interface Design 126
4.3.3 Parallel Computation Schemes 129
4.3.4 Platform Expandability 133
4.4 Case Study 135
4.4.1 Case 1 135
4.4.2 Case 2 140
4.4.3 Case 3 144
4.4.4 Discussions 148
4.5 Summary 150
Chapter 5 SIMULATION OF LARGE SCALE POWER SYSTEMS USING MULTI-FPGA BASED CO-SIMULATOR 151
5.1 Introduction 151
5.2 Expansion to Multi-FPGA 151
5.3 Methodology of Network Partition 152
5.4 Interface Design 154
5.5 Simulator Capability 157

5.6 Case study 159
5.6.1 Network Configuration 159
5.6.2 Hardware Implementation 161
5.6.3 Simulation Results 162
5.6.4 Discussions 170
5.7 Summary 171
Chapter 6 CONCLUSIONS AND FUTURE WORKS 172
6.1 Conclusions 172
6.2 Future Work 176
LIST OF PUBLICATIONS 177
REFERENCES 179
Appendix A. Data of Case Study in Chapter 2 190
Appendix B. Data of Two-Area Four-Machine Power System [147] 193
Appendix C. Data of IEEE 68-Bus, 16-Machine System [154] 194
Appendix D. Xilinx ISE Design Suite Environment 199
Appendix E. Sample VHDL Codes 207

LIST OF FIGURES

FIGURE 1-1 INSTALLED CAPACITY OF RENEWABLE SOURCES IN UK [27] 2

FIGURE 1-2 GENERAL FPGA ARCHITECTURE 17

FIGURE 2-1 BLOCK DIAGRAM REPRESENTATION OF SYNCHRONOUS MACHINE 25

FIGURE 2-2 SYNCHRONOUS MACHINE SCHEMATIC 26

FIGURE 2-3 SOLUTION STEPS FOR SYNCHRONOUS MACHINE 34

FIGURE 2-4 ROTOR SPEED AND POSITION PREDICTION MODULE 35

FIGURE 2-5 PARK TRANSFORMATION MODULE 36

FIGURE 2-6 READ-ONLY MEMORY OF PARK TRANSFORMATION MODULE 37

FIGURE 2-7 MATRIX MULTIPLICATION MODULE 38

FIGURE 2-8 XY MATRIX FORMAT 41

FIGURE 2-9 STATOR CURRENT SOLVING MODULE 43

FIGURE 2-10 ELECTROMAGNETIC TORQUE & ROTOR SPEED MODULE 44

FIGURE 2-11 EQUIVALENT CIRCUIT FOR LOSSLESS TRANSMISSION LINE 47

FIGURE 2-12 REPRESENTATION OF TRANSMISSION LINE WITH LOSSES 47

FIGURE 2-13 A SEGMENT OF THREE-PHASE MUTUALLY COUPLED LINE [42] 49

FIGURE 2-14 SOLUTION STEPS FOR MULTIPHASE COUPLED TRANSMISSION LINES 53

FIGURE 2-15 TRANSMISSION LINE MODULE 55

FIGURE 2-16 RESISTOR BRANCH 56

FIGURE 2-17 CAPACITOR BRANCH 57

FIGURE 2-18 INDUCTOR BRANCH 58

FIGURE 2-19 SERIES *RL* BRANCH 59

FIGURE 2-20 PASSIVE ELEMENTS MODULE 62

FIGURE 2-21 SOURCE MODULE 65

FIGURE 2-22 CIRCUIT BREAKER MODEL 66

FIGURE 2-23 GENERIC TRANSFER FUNCTION BLOCK 68
FIGURE 2-24 NON-WINDUP LIMITER 70
FIGURE 2-25 WINDUP LIMITER 71
FIGURE 2-26 INTERFACE BETWEEN CONTROL SYSTEM AND ELECTRICAL NETWORK 72
FIGURE 2-27 BLOCK DIAGRAM REPRESENTATION OF AC1A EXCITER BLOCK 74
FIGURE 2-28 SOLUTION STEPS FOR EXCITATION SYSTEM 77
FIGURE 2-29 EXCITATION MODULE 77
FIGURE 2-30 GOVERNOR/TURBINE SYSTEMS 80
FIGURE 2-31 SOLUTION STEPS FOR GOVERNOR/TURBINE SYSTEM 82
FIGURE 2-32 GOVERNOR MODULE 83
FIGURE 2-33 VERIFICATION OF SYNCHRONOUS MACHINE MODEL 84
FIGURE 2-34 STARTING PROCESS FOR SYNCHRONOUS MACHINE WITH CONSTANT EXCITATION VOLTAGE AND MECHANICAL TORQUE. 86
FIGURE 2-35 VERIFICATION OF CONTROL SYSTEMS FOR SYNCHRONOUS MACHINE 87
FIGURE 2-36 STARTING PROCESS FOR SYNCHRONOUS MACHINE WITH EXCITATION SYSTEM AND GOVERNOR SYSTEM 90
FIGURE 2-37 THREE PHASE DIAGRAM OF TEST SYSTEM 91
FIGURE 2-38 COMPARISONS OF SIMULATION RESULTS OF THE TEST SYSTEM 95
FIGURE 3-1 COMPUTATIONAL ARCHITECTURE 98
FIGURE 3-2 DESIGN FLOW 99
FIGURE 3-3 COMPUTATION SCHEME OF REAL-TIME SIMULATION PART 101
FIGURE 3-4 COMMUNICATION SIGNALS BETWEEN MODULES 104
FIGURE 3-5 GLOBAL CONTROL MODULE FSM 105
FIGURE 3-6 HARDWARE IMPLEMENTATION OF HISTORY CURRENT FORMATION SUBMODULE 110
FIGURE 3-7 HARDWARE IMPLEMENTATION OF CALCULATING NODAL CURRENT INJECTIONS 111
FIGURE 3-8 DETAILED HARDWARE ARCHITECTURE OF THE FPGA-BASED EMT SIMULATOR 113
FIGURE 3-9 SINGLE-LINE DIAGRAM OF CASE STUDY1 114

FIGURE 3-10 THREE-PHASE VOLTAGE AT BUS 6 115
FIGURE 3-11. THREE-PHASE LINE CURRENT BETWEEN BUS 7 AND BUS 8 (TL78A) 116
FIGURE 3-12 CURRENT AT FAULT LOCATION 117
FIGURE 3-13 SINGLE-LINE DIAGRAM OF CASE STUDY 2 118
FIGURE 3-14 SIMULATION RESULTS FOR CASE 2 120
FIGURE 4-1 ARCHITECTURE OF THE PROPOSED FPGA-RTDS CO-SIMULATOR 123
FIGURE 4-2 INTERFACE BETWEEN FPGA AND RTDS 126
FIGURE 4-3 HARDWARE MODULES FOR FPGA IMPLEMENTATION 130
FIGURE 4-4 COMPUTATION SCHEME OF THE CO-SIMULATOR 131
FIGURE 4-5 NETWORK EXPANSION OF TRANSMISSION LINE COMPONENTS 133
FIGURE 4-6 NETWORK EXPANSION OF ADMITTANCE MATRIX 133
FIGURE 4-7 SIMULATIONS OF TWO-AREA FOUR-MACHINE SYSTEM 136
FIGURE 4-8 SIMULATION RESULTS FOR CASE 1 139
FIGURE 4-9 SIMULATIONS OF TWO-AREA FOUR-MACHINE SYSTEM FOR CASE 2 140
FIGURE 4-10 SIMULATION RESULTS FOR CASE 2 144
FIGURE 4-11 SIMULATION OF A 141-BUS SYSTEM USING THE PROPOSED CO-SIMULATOR 145
FIGURE 4-12 CASE 3 SIMULATION RESULTS FROM RTDS 146
FIGURE 4-13 CASE 3 SIMULATION RESULTS FROM FPGA 147
FIGURE 4-14 BREAKDOWN OF MINIMUM SIMULATION TIME-STEP OF CASE 1 149
FIGURE 5-1 DIRECT COMMUNICATION BETWEEN FPGAS 155
FIGURE 5-2 INDIRECT COMMUNICATION THROUGH RTDS 156
FIGURE 5-3 SIMULATION CAPABILITY OF THE MULTI-FPGA BASED CO-SIMULATOR 157
FIGURE 5-4 SINGLE-LINE DIAGRAM OF SYSTEM CONFIGURATION 160
FIGURE 5-5 SIMULATION RESULTS WITH 100MS FAULT AT RECTIFIER SIDE 165
FIGURE 5-6 SIMULATION RESULTS WITH 100MS FAULT AT INVERTER SIDE 168
FIGURE 5-7 BREAKDOWN OF MINIMUM SIMULATION TIME-STEP 170
FIGURE D. 1 ISE CODING ENVIRONMENT 199

FIGURE D. 2 DESIGN SUMMARY WINDOW……200

FIGURE D. 3 RTL SCHEMATIC VIEW……201

FIGURE D. 4 ISIM SIMULATION ENVIRONMENT……203

LIST OF TABLES

TABLE 1.1 EXISTING AND FUTURE INTERCONNECTOR PROJECTS [28] 3

TABLE 1.2 FEATURE SUMMARY OF XILINX VIRTEX-6 XC6VLX240T FPGA [115] 16

TABLE 2.1 DEFINITIONS OF VARIABLES FOR SYNCHRONOUS MACHINE 28

TABLE 2.2 PARAMETERS FOR GOVERNOR/TURBINE SYSTEM 80

TABLE 4.1 FPGA RESOURCE UTILIZATION 148

TABLE A.1 SYNCHRONOUS MACHINE PARAMETERS 190

TABLE C.1 TRANSMISSION LINE PARAMETERS 194

TABLE C.2 LOAD PARAMETERS 196

TABLE C.3 GENERATOR PARAMETERS 198

LIST OF ABBREVIATIONS

AC	Alternating Current
ASIC	Application Specific Integrated Circuit
CLB	Configurable Logic Blocks
CPLD	Complex Programmable Logic Devices
CPU	Central Processing Unit
D/A	Digital To Analog
DC	Direct Current
DSP	Digital Signal Processing
EMT	Electromagnetic Transient
EMTP	Electromagnetic Transient Programme
FACTS	Flexible Ac Transmission Systems
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GTDI	Gigabit Transceiver Digital Input Card
GTDO	Gigabit Transceiver Digital Output Card
HIL	Hardware-in-the-Loop
HVDC	High Voltage Direct Current
I/O	Input/Output
ISP	In-System Programming
LCC	Line-Commutated Converter
LUT	Look-Up Tables

MMC	Modular Multilevel Converter
MMCM	Mixed-Mode Clock Managers
MSB	Most Significant Bit
OTP	One Time Programmable
PLD	Programmable Logic Devices
PLL	Phase Locked Loop
PV	Photovoltaics
PWM	Pulse Width Modulation
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory
RTDS	Real-Time Digital Simulator
RTL	Register Transfer Level
RTS	Real-Time Simulator
SRAM	Static Random Access Memory
STATCOM	Static Compensators
TCSCs	Thyristor-Controlled Series Compensations
TNA	Transient Network Analyser
TS	Transient Stability
UBC	University of British Columbia
USB	Universal Serial Bus
VCO	Voltage-Controlled Oscillator

Chapter 1

INTRODUCTION

1.1 Research Background

1.1.1 Evolution of Modern Electrical Power Systems

Modern electrical power systems are undergoing dramatic changes in a number of ways, including:

- Increasing level of integration from renewable energy sources and Electric Vehicles (EV) [3-7].
- Increasing level of interconnections between regional/national electrical power systems [8-16].
- Increasing applications of power electronic-based devices in existing networks [17-21].

Renewable sources such as wind, solar, hydro and EV are playing an increasingly important role in electrical power systems to provide low-carbon electricity. The speed of increase of renewable integration is further driven by government targets and national legislations. For example in the U.K., the target is to provide 15% of total energy demand from renewables by 2020, and 30% of electricity from renewables [22]. Scottish government's 2020 renewable target states that the equivalent of 100% of Scotland's electricity demand should be met by renewables [22]. Currently UK already has the largest offshore wind farms in the world with

further projects down the pipeline [23]. PV deployment in UK has reached more than 10GW, and is becoming the most popular renewable energy source among British electricity consumers, both in the domestic and commercial sectors [24]. More than 100,000 plug-in EVs have been registered in the UK up until March 2017 [25]. Figure 1-1 shows the installed capacity of different renewable sources in UK. Similar trend can also be found in other parts of the world, for example in China about 13 large hydro power centres are planned with a total generating capacity of about 215 GW [26].

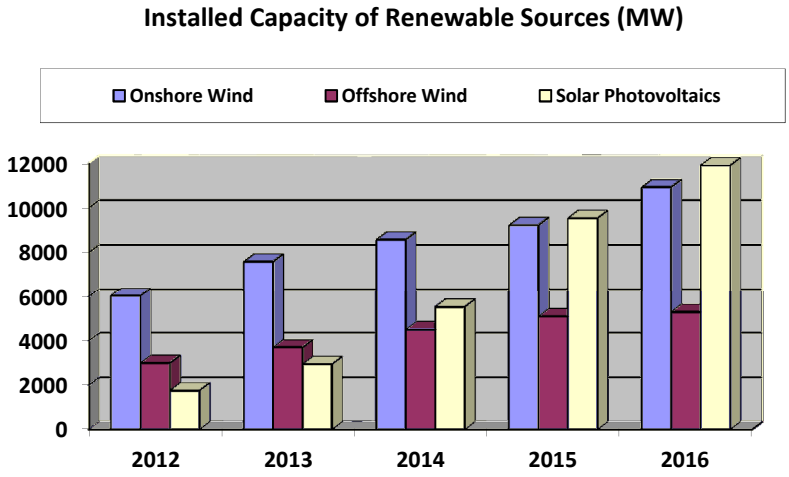


Figure 1-1 Installed capacity of renewable sources in UK [27]

The number of interconnections between power systems also grows significantly in recent years. Interconnectors are the physical links which allow the transfer of electricity across borders. It can improve the diversity and security of energy supplies, facilitate competition in wider electricity market and help the transition to a low carbon future by integrating renewable sources [28]. For example, UK currently has 4GW of interconnector capacity with France, Netherlands, Northern Ireland and the Republic of Ireland [28], with 7 more interconnectors to be built in the next five years linking the UK with other European countries [28]. A list of these projects is shown in Table 1.1. Furthermore, under the initiative

of global power and energy internet, interconnectors have been built or proposed world-wide in middle-east [29], Asia [11], Europe [13], North America [30] and South America [31, 32].

Table 1.1 Existing and Future Interconnector Projects [28]

Project Name	Connecting Country	Capacity	Delivery date/Estimated Delivery Date
IFA	France	2000MW	1986
Moyle	Ireland	500MW	2002
BritNed	Netherlands	1000MW	2011
EWIC	Ireland	500MW	2012
ElecLink	France	1000MW	2019
NEMO	Belgium	1000MW	2019
NSN	Norway	1400MW	2020
FAB Link	France	1400MW	2022
IFA2	France	1000MW	2020
Viking	Denmark	1000MW	2022
Greenlink	Ireland	500MW	2021

Due to the intermittent nature of renewables compared with traditional power plant, they bring additional risks to the operation, control and protection of power systems. To solve these problems, power electronic-based devices are installed to improve the stability and controllability of power systems. For example, in the UK two units of Thyristor-Controlled Series Compensations (TCSCs) have been installed to boost the boundary transfer capability between Scotland and England [33], and three units of Static Compensators (STATCOMs) are to be installed in the National Grid Electricity Transmission Network to improve the transient voltage stability [34].

All the changes mentioned above directly result in a significant increase in the size and complexity of power systems, which makes the power system planning, operation and optimization ever-more difficult. For example, for simulation studies with interconnectors,

detailed modelling of power systems at both ends of the interconnector is required for accurate representation of system dynamics and control/harmonic interactions. At the same time, the simulations of these interconnectors themselves are normally computationally intensive because of the large number and high-frequency switching of the power electronics involved.

1.1.2 The Need for Accurate Simulation Technology

The increasing complexity of power system drives the need for more accurate simulation technologies. For example, the switching dynamics from power electronic based devices can potentially lead to more serious transient overvoltage/overcurrent problems and harmonic problems. Also it poses great challenges to the protection and control of power systems as the voltage and current behaviour of the system are affected by the operations of power electronic devices. To accurately simulate these phenomena to facilitate the design and planning of protection/control, the power system component models should be able to represent high frequency dynamics of up to MHz. This means that the required simulation time-step is in the order of microsecond according to Nyquist criteria [35]. Normally, a simulation time-step of 1 μs – 50 μs is required [1] for the study of the above phenomenon.

Traditionally, electromechanical transient simulation programmes (also known as transient stability (TS) programmes) have been utilized to analyse the transient stability of power systems [36]. Typical TS programmes solve thousands of differential-algebraic equations for multi-machine power systems assuming single-phase fundamental frequency behaviour [37]. There are several commercial TS programmes available in the market such as the PSS/E from PTI [38], SIMPOW from ABB [39], NETOMAC from Siemens [40] and PowerFactory [41]. A simulation time-step of around 10ms is normally adopted by TS programmes, and with this

simulation time-step, high frequency dynamics as mentioned above cannot be accurately simulated. As a result, the use of TS programme is limited when higher simulation accuracy/smaller simulation time-step are required.

Electromagnetic transient programme (EMTP) was specifically developed with sufficient accuracy for the analysis of above aspects [42]. Unlike electromechanical transient, Electromagnetic transient (EMT) considers the interaction between magnetic fields of inductance and the electric fields of capacitances in the system [43]. The study of these interactions requires accurate simulation of the network elements such as synchronous generators, transmission lines, transformers, and power converters. The models should cover a wide range of frequencies from DC up to MHz, which indicates a required simulation time-step as small as a few microseconds [1, 43]. For example, to investigate 10 KHz voltage disturbance, a time-step of smaller than 50 μs is required according to Nyquist criterion [35].

Typically the following modelling aspects are considered in the EMTP:

- Non-linearity in electrical components;
- Electro-magnetic couplings;
- Travelling wave effect of transmission lines;
- Unbalance of three-phase systems and frequency dependent of transmission line parameters.

Normally the power system components are represented as differential equations using instantaneous values in EMTP rather than phasor values. Also single-phase representation is replaced by multi-phase representation [43]. Differential equations are solved using numerical method with simulation time-steps of around 50μs. With such small simulation time-step, the size of power system that can be simulated by EMTP is rather limited. The

original commercial EMTP is developed by Dommel [42], and a number of developments have been made ever since: PSCAD-EMTDC [44] developed by Manitoba HVDC Research Centre; NETOMAC from Siemens and a number of other EMT-based simulators developed in the past two decades which will be discussed in detail in the next section.

1.1.3 The Need for the Efficient Simulation of Larger Systems

The continuous increase of the number of interconnector and size of power system inevitably leads to the need for the efficient simulation of larger power systems. Although sufficient simulation accuracy can be provided by EMTP, the size of power system that can be efficiently simulated by EMTP is usually limited (e.g., IEEE 39-bus system can take up a whole rack of RTDS with detailed EMT modelling of synchronous machines and transmission lines). For example with most of the commonly used EMTP-based simulators, e.g. PSCAD, it is difficult to study the phenomenon of harmonic/control interactions of large scale hybrid AC/DC transmission network [37]. This is because smaller simulation time-step (in the order of few microseconds) is needed for the simulation of converter systems for DC transmission. As a result it takes up a significant amount of the computation resources and limits the size of AC system (to tens of buses) that can be simulated given a fixed amount of hardware resources. This type of study requires the detailed representation of synchronous machines and power electronic-based converters, and the size of network can be easily up to hundreds, if not thousands of nodes. Due to the high requirement of both accuracy and the size of the system, the use of EMTP becomes technically difficult and economically prohibitive in simulation of large power systems.

For example for the simulation of Modular Multilevel Converter (MMC) based HVDC system, a simulation time-step of less than 5μs is required for accurate results [45]. A normal

MMC converter has several thousand of power electronic switches, and the simulation of such a system using EMTP becomes very time and resource-consuming. Even if the MMC HVDC can be simulated, the connecting AC systems that can be modelled are normally limited to a maximum of tens of nodes using typical EMT programmes. Network equivalents can be adopted to reduce the computational burden, at the expense of simulation accuracy. From the point of view of system operators, it would certainly be beneficial for planning and analysis purposes if a larger part of the power network can be simulated under EMT type of simulation programmes.

1.1.4 Opportunities and Challenges

From the above analysis, it can be clearly seen that the simulation of modern electrical power system has the following two seemingly conflicting objectives:

1. To achieve accurate simulation of power electronic devices and nonlinear power system dynamics.
2. To achieve efficient simulation of larger-scale power systems;

Significant amount of efforts have been made to achieve the above two objectives. The main approach is to simplify the part of power system that is remote to the system of interest [46-51]. The way of simplification varies from each other and the resulting simulation accuracy and speed are also different. Nevertheless, more powerful simulation methodologies are desired to achieve both accuracy and simulation speed at the same time.

1.2 Literature Review

Previous research works can be classified into two main categories: one is focusing on improving the simulation capability for large power systems and the other is focusing on

improving the simulation accuracy. Literatures of these two categories are discussed first and then the FPGA technology and its application to power system simulation are reviewed.

1.2.1 Improvement of Simulation Capability for Large Power Systems

1.2.1.1 Network Equivalent & Network Partitioning

One of the earliest attempt to achieve efficient simulation is made in [48]. In the paper, EMT models of power system components were adopted for part of the network. Passive equivalent circuits were used to simplify the rest of the network. The passive equivalent circuits were used to represent the dynamics of the original power system. The simplification to RLC components inevitably causes the loss of important power system dynamics from synchronous machines and transmission lines. Another method was to partition the original network into smaller subsystems. In reference [52], the method of network partitioning was proposed utilizing the natural delay of travelling-wave transmission lines. The simulations of subsystems which are interconnected by relatively long transmissions lines are effectively decoupled with each other. Another method of network partitioning was proposed in [53]. The proposed method divided the original system using the idea of "boundary busses". One of the benefits of this method was that it did not relay on the transmission lines. A different approach which based on the idea of multi-rate simulation was proposed in [54]. The complete system was modelled using EMT component models and solved with different time-steps according to the speed of system dynamics. The part of the system with slow dynamics was simulated using larger time-steps, while the part of the system with faster dynamics was simulated using smaller time-steps. However, the drawback for the above methods is that the size of the system that can be simulated is limited [37].

1.2.1.2 Hybrid EMT/Transient Stability (TS) Simulation

In the last decade, the idea of hybrid simulation method was proposed [47, 51, 55-64] to achieve efficient simulation of relatively large power systems without significantly sacrificing the simulation accuracy. The main objective of hybrid simulation is to split the original network into two parts and the part that requires less simulation accuracy is simulated using TS programmes, while the other part is simulated using detailed EMT programmes. The EMT part usually includes HVDC links, Flexible AC Transmission (FACTS) devices or parts of the network that is sensitive to disturbances. The TS part tries to include extensive portions of the network, and detailed component models are not implemented.

The accuracy of hybrid simulation is largely determined by the method of network equivalent for both simulators. Reference [65] suggested to represent the TS part of the system by a Norton equivalent with fundamental frequency current source for the simulation of EMT part of the network. The drawback of this method is that the topological changes during simulation (e.g., faults) at TS part of the system can only be simulated by changing the impedance value of the Norton equivalent circuit [62]. More accurate approaches are to use the frequency dependent network equivalent [58, 61, 63]. The method of Vector Fitting was proposed in [66] to calculate the frequency dependent network equivalent. In reference [47], a method of multiport Norton equivalent with couplings was proposed for the simulation case with multiple interface buses. However these methods considerably increase the computational burden of EMT simulators.

Another aspect in hybrid simulation is the choice of interface variables and the conversion of data. The interface variables that have been chosen must be able to determine the power flow

into the interface buses [61]. Data converter blocks are required to connect two types of programmes. Conversion from phasors to instantaneous variables is required for EMT programme, and conversion from instantaneous variables to phasors is required for TS programme. Signal generators with the inputs of amplitude, phase and frequency information are normally used to provide interface data to EMT simulators [67]. Digital processing techniques such as curve fitting, Fast Fourier Transform (FFT) are utilized to provide interface data to TS simulators [47, 61, 67]. However the problems related to the speed of FFT method and the effectiveness in handling DC offset with curve fitting technique need to be solved [51].

From the above analysis, it can be seen that the actual implementation of hybrid simulation is difficult. It is understandable because two parts of the systems to be simulated are based on different mathematical models under different assumptions, and are simulated with different time-steps. Due to the nonlinear nature of power system, the interface between them will inevitably be difficult to realize. At the same time the potential interface errors cannot be eliminated.

Therefore the idea of utilizing modern Field Programmable Logic Arrays (FPGAs) to simulate large scale power systems with detailed EMT models becomes attractive. A review of the FPGA technology and associated applications are shown in the next sections.

1.2.2 Improvement of Simulation Accuracy

1.2.2.1 Real-Time Simulation

The idea of real-time simulator (RTS) was first developed in the 1950s to achieve high simulation accuracy for power system simulation. The main advantages of real-time simulation are its fast simulation speed (real time) and its ability to carry out HIL testing for

protection equipment and control systems. Therefore the testing costs can be reduced since the HIL test setup is less expensive than the physical setup. In addition, the testing using real-time simulation can replace risky or expensive test that uses physical high voltage/high current equipment [1]. The earliest RTS is called Transient Network Analyser (TNA) [68-70]. The working principle of TNA is to establish the physical model of power system but with lower levels of voltage and current. Therefore the physical characteristics in the TNA are effectively the same as those in real power systems, and the simulation using TNA is inherently real-time. The size of the network that can be simulated is largely dependent on the available equipment and the size of the lab, so the simulation of a relatively large power system is economically prohibitive. Furthermore, the setting up of new simulation cases or the modification to existing simulation cases are labour-intensive and require highly skilled personnel [71]. Due to these reasons, the flexibility and expandability of TNA is rather limited.

With the fast development of microprocessor and integrated circuit, digital RTS started to emerge based on Central Processing Unit (CPU), Digital Signal Processing (DSP) or Reduced Instruction Set Computer (RISC) [71]. Fully digital RTS have higher flexibility and lower cost compared with traditional TNA, and has been used till now. Significant advances were made since 1980s. Reference [72] proposed a real-time digital simulator for the testing of protection relays. The proposed simulator was based on EMTP using RISC and DSP as fundamental hardware layer. The IBM RISC 6000 work station performed the simulation of the network, while the DSP performed the simulation of transformers. Reference [73] proposed the use of dual DSP for real-time simulation. In the proposed method, each DSP was used to simulate one terminal of a transmission line. A real-time simulation framework fully based on IBM RISC 6000 work station was proposed in [74]. It achieved a simulation

time-step of 38μs to 107μs for network with 18 to 30 nodes. University of British Columbia (UBC) developed their real-time digital simulator (OVNI) based on Pentium II workstation [75]. It can achieve the real-time simulation of HVDC system with 24 switches at a simulation time-step of 81μs. However, all the above mentioned methods either used custom-made DSPs or commercial supercomputers. The high cost of the equipment and the cost associated with the inter-computer communication limited the further applications of such methodologies.

With further development of the processor technology, the PC-cluster based real-time simulation technique was adopted to cater for larger sized power networks with reduced cost. Reference [76], on the basis of reference [75], developed a PC cluster version of OVNI. The proposed method utilized 5 PCs to form the PC cluster. The speed of simulation and simulation time-step were considerably improved compared with the original OVNI. A demonstration of simulating 234 node transmission network in real-time is shown in the paper. University of Alberta developed another PC cluster based real-time digital simulator, targeting the electrical drive systems with power electronic devices [77]. The simulation time-step of 10μs was achieved. The PC cluster based method was further driven by the advent of low-cost, high-performance readily available multi-core processors [78]. It potentially reduced the need to cluster multiple PCs to conduct complex parallel simulation, thereby reducing dependence on the inter-computer communication technology.

The latest trend in real-time simulation consists of exporting simulation models to FPGA [79-92]. However, most of the works to date are only focusing on the simulation of power electronic converters or small sized power systems. Very few applications have been found to implement the EMTP for large-scale power systems using FPGA [93-95]. One of the important advantages of using FPGA is that computational burden (simulation time-step) is

not linearly related to the size of the system to be simulated, which is due to its paralleled architecture and pipelined way of computations. As a result, potentially very large systems using EMT models can be simulated in real-time.

In parallel with the continuous research and development in real-time simulation technologies, a number of real-time digital simulators have been made commercially available:

1. The Real-Time Digital Simulator (RTDS), from RTDS Technologies Inc. The company demonstrated the first commercial RTDS in 1991 using DSPs [96]. Modern RTDS uses PowerPC RISC processors that are implemented in cards (PB5) [2]. Several cards are used form a rack for the simulation a complete power system. With larger networks, multiple racks can be utilized together with fast interface cards.
2. eMEGAsim, from OPAL-RT Technologies Inc. [97]. The hardware includes multi-core CPU, FPGA, and commercial of-the-shelf motherboard. The models are constructed in MATLAB/SIMULINK, and are simulated in target PCs.
3. HYPERSIM from Hydro-Quebec [98, 99]. It is a supercomputer-based simulator, which is no longer commercially available. Recent development of the simulator uses PC clusters, supercomputers and 32 core Intel computers to suit multi-purpose simulations.
4. dSPACE [100]. It is purely CPU based with the modelled constructed in SIMULINK. It is mainly used for real-time control and rapid prototyping for automotive engineering and industrial control.
5. VTB [101-103]. It is based on DSP cluster or multi-core CPU/FPGA, and is mainly used for power system simulations.
6. xPC Target [104]. It is based on CPU and FPGA and is mainly used for rapid prototyping, real-time testing of applications and HIL simulation.

7. rtX from ADI [105]. Simulations are carried out in CPU and main purpose is for power system simulation for avionics and maritime industries, aircraft simulation and shipboard simulation.
8. Typhoon RTDs [106, 107]. Simulations are carried out in FPGA with models constructed in Typhoon schematic editor. It is mainly used for the testing of power electronics controller.
9. NETOMAC from Siemens [40, 108]. It is a PC based digital simulator which is able to carry out transient stability, electromagnetic transient and real-time simulations.

Among all the available real-time digital simulators, only a few of them are capable of simulating large systems [2]. The main problem is that these simulators are expensive, both in terms of initial investment and maintenance fees. With the growing size and complexity of power system, significant further investments are likely to be required.

1.2.3 FPGA Technology

Typical digital systems can be classified into full-custom integrated circuit and semi-custom integrated circuit [109]. Full-custom integrated circuit can be further categorized into general-purpose integrated circuit and special-purpose integrated circuit. The commonly used CPU, RISC, DSP all belong to general-purpose integrated circuit and Application Specific Integrated Circuit (ASIC) belongs to the special-purpose integrated circuit [110].

General-purpose integrated circuit can be fully compiled and controlled by software, hence achieves the decoupling between software and hardware. This feature makes the general-purpose integrated circuit very flexible but with lower computational efficiency. On the other hand, ASIC is the purposely built integrated circuit designed to meet specific requirements from customers. In this way, the circuit configuration, layout and wiring can be optimally

designed to achieve the best utilization of the circuit. The drawbacks of ASIC are that it is less flexible with high cost, and cannot be reconfigured.

Semi-custom integrated circuit includes various kinds of reconfiguration devices such as FPGA and Complex Programmable Logic Devices (CPLD). This kind of reconfigurable device lies between the general-purpose integrated circuit and special-purpose integrated circuit. The manufacturer provides basic layout of logic gates and reconfigurable logic blocks, and the customer is able to design the connections between logic gates using professional software. In this way, the integrated circuit is effectively configured by the customers according to their need. It can be seen that this type of devices combines the merits of both general-purpose integrated circuit and special-purpose integrated circuit, featuring low cost, high configurability and short development time. Comparing FPGA with CPLD, FPGA has better configurability, higher integration density in terms of logic gates and better computational ability.

Currently the FPGA in the market can be classified into three different categories: 1) Static Random Access Memory (SRAM)-based [111, 112]; 2) fuse-based [111, 112] and 3) flash-based [113]. Among them, the SRAM-based FPGA is the most widely used because of its fast computational speed and high reconfiguration ability [114]. Fuse-based FPGA only has One Time Programmable (OTP) ability. Flash-based FPGA is relatively new and does not have wide applications. In terms of the manufactures, Altera, Xilinx, Lattice and Actel are some of the largest FPGA manufactures in the world.

- Altera® is one of the most famous FPGA manufactures with fast developments after 1990s. It has now become one of the largest providers of programmable logic devices

(PLD). FPGA from Altera® has high performance, high integration density of logic gates and relatively low cost.

- Xilinx® is the inventor of FPGA and it has been manufacturing PLDs over several decades. The FPGA was first developed in 1985 by Xilinx and several generations of FPGAs have been developed with much improved performances.
- Lattice® is the inventor of In-System Programming (ISP). It enters into the FPGA markets after 2004 and is now the world's third largest provider for FPGA.
- Actel® is an American FPGA manufacturer originally focusing on products for military defence and aerospace engineering. The early generations of FPGAs from Actel® are mostly fuse-based. Recently it starts to develop commercially available flash-based FPGA.

Table 1.2 Feature Summary of Xilinx Virtex-6 XC6VLX240T FPGA [115]

Logic Cells		241,152
Configurable Logic Blocks (CLBs)	Slices	37,680
	Max Distributed RAM (Kb)	3,650
DSP48E1 Slices		768
Block RAM Blocks	18 Kb	832
	36 Kb	416
	Max (Kb)	14,976
MMCMs		12
Interface Blocks for PCI Express		2
Ethernet MACs		4
Maximum Transceivers GTX		24
Total I/O Banks		18
Max User I/O		720

As the Xilinx® Virtex-6 XC6VLX240T FPGA is used for the EMTP implementation in this thesis, its basic configuration and functions are briefly illustrated in the following sections.

Figure 1-2 shows the general architecture of FPGA. It can be seen from the figure that the

FPGA includes a large amount of Configurable Logic Blocks (CLB) Slices, Block RAMs, DSP-DSP48E1 Slices and Input/Output blocks, which are interconnected by a number of column and row programmable interconnects to form a two dimensional architecture.

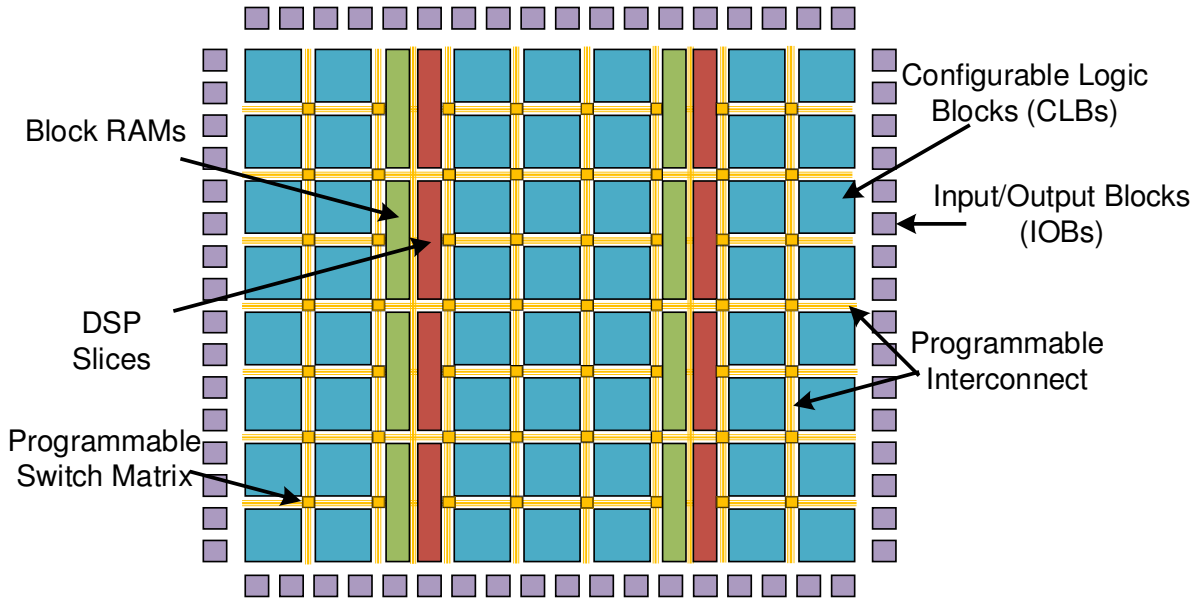


Figure 1-2 General FPGA architecture

Configuration Logic Block (CLB)

Configurable logic block (CLB) is the basic repeating logic resource on FPGA. CLBs contain smaller components, including flip-flops, look-up tables (LUTs), and multiplexers. The LUTs in Virtex-6 FPGAs can be configured as either one 6-input LUT (64-bit ROMs) with one output, or as two 5-input LUTs (32-bit ROMs) with separate outputs. Each LUT output can optionally be registered in a flip-flop. Four such LUTs and their eight flip-flops as well as multiplexers and arithmetic carry logic form a slice, and two slices form a CLB.

Block RAM (BRAM)

Block RAM (BRAM) on FPGA is a dedicated slice of random access memory that is embedded throughout an FPGA for data storage. Each block RAM has two completely

independent ports that share the stored data. Each memory access, read and write, is controlled by the clock. BRAM can be used to transfer data or store large data sets more efficiently than RAM built from LUTs.

DSP-DSP48E1 Slice

Digital signal processing applications require large number of binary multipliers and accumulators if realized using logic resources. Therefore dedicated DSP slices are provided to improve efficiency. The DSP48E1 slice provides extensive pipelining and extension capabilities that enhance speed and efficiency of many applications, such as wide dynamic bus shifters, memory address generators, wide bus multiplexers, and memory-mapped I/O register files. The accumulator can also be used as a synchronous up/down counter.

Input/Output

Input and output (I/O) resources on FPGA are physical structures that allow the user to connect FPGA to other devices. I/O blocks are usually arranged at the periphery of the FPGA architecture and also connected to the programmable interconnect. The FPGA I/O pins are configurable and can comply with different standards.

1.2.4 Application of FPGA in Real-Time Simulation

FPGA has already being widely used in communication [116-122], control [123-129], and computer engineering [130-134]. Due to its inherent parallel processing ability and pipelined design, FPGA is beginning to play a more important role in real-time simulations. For example, FPGA has been used in the capture of high-precision pulses, high density digital I/O applications and Pulse Width Modulation (PWM) signal generations [135-138]. In terms of large scale power system simulation, FPGA has the features of parallel processing, pipelined

design and high clock speed. At the same time, the calculation of power system dynamics can be carried out in a paralleled and pipelined manner, which will be discussed in detail in later chapters. Therefore, FPGA is particularly suitable for the simulation of large scale power systems. With the fast increasing logic density, capacity and speed as well as the reducing power consumption and price, the FPGA will play an even bigger role in real-time simulation of electrical power systems.

Because of these advantages, some effort has been made in utilizing FPGA to achieve real-time simulations of power systems. For example in RTDS and eMEGAsim, the FPGA has been used to simulate part of the power systems with small time-step [89, 139]. Currently the only model available in FPGA is the MMC HVDC converters. In these applications, the FPGA is used as supplementary simulation hardware to the main DSP/CPU/RISC based simulators. Some other works have focused on the use of FPGA as the main simulation hardware [79, 94, 95, 140-146]. Reference [94] implemented the EMT models of frequency dependent transmission lines and passive components into FPGA to achieve real-time simulations. The designed simulator is based on 46-bit floating-point number, and can achieve a simulation time-step of 12μs when simulating a system with 15 transmission lines, 4 synchronous generators and 8 loads. An oscilloscope was used to measure the system response. Reference [140] proposed an iterative non-linear EMT solver to implement the Newton-Raphson method in solving non-linear equations. Simulation case studies of surge arresters and non-linear reactors were carried out to demonstrate the performance. Reference [141, 142] implemented the EMT models of universal machines, universal lines, and transformers with hysteresis characteristics into the FPGA. Reference [144] proposed a generalized, parallel implementation methodology for real-time simulation of AC machine transients. A simulation time-step of a few hundred nanoseconds was achieved. Based on the

method proposed in [94], reference [95, 143] further develops the FPGA based simulator for the simulation of large power systems. A 420 bus system has been simulated with a simulation time-step of 45μs. The simulation of high-frequency power electronic devices in FPGA is discussed in [79]. With limited external AC systems, a simulation time-step of 60ns can be achieved. Reference [145, 146] introduces the switching functions of power electronic switches into FGPA to simulate power electronic based machine-driving systems. Simulation time-step of 12.5ns was achieved.

From the above analysis, it can be seen that the computational ability of FPGA is highly suitable for the application of real-time simulation of power systems. In particular very small time-steps can be achieved for fast-switching power electronic devices, and small time-steps can be achieved for very large AC power systems. The downside of FPGA is that currently there is no Graphical User Interface (GUI) for FPGA-based simulator, hence the visualisation of results and the manipulations of the simulated system (change of topology, system/controller parameters) are not convenient compared with commercial real-time simulators such as RSCAD or SIMULINK. Therefore it would be very advantageous to integrate FPGA with other simulators to form an integrated simulation platform. This platform will inherit the merits of better visualization and flexibility from commercial real-time simulators and the fast computational speed and parallelism from FPGAs. This thesis, by integrating the FPGA with RTDS, tries to achieve both fast simulation speed and high simulation accuracy for the simulation of large power systems.

1.3 Project Objective, Contributions and Thesis Outline

1.3.1 Project Objectives

The main objectives of the project are to:

1. Develop detailed EMT models of power system components suitable for implementation in FPGA. Models for synchronous machines, transmission lines, passive components, voltage/current sources and circuit breakers should be included.
2. Develop detailed EMT models of non-linear control systems for synchronous machines. Models for excitation systems and governor/turbine systems should be included.
3. Implement the developed models into hardware modules in FPGA. Pipelined computation schemes and paralleled architecture between hardware modules should be carefully designed.
4. Develop a generic FPGA-based EMT simulator for power system simulation.
5. Integrate FPGA with RTDS to develop a real-time FPGA-RTDS co-simulator.
6. Expand the capability of the co-simulator by introducing multi-FPGA architecture for the simulation of large scale power systems.

1.3.2 Contributions of the Thesis

The main contributions of the work in this thesis are summarized as follows:

1. The co-simulation methodology integrating FPGA and RTDS for real-time large scale power system simulations.
2. The massively paralleled and deeply pipelined FPGA algorithms for the EMT modelling of power system components and control systems.
3. The scalable multi-FPGA structure and algorithm in simulating very large scale power systems.

1.3.3 Thesis Outline

The outline of the thesis is as follows:

Chapter 2 In this chapter, a library of power system components using EMT models is developed in FPGA, including most of the common power system elements (synchronous machines, transmission lines, passive elements, voltage/current sources and circuit breakers) and their control systems (excitation systems and governor systems). The EMT modelling and hardware implementation of each module are described. Comparisons between FPGA and SIMULINK are made to verify the accuracy of the developed models.

Chapter 3 This chapter describes the hardware implementation of FPGA based real-time simulator from a system point of view. Deeply pipelined and massively paralleled computation scheme of FPGA are designed to minimize the requirement on hardware resources. A four-machine two-area power system is simulated using FPGA and comparisons are made with RTDS to verify the accuracy of the simulator and the developed algorithms.

Chapter 4 By integrating FPGA and RTDS together, the FPGA-RTDS co-simulator is developed. The mathematical foundations and hardware implementation of the interface between FPGA and RTDS are explained in detail. Case study of the four-machine two-area power system with one area simulated in FPGA and the other area simulated in RTDS is carried out and comparisons are made with the complete system simulated in RTDS. In addition, simulation results of a 141-bus system are presented to demonstrate the capability of the co-simulator. The expandability of the co-simulator and the relationships between the hardware resource, simulation time-step and the size of system is discussed in detail.

Chapter 5 In this chapter, multi-FPGA structure is proposed and the multi-FPGA-RTDS co-simulator is developed for the simulation of large scale power systems. Methods and guidelines of network partition are discussed for the allocation of computational load among FPGAs. Two types of architectures are proposed for the co-simulator. The advantages and

disadvantages of both architectures and corresponding interface designs are discussed. Simulation case study utilizing 10 FPGAs to simulate a large scale power system with more than 4000 nodes is presented to demonstrate the capability of the proposed co-simulator.

Chapter 6 This chapter draws the conclusion of the thesis and discusses possible future works.

1.3.4 Logical Relationships between Chapters

The relationships between the chapters are described as follows:

Chapter 2 develops a library of power system components and control systems in FPGA, which lays the foundation for later chapters.

Chapter 3, using the models developed in **Chapter 2**, implements the FPGA-based EMT simulator.

Chapter 4, based on the FPGA-based EMT simulator developed in **Chapter 3**, designs the interface and coordination between FPGA and RTDS and develops the FPGA-RTDS co-simulator.

Chapter 5 further expands the simulation capability of the co-simulator developed in **Chapter 4** by introducing the multi-FPGA structure to achieve the simulation of large-scale power systems.

Chapter 2

MODELLING AND HARDWARE IMPLEMENTATION OF POWER SYSTEM ELEMENTS

2.1 Introduction

This chapter explains the modelling of power system elements and their hardware implementations in FPGA. Section 2.2 describes the modelling and hardware implementation of the most commonly used power system components including synchronous machines, transmission lines, passive elements, voltage/current sources and circuit breakers. Section 2.3 describes the modelling and hardware implementation of control systems for synchronous machines. Section 2.4 verifies the developed models by comparing simulation results from FPGA with those from SIMULINK. Section 2.5 summarizes this chapter.

2.2 Power System Elements

2.2.1 Synchronous Machines

The synchronous machine is an indispensable part of the power system. It involves dynamics from both mechanical and electrical parts. The nonlinear behaviour of synchronous machine is closely related to the stability and dynamics of electrical power systems.

2.2.1.1 Overview of Solution for Synchronous Machine

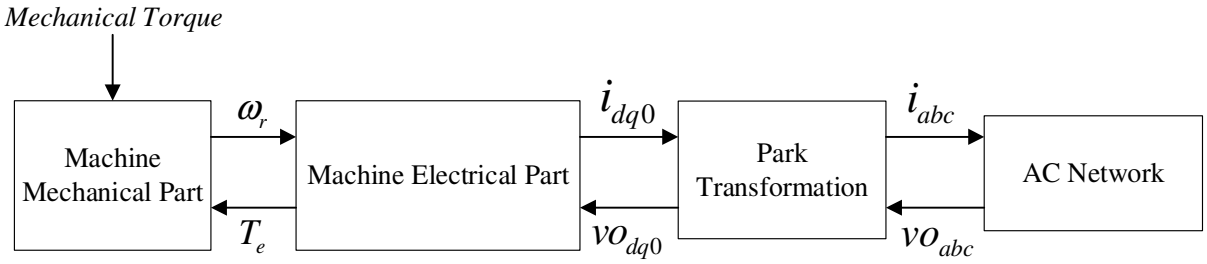


Figure 2-1 Block diagram representation of synchronous machine

Figure 2-1 shows the block diagram representation of synchronous machine. For the solution of synchronous machine, the following computational blocks are identified:

1. Electrical part.
2. Mechanical part.
3. External AC network

The mechanical part takes the inputs of mechanical torque T_m and electromagnetic torque T_e and calculates the rotor speed ω_r. The machine electrical part has the inputs of rotor speed ω_r and equivalent voltage source of external AC network vo_{dq0}, and calculates the terminal stator current in *dq0* reference frame i_{dq0}. The external AC network provides the equivalent Thevenin voltage source vo_{abc} for the solution of machine electrical part. Park transformation is utilized to transform the variables from *abc* to *dq0* reference frames.

The model of each part suitable for EMT implementation is described in following sections followed by the details solution steps of synchronous machine.

2.2.1.2 Model Formulation

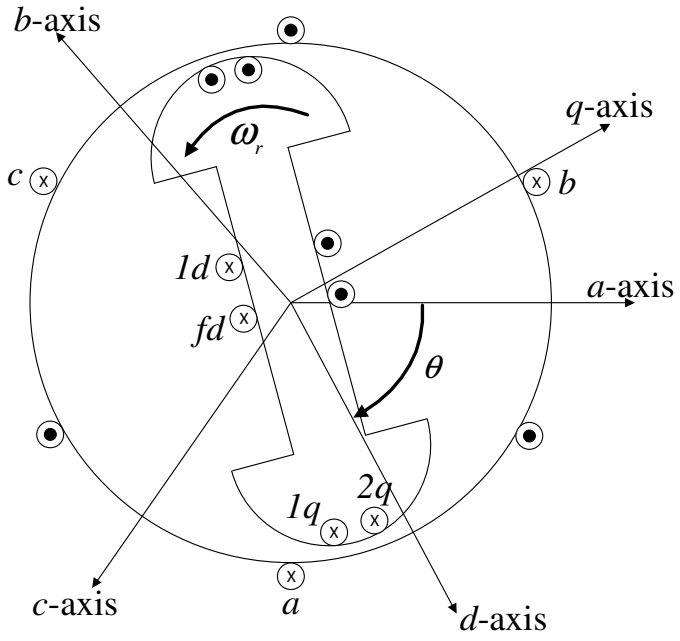


Figure 2-2 Synchronous machine schematic

Figure 2-2 shows a simplified schematic of the synchronous machine with the coil orientation, current polarities and rotor position reference. One damper winding on *d*-axis and two damper windings on *q*-axis are considered in this thesis. The direct axis (*d*-axis) lags the quadrature axis (*q*-axis) by 90 degrees. θ is the angle different between *d*-axis and the phase *a* axis. The direction of current on rotor winding is defined as positive when flowing into the machine; the direction of current on stator windings is defined as positive when flowing out of the machine.

Under the above definitions, the Park transformation which transforms the three-phase variables in abc reference frame into $dq0$ reference frame can be expressed as [147]:

$$P = \frac{2}{3}\begin{bmatrix} \cos(\theta) & \cos(\theta-\frac{2\pi}{3}) & \cos(\theta+\frac{2\pi}{3}) \\ -\sin(\theta) & -\sin(\theta-\frac{2\pi}{3}) & -\sin(\theta+\frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.1}$$

The inverse of Park transformation is given by

$$P^{-1} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 1 \\ \cos(\theta-\frac{2\pi}{3}) & -\sin(\theta-\frac{2\pi}{3}) & 1 \\ \cos(\theta+\frac{2\pi}{3}) & -\sin(\theta+\frac{2\pi}{3}) & 1 \end{bmatrix} \tag{2.2}$$

Electrical Part of Synchronous Machine

The complete dynamic behaviour of synchronous generator in *dq0* reference frame can be written as [147]:

$$\begin{bmatrix} v_d \\ v_q \\ v_0 \\ v_f \\ 0 \\ 0 \\ 0 \end{bmatrix} = -\begin{bmatrix} R_a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & R_a & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & R_a & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -R_{fd} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -R_{1d} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -R_{1q} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -R_{2q} \end{bmatrix}\begin{bmatrix} i_d \\ i_q \\ i_0 \\ i_{fd} \\ i_{1d} \\ i_{1q} \\ i_{2q} \end{bmatrix} + \frac{1}{\omega_{base}}\frac{d}{dt}\begin{bmatrix} \lambda_d & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_q & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_{fd} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda_{1d} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda_{1q} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \lambda_{2q} \end{bmatrix} + \begin{bmatrix} -\omega_r\lambda_q \\ \omega_r\lambda_d \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.3}$$

and

$$\begin{bmatrix} \lambda_d \\ \lambda_q \\ \lambda_0 \\ \lambda_{fd} \\ \lambda_{1d} \\ \lambda_{1q} \\ \lambda_{2q} \end{bmatrix} = \begin{bmatrix} -(L_{ad}+L_l) & 0 & 0 & L_{ad} & L_{ad} & 0 & 0 \\ 0 & -(L_{aq}+L_l) & 0 & 0 & 0 & L_{aq} & L_{aq} \\ 0 & 0 & -L_0 & 0 & 0 & 0 & 0 \\ -L_{ad} & 0 & 0 & L_{ffd} & L_{f1d} & 0 & 0 \\ -L_{ad} & 0 & 0 & L_{f1d} & L_{11d} & 0 & 0 \\ 0 & -L_{aq} & 0 & 0 & 0 & L_{11q} & L_{aq} \\ 0 & -L_{aq} & 0 & 0 & 0 & L_{aq} & L_{22q} \end{bmatrix} \begin{bmatrix} i_d \\ i_q \\ i_0 \\ i_{fd} \\ i_{1d} \\ i_{1q} \\ i_{2q} \end{bmatrix} \tag{2.4}$$

where the definitions of variables are listed in Table 2.1. In (2.3) and (2.4), all variables are in per unit except time, which is in second.

Table 2.1 Definitions of Variables for Synchronous Machine

Name	Definition	Name	Definition
v_d	*d*-axis stator voltage	λ_{1d}	*1d* damper winding flux linkage
v_q	*q*-axis stator voltage	λ_{1q}	*1d* damper winding flux linkage
v_0	*0*-axis stator voltage	λ_{2q}	*2q* damper winding flux linkage
v_f	Excitation voltage	L_{ad}	*d*-axis mutual inductance
i_d	*d*-axis stator current	L_{aq}	*q*-axis mutual inductance
i_q	*q*-axis stator current	L_l	Leakage inductance
i_0	*0*-axis stator current	L_{ffd}	Field winding self-inductance
i_{fd}	Field circuit current	L_{f1d}	Field winding leakage inductance
i_{1d}	*1d* damper circuit current	L_{11d}	*1d* damper winding self-inductance
i_{1q}	*1q* damper circuit current	L_{11q}	*1q* damper winding self-inductance
i_{2q}	*2q* damper circuit current	L_{22q}	*2q* damper winding self-inductance
λ_d	*d*-axis flux linkage	L_0	0-axis self-inductance
λ_q	*q*-axis flux linkage		
λ_0	*0*-axis flux linkage		
λ_{fd}	Field winding flux linkage		

The expressions of (2.3) and (2.4) can be simplified as:

$$v_{dq0} = -Ri_{dq0} + \frac{1}{\omega_{base}} \frac{d\lambda_{dq0}}{dt} + u \tag{2.5}$$

$$\lambda_{dq0} = Li_{dq0} \tag{2.6}$$

Equations (2.5) and (2.6) describe the dynamics of electrical part of synchronous machine.

Discretization Using Trapezoidal Rule of Integration

In order to obtain the discrete-time model for implementation on FPGA, trapezoidal rule of integration is applied to (2.5) from t-Δt to t:

$$\begin{aligned} \frac{[v_{dq0}(t) + v_{dq0}(t-\Delta t)]\Delta t}{2} &= -R\frac{[i_{dq0}(t) + i_{dq0}(t-\Delta t)]\Delta t}{2} \\ &+ \frac{1}{\omega_{base}}[\lambda_{dq0}(t) - \lambda_{dq0}(t-\Delta t)] + \frac{[u(t) + u(t-\Delta t)]\Delta t}{2} \end{aligned} \tag{2.7}$$

Rearrange (2.7)

$$v_{dq0}(t) = -Ri_{dq0}(t) + \frac{2}{\omega_{base}\Delta t}\lambda_{dq0}(t) + u(t) + v_{hist} \tag{2.8}$$

where

$$v_{hist} = -v_{dq0}(t-\Delta t) - Ri_{dq0}(t-\Delta t) - \frac{2}{\omega_{base}\Delta t}\lambda_{dq0}(t-\Delta t) + u(t-\Delta t) \tag{2.9}$$

From (2.8) it can be seen that a linear relationship between the stator voltage and current are obtained for each time-step if the rotor speed is known. The history term of v_{hist} is calculated using values from previous time-step.

Representation of External AC System

External AC system is solved without the machine to get the Thevenin equivalent impedance R_{th} and Thevenin voltage source vo_{abc} (details will be shown in Chapter 3). Then the AC system can be represented by a Thevenin equivalent circuit:

$$v_{abc} = vo_{abc} + R_{th} i_{abc} \tag{2.10}$$

Transform (2.10) into *dq0* reference frame by multiplying *P* (Park transformation matrix) to both sides,

$$\begin{bmatrix} v_d \\ v_q \\ v_0 \end{bmatrix} = \begin{bmatrix} vo_d \\ vo_q \\ vo_0 \end{bmatrix} + R_{th\_dq0} \begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} \tag{2.11}$$

where

$$\begin{bmatrix} vo_d \\ vo_q \\ vo_0 \end{bmatrix} = P \begin{bmatrix} vo_a \\ vo_b \\ vo_c \end{bmatrix} \tag{2.12}$$

And

$$R_{th\_dq0} = P R_{th} P^{-1} \tag{2.13}$$

To solve the stator currents in *dq0* reference frame, (2.11) is substituted into (2.8):

$$\begin{bmatrix} vo_d \\ vo_q \\ vo_0 \\ v_f \\ 0 \\ 0 \\ 0 \end{bmatrix} + R_{sys} i_{dq0} = -R i_{dq0} + \frac{2}{\omega_{base} \Delta t} \lambda_{dq0} + u + v_{hist} \tag{2.14}$$

where

$$R_{sys} = \begin{bmatrix} R_{th\_dq0(3\times3)} & 0_{3\times4} \\ 0_{4\times3} & 0_{4\times4} \end{bmatrix} \tag{2.15}$$

Substitute (2.6) into (2.14),

$$(R_{sys} + R - \frac{2}{\omega_{base} \Delta t} L - WL) i_{dq0} = v_{hist} - \begin{bmatrix} vo_d \\ vo_q \\ vo_0 \\ v_f \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.16}$$

where

$$W = \begin{bmatrix} 0 & -\omega_r & 0 & 0 & 0 & 0 & 0 \\ \omega_r & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.17}$$

(2.16) is used to calculate stator current using the known history term of v_{hist} and the calculated Thevenin equivalent of external AC system.

Mechanical Part of Synchronous Machine

According to Newton's second law, the mechanical part of the machine can be described by the following equations [147]:

$$\overline{T}_m = 2H\frac{d\overline{\omega}_r}{dt} + \overline{D}\overline{\omega}_r + \overline{T}_e \tag{2.18}$$

$$T_e = \lambda_d i_q - \lambda_q i_d \tag{2.19}$$

where H is the per unit inertia constant and D is the damping coefficient. In (2.18) and (2.19), all variables are in per unit except time. The unit of time in (2.18) is in second.

Discretisation of Mechanical Equations

According to the trapezoidal rule of integration, (2.18) can be discretised as

$$\begin{aligned}\frac{[T_m(t)+T_m(t-\Delta t)]\Delta t}{2} &= 2H[\omega(t)-\omega(t-\Delta t)]\\ +D\frac{[\omega(t)+\omega(t-\Delta t)]\Delta t}{2} &+ \frac{[T_e(t)+T_e(t-\Delta t)]\Delta t}{2}\end{aligned} \tag{2.20}$$

It can be simplified to

$$T_m(t) = (\frac{4H}{\Delta t} + D)\omega(t) + T_e(t) + T_{hist} \tag{2.21}$$

where

$$T_{hist} = -T_m(t-\Delta t) + (D - \frac{4H}{\Delta t})\omega(t-\Delta t) + T_e(t-\Delta t) \tag{2.22}$$

(2.21) is used to calculate the rotor speed with the inputs of mechanical and electromagnetic torque and history term from previous time-step.

Estimation of Rotor Speed & Rotor Angle

In order to calculate the machine terminal current, the machine electrical part requires the rotor speed and Thevenin equivalent voltage source of the network. However, the machine rotor speed is unknown at the beginning of the time-step, and estimation of rotor speed is needed for the calculation. Consider the large inertia of rotor and small simulation time-step of, simple linear estimation can be adopted:

$$\omega_p(t) = 2\omega(t-\Delta t) - \omega(t-2\Delta t) \tag{2.23}$$

In (2.23), all variables are expressed in per unit. Then the rotor position can be estimated using trapezoidal rule of integration,

$$\Delta\theta = \theta(t) - \theta(t-\Delta t) = \frac{\Delta t}{2}[\omega(t-\Delta t) + \omega_p(t)]\omega_{base} \tag{2.24}$$

Notice that in (2.24), the rotor speed is in per unit, and the unit for time is second. The base value of rotor speed is multiplied to get the rotor position in radians.

Solution Steps for Synchronous Machine

With the derived mathematical models for electrical and mechanical part of the machine, the solution steps can be summarized. Figure 2-3 shows the flow chart describing the solution steps for one time-step. Firstly, the Thevenin equivalent impedance and voltage source of the external AC network is calculated (details are explained in Chapter 3). The rotor speed and rotor position are also estimated. Secondly the Park transformation matrix is calculated using

the estimated rotor position. Thirdly the stator terminal currents are calculated, followed by the calculation of electromagnetic torque. Then the rotor mechanical speed is calculated using the calculated electromagnetic torque. Finally, the current injections from synchronous machine are exported to the external AC system for solution of the whole system.

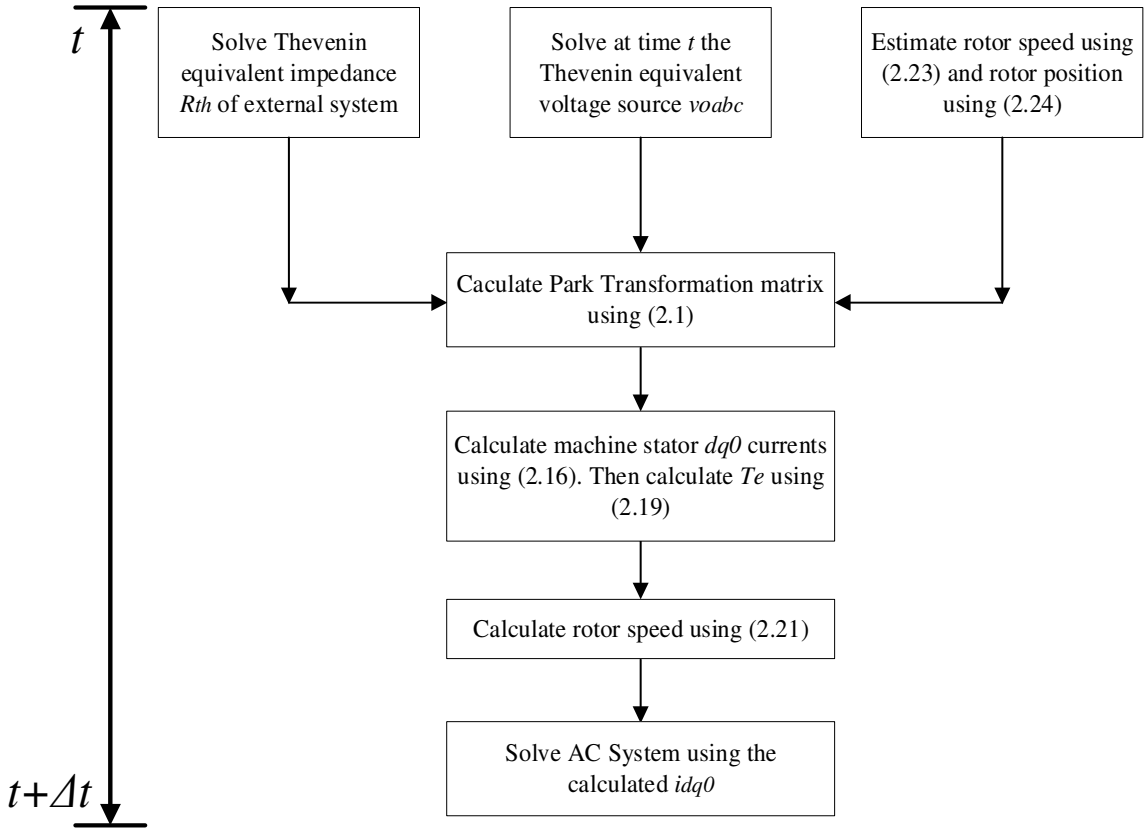


Figure 2-3 Solution steps for synchronous machine

2.2.1.3 Hardware Implementation

The hardware implementation of synchronous machine consists of five modules: rotor speed and position prediction module, Park and inverse Park transformation module, stator current module, flux linkage calculation module and electromagnetic torque & rotor speed module.

In this section, they will be described in detail according to the order of solution steps.

2.2.1.3.1 Rotor Speed and Position Prediction Module

Firstly, the rotor speed and rotor position are estimated by the rotor speed and position prediction module. Predicted machine rotor speed for the current time-step is made using rotor speed from last two time-steps as shown in equation (2.23). Then rotor angle is calculated using the predicted rotor speed as equation (2.24).

