Real-Time Emulation of Electrical Machines for Hardware-in-the-Loop Applications

by

Nariman Roshandel Tavana

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

Doctor of Philosophy
in
Energy Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Electrical machines are the critical components of many industrial systems, and their design, test, and simulation are now becoming increasingly demanding due to emphasis on energy efficiency, and performance improvement.

Today, hardware-in-the-loop (HIL) technology is progressively being utilized as the preferred, reliable, cost-effective alternative in a virtual scenario for tedious, time-consuming, and expensive tests on real devices. Thus, real-time digital hardware emulation of electrical machines in HIL configuration allows engineers to test the newly prototyped drive systems or controllers against the virtual machine model under hazardous and abnormal conditions in a non-destructive manner. Moreover, in the design procedure the emulated machine model can help designers optimize the machine performance by running the real-time model as many times as they need to reach the desired goals in a short time.

Owing to the rapid advances in the digital hardware technology, field programmable gate arrays (FPGAs) are gaining increasing popularity as the fastest, most reliable, and preferred computational engine by providing high frequency computational clock cycle and massive amount of logic resources for various computationally expensive applications.

This thesis provides a framework for real-time emulation of commonly used electrical machines with different levels of modeling complexity. The FPGA is employed in this work for the high performance data processing to meet the stringent real-time step-size constraints. The FPGA-based real-time emulated machine performances are compared with the experimental measurements and finite element solutions to demonstrate the accuracy and effectiveness of the proposed approaches for HIL applications.

To my parents for their endless support,

and encouragement.

Acknowledgements

I would like to express my deepest appreciation to my supervisor *Prof. Venkata Dinavahi* for his supportive attitude, encouragement, and guidance through my research at the University of Alberta. Undoubtedly, without his constant help and supervision, this dissertation would not have been possible.

It is an honor for me to extend my gratitude to my Ph.D. committee members *Dr. Shaahin Filizadeh*, *Dr. Hai Jiang*, *Dr. Yasser Abdel-Rady I. Mohamed*, *Dr. Ali Khajehoddin*, and *Dr. Qing Zhao* for reviewing my thesis and providing thoughtful comments to improve it. And my special thanks go to my colleagues and friends at the RTX-Lab with whom I had a wonderful time during my Ph.D. program.

Table of Contents

List of Tables

List of Figures

1 Introduction

Electrical machines are the primary components for the generation and utilization of most of electrical energy converted from fossil fuels and renewable sources [1]. Thus, high energy usage and their wide range of applications are enough reasons for the importance of validation, testing, and design optimization of electrical machines and drive systems. Traditionally, off-line digital simulation has been used to evaluate the performance of electrical machines and drive systems before their implementation and design as a new prototype for many years. Several commercial software packages have been developed for this purpose including model-based simulation tools such as Matlab/Simulink®, and numerical electromagnetic simulation tools such as JMAG® or ANSYS®. However, simulation results by this approach either suffer from inaccuracy due mainly to oversimplification in the physical models in the simulation procedure or enjoy high precision at the cost of long simulation time. Thus, off-line simulation of machine models and drive systems cannot reflect their true performance that would occur in the real environment with respect to time.

To overcome the aforementioned problem, real-time simulation is desired for modeling in which all the necessary calculations must be performed within the time frame as they take place in real world [2]. To achieve this goal, real-time simulators have been used usually based on general purpose processors or digital signal processors. However, there is an ever increasing demand to accommodate detailed models of electrical machines and drive systems within a very small simulation time-step, which essentially means the need for higher computational power of the simulator. FPGAs offer a viable alternative for speeding up the digital simulation without sacrificing the accuracy [3,4]. Owing to their parallel hardwired architecture, large logic resource count, huge amount of storage elements, and fast clock speeds, currently available FPGA devices are able to satisfy accuracy demands

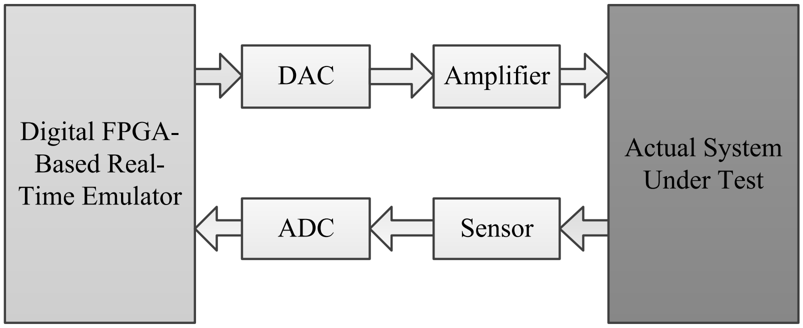


Figure 1.1: Generic hardware-in-the-loop configuration.

of complex machine and drive systems adequately by providing nanosecond resolution for the computational clock cycles within the simulation time-step in real-time.