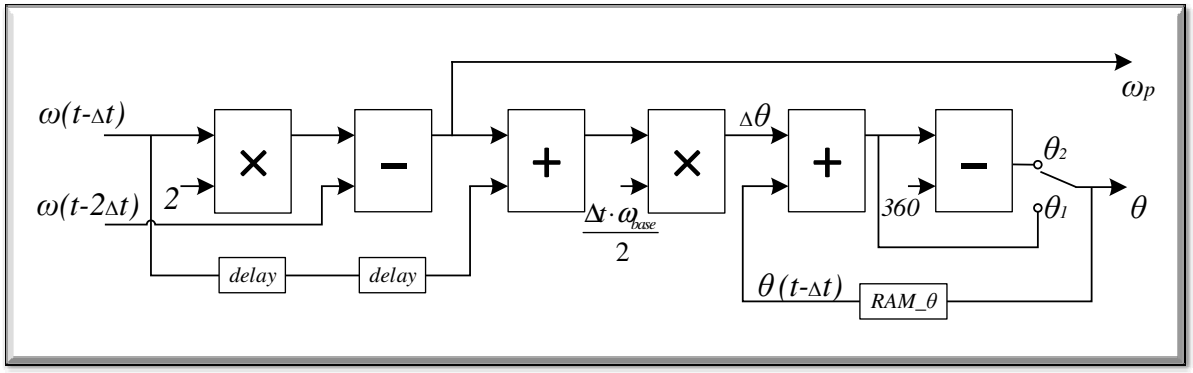


Figure 2-4 Rotor speed and position prediction module

As shown in Figure 2-4, the rotor speed and position prediction module consists of two floating point multipliers, two floating point adder and two floating point subtractors.

The algorithm in each time-step of this module is shown below:

1. Rotor Speed ω from last two time-steps t-Δt and t-$2\Delta t$ are fed into the calculation path and $\omega_p(t) = 2\omega(t-\Delta t) - \omega(t-2\Delta t)$ is calculated.
2. Then $\Delta\theta = \frac{\Delta t}{2}[\omega(t-\Delta t) + \omega_p(t)] \cdot \omega_{base}$ is calculated using ωp and ω(t-Δt).
3. Due to the requirement from downstream calculations, rotor angle θ is limited between 0 - 360 degrees. Two new rotor angles $\theta_1(t) = \Delta\theta + \theta(t-\Delta t)$ and $\theta_2(t) = \Delta\theta + \theta(t-\Delta t) - 360^\circ$ are calculated and a multiplexer is used to select the eligible one.

2.2.1.3.2 Park Transformation Module

With the calculated rotor angle, the Park Transformation matrix can be calculated using equation (2.1). For the convenience of hardware implementation, equation (2.1) is rearranged into (2.25). So there are only positive *sine* functions in the equation which means only one Look-Up Table (LUT) of *sine* is needed.

$$P=\frac{2}{3}\begin{bmatrix} \sin(\theta+\frac{\pi}{2}) & \sin(\theta+\frac{11\pi}{6}) & \sin(\theta+\frac{7\pi}{6}) \\ \sin(\theta+\pi) & \sin(\theta+\frac{\pi}{3}) & \sin(\theta+\frac{5\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.25}$$

Among the nine entries in P matrix, three are constant numbers 1/3, which require no calculation; the other six entries are processed in a pipeline. As shown in Figure 2-5, θ is rotor angle from the rotor speed and position prediction module, θ_0 is the phase angles of the 6 entries, which are stored in a dedicated ROM. The content of ROM_θ_0 is shown in Figure 2-6 where N is the size of *sine* LUT. In this thesis, $N = 4096$ and only half cycle ($0\sim180^\circ$) of *sine* function is stored to improve the accuracy. $\Delta\beta$ is the minimum unit angle in LUT and the value of $\Delta\beta = 180°/4096 \approx 0.044°$ is used in this thesis. The P matrix is calculated by rows in the order of $P_{11}, P_{12}, P_{13}, P_{21}, P_{22}, P_{23}$, and the details are shown as follows:

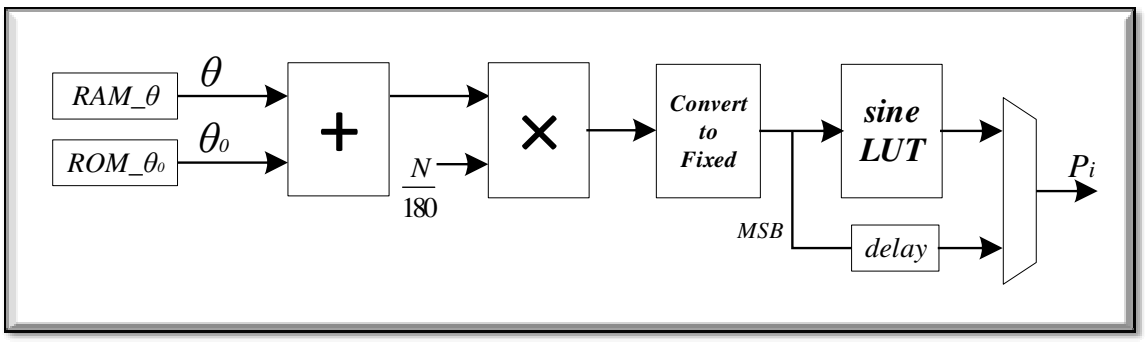


Figure 2-5 Park transformation module

ROM_θ₀	
	90
	330
	210
	180
	60
	300

sineLUT	
	sin(0)
	sin($\Delta\beta$)
	sin($2\Delta\beta$)
	sin($3\Delta\beta$)
	sin($4\Delta\beta$)
	⋮
	sin($(N-1)\Delta\beta$)

Figure 2-6 Read-only memory of Park transformation module

Firstly, the rotor angle from pervious hardware module is added by θ_0, then multiplied by $N/180$. The result is the address for the $sine$ LUT. However, it needs to be converted into fixed-point format integer before using as an address input of LUT. Since there is only half cycle of $sine$ function, the sign of LUT output will be inverted when $\theta + \theta_0$ is larger than $180°$. The detailed algorithm is shown below:

1. The rotor angle θ is read from RAM.
2. The θ is added by θ_0.
3. The result from step 2 is multiplied by $N/180$.
4. The result from step 3 is converted into fixed-point format integer.
5. The result from step 4 is used as an address input of LUT.
6. The final result of the Park transformation module is obtained by combining the most significant bit (MSB) of the address signal and the output of LUT.

For calculating the inverse of Park transformation matrix, the following relationship can be exploited to simplify the calculation:

$$P^{-1} = (PC)^T; \quad C = \begin{bmatrix} 2 & 0.5 & 0.5 \\ 0.5 & 2 & 0.5 \\ 0.5 & 0.5 & 2 \end{bmatrix} \tag{2.26}$$

It can be seen from (2.26) that the complex calculation of inverse is replaced with the multiplication of P and a constant matrix C. The multiplication of $P \times C$ is calculated using matrix multiplication module, which will be described in later section, and the address of results is customized to achieve matrix transpose.

2.2.1.3.3 Matrix Multiplication Module

With the calculated Park transformation matrix, the equivalent voltage source of external AC network in abc reference frame is converted into $dq0$ reference frame using the matrix multiplication module. The matrix multiplication module is one of the most frequently used modules in the proposed simulator. It is used in several other modules including the $abc/dq0$ reference frame transformation of terminal voltages and currents, inverse of Park transformation matrix and flux linkage module.

The hardware implementation of matrix multiplication module is shown in Figure 2-7.

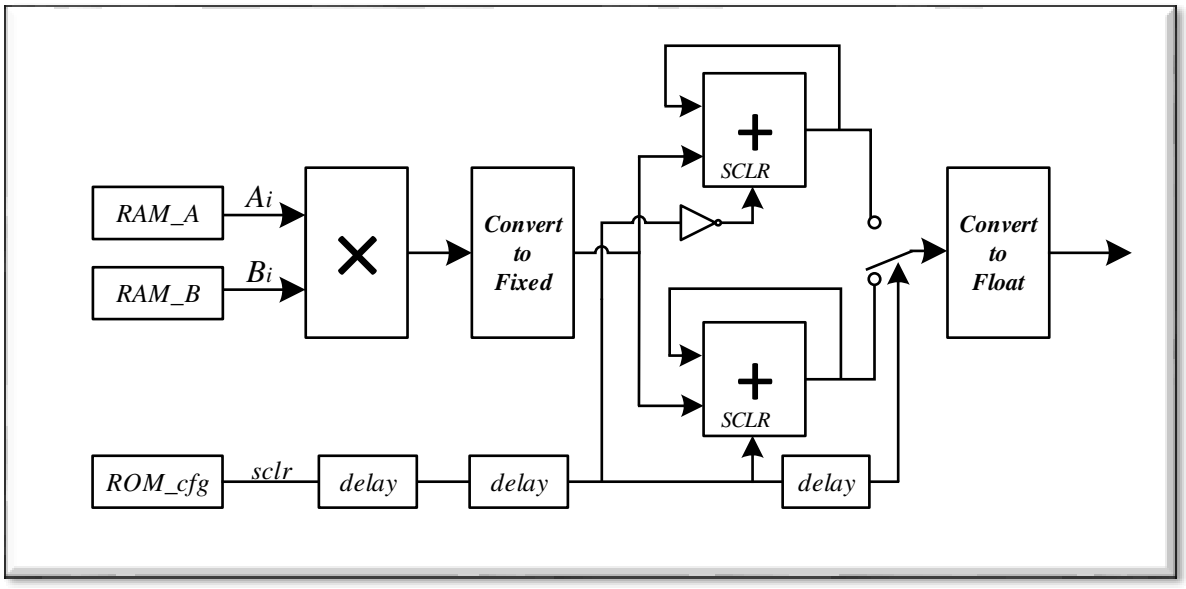


Figure 2-7 Matrix multiplication module

When performing matrix multiplication $A \times B$, only non-zero entries of matrixes are stored in RAM. Control signals, including read address of RAMs, synchronous clear (SCLR) signals for accumulators and write address for results, are stored in ROM_cfg. The algorithm in each FPGA clock cycle is shown below:

1. The matrix entries $A_i$and B_i are read from RAM_A and RAM_B,
2. Then multiplications $A_i \times B_i$ are performed by a floating-point multiplier.
3. The results from step 2 are converted into fixed-point number and fed into two accumulators.
4. The two accumulators run alternatively under the control of $SCRL$ signals. When one result is calculated, the accumulator will be reset and, at the same time, the other accumulator starts its calculation without any delay.
5. The $SCRL$ signal is also used to select the output from the two accumulators.
6. The output from accumulator is converted back to floating-point format.

In step 3, fixed-point accumulators are used to keep the pipeline running as it can ensure that each accumulation operation is completed in one clock cycle. If floating-point adder is used, several clock cycles will be required to complete one operation and pipelining cannot be achieved.

2.2.1.3.4 Stator Current Module

Stator current module is the most time- and resource- consuming module for the solution of synchronous machine. It takes the predicted rotor speed ω_p, historic voltage v_{hist}, excitation voltage v_f equivalent resistance of the network and network Thevenin equivalent voltage v_{dq0_0} as inputs to calculate the machine stator current i_{dq0}.

The stator current is calculated using equation (2.16). It is a set of 7 linear equations, which can be express as

$$X \cdot i_{dq0} = Y \tag{2.27}$$

where

$$X = R_{sys} + R - \frac{2}{\omega_{base}\Delta t} L - WL; \qquad Y = v_{hist} - \begin{bmatrix} vo_d \\ vo_q \\ vo_0 \\ v_f \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.28}$$

Gaussian elimination method is utilized in FPGA to solve this set of linear equations.

The stator current module can be divided into two submodules: the matrix formation submodule and the linear equations solver submodule.

Matrix formation submodule:

In matrix X, $R_{sys}, R, \omega_{base}, \Delta t, L$ are all constant, which are stored in read-only memory and imported in every time-step. Only W (2.17) containing rotor speed ω_r requires calculation. In matrix Y, the first four entries are calculated using floating-point subtraction and the last three entries are equal to corresponding history term values. For a system with n synchronous generators, the matrix X and Y for each generator are combined into a single large XY matrix as illustrated in Figure 2-8. The combined matrix is stored in RAMs in the form of rows. The seven rows of the combined matrix are stored in seven different RAMs.

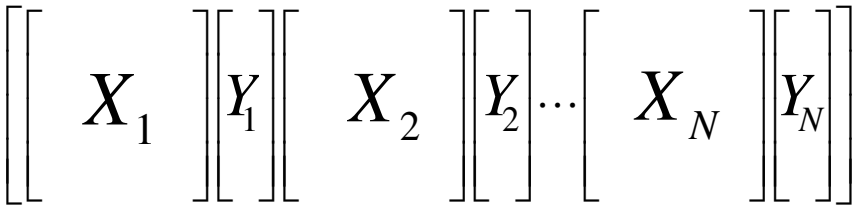


Figure 2-8 XY matrix format

Linear equations solver submodule:

When the XY matrix is ready, the linear equations are solved using the Gaussian elimination method. The hardware implementation of this module is shown in Figure 2-9. It contains one floating-point division module, seven RAMs and seven elimination modules. All the calculations are using single precision floating point numbers. The calculation process of this module is deeply pipelined and highly paralleled. It can be seen in Figure 2-9 that there is a specific hardware calculation channel for each row of the *XY* matrix. Seven hardware channels for the seven rows are calculated in parallel to improve the calculation efficiency.

The method of solving linear equations using matrix reduction is explained in [43] and is briefly reviewed here. For example for a set of linear equations expressed in matrix format

$$Ax = b \tag{2.29}$$

Triangular factorization is applied to matrix *A* so that it becomes an identity matrix. The same sequence of linear operations is applied to matrix *b* and the resulting matrix then contains the solutions of original linear equations.

For the hardware implementation, there are two output channels for each RAM as shown in Figure 2-9. The channel highlighted in red outputs all the elements in a row one by one, while

the channel highlighted in blue outputs the n^{th} element of the same row. To factorize the n^{th} element in n^{th} row,

1. The division module takes the outputs from the n^{th} RAM.
2. The division module then divides all the elements in n^{th} row (red channel) by the n^{th} element in the same row (blue channel). As a result, the n^{th} element in n^{th} row becomes 1.
3. The n^{th} row after division from step 2 now becomes the output for division module and is sent to the variable elimination modules (orange blocks) through the channel highlighted in green.
4. The same row from step 3 is then multiplied by the n^{th} element in each row,
5. The results from step 4 are subtracted from the original rows in the matrix, forming a new matrix where the n^{th} element in each row becomes zero.
6. This matrix is saved in RAMs and the same computation is carried out for all values of n.
7. Finally when matrix A becomes an identity matrix, the resulting elements in b are the solution of the original linear equation (2.29).

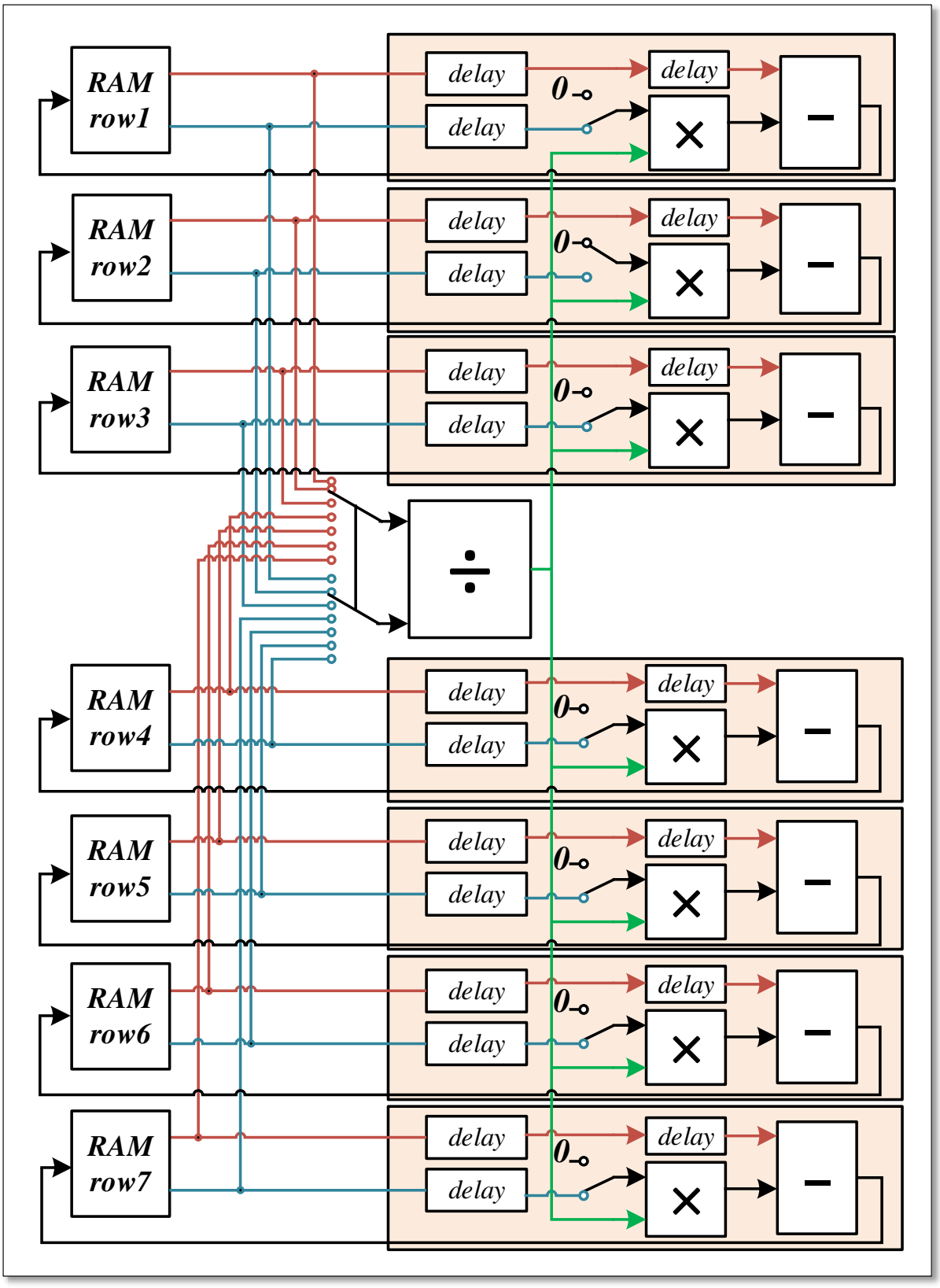


Figure 2-9 Stator current solving module

2.2.1.3.5 Electromagnetic Torque & Rotor Speed Module

With the calculated values of *idq0* from the Stator Current Module, flux linkages can be calculated using (2.6), and the stator current in *abc* reference frame can be calculated using the inverse of Park Transformation. The Matrix Multiplication Module described in Section 2.2.1.3.3 is used for these calculations.

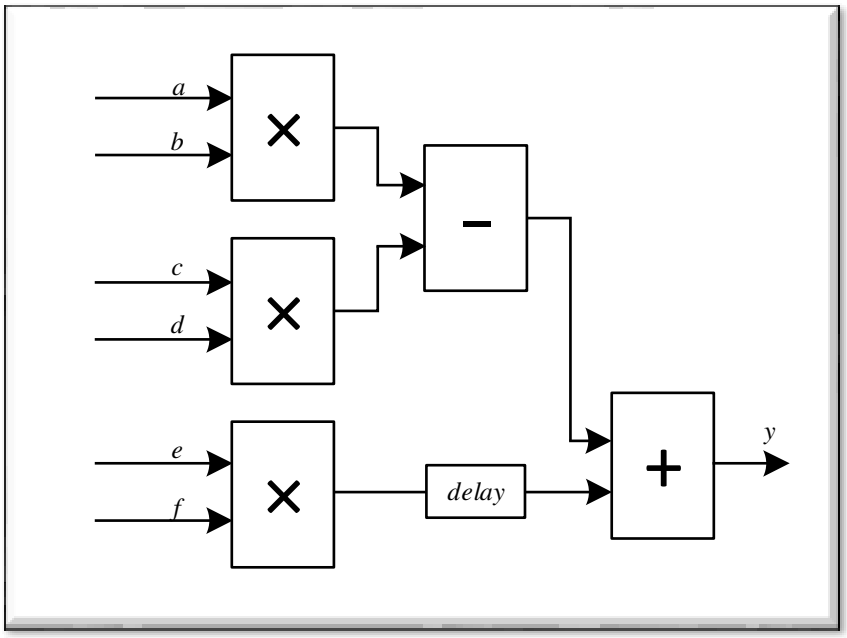


Figure 2-10 Electromagnetic torque & rotor speed module

Once the flux linkages are calculated, electromagnetic torque T_e is calculated using (2.19), and the rotor speed ω_r is calculated using (2.21). As (2.19) and (2.21) share the same calculation format, i.e., $y=ab+cd+ef$, the same hardware module is used as shown in Figure 2-10. The algorithm of calculating electromagnetic torque T_e in each time-step is:

1. The flux linkages λ_q, λ_d are fed into input paths *a* and *c* respectively, while the stator currents i_q and i_d are fed into input paths *b* and *d* respectively. Inputs *e* and *f* are forced to zero.

2. $T_e = \lambda_d i_q - \lambda_q i_d$ is calculated using the upper two multipliers and one subtractor in Figure 2-10. The output from the lower multiplier is zero.
3. T_e is assigned to the variable *y* as the output of the module.

The algorithm of calculating rotor speed ω_r in each time-step is:

1. Constant parameter *1/(4H/Δt+D)* is fed into input path *a* and *c*, *-1/(4H/Δt+D)* is fed into input path *e*, while mechanical torque T_m, electromagnetic torque T_e and history term T_{hist} are fed into input path *b, d* and *f*.
2. $\omega(t) = 1/(\frac{4H}{\Delta t} + D) \cdot (T_m(t) - T_e(t) - T_{hist})$ is calculated.

2.2.2 Transmission Lines

Another important power system element is the transmission line. This section explains the modelling and FPGA implementation of transmission lines.

2.2.2.1 Model Formulation

Lossless Line

According to the model proposed by [42], a lossless transmission line can be fully described by two Norton equivalents at line terminals as illustrated in Figure 2-11. Mathematically the two terminal currents are calculated as:

$$i_1(t) = \frac{1}{Z} v_1(t) + I_{h1}(t - \tau) \tag{2.30}$$

$$i_2(t) = \frac{1}{Z} v_2(t) + I_{h2}(t - \tau) \tag{2.31}$$

where $v_1(t), v_2(t), i_1(t)$, and $i_2(t)$ are terminal voltages and currents. *Z* is the surge impedance and is given by:

$$Z = \sqrt{\frac{L'}{C'}} \tag{2.32}$$

Where L' and C' are inductance and capacitance per unit length.

History current sources are calculated as:

$$I_{h1}(t-\tau) = -\frac{1}{Z} v_2(t-\tau) - i_2(t-\tau) \tag{2.33}$$

$$I_{h2}(t-\tau) = -\frac{1}{Z} v_1(t-\tau) - i_1(t-\tau) \tag{2.34}$$

Where τ is the travel time of the travelling wave through the length of transmission line. If the length of transmission line is *d*, the travelling time can be calculated as:

$$\tau = d\sqrt{L'C'} \tag{2.35}$$

Linear interpolation is used for the calculation of history currents when the travelling time is not integer times of the simulation time-step. For example if the travelling time is $\tau = k\Delta t$ where k is a non-integer value, the calculation of history current at terminal 1 becomes

$$I_{h1}(t-\tau) = a_0 \times I_{h1}\left(t - [k]\Delta t\right) + a_1 \times I_{h1}\left(t - ([k]+1)\Delta t\right) \tag{2.36}$$

Where $[k]$ represents the nearest integer of k, $a_1 = k - [k]$, and $a_0 = 1 - a_1$.

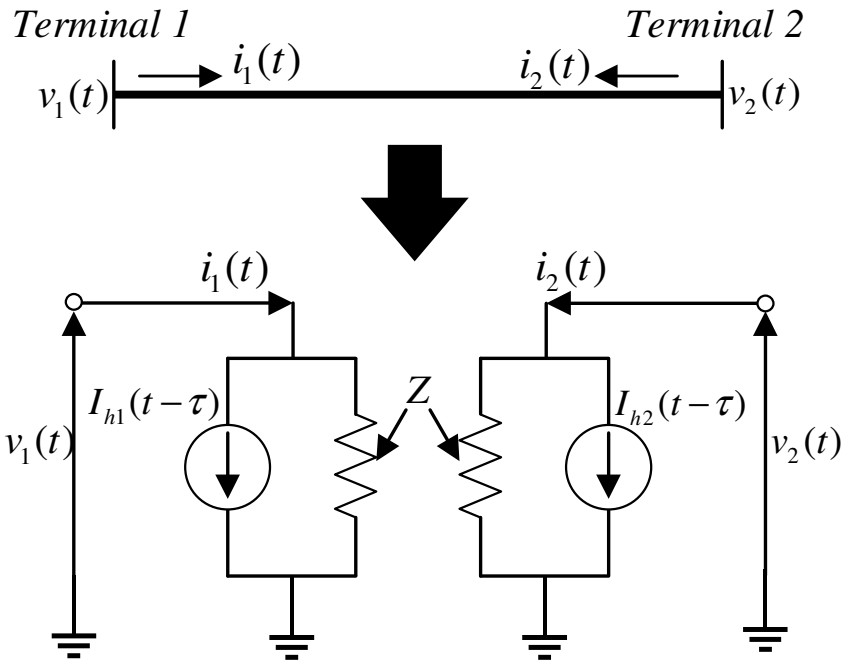


Figure 2-11 Equivalent circuit for lossless transmission line

Transmission Line with Losses

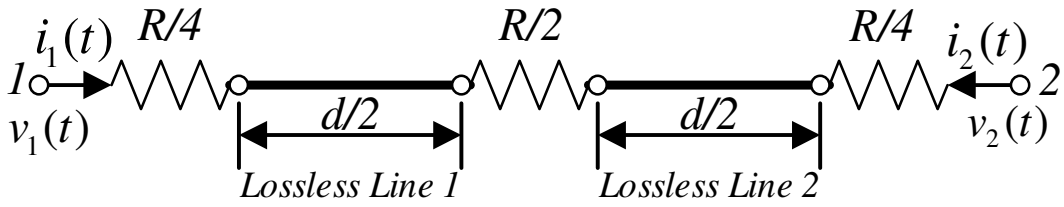


Figure 2-12 Representation of transmission line with losses

To include losses into the transmission line model, three lumped resistors are added to three different locations of a lossless line. It was demonstrated that no noticeable differences between lumped resistors inserted in few or in many places [43]. As shown in Figure 2-12, if the total resistance of transmission line is *R*, two resistors of value *R/4* are located at two ends and one resistors of value *R/2* is located in the middle of the lossless line. Using the lossless line model as calculated in (2.30) and (2.31) for the two lossless lines and the resistor model for the three lumped resistances, the equations for the transmission line with losses as shown

in Figure 2-12 can be obtained. By eliminating the internal variables, the relationships between voltages and currents at terminal 1 and 2 can be calculated as:

$$i_1(t) = \frac{1}{Z + R/4} v_1(t) + I_{h1}(t-\tau) \tag{2.37}$$

$$i_2(t) = \frac{1}{Z + R/4} v_2(t) + I_{h2}(t-\tau) \tag{2.38}$$

where the history terms are

$$\begin{aligned} I_{h1}(t-\tau) = &-\frac{Z}{\left(Z + R/4\right)^2}\left[v_2(t-\tau) + (Z - \frac{R}{4}) \times i_2(t-\tau) \right] \\ &-\frac{R/4}{\left(Z + R/4\right)^2}\left[v_1(t-\tau) + (Z - \frac{R}{4}) \times i_1(t-\tau) \right] \end{aligned} \tag{2.39}$$

$$\begin{aligned} I_{h2}(t-\tau) = &-\frac{Z}{\left(Z + R/4\right)^2}\left[v_1(t-\tau) + (Z - \frac{R}{4}) \times i_1(t-\tau) \right] \\ &-\frac{R/4}{\left(Z + R/4\right)^2}\left[v_2(t-\tau) + (Z - \frac{R}{4}) \times i_2(t-\tau) \right] \end{aligned} \tag{2.40}$$

It can be seen from (2.39) and (2.40) that the calculation of history terms now require information from both ends of the line.

Multiphase Transmission Lines with Mutual Coupling

Mutual couplings exist for multiphase transmission lines. Figure 2-13 shows a segment of a uniformly transposed three-phase mutually-coupled transmission line. It can be seen from the figure that the capacitances between phases are the same, and the zero sequence capacitance for each phase is the same due to the transposition [148].

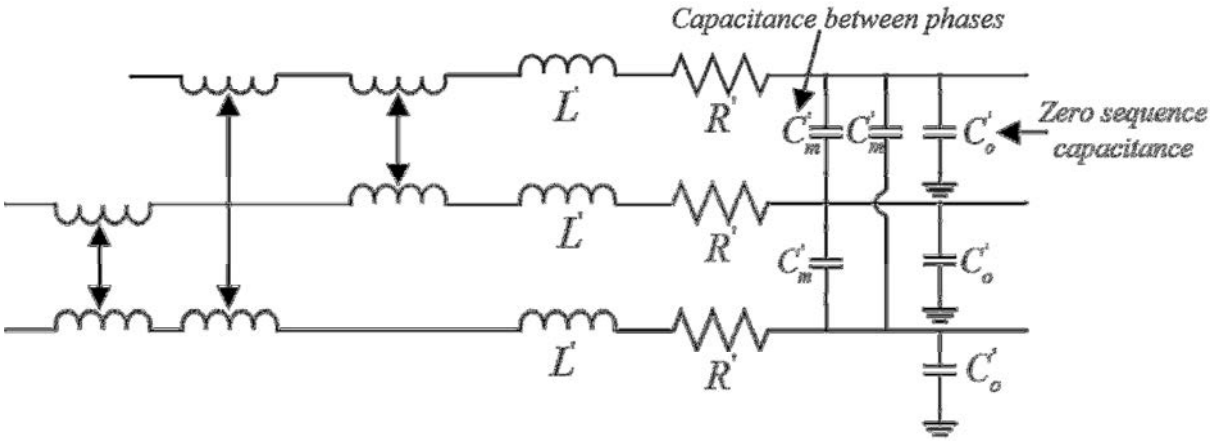


Figure 2-13 A segment of three-phase mutually coupled line [42]

The impedance and admittance matrix of per unit length of transposed transmission line can be expressed as:

$$\left[Z'_{phase}\right]=\begin{bmatrix} Z'_s & Z'_m & Z'_m \\ Z'_m & Z'_s & Z'_m \\ Z'_m & Z'_m & Z'_s \end{bmatrix} \tag{2.41}$$

$$\left[Y'_{phase}\right]=\begin{bmatrix} Y'_s & Y'_m & Y'_m \\ Y'_m & Y'_s & Y'_m \\ Y'_m & Y'_m & Y'_s \end{bmatrix} \tag{2.42}$$

where Z'_s, Y'_s are self-impedance and self-admittance for each phase, and Z'_m , Y'_m are mutual-impedance and mutual-admittance between phases. One possible transformation that can be used to remove the decoupling terms is Clarke transformation:

$$T_v = T_i = \frac{1}{\sqrt{3}}\begin{bmatrix} 1 & \sqrt{2} & 0 \\ 1 & -\frac{1}{\sqrt{2}} & \sqrt{\frac{3}{2}} \\ 1 & -\frac{1}{\sqrt{2}} & -\sqrt{\frac{3}{2}} \end{bmatrix} \tag{2.43}$$

$$[T_v]^{-1} = [T_i]^{-1} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ \sqrt{2} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \sqrt{\frac{3}{2}} & -\sqrt{\frac{3}{2}} \end{bmatrix} \tag{2.44}$$

By applying (2.43) and (2.44) to (2.41) and (2.42), the modal series impedance matrix and modal shunt admittance matrix can be calculated:

$$\left[Z'_{\text{mode}}\right] = [T_v]^{-1}\left[Z'_{phase}\right][T_i] = \begin{bmatrix} Z'_s + 2Z'_m & 0 & 0 \\ 0 & Z'_s - Z'_m & 0 \\ 0 & 0 & Z'_s - Z'_m \end{bmatrix} \tag{2.45}$$

$$\left[Y'_{\text{mode}}\right] = [T_i]^{-1}\left[Y'_{phase}\right][T_v] = \begin{bmatrix} Y'_s + 2Y'_m & 0 & 0 \\ 0 & Y'_s - Y'_m & 0 \\ 0 & 0 & Y'_s - Y'_m \end{bmatrix} \tag{2.46}$$

From (2.45) and (2.46) it can be seen that for uniformly transposed lines, the diagonal elements in $\left[Z'_{\text{mode}}\right]$ are equal to the zero sequence and positive sequence impedances, i.e.,

$$Z'_{zero} = Z'_s + 2Z'_m \tag{2.47}$$

$$Z'_{pos} = Z'_s - Z'_m \tag{2.48}$$

Similarly,

$$Y'_{zero} = Y'_s + 2Y'_m \tag{2.49}$$

$$Y'_{pos} = Y'_s - Y'_m \tag{2.50}$$

With the decoupled modal series impedance matrix and model shunt admittance matrix, the solution of transmission line within each time-step can be carried out in modal domain. The calculation for each mode is the same as that for single-phase transmission line mentioned above. The modal characteristic impedance for each mode is:

$$Z_{char\_mode\_0} = \frac{\sqrt{Z_s^{'} + 2Z_m^{'}}}{\sqrt{Y_s^{'} + 2Y_m^{'}}} = \sqrt{\frac{Z_{zero}^{'}}{Y_{zero}^{'}}} \tag{2.51}$$

$$Z_{char\_mode\_1} = Z_{char\_mode\_2} = \frac{\sqrt{Z_s^{'} - Z_m^{'}}}{\sqrt{Y_s^{'} - Y_m^{'}}} = \sqrt{\frac{Z_{pos}^{'}}{Y_{pos}^{'}}} \tag{2.52}$$

These are then used to calculate the surge admittance matrix in phase domain:

$$\left[Y_{surge}\right] = \left[T_i\right] \begin{bmatrix} \frac{1}{Z_{char\_mode\_0}} & 0 & 0 \\ 0 & \frac{1}{Z_{char\_mode\_1}} & 0 \\ 0 & 0 & \frac{1}{Z_{char\_mode\_2}} \end{bmatrix} \left[T_i\right]^T \tag{2.53}$$

$\left[Y_{surge}\right]$ is entered into the system admittance matrix of power system instead of a scalar value of *1/Z* for single-phase lines. The calculation of $\left[Y_{surge}\right]$ can be done before the start of programme and stay fixed if the transmission lines are intact.

The propagation velocity of electromagnetic wave of overhead line is slightly slower than the velocity of light (300,000 km/s) [147]. Therefore the required length of the line for system decoupling can be calculated by multiplying the propagation velocity and the simulation time-step. With a simulation time-step of 50μs, a transmission line with a length longer than

15km is required. Consider the large numbers of transmission lines within a typical power system, this decoupled way of calculation significantly reduces the computational burden.

Solution Steps for Transmission Lines

Based on the derived multiphase transmission model, Figure 2-14 shows the solution steps of multiphase transmission line within one time-step.

Step 1: The power system nodal voltages are solved in phase domain using the system admittance matrix and history currents. The surge admittance matrix in phase domain as calculated in (2.53) is included in the system matrix.

Step 2: The solved line terminal three-phase voltages and currents are transformed into the mode domain using (2.44):

$$\begin{bmatrix} I_{mode\_0} \\ I_{mode\_1} \\ I_{mode\_2} \end{bmatrix} = [T_i]^{-1} \begin{bmatrix} I_{phase\_a} \\ I_{phase\_b} \\ I_{phase\_c} \end{bmatrix} \tag{2.54}$$

$$\begin{bmatrix} V_{mode\_0} \\ V_{mode\_1} \\ V_{mode\_2} \end{bmatrix} = [T_i]^{-1} \begin{bmatrix} V_{phase\_a} \\ V_{phase\_b} \\ V_{phase\_c} \end{bmatrix} \tag{2.55}$$

Step 3: The calculated voltages and currents at line terminals are used to update history terms in mode domain for each mode using the same equation as single-phase lines.

Step 4: The updated history terms are transformed back to phase domain for use in later time-steps.

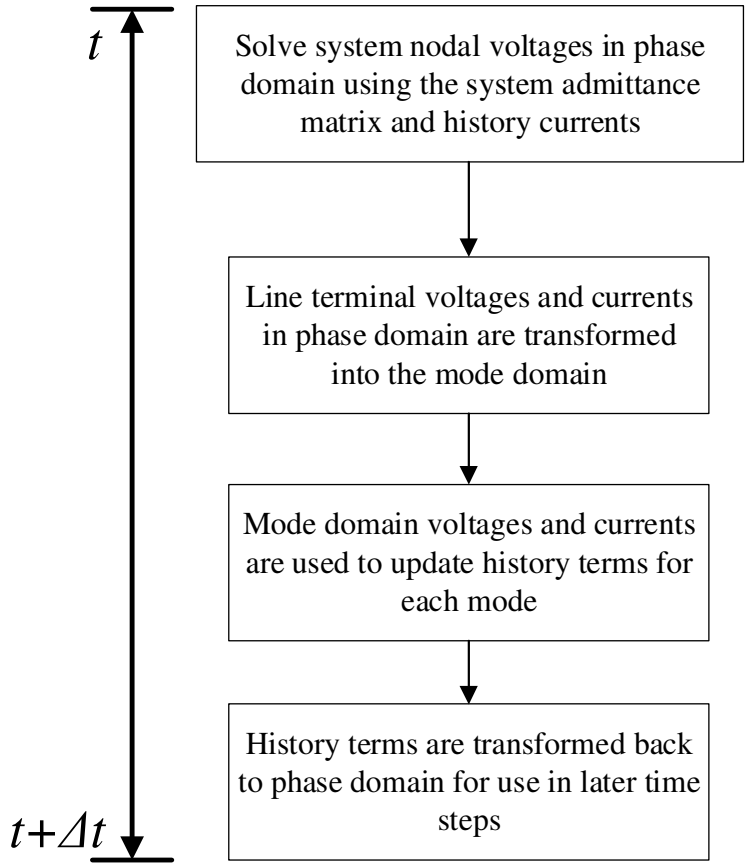


Figure 2-14 Solution steps for multiphase coupled transmission lines

2.2.2.2 Hardware Implementation

The hardware implementation of transmission lines consists of two parts: one is the calculation of history currents and the other is the calculation of line currents.

The calculation of history currents can be further divided into transformation between modal/phase domains and calculation of line currents in modal domain.

Calculation of History Currents

Transformation between modal and phase domain:

As the Clarke transformation matrix (2.43) and its inverse (2.44) are constant matrices, they are calculated and stored in ROM before the start of simulation. Since the transformation is standard matrix multiplication, the Matrix Multiplication Module described in Section 2.2.1.3.3 can again be used.

Calculation of history currents in modal domain:

Equation (2.39) and (2.40) can both be arranged as:

$$I_h(t-\tau) = K_1 \times v_1(t-\tau) + K_2 \times i_1(t-\tau) + K_3 \times v_2(t-\tau) + K_4 \times i_2(t-\tau) \tag{2.56}$$

where coefficients $\mathrm{K}_1, \mathrm{K}_2, \mathrm{K}_3, \mathrm{K}_4$ are constants and can be calculated in advance. The corresponding hardware module is shown in Figure 2-15, including four multipliers and three adders. Among the eight inputs, the four constant coefficients $\mathrm{K}_1, \mathrm{K}_2, \mathrm{K}_3, \mathrm{K}_4$ for history terms $\mathrm{I}_{h1}$and I_{h2} of both sending and receiving ends are stored in ROMs; the other four inputs from other hardware modules are updated every time-step and stored in RAMs. Control signals including read addresses for input RAMs and ROMs and write addresses for results are also stored in a ROM, which will ensure the efficient operation of pipeline.

The detailed algorithm for calculating the history current is shown below:

1. Inputs K_1, K_2, K_3, K_4 are read out from ROMs, $v_1, v_2, i_1, i_2$are read out from RAMs.
2. Four multiplications $K_1 \times v_1,\ K_2 \times\ i_1, K_{3\times} v_2,\ K_4 \times i_2$ are calculated at the same time.
3. The four multiplication results from step 2 are added together.

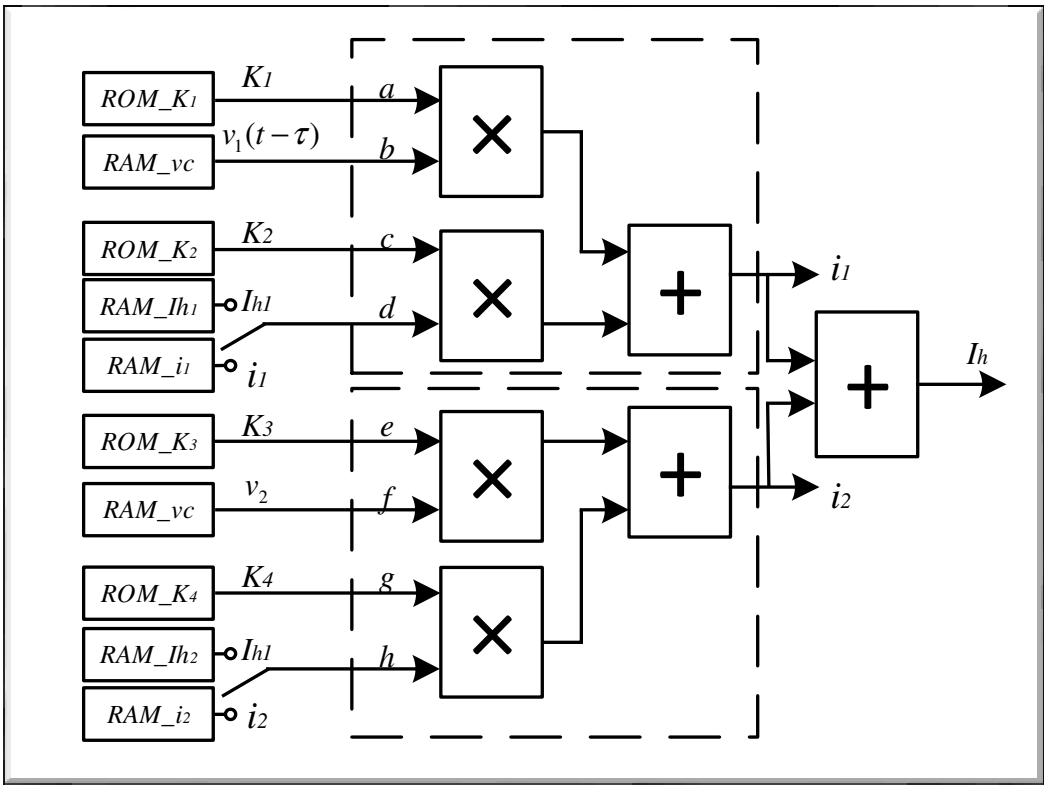


Figure 2-15 Transmission line module

Calculation of line currents:

The equations for the calculation of line currents are shown in (2.37) and (2.38). In general form, they can be expressed as

$$i(t) = \frac{1}{Z + R/4} v(t) + I_h(t-\tau) \tag{2.57}$$

Where $\frac{1}{Z + R/4}$ is a constant coefficient for each transmission line. It is important to point out that (2.56) and (2.57) have similar format and therefore the same hardware module as shown in Figure 2-15 can be used for calculation. The hardware blocks that can be shared are highlighted by dashed lines. For example, one of the two computation paths shown in Figure 2-15 can be used for calculating the line currents of sending end while the other path is

calculating the receiving end line current. In each simulation time-step, the calculation of history currents is the first step whereas the line currents are calculated after the solution of system nodal voltages. This means that these two calculations are decoupled and can share the same hardware module without losing any calculation efficiency, while maximizing the utilization of hardware resources.

2.2.3 Passive Elements

The basic passive elements in the power systems are resistor, capacitor and inductor, which are the most common components in power systems. This section describes the model formulation and hardware implementation for each of these elements. In addition, the modelling and hardware implementation of series *RL* branch is also included. The treatment of series *RL* branch as a single element is beneficial for the simulation as it reduces the number of nodes in the network.

2.2.3.1 Model Formulation

Resistance

Figure 2-16 Resistor branch

The continuous-time model for a resistor branch with resistance *R* as shown in Figure 2-16 is written as:

$$v_1(t) - v_2(t) = v_{12}(t) = i_1(t) \times R \tag{2.58}$$

A direct relationship between the voltage across and current through the branch is obtained from(2.58).

Capacitance

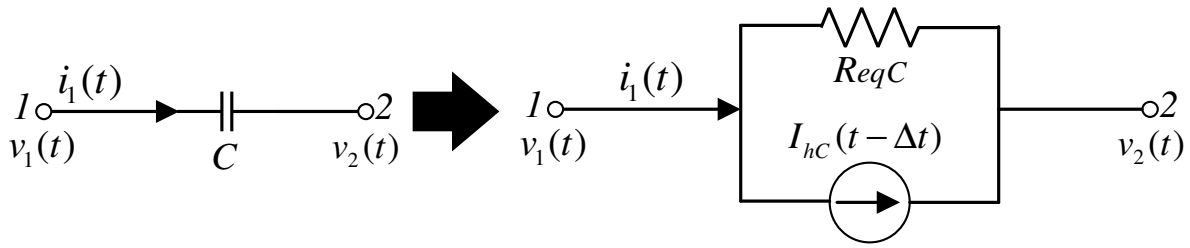


Figure 2-17 Capacitor branch

The continuous-time model for a capacitor branch as shown in Figure 2-17 is:

$$i_1(t) = C\frac{dv_{12}(t)}{dt} \tag{2.59}$$

Apply trapezoidal rule of integration from time *(t-Δt)* to *t* to equation (2.59) gives

$$i_1(t) = \frac{2C}{\Delta t}v_{12}(t) + I_{hL}(t-\Delta t) = \frac{v_{12}(t)}{R_{eqC}} + I_{hC}(t-\Delta t) \tag{2.60}$$

Where the history term is calculated as

$$I_{hC}(t-\Delta t) = -i_1(t-\Delta t) - \frac{2C}{\Delta t}v_{12}(t-\Delta t) \tag{2.61}$$

From (2.60) it can be observed that in discrete-time domain, the capacitor branch can be represented as an equivalent resistance R_{eqC} and a history current source connected in parallel (Figure 2-17). At each time-step, the history current is known from the solutions in previous time-step.

Inductance

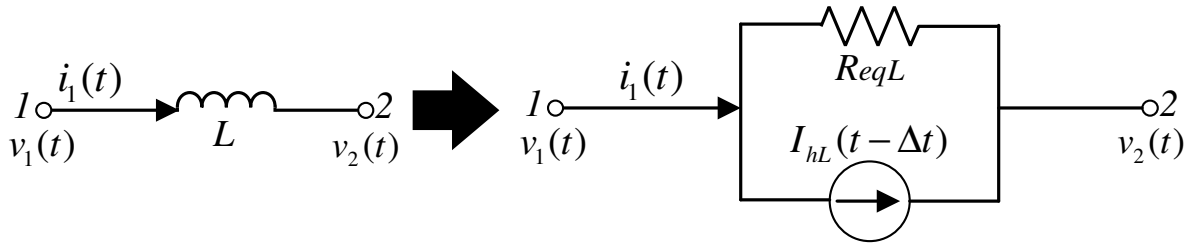


Figure 2-18 Inductor branch

The continuous-time model for an inductor branch as shown in Figure 2-18 is:

$$v_{12}(t) = L\frac{di_1(t)}{dt} \tag{2.62}$$

Apply trapezoidal rule of integration from time (t-Δt) to t to equation (2.62) gives:

$$i_1(t) = \frac{\Delta t}{2L}v_{12}(t) + I_{hL}(t-\Delta t) = \frac{v_{12}(t)}{R_{eqL}} + I_{hL}(t-\Delta t) \tag{2.63}$$

where the history current is given as

$$I_{hL}(t-\Delta t) = i_1(t-\Delta t) + \frac{\Delta t}{2L}v_{12}(t-\Delta t) \tag{2.64}$$

Similar to the capacitor branch, in discrete-time domain, the inductor branch can be represented as an equivalent resistance R_{eqL} and a history current source connected in parallel (Figure 2-18). The history current can be calculated using the solutions from previous time-step.

Series RL Branch

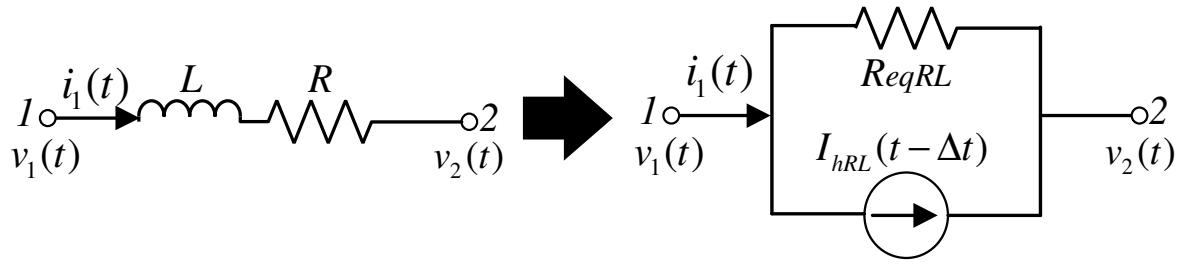


Figure 2-19 Series *RL* branch

As previously discussed, the treatment of a series *RL* branch as one single branch reduces the number of nodes in the system, therefore simplifies the computation. The continuous-time model for a series *RL* branch as shown in Figure 2-19 is:

$$v_{12}(t) = Ri_1(t) + L\frac{di_1(t)}{dt} \tag{2.65}$$

Apply trapezoidal rule of integration from time (t-Δt) to t to equation (2.65) gives:

$$i_1(t) = \frac{1}{R+\frac{2L}{\Delta t}} v_{12}(t) + I_{hRL}(t-\Delta t) = \frac{v_{12}(t)}{R_{eqRL}} + I_{hRL}(t-\Delta t) \tag{2.66}$$

where the history current is given as:

$$I_{hRL}(t-\Delta t) = \frac{1}{R+\frac{2L}{\Delta t}}\left[v_{12}(t-\Delta) - \left(R - \frac{2L}{\Delta t} \right) i_1(t-\Delta) \right] \tag{2.67}$$

From (2.66) it can be observed that, similar to the inductor and capacitor branch, the series *RL* branch can also be represented as an equivalent resistor and a history current source in

discrete-time domain. The history current is again calculated using the solutions from previous time-step.

From the above derivations it can be seen that that the passive elements can all be represented as a constant resistor connected with a history current source. It is of significance because in this way the system nodal equations can be easily obtained by forming the system admittance matrix using equivalent resistors, and nodal current injections using current sources.

2.2.3.2 Hardware Implementation

The above four types of passive power system elements have similar forms of formulation thus can be calculated using the same hardware module. The calculations related to this hardware module mainly consist of two parts: the first part is the calculation of the history current $I_h(t-\Delta t)$ and second part is the calculation of the branch current $i_1(t)$.

For resistor, capacitor, inductor and series *RL* branch, the general formula of their history current $I_h(t-\Delta t)$ can be expressed as:

$$I_h(t-\Delta t) = K_1 \times i_1(t-\Delta t) + K_2 \times (v_1(t-\Delta t) - v_2(t-\Delta t)) \tag{2.68}$$

For branch current $i_1(t)$, the general formula is:

$$i_1(t) = I_{hPE}(t-\Delta t) + \frac{1}{R_{eq}} \times v_{12}(t) \tag{2.69}$$

The coefficients of K_1 and K_2 are functions of component values and the size of simulation time-step, and R_{eq} is the equivalent resistance of the branch. For resistor R:

$$\mathrm{R}_{eq} = R; K_1 = 0; K_2 = 0; \tag{2.70}$$

For capacitor C:

$$\mathrm{R}_{eq} = \frac{\Delta t}{2C}; K_1 = -1; K_2 = -\frac{2}{\mathrm{R}_{eq}}; \tag{2.71}$$

For inductor L:

$$\mathrm{R}_{eq} = \frac{2L}{\Delta t}; K_1 = 1; K_2 = \frac{2}{\mathrm{R}_{eq}}; \tag{2.72}$$

For series *RL* branch:

$$\mathrm{R}_{eq} = R + \frac{\Delta t}{2L}; K_1 = -\frac{R - \frac{2L}{\Delta t}}{R + \frac{2L}{\Delta t}}; K_2 = \frac{1}{R + \frac{2L}{\Delta t}}; \tag{2.73}$$

As these values are fixed so they are calculated before the start of simulation and stored in read-only ROMs in advance.

As the calculation of history current $I_h(t-\Delta t)$ and branch current $i_1(t)$ are in different steps of simulation process the same computation path can be shared between them:

$$y = a \times b + c \times (d - e) \tag{2.74}$$

In terms of FPGA hardware implementation, there are several schemes to choose from. The actual choice is a trade-off between the hardware resource usage and the simulation speed.

If the goal is faster speed and shorter simulation time-step, then more parallel hardware modules should be implemented. With this design, each module is responsible for the calculation of a limited number of components therefore the required calculation time is reduced. In the most extreme case, each hardware module is responsible for the calculation of

only one passive element. The drawback of this way of implementation is that hardware resources are significantly increased.

If the requirement for simulation time-step is not very high, using less or single hardware module and with pipelined algorithm is a good choice. In a pipeline, all the inputs go one after another without waiting for the last calculation to complete and the results are obtained at every clock cycle. This time-overlapping calculation maximizes hardware efficiency.

In this thesis, the deeply pipelined calculations for passive element module are implemented. The design of hardware module is shown as Figure 2-20.

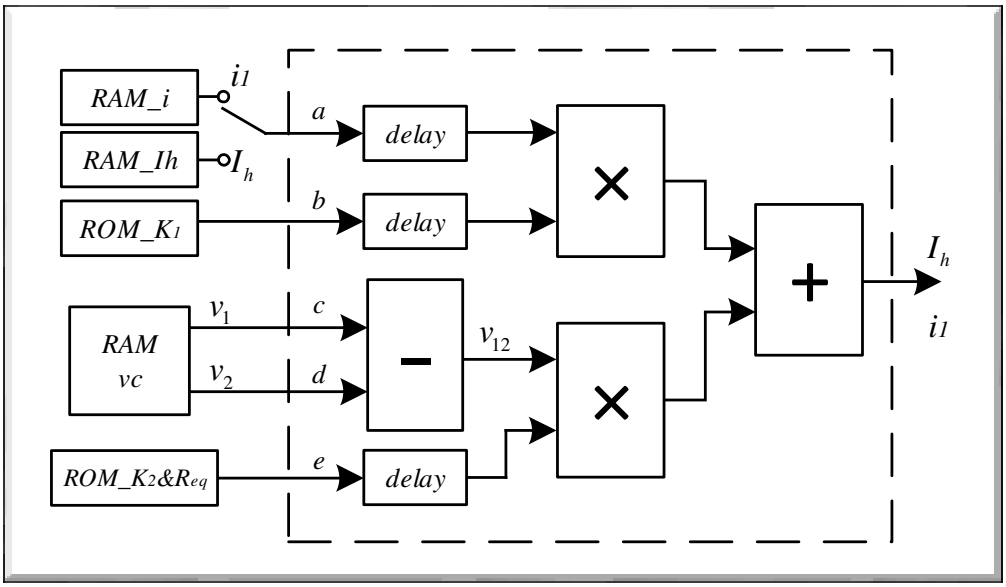


Figure 2-20 Passive elements module

The algorithm of this module is shown below:

1. Inputs *b* and *e* are read out from ROMs; inputs *a, c* and *d* are read out from RAMs.
2. The value of *c* minus *d* is calculated while other inputs are sent to the delay registers to wait until the calculation of *c* minus *d* is completed.
3. Two multiplications $a \times b$ and $v_{12} \times e$ are calculated at the same time.

4. The two multiplication results from step 3 are added together to obtain the final output.

As mentioned earlier, this module can be shared by both history current $I_h(t-\Delta t)$ and branch current $i_1(t)$ calculation. When calculating history current, input a is connected to the output of RAM_i, which stores $i_1(t)$ from last time-step. Also the output is directed and stored into RAM_Ih. When calculating branch currents, input a is connected to the output of RAM_Ih, which stores history currents, input b is held at 1 and the results are directed and stored into RAM_i.

In this module, it takes 1 clock cycle to read data from RAM or ROM, 4 clock cycles to perform floating point addition and subtraction and 3 clock cycles to do multiplication. In total the module latency is 12 clock cycles. If number of passive elements allocated for one module is N, the total processing time will be $12 + N - 1$ clock cycles when using the proposed calculation method.

2.2.4 Voltage and Current Sources

2.2.4.1 Model Formulation

Two kinds of sources are considered: 1) ideal voltage source and 2) ideal current source. In continuous-time domain the ideal voltage source and ideal current source are expressed as:

$$v(t) = V_{mag} \sin(\omega t + \varphi_0) \tag{2.75}$$

$$i(t) = I_{mag} \sin(\omega t + \varphi_0) \tag{2.76}$$

Where V_{mag} and I_{mag} are magnitudes of voltage and current, ω is the angular frequency, and φ_0 is the initial phase angle. In discrete-time domain, (2.75) and (2.76)are expressed as:

$$v(t) = V_{mag} \sin(\omega n \Delta t + \varphi_0) \tag{2.77}$$

$$i(t) = I_{mag} \sin(\omega n \Delta t + \varphi_0) \tag{2.78}$$

where n is the length of Look-Up-Table (LUT) that is used to store the function values. The value of n is selected to achieve acceptable accuracy while saving memory space. The detailed hardware implementation is explained in the next sections.

2.2.4.2 Hardware Implementation

The source module is also implemented as LUT, which is also adopted in the Park transformation module.

The 4096 data of half cycle of sine function is calculated and stored in LUT in advance. The initial phase angle φ_0 is reset and stored in RAM during initialization. $\Delta\varphi$ is the incremental change of phase angle for every time-step. When system frequency f is 60Hz and simulation time-step Δt is 50us, $\Delta\varphi$ is calculated as:

$$\Delta\varphi = \omega\Delta t = 2 \times 180^\circ \times f \times \Delta t = 1.08^\circ \tag{2.79}$$

The phase angles of source are updated every time-step by $\Delta\varphi$ and LUT addresses are calculated from the updated phase angles. This avoids the complexity in calculating the values of non-linear sinusoidal functions.

The hardware implementation of the source module is shown in Figure 2-21. It can be seen from the figure that there are two computation paths in this module.

The algorithm for the first path is explained as follows:

1. Calculate the address using phase angle information.
2. Retrieve the value of sinusoidal function from LUT using the address from step 1.
3. Floating-point multiplication is used to multiply the sinusoidal value from step 2 and the pre-stored value of source magnitude to get the source output.

The second path, in parallel with the first path, is responsible for updating the phase angle that will be used by the first path in the next time-step.

1. The phase angle is summed with Δφ and then compared with 360 degrees.
2. It will be reduced by 360 degrees if the value of phase angle is larger than 360 degrees. This is to make sure the phase angle is always between 0-360 degrees.

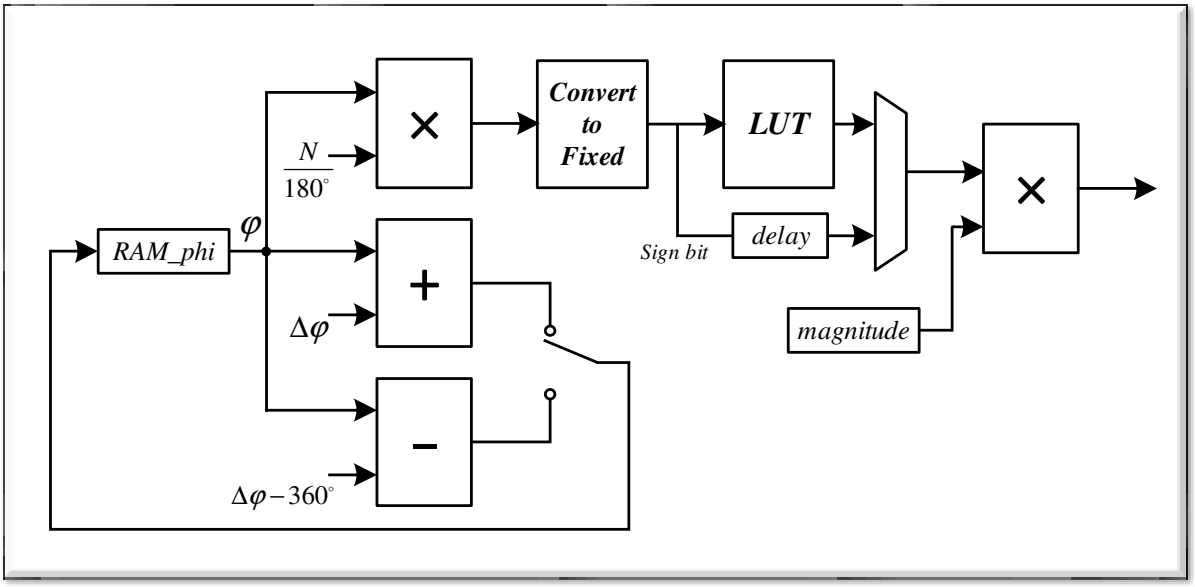


Figure 2-21 Source module

2.2.5 Circuit Breakers

2.2.5.1 Model Formulation

Circuit breaker component is used to simulate various kinds of fault, the protection of a transmission line and the switching of shunt-connected devices. The modelling of circuit breaker in discrete-time domain is achieved by the use of equivalent resistors. When the circuit breaker is closed, it is modelled as a small resistor (10^{-6} Ohm in this thesis). When the circuit breaker is open, it is modelled as a large resistor (10^{6} Ohm in this thesis). This method of modelling is illustrated in Figure 2-22.

The action of circuit breaker causes a change of network admittance matrix. To improve simulation speed and minimize hardware resource requirement, the possible changes of system admittance matrix are pre-stored in RAM. The damping adjustment scheme (CDA) [149] can be implemented to avoid the potential numerical instabilities caused by switching of circuit breakers

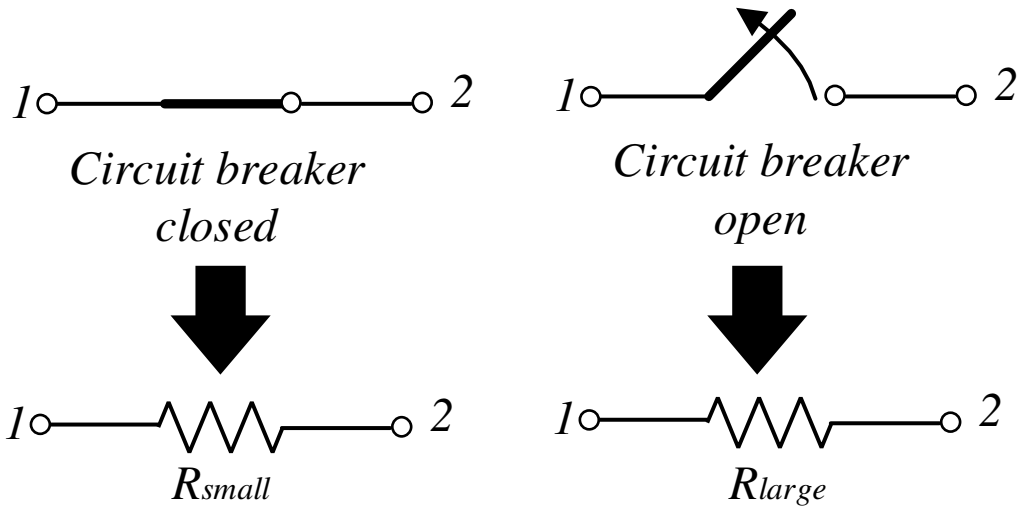


Figure 2-22 Circuit breaker model

2.2.5.2 Hardware Implementation

The fault changes the system topology and this directly leads to the change of network admittance matrix Y of network solver, and system equivalent resistance R_{eq_sys} for the solution of synchronous generator. In the proposed FPGA-based EMT simulator, the system matrix Y and system equivalent resistance of R_{eq_sys} for different fault scenarios are pre-calculated and stored in ROMs. A top control module chooses the correct matrix according to the fault to be applied.

2.2.6 Loads

For stability and power flow analysis, the power system loads are commonly modelled as a composition of constant power, constant current and constant impedance loads (also known as ZIP loads) [147]. The frequency dependency of load characteristics can be taken into account by multiplying a factor related to the frequency deviation to the ZIP load model [147]. However for EMT studies, as stated in [148] and [43], the lumped RLC branch model is used to represent the power system loads, and this way of representation is adopted in case studies of this thesis.

2.3 Power System Controllers

To model control systems in EMT simulations, the following aspects need to be considered:

- Modelling of transfer functions. Transfer function is used in all kinds of control systems, so a correct modelling of it is vitally important for the accuracy of EMTP.
- Handling of limiters. Limiters exist in most of the control systems and they introduce non-linearity into the system. An accurate way of modelling limiters is also important for the accuracy of EMTP.

- Interface with the electrical network.

These three aspects are discussed in following sections.