Many studies have been conducted on FPGA-based real-time simulation of power networks, controllers, and power electronics apparatus. Nevertheless, detailed real-time modeling of electrical machines on FPGA has so far gained less attention. In this research, the main focus is on the real-time emulation of electrical machine models, which can be developed based on different approaches with varying levels of detail and reasonable accuracy for hardware-in-the-loop (HIL) applications.

1.1 Real-Time Hardware-in-the-Loop Simulation

Nowadays, real-time HIL technology is rapidly becoming the preferred, reliable, and cost-effective substitute in a virtual scenario for tedious, time-consuming, expensive tests and analyses on real devices. HIL simulation allows designers to test the newly prototyped design of a portion of a system against the virtual model of other parts especially under hazardous and borderline conditions in a non-destructive manner.

The real-time simulator can be equipped with a set of input and output ports to be interfaced and connected to real parts of the system, in which is commonly called HIL simulation [5].

Since there is a gap between the off-line behavior of the model and its actual performance in the real word with respect to time, only a real-time emulated machine model can be used in the HIL platform to interact with physical systems [6].

As can be seen from Fig. 1.1, the generated digital real-time emulator signals, which transmit low-level voltages and currents, can be interfaced with an amplifier where an actual real-time emulated machine works in HIL configuration. The main purpose of the HIL platform is to simulate the real world as closely as possible. Thus, the output digital-to-analog ports for sending signals as well as analog-to-digital ports for receiving them must work as accurately and deterministically as possible.

1.2 Machine Models

To realistically reproduce the electrical machine transients with high fidelity, an accurate modeling of the machines with a small simulation time-step is a necessity for the real-time emulator. Different techniques are available for the machine modeling, which can be categorized into four main groups, and their general aspects are presented in this section.

1.2.1 qd Model

The equations that describe the dynamic performance of AC machines in the phase domain consists of an inductance matrix being a function of rotor position. A change of frame is often used to reduce the complexity of such matrix by referring the machine variables to a reference frame that rotates at an arbitrary angular velocity. This approach is based on a lumped parameter model in which the magnetic coupling between different parts of machine are defined by a set of self and mutual inductances of the windings. This model is widely used in the transient simulation of machines because of its fast solution. However, the model is not capable to take spatial effects inside the machine into account [7].

A unified framework is proposed in this project for FPGA-based real-time simulation of electrical machines by using state-space formulation in the orthogonal $q-d$ axis model. The machine models are implemented on the FPGA by both the schematic and textual programming methods. Different hardware realization techniques of *qd*-type machine models and their advantages and disadvantages will be discussed with more details later.

1.2.2 Magnetic Equivalent Circuit

The magnetic Equivalent Circuit (MEC) is another technique for machine analysis. MEC develops a lumped representation of main flux paths in a machine with permeance elements as well as Magnetomotive Force (MMF) sources [8]. Unlike the qd model, the MEC is based on geometrical data and material properties and is able to capture local phenomena. The MEC is computationally less intensive and relatively less time-consuming compared with the finite element model (FEM) and analytical approach, as well as offers satisfactory accuracy. This technique is gaining increasing popularity in the design procedure and transient simulation of machines where repetitive computations are required in a short time. This method is successfully applied to different types of machines such as induction machines, synchronous machine, switched reluctance machine, etc. [8–10].

An induction machine (IM) will be emulated in this work in real-time on FPGA by a new MEC, which is suitable for the detailed real-time machine simulation. For the modeling of IM, MEC is the most accurate technique after FEM. Moreover, to have more realistic results, the nonlinear effects due to local saturation of stator and rotor teeth as well as yoke will also be taken into consideration in the real-time emulation.

1.2.3 Analytical Model

In order to facilitate real-time simulation, design optimization, and accurate dynamic modeling of electrical machines, various approaches have been employed to predict the magnetic field distribution. The most common method is to use a lumped magnetic equivalent circuit model. Although it is fast and allows the relationship between critical design parameters and machine performance to be established, it suffers from the problem associated with the model inaccuracy, particularly when flux leakage is significant and flux paths are complex [11]. Numerical techniques such as FEM provide high accuracy but remain time-consuming and are not able to clarify the exact influence of geometrical data on the machine behavior. To overcome these problems, the analytical method has been established.

In this research, an analytical model incorporating Maxwell's equations is developed for the modeling and real-time emulation of a permanent magnet synchronous machine (PMSM) on the FPGA. In order to establish the analytical solution for the magnetic field distribution in the PMSM, the magnetic field analysis is confined to various layers. It is convenient to formulate the field distribution in terms of a magnetic vector potential. In the governing field equations obtained by this method, the perpendicular and tangential magnetic boundary conditions should be satisfied between layers to predict magnetic flux density distribution inside the machine and its performance [12, 13].