Transfer Functions

Figure 2-23 shows a generic transfer function block that is normally used in power system controllers.

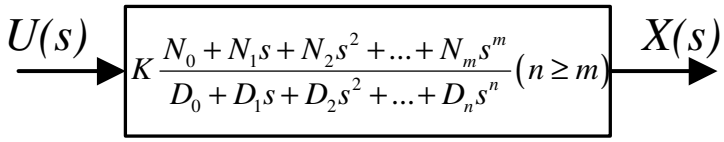


Figure 2-23 Generic transfer function block

By introducing internal variables for derivatives of *u* and *x*,

$$x_1 = \frac{dx}{dt} \quad x_2 = \frac{dx_1}{dt} \quad \ldots \quad x_n = \frac{dx_{n-1}}{dt} \tag{2.80}$$

$$u_1 = \frac{du}{dt} \quad u_2 = \frac{du_1}{dt} \quad \ldots \quad u_m = \frac{dx_{m-1}}{dt} \tag{2.81}$$

the transfer function in time domain becomes an algebraic equation,

$$D_0x(t)+D_1x_1(t)+...+D_nx_n(t) = K\left(N_0u(t)+N_1u_1(t)+...+N_mu_m(t)\right) \tag{2.82}$$

To convert it into discrete-time domain, trapezoidal rule of integration is applied to (2.80) and (2.81):

$$x_i(t) = \frac{2}{\Delta t}x_{i-1}(t) - \left[x_i(t-\Delta t) + \frac{2}{\Delta t}x_{i-1}(t-\Delta t) \right] \tag{2.83}$$

$$u_j(t) = \frac{2}{\Delta t} u_{j-1}(t) - \left[u_j(t-\Delta t) + \frac{2}{\Delta t} u_{j-1}(t-\Delta t) \right] \tag{2.84}$$

where $i=1,2 \ldots n$ and $j=1,2 \ldots m$. Substitute (2.83) and (2.84) back into (2.82), a single input-output relationship between u and x can be obtained:

$$cx(t) = Kdu(t) + hist(t-\Delta t) \tag{2.85}$$

Where

$$c_i = c_{i-1} + (-2)^i \left[\binom{i}{i} \left(\frac{2}{\Delta t} \right)^i D_i + \binom{i+1}{i} \left(\frac{2}{\Delta t} \right)^{i+1} D_{i+1} + \ldots + \binom{n}{i} \left(\frac{2}{\Delta t} \right)^n D_n \right] \tag{2.86}$$

$$d_i = d_{i-1} + (-2)^i \left[\binom{i}{i} \left(\frac{2}{\Delta t} \right)^i N_i + \binom{i+1}{i} \left(\frac{2}{\Delta t} \right)^{i+1} N_{i+1} + \ldots + \binom{n}{i} \left(\frac{2}{\Delta t} \right)^m N_m \right] \tag{2.87}$$

$$hist_i(t) = Kd_i u(t) - c_i x(t) - hist_i(t-\Delta t) + hist_i(t-\Delta) \tag{2.88}$$

With

$$c = c_0 = \sum_{i=0}^{n} \left(\frac{2}{\Delta t} \right)^i D_i \tag{2.89}$$

$$d = d_0 = \sum_{i=0}^{m} \left(\frac{2}{\Delta t} \right)^i N_i \tag{2.90}$$

$\binom{n}{k}$ represents the binomial coefficient, i.e.,

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{2.91}$$

Equation (2.85) directly relates the input and output at time *t* and therefore can be used for the implementation on FPGA. At each time-step, with the known history term and control input, the output of *x* is calculated. Then *n* history terms are updated for the solution in next time-step.

Non-windup Limiters

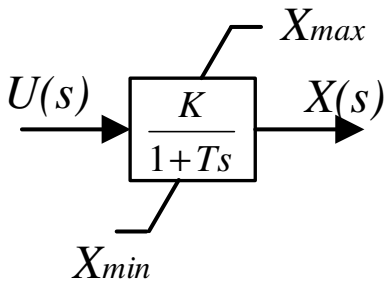


Figure 2-24 Non-windup limiter

Figure 2-24 shows a first-order transfer function with non-windup limiter. The system equation and limiting actions in continuous-time domain are:

$$x(t)+T\frac{dx(t)}{dt}=Ku(t) \quad if \quad x_{\min}<x(t)<x_{\max} \tag{2.92}$$

$$x(t)=x_{\min} \quad if \quad x(t)\le x_{\min} \quad and \quad Ku(t)-x(t)<0 \tag{2.93}$$

$$x(t)=x_{\max} \quad if \quad x(t)\ge x_{\max} \quad and \quad Ku(t)-x(t)>0 \tag{2.94}$$

It is converted into discrete-time domain by applying trapezoidal rule of integration:

$$
\begin{aligned}
&cx(t) = du(t) + hist(t-\Delta t) \quad when \quad x_{\min} < x < x_{\max} \\
&with \quad hist(t-\Delta t) = Ku(t-\Delta t) - \left(1 - \frac{2T}{\Delta t} x(t-\Delta t)\right) \\
&c = \left(1 + \frac{2T}{\Delta t}\right) \\
&d = K
\end{aligned}
\tag{2.95}
$$

$$x(t) = x_{\min} \quad if \quad x(t) \le x_{\min} \quad and \quad Ku(t) - x(t) < 0 \tag{2.96}$$

$$x(t) = x_{\max} \quad if \quad x(t) \ge x_{\min} \quad and \quad Ku(t) - x(t) > 0 \tag{2.97}$$

From (2.95) to (2.97) it can be seen that with non-windup limits, the output variables is limited. It comes off the limit as soon as the input *u* changes its sign.

Windup Limiters

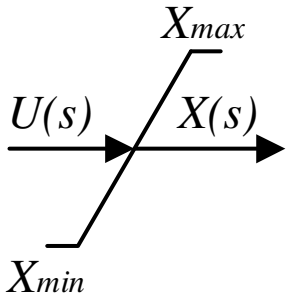


Figure 2-25 Windup limiter

Figure 2-25 shows a windup limiter. In continuous-time domain, it is expressed as

$$x(t) = \begin{cases} u(t) & if \quad x_{\min} < u(t) < x_{\max} \\ x_{\min} & if \quad u(t) \le x_{\min} \\ x_{\max} & if \quad u(t) \ge x_{\max} \end{cases} \tag{2.98}$$

Since no integrations or differentiations are included in the limiter, the discrete-time domain representation is the same as that in the continuous time domain. Different from the non-

windup limiter, the output *x(t)* of wind-up limiter does not come off the limit until input *u(t)* is within the limit.

Interface with Electrical Network

From previous sections it can be seen that the control systems cannot be represented as equivalent resistance matrices with parallel history current sources. So they cannot be readily fit into the nodal network equations.

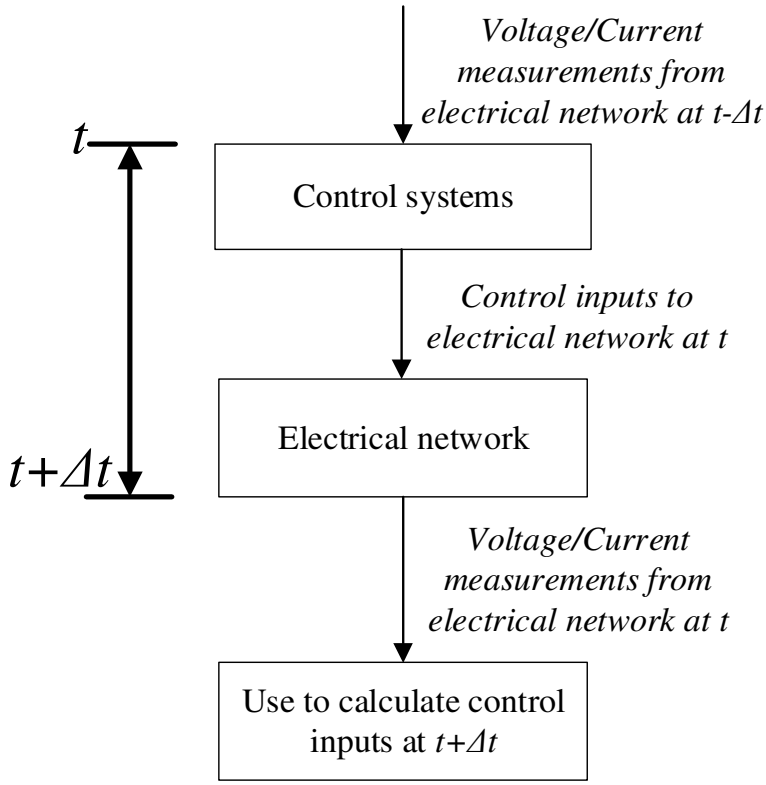


Figure 2-26 Interface between control system and electrical network

The common solution in real-time simulation is to solve the electrical network and control systems separately. Electrical network is solved from *t* to *t+Δt* using the output from control systems computed at *t-Δt* to *t*. Once the electrical network is solved, the solutions are fed back to control systems to get all the control inputs to electrical network for next time-step (from *t* to *t+Δt*). As a consequence, electrical network is always using the control inputs from

control systems computed one time-step earlier. The flow chart illustrating the interface process is shown in Figure 2-26.

The delay of one time-step between the control system and the electrical network can be mitigated by predicting the output from control system at time *t* rather than calculate it using measurements from previous time-step. With the predicted value, the electrical network is solved and then the control system output can be calculated. The difference between this calculated output of control system and the predicted value can be minimised through iterations. However the iteration-type of solution method is not utilized in this thesis to minimize the complexity of implementation and the required hardware resources. This is because the time-step normally required by real-time simulation is very small (up to 50 μs) and the error caused by the non-iterative method as illustrated in Figure 2-26 is small. This non-iterative method is also adopted and verified by many widely-used simulators such as RTDS.

2.3.1 Excitation System

2.3.1.1 Model Formulation

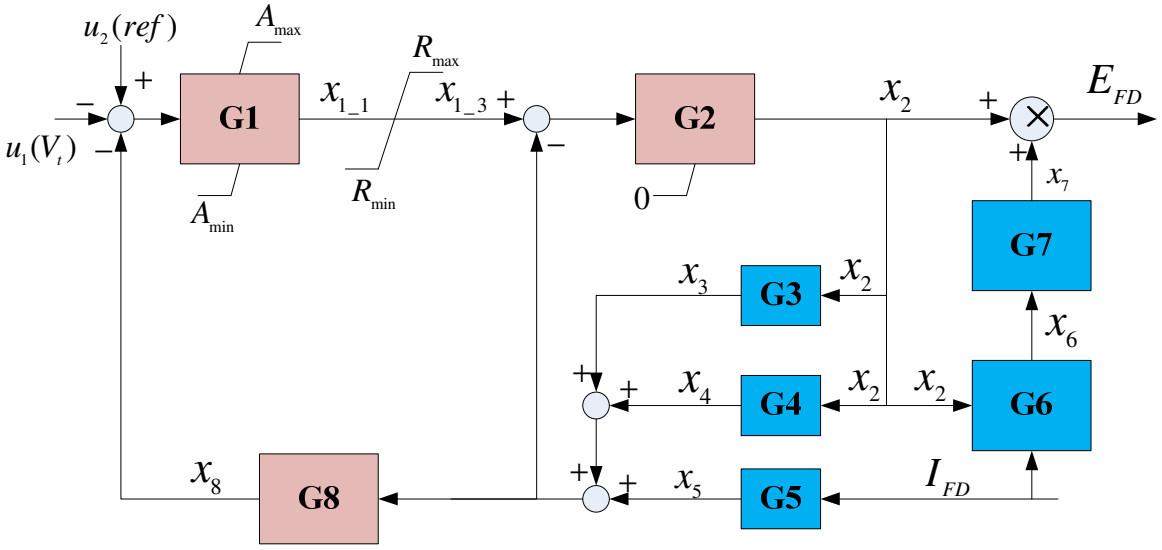


Figure 2-27 Block diagram representation of AC1A exciter block

As an example, the formulation of IEEE Type AC1A excitation system [150] is explained in detail. The block diagram representation of the exciter is shown in Figure 2-27, where G1 to G8 represent the transfer functions (the same transfer functions are used as that in [150]), u_1 is the measured terminal voltage of synchronous machine, u_2 is the reference terminal voltage, I_{FD} is the measured field current, and E_{FD} is the excitation voltage. x_1 to x_8 are internal variables.

Discretize all the transfer functions using the equations of (2.85) and (2.95)-(2.98), the expression for all internal variables and control outputs are obtained as follows:

$$x_{1\_1}(t)=\begin{cases}\dfrac{1}{c_1}[K_1(u_2(t)-u_1(t-\Delta t)-x_8(t))+hist_1] & if\ \ A_{\min}\le x_{1\_1}(t)\le A_{\max}\\ A_{\min} \quad if\ \ x_{1\_1}\le A_{\min}\ \&\ K_1(u_2(t)-u_1(t-\Delta t)-x_8(t))-x_{1\_1}(t)\le 0\\ A_{\max} \quad if\ \ x_{1\_1}\ge A_{\max}\ \&\ K_1(u_2(t)-u_1(t-\Delta t)-x_8(t))-x_{1\_1}(t)\ge 0\end{cases} \tag{2.99}$$

$$x_{1\_3}(t)=\begin{cases}x_{1\_1}(t) & if \quad R_{\min}<x_{1\_1}(t)<R_{\max}\\ R_{\min} & if \quad x_{1\_1}(t)\le R_{\min}\\ R_{\max} & if \quad x_{1\_1}(t)\ge R_{\max}\end{cases} \tag{2.100}$$

$$x_2(t)=\begin{cases}\dfrac{1}{c_2}[K_2(x_{1\_3}(t)-x_3(t)-x_4(t)-x_5(t))+hist_2] & if\ \ x_2(t)\ge 0\\ 0 \quad if\ \ x_2(t)\le 0\ \&\ K_2(x_{1\_3}(t)-x_3(t)-x_4(t)-x_5(t))+hist_2\end{cases} \tag{2.101}$$

$$x_3(t)=(ax_2(t-\Delta t)+b)x_2(t-\Delta t) \tag{2.102}$$

$$x_4(t)=K_e x_2(t-\Delta t) \tag{2.103}$$

$$x_5(t)=K_d I_{FD}(t-\Delta t) \tag{2.104}$$

$$x_6(t) = \frac{I_{FD}(t-\Delta t)K_c}{x_2(t-\Delta t)} \tag{2.105}$$

$$x_7(t) = f(x_6(t)) \tag{2.106}$$

$$c_8 x_8(t) = K_8(x_3(t) + x_4(t) + x_5(t)) + hist_8 \tag{2.107}$$

$$I_{FD}(t-\Delta t) = i_{fd}(t-\Delta t)L_{adu} \tag{2.108}$$

where

$$\begin{aligned} &c_1 = 1 + \frac{2T_A}{\Delta t} \quad K_1 = K_A = 400 \quad T_A = 0.02 \\ &hist_1 = K_A[u_2(t-\Delta t) - u_1(t-\Delta t) - x_8(t-\Delta t)] - (1 - \frac{2T_A}{\Delta t})x_{1\_1}(t-\Delta t) \end{aligned} \tag{2.109}$$

$$\begin{aligned} &c_2 = \frac{2T_e}{\Delta t} \quad K_2 = 1 \quad T_e = 0.8 \\ &hist_2 = x_{1\_3}(t-\Delta t) - x_3(t-\Delta t) - x_4(t-\Delta t) - x_5(t-\Delta t) + \frac{2T_e}{\Delta t}x_2(t-\Delta t) \end{aligned} \tag{2.110}$$

$$\begin{aligned} &a = 0.0673 \quad b = -0.1814 \quad when \quad x_2(t) \geq 3.14 \\ &a = \frac{3}{314} \qquad b = 0 \qquad when \quad 0 \leq x_2(t) \leq 3.14 \end{aligned} \tag{2.111}$$

$$K_e = 1 \quad K_d = 0.38 \quad K_c = 0.2 \tag{2.112}$$

$$f(x_6) = \begin{cases} 1 - 0.577x_6(t) & if \ \ x_6(t) \leq 0.433 \\ \sqrt{0.75 - x_6(t)^2} & if \ \ 0.433 \leq x_6(t) \leq 0.75 \\ 1.732(1 - x_6(t)) & if \ \ 0.75 \leq x_6(t) \leq 1 \\ 0 \quad if \ \ x_6(t) \geq 1 \end{cases} \tag{2.113}$$

$$c_8 = (1+\frac{2T_f}{\Delta t}) \quad K_8 = \frac{2K_f}{\Delta t} \quad T_f = 1 \quad K_f = 0.03$$
$$hist_8 = -\frac{2K_f}{\Delta t}[x_3(t-\Delta t) + x_4(t-\Delta t) + x_5(t-\Delta t)] + (\frac{2T_f}{\Delta t} - 1)x_8(t-\Delta t) \tag{2.114}$$

i_{fd} is the per unit excitation current in synchronous machine per unit system, L_{adu} is the d-axis equivalent inductor. Since E_{FD} calculated in Figure 2-27 is under the per unit system for excitation system, conversion of per unit system is required:

$$e_{fd}(t) = E_{FD}(t) \times R_{fd} \div L_{adu} \tag{2.115}$$

where e_{fd} is the per unit excitation voltage under machine per unit systems, R_{fd} is the per unit field winding resistance. Equation (2.95) to (2.107) and (2.114) are used to calculate the excitation voltage for synchronous machines.

Solution Steps of Excitation system

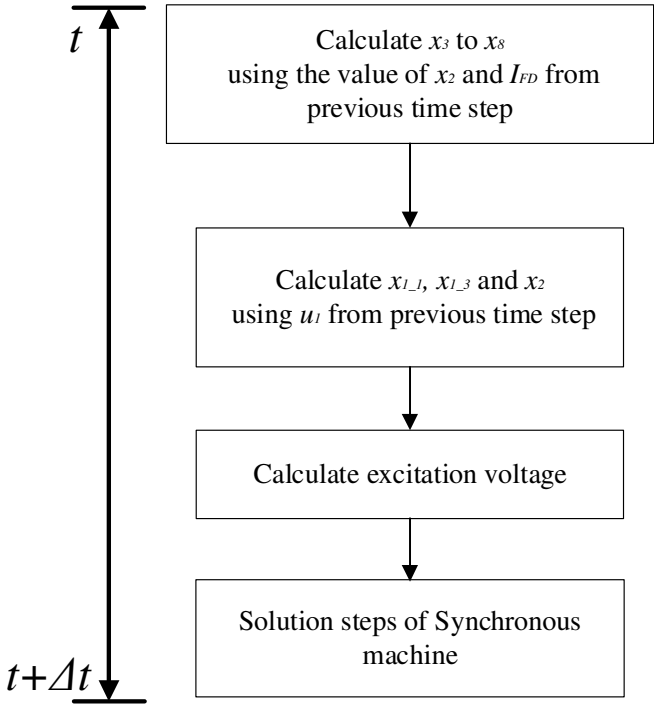


Figure 2-28 Solution steps for excitation system

The flow chart illustrating the solution steps for excitation system is shown in Figure 2-28. Firstly with known values of x_2 and field winding current I_{FD} from previous time-step, the internal variables x_3 to x_8 are calculated using (2.102)-(2.107). Secondly the values of x_{1_1}, x_{1_3} and x_2 are calculated using the measured machine terminal voltage from previous time-step, i.e., $u_1(t-\Delta t)$. The calculated value of x_2 will be used in the next time-step. Thirdly, equation (2.115) is used to get the excitation voltage for the solution of synchronous machine.

2.3.1.2 Hardware Implementation

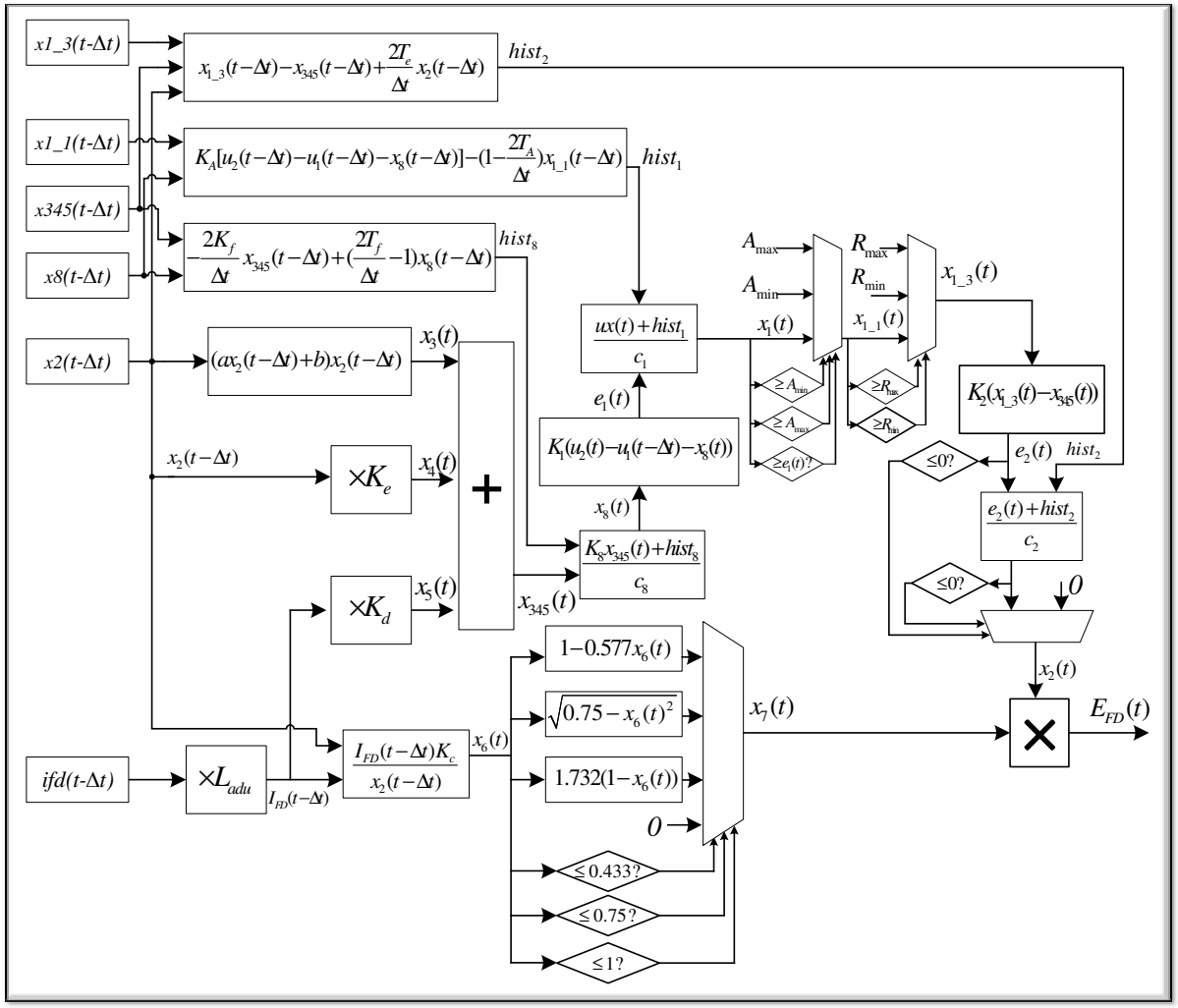


Figure 2-29 Excitation module

The hardware module for excitation system is shown in Figure 2-29. It has two inputs: one is the terminal voltage of synchronous machine from last time-step; the other is the field excitation current from this time-step. The output of this module is the field excitation voltage. The algorithm of Figure 2-29 for excitation system is shown as follows:

1. The values of x_3, x_4, x_5, x_6 are calculated using the values of x_2 and I_{FD} from last time-step. The calculations of $hist_1$, $hist_2$, $hist_8$ are begun at the same time using the values of variables from last time-step.

2. When x_3, x_4 and x_5 are calculated, x_{345} is calculated by adding them together.

3. The results of $hist_1$, $hist_2$, $hist_8$ are obtained. Calculate x_7 using the value of x_6 from step 1 and x_8 using the value of x_{345} from step 2.

4. The values of x_1, x_{1_1}, x_{1_3} are calculated in sequence. The result of x_7 from step 3 is sent to the delay register.

5. The value of x_2 is calculated using the values of $hist_2$ from step 3 and x_{1_3} from step 4.

6. The excitation voltage of E_{fd} is calculated using the value of x_2 from step 1 and the value of x_7 from step 3.

Some parts of the calculations are used multiple times in the computation of excitation module, so dedicated computation blocks and memory space have been designed for them to avoid duplicate computations. These calculations includeFigure 2-29:

$$x_{345}(t) = x_3(t) + x_4(t) + x_5(t) \tag{2.116}$$

$$e_1(t) = K_1(u_2(t) - u_1(t - \Delta t) - x_8(t)) \tag{2.117}$$

$$e_2(t) = K_2(x_{1\_3}(t) - x_{345}(t)) \tag{2.118}$$

Also, dedicated RAMs have been designed to store the values of x_{1_1}, x_{1_3}, x_2, x_3, x_4, x_5, x_8 from previous time-step. This increases the efficiency of calculation when these values are required at the current simulation time-step.

There are four limiters in the excitation module. The limiters are implemented using floating-point comparator and multiplexer. The calculation of $x_7=f(x_6)$ is used as an example to illustrate the detailed implementation of limiters. As shown in (2.113), this function has four output expressions depending on the input condition. The results for four outputs and the value of x_6 are calculated and sent to the multiplexer and the comparator, respectively (Figure 2-29). The following conditions are evaluated to determine the final output of this function:

- The output of the first comparator $c1$=1 if x_6 <= 0.433; The output of the second comparator $c2$ = 1 if $x6$ <= 0.0.75; The output of the first comparator $c3$ = 1 if $x6$ <= 0.1;
- If the output of first comparator c_1 is 1, no matter what are the values of c_2 and c_3, the multiplexer will select and forward the value of first input to the output;
- If $c1$ = 0 and $c2$ = 1, the multiplexer will select and forward the second input to the output;
- If $c1$ = 0,$c2$ = 0 and $c3$ = 1, the third input of multiplexer will be selected and forward to the output;
- If the values of c_1, c_2 and c_3 do not belong to any of the above cases, the fourth input "0" will be selected.

2.3.2 Governor/Turbine System

2.3.2.1 Model Formulation

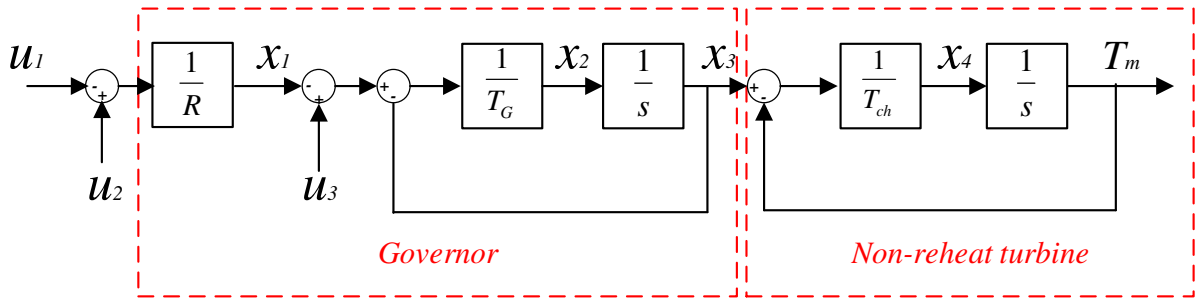


Figure 2-30 Governor/turbine systems

Table 2.2 Parameters for Governor/Turbine System

R	T_G	T_{ch}
20 (5% droop)	0.2s	0.3s

Figure 2-30 shows an example of governor/turbine system. In the figure, u_1 is the speed reference, u_2 is the measured rotor speed, u_3 is the load set point, x_1 to x_4 are internal variables and T_m represents the mechanical torque in per unit. *1/R* represents the percentage of droop, T_G is the time constant of governor and T_{ch} is the time constant of main inlet volumes and steam chest. The values for these parameters are listed in Table 2.2.

According to equation (2.85), the integrators can be discretized according to the trapezoidal rule of integration. The input-output relationship for each block can then be obtained as:

$$x_1(t) = \frac{1}{R} \times \left(u_2(t) - u_1(t)\right) \tag{2.119}$$

$$x_2(t) = \frac{1}{T_G} \times \left(u_3(t) - x_1(t) - x_3(t)\right) \tag{2.120}$$

$$\frac{2}{\Delta t} x_3(t) = x_2(t) + hist_3 \tag{2.121}$$

$$x_4(t) = \frac{1}{T_{ch}} \times \left(x_3(t) - T_m(t) \right) \tag{2.122}$$

$$\frac{2}{\Delta t} T_m(t) = x_4(t) + hist_5 \tag{2.123}$$

Where

$$hist_3 = x_2(t - \Delta t) + \frac{2}{\Delta t} x_3(t - \Delta t) \tag{2.124}$$

$$hist_5 = x_4(t - \Delta t) + \frac{2}{\Delta t} T_m(t - \Delta t) \tag{2.125}$$

Solution Steps of Governor/turbine System

The solution steps for governor/turbine system are illustrated in Figure 2-31.

Step 1: $x_1(t)$ is calculated using the estimated value of rotor speed, i.e, $u(t)$. Linear interpolation is used for the estimation.

Step 2: $x_2(t)$ is calculated using $x_3(t-\Delta t)$ from last time-step.

Step 3: $x_3(t)$ is calculated using $x_2(t)$ from Step 2.

Step 4: $T_m(t)$ is calculated using $x_3(t)$ from step 3 and $T_m(t-\Delta t)$ from last time-step.

The potential error caused by using values form previous time-step is minimised by the fact that the governor/turbine system has a much larger time constant than the simulation time-step. Alternatively a direct relationship between inputs and output can be obtained by

eliminating the internal variables in (2.119) to (2.123). However such method is only applicable to simple linear control systems and cannot be considered as a generic solution methodology for power system controllers.

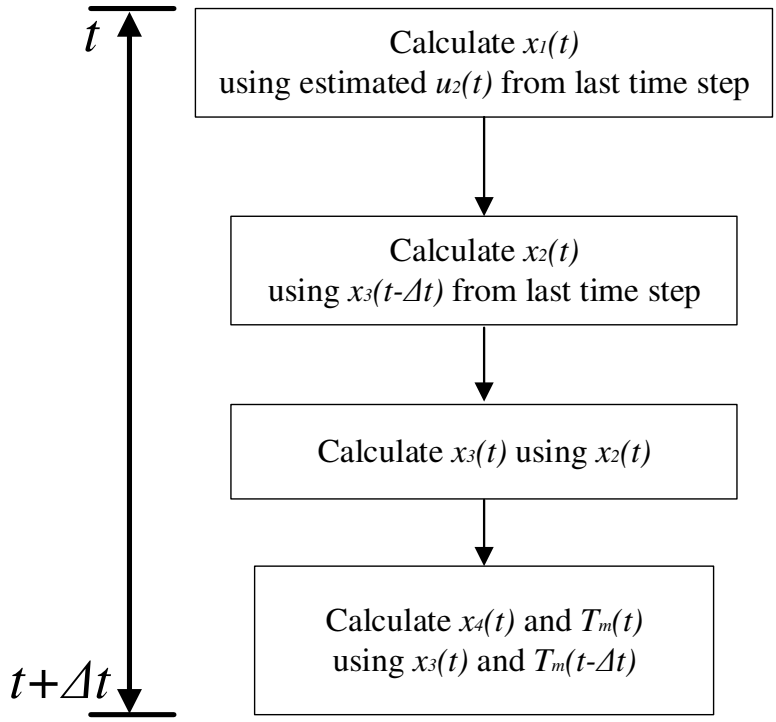


Figure 2-31 Solution steps for governor/turbine system

2.3.2.2 Hardware Implementation

For hardware implementation, equations (2.119) to (2.125) are re-arranged into the following five equations:

$$T_{mh}(i) = g_3(i-1) + KCH \times T_m(i-1) \tag{2.126}$$

$$g_1(i) = Kg1 \times \left(\omega_{ref} - \omega_p(i)\right) + loadpoint \tag{2.127}$$

$$g_{3h}(i) = g_1(i-1) + \left(KTG - 1\right) \times g_3(i-1) \tag{2.128}$$

$$g_3(i) = \left(1/\left(1+KTG\right)\right)\times\left(g_1(i)+g_{3h}(i)\right) \tag{2.129}$$

$$T_m(i) = \left(1/\left(1+KCT\right)\right)\times\left(g_3(i)+T_{mh}(i)\right) \tag{2.130}$$

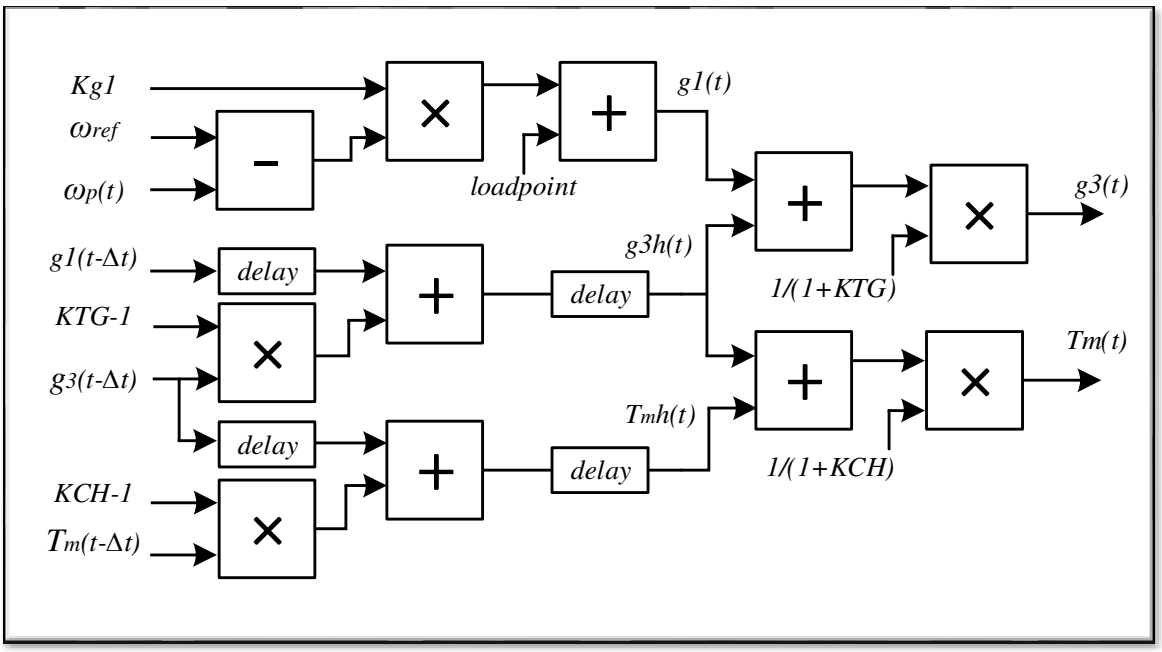


Figure 2-32 Governor module

The hardware implementation of governor module according to (2.126) - (2.130) is shown in Figure 2-32. It can be seen from the figure that g_1, g_{3h} and T_{mh} are firstly calculated in parallel and the results are used for the parallel computation of g_3 and T_m. The value of T_m is used as the mechanical torque input to the synchronous machine. Memory spaces are designed for the values of G_1, g_3 and T_m for the calculations in next time-step.

The detailed algorithm in each time-step of this module is shown as follows:

1. Constant parameters ω_{ref}, *Kg1*, *KTG* and *KCH*, variables ω_p from upstream module, and $g_1(t-\Delta t)$, $g_2(t-\Delta t)$ and T_m from last time-step are fed into the computation path.
2. The values of *g1*, *g3h* and *Tmh* are calculated.

3. The values of *g3* and T_m(t) are calculated and saved for the calculations in next time-step.

2.4 Model Verification

In this section, the models of power system elements are verified by comparing simulation results from FPGA with those from SIMULINK. System data and per unit base are given in Appendix A.

2.4.1 Case 1: Synchronous Machine with Constant Excitation and Governor Inputs

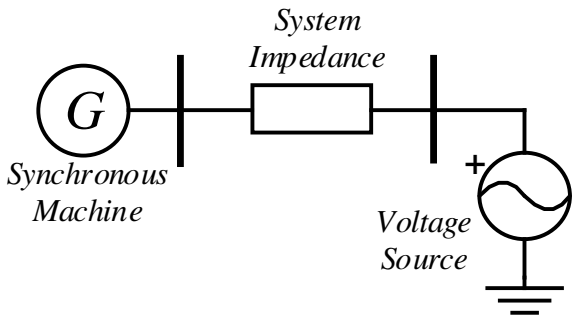
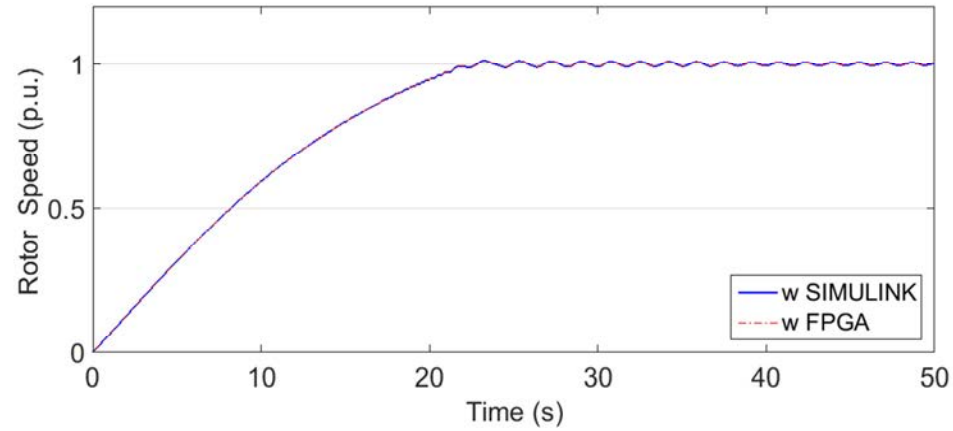


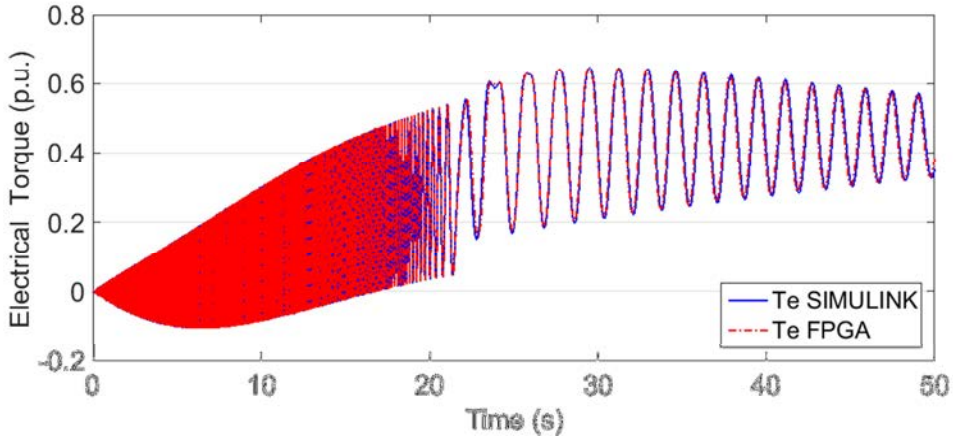
Figure 2-33 Verification of synchronous machine model

To verify the accuracy of synchronous machine model, the system shown in Figure 2-33 is simulated using both FPGA and SIMULINK. For the verification of the non-linear models, it is common to represent the non-linear model in detail while adopting simplified representation for the rest of the network [141, 151]. As shown in Figure 2-33, the system consists of a synchronous machine G and a voltage source. They are interconnected through system impedance. Constant excitation voltage and mechanical torque for the synchronous machine are used. The synchronous machine model described in Section 2.2.1, the passive

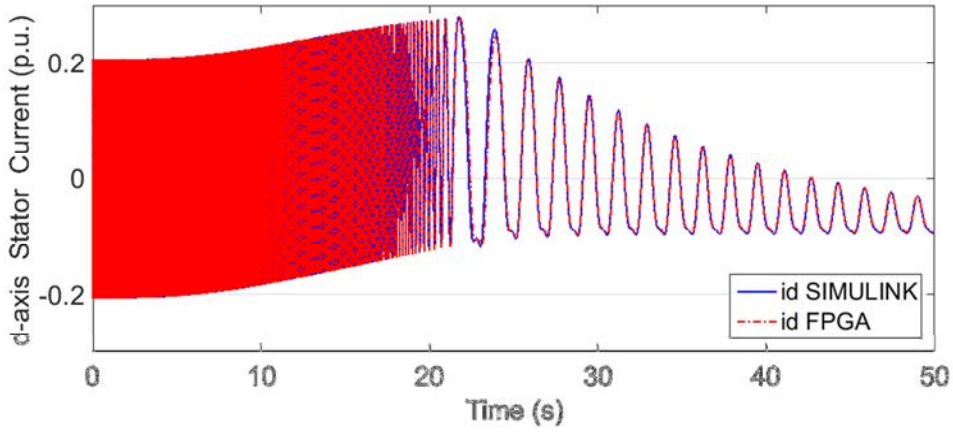
elements models described in Section 2.2.3 and the voltage source model described in Section 2.2.4 are used in this case.



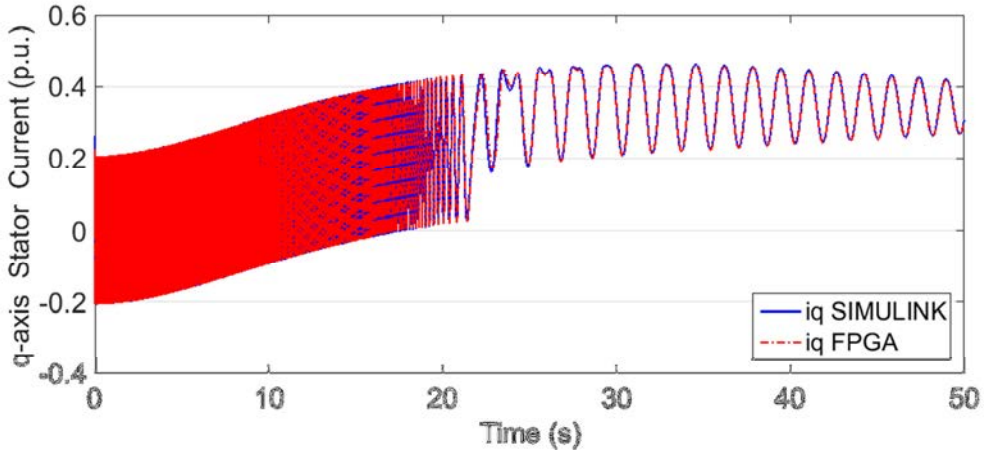
(a) Rotor speed



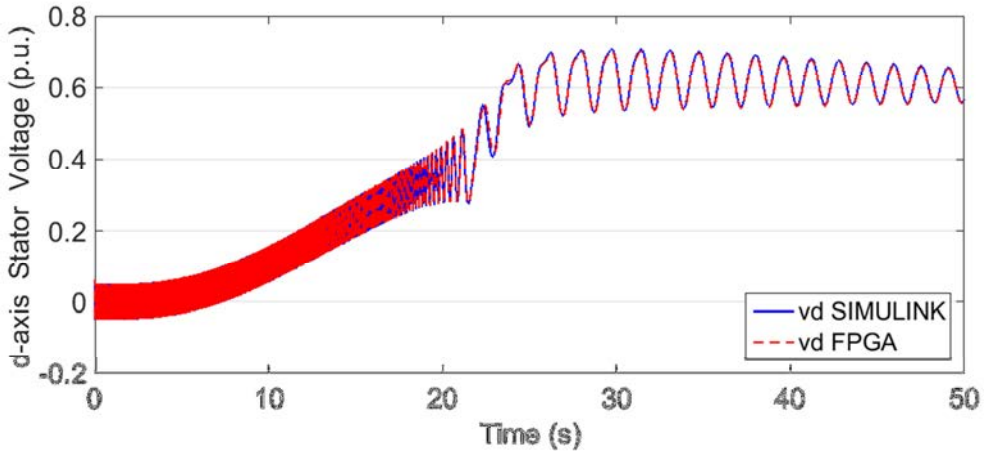
(b) Machine electromagnetic torque



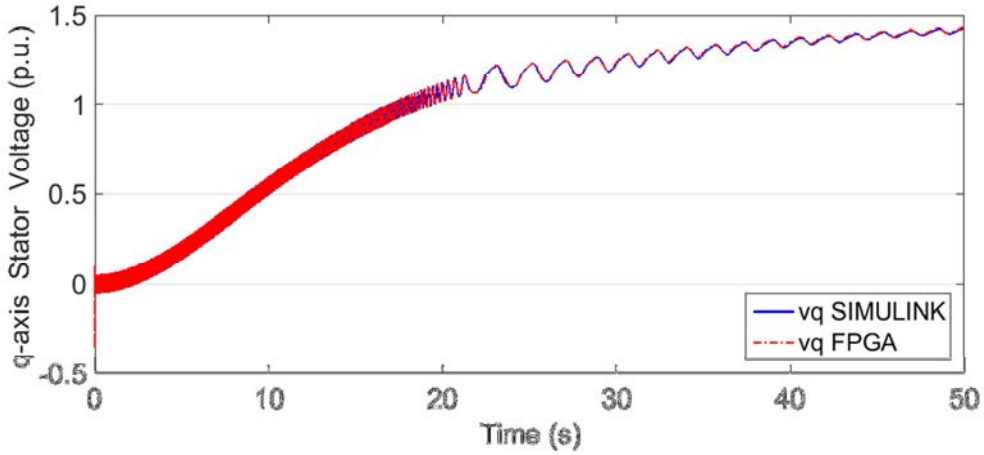
(c) *d*-axis stator current



(d) *q*-axis stator current



(e) *d*-axis stator voltage



(f) *q*-axis stator voltage

Figure 2-34 Starting process for synchronous machine with constant excitation voltage and mechanical torque.

Simulation results of the machine starting process are shown in Figure 2-34. It can be seen from Figure 2-34(a) that the rotor speed of machine is increased to 1 p.u. in about 22s and close agreement between FPGA and SIMULINK are observed. Figure 2-34(b) shows the electromagnetic torque simulated using FPGA and SIMULINK. High-frequency component can be observed and detailed agreement is achieved between results from FPGA and SIMULINK. Figure 2-34(c) - Figure 2-34(f) further compare the simulation results of stator voltages and currents in *dq0* reference frame. Detailed agreement can be observed between the simulation results from FPGA and SIMULNIK.

2.4.2 Case 2: Synchronous Machine with Excitation and Governor Systems

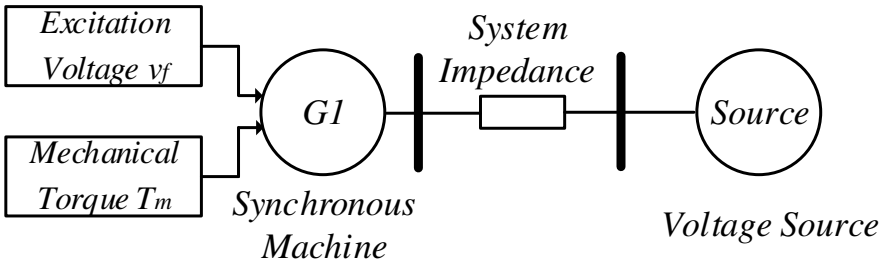
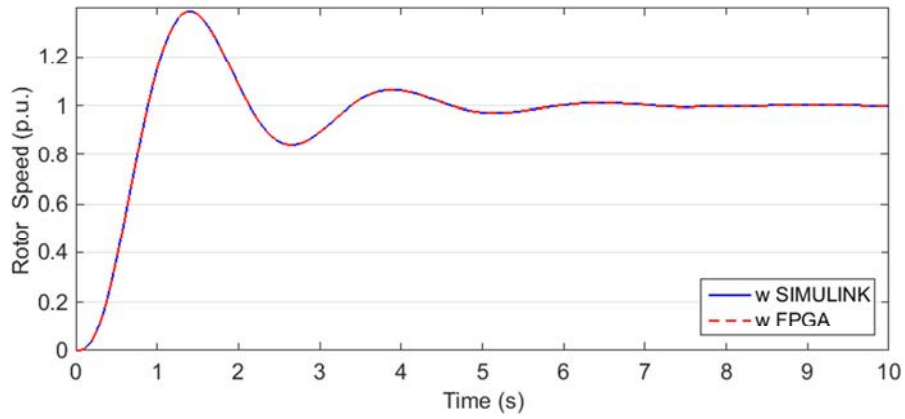
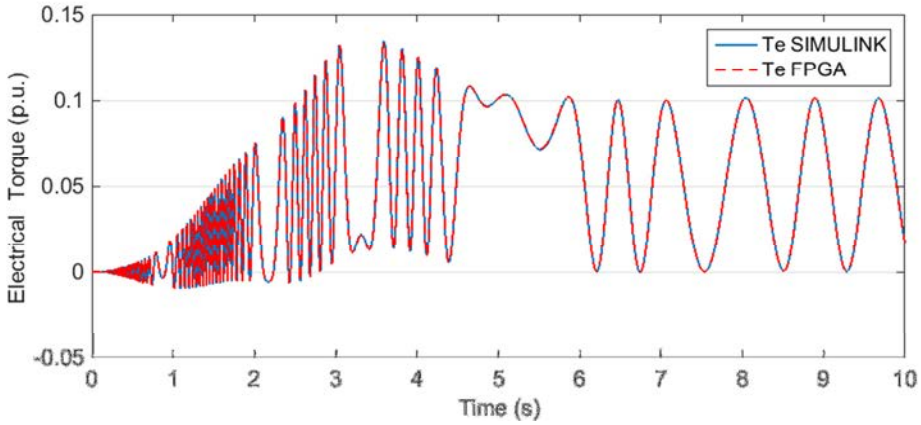


Figure 2-35 Verification of control systems for synchronous machine

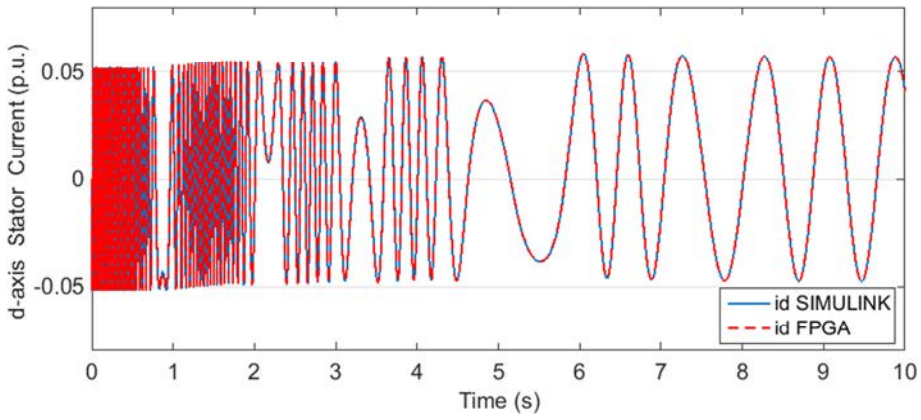
To verify the accuracy of the control systems models, the test system in Figure 2-34 is modified by including the excitation system and governor system described in Section 2.3.1 and Section 2.3.2. The system configuration is shown in Figure 2-35.



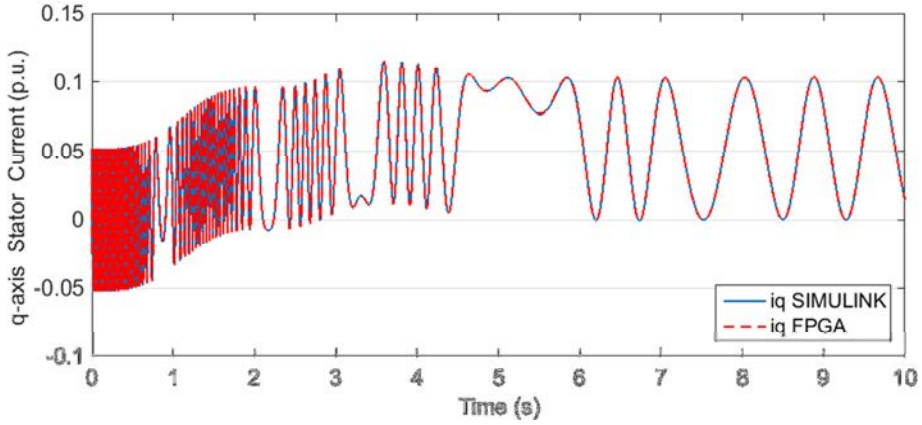
(a) Rotor speed



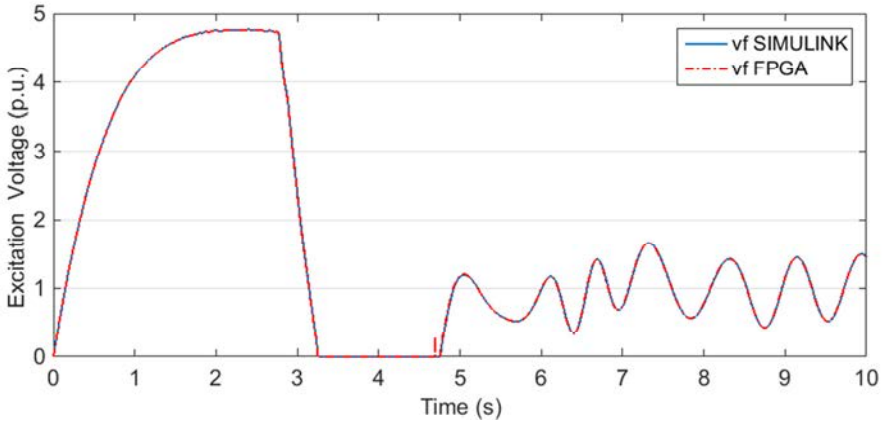
(b) Electromagnetic torque



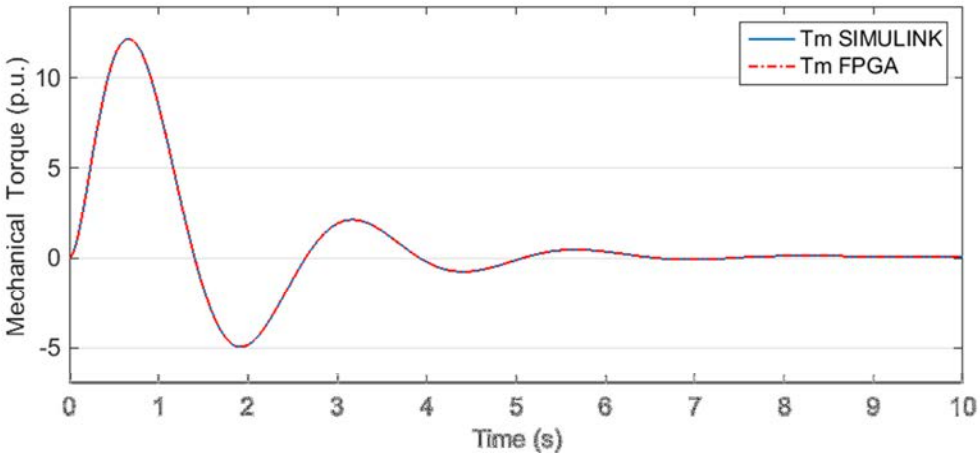
(c) Stator *d*-axis current



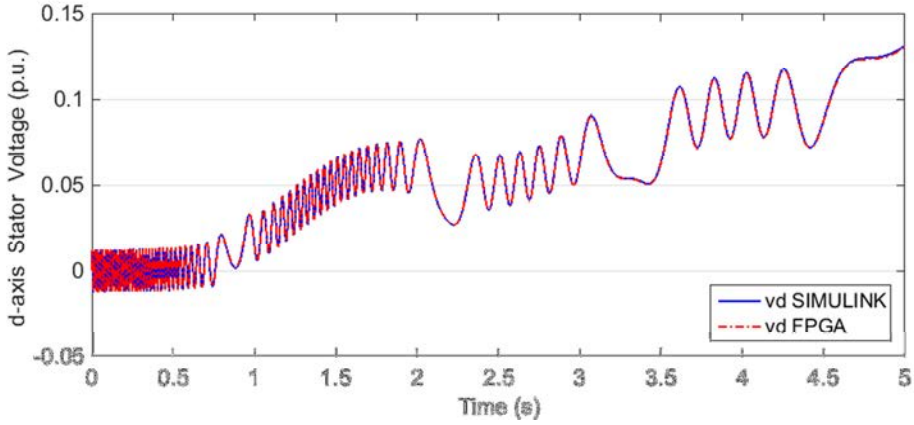
(d) Stator *q*-axis current



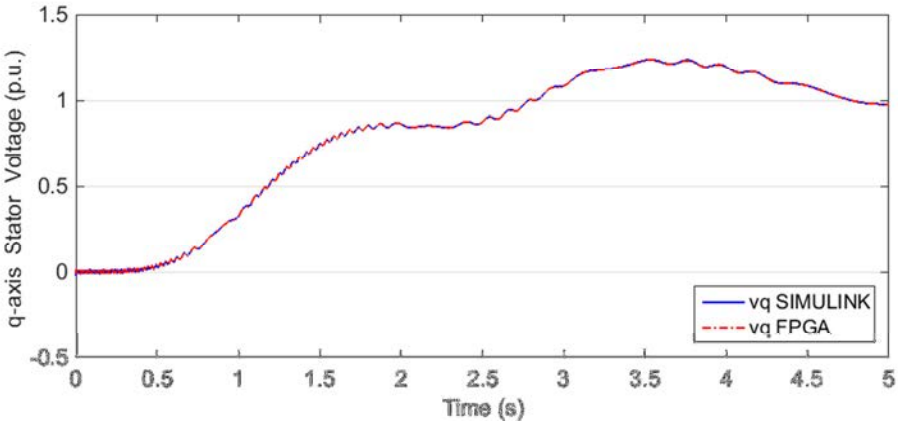
(e) Excitation voltage



(f) Mechanical torque



(g) Stator *d*-axis voltage



(h) Stator *q*-axis voltage

Figure 2-36 Starting process for synchronous machine with excitation system and governor system

Figure 2-36 shows the simulation results of the starting process of synchronous machine with excitation and governor system. It can be seen from Figure 2-36(a) that the rotor speed reaches the reference value of 1 p.u. at around 6s under the governor control. The mechanical torque input from Figure 2-36(f) shows how the governor system is controlling the rotor speed by varying the mechanical torque input. Detailed agreement between the FPGA and SIMULINK can be observed. The simulation results of the change of excitation voltage is shown in Figure 2-36(e), and the *d*- and *q*-axis stator voltages are shown in Figure 2-36(g) and Figure 2-36(h), respectively. It can be seen form the figures that both high- and low-

frequency oscillatory behaviours are correctly simulated using the FPGA. Furthermore, Figure 2-36(b) - Figure 2-36(d) present the comparison results of electromagnetic torque, stator *d*-axis current and stator *q*-axis current. Detailed agreement between the results from FPGA and those from SIMULINK can also be observed.

2.4.3 Case 3: Model Verification of Transmission Lines, Passive Elements, Voltage Sources and Circuit Breakers

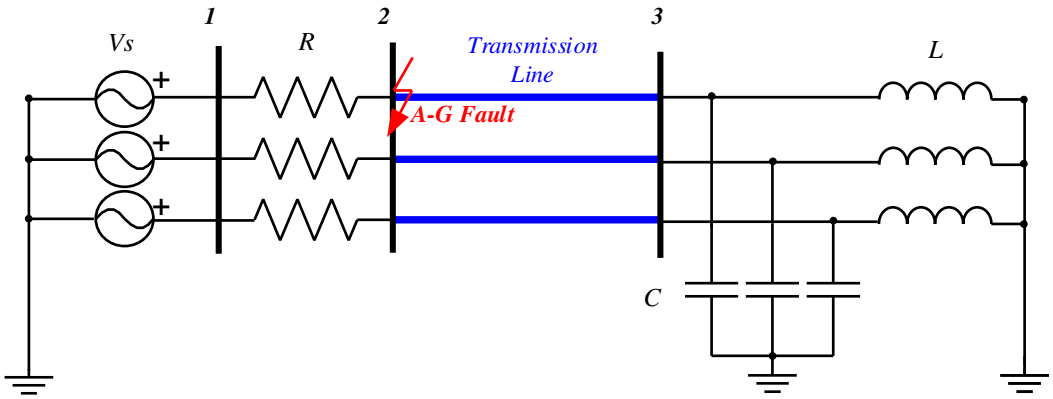
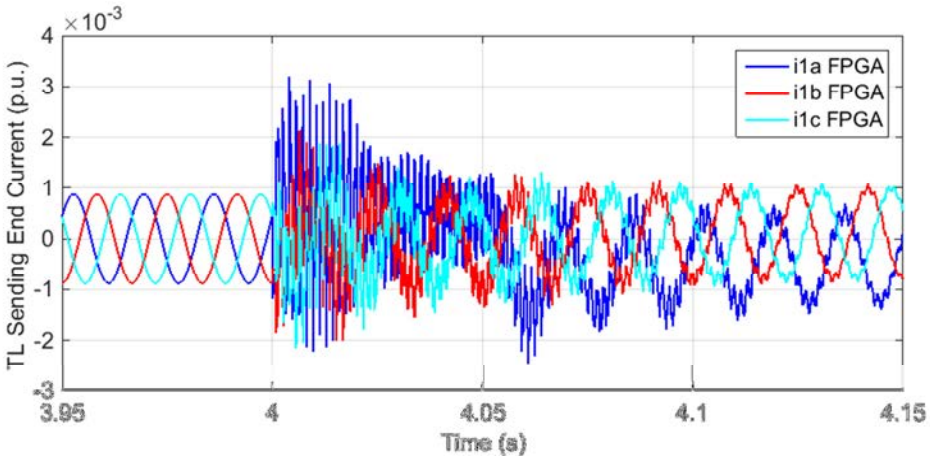
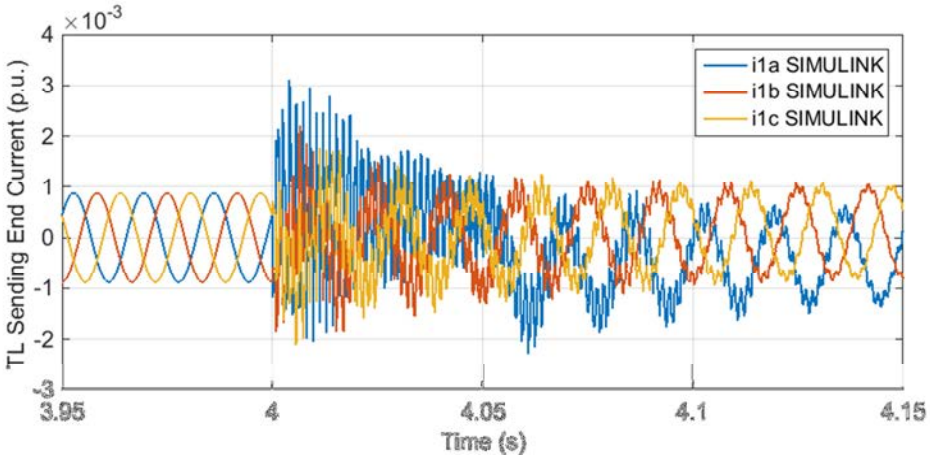


Figure 2-37 Three phase diagram of test system

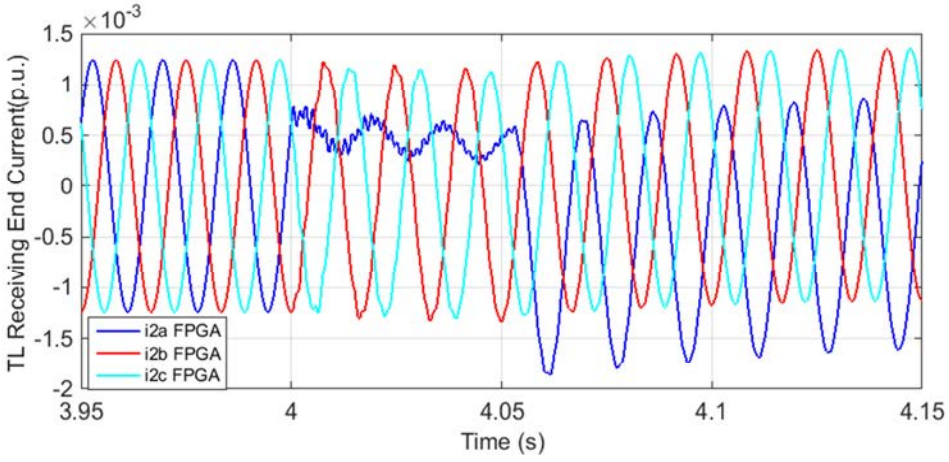
To verify the accuracy of the models of transmission line, passive elements, voltage sources and circuit breakers described in Section 2.2.2 - 2.2.5, the network as shown in Figure 2-37 is simulated using FPGA and comparisons are made with the results from SIMULINK. As shown in Figure 2-37, the network consists of voltage source *Vs*, resistor, capacitor, inductor and transmission line. The circuit breaker is used to simulate the single-phase (phase A) to ground fault which is located at Bus 2 as shown in the figure. The bus 2 is denoted as the sending end and bus 3 is denoted as the receiving end of the transmission line.



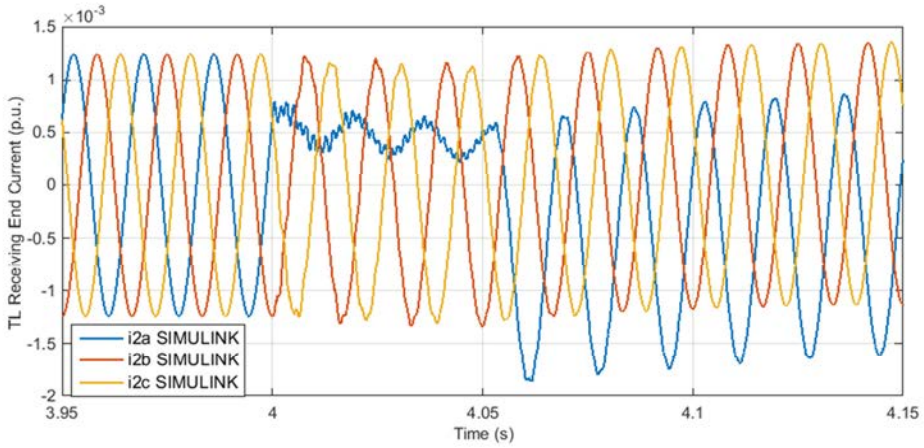
(a) Three-phase sending end currents of transmission line from FPGA



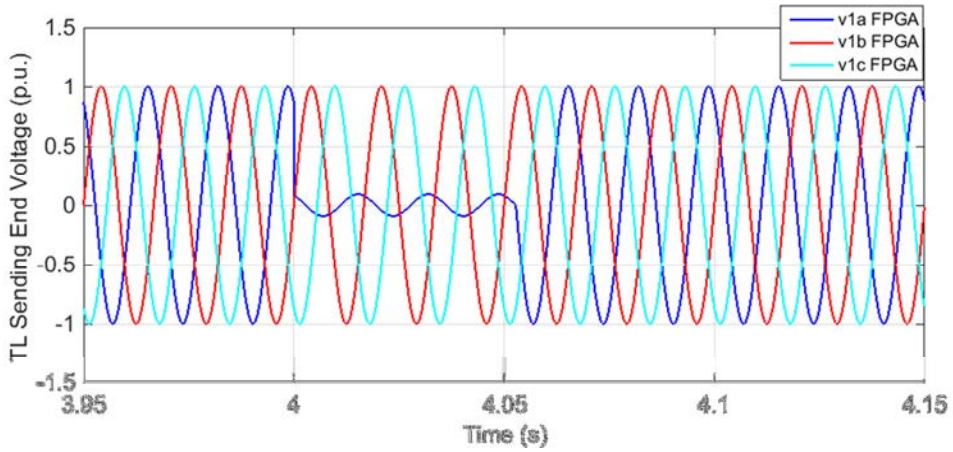
(b) Three-phase sending end currents of transmission line from SIMULINK



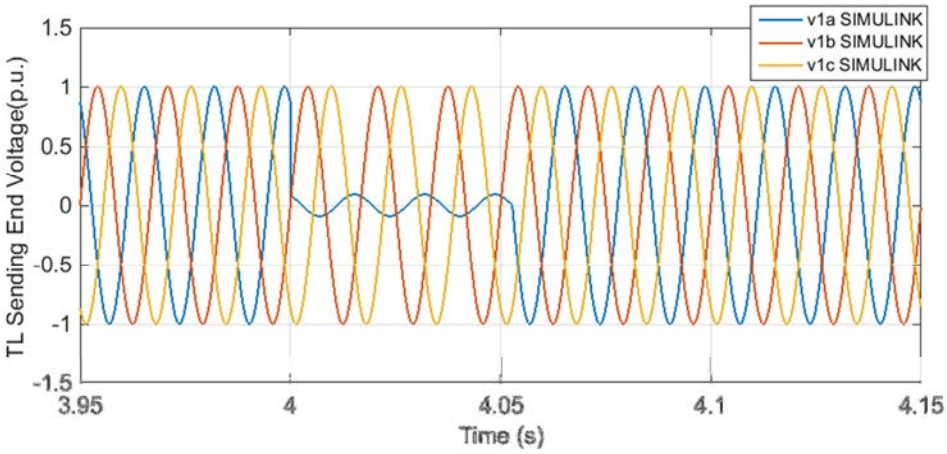
(c) Three-phase receiving end currents of transmission line from FPGA



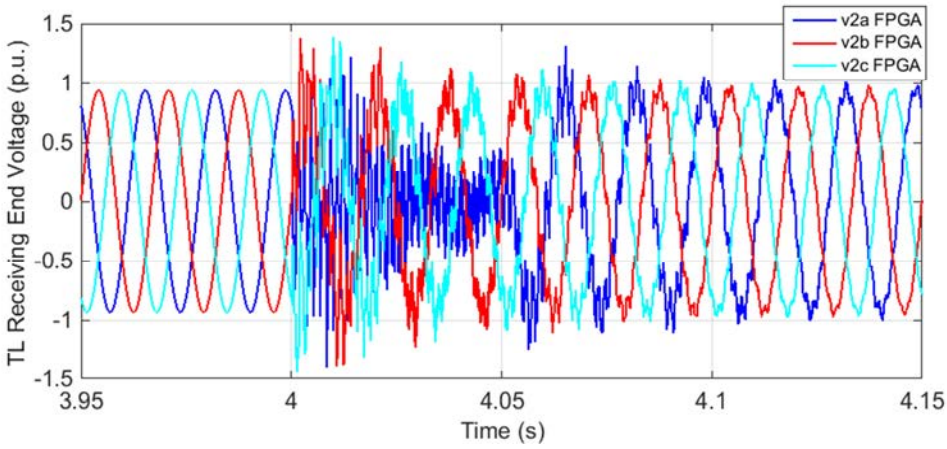
(d) Three-phase receiving end currents of transmission line from SIMULINK



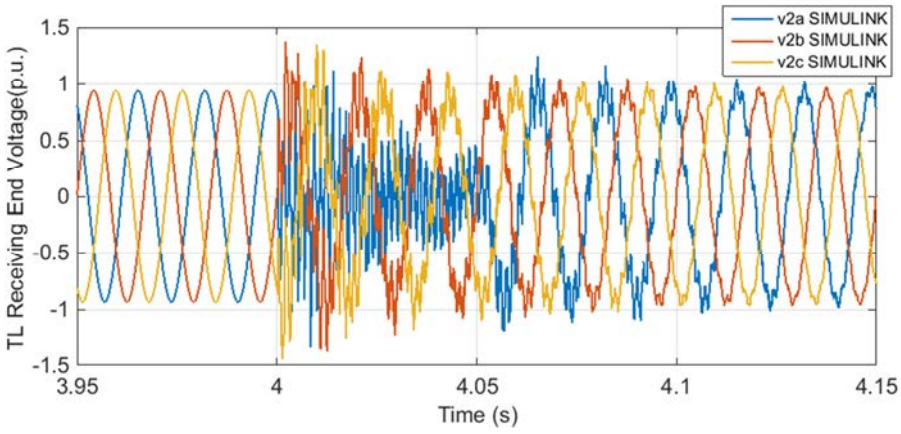
(e) Three-phase sending end voltages of transmission line from FPGA



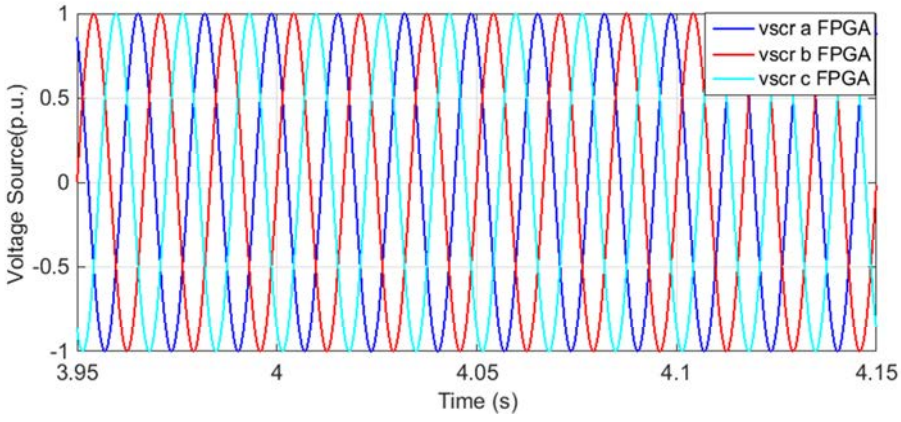
(f) Three-phase sending end voltages of transmission line from SIMULINK



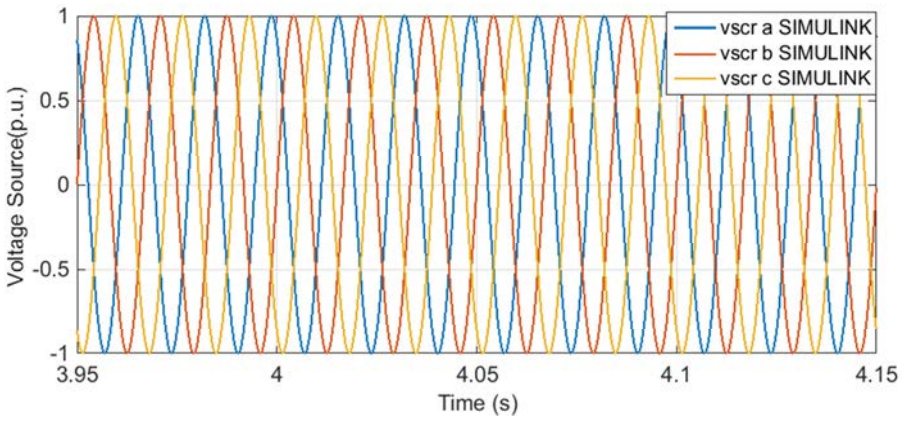
(g) Three-phase receiving end voltages of transmission line from FPGA



(h) Three-phase receiving end voltages of transmission line from SIMULINK



(i) Three-phase voltages of voltage source from FPGA



(j) Three-phase voltages of voltage source from SIMULINK

Figure 2-38 Comparisons of simulation results of the test system

Figure 2-38 compares the simulation results from the FPGA and the SIMULINK. Figure 2-38(a) - Figure 2-38(d) compare the simulation results of the sending end and receiving end currents of the transmission line from FPGA and SIMULINK. It can be seen that High-frequency oscillatory behaviours in the faulty phase (phase A) is accurately simulated using the FPGA. At the same time the effects on phase B and phase C due to the coupling between phases are correctly simulated. Figure 2-38(e) - Figure 2-38(h) compare the simulation results of the three-phase voltages of the sending end and receiving end of the transmission line. It can be seen that the magnitude of the drop of phase A voltage at sending end is accurately simulated and the high-frequency oscillations of phase A voltage at receiving end are also accurately represented. Effects of fault on phase B and phase C due to the couplings between phases are again correctly simulated by the FPGA. Finally Figure 2-38(i) and Figure 2-38(j) compare the three-phase voltages of the voltage sources and detailed agreement can be observed.

2.5 Summary

In this chapter, a library of power system components is developed in FPGA, including most of the common power system elements (synchronous machines, transmission lines, passive elements, voltage/current sources and circuit breakers) and control systems for synchronous machines (excitation systems and governor systems). The EMT models are obtained by discretizing the continuous-time domain models using the trapezoidal rule of integration. The detailed solution steps and FPGA hardware implementations of the developed models have been described. A deeply pipelined algorithm is used to maximize the hardware efficiency. To verify the accuracy of the developed models, test systems have been simulated in both FPGA and SIMULINK and comparisons are made. Close agreements between the simulation results from FPGA and SIMULINK have been observed.

Chapter 3

FPGA BASED REAL-TIME EMT SIMULATOR

3.1 Introduction

Based on the models of power system elements and control systems described in last chapter, this chapter explains the implementation of the FPGA based real-time simulator from a system point of view. Section 3.2 introduces the overall computational architecture of the FPGA based simulator, emphasizing the parallelism between different hardware modules. Section 3.3 introduces the global controller module of the simulator. Section 3.4 explains the methodology and hardware implementation of the power system solver on FPGA. The parallelism of the network solution is fully exploited. Section 3.5 verifies the performance of the FPGA based real-time simulator by simulating the two-area four-machine power system. Comparisons with RTDS are made. Finally Section 3.6 summarized the main aspects of this chapter.

3.2 Computational Architecture

The overall computational architecture of the FPGA based real-time simulator consists of two main parts: the off-line Pre-Simulation part and the Real-Time Simulation part.

The design and implementation of each part is described in next sections.

3.2.1 Pre-Simulation Part

The pre-simulation part is responsible for the preparation of real-time simulation. The pre-calculation and importing of system data, component data, network partition, formation of system matrix and determining the simulation time-step are carried out in this part. The data and parameters will be stored in ROM to reduce computation burden of the real-time simulation part.

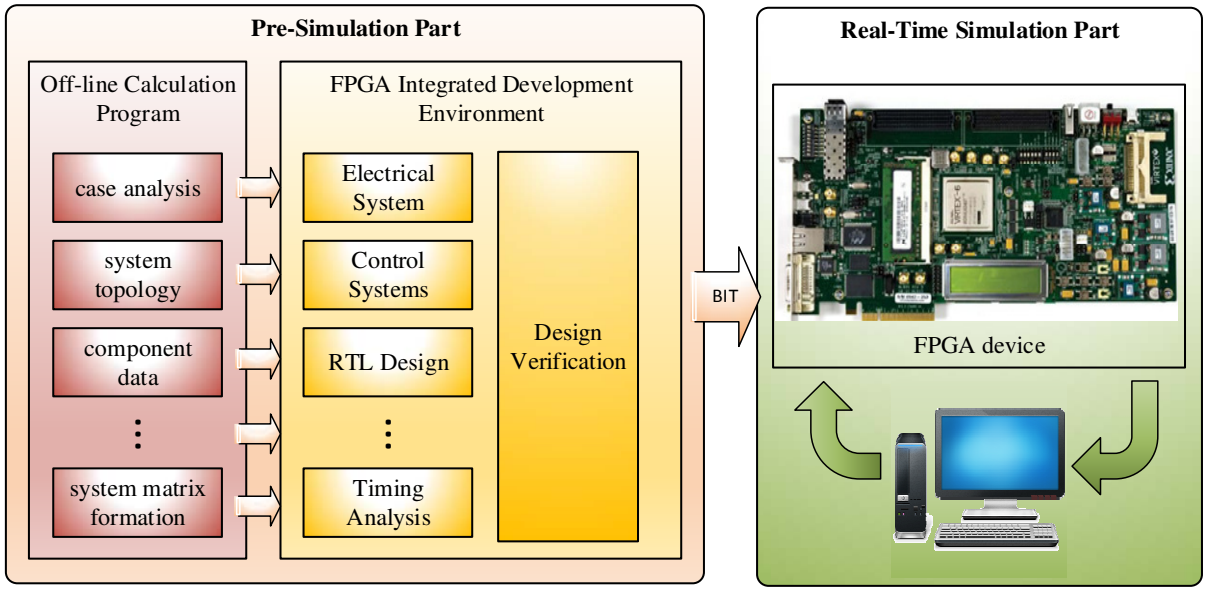


Figure 3-1 Computational architecture

All the pre-calculated data are imported into the FPGA integrated development environment, in this thesis, the Xilinx ISE Design Suit is used. Hardware description language of VHDL is used to design the hardware modules of electrical network and control systems. In this step, the program goes though the FPGA design flow, as shown in Figure 3-2, including steps of design entry, design synthesis, design implementation, and device programming. After several simulation and verifications at different points during the design flow, the developed programme is synthesized and implemented. A programming file (BIT) will be generated and finally downloaded to the FPGA device using JTAG interface and programming cable.

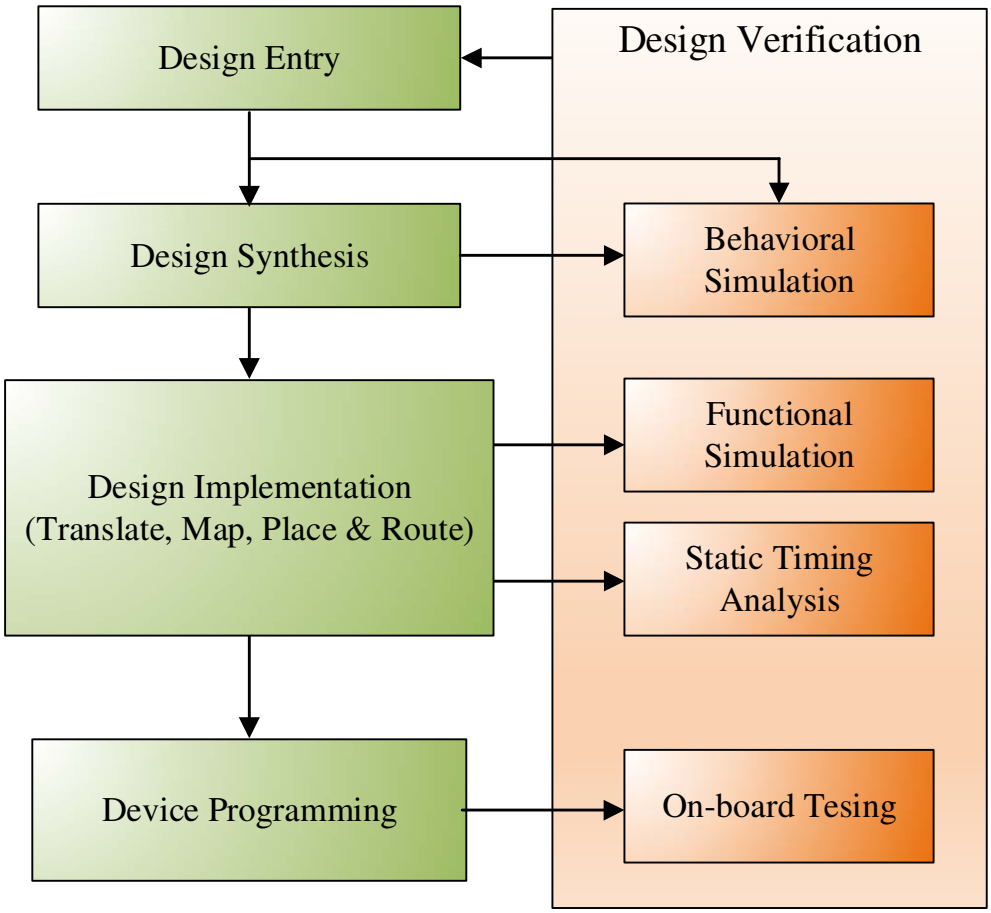


Figure 3-2 Design flow

Once the BIT file is downloaded into FPGA, and the design has been verified through on-board testing, real-time simulation can be carried out. A reset button is used to reset all the registers and states. An output selection button is used to select the output signals and a start button is used to start the real-time simulation. When the simulation has started, the output signals can be sent back to user PC through USB cable, RS232 or other interface methods. Also D/A converter can be used to convert the outputs into analogue signals which can be displayed through oscilloscope.

To improve simulation accuracy and numerical stability, the proposed real-time simulator uses the single-precision floating-point number as the basic number format. The fixed-point

number is used as the supporting number format. This combination of number formats has the advantages of both high accuracy and fast computational speed.

3.2.2 Real-Time Simulation Part

The real-time simulation part achieves its function by cooperating system component computation modules, power system controller modules (as described in Chapter 2), power system solver module and simulator global controller which will be described in the following sections.

3.2.2.1 Computation Schemes

For the calculation of hardware modules, parallelism between modules and the pipelining within each module need to be carefully considered in the design stage.

With one module responsible for the calculation of one type of power system/control element, independent and parallel calculations can be achieved with appropriate allocation of hardware resources. This kind of parallel processing takes advantage of the inherent spatial parallelism between hardware modules, and is of great importance in guaranteeing the speed of real-time simulator.

Apart from the spatial parallelism between hardware modules, deeply pipelined calculation algorithms have been designed within each module itself. Elements of the same type may share the same hardware module, and the calculation is carried out in a pipelined manner. The outputs are obtained every FPGA clock cycle. This kind of processing takes advantage of the parallelism of time in the calculation of same type of element. In some cases, the number of hardware modules which are shared by the same type of elements may need to be increased, when the number of that type of elements is too large or the computation is too

complex or time-consuming. As a result, the hardware resources are balanced and the problem of computation delay caused by one excessively complex module is avoided.

According to the above discussion, the main considerations in the design of computation structure is to take full advantage of the inherent spatial parallelism between modules and a careful design of pipelined processing within each module. In this way, real-time simulation with relatively small time-step can be achieved.

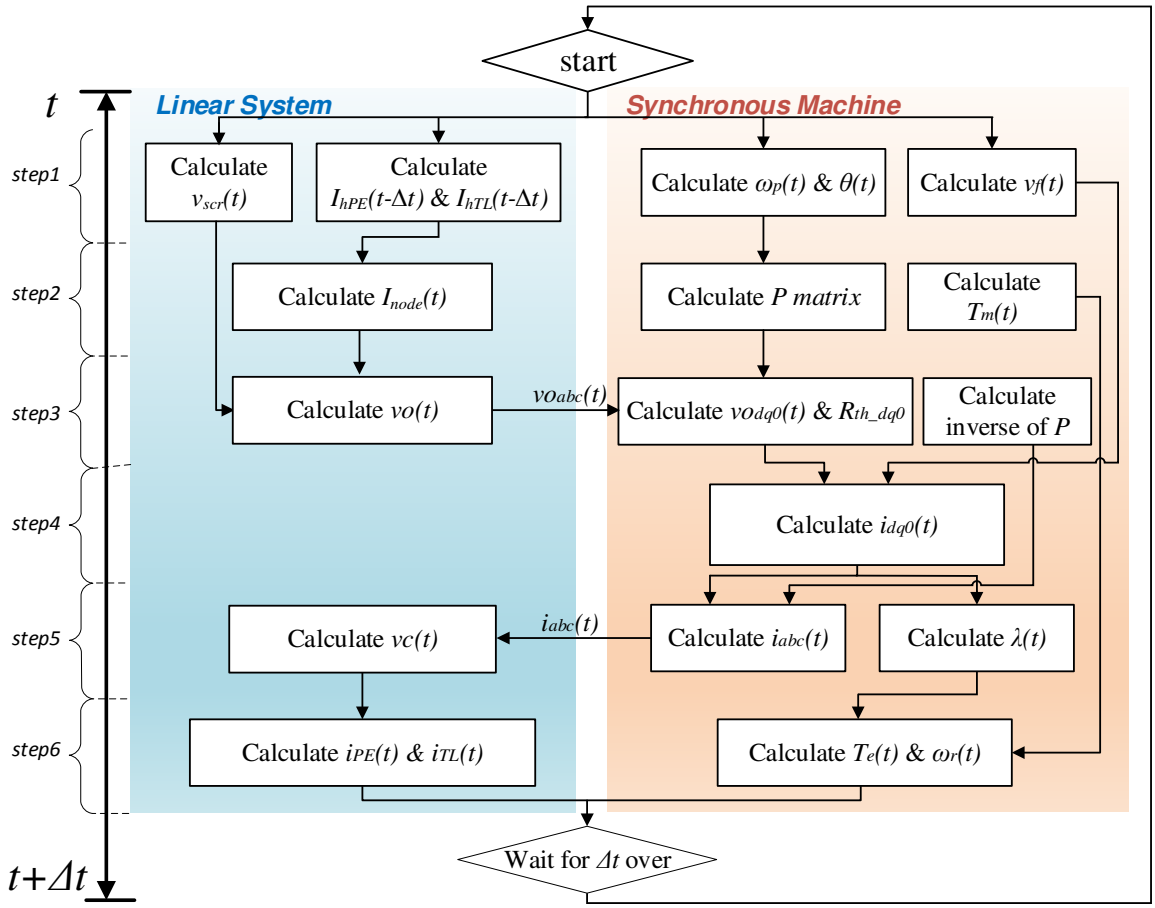


Figure 3-3 Computation scheme of real-time simulation part

In this thesis, the module calculations designed for each time-step is shown in

Figure 3-3. The following aspects are considered in the design:

1. If the calculation of one module requires the results of another module as inputs, the two modules can only be calculated in series and cannot be paralleled. For example to calculate the nodal current injections of the network, history currents are needed and therefore the hardware modules responsible for the relevant calculations can only be processed in series. On the other hand, there is no such problem in the calculations of history currents of the passive elements and transmission lines. These two modules can be arranged to run in parallel.

2. If several modules are designed to work in parallel, dedicated hardware computation path should be designed for each of them. Otherwise it will cause conflict of hardware resources.

3. In general, modules that require similar computing time should be calculated in parallel to maximize efficiency.

Considering all these aspects, six steps of calculation are designed within each time-step. For each step, parallelism between hardware modules is fully exploited. For example parallel calculation is executed between the external network and the solution of synchronous generators. The prediction of rotor speed does not require the system open circuit voltage and therefore can be calculated in parallel with the calculation of system history currents. The calculation of control systems are also processed in parallel with the electrical part of the system. It can be seen that at least two hardware modules are processed in parallel in most of the steps.

The detailed descriptions for each step are:

Step 1: At the start of simulation time-step, the calculations that require input from previous time-step are carried out. In this step, the following hardware modules are calculated in parallel:

- Source module for the calculation of voltage source values
- Passive element module and transmission line module for the calculation of history currents for passive elements and transmission lines
- Prediction module for the prediction of rotor speed and angle of synchronous machines
- Excitation system module for the calculation of excitation voltages

Step 2: In this step, the I_{node} module calculates the nodal current injections using the history currents calculated in step 1. The Park transformation matrix P is calculated using the predicted rotor angle from step 1. These modules are calculated in parallel.

Step 3: Using the nodal current injections from step 2 the equivalent system open-circuit voltage v_0, and its value in *dq0* reference frame, i.e. v_{dq0} are calculated. The inverse of Park transformation matrix *invP* is also calculated in this step.

Step 4: This step is to calculate the stator currents in *dq0* reference frame, i.e. i_{dq0}. The values of v_{dq0}, v_f, and predicted rotor speed w_p as results from other modules are required as inputs.

Step 5: The calculated i_{dq0} from step 4 is taken as input for the calculation of flux linkages, and is transformed into *abc* reference frame (i_{abc}). Then the system nodal voltages are calculated using the values of i_{abc}.

Step 6: Branch currents are calculated in this step. Also the rotor speed is calculated using the flux linkage and mechanical torque input. These values will be used in step 1 of next time-step.

3.3 Simulator Global Controller

In order to implement the above paralleled algorithms, a global control module is designed to coordinate the operations of all the modules.

3.3.1 Module Coordination

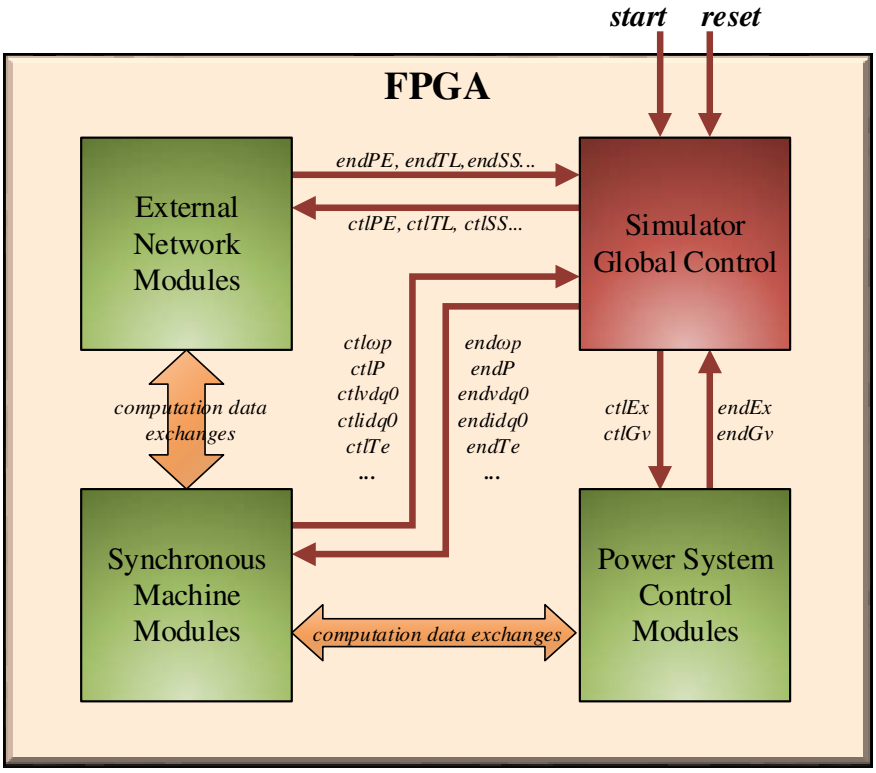


Figure 3-4 Communication signals between modules

As shown in Figure 3-4, the global control module receives the computation completed indicating signals from each module as inputs and sends out start/stop control signals to each module as outputs. It also processes the start and reset command signals from the user and passes them to other modules. The signals of *"endPE" "endTL" "endGv"* etc. are the status indicating signal from each module. When the module completes the calculation required, its indicating signal becomes high and is sent to the global control module. The "*ctlPE" "ctlTL" "ctlGv"* etc. are control signals of each module. They are sent by the global controller

instructing each module on when and what operation should be carried out. As the detailed structure of the FPGA based real-time simulator is complex, only the main control signals are shown in the figure. The synchronous machine modules includes five modules: rotor speed and position prediction module, Park and inverse Park transformation module, stator current module, flux linkage calculation module and electrical torque & rotor speed module. The power system control modules include the excitation system module and governor modules. The rest of modules are represented by the external system modules, including the passive element module, transmission line module, etc.

3.3.2 Hardware Implementation

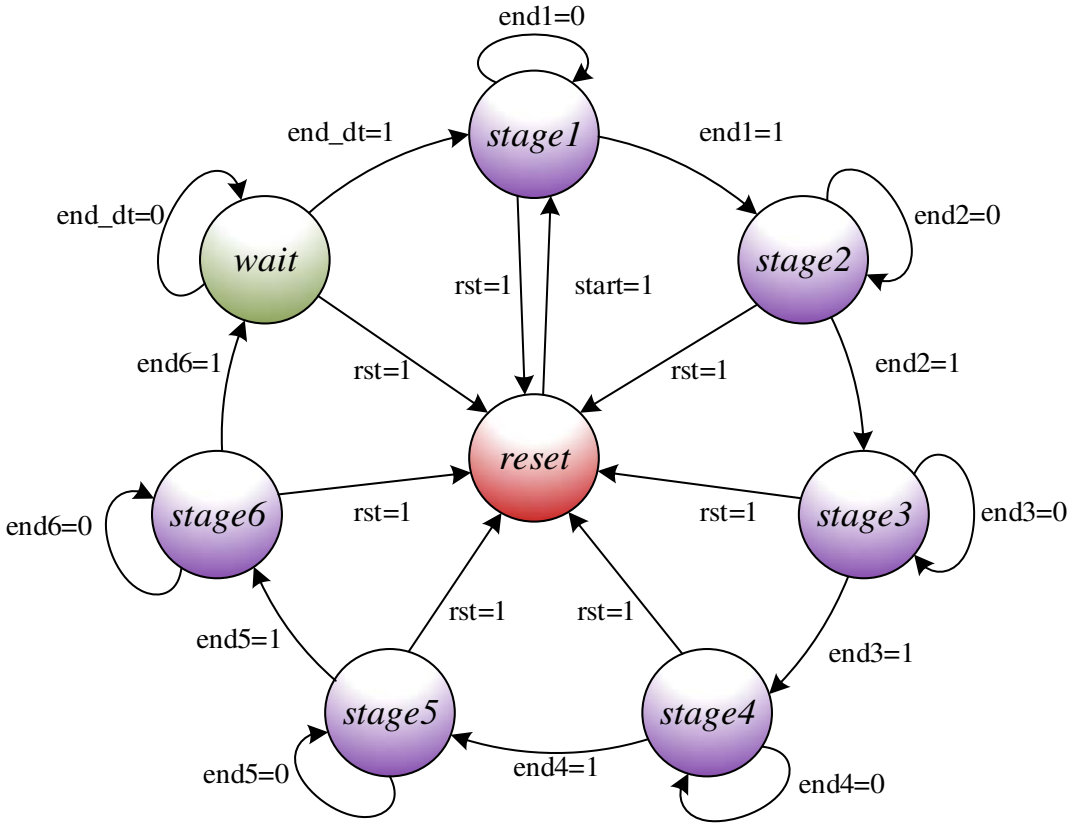


Figure 3-5 Global control module FSM

As the paralleled algorithm consists of 6 steps, a finite-state machine (FSM) is designed as shown in Figure 3-5. The stage 1 to 6 of the FSM correspond to the 6 computation steps of

Figure 3-3 discussed in last section. At stage N, the global control module activates all the parallel modules of that step and monitors their status. When all the calculation of the N^{th} stage is finished, the computation completed indicating signal "*endN*" of that stage will become high. So the finite-state machine proceeds to the next stage and at the same time, sends out starting control signals to modules of the N+1 step, stop signals to modules of the N step. At the end of step 6, which means all the operations in that simulation time-step have been completed, the FSM proceeds to the "wait" stage, where all the modules idle until the start of next time-step. If the reset command "*rst*" is given by the user, the FSM will be forced into the reset stage no matter which stage it is currently at. Stop and reset control signals will be sent to all other modules.

3.4 Power System Solver

With the developed models for power system element and control systems as described in Chapter 2, the power system solver is required to find the complete network solution. In this thesis, the developed power system solver consists of two hardware modules: 1) nodal current injections module and 2) nodal voltages module.

3.4.1 Network Solution

The network solution is to solve the following system equation for nodal voltages:

$$YV = I_{node} \tag{3.1}$$

Where Y is the system admittance matrix, I_{node} is the vector of nodal current injections and V is the vector of system nodal voltages. To obtain the inverse of system admittance matrix Y (N×N) using the conventional explicit calculation method requires N×N×N multiply adds

operations (multiplication and addition) [43]. It means that the computational effort to calculate the inverse of Y becomes prohibitive with large values of N for large scale power systems [152]. Therefore to save computational time of calculating the inverse of system matrix *Y*, it is calculated and stored in FPGA before the start of simulation during the pre-simulation process. This method also avoids the relatively complex calculation of matrix inversion, which at the same time results in significant savings of hardware resources.

3.4.1.1 Network Decoupling and Parallel Computation

Due to the delay of traveling time, the transmission lines within a power system effectively decouple the system into smaller subsystems. From equation (2.33) and (2.34) it can be seen that the current injections at both terminals of a transmission line are only related to the terminal voltage and line current from other terminal at previous time-steps. As the solution of nodal voltages for system at either end of the transmission line only requires local current injections, they can be calculated in a decoupled manner provided that the transmission line is sufficiently long (about 15km for a simulation time-step of 50μs) as discussed in Section 2.2.2.1. This then allows the paralleled computation of each subsystem, hence significantly improves the computational efficiency. Mathematically, the system equation (3.1) after decoupling can be written as:

$$\begin{bmatrix} [Y_1] & & & \\ & [Y_2] & & \\ & & \ddots & \\ & & & [Y_N] \end{bmatrix} \begin{bmatrix} [v_1] \\ [v_2] \\ \vdots \\ [v_N] \end{bmatrix} = \begin{bmatrix} [I_1] \\ [I_2] \\ \vdots \\ [I_N] \end{bmatrix} \tag{3.2}$$

It can be seen from (3.2) that the network decoupling breaks down the system matrix Y into a block diagonal matrix with block elements of *Y1, Y2, … , YN*. Each block element represents the system matrix for each subsystem. Therefore these subsystems can be solved

independently. In FPGA, this means the solution of subsystems can be carried out in parallel to increase the speed of computation.

Another benefits through the decoupling is that its makes the simulation of faults and circuit breakers computationally less complex. The faults/breakers only change the system matrix of that particular subsystem, while the system matrixes for other subsystem are left unchanged. Therefore it becomes easier to store the system matrix for different kinds of fault.

3.4.2 Non-Linear Elements: Compensation Method

For the nonlinear elements in power systems, special solution method needs to be applied. In this thesis, the compensation method is used to handle the nonlinear elements in power systems [43]. The steps of this method are listed as follows:

Step 1: The network is solved with current injection of one from all nonlinear branches. The calculated terminal voltages for nonlinear branches are their equivalent Thevenin impedances.

Step 2: The linear part of the network is solved without the nonlinear elements. The open-circuit voltages of the nodes connecting to the nonlinear elements are obtained.

Step 3: The nonlinear elements are solved with the external network represented as a Thevenin equivalent. The Thevenin equivalent is formed using the open-circuit voltage from step 2 and the impedance from step 1. The results of calculation are the current injections from nonlinear branches.

Step 4: The calculated current injections from step 3 are then super-imposed on the linear part of the network to get the actual system nodal voltages.

Step 1 is carried out in the pre-simulation part. Step 3 is completed by the modules of nonlinear components, e.g. the synchronous machine related modules. Step 2 and 4 both require the system solver, which consists of the nodal current injections module and the nodal voltages module.

3.4.3 Hardware Implementation

To solve the system nodal equation(3.1), firstly the nodal current injections I_{node} need to be calculated, then the nodal voltages can be calculated.

3.4.3.1 Nodal Current Injections

The hardware module of nodal current injections is responsible for calculating the I_{node} . This hardware module includes two sub-modules:

- One is responsible for the formation of history current vector consisting of history currents from different power system elements (Figure 3-6).
- The other is responsible for the formation of nodal current injections (Figure 3-7).

For the first sub-module, the inputs are history currents from each individual element and the associated addresses of nodes that element is connected to. For passive elements, the directions of history currents are defined as flowing out of the starting node and into the ending node. The formation of history current vector then becomes the addition of history currents for starting node, and subtraction of history currents for ending node. It can be seen from the upper part of Figure 3-6 that the input history current is firstly converted from floating-point number into fixed-point number and then sent to the accumulation unit and subtraction unit. The results of addition and subtraction are stored into the corresponding

addresses in the output RAM. The calculations of accumulation unit and subtraction unit are in parallel for higher efficiency.

For the transmission line elements, the hardware implementation is shown in the lower part of Figure 3-6. It can be seen from the figure that it is very similar to that of passive element. The main difference is that two subtraction units are used for both starting and ending node. This is because the history currents are flowing out of both nodes, as shown in Figure 2-11.

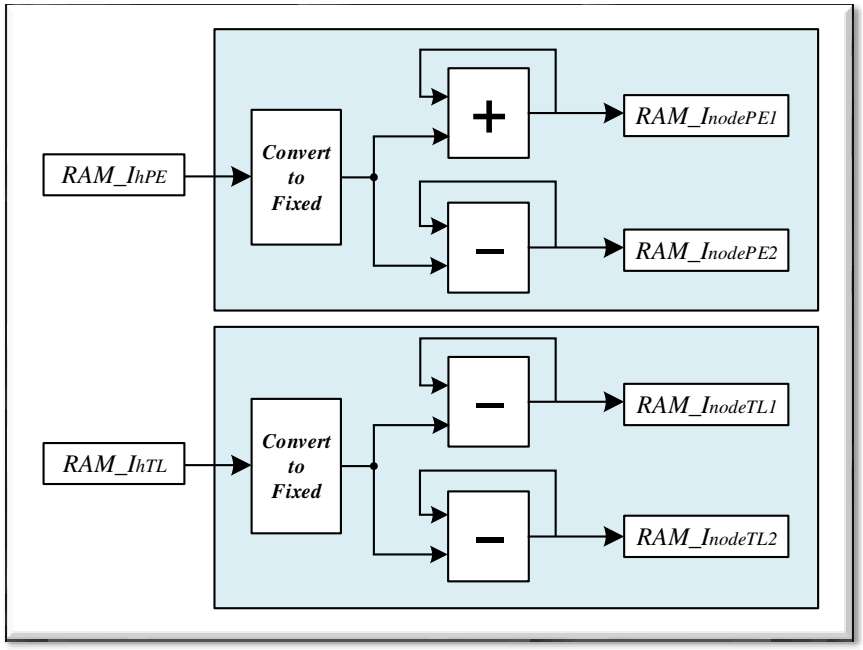


Figure 3-6 Hardware implementation of history current formation submodule

For the second sub-module, the inputs are the history current vectors calculated from the first sub-module. This module then adds all the history current vectors for each node to form the nodal current injections as illustrated in Figure 3-7. To achieve high computational efficiency, floating-point calculations are used for this sub-module. The calculated values need to be converted back to floating-point numbers for other hardware modules.

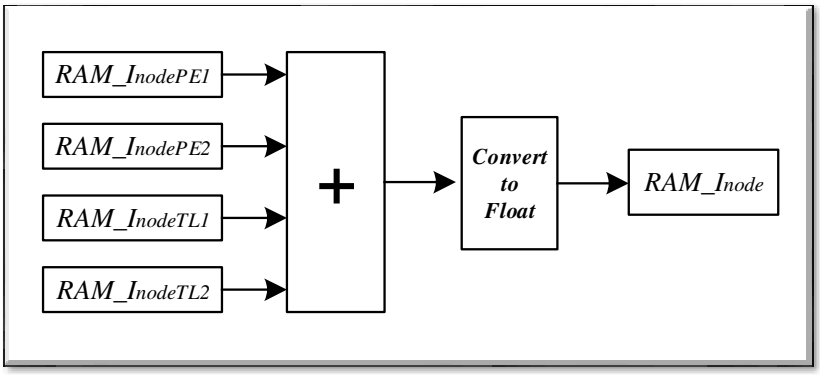


Figure 3-7 Hardware implementation of calculating nodal current injections

3.4.3.2 Nodal Voltages

The nodal voltages module is the module for calculating open-circuit voltage of nonlinear branches and calculating system nodal voltages. Both steps are essentially solving the system equation of (3.1), where the nodal current injections are obtained from nodal current injections module discussed in last section. For each subsystem, the nodal voltages can be classified into those that are known and those that are unknown. Then the system equation for each subsystem becomes:

$$\begin{bmatrix} Y_A & Y_B \\ Y_C & Y_D \end{bmatrix} \begin{bmatrix} v_{known} \\ v_{unknow} \end{bmatrix} = \begin{bmatrix} I_A \\ I_B \end{bmatrix} \tag{3.3}$$

Where v_{known} are the vector containing known nodal voltages and v_{nknown} is the vector containing unknown nodal voltages. I_A and I_B are the nodal current injections for the corresponding nodes. The $v_{unknown}$ can then be expressed as:

$$v_{unknown} = Y_D^{-1}(I_B - Y_C v_{known}) = Y_D^{-1} I_B - Y_D^{-1} Y_C v_{known} \tag{3.4}$$

For typical power systems, the system matrix Y is mostly sparse as there is no direct connection between most of the nodes. Therefore only the non-zero elements in the system

matrix are stored in RAM to minimize the computational time and hardware requirement. These elements, together with the corresponding nodal current injections and known node voltages, are sent to the matrix multiplication module to calculate the unknown nodal voltages.

Based on all the hardware modules introduced in previous sections, the FPGA-based EMT simulator is developed. Figure 3-8 shows the hardware architecture of the implementation in detail. The control signals and data exchanges between modules are illustrated in the figure

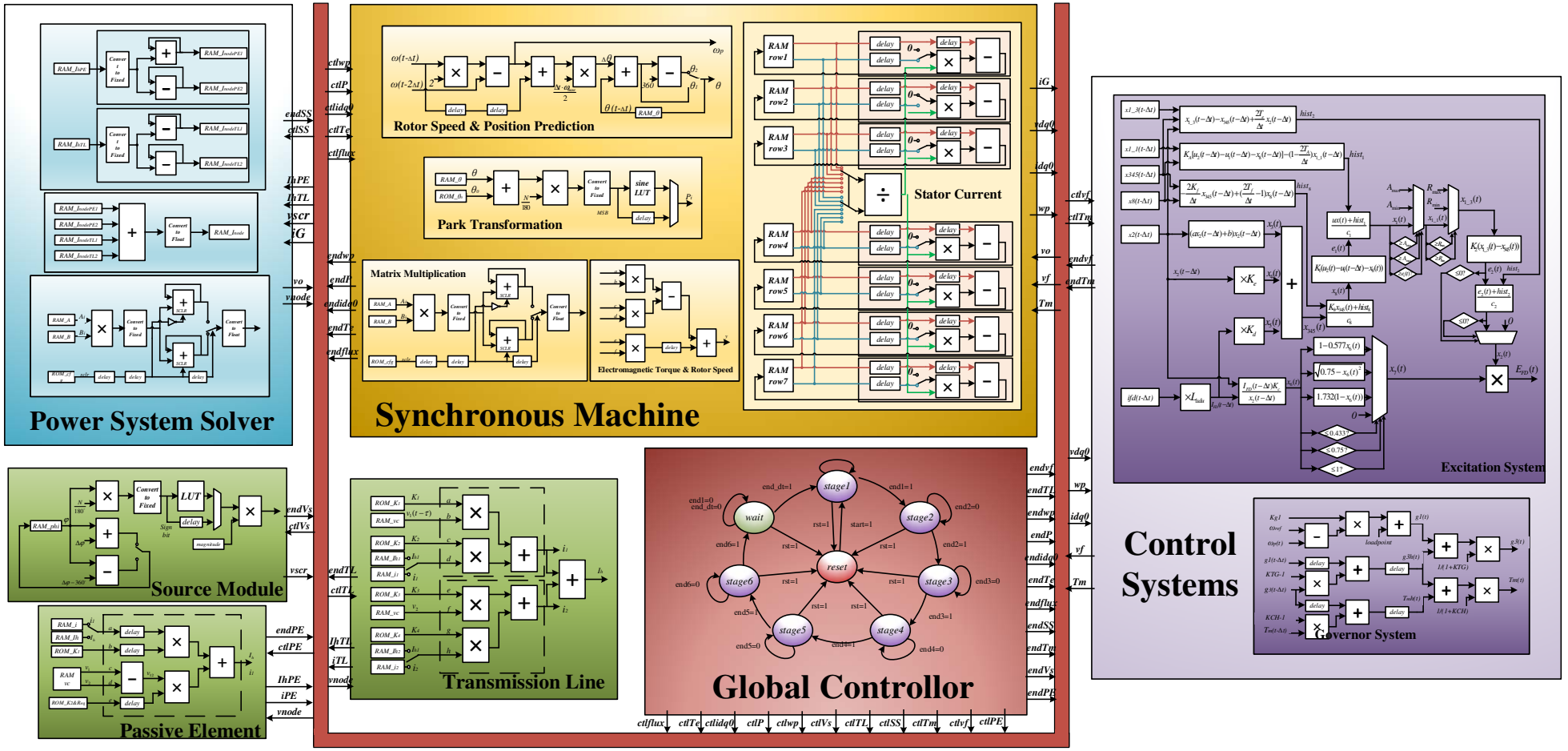


Figure 3-8 Detailed hardware architecture of the FPGA-based EMT simulator

3.5 Case Study

To validate the proposed FPGA-based EMT simulator, an example power system as shown in Figure 3-9 is modelled and simulated on Xilinx Virtex-6 FPGA. Simulation results from proposed simulator are compared with that from SIMULINK and RTDS. It is a two-area power system consists of four synchronous generators with their excitation and governor systems. There are eight transmission lines, four transformers, two loads and two shunt capacitor banks in the system. The models described in Chapter 2 are used for generators, excitation systems, transmission lines, RLC components and faults. The computational algorithms described in Section 3.2 and Section 3.3 are implemented in FPGA to solve the network. The complete system data is obtained from [147] and listed in Appendix B.

3.5.1 Case 1

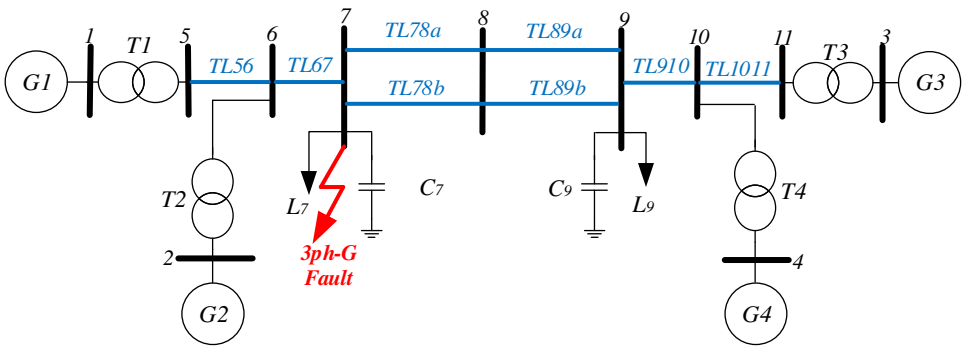
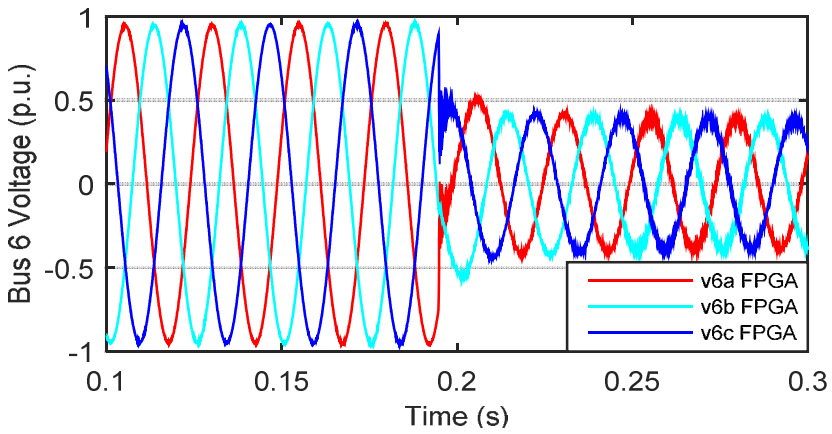
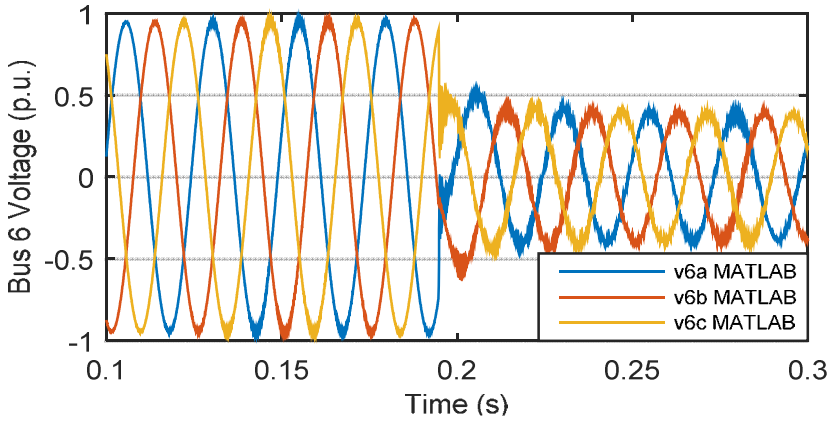


Figure 3-9 Single-line diagram of case study1

A three-phase grounding fault located at bus 7 is simulated and the results from FPGA are compared with those from SIMULINK. The fault is initiated at 0.195s and the simulation results are shown in Figure 3-10 to Figure 3-12. The per unit base value is 100MVA and 230kV.



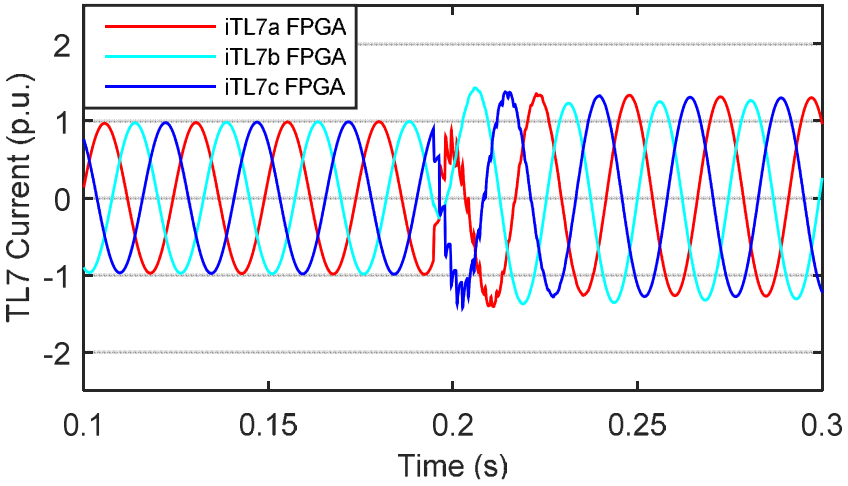
(a)) Simulation results from FPGA;



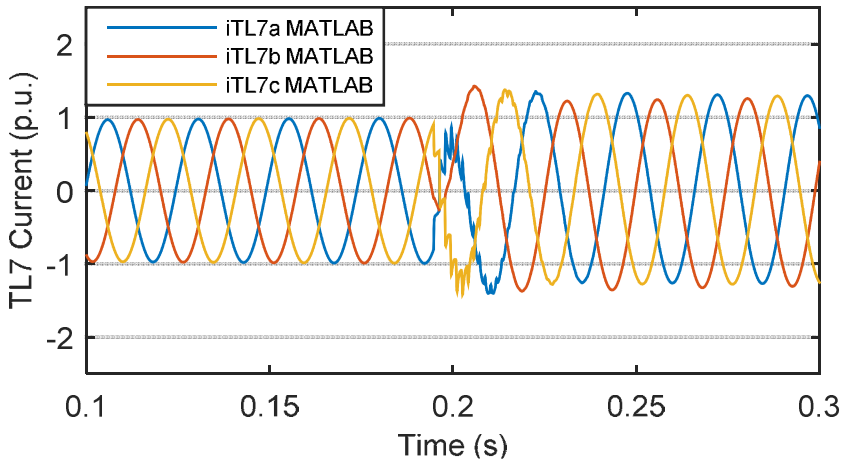
(b) Simulation results from MATLAB

Figure 3-10 Three-phase voltage at bus 6

Figure 3-10 shows the three-phase voltage at bus 6. Figure 3-10(a) is the simulation results from FPGA-based simulator and Figure 3-10(b) is the results from SIMULINK. It can be seen from the figure that as fault happens, the three-phase voltages are dropped and very similar results can be observed between FPGA and SIMULINK.



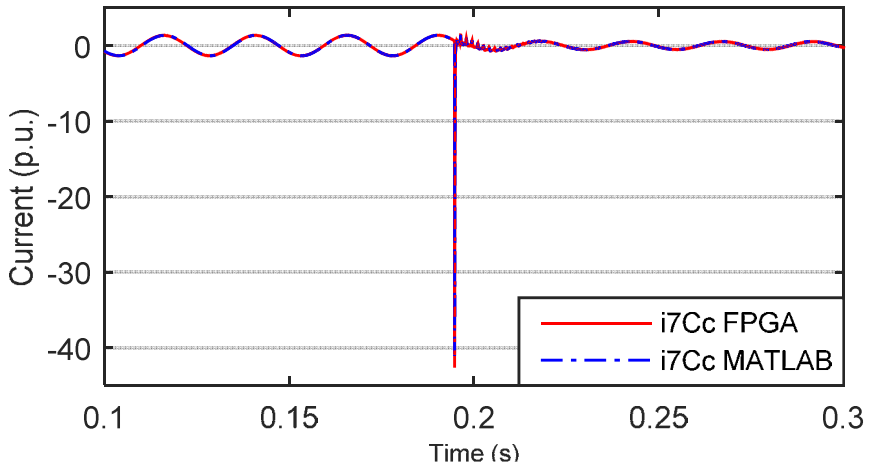
(a) Simulation results from FPGA



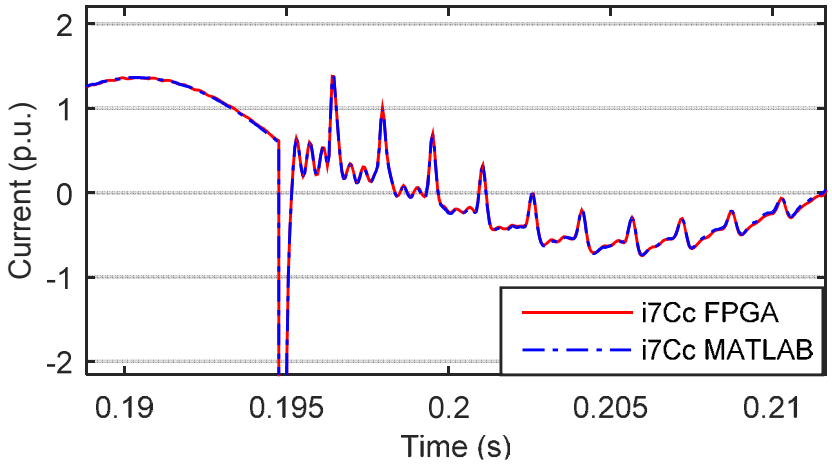
(b) Simulation results from MATLAB

Figure 3-11. Three-phase line current between bus 7 and bus 8 (TL78a)

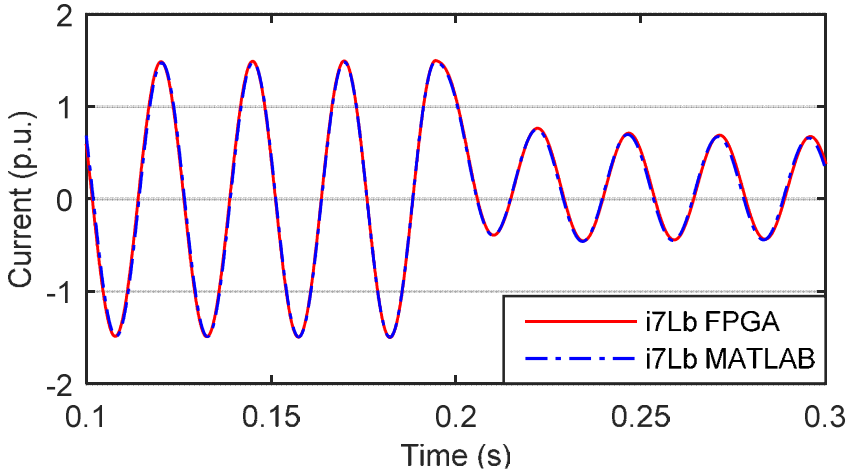
Figure 3-11 shows the simulation results of three-phase currents through transmission line *TL78a* from FPGA-based simulator and SIMULINK. High-frequency oscillations can be seen after the fault is initiated from Figure 3-11 (a) and Figure 3-11 (b), and detailed agreement between the two results can be observed.



(a) Phase C current through) C7



(b) zoomed in version of (a) for the period when fault is initiated



(c) phase B current through L

Figure 3-12 Current at fault location

Figure 3-12(a) shows the phase C current through C7 and Figure 3-12(c) shows the phase B current through L7. Figure 3-12(b) is the zoomed in version of Figure 3-12(a) around the time when fault is initiated. It can be seen in Figure 3-12(a) that a transient current with very large magnitude goes through C7 as fault happens and the proposed FPGA-based simulator correctly reflects the phenomena. Figure 3-12(b) demonstrates that a detailed agreement between results obtained from the FPGA and SIMULINK is achieved. The high frequency oscillatory behaviours of the capacitor current are successfully simulated by the proposed simulator. Similarly, for phase B current through L7 during and after fault, very similar behaviours can be observed.

3.5.2 Case 2

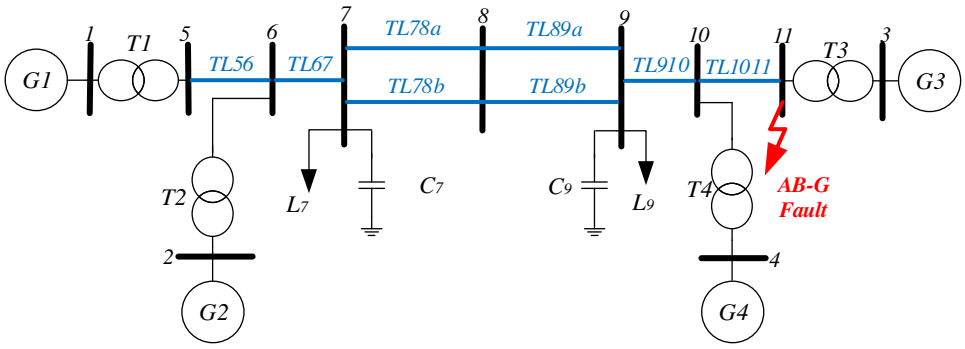
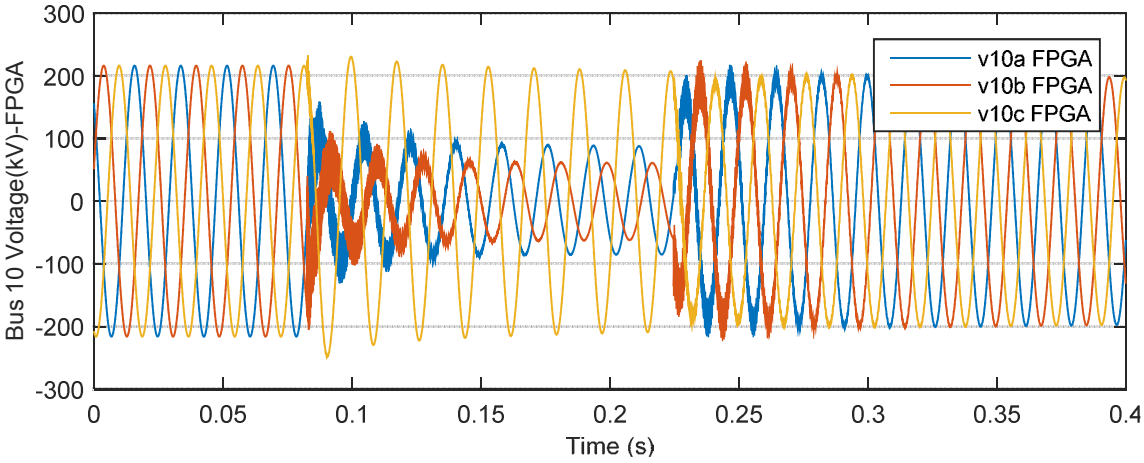
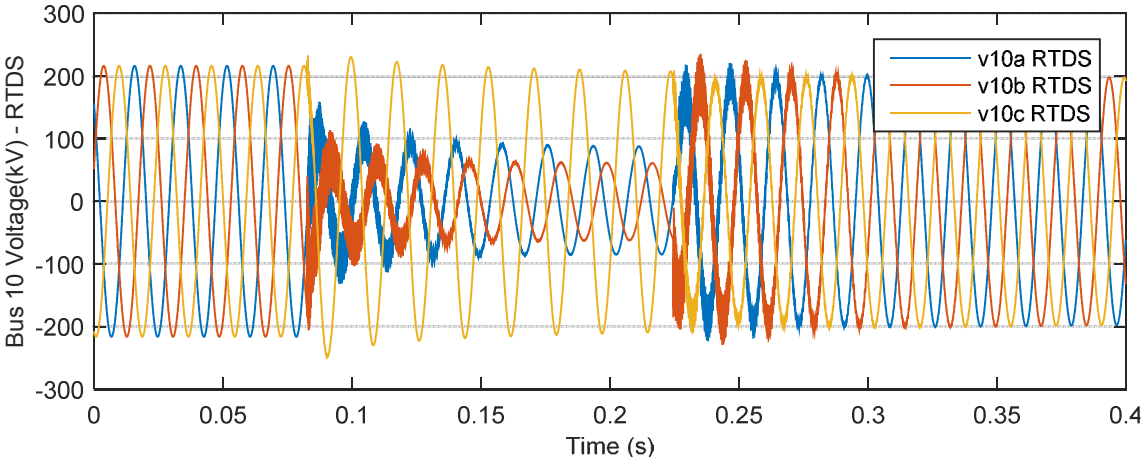


Figure 3-13 Single-line diagram of case study 2

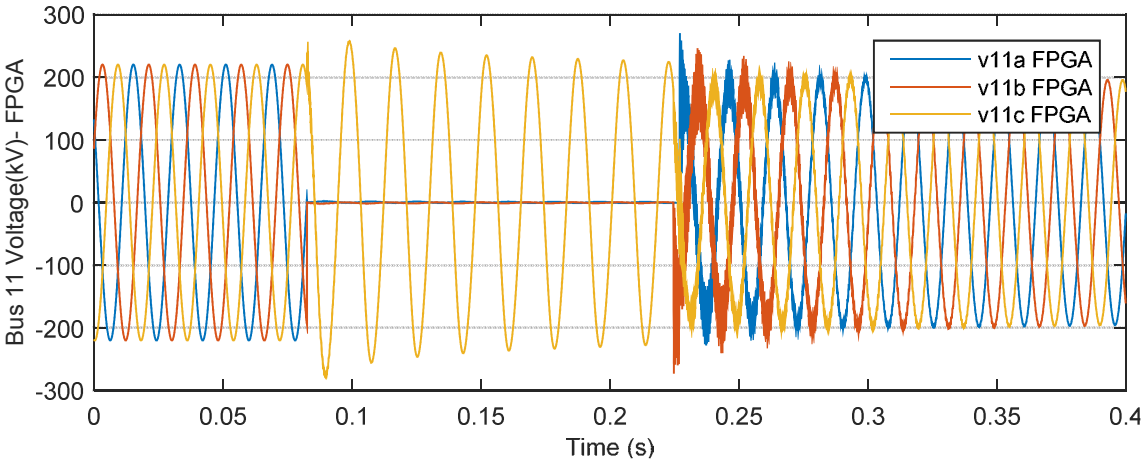
An unbalanced double phase (phase A and phase B) to ground fault located at bus 11 is simulated and the results from FPGA are compared with those from RTDS. The fault duration is 150 ms and the simulation results are shown in Figure 3-14.