1.2.4 Finite Element Method

Finite element analysis is a powerful method in various areas of engineering for solving partial differential equations or boundary value problems. For modeling of electrical machines by FEM, Maxwell's equations of machine are formulated in terms of magnetic vector potentials and then solved based on boundary conditions. The main advantage of the FEM is its high accuracy for modeling all details inside the machine such as nonlinear and local effects. However, this method is extremely time-consuming even for a moderately complex structure of machine. Therefore, it is not suitable for real-time simulation due to the need of large amount of computing resources. FEM is usually chosen to verify the correctness of other methods and final designs [14]. In this research, off-line FEM simulation is used to validate the various real-time emulated machine models.

1.3 Literature Review

This section provides a literature review on the studies conducted in the development of FPGA based real-time digital emulation of electrical machines.

Reference [15] proposed a digital hardware realization of a real-time simulator for an induction machine with a field-oriented control using the FPGA as the computational engine. The authors used state-space formulation to model the electric machine and trapezoidal rule for its discretization. The simulator was developed using HDL coding for the controller and schematic method for the asynchronous motor. In a similar attempt, an induction machine dynamic simulation on FPGA board has also been done in [16]. In this paper, Runge-Kutta 4^{th} order numerical integration algorithm is used to discretize the asynchronous machine equations and also used HDL method to implement machine model on FPGA. Additionally, a double and single three phase induction machine have been emulated in real-time on FPGA and verified by experimental measurements in [17] and [18], respectively. In the recent studies, a full FPGA-based real-time system of power converters and induction machine electric vehicle HIL applications has been presented in [19]. In this work, Thevenin based equivalent model is employed to enhance the modeling of switches in converters and saturation effect is also taken into account in *qd* model of IM machine for more accurate emulation. Opal-RT® platform is utilized in this paper for real-time HIL studies. A ModelSim® result of FPGA-based real-time transients of IM machine is also provided in [20]. In this paper, a comparison between various discretization approach including Shift, Tustin, and Delta method has been presented. According to digital implementations, the performance of Delta operator has been demonstrated to be better than other techniques in terms of accuracy, stability, robustness. Plus, a similar trend by the authors has been applied to synchronous motor in [21]. Furthermore, in [22–25], a real-time simulator of a permanent magnet synchronous motor drive based on a finite-element method has been realized on a FPGA card for HIL testing of motor drive controllers. In these papers, the authors proposed PMSM model in the phase-domain with inductances and flux profiles computed from JMAG-RT® finite element software on a CPU of RT-LAB® real-time platform and implemented PMSM model as well as 3-phase inverter on a FPGA card of the real-time simulator using look-up tables. Furthermore, they implemented *qd* model of PMSM without any calculation by finite element method on the FPGA chip in [26, 27]. A dSPACE® setup is chosen in [28] for real-time simulation of interior type PMSM on FPGA. The authors used 3-dimensional look-up-table (LUT) for inductances in *q*-*d* axis and Verilog HDL coding for hardware realization. The LUT is presented to consider the saturation and cross coupling effect in this machine for HIL tests. Moreover, they considered iron loss effect in this type of machine for FPGA-based real-time studies in [29]. However, the lumped mathematical models of all AC machines have been presented in [30] and the authors used Backward Euler to discretize machine models and developed its real time simulation on FPGA by HDL codes for the implementation with fixed-point number representation. Finally, they compared the FPGA-based real-time simulated results with off-line results from Matlab/Simulink to demonstrate the method accuracy.

On the other hand, a number of studies have focused on real-time simulation of special machines such as synchronous reluctance machine, switched reluctance and brushless DC (BLDC) machines. For instance, References [31, 32] presented the state-space approach to model BLDC and the induction machine, and then applied forward Euler method for their discretization and finally evaluated their performance by FPGA implementation. Moreover, another research has been done by a similar method for FPGA based-digital hardware realization of BLDC in [33]. A hardware real-time simulator of synchronous reluctance motor also proposed in [34]. For this realization, the motor model in $q-d$ rotating coordinate system was discretized by Runge-Kutta 4^{th} order method and HDL code was used to program the FPGA chip. In addition, a real-time simulation of switched reluctance motor (SRM) and DC machine on FPGA has been presented by schematic method in [35–37].

This thesis proposes a general framework for real-time simulation of all electrical machines using qd model. Then, a novel MEC approach is selected for detailed nonlinear FPGA-based real-time emulation of induction machine by HDL coding. Finally, a distributed analytical model is employed to model dynamic behavior of PMSM in real-time on FPGA. All FPGA-based real-time emulated machine behaviors are compared and verified by FEM and experimental results to demonstrate the correctness of the proposed techniques.

1.4 Motivation of this work

Unlike general purpose CPUs or DSPs which are basically sequential devices, FPGA is a digital hardware device, which provides massively parallel processing, supporting multiple simultaneous threads of execution. FPGA offers a viable alternative computational engine for speeding up real-time simulators for endless growing complexity and increasing accuracy demand of a system model. Since FPGA is a fully user reconfigurable device, its customized hardware architecture makes it a processer of choice for real-time emulation.

Moreover, today FPGAs are more and more being employed to design high-end computationally intense applications. Until now most of studies in the area of FPGA-based real-time simulation have been focused on accurate modeling of power electronic apparatus [38–41], control and drive systems [42–45], protective devices [46], and power system simulation [47, 48]. While these are important issues, the real-time emulation of electrical machines seems to be necessary and challenging for performing realistic simulations.

The first challenge is that the FPGA hardware design is quite difficult compared with software programming using high-level language such as C/C++. Every calculation needs to be assigned to a particular hardware module and many independent hardware modules may work simultaneously in a parallel fashion when the entire algorithm is processing. Thus, to implement a complicated system model, all parallel pathes must be identified and realized for exploiting the internal parallelism in order to expedite computations. The second challenge is that a real-time emulation of complex systems on an FPGA for real-time simulation is usually carried out by hardware description language (HDL), which makes it restricted only for expert FPGA users. This thesis also attempts to provide a framework for hardware realization of machine models on an FPGA for novice FPGA users by schematic method. Additionally, a detailed nonlinear real-time electrical machine model is developed in this work. An FPGA hardware implementation of an iterative solution approach is another challenge for solving the governing nonlinear equations of the system model. For real-time purposes, the main obstacle is to perform the iterative procedure within the specified simulation time-step, which can only be overcome through the use of deep pipeline and parallel realization combined with the optimized numerical procedure on FPGA.

This thesis focuses on the aforementioned challenges to design an FPGA-based real-time emulator of different types of machine model. In the first part of this project, the *qd* model of electrical machines is chosen for real-time simulation. This part provides a useful and comprehensive comparison between floating-point and fixed-point arithmetic for hardware implementation, and addresses the differences of deeply pipelined and highly parallelled realization schemes, and the contribution of schematic and textual programming language methods for design configuration of electrical machine model. Hardware implementation by these approaches are evaluated in terms of real-time step-size, accuracy, and hardware resource consumption. In the second part, due mainly to the key advantages of magnetic equivalent circuits (MEC) for modeling induction machines compared with finite-element analysis and electric equivalent circuits in terms of computational expense and achieved accuracy, this work proposes a real-time nonlinear MEC of the induction machine. The model is emulated in real-time on the FPGA by exploiting the parallel hardware architecture, and fully pipelined arithmetic processing. The performance of FPGA-based real-time emulated induction machine model is finally investigated and compared with the behavior of an experimental setup of induction machine and finite-element results to demonstrate the effectiveness and accuracy of proposed approach for HIL applications. The third part presents a real-time emulation of an analytical space harmonic model of a permanent magnet synchronous machines with shaped poles. The goal of this configuration is to produce an air-gap flux density distribution as close to sinusoidal as possible to enhance the machine performance. The analytical model is derived to predict the magnetic fields and machine behavior by solving Maxwell equations and applying the superposition theorem. The digital hardware realization of the model is developed on FPGA in a parallel and pipelined paradigm for an efficient hardware design. Such real-time emulation on FPGA can be used in the design procedure, and the assessment of newly-prototyped controllers and drive systems in the hardware-in-the-loop (HIL) platform.

1.5 Thesis Objectives

The main objective of this thesis is to develop real-time models of electrical machines emulated in hardware on the FPGA with different levels of realism. To achieve this goal, the following steps are taken:

- Mathematical modeling of electrical machines needs to be developed and then implemented on FPGA for real-time emulation. For more realistic emulation, the complexity of machine models have to be increased by including geometrical parameters, spatial effects, and saturation phenomena, which implies more hardware resource consumption and longer execution time. Through the chapters of this thesis, it can be observed that the accuracy and effectiveness of the FPGA-based emulator are improved by increasing the complexity of the machine model.
- A new magnetic equivalent circuit model needs to be proposed for real-time simulation of induction machine on FPGA. The MEC model is developed in such a way that finer permeance element sizes compared with similar MEC models are defined to predict machine behavior more precisely. Moreover, all unnecessary flux pathes are neglected and only flux tubes are considered in the path of magnetic materials to improve the computational time significantly by reducing the computational efforts and slight sacrificing the results accuracy.
- A new magnetic pole shape as well as pole shape optimization need to be proposed for the performance improvement of permanent-magnet machines. An analytical method is established to predict magnetic fields in the different regions of the machine by solving Maxwell's equations and applying boundary conditions. Analytically derived formulas allow the prediction of machine performance in closed form as a function of physical dimensions and parameters of the machine. They facilitate the characterization of machine topologies, sensitivity analysis, iterative design optimization procedure, and real-time dynamic modeling of the machine.
- To take the advantages of parallel architecture of FPGA for very fast computation, the possible parallel computational tasks have to be extracted and allocated into parallel computational hardware.
- A highly parallel iterative numerical technique needs to be developed on FPGA for solving nonlinear equations of electrical machines in order to meet real-time requirements.
- The digital hardware modules have to be developed to realize basic arithmetic operations and functions on FPGA. In order to implement the developed machine models, the hardware modules such as three-phase voltage source unit, sparse matrix multiplication module, floating-point multiply add/subtract unit, nonlinear function unit,

Gauss-Jordan Elimination module, Newton-Raphson unit, residual vector calculator unit, and Fourier series module are efficiently designed in pipeline and parallel schemes on FPGA. These hardware modules need to be wired up together properly to form a hardware architecture of machine models.

- The real-time-emulated machine behavior needs to be evaluated by the results obtained from other methods. The real-time results captured from oscilloscope are compared with the measured performance of actual machine in an experimental setup or finite-element solutions.

1.6 Thesis Outline

This thesis consists of six chapters and is organized as follows:

- **Chapter 2: An Overview OF FPGA Architecture and Design** - This chapter presents an introduction about FPGA technology and its architecture. Different FPGA design methodologies are also discussed for real-time emulation of electrical machines.
- **Chapter 3: Real-Time *qd*-type model of Electrical Machines** - The purpose of this chapter is to provide a general framework for real-time simulation of *qd* model of all types of electrical machines on FPGA. State-space approach is employed to present the *qd*-type machine model in a unified framework and also a comprehensive comparison is carried out between various hardware implementation techniques of machine models in terms of obtained accuracy, FPGA resource utilization, and achieved time-step size.
- **Chapter 4: Real-Time Nonlinear Magnetic Equivalent Circuit Model of Induction Machine** - This chapter proposes a magnetic equivalent circuit model suitable for real-time emulation of induction machine. An iterative under-relaxed Newton-Raphson method along with the anti-periodicity technique is realized on FPGA to solve the governing equations of the machine in real-time.
- **Chapter 5: Real-Time Analytical Space Harmonic Model of Permanent Magnet Machines** - A detailed analytical space harmonic model is developed to predict magnetic fields and consequently permanent magnet machine performances. Analytical formulas are based on Fourier series. Real-time emulation of the machine model is then realized on FPGA for hardware-in-the-loop simulation. Finally, finite-element analysis is employed to evaluate the accuracy of presented approach.
- **Chapter 6: Conclusions and Future Works** - The contribution of this research and the future works are summarized in this chapter.

2 An Overview OF FPGA Architecture and Design

2.1 Introduction

Nowadays, the coupling of falling prices and ever-increasing number of transistors per chip has shifted the role of FPGAs from a low-volume electric subsystems, toward being a main processing engine and building block in the electronic market [49].

In general, FPGA is a programmable logic device. The basic idea behind programmable hardware is to have a generic circuit where the functionality can be programmed for a particular application [50].

Conventional CPUs are based on the idea that the ALU can perform only one of several operations at a time based on the control signals. Thus, a particular application must be broken into the sequence of control signals for controlling of the function of the ALU. However, programmable logics in FPGAs represent the functionality as a circuit where the particular circuit can be programmed to meet the requirements of an application. The key difference is that the functionality is realized as a parallel system in an FPGA rather than sequential in a CPU.

Since FPGA offers a parallel hardware architecture, this features very high speed processing as a result of hardware design while retaining the reprogrammability of software at a relatively low cost. This makes FPGAs well suited for real-time emulation, where all required computations of a system model must be performed within a very small simulation time-step.

Due to its unique features, now FPGAs are also used in various fields of applications such as telecommunications, image and signal processing, robotics, automotive, and aircraft systems, as well as power convertor control [51].