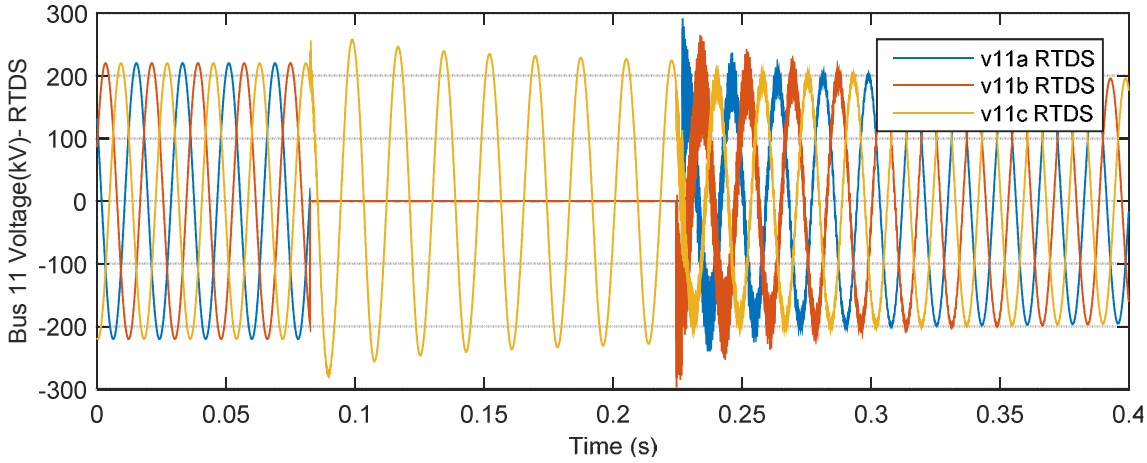
(a) Bus 10 voltages from FPGA



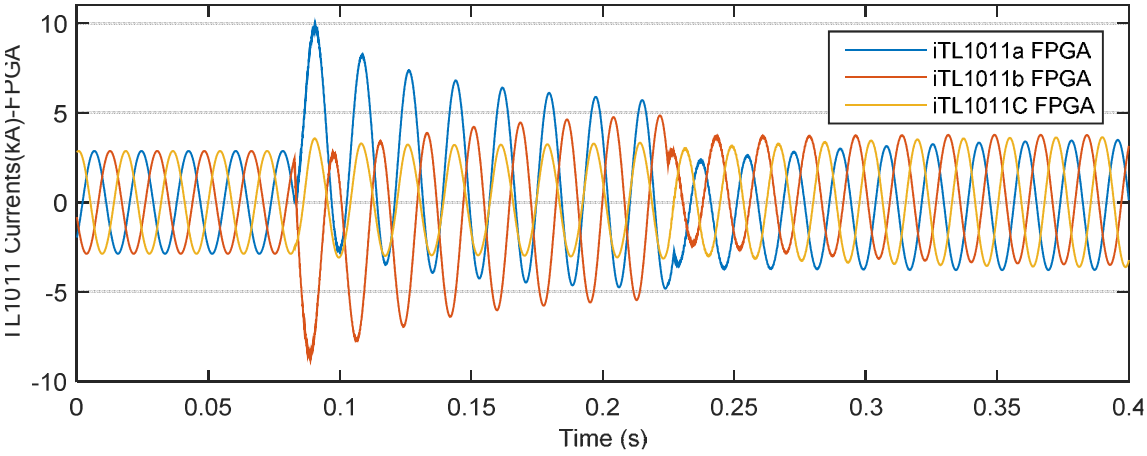
(b) Bus 10 voltages from RTDS



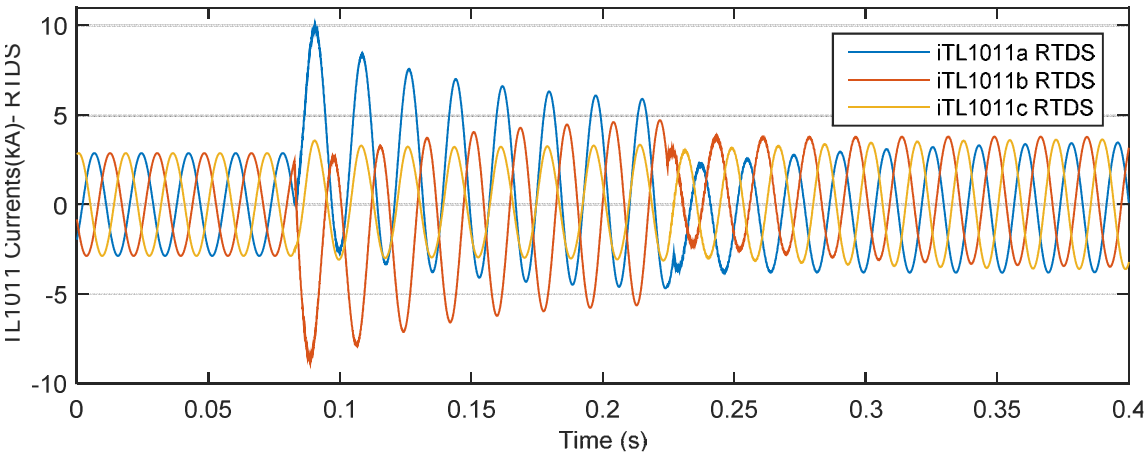
(c) Bus 11 voltages from FPGA



(d) Bus 11 voltages from RTDS



(e) Three-phase line currents on TL1011 from FPGA



(f) Three-phase line currents on TL1011 from RTDS

Figure 3-14 Simulation results for Case 2

Figure 3-14(a) and Figure 3-14(b) show the three-phase voltages at bus 10 from FPGA and RTDS, respectively. It can be seen that when fault happens, voltages of phase A and phase B are dropped and the voltage of phase C is also affected, which is caused by the mutual coupling between phases. Figure 3-14(c) and Figure 3-14(d) show the voltages at bus 11 from FPGA and RTDS. It can be seen that the high-frequency dynamics after fault clearance are accurately simulated. Figure 3-14(e) and Figure 3-14(f) compare the results of the transmission line currents on TL1011 and detailed agreements can be observed.

3.6 Summary

In this chapter, a FPGA-based EMT simulator has been developed. Detailed modelling of power system components including synchronous generator, excitation & governor systems and transmission line models has been implemented in FPGA. The design of simulator global controller module and power system solver module have been described in detail. Carefully designed paralleled and pipelined algorithm during hardware implementation optimizes the hardware resources and permits the real-time computation of the proposed simulator. Its simulation accuracy has been verified through comparison with SIMULINK and detailed agreement has been achieved.

Chapter 4

REAL-TIME FPGA-RTDS CO-SIMULATOR

4.1 Introduction

This chapter introduces the FPGA-RTDS co-simulator, based on the FPGA-based real-time simulator discussed in Chapter 3. Section 4.2 briefly discusses the advantages and disadvantages of FPGA and RTDS. Section 4.3 introduces the architecture and hardware implementation of the proposed co-simulator. Discussion on the expandability of the model in FPGA is presented. In Section 4.4 two case studies are presented to verify the simulation accuracy and capability of the proposed co-simulator. Section 4.5 summarises this chapter.

4.2 FPGA vs RTDS

RTDS has been widely used in academia and industry for over two decades, mainly due to its real-time nature and accuracy of simulation. It has a comprehensive library of power system components and mature Graphical User Interface (GUI). However, when it comes to simulation of large scale power systems with detailed EMT models, RTDS may not be the best choice. For example, for the simulation of a 68-bus system with 16 synchronous machines, 4 RTDS racks are required. The hardware requirements and the associated costs can be very high for the simulation of system with hundreds, if not thousands of buses.

On the other hand, with the massively paralleled and pipelined design, the FPGA can be an economical option in simulating relatively large power systems. For example, the cost of one

FPGA board is normally less than 10% the cost of one RTDS rack, while the simulation capability of one FPGA is similar or even higher than one RTDS rack. However, one disadvantage of using FPGA is that it does not have a Graphical User Interface (GUI) for power system simulation as that of RTDS. The flexibility of visualizing results and modifying the network that is simulated in FPGA is limited to professional personals.

Therefore it would be significantly beneficial that if the flexibility of RTDS and the simulation capability of FPGA can be combined for the simulation of large power systems.

It should be emphasized that compared with the hybrid simulation using EMT and TS programmes, the proposed approach does not have interface problems and has better accuracy. This is because EMT models are used at both RTDS and FPGA sides and no simplifications are made for the models and interfaces.

4.3 Architecture of the FPGA-RTDS Co-Simulator

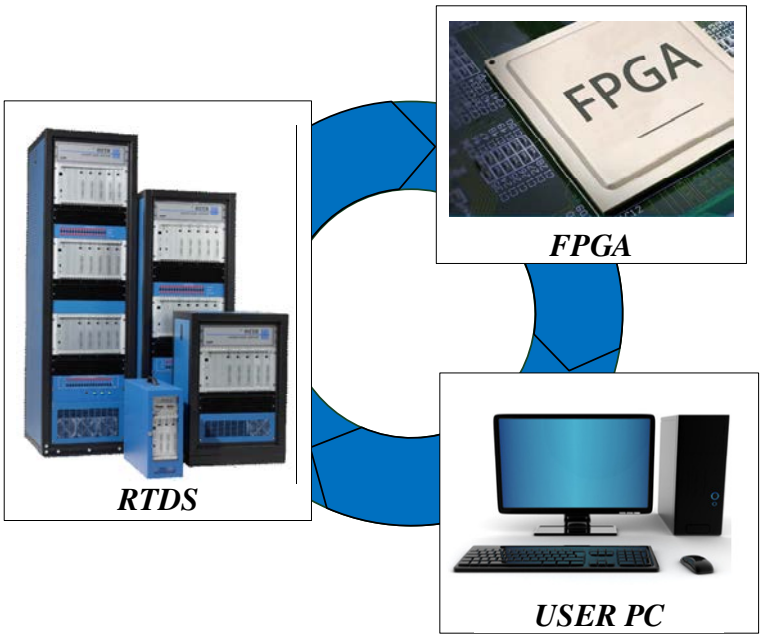


Figure 4-1 Architecture of the proposed FPGA-RTDS co-simulator

To combine the advantages of both RTDS and FPGA, a power system co-simulator is proposed. As shown in Figure 4-1, the proposed FPGA-RTDS co-simulator consists of three main parts: 1) FPGA; 2) RTDS racks and 3) user PC. The FPGA and RTDS carry the main computational burdens, while the user PC is used for results visualization, system control and manipulations (changes of controller parameters, system topologies, etc.).

4.3.1 Study System & External System

The study of large scale power system can be carried out by dividing the system into two subsystems: 1) the study system and 2) external system.

The study system is the part of system that is of research interest. For example it can be an area with multiple HVDC and FACTS devices where the control interactions, harmonic interactions, potential overvoltage/overcurrent problems or performance of protection equipment need to be studied. It can also be an area where the location selection of power system equipment needs to be investigated. For these kinds of analysis and studies, frequent modifications of controller parameters, system topologies and fault cases are normally required.

The external system is the rest part of the system external to the study system. It is normally of much larger size than the study system, and mainly consists of traditional AC components. The topology and parameters of the external system is normally fixed, unlike that of the study system. Traditionally fundamental frequency Thevenin equivalents have been used to simplify the external system. However the simplification neglects the nonlinear, dynamic behaviour of AC system and could cause inaccurate simulation results of the study system.

The study system and external systems are interconnected through one or more transmission lines. As illustrated in Chapter 2, the two systems interconnected by a transmission line are computationally decoupled as long as the wave travelling time of transmission line is equal to or larger than one time-step.

According to the above discussions, the following arrangement has been made for the proposed FPGA-RTDS co-simulator: the study system is simulated in RTDS and the external system is simulated in FPGA. The main reasons for this arrangement are:

1. Better flexibility of RTDS. Compared with FPGA, the GUI of RTDS makes the change of system topology and controller parameters much more convenient than FPGA. This is further improved by the rich library of power system & control components in RTDS. The benefits are significant especially when the objective of simulation is to develop novel topology of devices or to tune controller parameters. On the other hand, FPGA has relatively long compiling and download time for the modification of system topology.
2. Better results visualization from RTDS. This is particularly advantageous for the study system because the number of outputs for result visualization can be limited given the limited I/O ports of FPGA.
3. Paralleled and pipelined computational algorithm of FPGA. This is very useful for the simulation of large AC systems, taking advantage of the inherent parallelism and decoupling effect of transmission lines. EMT models can therefore be used for the entire external system, and interface errors can be eliminated.
4. Mature and reliable AC component models. EMT models of AC components have been widely used for decades with high accuracy and reliability. It is therefore suitable for implementation in FPGA where system modifications can be time-

consuming. On the other hand, models for power electronic devices are still evolving with changing controllers and configurations. It makes the model implementation in FPGA more difficult.

4.3.2 Interface Design

4.3.2.1 Theoretical Analysis

The study system and external systems are interconnected through one or more transmission lines. In this way, the subsystems at both ends of the transmission lines are computationally decoupled, as long as the wave travelling time is equal to or longer than one simulation time-step.

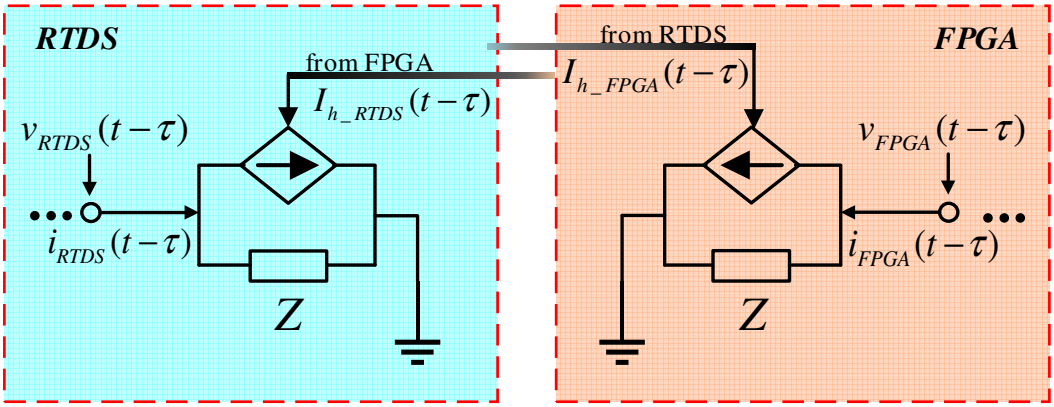


Figure 4-2 Interface between FPGA and RTDS

Figure 4-2 shows the schematic diagram of the interface between FPGA and RTDS, where v_{RTDS} and i_{RTDS} are the terminal voltage and line current at RTDS side; v_{FPGA} and i_{FPGA} are terminal voltage and line current at FPGA side; I_{h_FPGA} and I_{h_RTDS} are the history current terms; τ is the wave travelling time of the transmission line. Transmission line is represented by two Norton equivalent circuits, one for each terminal. As discussed in Section 2.2.2, the

history current terms of the Norton equivalents depend on the voltages and line currents from both ends, i.e.,

$$I_{h\_FPGA}(t-\tau) = f\left(v_{RTDS}(t-\tau), i_{RTDS}(t-\tau)\right) + g\left(v_{FPGA}(t-\tau), i_{FPGA}(t-\tau)\right) \tag{4.1}$$

$$I_{h\_RTDS}(t-\tau) = g\left(v_{RTDS}(t-\tau), i_{RTDS}(t-\tau)\right) + f\left(v_{FPGA}(t-\tau), i_{FPGA}(t-\tau)\right) \tag{4.2}$$

Therefore from the point of view of RTDS, only the voltage and line current from previous time-steps at FPGA end are required for the solution of complete system. Similarly, only the terminal voltage and line current from previous time-steps at RTDS end are required for the computation of system at FPGA side. For single-circuit interface lines, a total of 12 single-phase singles are required for communication. The computations for both I_{h_FPGA} and I_{h_RTDS} can be carried out in FPGA or RTDS, and the calculated results need to be sent to the other end. Alternatively they can be calculated in a distributed manner at their respective ends.

It is important to mention that the implementation of Norton equivalent at RTDS side does not require access to the low level codes of RTDS, as its impedance is purely resistive and no "history terms" are required. The history current of Norton equivalent can be implemented using controlled current sources, and equivalent parallel impedance can be implemented using resistors. To account for the mutual coupling between phases, mutually-connected resistors can be used.

4.3.2.2 Interface Implementation

There are a number of ways to physically implement the interface between FPGA and RTDS. The first option is direct fibre connections. It can be used to directly connect I/O ports of

FPGA with the processor cards of RTDS. If more communication channels are required, FPGA extension boards can be utilized. The second option is to use I/O cards from RTDS. The Gigabit Transceiver Digital Input/Output Card (GTDI/GTDO) can be used to interface digital signals from FPGA to the RTDS. In this case, the data is exchanged between the I/O card and processor card through fibre connections.

4.3.2.3 Unit Conversions

Another important aspect in the interface design is the unit conversion. In the FPGA part of the co-simulator, all calculations are carried out using per unit values for simplicity and accuracy. On the RTDS side, the output variables for interface are all based on actual values although the calculations are done in per unit. For example, the unit of kV and kA are used as default for the voltages and currents at RTDS side. The output of a bus voltage of 230kV is a scalar value of 230, while on the FPGA side the output of the same voltage would be 1 p.u. if the base voltage is 230kV. Therefore a unit converter module needs to be designed when designing the interface.

The following manipulations of the interface signals are required to achieve the unit conversions:

- For signals from FPGA to RTDS, they are multiplied by the corresponding base values and then divided by 1000 to get the scalar values under kV/kA.
- For signals from RTDS to FPGA, they are multiplied by 1000 and then divided by the base value.

The above calculations can be carried out on either FPGA or RTDS side. In this thesis, all the conversions are calculated at FPGA side using the floating-point multiplier.

4.3.2.4 Output from FPGA

The simulation results from FPGA can be exported in various ways. The data can be exported to PC through USB cable or RS232. It can also be displayed on the oscilloscope using D/A converter.

For the proposed co-simulator, FPGA output signals are exported to RTDS and visualized in RSCAD. On one hand, the GUI of RSCAD can be used for better manipulation of the output signals. On the other hand, it would be more convenient when simulation results from FPGA and RTDS can both be accessed through RSCAD

Due to the limited connection bandwidth, the 32-bit single-precision floating-point outputs from the FPGA are in serial in each simulation time-step. So the number of outputs that can be updated in each time-step cannot be too large, otherwise it will affect the speed of simulation. Considering both of the simulation speed and the number of output signals, the maximum number of signals that can be transmitted between RSCAD and FPGA in each time-step is set to be 64. For large scale power system simulation, there may be more than 64 signals need to be exported. To mitigate this problem, a number of output channels are specifically designed. All the output signals are divided into groups. Each group has 64 signals and is assigned to one output channel. User can select the channel, i.e. group of outputs, that needs to be visualized or monitored. It should be mentioned that the switching of output channel can be carried out in real-time while the simulation is running.

4.3.3 Parallel Computation Schemes

For the hardware implementation using FPGA, hardware modules introduced in Chapter 2 & 3 have been designed including most of the commonly used power system components.

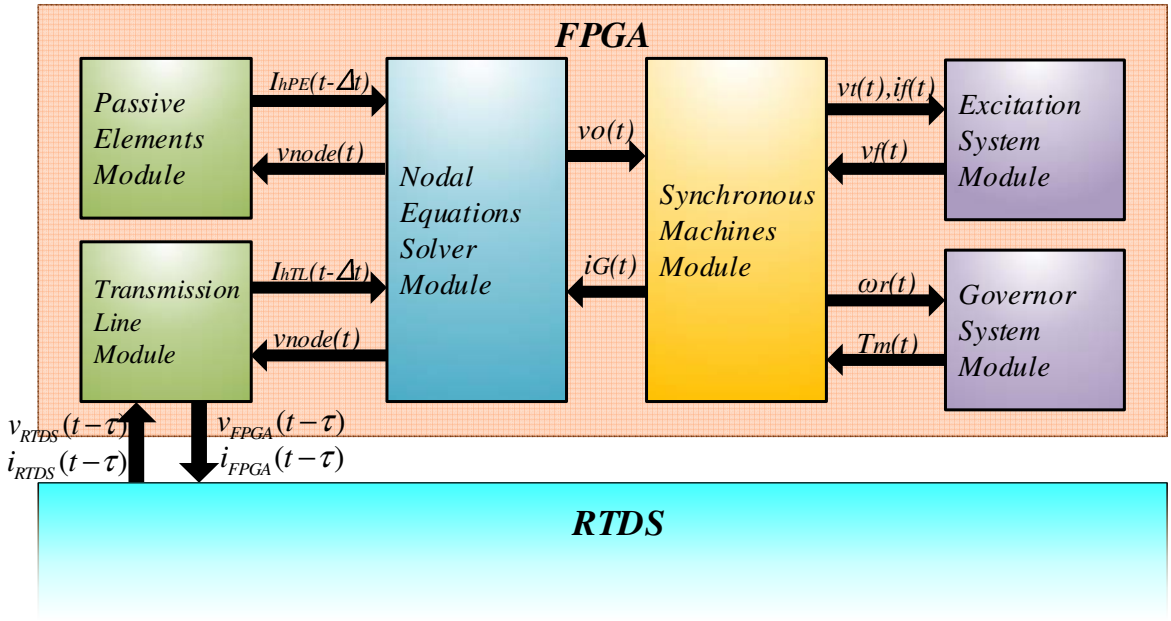


Figure 4-3 Hardware modules for FPGA implementation

The interconnections between these modules and the interface with RTDS are illustrated in Figure 4-3. It can be seen form the figure that each module is made responsible for the calculation of one type of power system/control element to take advantage of the inherent parallelism between different hardware modules.

At the same time, deeply pipelined calculation algorithms have been designed within each module itself. So the calculations of all the elements of the same type that are sharing the same hardware module are carried out in a pipelined manner. In some cases more than one hardware modules may be needed for the one type of element when the number of elements is large or the computation is complex for that particular type of element. This way of design effectively avoids the potential problem of computation delay caused by one excessively complex module.

Considering the inherent parallelism between modules and the pipelined computation within each module, 7 steps of calculations are designed within each time-step as shown in Figure

4-4. It can be seen that more than one hardware modules are processed in parallel in most of the steps.

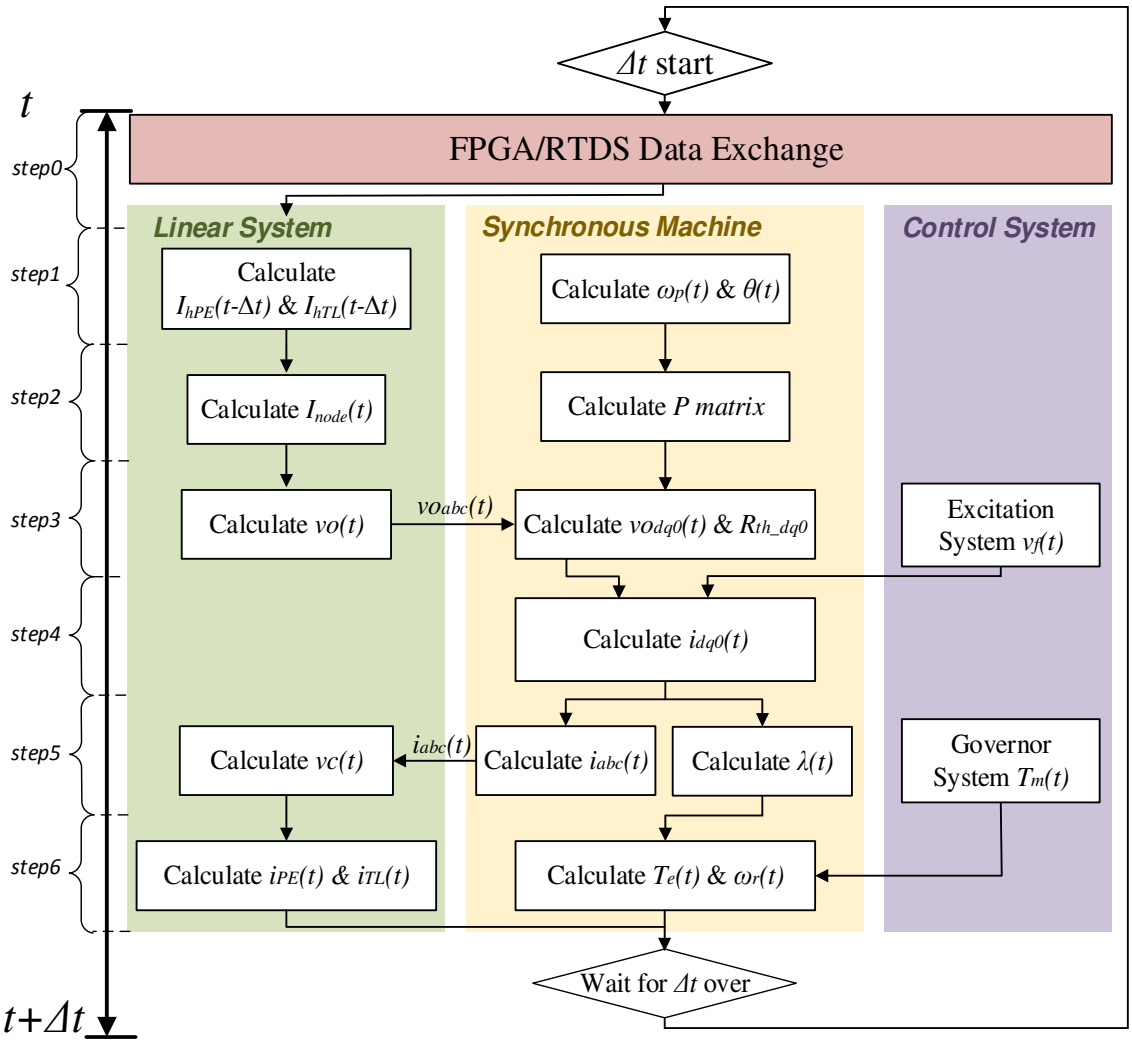


Figure 4-4 Computation scheme of the co-simulator

Step 0: The terminal voltages and currents of the interface transmission line are exchanged between FPGA and RTDS.

Step 1: The calculation of history currents for passive elements and transmission lines and the prediction of rotor speed and angle are carried out in parallel.

Step 2: Nodal current injections (I_{node}) and the Park transformation matrix are calculated in parallel using the history currents and the predicted rotor angle from step 1.

Step 3: Using the nodal current injections from step 2, the equivalent system open-circuit voltage v_o for each synchronous machine and its value in *dq0* reference frame, i.e. v_{dq0} are calculated. At the same time, the inverse of Park transformation matrix and the excitation voltage v_f for each synchronous generator are calculated in parallel.

Step 4: This step is to calculate the stator currents in *dq0* reference frame, i.e. i_{dq0} for each synchronous machine. The values of v_{dq0}, v_f, and predicted rotor speed ω_p from previous time-steps are required as inputs.

Step 5: The calculated i_{dq0} from step 4 is taken as input for the calculation of flux linkages, and stator current in *abc* reference frame (i_{abc}) for each synchronous machine. Then the system nodal voltages are calculated using the values of i_{abc}.

Step 6: The history currents are updated in this step. Also the rotor speed of each synchronous machine is calculated using the flux linkage and mechanical torque input from the governor system. These values will be used in step 1 of next time-step.

Upon completion of step 6, the simulator will wait until the end of current time-step before starting the next step 0.

4.3.4 Platform Expandability

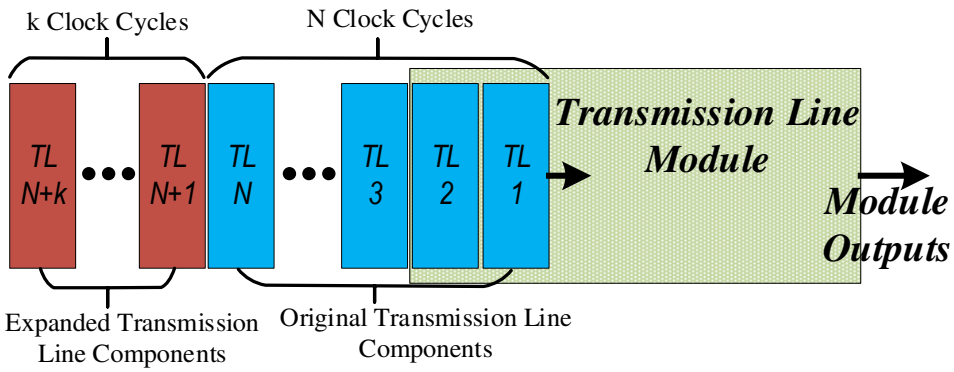


Figure 4-5 Network expansion of transmission line components

The proposed co-simulator has been designed so that the simulated AC system in FPGA can be conveniently expanded without significant sacrifice of hardware resources and computational efficiency. Therefore each of the hardware modules has been designed as a fully self-contained module with dedicated interface signals for interconnection with other modules. In addition, the pipelined computation schemes of elements in each module greatly facilitate the system expansion.

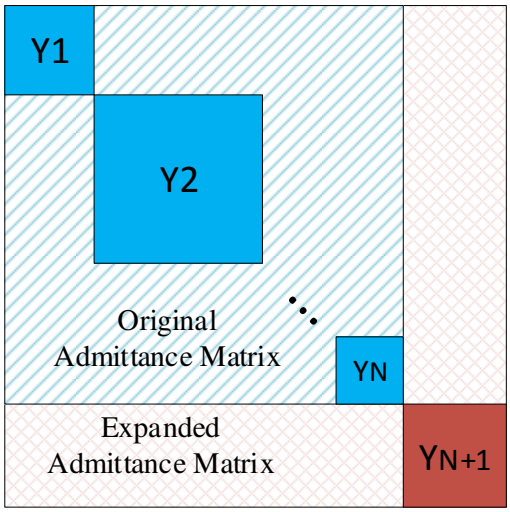


Figure 4-6 Network expansion of admittance matrix

Figure 4-5 and Figure 4-6 illustrate how system expansion can be carried out with the modular designs. Take transmission line module as an example, Figure 4-5 schematically illustrates how the additional components can be accommodated in the designed hardware modules. In the figure, TL1-TLN are N transmission line components that have been simulated in the original system, and suppose that additional k transmission line components need to be added. The computations of these additional transmission lines can be pipelined with the existing ones so the increase of computational time of this hardware module is k FPGA clock cycles. Due to the high clock frequency of FPGA, the corresponding increase of time is very small. Similar methods are applied to other modules. At the same time the system admittance matrix is to be expanded as shown in Figure 4-6. Due to the decoupling effect of transmission lines, the expanded part of the system (with system matrix YN+1) can be solved independently from and in parallel with existing networks.

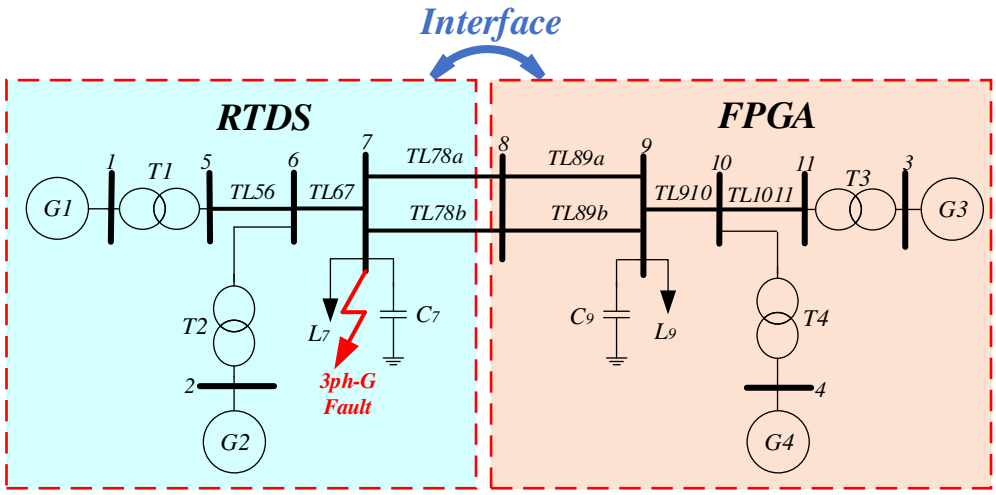
Another important aspect with system expansion is its effect on the minimum simulation time-step. The proposed expansion method can effectively minimize the increase of the minimum simulation time-step. If the method of pipelining to accommodate more components does not lead to longer solution time than the hardware modules that are processed in parallel, its impact on the overall simulation time-step is minimal. On the contrary, if the network expansion is significant and results in an excessive amount of one particular type of component, additional hardware modules may need to be added. This is to balance the computational burden of each hardware module and achieve a smaller simulation time-step. A more detailed discussion on this aspect will be provided in the next section based on two case studies.

4.4 Case Study

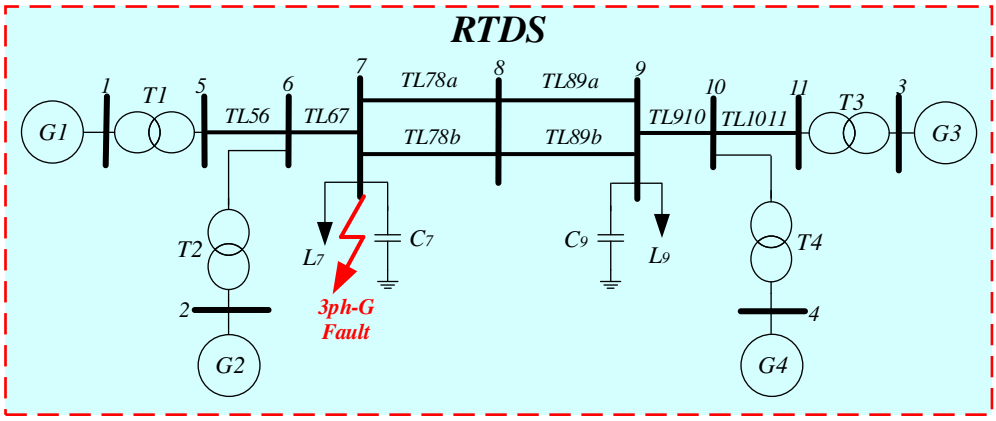
Two case studies are presented in this section. The first case simulates a two-area four-machine power system with one area simulated in FPGA and the other area in RTDS. Comparisons are made with the case where the complete system is simulated in RTDS. The second case simulates a system of 141 buses in FPGA to demonstrate the simulator's capability in simulating large power systems. The achieved minimum simulation time-steps are 6μs for Case 1 and Case 2, 21μs for Case 3. Detailed system data can be found in Appendix B and Appendix C.

4.4.1 Case 1

Figure 4-7 illustrates the network configuration for Case 1. It can be seen in Figure 4-7(a) that the system is divided into two subsystems when simulating using the proposed co-simulator. The area with generator G1 and generator G2 are simulated using RTDS, while the other area with G3 and G4 are simulated using FPGA. The two subsystems are decoupled and



(a) Using the proposed co-simulator

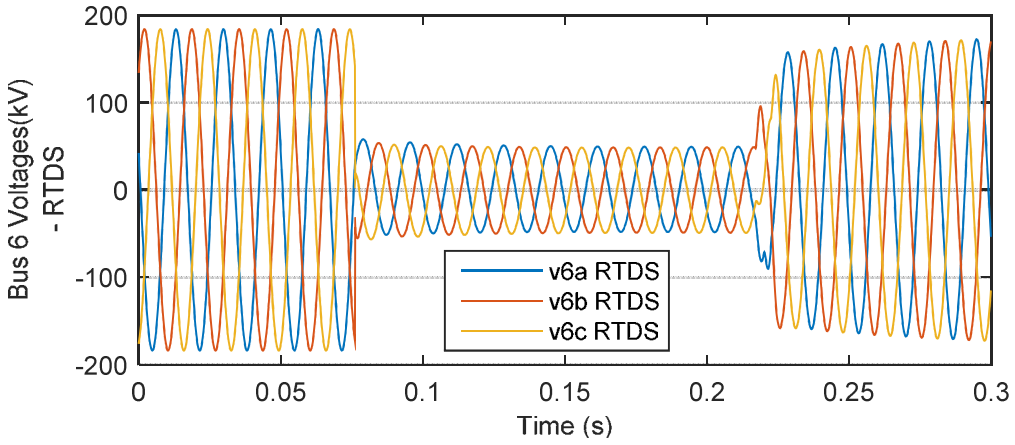


(b) Using RTDS only

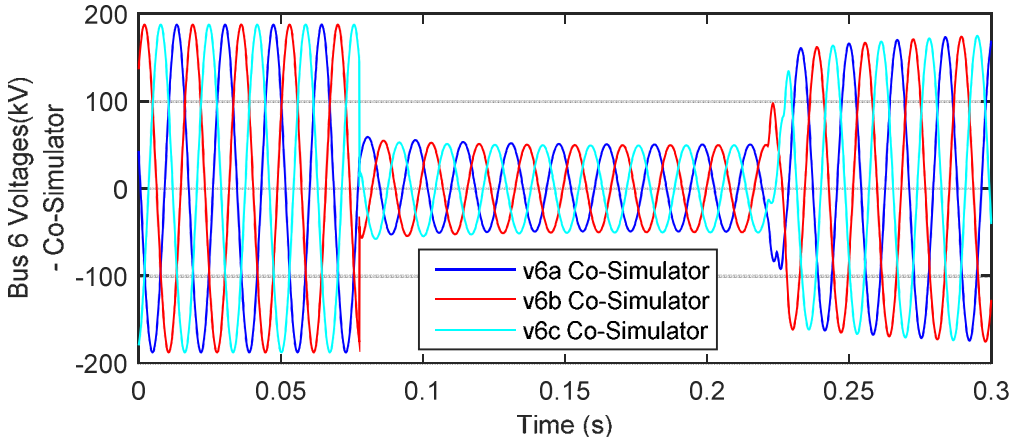
Figure 4-7 Simulations of two-area four-machine system

interfaced by the transmission line TL78. As comparison, the same system is simulated in RTDS as illustrated in Figure 4-7(b). The fault to be simulated is a three-phase fault located as bus 7. The fault duration is 150ms.

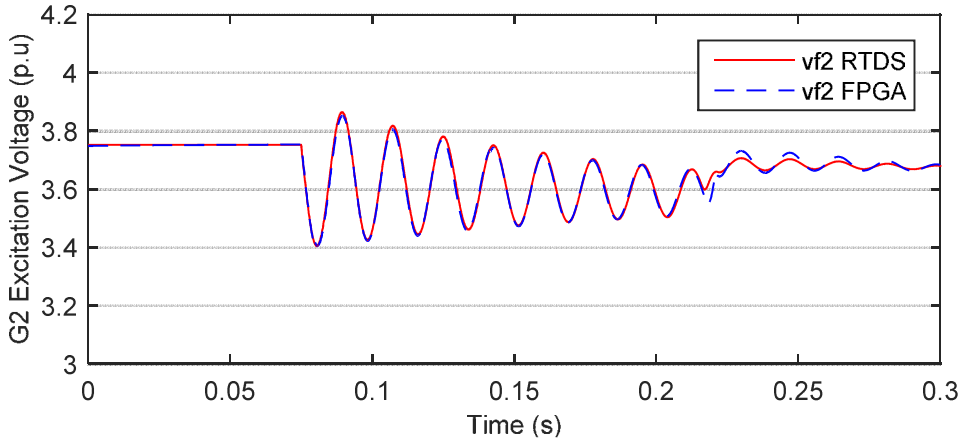
Figure 4-8 shows the simulation results. Figure 4-8(a) and Figure 4-8(b) show the three-phase voltages at bus 6 using the co-simulator and RTDS respectively. It can be seen that once fault happens, the voltages are dropped and detailed agreement can be observed. Figure 4-8(c) shows the comparison of excitation voltages of G2. It can be seen that the dynamic behaviours simulated using the proposed simulator correctly captures those simulated by RTDS. Figure 4-8(d) and Figure 4-8(e) illustrates the transmission line currents between bus 6 and bus 7. It can be seen from the figures that both the transient increase of line current and the unbalance between three-phases are accurately simulated using the proposed co-simulator. Figure 4-8(f) and Figure 4-8(g) show the comparisons of electrical torques of G3 and G4. The high-frequency dynamics of electrical torques during and after fault are correctly simulated. Figure 4-8(h) and Figure 4-8(i) compare the results of line currents of TL78 between the co-simulator and RTDS. Again detailed agreements have been achieved.



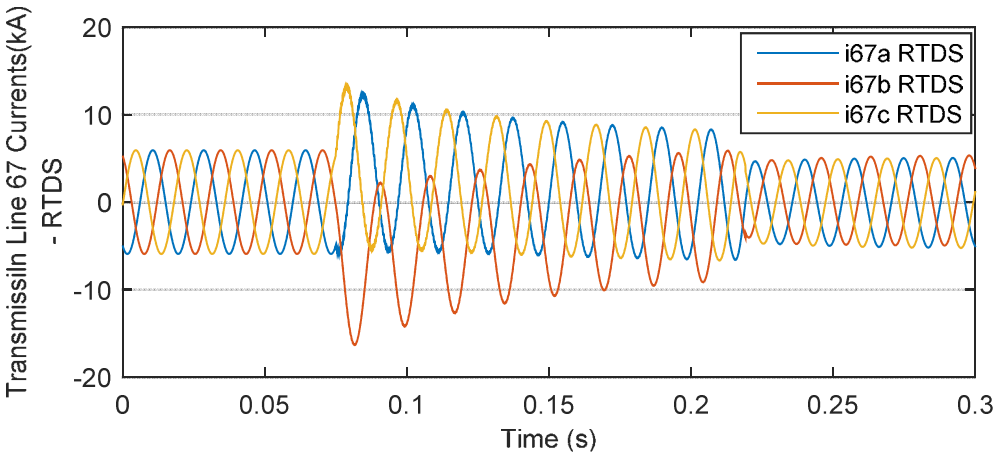
(a) Bus 6 voltages from RTDS



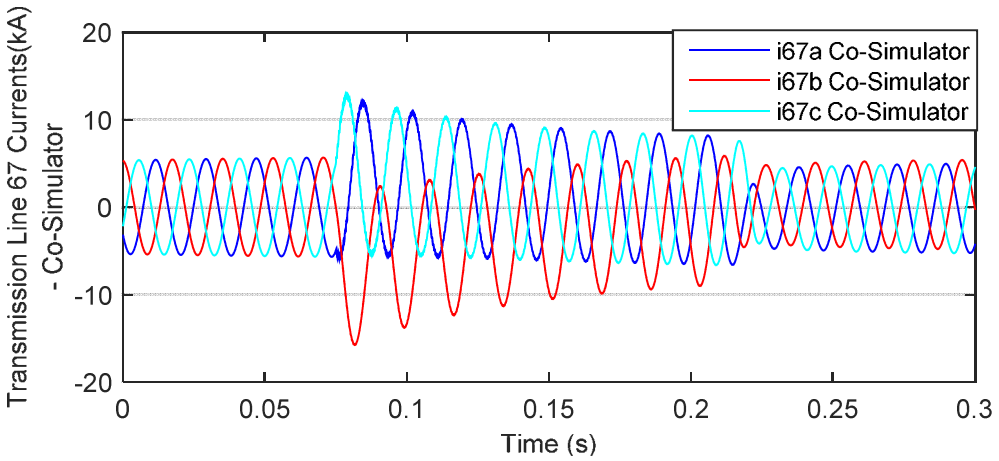
(b) Bus 6 voltages from co-simulator



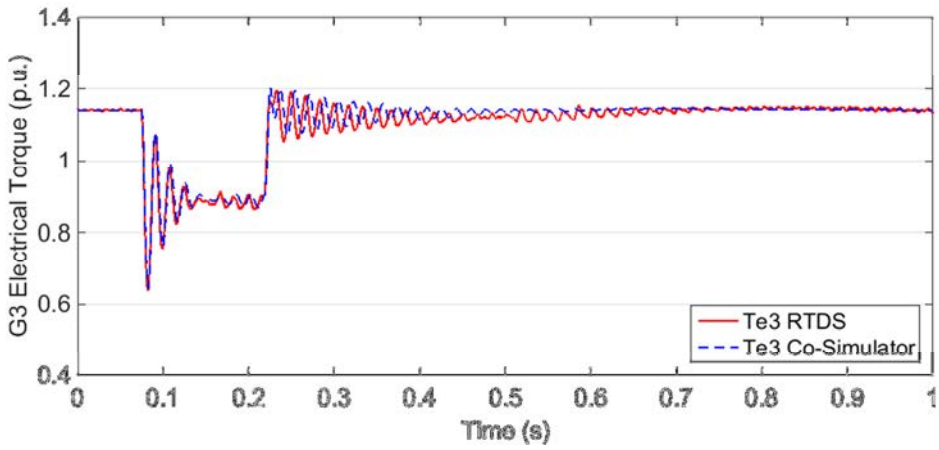
(c) Excitation voltage of G2 from RTDS and co-simulator



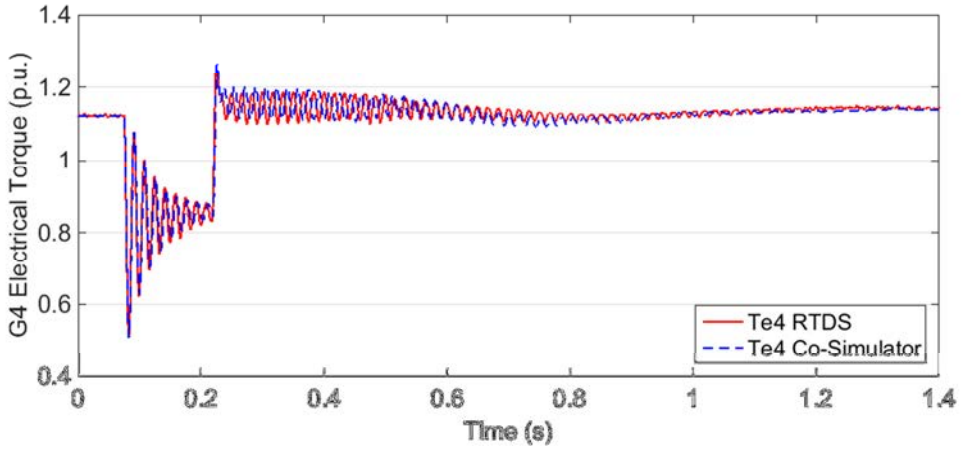
(d) Three-phase line currents on TL67 from RTDS



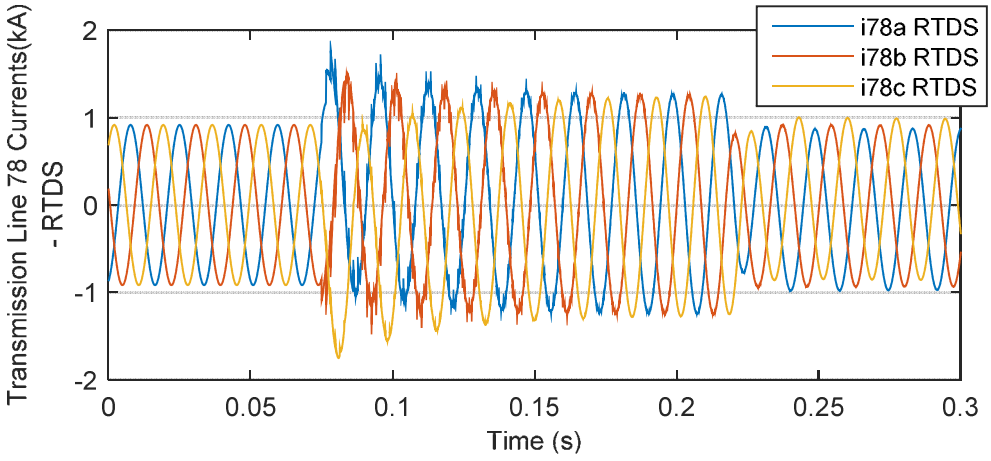
(e) Three-phase line currents on TL67 from co-simulator



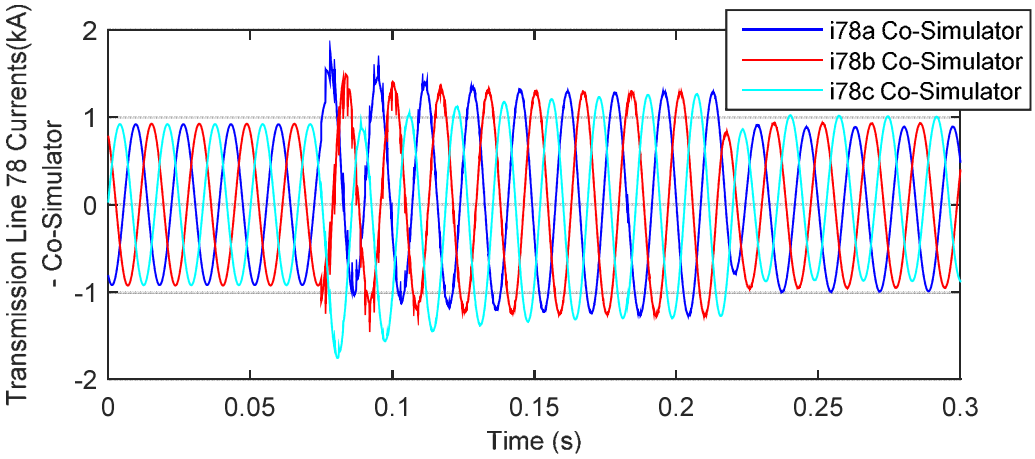
(f) Electrical torques of G3 from RTDS and the co-simulator



(g) Electrical torques of G4 from RTDS and the co-simulator



(h) Three-phase line currents on TL78a from RTDS



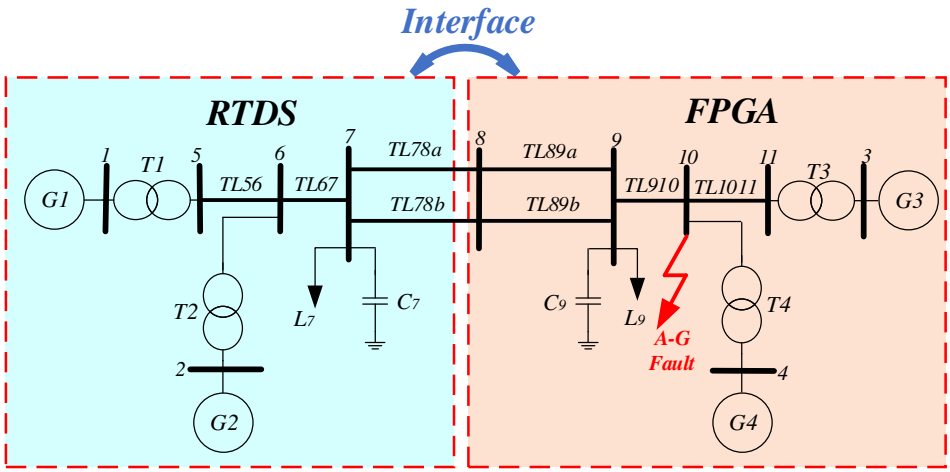
(i) Three-phase line currents on TL78a from the co-simulator

Figure 4-8 Simulation results for Case 1

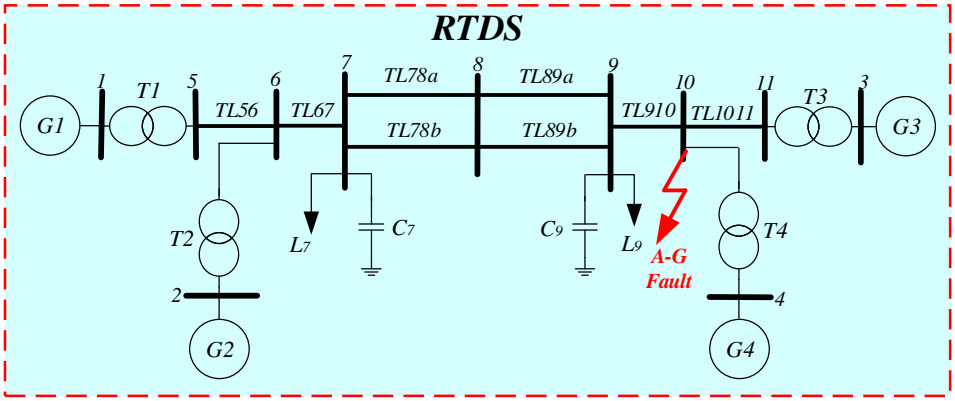
Some discrepancies can be observed between the simulation results from proposed co-simulator and those from RTDS (Figure 4-8). There are two potential sources of error:

1. Calculation of control systems. In the proposed co-simulator, the control systems are using the measured signals from previous time-steps rather than current time-step. In RTDS, the detailed implementations of control systems are not revealed.
2. Accuracy of number format. Single-floating point numbers are used for the calculations in FPGA. Double-floating point numbers are used for the calculations in RTDS.

4.4.2 Case 2



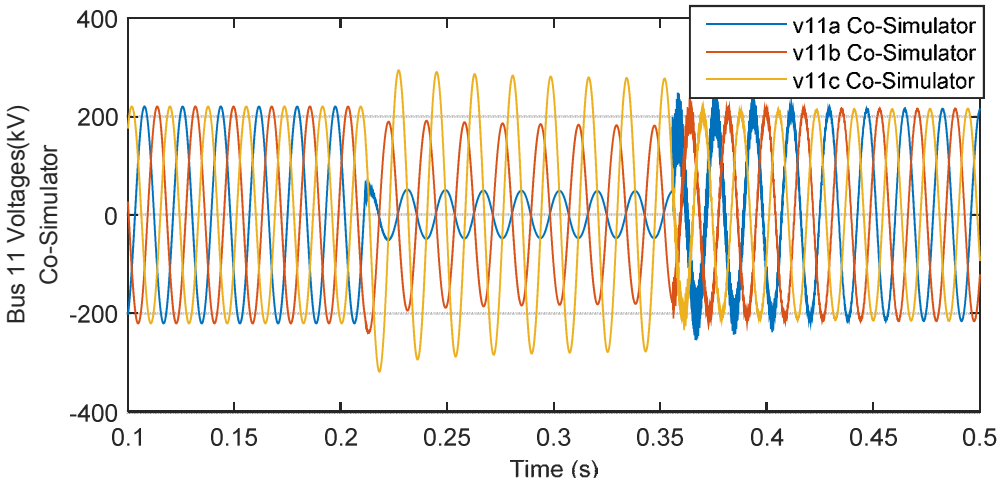
(a) Using the proposed co-simulator



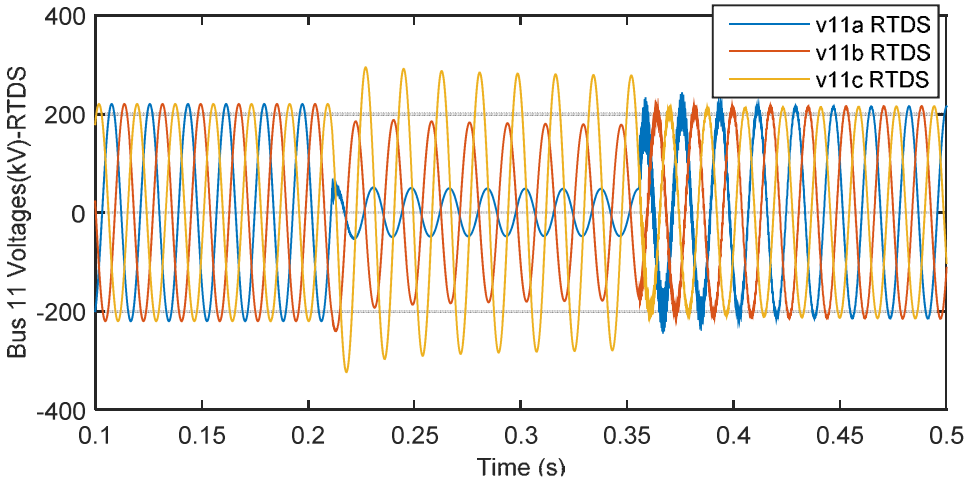
(b) Using RTDS only

Figure 4-9 Simulations of two-area four-machine system for case 2

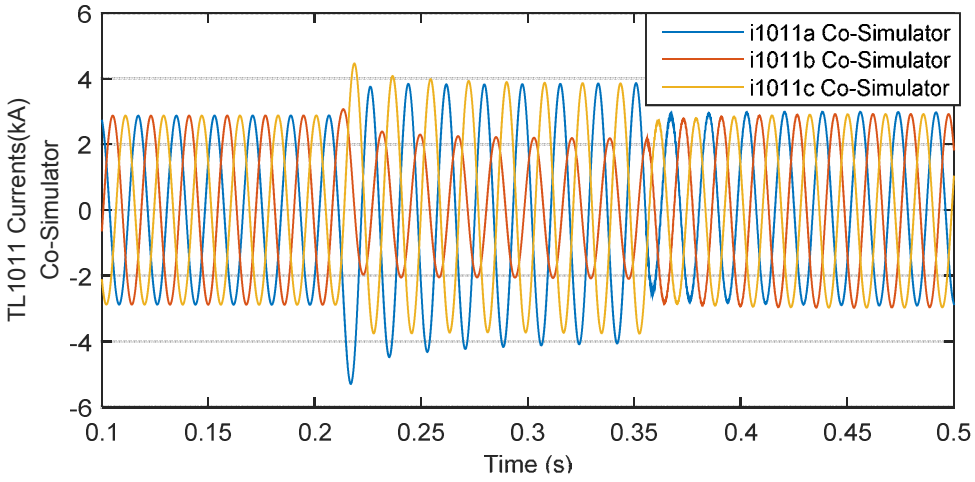
As shown in Figure 4-9(a), an unbalanced single-phase (phase A) to ground fault at bus 10 is simulated using the proposed co-simulator. For the purpose of comparison, the same system is simulated in RTDS as illustrated in Figure 4-9(b). The fault duration is 150ms, and the simulation results are shown in Figure 4-10.



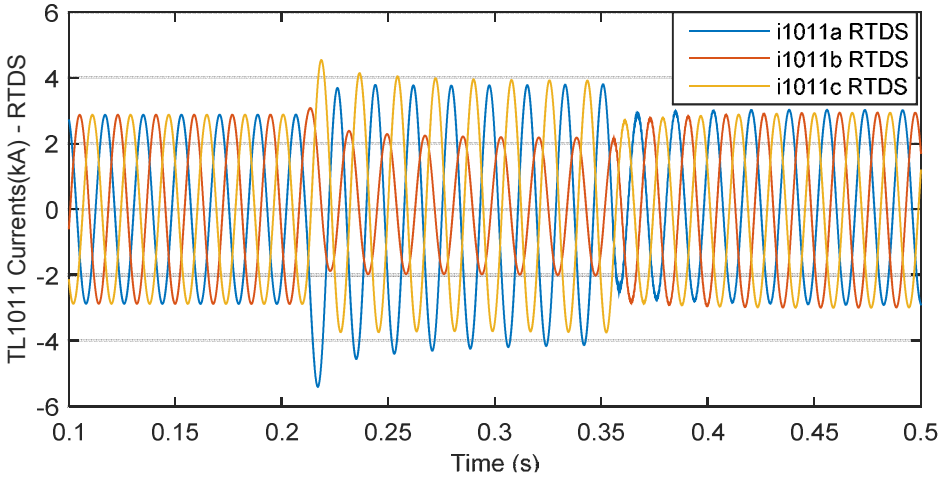
(a) Bus 11 voltages from the co-simulator



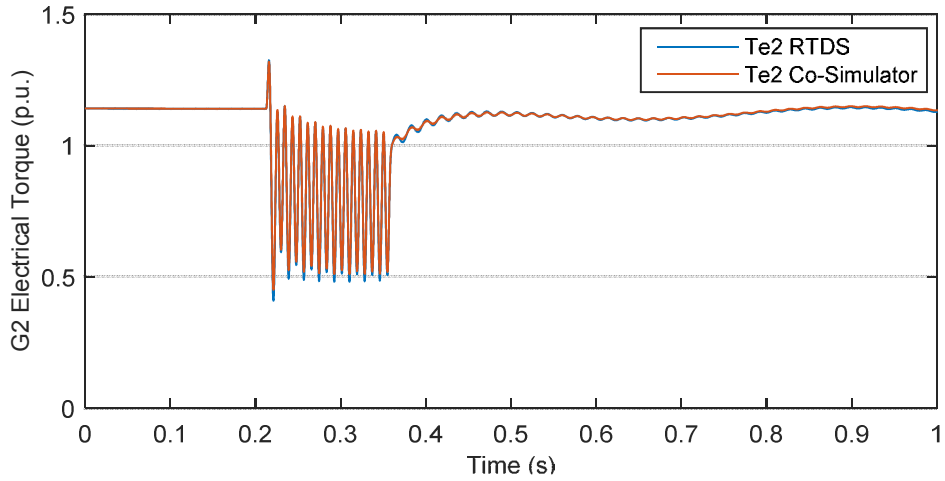
(b) Bus 11 voltages from RTDS



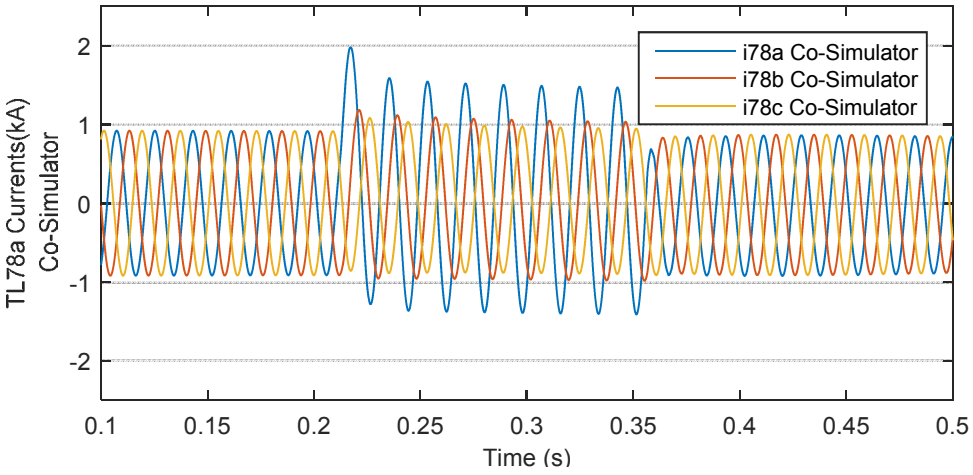
(c) Three-phase line currents on TL1011 from co-simulator



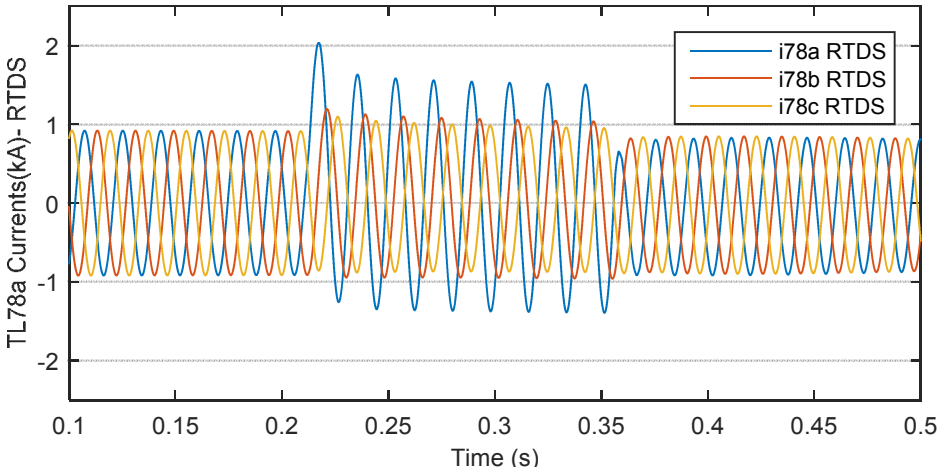
(d) Three-phase line currents on TL1011 from RTDS



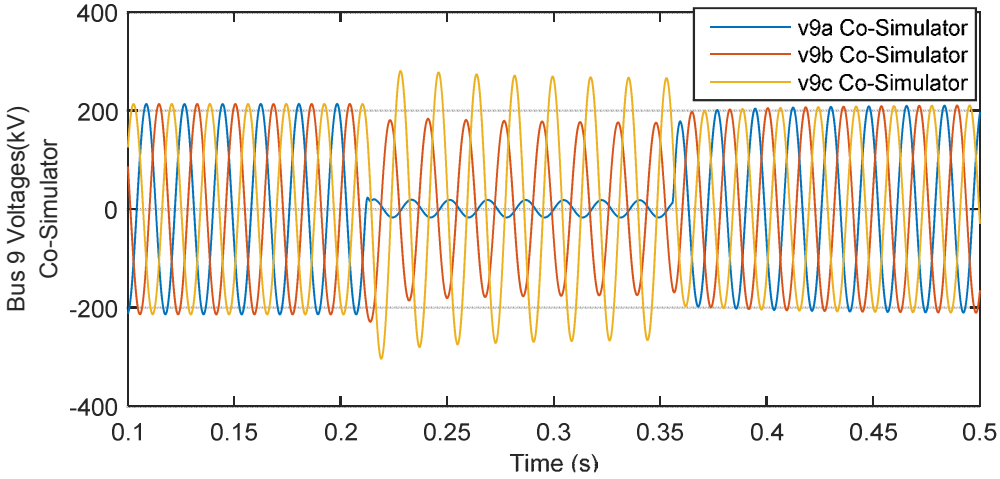
(e) Electrical torques of G2 from RTDS and the co-simulator



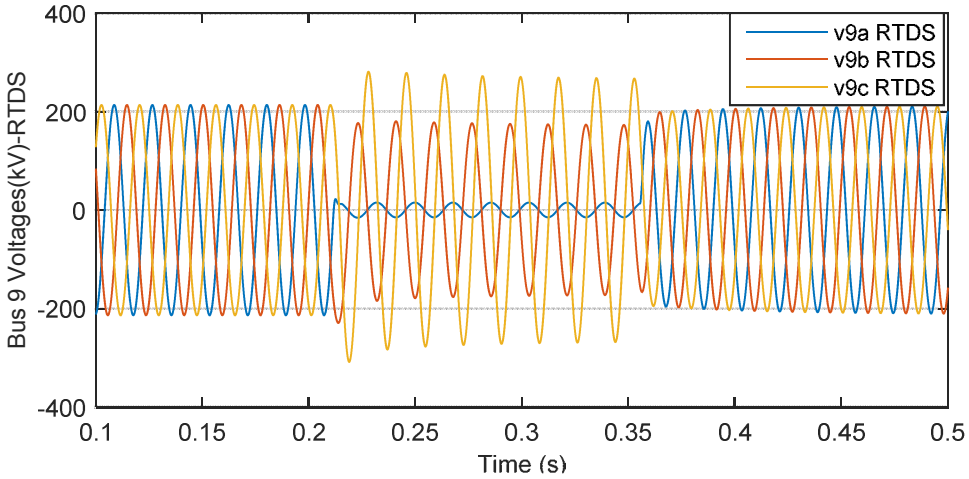
(f) Three-phase line currents on TL78a from the co-simulator



(g) Three-phase line currents on TL78a from RTDS



(h) Bus 9 voltages from the co-simulator



(i) Bus 9 voltages from RTDS

Figure 4-10 Simulation results for Case 2

Figure 4-10(a) and Figure 4-10(b) show the three-phase voltages at bus 11 from the co-simulator and RTDS respectively. It can be seen that when fault happens, the voltage of phase A is dropped. Figure 4-10(c) and Figure 4-10(d) illustrate the transmission line currents between bus 10 and bus 11. Detailed agreement between results from the co-simulator and RTDS can be observed. Figure 4-10(e) shows the comparison of electrical torques of G2. The high-frequency dynamics during fault are accurately simulated (slight differences can be observed and potential causes of the error have been discussed in Case 1). Figure 4-10(f) and Figure 4-10(g) show the transmission line currents of TL78a from the co-simulator and RTDS respectively. The increase of line current and the unbalance between three-phases due to the fault are accurately simulated using the proposed co-simulator. Figure 4-10(h) and Figure 4-10(i) compare the results of bus 9 voltages between the co-simulator and RTDS. Again detailed agreements have been achieved.