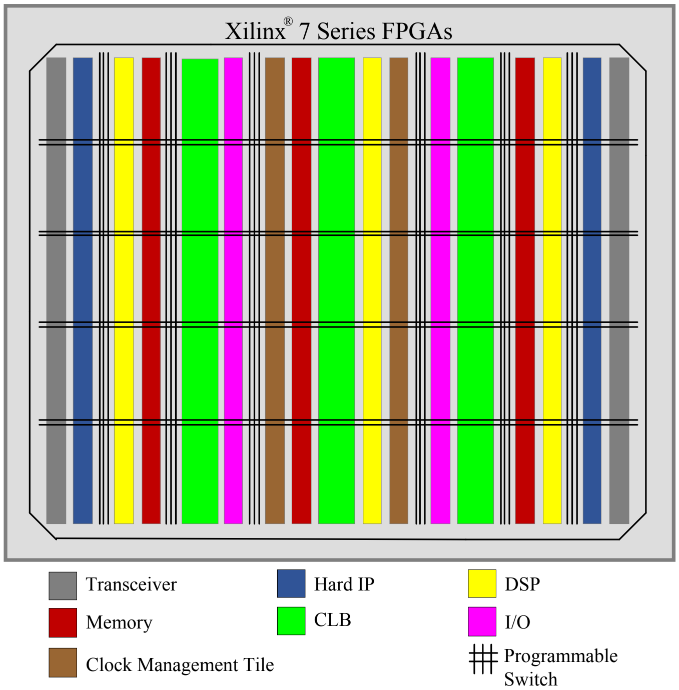


Figure 2.1: FPGA hardware architecture based on a columnar approach [52].

Table 2.1: Main hardware resources of FPGA chip.

	Virtex®-7 FPGA	
	FPGA Number	**VC7VX485T**
Logic Resources	Slices	75,900
	Logic Cells	485,760
Memory Resources	Distributed RAM (Kb)	8,175
	Total Block RAM (Kb)	37,080
Clocking Resources	CMTs (1 MMCM + 1 PLL)	14
I/O Resources	Single-Ended I/O	700
	Differential I/O Pairs	336
Integrated IP Resources	DSP Slices	2,800
	GTX Transceivers (12.5 Gb/S)	56

In this chapter, a general architecture of Xilinx-7 FPGA, which is utilized in this work for real-time electrical machine emulation will be introduced briefly. Then, the FPGA design flow, and some important points about hardware realization schemes are discussed.

2.2 FPGA Architecture

The generic FPGA is a two-dimensional (2-D) array of programmable logic cells interconnected by a matrix of wires and programable switch matrices. Each logic cell performs

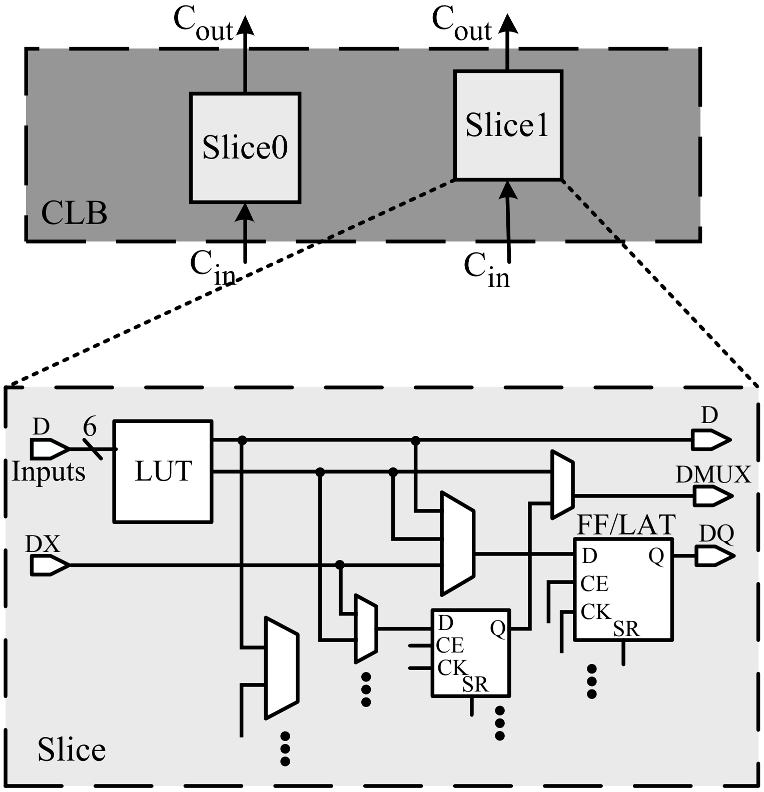


Figure 2.2: Configurable logic block, and slice structures [52].

a basic logic function, while switch matrices control the configurable logic cells and the interconnection of the wires, thus achieving filed programmability. Fig. 2.1 illustrates the advanced 28-nm Xilinx Virtex®-7 FPGA hardware architecture and its features, which are used in this work for FPGA-based real-time emulation. Xilinx created the column based advanced silicon modular block architecture to enable FPGA platforms with varying feature mixes optimized for different applications. Basically, the recent FPGA chips consist of an ocean of configurable logic blocks (CLBs), memory elements, DSP blocks, I/O blocks, clock management tiles, transceivers, and Hard IP blocks which are interconnected by an entirely reprogrammable matrix of switch elements.