4.4.3 Case 3

To demonstrate the capability of simulating large AC networks using the proposed co-simulator, a system with 141 three-phase buses are developed as shown in Figure 4-11. Five

buses and two machines (G37 and G38) are simulated in RTDS, and the rest of the network is simulated in FPGA. The two subsystems in RTDS and FPGA are interfaced by transmission line *TL3A* and *TL3B*. The simulated fault is a 100ms three-phase fault located at bus 141.

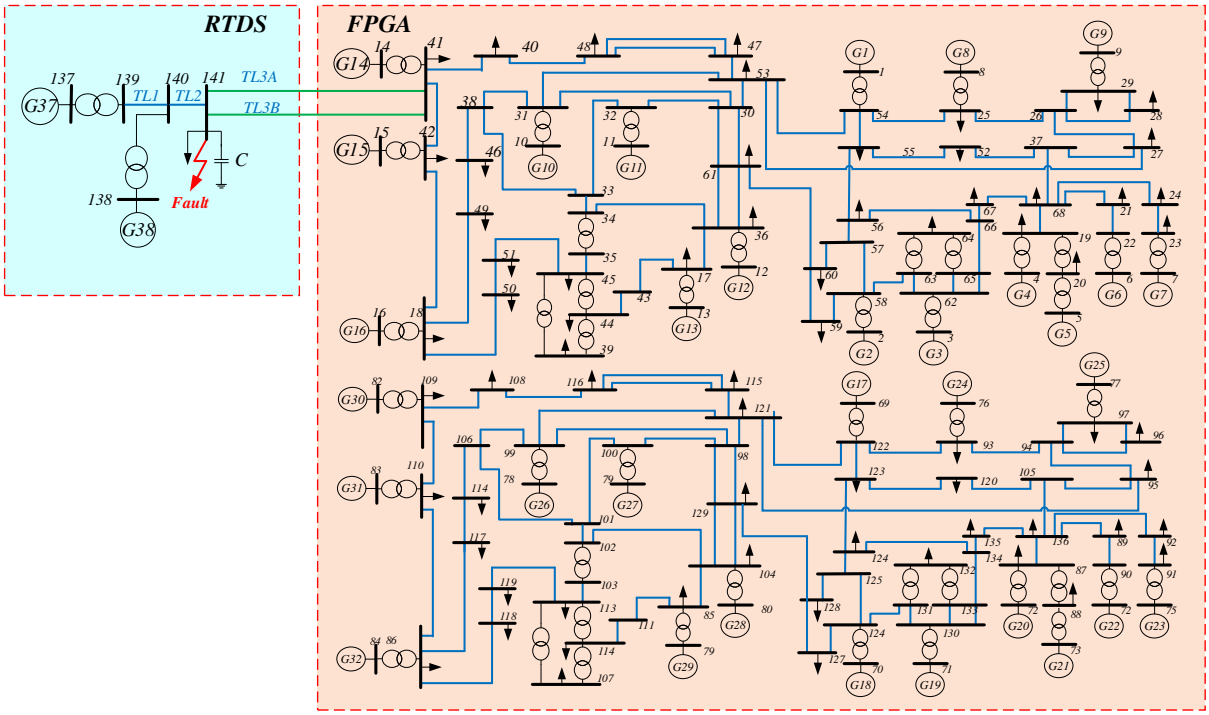
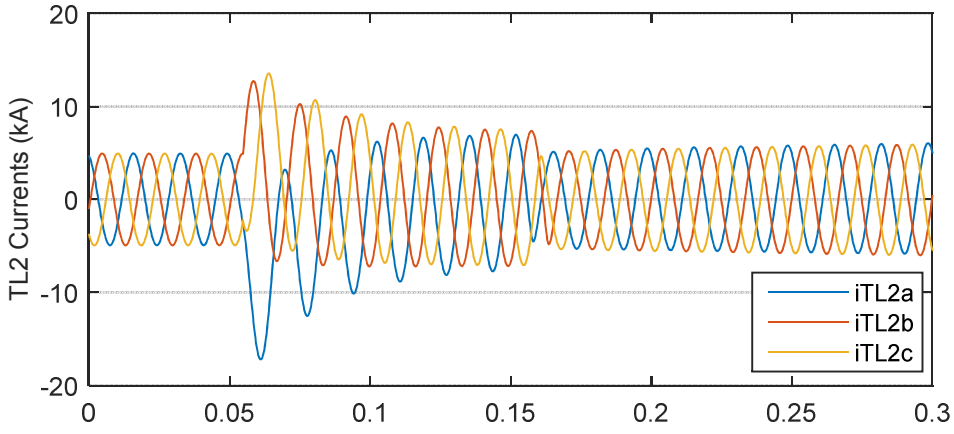
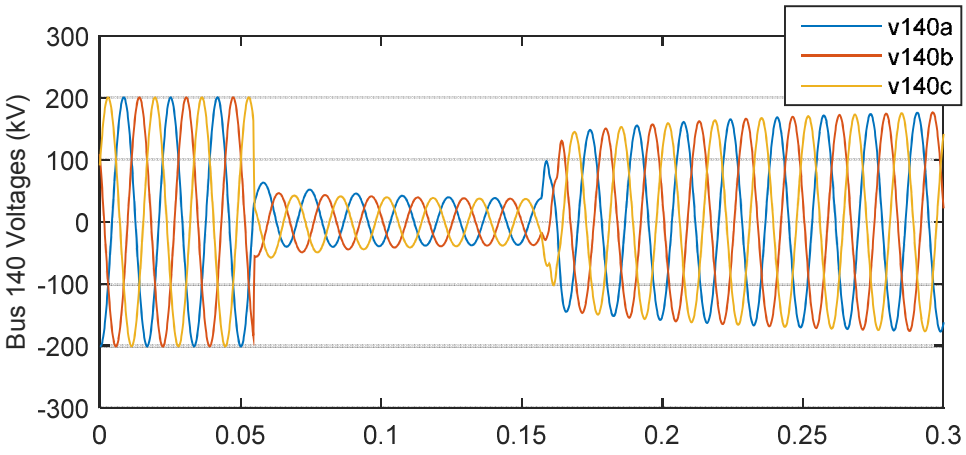


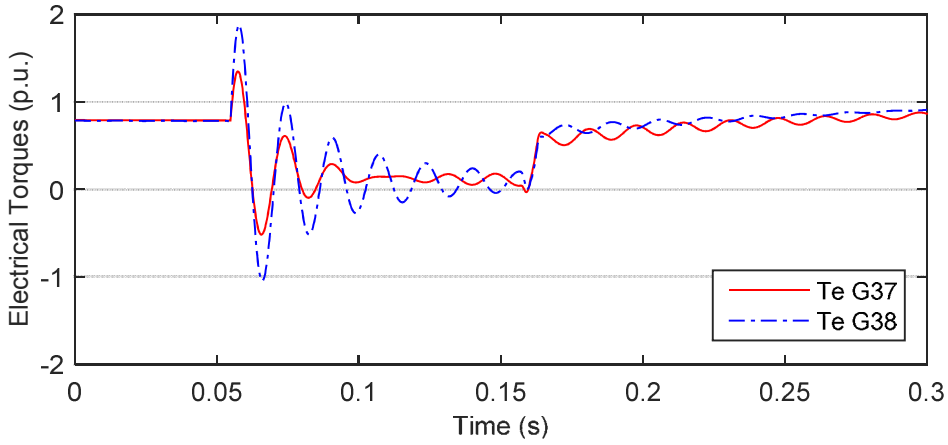
Figure 4-11 Simulation of a 141-bus system using the proposed co-simulator



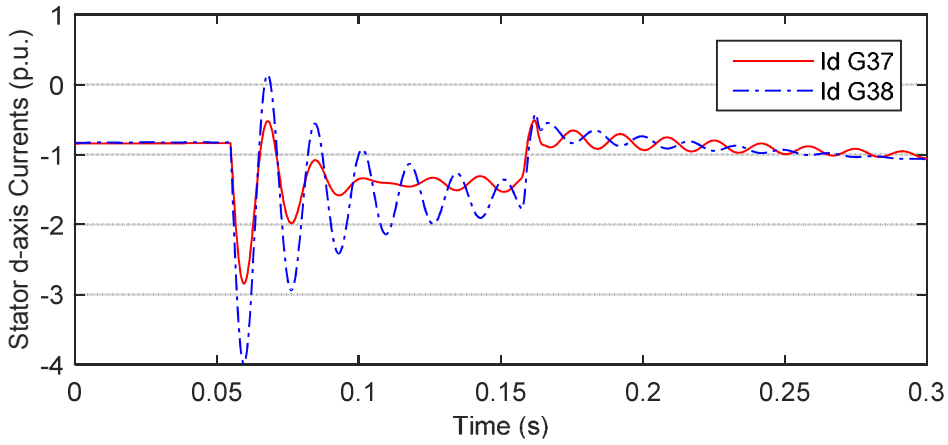
(a) Three-phase currents on *TL2*



(b) Three-phase voltages at bus 140

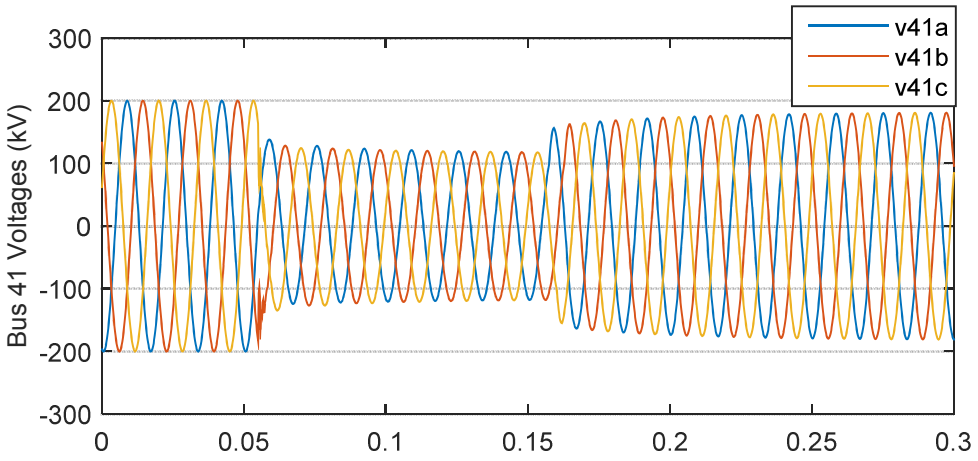


(c) Electrical torques of *G37* and *G38*

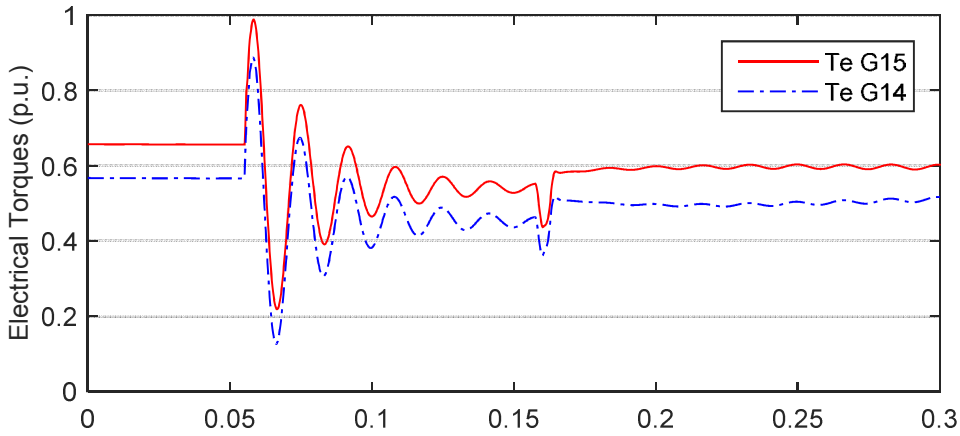


(d) Stator *d*-axis current of G37 and G38

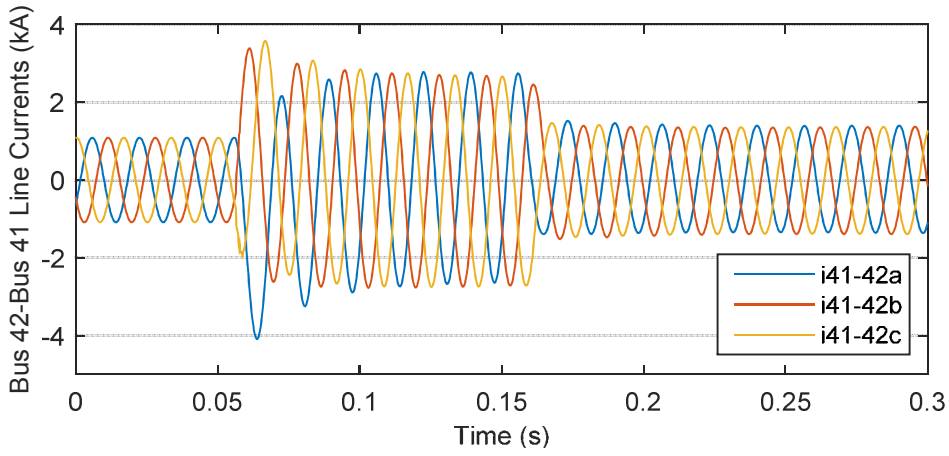
Figure 4-12 Case 3 simulation results from RTDS



(a) Three-phase voltages at bus 41



(b) Electrical torques of *G14* and *G15*



(c) Three-phase line currents between bus 42 and bus 41

Figure 4-13 Case 3 simulation results from FPGA

Figure 4-12 and Figure 4-13 show the simulation results from the proposed co-simulator. It can be seen that once fault happens at bus 141, the voltages at bus 140 (Figure 4-12(b)) at RTDS side and bus 41 (Figure 4-13(a)) at FPGA side are decreased. The three-phase currents on the connected transmission lines, i.e., *TL2* at RTDS side and the line between bus 42 and bus 41 at FPGA side are increased as shown in Figure 4-12(a) and Figure 4-13(c). The electrical torques of G37 and G38 at RTDS side are shown in Figure 4-12(c). Higher oscillations are observed in electrical torques of G38 as it is located electrically closer to the fault. Similar oscillatory responses of the stator *d*-axis current of G37 and G38 are shown in Figure 4-12(d). The electrical torques of G14 and G15 are shown in Figure 4-13(b) where smaller oscillations can be seen.

4.4.4 Discussions

Table 4.1 FPGA Resource Utilization

FPGA Hardware Resource Utilization			
Logic Utilization	**Case1 & 2**	**Case3**	**Available**
Slice Registers	15860(5.3%)	52466(17.4%)	301440
Slice LUTs	30242(20.1%)	110283(73.2%)	150720
Block RAMs	13(3.1%)	45(10.8%)	416
DSP48E1s	87(11.3%)	307(40.0%)	768

To examine the relationship between hardware utilization and the size of AC network, Table 4.1 shows the FPGA resource utilizations for Case 1/Case 2 and Case 3. It can be seen from the table that the utilization rate of each type of hardware resource is increased by a factor of 3.3 – 3.7, while the size of FPGA part of system is increased by more than 20 times. This is mainly due to the expansion method as discussed in Section 4.3.4.

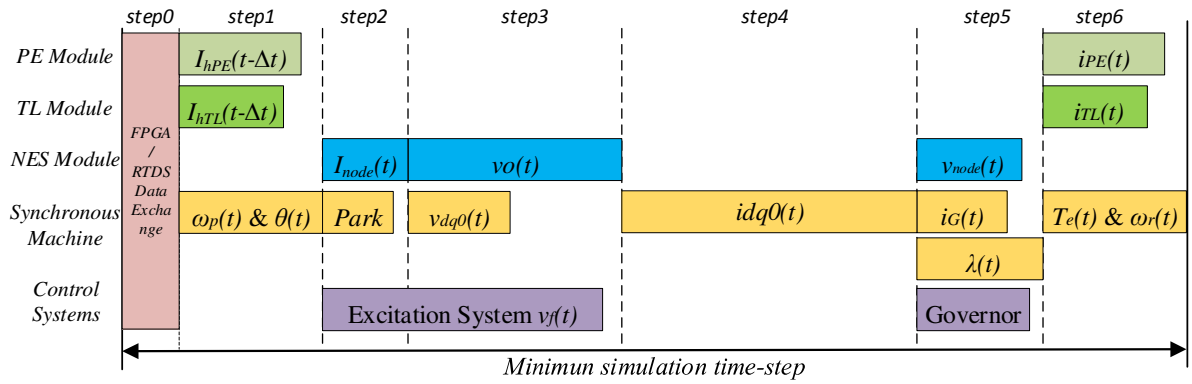


Figure 4-14 Breakdown of minimum simulation time-step of Case 1

To analyse how network expansion affects the minimum simulation time-step, Figure 4-14 shows how the simulation time-step is affected by the processing time of paralleled hardware modules. The horizontal axis represents the minimum simulation step of FPGA, which can be further divided into step 0-6 as discussed in Section 0. As can be seen from the figure, the processing time of each step is determined by the most time-consuming hardware module in that step. For example, the processing time of step 1 is determined by the ω_p & θ module (prediction of rotor speed and angle). Therefore, the increase of calculation time of other modules in the same step, as long as it is still shorter than the longest one, does not lead to an increase of the minimum simulation time-step. For example, in step 1, the processing time of Passive Element, Transmission line and rotor speed & angle prediction modules are 29, 26 and 30 FPGA clock cycles respectively. So up to 4 additional transmission lines can be pipelined into the existing hardware module without affecting the processing time of step 1.

When a large numbers of synchronous generators need to be added, the processing time of all steps are affected. Additional hardware modules may be needed to increase the level of parallelism to maintain the simulation time-step. However considering the achieved minimum simulation time-steps of 6μs for Case 1/Case 2 and 21μs for Case 3, the size of

network expansion that can be accommodated would normally be sufficient before the minimum simulation time-step becomes larger than the typical RTS time-step of around $50\mu s$.

4.5 Summary

This chapter describes the design and implementation of a FPGA-RTDS co-simulator for the real-time simulation of large power systems. In the co-simulator, the FPGA provides an efficient extension to the RTDS, alleviating the limitations on the size of AC system that can be simulated. At the same time, paralleled and pipelined computation of FPGA enables the use of EMT models at FPGA side. Therefore the complete system is simulated using EMT models, and the interface errors existed in hybrid EMT/TS simulation method can be eliminated. To achieve better expandability of the co-simulator, hardware modules have been designed as fully self-contained hardware components for most of the commonly used power system elements. They can be conveniently duplicated when the size of the system expands, and the effect on simulation time-step is minimized. Two case studies have been presented to verify the simulation accuracy and capability of the proposed co-simulator. Discussions have been provided on the aspects of hardware resource utilization, simulation time-step and size of AC system based on the results of two case studies.

Chapter 5

SIMULATION OF LARGE SCALE POWER SYSTEMS USING MULTI-FPGA BASED CO-SIMULATOR

5.1 Introduction

This chapter introduces the multi-FPGA based co-simulator, based on the FPGA-RTDS co-simulator describes in Chapter 4. Section 5.2 explains the necessity and benefits of multi-FPGA architecture. Section 5.3 discusses the issue of network partition. Section 5.4 introduces the architecture and communication interface for the multi-FPGA co-simulator. Section 5.5 discusses the capability of the co-simulator. Section 5.6 verifies the performance of the simulator by simulating a large scale power system with more than 4000 nodes. Section 5.7 summarises this chapter.

5.2 Expansion to Multi-FPGA

A single FPGA would normally be sufficient for the simulation of hundreds of nodes. However, as the size of the simulation system increases, the resources of a single FPGA may be insufficient. Multi-FPGA architecture is introduced for the following reasons:

- Modular design with better expandability and easier network modification. Each FPGA is running more or less independent of other FPGAs so network extension can be conveniently achieved by including additional FPGAs without affecting existing ones. At the same time, minor network modifications only takes place in one of the FPGA rather than the whole system.
- It is economically more advantageous to use multi-FPGA than one single powerful FPGA when simulating large power systems. The cost of FPGA is increased significantly with increasing hardware resources.
- Increased level of parallel computing. Each single FPGA is responsible for the computation of one part of the network, and runs in parallel with the rest of the network. Therefore the more FPGAs are used the increased level of parallelism is achieved. This enables the use of even smaller time-steps especially when fast-switching power electronic devices are to be simulated.
- Increased number of IO ports. The number of IO port for a single FPGA is limited due to packaging technique. During simulation of complex systems, a large number of IO ports are required, so the option of using multi-FPGA effectively solves the problem.

5.3 Methodology of Network Partition

When it comes to multi-FPGA simulation, system partition is inevitable. As the actual allocation of computational tasks directly affects the efficiency and hardware resource required of the simulator, the following aspects should be considered when partitioning the network:

- Try to avoid splitting an independent module into different FPGAs.

- Try to allocate closely-coupled modules into the same or neighbouring FPGAs.
- Try to minimise the data interface between FPGAs.
- Try to balance the computational burden in each FPGA.

It is worth mentioning high-speed transceivers are normally equipped in the latest version of FPGA boards. For example multiple 14.1Gbps high-speed transceivers have been integrated in the Stratix V series FPGA boards. It further facilitates the high-speed communication between multiple FPGAs and lays the foundation of multi-FPGA simulation of large scale power systems.

Once the systems that are to be simulated on FPGAs are determined, the computational works need to be allocated to each FPGA. Two types of allocation structure can be adopted:

1. Allocation according to physical network structure. In this method, one large power system is divided into several subsystems according to its physical structure. Each FPGA is responsible for the simulation of one subsystem.
2. Allocation according to simulation functions. In this method, each FPGA is dedicated to the simulation of one type of element or control systems. For example, the first board can be responsible for the computation of transmission lines, the second board can be responsible for the solving the synchronous machines, the third board can be dedicated to the solution of network equation, etc..

In this thesis, the first allocation method is applied due to the following reasons:

1. Decoupling of the network can be easily achieved due to the large number of transmission lines. Most of the large scale power systems can be divided into subsystems due to their geographical/physical structures. As discussed in section

2.2.2.1, a transmission line with a length of more than 15km (for a simulation time-step of 50μs) is required for decoupled network solutions of systems connected to both ends of the line. Transmission lines with such length are common in high voltage transmission network (for example the high voltage electricity transmission network in UK [153]).

2. Better expandability. The simulation of larger network can be conveniently done by adding additional FPGAs without affecting the existing ones.
3. Less interface communications between FPGAs. This alleviates the potential problem of insufficient IO ports.

A case study will be given in Section 5.5 to demonstrate the process.

5.4 Interface Design

With multi-FPGA structure, the design of communication between different parts greatly affects the efficiency and performance of the proposed co-simulator. It includes communication between different FPGAs, communication between FPGAs and RTDS, and the communication between simulator and user PCs.

The method of interface also plays an important role in the expandability of the multi-FPGA co-simulator. The required level of communication depends on how the system is partitioned. Generally it is desirable that the number of available communication ports can be maximized when allocating the computation works among FPGAs.

There are different options of communication between FPGAs, including direct communication through fibre connections, direct communication through FPGA Mezzanine Card, communications through RTDS interface and indirect communication through

neighbouring FPGAs. For the proposed Multi-FPGA based co-simulator, the methods of direct interface through FPGA Mezzanine Card expanded I/O ports and interconnection through RTDS are available.

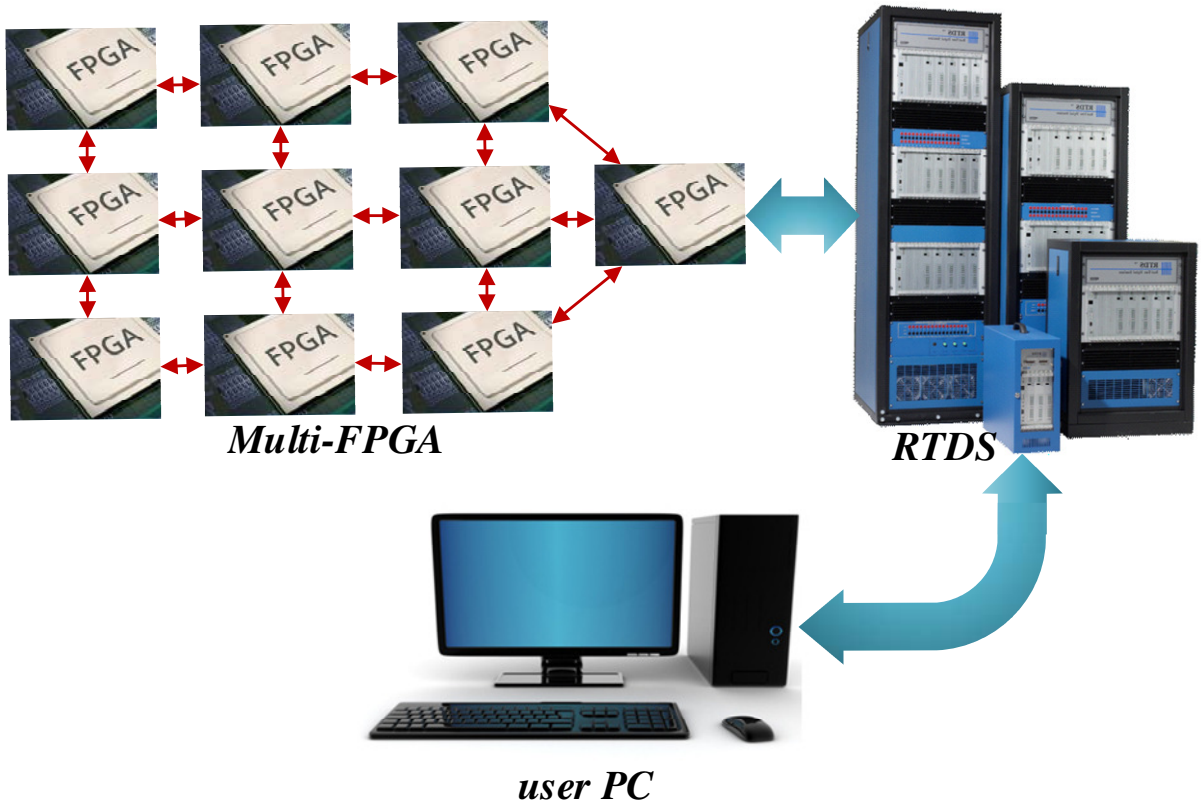


Figure 5-1 Direct communication between FPGAs

The direct connection between FPGAs is straightforward. In general, this method provides better interface speed and bandwidth but is less flexible. As shown in Figure 5-1, red arrows between FPGAs represent the fibre connections. Each FPGA is connected to its neighbouring boards and usually one FPGA is connected with RTDS. The specific way of connections is determined by the simulation network and the way it is partitioned. User can access the simulation results from both RTDS and FPGAs using user PC. The control and manipulation of the simulator are also made from user PC.

The main advantage of this this method is that the interface signals are sent directly to the neighbouring FPGAs though fibres, thereby minimizing the communication delay. The potential drawback of this arrangement is that the hardware interconnections cannot be modified during simulations. Therefore, it is suitable for the cases where no frequent manipulation is required for the topology of external system and the interface transmission lines between FPGAs are relatively short.

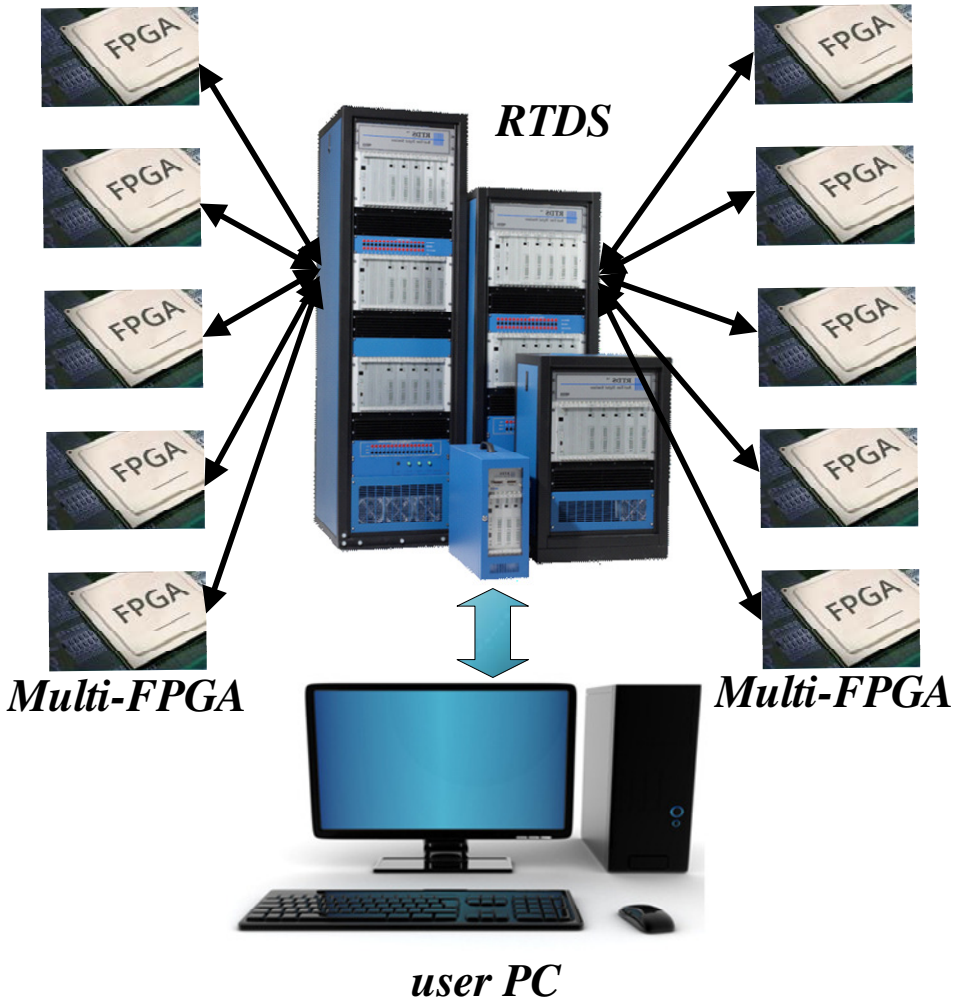


Figure 5-2 Indirect communication through RTDS

The second option is to connect all FPGAs with the RTDS as shown in Figure 5-2. Under this arrangement, the communications between FPGAs are achieved through RTDS, and no

interconnections between FPGAs are required. The implementation of the interface between FPGA and RTDS is described in Section 0. The advantage of this method is that the way of interconnection between FPGAs can be conveniently modified in RTDS and all the interface signals can be accessed from RTDS. The potential disadvantage of this method is that additional simulation time-step is required for the interface between FPGAs, and therefore subsystems in each FPGA need to be interfaced with each other through longer transmission lines. However with typical real-time simulation time-step of 50μs, transmission lines with travelling time longer than 2-3 simulation time-steps are common for large scale networks, effectively alleviating the problem. Under special cases where the number of such transmission lines is limited, reduction of simulation time-steps or artificial transmission lines can be added.

5.5 Simulator Capability

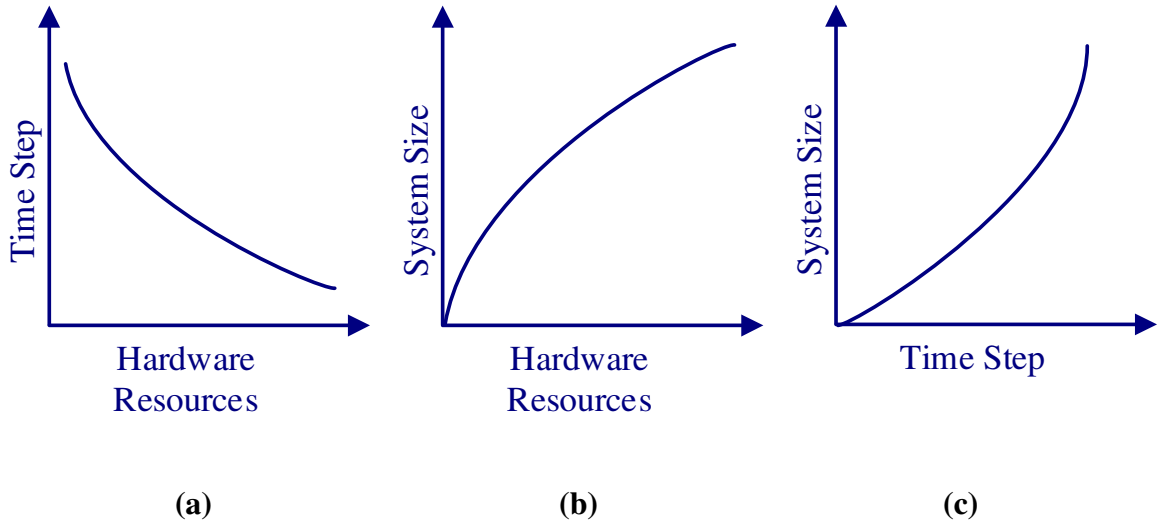


Figure 5-3 Simulation capability of the multi-FPGA based co-simulator

The performance of a simulator can be analysed from two aspects: the size of the network that can be simulated, and the minimum simulation time-step that is required. The capability of the simulator can be represented by the maximum number of nodes that can be simulated.

The required simulation time-step depends on the number of nodes to be simulated, the type of models adopted and the speed of the hardware. Normally the simulation time-step of 50μs or lower is required for the real-time simulation of AC network.

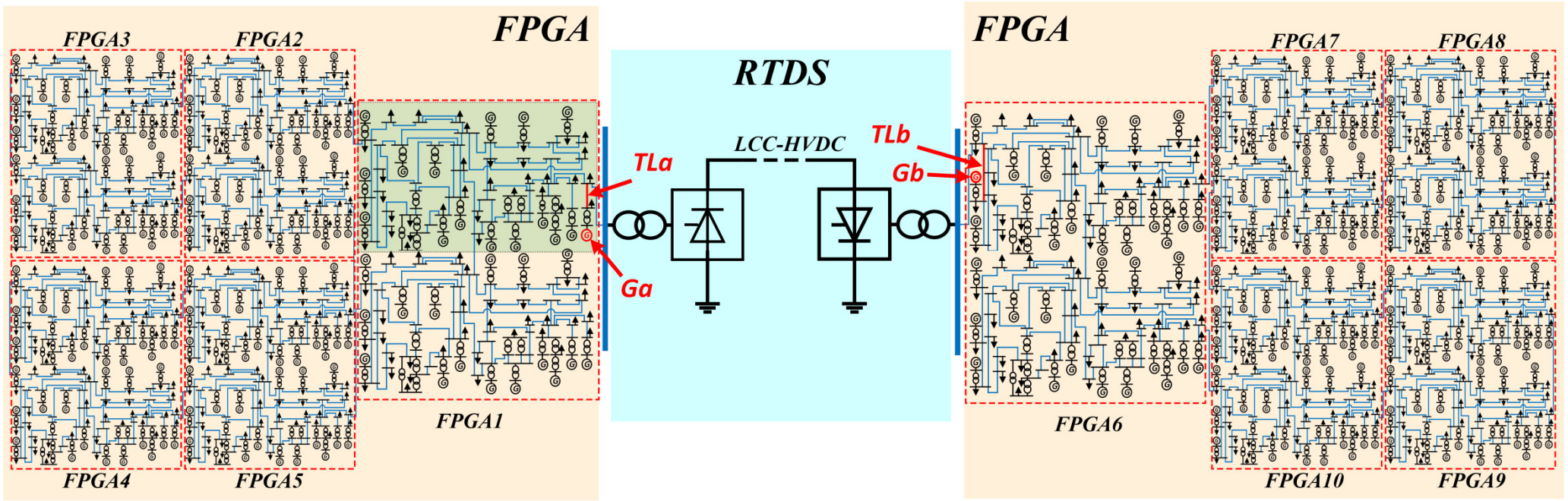
Figure 5-3 shows the relationship between the size of the simulation system, the minimum simulation time-step and the required hardware resources. Figure 5-3(a) shows that the required hardware resources are increased when the required simulation time-step is reduced. This is because with larger simulation time-step, the number of paralleled hardware modules is reduced and the same hardware module can be shared between similar types of calculations. To achieve a small simulation time-step, more paralleled hardware modules are needed which results in an increase of required hardware resources. Figure 5-3(b) shows that under the same simulation time-step, the larger the size of the network, the more hardware resources are required. It is clear that as the size of network increases, more power system components and control systems need to be solved hence more hardware resources are required. Figure 5-3(c) shows that with the same hardware resource, the simulation time-step that can be achieved increases with the increasing size of the network. It indicates that given fixed amount of hardware resource, a trade-off needs to be made between the size of the network and the achievable simulation time-step.

It should be pointed out that a detailed and accurate relationship between these three aspects is difficult to obtain as the actual hardware implementation of FPGA for different power system networks can be very different. Therefore Figure 5-3 is aiming to provide a general trend of the relationships.

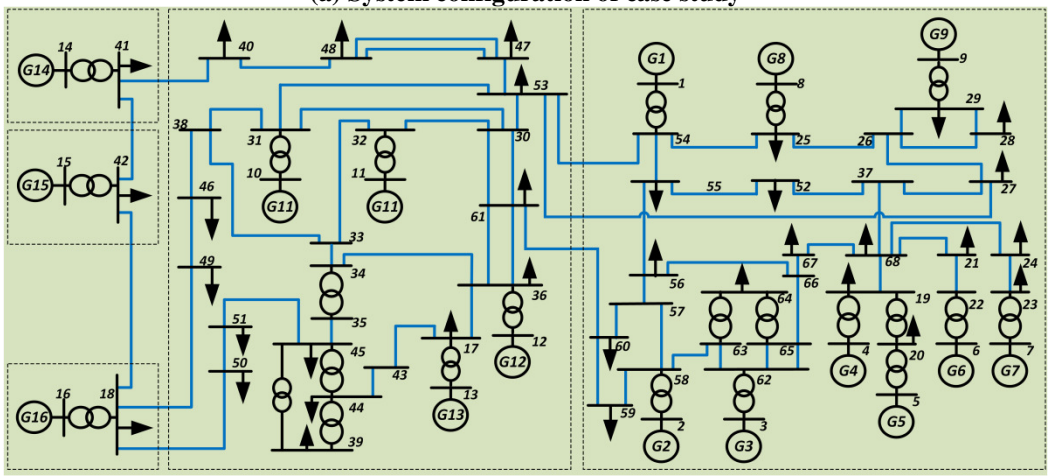
5.6 Case study

5.6.1 Network Configuration

To demonstrate the simulation capability of the proposed multi-FPGA based co-simulator, a large scale power system with 4080 AC nodes is simulated using 10 interconnected FPGAs.



(a) System configuration of case study



(b) Subsystem configuration

Figure 5-4 Single-line diagram of system configuration

Figure 5-4(a) shows the system to be simulated where the RTDS is simulating a Line-Commutated Converter (LCC) based High Voltage Direct Current (HVDC) system and the FPGAs are used to simulate the AC networks. The subsystem highlighted in Figure 5-4(a) is the IEEE standard 68-bus, 16-machine New York/New England system as shown in Figure 5-4(b). Detailed system data can be found in Appendix C.

20 of such subsystems are simulated using 10 FPGAs, with a total of 4080 nodes. 10 of the subsystems form the AC system of rectifier side of LCC HVDC, and the other 10 subsystems form the AC system of inverter side. Detailed models of synchronous generators, excitation systems, governor systems, transmission lines and passive elements as described in Chapter 2 are used.

According to the discussion in Section 5.3.2, the computation of each FPGA is decoupled from others and all FPGAs run in parallel. This largely improves the simulation efficiency and the expandability of the co-simulator.

5.6.2 Hardware Implementation

In this case study, a 136-bus system consisting of two interconnected 68-bus subsystems is simulated on each FPGA. Five FPGAs (FPGA1 - FPGA5) are used to simulate the AC system of rectifier side of LCC HVDC, and the other five FPGAs (FPGA6 - FPGA10) are used to simulate the AC system of inverter side. The subsystems are interconnected by transmission lines and each terminal of the transmission line requires only the voltage and current from the other end as discussed in detailed in 2.2.2.1. In this case study, there are two transmission lines in each 68-bus system interconnecting with other subsystems, located at bus 41 and bus 24. The hardware implementation for each 68-bus subsystem is fully self-

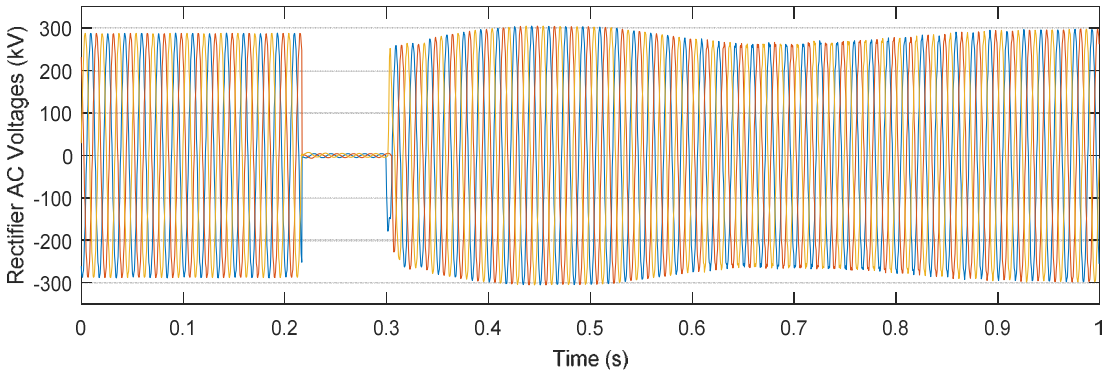
contained with its own power system components modules, synchronous machines, control systems and system solver. That means each subsystem can be seen as a black box with connection ports of transmission lines. The outputs to external networks are the voltages of interconnecting buses, and the line currents on the interconnecting transmission lines. The inputs to this subsystem are the terminal voltages and line currents at the other end of both transmission lines.

There are two options of communication interface between FPGAs as described in Section 5.4. One is direct fibre connection between FPGAs and the other is to use the GUI of RTDS for interconnecting FPGA parts. In this thesis, the interface through RTDS is selected due to the following reasons:

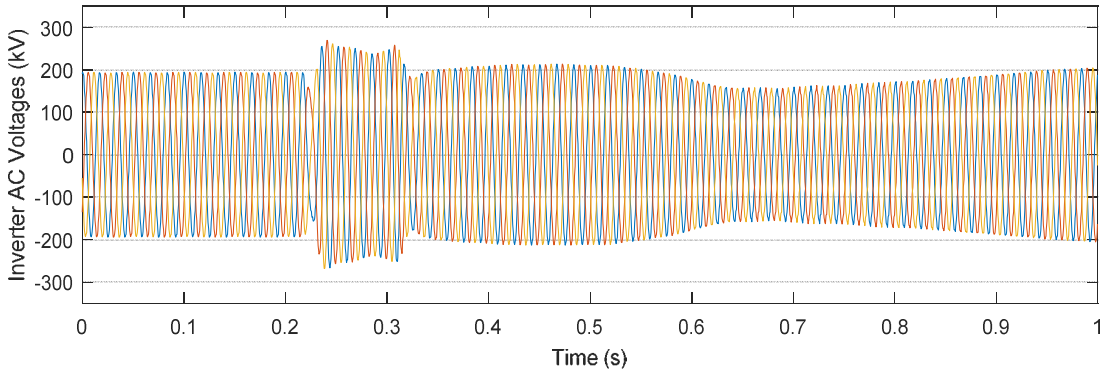
- Each subsystem can be packaged as an independent module with interconnection ports of transmission lines in RSCAD. The structure of the 20 interconnected subsystems can be conveniently configured by the users within GUI. No physical re-wiring is required.
- The number of the FPGA subsystems that needs to be simulated can be conveniently modified within the GUI.

  The visualization of simulation results can be easily accessed and configured within the GUI.

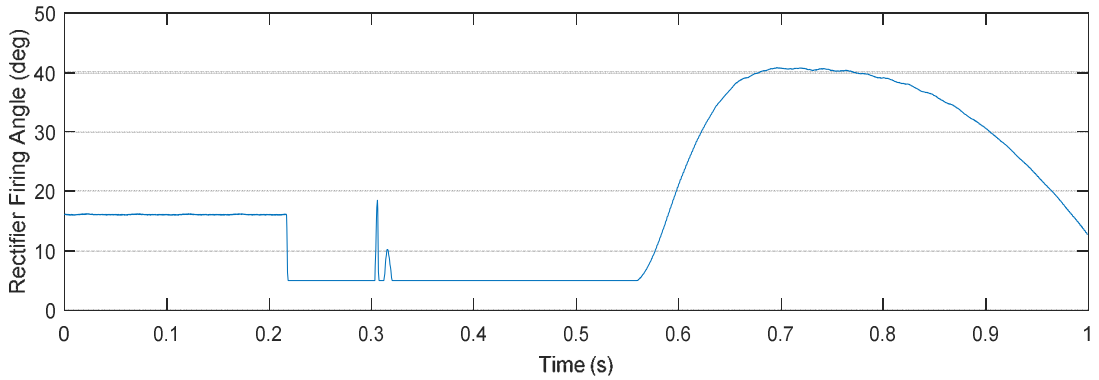
5.6.3 Simulation Results



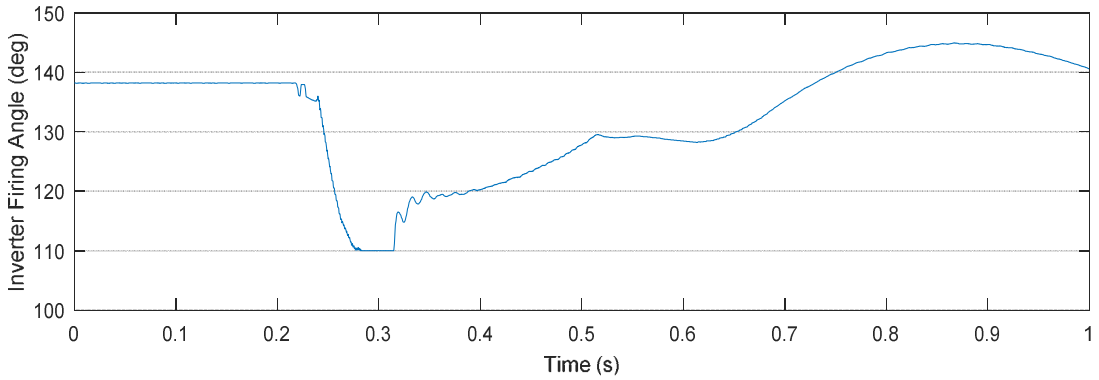
(a) Rectifier three-phase voltages



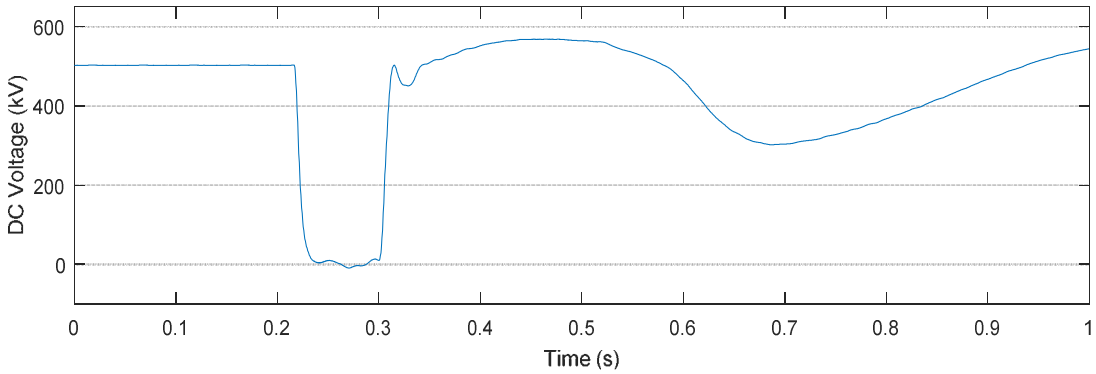
(b) Inverter three-phase voltages



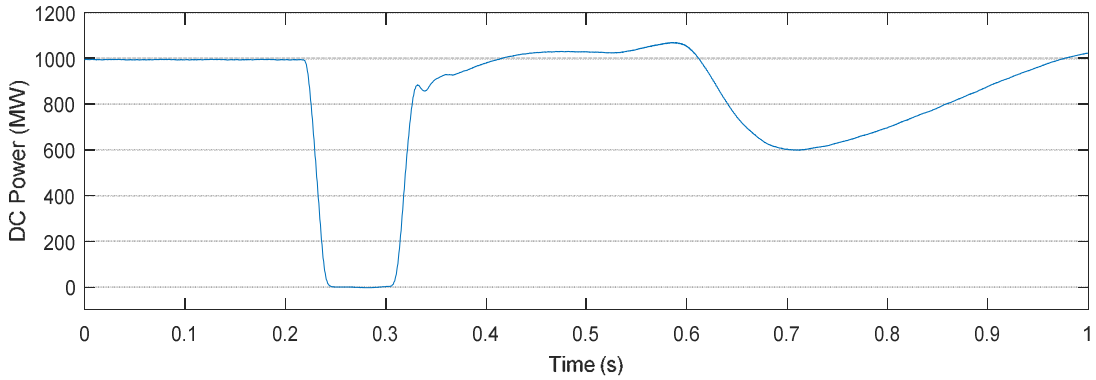
(c) Rectifier firing angle



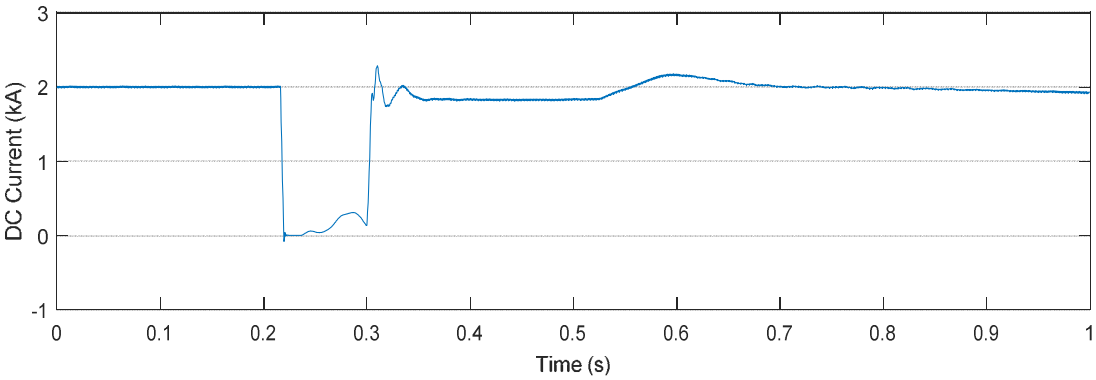
(d) Inverter firing angle



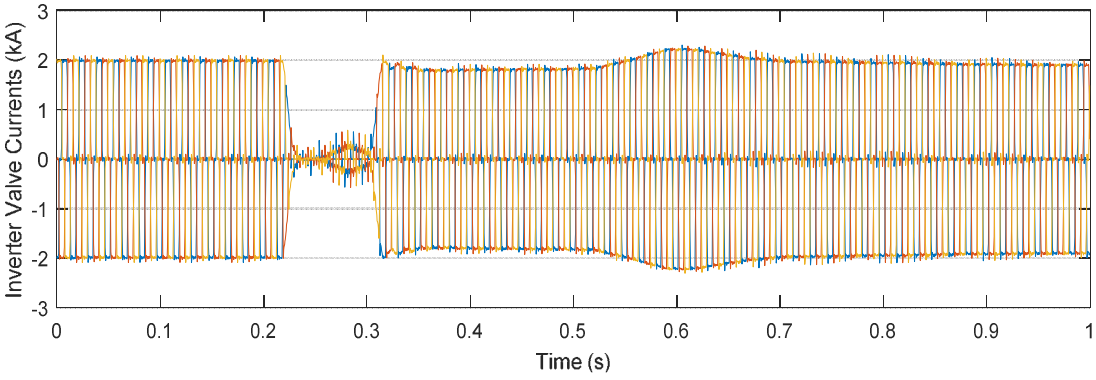
(e) DC Voltage



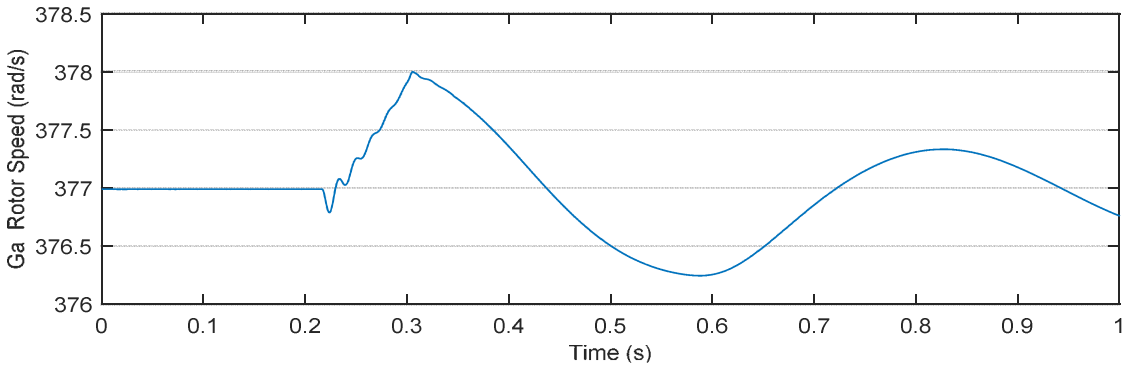
(f) DC Power



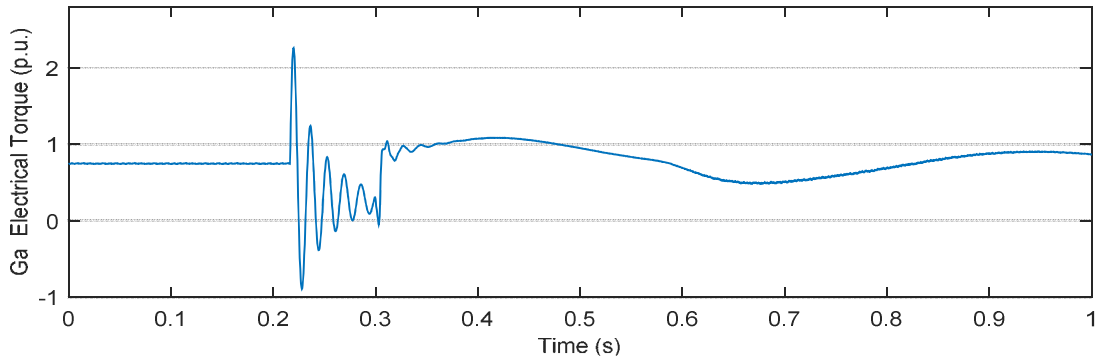
(g) DC current



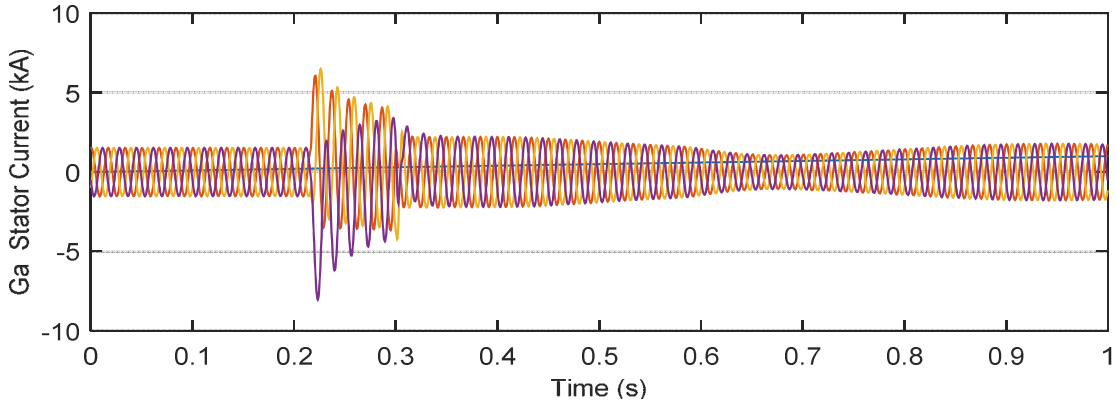
(h) Inverter valve currents



(i) Ga rotor speed



(j) Ga electrical torque



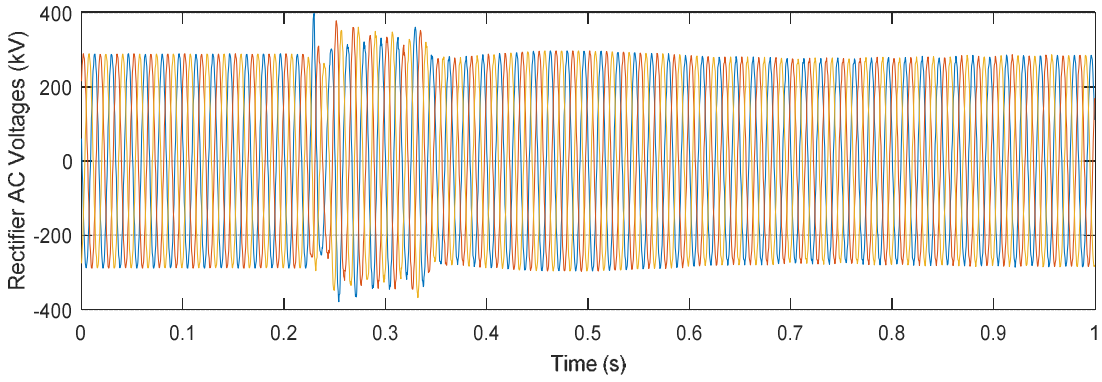
(k) Ga stator current

Figure 5-5 Simulation results with 100ms fault at rectifier side

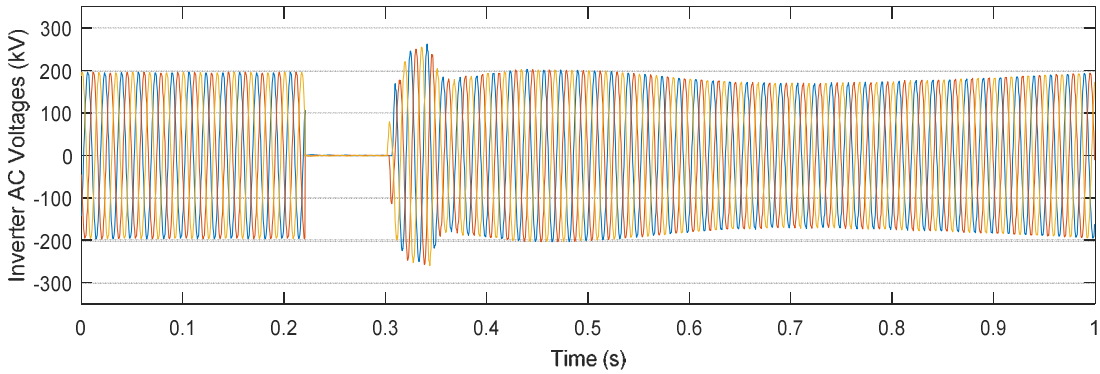
Figure 5-5 shows the simulation results for a 100ms three-phase fault located at the rectifier AC bus. It can be seen from Figure 5-5(a) that when fault happens, the rectifier three-phase AC voltages drop to zero. This directly leads to the drop of DC voltage, DC current, and the transmitted DC power as shown in Figure 5-5(e), Figure 5-5(g) and Figure 5-5(f), respectively. The drop of DC power causes a significant decrease of the reactive power

consumption at inverter side, therefore resulting in an increase of inverter AC bus voltage (Figure 5-5(b)). Also the drop of DC power transfer causes an increase of rotor speed, and the stator current of synchronous machine *Ga* as illustrated in Figure 5-5(i) and Figure 5-5(k), respectively. The oscillatory behaviour of electrical torque of *Ga* due to the cessation of active power transfer can also be observed in Figure 5-5(j). The decrease of DC voltage causes the rectifier side to switch from DC current control to DC voltage control, while the inverter side is switched from minimum extinction angle control to DC current control. As a consequence, both the firing angles of rectifier side and inverter side are decreased as shown in Figure 5-5(c) and Figure 5-5(d), respectively. The drop of DC current causes a drop of the currents flowing through the valves at inverter side (Figure 5-5(h)). It should be pointed out that the dynamic behaviours related to the synchronous machines are normally not available using the traditional RTS due to the limitation of computational resources. Instead of detailed representation of the synchronous machines, Thevenin equivalents are adopted to reduce the computational burden in traditional RTS.