Table. 2.1 lists the main hardware resources of Xilinx Vitrex®-7 XC7VX485T FPGA. In this section, the main logic resources of all FPGA types, namely, configurable logic blocks (CLBs), memory elements, and DSP blocks, will be explained with further details.

2.2.1 Configurable Logic Blocks (CLBs)

The logic cells in this FPGA are configurable logic blocks (CLBs), which are the main resources for the realization of combinatorial and sequential circuits.

7-series CLBs are composed of a pair of slices. Each slice contains four of either 6-input

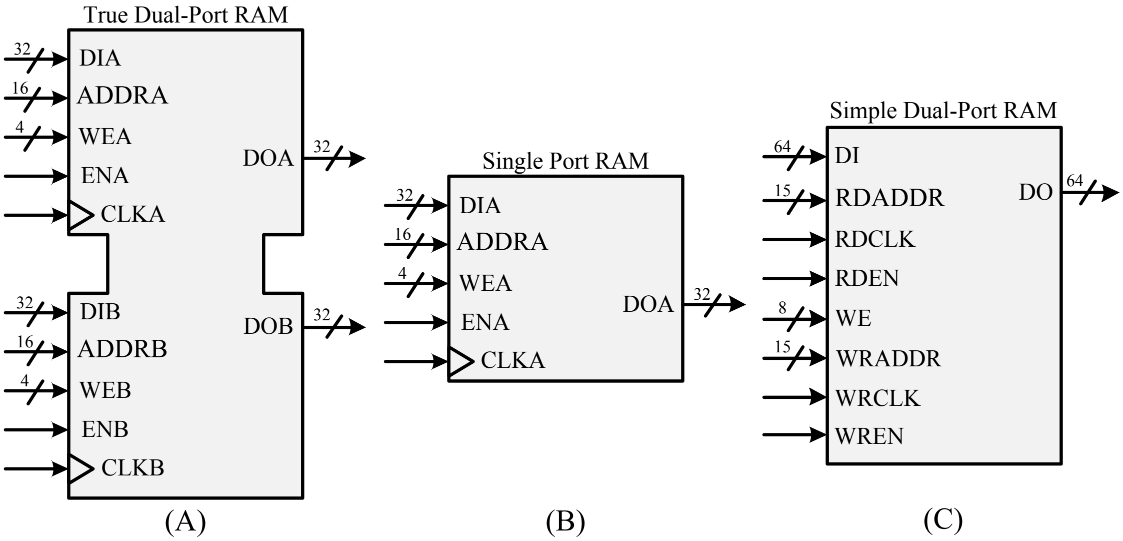


Figure 2.3: Block RAM in 7-series Xilinx FPGA. (a) True dual-port RAM, (b) single port RAM, and (c) simple dual-port RAM.

Table 2.2: Port functions and descriptions of 7-series FPGA RAMs [53].

Port Functions	**Description**
DI[A—B]	Data input bus.
ADDR[A—B]	Address bus.
WE[A—B]	Byte-wide write enable.
EN[A—B]	When inactive no data is written to the block RAM.
CLK[A—B]	Clock input.
DO[A—B]	Data output bus.
RDADDR	Read data address bus.
RDCLK	Read data clock.
RDEN	Read port enable.
WRADDR	Write data address bus.
WRCLK	Write data clock.
WREN	Write port enable.

LUTs with one output or two five-input LUTs with independent outputs with common addresses or logic inputs, eight flip-flops, and one arithmetic and carry chain. Almost two-thirds of the slices are logic slices and the rest can also use their LUTs as distributed 64-bit RAM or as 32-bit shift registers or as two 16-bit shift registers. A simplified structure of 7-series CLBs and the slice architecture are depicted in Fig. 2.2 [52].

2.2.2 Block RAMs

In Xilinx®-7 series FPGAs, the block RAMs store up to 36 Kbits of data configured either two separate 18 Kb RAMs, or one 36 Kb RAM. Thus, in addition to distributed RAM, 7