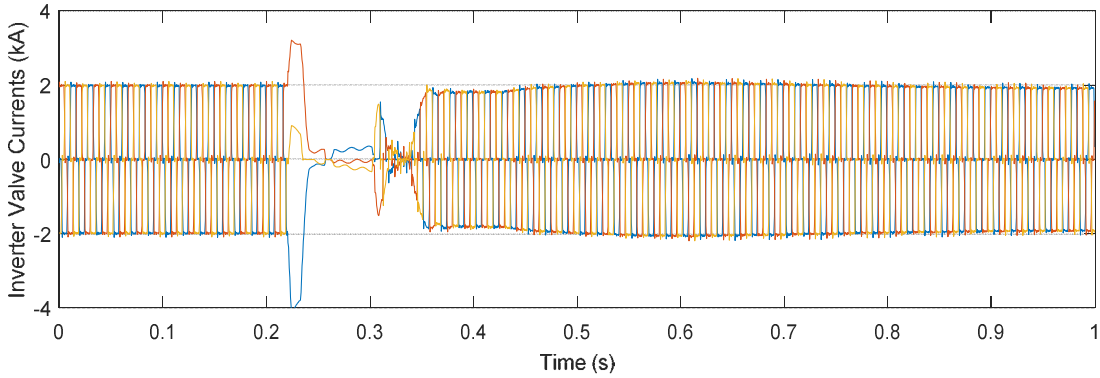
When fault is cleared, low frequency oscillation can be observed in all of the electrical variables shown in Figure 5-5. This is again due to the detailed representation of synchronous generators at either side of the AC system.



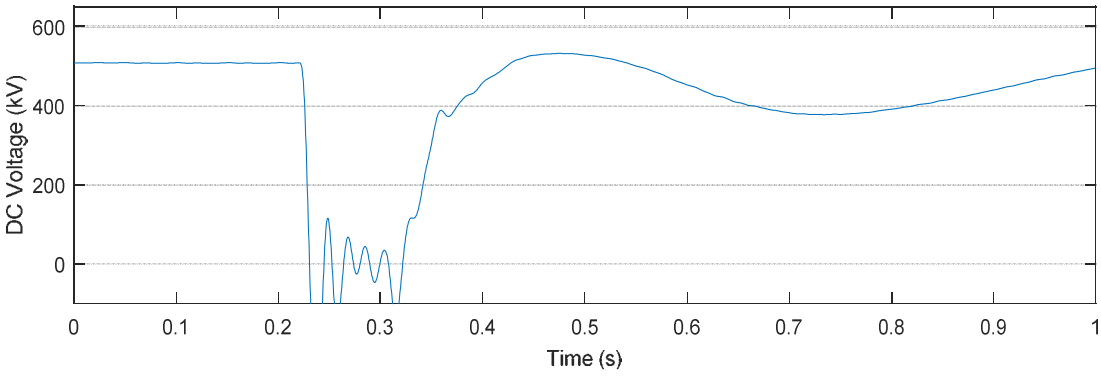
(a) Rectifier three-phase AC voltages



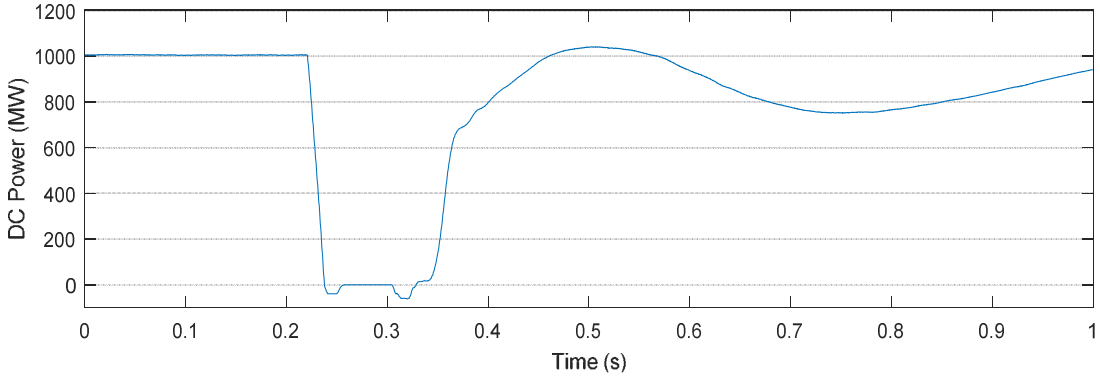
(b) Inverter three-phase AC voltages



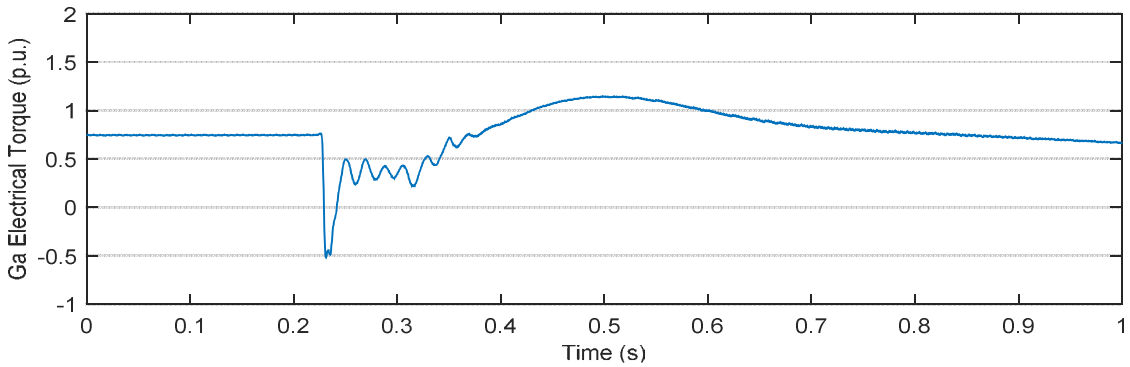
(c) Inverter valve currents



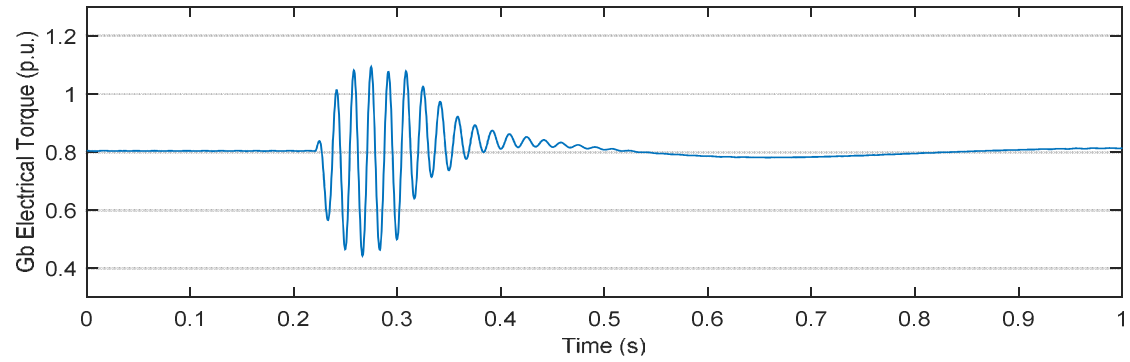
(d) DC voltage



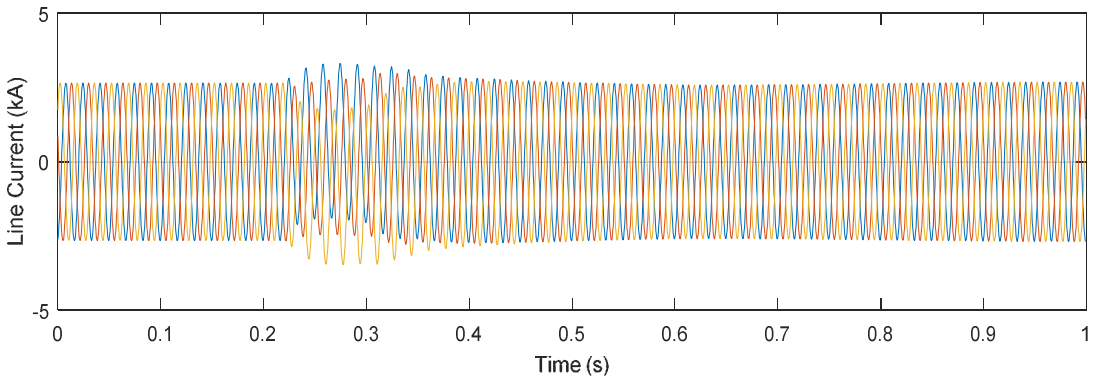
(e) DC active power



(f) Electrical torque of Ga



(g) Electrical torque of Gb



(h) Three-phase line currents of *TLb*

Figure 5-6 Simulation results with 100ms fault at inverter side

Figure 5-6 shows the simulation results of a 100ms three-phase fault at inverter side. When fault is initiated, the inverter side AC voltage is dropped to zero as shown in Figure 5-6(b). This significant voltage drop directly causes the commutation failures as seen from Figure 5-6(c). The commutation failure leads to a DC short circuit at inverter side and therefore causes the drop of DC voltage (Figure 5-6(d)) and DC active power transfer (Figure 5-6(e)).

With the decrease of active power transfer, the reactive power consumption at rectifier side is also significantly decreased. This causes an increase of rectifier side voltage as shown in Figure 5-6(a). Figure 5-6(f) shows the dynamic behaviour of electrical torque of synchronous machine *Ga.* The average torque is decreased as the loading is decreased due to the cessation of active power transfer. Figure 5-6(g) and Figure 5-6(h) show the impacts of fault on the electrical torque of synchronous machine *Gb* and the three-phase line current of *TLb* as highlighted in Figure 5-4.

Similar to the results from rectifier fault, the dynamic effects of synchronous generator on the behaviours of HVDC system can be observed in most of the electrical variables as shown in Figure 5-6. At the same time, the impacts of HVDC system on the synchronous generators at both sides of the AC system can be simulated. It is important that these aspects can be correctly simulated as the interaction between AC and HVDC system can become stronger with increasing installed capacity of HVDC systems.

Results comparisons between the co-simulator and RTDS are not made for this case because the model and algorithm verifications have already been carried out at both component level (Chapter 2) and system level (Chapter 3 and Chapter 4). Similar method of verification is also adopted by RTDS and other commercial simulators. Furthermore the very large size of the system used in this case study is beyond our available RTDS simulation capability.

5.6.4 Discussions

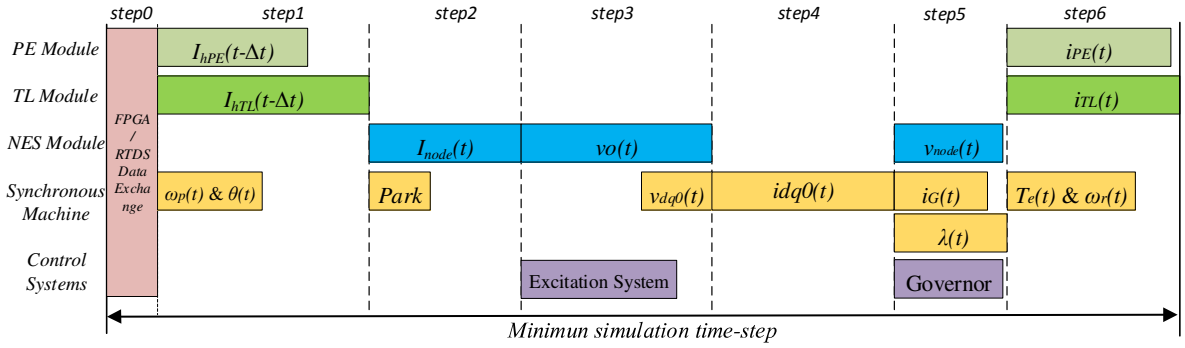


Figure 5-7 Breakdown of minimum simulation time-step

Figure 5-7 shows how the simulation time-step is affected by the processing time of paralleled hardware modules. There are some notable differences when comparing with the time-step break down of Case 1 in Chapter 4(Figure 4-14). It is mainly due to the changes of the number of different power system components. For example, in step 1, the most time-consuming module becomes the transmission line module instead of rotor speed and angle prediction module because the number of transmission lines in this case is significantly larger than that of Case 1 in Chapter 4. Also, as the number of buses increases, the time required for nodal voltages calculation increases accordingly. However, although the number of synchronous machines is increased by several times, the computation time required for synchronous machine modules are more or less kept the same. This is because more paralleled hardware modules have been implemented for the synchronous machines, considering the complexity of the associated calculations. So the calculations of synchronous machines are massively paralleled, resulting in small increase of the computation times. For this case study, the achieved minimum simulation time-step is 22μs under a 100MHz FPGA clock.

5.7 Summary

This chapter describes the developed multi-FPGA based co-simulator for the simulation of large scale power systems. The network partition and the design of communication interfaces are discussed in detail. The capability of the simulator is analysed by considering the interactive effects between the hardware resources, the simulation time-step and the size of the network. Simulation studies of a large scale power system with more than 4000 nodes and detailed power system components have been carried out to demonstrate the capability of the proposed co-simulator. Based on the simulation results, aspects of the minimum simulation time-step and expandability of co-simulator is discussed.

Chapter 6

CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

The modern electrical power system is undergoing dramatic changes, which leads to the continuous increase of its size and complexity. Power system simulation technology, as one of the most important tools in power system planning, operation and analysis, is facing two main challenges. One is the increasing requirement on simulation accuracy and the other is the requirement of efficient simulation of larger scale power systems. One of the most popular methods to accommodate larger AC system is to interface TS with EMT programme. However it requires complex design of interface techniques and the interface error cannot be eliminated. One of the most widely used methods to achieve both simulation speed and accuracy is the EMT-based RTS [1, 2]. However, most of the currently available RTS platforms are either not economical or not powerful enough when simulating large scale power systems (power system with more than 100 three-phase buses [2]). Therefore this thesis proposes an FPGA-RTDS co-simulator to achieve the efficient and accurate simulation of large scale power system. The proposed co-simulator combines the advantages of 1) paralleled architecture and high clock speed from FPGA and 2) better modelling flexibility and user-friendly GUI from RTDS together. The main contributions of this thesis and the development milestones are listed as follows:

- A library of power system components (including synchronous machines, transmission lines, passive RLC elements, voltage/current sources, circuit breakers and the associated control systems) using EMT models have been developed in FPGA. The components models in continuous-time domain have been discretized and then implemented using the hardware description language of VHDL. The accuracy of the models has been verified through comparison with SIMULINK and RTDS. Close agreement with both SIMULINK and RTDS have been reached.
- Based on the established models of power system components and control systems, the FPGA-based real-time EMT simulator has been developed for the simulation of AC power systems. It has been shown that in discretized-time domain, the linear power system components can be represented as Norton equivalent circuits. Therefore the nodal voltage equations can be conveniently formulated for the solution of the complete AC system. To handle the nonlinear components such as synchronous machines, the compensation method has been used to interface with the rest of the AC network. For the solution of nonlinear control systems, the values of feedback signals from previous time-step have been used to greatly reduce the computational burdens. The simulator global controller module has been designed to coordinate the computation of different hardware modules. The power system solver module has been designed to solve the network. Taking advantage of the inherent parallel architecture of FPGAs, the parallelism between different system components and the parallelism of computations within the component modules have been fully exploited for maximized simulation efficiency.
- By integrating RTDS with the FPGA-based EMT simulator, the real-time FPGA-RTDS co-simulator has been developed to further extend the simulation flexibility and capability. The proposed co-simulator combines the advantages of 1) paralleled

architecture and high clock speed from FPGA and 2) better modelling flexibility and user-friendly Graphical User Interface (GUI) from RTDS together. The interface errors are eliminated by the use of EMT models on both FPGA and RTDS. In the proposed co-simulator, the system of interest (study system) is simulated in RTDS while the rest of system (external system) is simulated in FPGA. The decoupling effect due to traveling time delay of transmission line has been utilized to achieve independent and parallel computation of the study system and the external system. The exchange of interface signals have been implemented through direct fibre connection between FPGA and processor cards of RTDS. Fibre connection is used to export the simulation results from FPGA to RTDS so that simulation results from both parts of the co-simulator can be conveniently accessed through RSCAD. Comparisons of simulation results using the proposed co-simulator and RTDS have been made to verify the accuracy of the co-simulator. A large AC system with 141 buses has been simulated to demonstrate the capability of the proposed co-simulator.

- For better expandability of the proposed co-simulator, the hardware modules for power system components have been designed as fully self-contained components with dedicated interface signals. Then expansion of AC network can be conveniently accommodated by pipelining the additional power system component data into existing hardware modules. It leads to minimal increase of the hardware resources and simulation time. Case studies have been used to demonstrate that the minimum simulation time-step is only increased by less than four times when the size of the AC network is increased by more than 20 times. At the same time the utilization of hardware resources is only increased by about 3.5 times.
- The extension to multi-FPGA platform significantly improves the simulation capability of the FPGA-RTDS co-simulator. This development is particular suitable

for the simulation of very large power systems with thousands of buses. To optimize the hardware utilization and minimize the intercommunication requirement between FPGAs, the AC network to be simulated is firstly partitioned into subsystems with interconnecting transmission lines. Then each FPGA is used to simulate one of the subsystems. The required communications between FPGAs depend on the number of interconnected lines and direct fibre connections have been implemented for the intercommunications. The simulation system with large AC networks connected through HVDC link has been modelled using the proposed multi-FPGA based co-simulator. The HVDC link is simulated using the RTDS while the AC systems at rectifier and inverter side are simulated using the multi-FPGA boards. The AC systems with total of 4080 nodes have been used to demonstrate the simulation capability of the proposed method. The interactions between HVDC system and the AC system have been observed from the simulation results, which are usually not available with traditional methods, as the AC system is normally simplified to Thevenin equivalent circuits. In addition, the analysis of hardware utilization has indicated that the proposed method has the potential of simulating even larger power systems.

One of the main difficulties in implementing the proposed method is the lack of user-friendly GUI for power system modelling. This can be partially overcome by utilizing the GUI of RTDS in the proposed co-simulator. The hardware implementation of the detailed power system components requires a relatively high level of programming skill of FPGA. In particular, detailed knowledge on the parallel processing and pipelined design of FPGA is needed to maximize the benefits (fast simulation speed with high accuracy) brought by the use of FPGA in power system simulation.

6.2 Future Work

The following is a list of the possible future work:

- A GUI could be developed for the system modelling using FPGA. At the moment, access to FPGA programming is still limited to hardware level. Reasonable understanding of FPGA and its programming language is required for the modelling of power system using FPGA. With GUI, similar to RSCAD for RTDS, the potential of the co-simulator can be maximized.
- Detailed transformer models could be developed for the integrated simulator. Different models could be developed for various types of transformers such as auto-transformers, multi-winding transformers and quadrature boosters. The nonlinear behaviour and different configurations of these transformers can be modelled to improve the comprehensiveness of the proposed simulator.
- To generalize the FPGA-based simulator as a generic simulation extension to various types of real-time or non-real-time simulators. A generic interface between the FPGA-based simulator and other simulators could be developed to achieve this objective. It would provide a techno-economic feasible solution to the existing simulators.

LIST OF PUBLICATIONS

Journal Papers

1. C. Yang, Y. Xue, X. P. Zhang, Y. Zhang and Y. Chen, "Real-Time FPGA-RTDS Co-Simulator" IEEE Transactions on Power Systems (under review)

2. Y. Xue, X. P. Zhang and C. Yang, "Commutation Failure Elimination of LCC HVDC Systems using Thyristor-Based Controllable Capacitors" IEEE Transactions on Power Delivery (accepted)

3. Y. Xue, X. P. Zhang and C. Yang, " AC Filterless Flexible LCC HVDC with Reduced Voltage Rating of Controllable Capacitors" IEEE Transactions on Power Systems (under review)

4. Y. Xue, X. P. Zhang and C. Yang, "Elimination of Commutation Failures of LCC HVDC System with Controllable Capacitors," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 3289-3299, July 2016. (SCI indexed) Most popular articles among articles published since 1986 (Ranks 5th in August 2016), College of Engineering and Physical Sciences at the University of Birmingham Research Paper of the Month award for July 2016

Conference Papers

1. C. Yang, Y. Xue and X. P. Zhang, "FPGA-based detailed EMTP," *2017 IEEE Manchester PowerTech*, Manchester, 2017, pp. 1-6.

2. Y. Xue, C. Yang and X. P. Zhang, "Investigation of black start capability of LCC HVDC system with controllable capacitors," *12th IET International Conference on AC and DC Power Transmission (ACDC 2016)*, Beijing, 2016, pp. 1-6.

3. Ying Xue, Conghuan Yang, Xiao-Ping Zhang, “Analysis of the Required Insertion Voltage Level of LCC HVDC with Controllable Capacitors for Commutation Failure Elimination”, *2nd International Conference on High Voltage Direct Current,* Shanghai, 2016.

REFERENCES

[1] J. Belanger, P. Venne, and J.-N. Paquin, "The What, Where and Why of Real-Time Simulation," in *IEEE PES General Meeting*, Minneapolis, USA (July, 2010).

[2] Faruque, M. D. O., Strasser, T., Lauss, G*., et al.*, "Real-Time Simulation Technologies for Power Systems Design, Testing, and Analysis," *IEEE Power and Energy Technology Systems Journal,* vol. 2, pp. 63-73, 2015.

[3] IRENA, I.-E. a. Renewable Energy Integration in Power Grids-Technology Brief [Online]. Available: http://www.irena.org/DocumentDownloads/Publications/IRENA-ETSAP_Tech_Brief_Power_Grid_Integration_2015.pdf

[4] Beik, O. and Schofield, N., "An Offshore Wind Generation Scheme With a High-Voltage Hybrid Generator, HVDC Interconnections, and Transmission," *IEEE Transactions on Power Delivery,* vol. 31, pp. 867-877, 2016.

[5] Barker, C. D. and Whitehouse, R. S., "An alternative control strategy for the thyristor based HVDC interconnection of renewable energy supplies," in *The 8th IEE International Conference on AC and DC Power Transmission*, 2006, pp. 225-229.

[6] IEC, "Grid integration of large-capacity Renewable Energy sources and use of large-capacity Electrical Energy Storage," Geneva, Switzerland2012.

[7] Su, W., Eichi, H., Zeng, W*., et al.*, "A Survey on the Electrification of Transportation in a Smart Grid Environment," *IEEE Transactions on Industrial Informatics,* vol. 8, no. 1, pp. 1-10, Feb. 2012.

[8] Ibrahim, E. S., "Interconnection of electric power systems in the Arab world," *Power Engineering Journal,* vol. 10, pp. 121-127, 1996.

[9] Bartlett, S., "Trans-Australian HVDC interconnection investigation," in *2016 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, 2016, pp. 666-670.

[10] Perez, A., Jóhannsson, H., Lund, P*., et al.*, "Evaluation of HVDC interconnection models for considering its impact in real-time voltage stability assessment," in *2015 Modern Electric Power Systems (MEPS)*, 2015, pp. 1-6.

[11] Rodrigo, A. S. and Perera, C. U., "Modeling and simulation of current source converter for proposed India-Sri Lanka HVDC interconnection," in *2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS)*, 2015, pp. 232-237.

[12] Petropoulos, D., Voumvoulakis, E. M., and Hatziargyriou, N. D., "Dynamic stability analysis of HVDC interconnection of autonomous power system of Crete island," in *MedPower 2014*, 2014, pp. 1-8.

[13] Francos, P. L., Verdugo, S. S., Álvarez, H. F*., et al.*, "INELFE & - 2014; Europe's first integrated onshore HVDC interconnection," in *2012 IEEE Power and Energy Society General Meeting*, 2012, pp. 1-8.

[14] Hualei, W. and Redfern, M. A., "Enhancing AC networks with HVDC interconnections," in *CICED 2010 Proceedings*, 2010, pp. 1-7.

[15] Kim, C. K., Shim, E. B., and Lee, S. D., "Feasibility study of HVDC interconnection between south Korea and north Korea," in *2009 Transmission & Distribution Conference & Exposition: Asia and Pacific*, 2009, pp. 1-8.

[16] Cova, B., Pincella, C., Simioli, G*., et al.*, "HVDC interconnections in the Mediterranean Basin," in *2005 IEEE Power Engineering Society Inaugural Conference and Exposition in Africa*, 2005, pp. 143-148.

[17] Liserre, M., Sauter, T., and Hung, J. Y., "Future Energy Systems: Integrating Renewable Energy Sources into the Smart Power Grid Through Industrial Electronics," *IEEE Industrial Electronics Magazine,* vol. 4, pp. 18-37, 2010.

[18] Guerrero, J. M., Hang, L., and Uceda, J., "Control of Distributed Uninterruptible Power Supply Systems," *IEEE Transactions on Industrial Electronics,* vol. 55, pp. 2845-2859, 2008.

[19] Blaabjerg, F., Liserre, M., and Ma, K., "Power Electronics Converters for Wind Turbine Systems," *IEEE Transactions on Industry Applications,* vol. 48, pp. 708-719, 2012.

[20] Li, Y. W. and Kao, C. N., "An Accurate Power Control Strategy for Power-Electronics-Interfaced Distributed Generation Units Operating in a Low-Voltage Multibus Microgrid," *IEEE Transactions on Power Electronics,* vol. 24, pp. 2977-2988, 2009.

[21] Carrasco, J. M., Franquelo, L. G., Bialasiewicz, J. T*., et al.*, "Power-Electronic Systems for the Grid Integration of Renewable Energy Sources: A Survey," *IEEE Transactions on Industrial Electronics,* vol. 53, pp. 1002-1016, 2006.

[22] ENSG, "Our Electricity Transmission Network: A Vision For 2020," 2012.

[23] Offshore wind operational report [Online]. Available: https://www.thecrownestate.co.uk/media/1050888/operationalwindreport2017_final.pdf

[24] Department for Business, E. I. S. Solar photovoltaics deployment [Online]. Available: https://www.gov.uk/government/statistics/solar-photovoltaics-deployment

[25] Wikipedia. Electric vehicle [Online]. Available: https://en.wikipedia.org/wiki/Electric_vehicle

[26] Zhen-ya, L., Yin-biao, S., Wen-liang, Z*., et al.*, "Study on Voltage Class Series for HVDC Transmission System," *Proceeding of the CSEE,* pp. 1-8, 2008.

[27] Department for Business, E. I. S. Digest of UK Energy Statistics (DUKES): renewable sources of energy [Online]. Available: https://www.gov.uk/government/statistics/renewable-sources-of-energy-chapter-6-digest-of-united-kingdom-energy-statistics-dukes

[28] Ofgem. Existing and future interconnector projects [Online]. Available: https://www.ofgem.gov.uk/electricity/transmission-networks/electricity-interconnectors

[29] Al-Mohaisen, A., Chausse, L., and Sud, S., "Progress Report on the GCC Electricity Grid System Interconnection in the Middle East," in *2007 IEEE Power Engineering Society General Meeting*, 2007, pp. 1-7.

[30] Salloum, A. and Heydt, G. T., "Innovative HVDC connections in power transmission systems," in *PES T&D 2012*, 2012, pp. 1-8.

[31] Aredes, M., Dias, R., Aquino, A. F. D. C. D*., et al.*, "Going the Distance," *IEEE Industrial Electronics Magazine,* vol. 5, pp. 36-48, 2011.

[32] Horwill, C., Macleod, N. M., Bonchang, R. E*., et al.*, "A new 500MW frequency converter station to exchange power between Uruguay and Brazil," in *2011 IEEE/PES Power Systems Conference and Exposition*, 2011, pp. 1-6.

[33] Xue, Y., Kong, D., Guan, R*., et al.*, "System performance studies in RTDS for a complex power network with multiple FACTS devices," in *13th IET International Conference on AC and DC Power Transmission (ACDC 2017)*, 2017, pp. 1-6.

[34] GE to provide National Grid with largest, most stable Utility STATCOM in Europe [Online]. Available: https://www.gegridsolutions.com/press/gepress/GE_Statcom_Europe.pdf

[35] Chen, Y., "Large-Scale Real-Time Electromagnetic Transient Simulation of Power Systems Using Hardware Emulation on FPGAs," Department of Electrical and Computer Engineering, University of Alberta, 2012.

[36] Proulx, R., Valette, A., Gingras, J. P*., et al.*, "User-Oriented Simulation of HVDC Control in a Transient Stability Program," *IEEE Power Engineering Review,* vol. PER-5, pp. 23-24, 1985.

[37] Jalili-Marandi, V., Dinavahi, V., Strunz, K*., et al.*, "Interfacing Techniques for Transient Stability and Electromagnetic Transient Programs IEEE Task Force on Interfacing Techniques for Simulation Tools," *IEEE Transactions on Power Delivery,* vol. 24, pp. 2385-2395, 2009.

[38] Siemens. Power Transmission System Planning Software [Online]. Available: http://w3.siemens.com/smartgrid/global/en/products-systems-solutions/software-solutions/planning-data-management-software/planning-simulation/pages/pss-e.aspx

[39] ABB. SIMPOW® - Power system simulation software [Online]. Available: https://library.e.abb.com/public/f683e85bfe59a232c1257b0c0055252f/A02-0050E%20Simpow.pdf

[40] Krebs, R. and Ruhle, O., "NETOMAC real-time simulator - a new generation of standard test modules for enhanced relay testing," in *2004 Eighth IEE International Conference on Developments in Power System Protection*, 2004, pp. 669-674 Vol.2.

[41] GmbH, D. DIgSILENT PowerFactory 15 [Online]. Available: http://www.digsilent.de/tl_files/digsilent/files/powerfactory/PowerFactory15/WhatsNew_PF150.pdf

[42] Dommel, H. W., "Digital Computer Solution of Electromagnetic Transients in Single-and Multiphase Networks," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-88, pp. 388-399, 1969.

[43] Dommel, H. W., *EMTP theory book*. Bonneville Power Administration, 1984.

[44] Centre, M. H. R. PSCAD User's Guide [Online]. Available: https://hvdc.ca/uploads/knowledge_base/pscad_users_guide_v4_6.pdf?t=1497534232

[45] Ould-Bachir, T., Saad, H., Dennetière, S.*, et al.*, "CPU/FPGA-Based Real-Time Simulation of a Two-Terminal MMC-HVDC System," *IEEE Transactions on Power Delivery,* vol. 32, pp. 647-655, 2017.

[46] Plumier, F., Aristidou, P., Geuzaine, C.*, et al.*, "Co-Simulation of Electromagnetic Transients and Phasor Models: A Relaxation Approach," *IEEE Transactions on Power Delivery,* vol. 31, pp. 2360-2369, 2016.

[47] Xuegong, W., Wilson, P., and Woodford, D., "Interfacing transient stability program to EMTDC program," in *Proceedings. International Conference on Power System Technology*, 2002, pp. 1264-1269 vol.2.

[48] Morched, A. S. and Brandwajn, V., "Transmission Network Equivalents for Electromagnetic Transients Studies," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-102, pp. 2984-2994, 1983.

[49] Abdel-Rahman, M., Semlyen, A., and Iravani, M. R., "Two-layer network equivalent for electromagnetic transients," *IEEE Transactions on Power Delivery,* vol. 18, pp. 1328-1335, 2003.

[50] Nie, X., Chen, Y., and Dinavahi, V., "Real-Time Transient Simulation Based on a Robust Two-Layer Network Equivalent," *IEEE Transactions on Power Systems,* vol. 22, pp. 1771-1781, 2007.

[51] Wang, L., Fang, D. Z., and Chung, T. S., "New techniques for enhancing accuracy of EMTP/TSP hybrid simulation algorithm," in *2004 IEEE International Conference on Electric Utility Deregulation, Restructuring and Power Technologies. Proceedings*, 2004, pp. 734-739 Vol.2.

[52] Falcao, D. M., Kaszkurewicz, E., and Almeida, H. L. S., "Application of parallel processing techniques to the simulation of power system electromagnetic transients," *IEEE Transactions on Power Systems,* vol. 8, pp. 90-96, 1993.

[53] "Parallel processing in power systems computation," *IEEE Transactions on Power Systems,* vol. 7, pp. 629-638, 1992.

[54] Semlyen, A. and Leon, F. d., "Computation of electromagnetic transients using dual or multiple time steps," *IEEE Transactions on Power Systems,* vol. 8, pp. 1274-1281, 1993.

[55] Su, H. T., Chan, K. W., Snider, L. A.*, et al.*, "Recent advancements in electromagnetic and electromechanical hybrid simulation," in *2004 International Conference on Power System Technology, 2004. PowerCon 2004.*, 2004, pp. 1479-1484 Vol.2.

[56] Turner, K. S., Heffernan, M. D., Arnold, C. P.*, et al.*, "Computation of A.C.-D.C. System Disturbances. PT. III-Transient Stability Assessment," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-100, pp. 4356-4363, 1981.

[57] Turner, K. S., Heffernan, M. D., Arnold, C. P.*, et al.*, "Computation of A.C.-D.C. System Disturbances. PT. II - Derivation of Power Frequency Variables from Convertor Transient Response," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-100, pp. 4349-4355, 1981.

[58] Sultan, M., Reeve, J., and Adapa, R., "Combined transient and dynamic analysis of HVDC and FACTS systems," *IEEE Transactions on Power Delivery,* vol. 13, pp. 1271-1277, 1998.

[59] Reeve, J. and Adapa, R., "A new approach to dynamic analysis of AC networks incorporating detailed modeling of DC systems. I. Principles and implementation," *IEEE Transactions on Power Delivery,* vol. 3, pp. 2005-2011, 1988.

[60] Adapa, R. and Reeve, J., "A new approach to dynamic analysis of AC networks incorporating detailed modeling of DC systems. II. Application to interaction of DC and weak AC systems," *IEEE Transactions on Power Delivery,* vol. 3, pp. 2012-2019, 1988.

[61] Anderson, G. W. J., Watson, N. R., Arnold, N. P.*, et al.*, "A new hybrid algorithm for analysis of HVDC and FACTS systems," in *Energy Management and Power Delivery, 1995. Proceedings of EMPD '95., 1995 International Conference on*, 1995, pp. 462-467 vol.2.

[62] Hongtian, S., Chan, K. K. W., and Snider, L. A., "Interfacing an electromagnetic SVC model into the transient stability simulation," in *Proceedings. International Conference on Power System Technology*, 2002, pp. 1568-1572 vol.3.

[63] Su, H. T., Chan, K. W., and Snider, L. A., "Parallel interaction protocol for electromagnetic and electromechanical hybrid simulation," *IEE Proceedings - Generation, Transmission and Distribution,* vol. 152, pp. 406-414, 2005.

[64] Tian, F., Yue, C., Wu, Z.*, et al.*, "Realization of Electromechanical Transient and Electromagnetic Transient Real Time Hybrid Simulation in Power System," in *2005 IEEE/PES Transmission & Distribution Conference & Exposition: Asia and Pacific*, 2005, pp. 1-6.

[65] Heffernan, M. D., Turner, K. S., Arrillaga, J.*, et al.*, "Computation of A.C.-D.C. System Disturbances - Part I. Interactive Coordination of Generator and Convertor Transient Models," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-100, pp. 4341-4348, 1981.

[66] Gustavsen, B. and Semlyen, A., "Rational approximation of frequency domain responses by vector fitting," *IEEE Transactions on Power Delivery,* vol. 14, pp. 1052-1061, 1999.

[67] Kasztenny, B. and Kezunovic, M., "A method for linking different modeling techniques for accurate and efficient simulation," *IEEE Transactions on Power Systems,* vol. 15, pp. 65-72, 2000.

[68] Johnson, W. R., Wilson, D. D., and Anderson, J. G., "500-Kv-Line Design," *IEEE Transactions on Power Apparatus and Systems,* vol. 82, pp. 572-581, 1963.

[69] Hopkinson, R. H., "Ferroresonance During Single-Phase Switching of 3-Phase Distribution Transformer Banks," *IEEE Transactions on Power Apparatus and Systems,* vol. 84, pp. 289-293, 1965.

[70] Hopkinson, R. H., "Ferroresonant Overvoltage Control Based on TNA Tests on Three-Phase Delta - Wye Transformer Banks," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-86, pp. 1258-1265, 1967.

[71] Isaacs, A., "Simulation Technology: The Evolution of the Power System Network [History]," *IEEE Power and Energy Magazine,* vol. 15, pp. 88-102, 2017.

[72] Kezunovic, M., Aganagic, M., Skendzic, V.*, et al.*, "Transients computation for relay testing in real-time," *IEEE Transactions on Power Delivery,* vol. 9, pp. 1298-1307, 1994.

[73] Dufour, C., Hoang, L.-H., Soumagne, J. C.*, et al.*, "Real-time simulation of power transmission lines using Marti model with optimal fitting on dual-DSP card," *IEEE Transactions on Power Delivery,* vol. 11, pp. 412-419, 1996.

[74] Marti, J. R. and Linares, L. R., "Real-time EMTP-based transients simulation," *IEEE Transactions on Power Systems,* vol. 9, pp. 1309-1317, 1994.

[75] Marti, J. R., Linares, L. R., Calvino, J.*, et al.*, "OVNI: an object approach to real-time power system simulators," in *Power System Technology, 1998. Proceedings. POWERCON '98. 1998 International Conference on*, 1998, pp. 977-981 vol.2.

[76] Hollman, J. A. and Marti, J. R., "Real time network simulation with PC-cluster," *IEEE Transactions on Power Systems,* vol. 18, pp. 563-569, 2003.

[77] Lok-Fu, P., Faruque, M. O., Xin, N.*, et al.*, "A versatile cluster-based real-time digital simulator for power engineering research," *IEEE Transactions on Power Systems,* vol. 21, pp. 455-465, 2006.

[78] Ramanathan, R. M. Pogo Linux [Online]. Available: http://www.pogolinux.com/learn/files/quad-core-06.pdf

[79] Matar, M. and Iravani, R., "FPGA Implementation of the Power Electronic Converter Model for Real-Time Simulation of Electromagnetic Transients," *IEEE Transactions on Power Delivery,* vol. 25, pp. 852-860, 2010.

[80] Matar, M. and Iravani, R., "The Reconfigurable-Hardware Real-Time and Faster-Than-Real-Time Simulator for the Analysis of Electromagnetic Transients in Power Systems," *IEEE Transactions on Power Delivery,* vol. 28, pp. 619-627, 2013.

[81] Kidokoro, H. and Nakahara, M., "FPGA-based hardware-in-the-loop simulator of high switching frequency power converters," in *2015 IEEE International Telecommunications Energy Conference (INTELEC)*, 2015, pp. 1-6.

[82] Saad, H., Ould-Bachir, T., Mahseredjian, J.*, et al.*, "Real-Time Simulation of MMCs Using CPU and FPGA," *IEEE Transactions on Power Electronics,* vol. 30, pp. 259-267, 2015.

[83] Ould-Bachir, T., Blanchette, H. F., and Al-Haddad, K., "A Network Tearing Technique for FPGA-Based Real-Time Simulation of Power Converters," *IEEE Transactions on Industrial Electronics,* vol. 62, pp. 3409-3418, 2015.

[84] Razzaghi, R., Paolone, M., and Rachidi, F., "A general purpose FPGA-based real-time simulator for power systems applications," in *IEEE PES ISGT Europe 2013*, 2013, pp. 1-5.

[85] Ould-Bachir, T., Dufour, C., Bélanger, J*., et al.*, "Effective floating-point calculation engines intended for the FPGA-based HIL simulation," in *2012 IEEE International Symposium on Industrial Electronics*, 2012, pp. 1363-1368.

[86] Larijani, M. R., Zolghadri, M. R., and Shahbazi, M., "Design and implementation of an FPGA-based Real-time simulator for H-Bridge converter," in *2016 7th Power Electronics and Drive Systems Technologies Conference (PEDSTC)*, 2016, pp. 504-510.

[87] Bachir, T. O. and David, J. P., "FPGA-based real-time simulation of state-space models using floating-point cores," in *Proceedings of 14th International Power Electronics and Motion Control Conference EPE-PEMC 2010*, 2010, pp. S2-26-S2-31.

[88] Esparza, M. A., Alvarez-Salas, R., Miranda, H*., et al.*, "Real-time emulator of an induction motor: FPGA-based implementation," in *2012 9th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2012, pp. 1-6.

[89] Meka, R., Sloderbeck, M., Faruque, M. O*., et al.*, "FPGA model of a high-frequency power electronic converter in an RTDS power system co-simulation," in *2013 IEEE Electric Ship Technologies Symposium (ESTS)*, 2013, pp. 71-75.

[90] Dagbagi, M., Idkhajine, L., Monmasson, E*., et al.*, "FPGA implementation of Power Electronic Converter real-time model," in *International Symposium on Power Electronics Power Electronics, Electrical Drives, Automation and Motion*, 2012, pp. 658-663.

[91] Bai, H., Zhou, Z., and Liu, Z., "Design and implementation of an FPGA-based real time simulation system for photovoltaic power generation," in *2014 IEEE Conference and Expo Transportation Electrification Asia-Pacific (ITEC Asia-Pacific)*, 2014, pp. 1-5.

[92] Milton, M., Benigni, A., and Bakos, J., "System-Level, FPGA-Based, Real-Time Simulation of Ship Power Systems," *IEEE Transactions on Energy Conversion,* vol. 32, pp. 737-747, 2017.

[93] Jena, S., Panda, G., and Peesapati, R., "Real-time analysis and simulation of multi-string grid connected photovoltaic inverter using FPGA," in *2016 IEEE 6th International Conference on Power Systems (ICPS)*, 2016, pp. 1-6.

[94] Chen, Y. and Dinavahi, V., "FPGA-Based Real-Time EMTP," *IEEE Transactions on Power Delivery,* vol. 24, pp. 892-902, 2009.

[95] Chen, Y. and Dinavahi, V., "Hardware Emulation Building Blocks for Real-Time Simulation of Large-Scale Power Grids," *IEEE Transactions on Industrial Informatics,* vol. 10, pp. 373-381, 2014.

[96] McLaren, P. G., Kuffel, R., Wierckx, R*., et al.*, "A real time digital simulator for testing relays," *IEEE Transactions on Power Delivery,* vol. 7, pp. 207-213, 1992.

[97] J, B., V, L., and C, D., "eMEGAsim: An open high-performance distributed real-time power grid simulator. architecture and specification," presented at the International Conference on Power Systems (ICPS'07), Bangalore, India, 2007.

[98] Do, V. Q., Soumagne, J. C., Sybille, G*., et al.*, "HYPERSIM, an integrated real-time simulator for power network and control systems," in *Proc. of International Conference on Digital Power System Simulators (ICDS 1999)*, Vasteras, Sweden, 1999.

[99] Pare, D., Turmel, G., Soumagne, J.-C*., et al.*, "Validation tests of the HYPERSIM digital real time simulator with a large AC-DC network," in *Proc. of the International Conference on Power Systems Transients (IPST 2003)*, New Orleans, Louisiana, USA, September 2003.

[100] dSPACE GmbH. dSPACE Product Manual [Online]. Available: http://www.dspace.com

[101] Lu, B., Wu, X., Figueroa, H*., et al.*, "A Low-Cost Real-Time Hardware-in-the-Loop Testing Approach of Power Electronics Controls," *IEEE Transactions on Industrial Electronics,* vol. 54, pp. 919-931, 2007.

[102] Adler, F., Benigni, A., Stagge, H*., et al.*, "A new versatile hardware platform for digital real-time simulation: Verification and evaluation," in *2012 IEEE 13th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2012, pp. 1-8.

[103] Hui, P., Lee, B., and Chikkagoudar, S., "Towards Real-Time High Performance Computing for Power Grid Analysis," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, 2012, pp. 306-312.

[104] Inc, M. xPC Target [Online]. Available: http://www.mathworks.com

[105] Inc., A. D. I. ADI rtX RTS [Online]. Available: http://www.adi.com

[106] Majstorovic, D., Celanovic, I., Teslic, N. D*., et al.*, "Ultralow-Latency Hardware-in-the-Loop Platform for Rapid Validation of Power Electronics Designs," *IEEE Transactions on Industrial Electronics,* vol. 58, pp. 4708-4716, 2011.

[107] GmbH, T. H. Typhoon HIL RTS [Online]. Available: http://www.typhoon-hil.com

[108] Kulicke, B., Lerch, E., Ruhle, O*., et al.*, "NETOMAC - calculating, analyzing and optimization the dynamic of electrical systems in time and frequency domain," presented at the Proc. of International Conference on Power Systems Transients (IPST 1999), Budapest, Hungary, 1999.

[109] Dally, W. J., *Digital Systems Engineering* 1ed.: Cambridge University Press, 2008.

[110] Martin, K. W., *Digital Integrated Circuit Design*: Oxford University Press, 2000.

[111] Altera data book [Online]. Available: www.altera.com

[112] Xilinx data book [Online]. Available: www.xilinx.com

[113] Brown, S., "FPGA architectural research: a survey," *IEEE Design & Test of Computers,* vol. 13, pp. 9-15, 1996.

[114] Trimberger, S., "A reprogrammable gate array and applications," *Proceedings of the IEEE,* vol. 81, pp. 1030-1041, 1993.

[115] Xilinx. Virtex-6 Family Overview [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf

[116] Djordjevic, I. B., Arabaci, M., and Minkov, L. L., "Next Generation FEC for High-Capacity Communication in Optical Transport Networks," *Journal of Lightwave Technology,* vol. 27, pp. 3518-3530, 2009.

[117] Bueno, E. J., Hernandez, A., Rodriguez, F. J.*, et al.*, "A DSP- and FPGA-Based Industrial Control With High-Speed Communication Interfaces for Grid Converters Applied to Distributed Power Generation Systems," *IEEE Transactions on Industrial Electronics,* vol. 56, pp. 654-669, 2009.

[118] Tianqi, L., Dumpala, N. K., and Tessier, R., "Hybrid hard NoCs for efficient FPGA communication," in *2016 International Conference on Field-Programmable Technology (FPT)*, 2016, pp. 157-164.

[119] Costas, L., Fernández-Molanes, R., Rodríguez-Andina, J. J.*, et al.*, "Characterization of FPGA-master ARM communication delays in zynq devices," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, 2017, pp. 942-947.

[120] Wu, A., Jin, X., Du, X.*, et al.*, "A flexible FPGA-to-FPGA communication system," in *2017 19th International Conference on Advanced Communication Technology (ICACT)*, 2017, pp. 836-843.

[121] Gong, J., Wang, T., Chen, J.*, et al.*, "An efficient and flexible host-FPGA PCIe communication library," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, 2014, pp. 1-6.

[122] Dong-Hahk, L., Anna, C., Jun-Mo, K.*, et al.*, "A wideband DS-CDMA modem for a mobile station," *IEEE Transactions on Consumer Electronics,* vol. 45, pp. 1259-1269, 1999.

[123] Monmasson, E. and Cirstea, M. N., "FPGA Design Methodology for Industrial Control Systems—A Review," *IEEE Transactions on Industrial Electronics,* vol. 54, pp. 1824-1842, 2007.

[124] Cirstea, M. N. and Dinu, A., "A VHDL Holistic Modeling Approach and FPGA Implementation of a Digital Sensorless Induction Motor Control Scheme," *IEEE Transactions on Industrial Electronics,* vol. 54, pp. 1853-1864, 2007.

[125] Chan, Y. F., Moallem, M., and Wang, W., "Design and Implementation of Modular FPGA-Based PID Controllers," *IEEE Transactions on Industrial Electronics,* vol. 54, pp. 1898-1906, 2007.

[126] Kung, Y. S., Huang, C. C., and Tsai, M. H., "FPGA Realization of an Adaptive Fuzzy Controller for PMLSM Drive," *IEEE Transactions on Industrial Electronics,* vol. 56, pp. 2923-2932, 2009.

[127] Idkhajine, L., Monmasson, E., Naouar, M. W.*, et al.*, "Fully Integrated FPGA-Based Controller for Synchronous Motor Drive," *IEEE Transactions on Industrial Electronics,* vol. 56, pp. 4006-4017, 2009.

[128] Karimi, S., Poure, P., and Saadate, S., "An HIL-Based Reconfigurable Platform for Design, Implementation, and Verification of Electrical System Digital Controllers," *IEEE Transactions on Industrial Electronics,* vol. 57, pp. 1226-1236, 2010.

[129] Monmasson, E., Idkhajine, L., Cirstea, M. N.*, et al.*, "FPGAs in Industrial Control Applications," *IEEE Transactions on Industrial Informatics,* vol. 7, pp. 224-243, 2011.

[130] Rodriguez-Andina, J. J., Moure, M. J., and Valdes, M. D., "Features, Design Tools, and Application Domains of FPGAs," *IEEE Transactions on Industrial Electronics,* vol. 54, pp. 1810-1823, 2007.

[131] Jin, S., Cho, J., Pham, X. D.*, et al.*, "FPGA Design and Implementation of a Real-Time Stereo Vision System," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 20, pp. 15-26, 2010.

[132] Cummings, M. and Haruyama, S., "FPGA in the software radio," *IEEE Communications Magazine,* vol. 37, pp. 108-112, 1999.

[133] Gokhale, M., Stone, J., Arnold, J.*, et al.*, "Stream-oriented FPGA computing in the Streams-C high level language," in *Proceedings 2000 IEEE Symposium on Field-Programmable Custom Computing Machines (Cat. No.PR00871)*, 2000, pp. 49-56.

[134] Page, A. and Mohsenin, T., "FPGA-Based Reduction Techniques for Efficient Deep Neural Network Deployment," in *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2016, pp. 200-200.

[135] Huerta, S. C., Castro, A. d., GarcÍa, O.*, et al.*, "FPGA-Based Digital Pulsewidth Modulator With Time Resolution Under 2 ns," *IEEE Transactions on Power Electronics,* vol. 23, pp. 3135-3141, 2008.

[136] Oriti, G. and Julian, A. L., "Three-Phase VSI with FPGA-Based Multisampled Space Vector Modulation," *IEEE Transactions on Industry Applications,* vol. 47, pp. 1813-1820, 2011.

[137] Alvarez, J., Ó, L., Freijedo, F. D.*, et al.*, "Digital Parameterizable VHDL Module for Multilevel Multiphase Space Vector PWM," *IEEE Transactions on Industrial Electronics,* vol. 58, pp. 3946-3957, 2011.

[138] Foley, R., Kavanagh, R., Marnane, W.*, et al.*, "Multiphase digital pulsewidth modulator," *IEEE Transactions on Power Electronics,* vol. 21, pp. 842-846, 2006.

[139] Dufour, C., Belanger, J., and Lapointe, V., "FPGA-based Ultra-Low Latency HIL Fault Testing of a Permanent Magnet Motor Drive using RT-LAB-XSG," in *2008 Joint International Conference on Power System Technology and IEEE Power India Conference*, 2008, pp. 1-7.

[140] Chen, Y. and Dinavahi, V., "An Iterative Real-Time Nonlinear Electromagnetic Transient Solver on FPGA," *IEEE Transactions on Industrial Electronics,* vol. 58, pp. 2547-2555, 2011.

[141] Chen, Y. and Dinavahi, V., "Digital Hardware Emulation of Universal Machine and Universal Line Models for Real-Time Electromagnetic Transient Simulation," *IEEE Transactions on Industrial Electronics,* vol. 59, pp. 1300-1309, 2012.

[142] Liu, J. and Dinavahi, V., "A Real-Time Nonlinear Hysteretic Power Transformer Transient Model on FPGA," *IEEE Transactions on Industrial Electronics,* vol. 61, pp. 3587-3597, 2014.

[143] Chen, Y. and Dinavahi, V., "Multi-FPGA digital hardware design for detailed large-scale real-time electromagnetic transient simulation of power systems," *IET Generation, Transmission & Distribution,* vol. 7, pp. 451-463, 2013.

[144] Matar, M. and Iravani, R., "Massively Parallel Implementation of AC Machine Models for FPGA-Based Real-Time Simulation of Electromagnetic Transients," *IEEE Transactions on Power Delivery,* vol. 26, pp. 830-840, 2011.

[145] Parma, G. G. and Dinavahi, V., "Real-Time Digital Hardware Simulation of Power Electronics and Drives," *IEEE Transactions on Power Delivery,* vol. 22, pp. 1235-1246, 2007.

[146] Myaing, A. and Dinavahi, V., "FPGA-Based Real-Time Emulation of Power Electronic Systems With Detailed Representation of Device Characteristics," *IEEE Transactions on Industrial Electronics,* vol. 58, pp. 358-368, 2011.

[147] Kundur, P., *Power System Stability and Control*. New York, NY, USA: McGraw-Hill, 1994.

[148] Haginomori, E., Koshiduka, T., Arai, J.*, et al.*, *Power System Transient Analysis: Theory and Practice using Simulation Programs (ATP-EMTP)*: Wiley, May 2016.

[149] Marti, J. R. and Lin, J., "Suppression of numerical oscillations in the EMTP power systems," *IEEE Transactions on Power Systems,* vol. 4, pp. 739-747, 1989.

[150] "IEEE Recommended Practice for Excitation System Models for Power System Stability Studies," *IEEE Std 421.5-2016 (Revision of IEEE Std 421.5-2005),* pp. 1-207, 2016.

[151] Dommel, H. W., "Nonlinear and Time-Varying Elements in Digital Simulation of Electromagnetic Transients," *IEEE Transactions on Power Apparatus and Systems,* vol. PAS-90, pp. 2561-2567, 1971.

[152] Tinney, W. and Meyer, W., "Solution of large sparse systems by ordered triangular factorization," *IEEE Transactions on Automatic Control,* vol. 18, no. 4, pp. 333-346, Aug 1973.

[153] Grid, N. Electricty Ten Year Statement [Online]. Available: https://www.nationalgrid.com/uk/publications/electricity-ten-year-statement-etys

[154] Pal, B. and Chaudhuri, B., *Robust Control in Power Systems*. New York, U.S.A: Springer, 2005.

Appendix A.

Data of Case Study in Chapter 2

A.1 Case 1

Synchronous Machine Parameters:

555MVA, 24kV, 1 pole pair, 60Hz, 3600RPM, Field circuit current: 1300.2A;

The winding resistances and leakage inductances are listed in Table A.1. The machine rated values are used as the base values.

Table A.1 Synchronous Machine Parameters

Winding Resistances (p.u.)		*Leakage Inductances (p.u.)*	
R_d	0.003	L_{ad}	1.66
R_q	0.003	L_{aq}	1.61
R_0	0.003	L_0	0
R_{D1}	0.0284	L_l	0.15
R_{Q1}	0.00619	L_{f1d}	1.66
R_{Q2}	0.02368	L_{ffd}	1.825
R_f	0.0006	L_{11q}	2.3352
		L_{11d}	1.8313
		L_{22q}	1.735

Constant excitation voltage v_f=123.8825 (V).

Constant mechanical torque T_m=671380 (N*m).

Voltage source: 24kV (phase-to-phase RMS), 60Hz. *System equivalent impedance*: 5Ω.

A.2 Case 2:

The machine rated values are used as the base values.

Excitation System:

based on IEEE standard AC1A excitation system model [150]:

Low-pass filter time constant: T_r=20e-4(s)

Voltage regulator gain and time constant: K_a=400, T_a=0.02(s).

Voltage regulator internal limits: V_{Amin}=-14.5 (pu), V_{Amax}=14.5 (pu).

Voltage regulator output limits: V_{Rmin}=-5.43 (pu), V_{Rmax}=6.03 (pu).

Damping filter gain and time constant: K_f=0.03, T_f=1.0.

Lead and lag time constants for transient gain reduction: T_b=0 (s), T_c=0 (s).

Exciter gain and time constant: K_e=1.0, T_e=0.80 (s).

Exciter alternator voltage values: V_{e1}=4.18 (pu), V_{e2}=3.14 (pu).

Exciter saturation function values: *SeVe1*=0.10 (pu), *SeVe2*=0.03 (pu).

Demagnetizing factor: K_d=0.38 (pu).

Rectifier loading factor: K_c=0.20 (pu).

Governor System:

Speed reference: 0.998 (pu)

Percentage droop: 5%

Load reference set point: 0.5 (pu)

Time constant of governor: 0.2 (s)

Time constant of main inlet volumes and steam chest: 0.3 (s)

Base torque: 1.47218e+06 (N*m)

Initial torque: 0 (pu)

Synchronous Machine Parameters: The same as Case 1.

Voltage source: 24kV (phase-to-phase RMS), 60Hz. *System equivalent impedance*: 20Ω.

A.3 Case 3:

Transmission line:

Number of phases: 3

MVA base: 100 MVA

Frequency used for RLC specifications: 60Hz

Resistance per unit length: r_1 *(positive and negative sequence)*=0.0529 (pu/km),

r_0 *(zero sequence)*=0.3864 (pu/km)

Inductance per unit length: l_1 *(positive and negative sequence)*=1.4032e-3 (pu/km)

l_0 *(zero sequence)*=4.1264e-3 (pu/km)

Capacitance per unit length: c_1 *(positive and negative sequence)*=8.7751e-9 (pu/km)

c_0 *(zero sequence)*=7.751e-9 (pu/km)

Line length: 114 (km)

Three phase inductance: *L*=2 (pu)

Three phase capacitance: *C*= 3e-10 pu

Voltage source: 24kV (phase-to-phase RMS), 60Hz.

Phase A to ground fault resistance: 0.01 (pu)

Appendix B.

Data of Two-Area Four-Machine Power System [147]

Transmission line:

Number of phases: 3

Frequency used for RLC specification: 60Hz

Resistance per unit length: r_1=0.0001 (pu/km), r_0=0.0002 (pu/km)

Inductive reactance per unit length: x_1=0.001 (pu/km), x_0=0.002 (pu/km)

Capacitive reactance per unit length: b_1=0.00175 (pu/km), b_0=8.75e-4 (pu/km)

Line length: *TL56*=25(km), *TL1011*=25(km), *TL67*=10 (km), *TL910*=10 (km), *TL78*=110(km), *TL89*=110(km)

Transformer T1 and T3 impedance: X_T=0.15 (pu)

Load data and shunt capacitors at bus 7 and 9:

Bus7: P_L=967MW, Q_L=100MVAr, Q_C=200MVAr.

Bus9: P_L=1767MW, Q_L=100MVAr, Q_C=350MVAr.

Bus 7 fault resistance: 0.01(pu)

Synchronous machines parameters: The same as A.1.

Excitation and governor system: The same as A.2.

Appendix C.

Data of IEEE 68-Bus, 16-Machine System [154]

C.1 Transmission Line Parameters

Table C.1 Transmission Line Parameters

No.	From bus	To bus	R (p.u.)	X (p.u.)	b (p.u.)	Tap ratio
1	53	54	0.007	0.0822	0.3493	
2	53	30	0.0008	0.0074	0.48	
3	54	55	0.0013	0.0151	0.2572	
4	54	25	0.007	0.0086	0.146	
5	54	1	0	0.0181	0	1.025
6	55	56	0.0013	0.0213	0.2214	
7	55	52	0.0011	0.0133	0.2138	
8	56	57	0.0008	0.0128	0.1342	
9	56	66	0.0008	0.0129	0.1382	
10	57	58	0.0002	0.0026	0.0434	
11	57	60	0.0008	0.0112	0.1476	
12	58	59	0.0006	0.0092	0.113	
13	58	63	0.0007	0.0082	0.1389	
14	58	2	0	0.025	0	1.07
15	59	60	0.0004	0.0046	0.078	
16	60	61	0.0023	0.0363	0.3804	
17	61	30	0.0019	0.0183	0.29	
18	62	63	0.0004	0.0043	0.0729	
19	62	65	0.0004	0.0043	0.0729	
20	62	3	0	0.02	0	1.07
21	64	63	0.0016	0.0435	0	1.06
22	64	65	0.0016	0.0435	0	1.06
23	65	66	0.0009	0.0101	0.1723	
24	66	67	0.0018	0.0217	0.366	
25	67	68	0.0009	0.0094	0.171	
26	68	37	0.0007	0.0089	0.1342	

27	68	19	0.0016	0.0195	0.304	
28	68	21	0.0008	0.0135	0.2548	
29	68	24	0.0003	0.0059	0.068	
30	37	52	0.0007	0.0082	0.1319	
31	37	27	0.0013	0.0173	0.3216	
32	19	20	0.0007	0.0138	0	1.06
33	19	4	0.0007	0.0142	0	1.07
34	20	5	0.0009	0.018	0	1.009
35	21	22	0.0008	0.014	0.2565	
36	22	23	0.0006	0.0096	0.1846	
37	22	6	0	0.0143	0	1.025
38	23	24	0.0022	0.035	0.361	
39	23	7	0.0005	0.0272	0	
40	25	26	0.0032	0.0323	0.531	
41	25	8	0.0006	0.0232	0	1.025
42	26	27	0.0014	0.0147	0.2396	
43	26	28	0.0043	0.0474	0.7802	
44	26	29	0.0057	0.0625	1.029	
45	28	29	0.0014	0.0151	0.249	
46	29	9	0.0008	0.0156	0	1.025
47	61	30	0.0019	0.0183	0.29	
48	61	36	0.0022	0.0196	0.34	
49	61	36	0.0022	0.0196	0.34	
50	36	17	0.0005	0.0045	0.32	
51	34	36	0.0033	0.0111	1.45	
52	35	34	0.0001	0.0074	0	0.946
53	33	34	0.0011	0.0157	0.202	
54	32	33	0.0008	0.0099	0.168	
55	30	31	0.0013	0.0187	0.333	
56	30	32	0.0024	0.0288	0.488	
57	53	31	0.0016	0.0163	0.25	
58	31	38	0.0011	0.0147	0.247	
59	33	38	0.0036	0.0444	0.693	
60	38	46	0.0022	0.0284	0.43	
61	46	49	0.0018	0.0274	0.27	
62	53	47	0.0013	0.0188	1.31	

63	47	48	0.0025	0.0268	0.4	
64	47	48	0.0025	0.0268	0.4	
65	48	40	0.002	0.022	1.28	
66	35	45	0.0007	0.0175	1.39	
67	17	43	0.0005	0.0276	0	
68	43	44	0.0001	0.0011	0	
69	44	45	0.0025	0.073	0	
70	39	44	0	0.0411	0	
71	39	45	0	0.0839	0	
72	45	51	0.0004	0.0105	0.72	
73	50	18	0.0012	0.0288	2.06	
74	50	51	0.0009	0.0221	1.62	
75	49	18	0.0076	0.1141	1.16	
76	18	42	0.004	0.06	2.25	
77	42	41	0.004	0.06	2.25	
78	41	40	0.006	0.084	3.15	
79	31	10	0	0.026	0	1.04
80	32	11	0	0.013	0	1.04
81	36	12	0	0.0075	0	1.04
82	17	13	0	0.0033	0	1.04
83	41	14	0	0.0015	0	1
84	42	15	0	0.0015	0	1
85	18	16	0	0.003	0	1
86	53	27	0.032	0.32	0.41	1

The base value for power is 100 MVA and for frequency it is 60 Hz.

C.2 Load Parameters

Table C.2 Load Parameters

Bus No.	Active Power (MW)	Reactive Power (MVar)
17	6000.00	300.00
18	2470.00	123.00
20	680.00	103.00
21	274.00	115.00

23	248.00	85.00
24	309.00	-92.00
25	224.00	47.00
26	139.00	17.00
27	281.00	76.00
28	206.00	28.00
29	284.00	27.00
33	112.00	0.00
36	102.00	-19.46
39	267.00	12.60
40	65.63	23.53
41	1000.00	250.00
42	1150.00	250.00
44	267.55	4.84
45	208.00	21.00
46	150.70	28.50
47	203.12	32.59
48	241.20	2.20
49	164.00	29.00
50	100.00	-147.00
51	337.00	-122.00
52	158.00	30.00
53	252.70	118.56
55	322.00	2.00
56	200.00	73.60
59	234.00	84.00

C.3 Generator Parameters

Table C.3 Generator Parameters

No.	Rated Power	X_{li}	R_{si}	X_{di}	X'_{di}	X''_{di}	T'_{doi}	T''_{doi}	X_{qi}	X'_{qi}	X''_{qi}	T'_{qoi}	T''_{qoi}	H_i
G1	2200	0.0125	0	1.8	0.56	0.45	10.2	0.05	1.24	0.5	0.45	1.5	0.035	2.33
G2	800	0.035	0	1.8	0.43	0.31	6.56	0.05	1.72	0.37	0.31	1.5	0.035	4.95
G3	800	0.0304	0	1.8	0.38	0.32	5.7	0.05	1.71	0.36	0.32	1.5	0.035	4.96
G4	800	0.0295	0	1.8	0.3	0.24	5.69	0.05	1.77	0.27	0.24	1.5	0.035	4.16
G5	700	0.027	0	1.8	0.36	0.27	5.4	0.05	1.69	0.33	0.27	0.44	0.035	4.77
G6	900	0.0224	0	1.8	0.35	0.28	7.3	0.05	1.71	0.32	0.28	0.4	0.035	4.91
G7	800	0.0322	0	1.8	0.3	0.24	5.66	0.05	1.78	0.27	0.24	1.5	0.035	4.33
G8	800	0.028	0	1.8	0.35	0.28	6.7	0.05	1.74	0.31	0.28	0.41	0.035	3.92
G9	1000	0.0298	0	1.8	0.49	0.38	4.79	0.05	1.75	0.43	0.38	1.96	0.035	4.04
G10	1200	0.0199	0	1.8	0.49	0.43	9.37	0.05	1.22	0.48	0.43	1.5	0.035	2.91
G11	1600	0.0103	0	1.8	0.25	0.17	4.1	0.05	1.73	0.21	0.17	1.5	0.035	2.01
G12	1900	0.022	0	1.8	0.55	0.45	7.4	0.05	1.69	0.5	0.45	1.5	0.035	5.18
G13	12000	0.003	0	1.8	0.33	0.24	5.9	0.05	1.74	0.3	0.24	1.5	0.035	4.08
G14	10000	0.0017	0	1.8	0.29	0.23	4.1	0.05	1.73	0.25	0.23	1.5	0.035	3
G15	10000	0.0017	0	1.8	0.29	0.23	4.1	0.05	1.73	0.25	0.23	1.5	0.035	3
G16	11000	0.0041	0	1.8	0.36	0.28	7.8	0.05	1.69	0.3	0.28	1.5	0.035	4.45

Appendix D. Xilinx ISE Design Suite Environment

D.1 Coding Environment

Figure D.1 shows the coding environment of FPGA design where the left hand side is the list of code files and the right hand side is the VHDL codes.

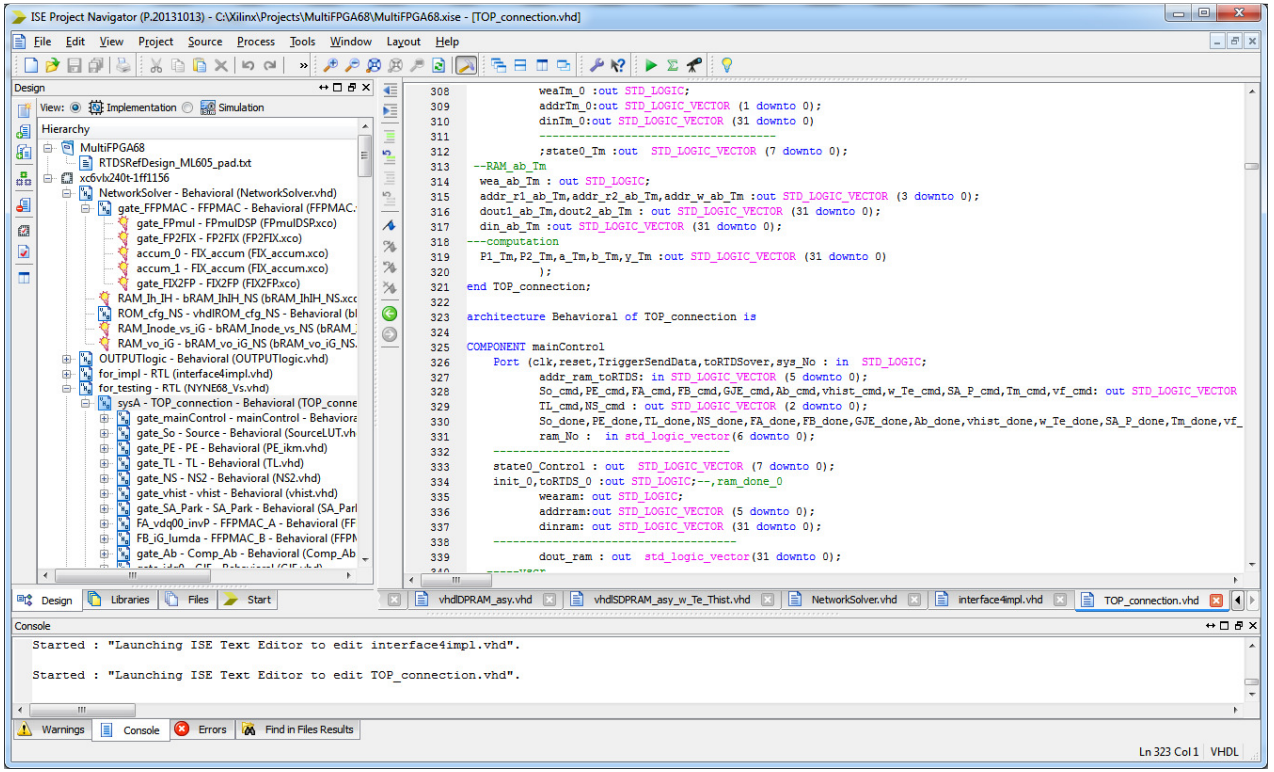


Figure D. 1 ISE coding environment

D.2 Design Reports Summary Window

Figure D.2 shows the summary of reports (for example the utilization rate of hardware resources, numbers of warnings and errors) after compiling the VHDL codes of hardware components.

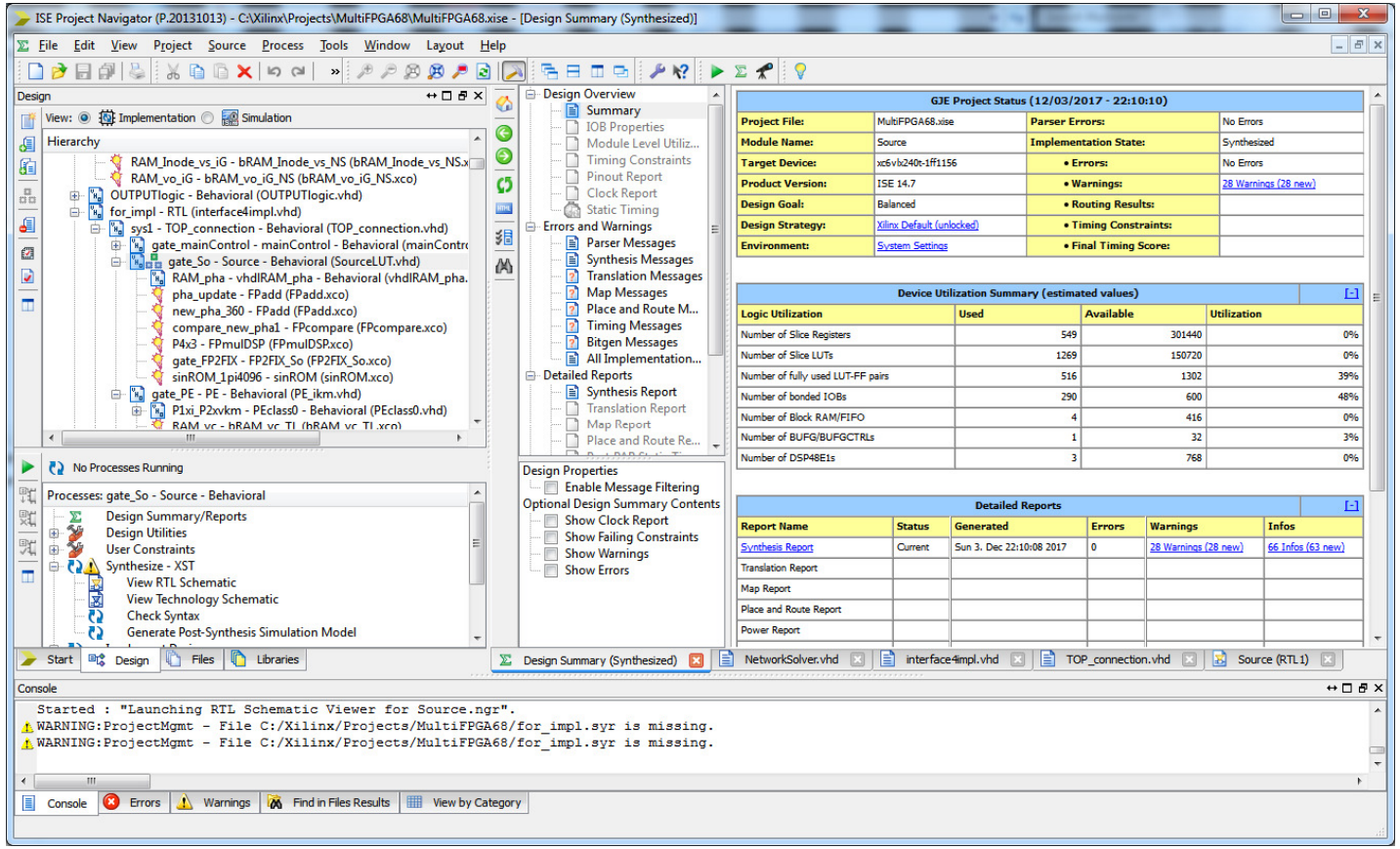


Figure D. 2 Design Summary Window

D.3 RTL (Register Transfer Level) Schematic Environment

Figure D.3 shows the RTL schematic view for FPGA design. RTL schematic shows the representation of hardware design in terms of logic elements such as adders, multipliers, AND gates and OR gates. It allows designer to visualize a gate-level representation of VHDL codes.

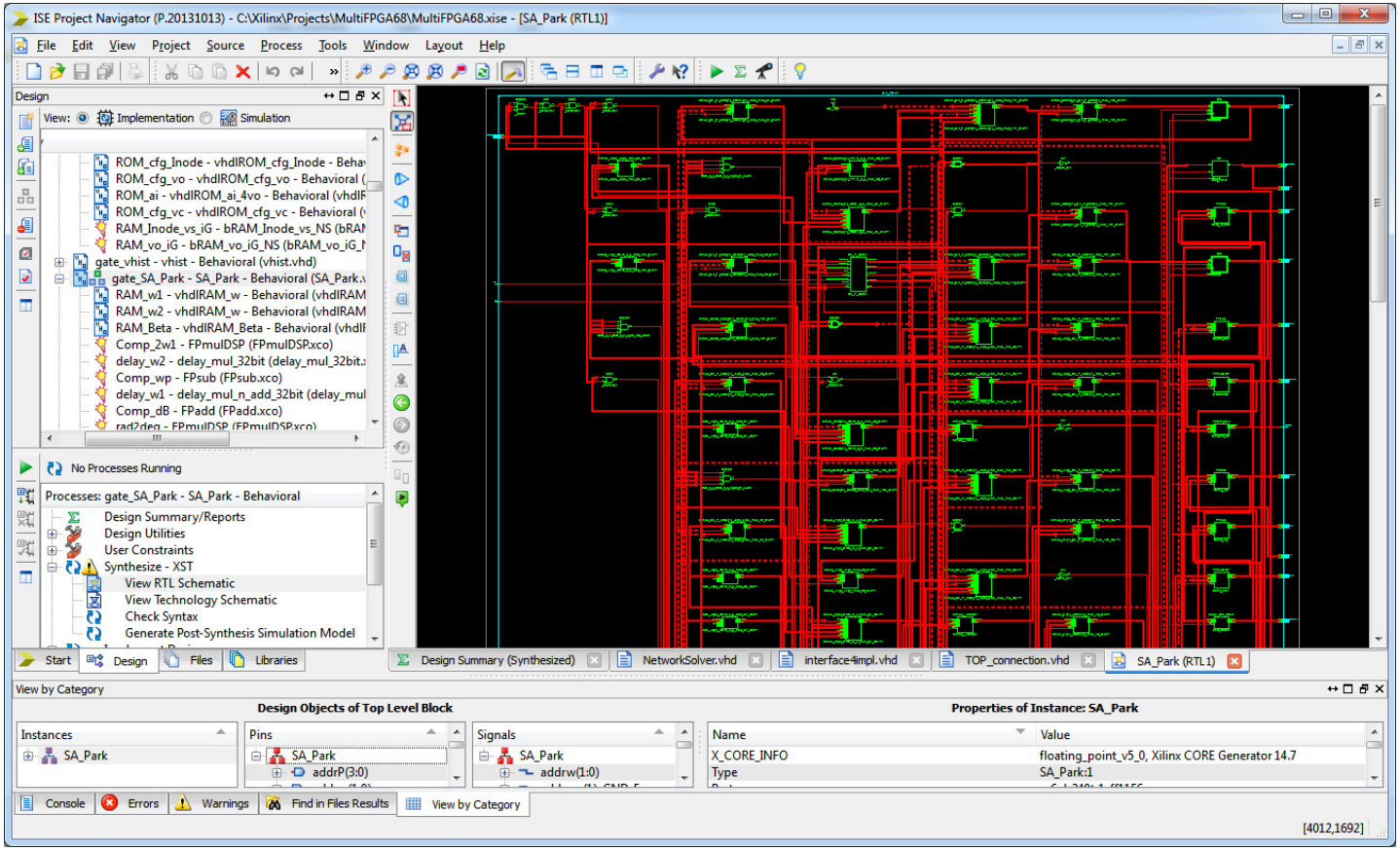


Figure D. 3 RTL schematic view

D.4 ISIM Simulation Environment

Figure D.4 shows the simulation environment of the hardware design using the built-in ISIM simulation tool from XILINX ISE. This simulation verifies the functionality and reports errors of the design before on-board testing.

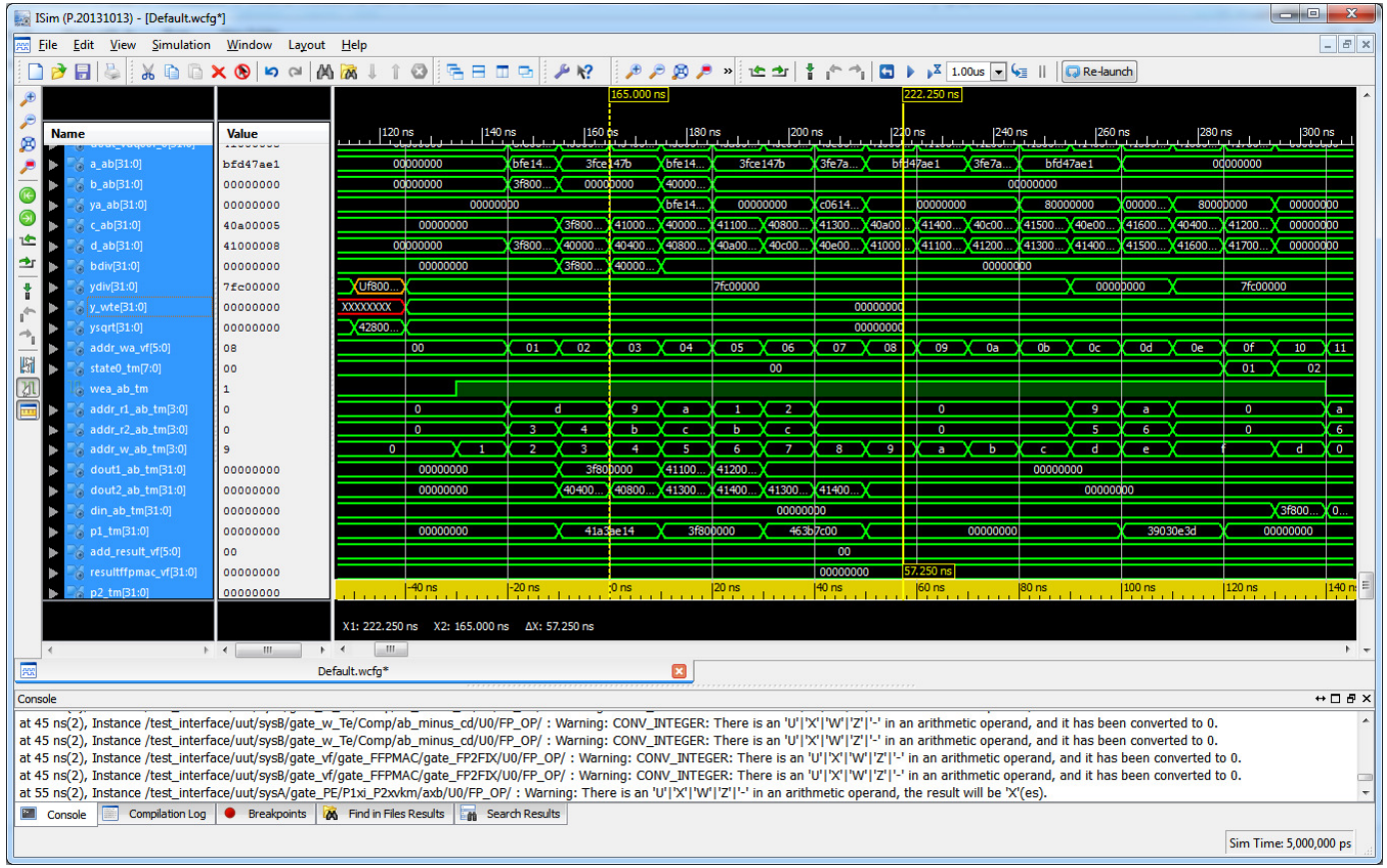
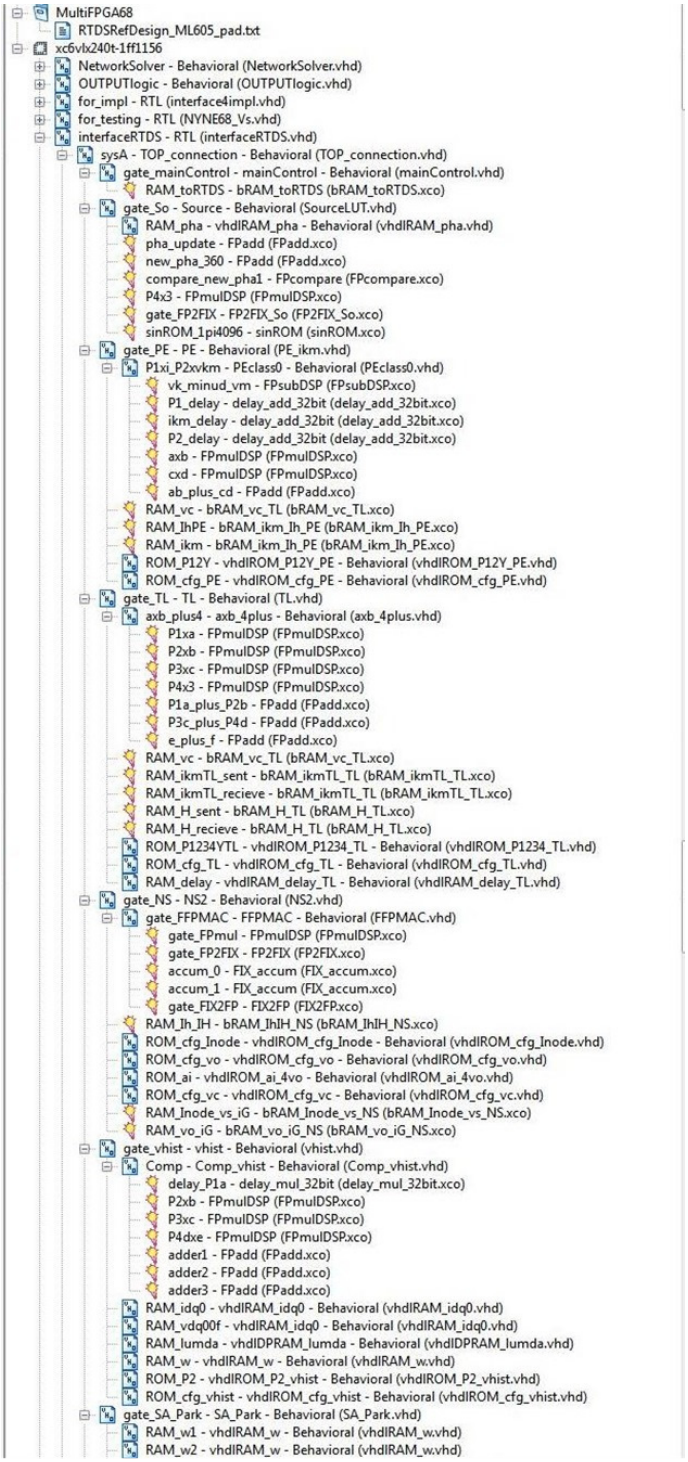


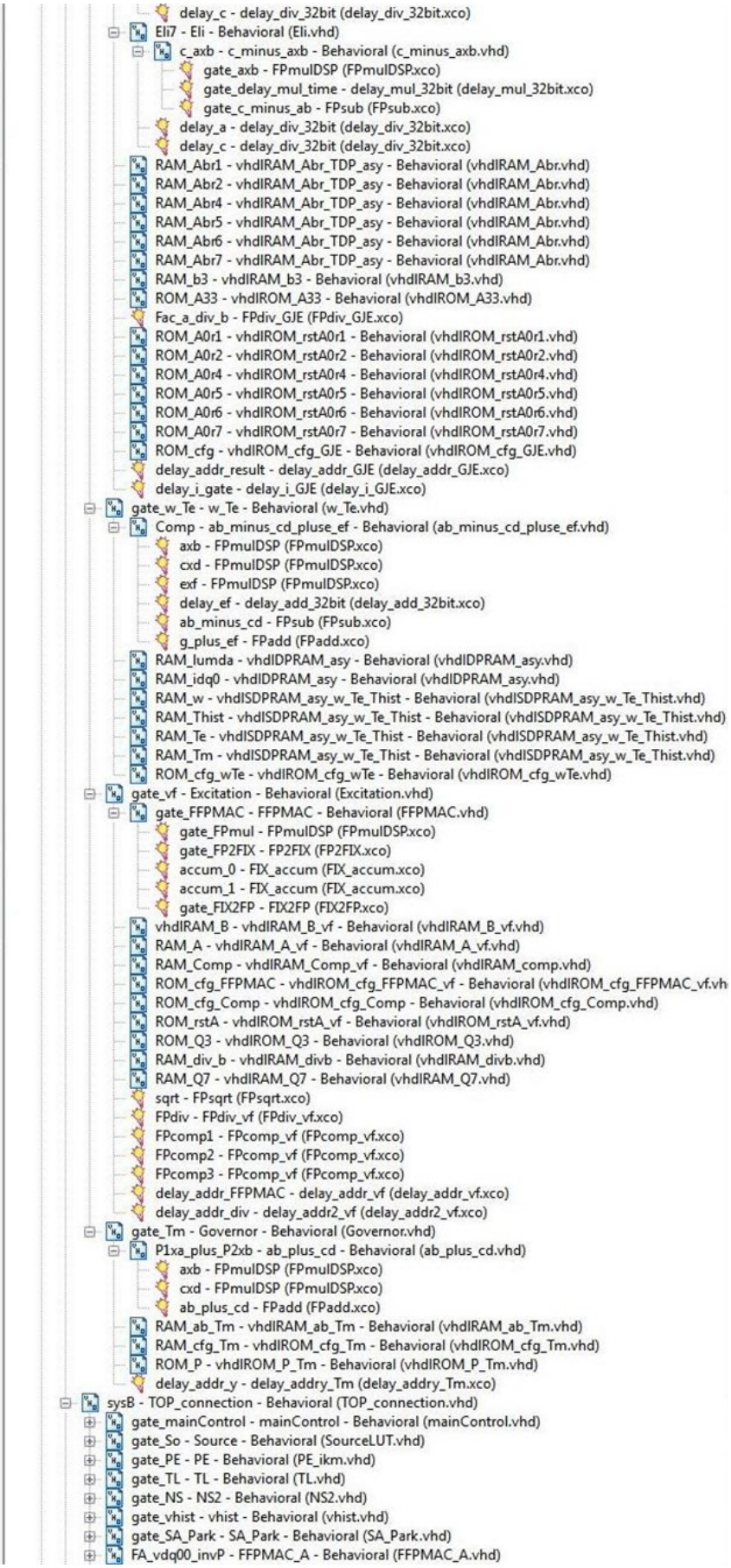
Figure D. 4 ISIM simulation environment

D.5 List of Main Code Files for Case Study 2 in Chapter 4

Figure D.5 shows the design structure in Xilinx ISE Project for case 2 in Chapter 4. Each file in the list represents a VHDL code file for one hardware module. One of the code files is shown in Appendix E as an example.




RAM_Beta - vhdlRAM_Beta - Behavioral (vhdlRAM_Beta.vhd)
Comp_2w1 - FPmulDSP (FPmulDSP.xco)
delay_w2 - delay_mul_32bit (delay_mul_32bit.xco)
Comp_wp - FPsub (FPsub.xco)
delay_w1 - delay_mul_n_add_32bit (delay_mul_n_add_32bit.xco)
Comp_dB - FPadd (FPadd.xco)
rad2deg - FPmulDSP (FPmulDSP.xco)
Beta_update - FPadd (FPadd.xco)
Beta0_360 - FPsub (FPsub.xco)
delayBeta0 - delay_add_32bit (delay_add_32bit.xco)
compare_B1 - FPcompare (FPcompare.xco)
Beta2addr - FPmulDSP (FPmulDSP.xco)
gate_FP2FIX - FP2FIX_So (FP2FIX_So.xco)
sinROM_1pi4096_06667 - sinROM_Park (sinROM_Park.xco)
ROM_iniBeta - vhdlROM_iniBeta - Behavioral (vhdlROM_iniBeta.vhd)
FA_vdq00_invP - FFPMAC_A - Behavioral (FFPMAC_A.vhd)
gate_FFPMAC - FFPMAC - Behavioral (FFPMAC.vhd)
gate_FPmul - FPmulDSP (FPmulDSP.xco)
gate_FP2FIX - FP2FIX (FP2FIX.xco)
accum_0 - FIX_accum (FIX_accum.xco)
accum_1 - FIX_accum (FIX_accum.xco)
gate_FIX2FP - FIX2FP (FIX2FP.xco)
RAM_P - vhdlRAM_P - Behavioral (vhdlRAM_P.vhd)
RAM_vo - vhdlRAM_vo - Behavioral (vhdlRAM_vo.vhd)
ROM_cfg_vdq00 - vhdlROM_cfg_vdq00 - Behavioral (vhdlROM_cfg_vdq00.vhd)
ROM_C - vhdlROM_C - Behavioral (vhdlROM_C.vhd)
ROM_cfg_invP - vhdlROM_cfg_invP - Behavioral (vhdlROM_cfg_invP.vhd)
FB_iG_lumda - FFPMAC_B - Behavioral (FFPMAC_B.vhd)
gate_FFPMAC - FFPMAC - Behavioral (FFPMAC.vhd)
gate_FPmul - FPmulDSP (FPmulDSP.xco)
gate_FP2FIX - FP2FIX (FP2FIX.xco)
accum_0 - FIX_accum (FIX_accum.xco)
accum_1 - FIX_accum (FIX_accum.xco)
gate_FIX2FP - FIX2FP (FIX2FP.xco)
RAM_invP - vhdlRAM_invP - Behavioral (vhdlRAM_invP.vhd)
RAM_idq0 - vhdlRAM_idq0 - Behavioral (vhdlRAM_idq0.vhd)
ROM_cfg_iG - vhdlROM_cfg_iG - Behavioral (vhdlROM_cfg_iG.vhd)
ROM_LG - vhdlROM_LG - Behavioral (vhdlROM_LG.vhd)
ROM_cfg_lumda - vhdlROM_cfg_lumda - Behavioral (vhdlROM_cfg_lumda.vhd)
gate_Ab - Comp_Ab - Behavioral (Comp_Ab.vhd)
Comp_A - FPmulDSP (FPmulDSP.xco)
RAM_wp - vhdlRAM_w - Behavioral (vhdlRAM_w.vhd)
ROM_cfg_A - vhdlROM_cfg_A - Behavioral (vhdlROM_cfg_A.vhd)
gate_c_minus_d - FPsub (FPsub.xco)
RAM_vdq00f - vhdlRAM_idq0 - Behavioral (vhdlRAM_idq0.vhd)
RAM_vhist - vhdlRAM_idq0 - Behavioral (vhdlRAM_idq0.vhd)
ROM_cfg_b - vhdlROM_cfg_b - Behavioral (vhdlROM_cfg_b.vhd)
gate_idq0 - GJE - Behavioral (GJE.vhd)
Eli1 - Eli - Behavioral (Eli.vhd)
c_axb - c_minus_axb - Behavioral (c_minus_axb.vhd)
gate_axb - FPmulDSP (FPmulDSP.xco)
gate_delay_mul_time - delay_mul_32bit (delay_mul_32bit.xco)
gate_c_minus_ab - FPsub (FPsub.xco)
delay_a - delay_div_32bit (delay_div_32bit.xco)
delay_c - delay_div_32bit (delay_div_32bit.xco)
Eli2 - Eli - Behavioral (Eli.vhd)
c_axb - c_minus_axb - Behavioral (c_minus_axb.vhd)
gate_axb - FPmulDSP (FPmulDSP.xco)
gate_delay_mul_time - delay_mul_32bit (delay_mul_32bit.xco)
gate_c_minus_ab - FPsub (FPsub.xco)
delay_a - delay_div_32bit (delay_div_32bit.xco)
delay_c - delay_div_32bit (delay_div_32bit.xco)
Eli4 - Eli - Behavioral (Eli.vhd)
c_axb - c_minus_axb - Behavioral (c_minus_axb.vhd)
gate_axb - FPmulDSP (FPmulDSP.xco)
gate_delay_mul_time - delay_mul_32bit (delay_mul_32bit.xco)
gate_c_minus_ab - FPsub (FPsub.xco)
delay_a - delay_div_32bit (delay_div_32bit.xco)
delay_c - delay_div_32bit (delay_div_32bit.xco)
Eli5 - Eli - Behavioral (Eli.vhd)
c_axb - c_minus_axb - Behavioral (c_minus_axb.vhd)
gate_axb - FPmulDSP (FPmulDSP.xco)
gate_delay_mul_time - delay_mul_32bit (delay_mul_32bit.xco)
gate_c_minus_ab - FPsub (FPsub.xco)
delay_a - delay_div_32bit (delay_div_32bit.xco)
delay_c - delay_div_32bit (delay_div_32bit.xco)
Eli6 - Eli - Behavioral (Eli.vhd)
c_axb - c_minus_axb - Behavioral (c_minus_axb.vhd)
gate_axb - FPmulDSP (FPmulDSP.xco)
gate_delay_mul_time - delay_mul_32bit (delay_mul_32bit.xco)
gate_c_minus_ab - FPsub (FPsub.xco)
delay_a - delay_div_32bit (delay_div_32bit.xco)


delay_c - delay_div_32bit (delay_div_32bit.xco)
Eli7 - Eli - Behavioral (Eli.vhd)
c_axb - c_minus_axb - Behavioral (c_minus_axb.vhd)
gate_axb - FPmulDSP (FPmulDSP.xco)
gate_delay_mul_time - delay_mul_32bit (delay_mul_32bit.xco)
gate_c_minus_ab - FPsub (FPsub.xco)
delay_a - delay_div_32bit (delay_div_32bit.xco)
delay_c - delay_div_32bit (delay_div_32bit.xco)
RAM_Abr1 - vhdlRAM_Abr_TDP_asy - Behavioral (vhdlRAM_Abr.vhd)
RAM_Abr2 - vhdlRAM_Abr_TDP_asy - Behavioral (vhdlRAM_Abr.vhd)
RAM_Abr4 - vhdlRAM_Abr_TDP_asy - Behavioral (vhdlRAM_Abr.vhd)
RAM_Abr5 - vhdlRAM_Abr_TDP_asy - Behavioral (vhdlRAM_Abr.vhd)
RAM_Abr6 - vhdlRAM_Abr_TDP_asy - Behavioral (vhdlRAM_Abr.vhd)
RAM_Abr7 - vhdlRAM_Abr_TDP_asy - Behavioral (vhdlRAM_Abr.vhd)
RAM_b3 - vhdlRAM_b3 - Behavioral (vhdlRAM_b3.vhd)
ROM_A33 - vhdlROM_A33 - Behavioral (vhdlROM_A33.vhd)
Fac_a_div_b - FPdiv_GJE (FPdiv_GJE.xco)
ROM_A0r1 - vhdlROM_rstA0r1 - Behavioral (vhdlROM_rstA0r1.vhd)
ROM_A0r2 - vhdlROM_rstA0r2 - Behavioral (vhdlROM_rstA0r2.vhd)
ROM_A0r4 - vhdlROM_rstA0r4 - Behavioral (vhdlROM_rstA0r4.vhd)
ROM_A0r5 - vhdlROM_rstA0r5 - Behavioral (vhdlROM_rstA0r5.vhd)
ROM_A0r6 - vhdlROM_rstA0r6 - Behavioral (vhdlROM_rstA0r6.vhd)
ROM_A0r7 - vhdlROM_rstA0r7 - Behavioral (vhdlROM_rstA0r7.vhd)
ROM_cfg - vhdlROM_cfg_GJE - Behavioral (vhdlROM_cfg_GJE.vhd)
delay_addr_result - delay_addr_GJE (delay_addr_GJE.xco)
delay_i_gate - delay_i_GJE (delay_i_GJE.xco)
gate_w_Te - w_Te - Behavioral (w_Te.vhd)
Comp - ab_minus_cd_pluse_ef - Behavioral (ab_minus_cd_pluse_ef.vhd)
axb - FPmulDSP (FPmulDSP.xco)
cxd - FPmulDSP (FPmulDSP.xco)
exf - FPmulDSP (FPmulDSP.xco)
delay_ef - delay_add_32bit (delay_add_32bit.xco)
ab_minus_cd - FPsub (FPsub.xco)
g_plus_ef - FPadd (FPadd.xco)
RAM_lumda - vhdlDPRAM_asy - Behavioral (vhdlDPRAM_asy.vhd)
RAM_idq0 - vhdlDPRAM_asy - Behavioral (vhdlDPRAM_asy.vhd)
RAM_w - vhdlSDPRAM_asy_w_Te_Thist - Behavioral (vhdlSDPRAM_asy_w_Te_Thist.vhd)
RAM_Thist - vhdlSDPRAM_asy_w_Te_Thist - Behavioral (vhdlSDPRAM_asy_w_Te_Thist.vhd)
RAM_Te - vhdlSDPRAM_asy_w_Te_Thist - Behavioral (vhdlSDPRAM_asy_w_Te_Thist.vhd)
RAM_Tm - vhdlSDPRAM_asy_w_Te_Thist - Behavioral (vhdlSDPRAM_asy_w_Te_Thist.vhd)
ROM_cfg_wTe - vhdlROM_cfg_wTe - Behavioral (vhdlROM_cfg_wTe.vhd)
gate_vf - Excitation - Behavioral (Excitation.vhd)
gate_FFPMAC - FFPMAC - Behavioral (FFPMAC.vhd)
gate_FPmul - FPmulDSP (FPmulDSP.xco)
gate_FP2FIX - FP2FIX (FP2FIX.xco)
accum_0 - FIX_accum (FIX_accum.xco)
accum_1 - FIX_accum (FIX_accum.xco)
gate_FIX2FP - FIX2FP (FIX2FP.xco)
vhdlRAM_B - vhdlRAM_B_vf - Behavioral (vhdlRAM_B_vf.vhd)
RAM_A - vhdlRAM_A_vf - Behavioral (vhdlRAM_A_vf.vhd)
RAM_Comp - vhdlRAM_Comp_vf - Behavioral (vhdlRAM_comp.vhd)
ROM_cfg_FFPMAC - vhdlROM_cfg_FFPMAC_vf - Behavioral (vhdlROM_cfg_FFPMAC_vf.vh
ROM_cfg_Comp - vhdlROM_cfg_Comp - Behavioral (vhdlROM_cfg_Comp.vhd)
ROM_rstA - vhdlROM_rstA_vf - Behavioral (vhdlROM_rstA_vf.vhd)
ROM_Q3 - vhdlROM_Q3 - Behavioral (vhdlROM_Q3.vhd)
RAM_div_b - vhdlRAM_divb - Behavioral (vhdlRAM_divb.vhd)
RAM_Q7 - vhdlRAM_Q7 - Behavioral (vhdlRAM_Q7.vhd)
sqrt - FPsqrt (FPsqrt.xco)
FPdiv - FPdiv_vf (FPdiv_vf.xco)
FPcomp1 - FPcomp_vf (FPcomp_vf.xco)
FPcomp2 - FPcomp_vf (FPcomp_vf.xco)
FPcomp3 - FPcomp_vf (FPcomp_vf.xco)
delay_addr_FFPMAC - delay_addr_vf (delay_addr_vf.xco)
delay_addr_div - delay_addr2_vf (delay_addr2_vf.xco)
gate_Tm - Governor - Behavioral (Governor.vhd)
P1xa_plus_P2xb - ab_plus_cd - Behavioral (ab_plus_cd.vhd)
axb - FPmulDSP (FPmulDSP.xco)
cxd - FPmulDSP (FPmulDSP.xco)
ab_plus_cd - FPadd (FPadd.xco)
RAM_ab_Tm - vhdlRAM_ab_Tm - Behavioral (vhdlRAM_ab_Tm.vhd)
RAM_cfg_Tm - vhdlROM_cfg_Tm - Behavioral (vhdlROM_cfg_Tm.vhd)
ROM_P - vhdlROM_P_Tm - Behavioral (vhdlROM_P_Tm.vhd)
delay_addr_y - delay_addry_Tm (delay_addry_Tm.xco)
sysB - TOP_connection - Behavioral (TOP_connection.vhd)
gate_mainControl - mainControl - Behavioral (mainControl.vhd)
gate_So - Source - Behavioral (SourceLUT.vhd)
gate_PE - PE - Behavioral (PE_ikm.vhd)
gate_TL - TL - Behavioral (TL.vhd)
gate_NS - NS2 - Behavioral (NS2.vhd)
gate_vhist - vhist - Behavioral (vhist.vhd)
gate_SA_Park - SA_Park - Behavioral (SA_Park.vhd)
FA_vdq00_invP - FFPMAC_A - Behavioral (FFPMAC_A.vhd)

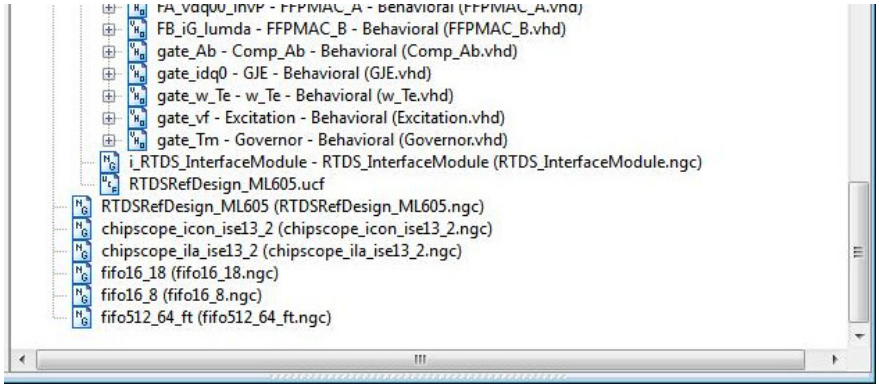


Figure D. 5 FPGA project design structure

Appendix E.

Sample VHDL Codes

E.1 Upper Layer Codes of the Global Controller Module

The top layer codes with comments for the global controller module (described in Section 3.3) are shown below as an example. The codes are used to design the finite-state machine (FSM) shown in Figure 3-5.

```
----------------------------------------------------------------------------------
-- Company: University of Birmingham
-- Engineer: Conghuan Yang
-- Create Date:     12:32:18 03/25/2017
-- Design Name:     Global Controller
-- Module Name:     mainControl - Behavioral
-- Project Name:    Co-simulation System
-- Target Devices:  Xilinx Virtex 6
-- Revision 0.01 - File Created
-- Additional Comments:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity mainControl is
    Port ( clk,reset,TriggerSendData,toRTDSover,sys_No : in  STD_LOGIC;
           addr_ram_toRTDS: in STD_LOGIC_VECTOR (5 downto 0);
So_cmd,PE_cmd,FA_cmd,FB_cmd,GJE_cmd,Ab_cmd,vhist_cmd,w_Te_cmd,SA_P_cmd,Tm_cmd,vf_cmd: out
STD_LOGIC_VECTOR (1 downto 0);
           TL_cmd,NS_cmd : out STD_LOGIC_VECTOR (2 downto 0);
So_done,PE_done,TL_done,NS_done,FA_done,FB_done,GJE_done,Ab_done,vhist_done,w_Te_done,SA_P_don
e,Tm_done,vf_done: in STD_LOGIC;
           ram_No :  in std_logic_vector(6 downto 0);
           dout_ram : out  std_logic_vector(31 downto 0);
     ------------------------------------
     state0_Control : out  STD_LOGIC_VECTOR (7 downto 0);
    init_0,toRTDS_0 :out STD_LOGIC;--,ram_done_0
           wearam: out STD_LOGIC;
           addrram:out STD_LOGIC_VECTOR (5 downto 0);
           dinram: out STD_LOGIC_VECTOR (31 downto 0);
     ------------------------------------
    -----vscr
           weaVscr : in STD_LOGIC_VECTOR (0 downto 0);
           addrVscr : in STD_LOGIC_VECTOR (2 downto 0);
           dinVscr : in STD_LOGIC_VECTOR (31 downto 0) ;
    -----PE
         ---output Ih
           weaIh : in STD_LOGIC_VECTOR (0 downto 0);
           addrIh: in STD_LOGIC_VECTOR(7 DOWNTO 0);
           dinIh: in STD_LOGIC_VECTOR(31 DOWNTO 0);
         ---output  ikm
```

```
            weaiPE : in STD_LOGIC_VECTOR (0 downto 0);
            addriPE: in STD_LOGIC_VECTOR(7 DOWNTO 0);
            diniPE: in STD_LOGIC_VECTOR(31 DOWNTO 0);
    ------TL
          addrH : in std_logic_vector(7 downto 0);
          weaHs,weaHr: in  STD_LOGIC_VECTOR (0 downto 0);
          HTL,IHTL_s,IHTL_r: in std_logic_vector(31 downto 0);
        ----RAM_ikmTL
            weair: in STD_LOGIC_VECTOR (0 downto 0);
            addrir: in  STD_LOGIC_VECTOR (7 downto 0);
            dinir: in  STD_LOGIC_VECTOR (31 downto 0) ;
            weais: in  STD_LOGIC_VECTOR (0 downto 0);
            addris: in  STD_LOGIC_VECTOR (7 downto 0);
            dinis: in  STD_LOGIC_VECTOR (31 downto 0);
    -----delay ram
           addrface :in  STD_LOGIC_VECTOR (3 downto 0);--from
         weaface : in  STD_LOGIC_VECTOR (0 downto 0);
         dinface : in  STD_LOGIC_VECTOR (31 downto 0);
          addr_Rxsys : in  STD_LOGIC_VECTOR (3 downto 0);--from
         wea_Rxsys : in  STD_LOGIC_VECTOR (0 downto 0);
         din_Rxsys : in  STD_LOGIC_VECTOR (31 downto 0);
    ------NS
        weavo : in STD_LOGIC_VECTOR (0 downto 0);
      addrvo: in STD_LOGIC_VECTOR(7 DOWNTO 0);
      dinvo: in STD_LOGIC_VECTOR(31 DOWNTO 0) ;
        weavc : in STD_LOGIC_VECTOR (0 downto 0);
      addrvc: in STD_LOGIC_VECTOR(7 DOWNTO 0);
      dinvc: in STD_LOGIC_VECTOR(31 DOWNTO 0) ;
    --RAM_Inode
      wea_Inode :in STD_LOGIC_VECTOR (0 downto 0);
      addr_Inode:in STD_LOGIC_VECTOR(7 DOWNTO 0);
      din_Inode :in STD_LOGIC_VECTOR(31 DOWNTO 0)
------RAM_IH_Ih
----       wea_Ih_NS : in STD_LOGIC_VECTOR (0 downto 0);
----       addr_Ih_NS:in STD_LOGIC_VECTOR(9 DOWNTO 0);
----       din_Ih_NS :in STD_LOGIC_VECTOR(31 DOWNTO 0);
------vhist
            ; weavhist : in STD_LOGIC ;
             addrvhist :in STD_LOGIC_VECTOR (3 downto 0);
             dinvhist : in STD_LOGIC_VECTOR (31 downto 0);
------SA_P
             weaP : in STD_LOGIC ;
             addrP : in STD_LOGIC_VECTOR (3 downto 0);
             dinP : in STD_LOGIC_VECTOR (31 downto 0);
-----FA-invP
             weainvP : in STD_LOGIC ;
             addrinvP: in STD_LOGIC_VECTOR (4 downto 0);
             dininvP : in STD_LOGIC_VECTOR (31 downto 0);
-----FB -iabc,lumda, invP
             weaiG : in STD_LOGIC_VECTOR (0 downto 0);
             addriG: in STD_LOGIC_VECTOR (2 downto 0);
             diniG : in STD_LOGIC_VECTOR (31 downto 0) ;
             wealumda : in STD_LOGIC ;
             addrlumda: in STD_LOGIC_VECTOR (3 downto 0);
             dinlumda : in STD_LOGIC_VECTOR (31 downto 0)  ;
---Ab-b  vdq00f
         send_vdq00 : in STD_LOGIC ;
             send_addrvdq00 : in STD_LOGIC_VECTOR (3 downto 0);
             send_dinvdq00 : in STD_LOGIC_VECTOR (31 downto 0);
             weaAbr1,weaAbr2: in  STD_LOGIC;
             addrAbr : in STD_LOGIC_VECTOR (3 downto 0);
             dinAbr : in STD_LOGIC_VECTOR (31 downto 0);
----idq0,
             weaidq0 : in  STD_LOGIC;
             addridq0 : in STD_LOGIC_VECTOR (3 downto 0);
             dinidq0 : in STD_LOGIC_VECTOR (31 downto 0)  ;
-----w_Te_Thist
             weaw : in STD_LOGIC ;
             addrw : in STD_LOGIC_VECTOR (1 downto 0);
             dinw : in STD_LOGIC_VECTOR (31 downto 0);
        weaTe :in STD_LOGIC;
        addrTe:in STD_LOGIC_VECTOR (1 downto 0);
        dinTe:in STD_LOGIC_VECTOR (31 downto 0);
        weaThist :in STD_LOGIC;
        addrThist:in STD_LOGIC_VECTOR (1 downto 0);
        dinThist:in STD_LOGIC_VECTOR (31 downto 0)
    ---vf
```

```
        ;send_B_vf:in STD_LOGIC;
     send_addr_B_vf : in STD_LOGIC_VECTOR(5 DOWNTO 0);
    send_doutB_vf : in STD_LOGIC_VECTOR(31 DOWNTO 0)
    ---Tm
    ; send_ab_Tm : in STD_LOGIC;
  send_addr_ab_Tm :in STD_LOGIC_VECTOR (3 downto 0);
  send_dinab_Tm :in STD_LOGIC_VECTOR (31 downto 0)

            );
end mainControl;

architecture Behavioral of mainControl is

COMPONENT bRAM_toRTDS
  PORT (
    clka : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END COMPONENT;
---RAM_toRTDS
signal wea_ram:     STD_LOGIC_VECTOR (0 downto 0);
signal addr_ram :   STD_LOGIC_VECTOR (5 downto 0);
signal din_ram:     STD_LOGIC_VECTOR (31 downto 0)  ;--,dout_ram

type   state_type is    (s_rst,s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11);--,s12,s2_2
signal state         : state_type:=s0;
signal init,toRTDS: STD_LOGIC;--,ram_done
signal J1 : STD_LOGIC_VECTOR(5 DOWNTO 0):=(others=>'0');

begin
--============testing=========================
init_0<=init; toRTDS_0<=toRTDS; --ram_done_0 <=ram_done;
wearam<=wea_ram(0); addrram<=addr_ram;dinram<=din_ram;
process(state)
begin
    if state=s0 then state0_Control<=x"00";
    elsif state=s1 then state0_Control<=x"01";
    elsif state=s2 then state0_Control<=x"02";
    elsif state=s3 then state0_Control<=x"03";
    elsif state=s4 then state0_Control<=x"04";
    elsif state=s5 then state0_Control<=x"05";
    elsif state=s6 then state0_Control<=x"06";
    elsif state=s7 then state0_Control<=x"07";
    elsif state=s8 then state0_Control<=x"08";
    elsif state=s9 then state0_Control<=x"09";
    elsif state=s10 then state0_Control<=x"0a";
    elsif state=s11 then state0_Control<=x"0b";
--  elsif state=s12 then state0_Control<=x"0c";
    elsif state=s_rst then state0_Control<=x"FF";
    else state0_Control<=x"10";
    end if;
end process;
--============testing=========================

process(clk,reset)
begin

if reset='1' then
    state<=s_rst;
    TL_cmd<=(others=>'0'); PE_cmd<=(others=>'0');NS_cmd<=(others=>'0'); So_cmd<=(others=>'0');
    FA_cmd<=(others=>'0');FB_cmd<=(others=>'0');GJE_cmd<=(others=>'0');Ab_cmd<=(others=>'0');
    vhist_cmd<=(others=>'0');w_Te_cmd<=(others=>'0');SA_P_cmd<=(others=>'0');
    Tm_cmd<=(others=>'0');vf_cmd<=(others=>'0');
    init<='1';
    J1<=(others=>'0'); toRTDS<='0';--ram_done<='0';

elsif clk'event and clk='1' then
----=================================================
----                    initializing
----=================================================
        case state is
        when s_rst=> --reset initializing
    TL_cmd<=(others=>'1'); PE_cmd<=(others=>'1');So_cmd<=(others=>'1');
```

```
FA_cmd<=(others=>'1');FB_cmd<=(others=>'1');GJE_cmd<=(others=>'1');Ab_cmd<=(others=>'1');SA_P_
cmd<=(others=>'1');

vhist_cmd<=(others=>'1');w_Te_cmd<=(others=>'1');Tm_cmd<=(others=>'1');vf_cmd<=(others=>'1');
    NS_cmd<=(others=>'0');
            toRTDS<='0';init<='1';
            if TL_done= '1' and PE_done='1'  then ---ram IHTL 945T
                state<=s0;
            end if;
            J1<=J1+1;
--============================================================
--                    running--Main Loop                    --
--============================================================
        when s0=> --wait
        J1<=(others=>'0');init<='0';--ram_done<='0';
            So_cmd<="00";
            TL_cmd<="000";
            PE_cmd<="00";
            NS_cmd<="000";
            FA_cmd<="00";
            FB_cmd<="00";
            GJE_cmd<="00";
            Ab_cmd<="00";
            vhist_cmd<="00";
            w_Te_cmd<="00";
            SA_P_cmd<="00";
            Tm_cmd<="00";
            vf_cmd<="00";
            IF TriggerSendData='1' THEN
                toRTDS<='1';    --send to RTDS flag for RAM
            END IF;
            if toRTDSover='1' then
                toRTDS<='0';  state<=s1;
            end if;


        when s1=>
            So_cmd<="00";
            TL_cmd<="000";
            PE_cmd<="01";   --PE Ih
            NS_cmd<="000";
            vhist_cmd<="01"; --vhist
            SA_P_cmd<="00";
            FA_cmd<="00";
            FB_cmd<="00";
            GJE_cmd<="00";
            Ab_cmd<="00";
            w_Te_cmd<="00";
            Tm_cmd<="00";
            vf_cmd<="00";
            if PE_done='1' and vhist_done='1' then
                state<=s2;
            end if;

        when s2=>
            So_cmd<="00";
            TL_cmd<="001"; --IHTL
            PE_cmd<="00";
            NS_cmd<="000";
            vhist_cmd<="00";
            SA_P_cmd<="00";
            FA_cmd<="00";
            FB_cmd<="00";
            GJE_cmd<="00";
            Ab_cmd<="00";
            w_Te_cmd<="00";
            Tm_cmd<="00";
            vf_cmd<="01";   --vf
            if TL_done='1' and vf_done='1' then
                state<=s3; --TL_cmd<="000";
            end if;
        when s3=>
            So_cmd<="01";  ---Vscr
            TL_cmd<="000";
            PE_cmd<="00";
            NS_cmd<="001"; --Inode
```

```
        vhist_cmd<="00";
        Ab_cmd<="00";
        SA_P_cmd<="01";  --SA_P
        FA_cmd<="00";
        w_Te_cmd<="00";
        Tm_cmd<="00";
        vf_cmd<="00";
        if NS_done='1' and SA_P_done='1' and So_done='1' then
            state<=s4; NS_cmd<="000";
        end if;
    when s4=>
        So_cmd<="00";
        TL_cmd<="000";
        PE_cmd<="00";
        NS_cmd<="010"; --vo
        vhist_cmd<="00";
        SA_P_cmd<="00";
        FA_cmd<="00";
        Ab_cmd<="01"; --A
        GJE_cmd<="00";
        w_Te_cmd<="00";
        Tm_cmd<="00";
        vf_cmd<="00";
        if NS_done='1' and Ab_done='1' then
            state<=s5;
        end if;
    when s5=>
        So_cmd<="00";
        TL_cmd<="000";
        PE_cmd<="00";
        NS_cmd<="000";
        vhist_cmd<="00";
        SA_P_cmd<="00";
        FA_cmd<="01"; --vdq0_0
        Ab_cmd<="00";
        FB_cmd<="00";
        if FA_done='1' then
            state<=s6;
        end if;

    when s6=>
        So_cmd<="00";
        TL_cmd<="000";
        PE_cmd<="00";
        NS_cmd<="000";
        vhist_cmd<="00";
        SA_P_cmd<="00";
        FA_cmd<="00";
        Ab_cmd<="10"; --b
        FB_cmd<="00";
        GJE_cmd<="00";
        w_Te_cmd<="00";
        Tm_cmd<="00";
        vf_cmd<="00";
        if Ab_done='1'  then
            state<=s7;
        end if;

    when s7=>
        So_cmd<="00";
        TL_cmd<="000";
        vhist_cmd<="00";
        SA_P_cmd<="00";
        FA_cmd<="10"; ---invP
        Ab_cmd<="00";
        FB_cmd<="00";
        GJE_cmd<="01"; ---idq0
        w_Te_cmd<="00";
        Tm_cmd<="00";
        vf_cmd<="00";
        if FA_done='1' and GJE_done='1' then
            state<=s8;
        end if;

    when s8=>
        So_cmd<="00";
        TL_cmd<="000";
```

```
            PE_cmd<="00";
            NS_cmd<="000";
            vhist_cmd<="00";
            SA_P_cmd<="00";
            FA_cmd<="00";
            FB_cmd<="01"; --iabc_G
            GJE_cmd<="00";
            w_Te_cmd<="00";
            Tm_cmd<="01"; --Tm
            vf_cmd<="00";
            if Tm_done='1' and FB_done='1' then
                state<=s9; FB_cmd<="00";
            end if;
        when s9=>
            So_cmd<="00";
            TL_cmd<="000";
            PE_cmd<="00";
            NS_cmd<="011"; ---vc
            vhist_cmd<="00";
            SA_P_cmd<="00";
            FA_cmd<="00";
            Ab_cmd<="00";
            FB_cmd<="10"; --lumda
            vf_cmd<="00";
            if NS_done='1' and FB_done='1' then
                state<=s10;
            end if;
        when s10=>
            So_cmd<="00";
            TL_cmd<="010"; --ikmTL
            PE_cmd<="10";  --ikmPE
            NS_cmd<="000";
            vhist_cmd<="00";
            SA_P_cmd<="00";
            FA_cmd<="00";
            Ab_cmd<="00";
            FB_cmd<="00";
            GJE_cmd<="00";
            w_Te_cmd<="01"; ---w Te
            Tm_cmd<="00";
            vf_cmd<="00";
            if TL_done='1' and PE_done='1' and w_Te_done='1' then
                state<=s11; TL_cmd<="000";
            end if;
        when s11=>
            So_cmd<="00";
            TL_cmd<="110"; ---delay v14 ir1
            PE_cmd<="00";
            NS_cmd<="000";
            vhist_cmd<="00";
            SA_P_cmd<="00";
            FA_cmd<="00";
            FB_cmd<="00";
            GJE_cmd<="11"; ---rst idq0
            w_Te_cmd<="00";
            Tm_cmd<="00";
            vf_cmd<="00";
            if TL_done='1' and GJE_done='1' then
                state<=s0;
            end if;

        when others=>
        end case;
    end if;
end process;

process(init,J1,addr_ram_toRTDS,toRTDS,clk)
begin
IF init='1' then --reset
    addr_ram<=J1; din_ram<=(others=>'0'); wea_ram<="1";
ELSE
    IF toRTDS='1' THEN --send to RTDS
        addr_ram<=addr_ram_toRTDS;
        din_ram<=(others=>'0'); wea_ram<="0";
    ELSE --write ram
    if  clk'event and clk='1' then
```

```
IF weaface="1" THEN ---ir21 is22 vc14 vc24
    wea_ram <= "1"; din_ram<=dinface;
        if sys_No='0' THEN  addr_ram<="00"& addrface+39; --40-51
        else  addr_ram<="00"& addrface+51; --52-63
        end if;
ELSIF wea_Rxsys="1" THEN    ---
    wea_ram <= "1"; din_ram<=din_Rxsys;
        if sys_No='0' THEN  addr_ram<="00"& addr_Rxsys+51; --52-63
        else  addr_ram<="00"& addr_Rxsys+39; --40-51
        end if;
ELSIF ram_No=1 THEN
    wea_ram<= weaVscr OR weavo;
    IF weaVscr="1" THEN
        addr_ram<= "000" & addrVscr+36;
        din_ram<=dinVscr;
    ELSIF weavo="1" and addrvo<=75 THEN
        addr_ram<=  addrvo(5 downto 0)-39;
        din_ram<=dinvo;
    ELSE
        addr_ram<=(others=>'0');
        din_ram<=(others=>'0');
    END IF;
ELSIF ram_No=2 THEN
    wea_ram<= weavo;
    IF weavo="1" and addrvo<=114 and addrvo>=76 THEN
        addr_ram<=  addrvo(5 downto 0)-75;
        din_ram<=dinvo;
    ELSE
        addr_ram<=(others=>'0');
        din_ram<=(others=>'0');
    END IF;
ELSIF ram_No=3 THEN
    wea_ram<= weavo;
    IF weavo="1" and addrvo<=153 and addrvo>=115 THEN
        addr_ram<=  addrvo(5 downto 0)-114;
        din_ram<=dinvo;
    ELSE
        addr_ram<=(others=>'0');
        din_ram<=(others=>'0');
    END IF;
ELSIF ram_No=4 THEN
    wea_ram<= weavo;
    IF weavo="1" and addrvo<=192 and addrvo>=154 THEN
        addr_ram<=  addrvo(5 downto 0)-153;
        din_ram<=dinvo;
    ELSE
        addr_ram<=(others=>'0');
        din_ram<=(others=>'0');
    END IF;
ELSIF ram_No=5 THEN
    wea_ram(0)<= weavo(0) OR weaiPE(0) OR send_ab_Tm;
    IF weavo="1" and addrvo>=193 THEN
        addr_ram<=  addrvo(5 downto 0)-192;
        din_ram<=dinvo;
    ELSIF weaiPE="1" and addriPE>=157  THEN
        addr_ram<=  addriPE(5 downto 0)-144;
        din_ram<=diniPE;
    ELSIF send_ab_Tm='1' and send_addr_ab_Tm<=4 THEN
        addr_ram<="00"&send_addr_ab_Tm+34; din_ram<=send_dinab_Tm; --Tm
    ELSE addr_ram<=(others=>'0'); din_ram<=(others=>'0');
    END IF;
ELSIF ram_No=6 THEN
    wea_ram<= weaiPE;
    IF weaiPE="1" and addriPE<=39  THEN
        addr_ram<=  addriPE(5 downto 0);
        din_ram<=diniPE;
    ELSE addr_ram<=(others=>'0'); din_ram<=(others=>'0');
    END IF;
ELSIF ram_No=7 THEN
    wea_ram<= weaiPE;
    IF weaiPE="1" and addriPE>=40 and addriPE<=78 THEN
        addr_ram<=  addriPE(5 downto 0)-39;
        din_ram<=diniPE;
    ELSE addr_ram<=(others=>'0'); din_ram<=(others=>'0');
    END IF;
ELSIF ram_No=8 THEN
    wea_ram<= weaiPE;
```

```
            IF weaiPE="1" and addriPE>=79 and addriPE<=117 THEN
                addr_ram<=  addriPE(5 downto 0)-78;
                din_ram<=diniPE;
            ELSE addr_ram<=(others=>'0'); din_ram<=(others=>'0');
            END IF;
        ELSIF ram_No=9 THEN
            wea_ram<= weaiPE;
            IF weaiPE="1" and addriPE>=118 and addriPE<=156 THEN
                addr_ram<=  addriPE(5 downto 0)-117;
                din_ram<=diniPE;
            ELSE addr_ram<=(others=>'0'); din_ram<=(others=>'0');
            END IF;

        ELSIF ram_No=10 THEN
            wea_ram(0)<= weavc(0) OR weaidq0 OR weaTe OR weaw OR weaThist or send_vdq00 ;
            IF weavc="1" and addrvc>=40 THEN
                addr_ram<=addrvc(5 downto 0)-42; din_ram<=dinvc; --vc
            ELSIF weaidq0='1' THEN
                addr_ram<="00"&addridq0+20; din_ram<=dinidq0; --idq0
            ELSIF weaTe='1' THEN
                addr_ram<="0000"&addrTe+6; din_ram<=dinTe; --Te
            ELSIF weaw='1' THEN
                addr_ram<="0000"&addrw+12; din_ram<=dinw; --w
            ELSIF weaThist='1' THEN
                addr_ram<="0000"&addrThist+14; din_ram<=dinThist; --Thist
            ELSIF send_vdq00='1' and send_addrvdq00<=4  THEN
                addr_ram<="00"&send_addrvdq00+35; din_ram<=send_dinvdq00; --vdq00f
            ELSE addr_ram<=(others=>'0');din_ram<=(others=>'0');
            END IF;

--ELSIF ram_No=41 THEN
--          wea_ram(0)<= weaHs(0) OR weaP OR weavhist OR weaAbr1 OR weaAbr2 OR weaiG(0) OR
weainvP;
--          IF weaHs="1" and addrH<=3 THEN
--              addr_ram<=addrH(5 downto 0)+60;din_ram<=IHTL_s;     --Hs1  --61-63
--          ELSIF weaP='1' THEN
--              addr_ram<="00"&addrP; din_ram<=dinP; --P
--          ELSIF weavhist='1' THEN
--              addr_ram<="00"&addrvhist+15; din_ram<=dinvhist; --vhist
--          ELSIF weaAbr1='1' OR weaAbr2='1' THEN
--              addr_ram<="00"&addrAbr+30; din_ram<=dinAbr; --A
--          ELSIF weaiG(0)='1' THEN
--              addr_ram<="000"&addriG+45; din_ram<=diniG; --iG
--          ELSIF weainvP='1' and addrinvP<=9 THEN
--              addr_ram<="0"&addrinvP+51; din_ram<=dininvP; --invP
--          ELSE addr_ram<=(others=>'0');din_ram<=(others=>'0');
--          END IF;
--ELSIF ram_No=42 THEN
--          wea_ram(0)<= weaHs(0) OR wealumda OR send_B_vf;
--          IF weaHs="1" and addrH<=3 THEN
--              addr_ram<=addrH(5 downto 0)+60;din_ram<=IHTL_s;     --Hs1  --61-63
--          ELSIF send_B_vf='1' THEN
--              addr_ram<=send_addr_B_vf; din_ram<=send_doutB_vf; --vf
--          ELSIF wealumda='1' and addrlumda<=9 THEN
--              addr_ram<="00"&addrlumda+51; din_ram<=dinlumda; --lumda
--          ELSE addr_ram<=(others=>'0');din_ram<=(others=>'0');
--          END IF;
        ELSE    wea_ram<="0";
                addr_ram<=(others=>'0');
                din_ram<=(others=>'0');
        END IF;---- ram_No
    end if;--clk
    END IF;
END IF;
END PROCESS;


RAM_toRTDS : bRAM_toRTDS
  PORT MAP (
    clka => clk,
    wea => wea_ram,
    addra => addr_ram,
    dina => din_ram,
    douta => dout_ram
  );

end Behavioral;
```