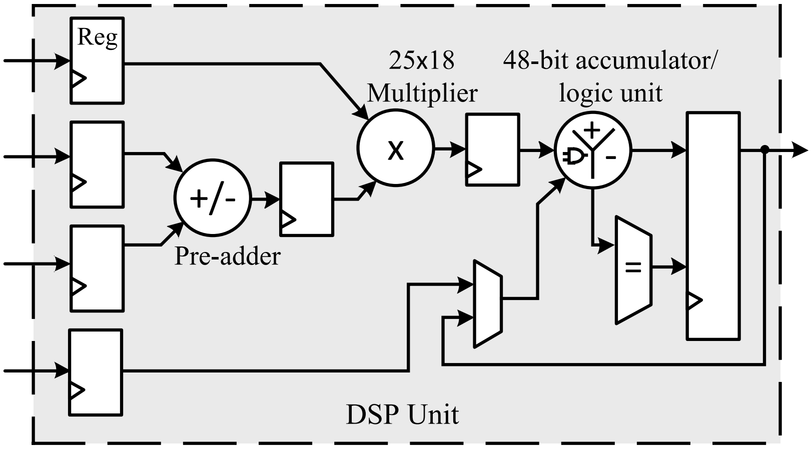


Figure 2.4: Digital signal processing unit [54].

series devices feature a large number of RAMs. Embedded dual- or single- port RAM modules, ROM modules, and synchronous FIFO are implemented using the Xilinx CORE Generator^{TM} block memory modules. Block RAMs can be used to configure true and simple dual-port RAMs as well as single-port RAMs [53].

The true dual-port RAM has two completely independent access ports (A and B) and data can be written to either or both ports and read from either or both ports, while there is only one access port for single-port RAM. However, in the simple dual-port RAM mode, independent read and write operations can occur simultaneously, where port A is designated as the read port and port B as the write port. Fig. 2.3 shows the data flow of various RAM types and Table. 2.2 describes their functions.

2.2.3 Digital Signal Processing Blocks (DSPs)

Besides the basic logic blocks, and massive embedded memory blocks, the FPGA also provide the dedicated circuits such as DSP blocks. These blocks can implement custom, fully parallel algorithms and be used for computationally intensive digital signal processing applications with high speed and accuracy. Dedicated DSP slices are the best blocks for the implementation of many binary multipliers and accumulators. Thus, they can enhance the speed and efficiency of digital signal processing in many applications.

The basic functionality of DSP48E1 is presented in Fig. 2.4. Some highlights of DSP functionality are 25×18 two's-complement multiplier, 48-bit accumulator, power saving pre-adder, and single-instruction-multiple-data arithmetic unit [54].

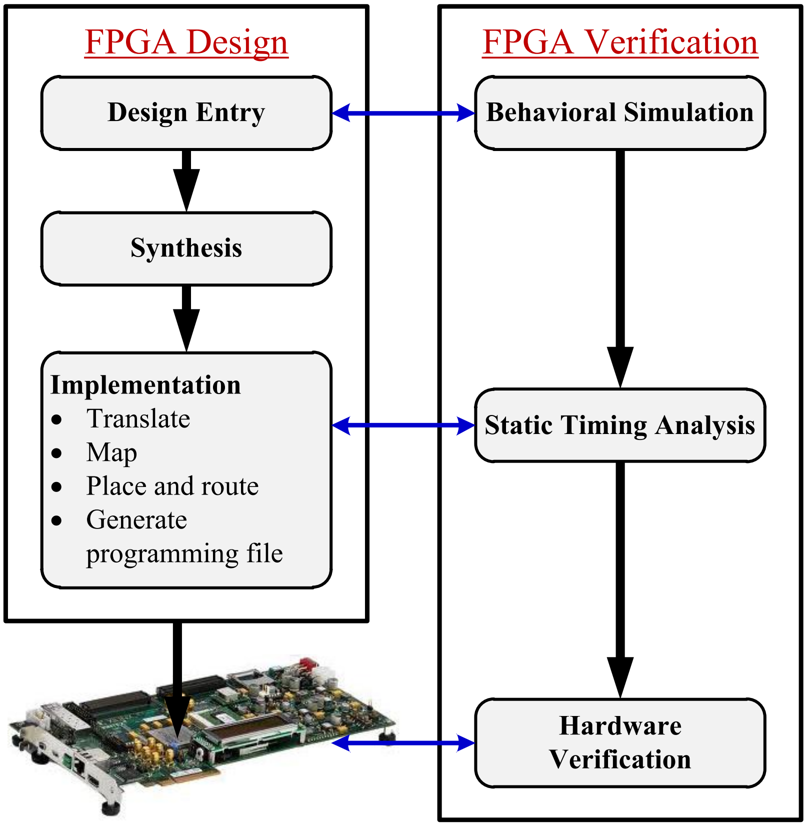


Figure 2.5: Generic FPGA design flow.

2.3 FPGA Design Flow

Today, FPGA vendors provide a fully integrated development environment for FPGA design procedure from design entry all the way down to a generated FPGA bitstream downloaded into the chip. Currently, Xilinx introduces two development platforms for FPGA users. Xilinx ISE Design Suit supports all Xilinx programmable devices, and Vivado Design Suit is the next generation of development platform for the Xilinx FPGAs. As shown in Fig. 2.5, the FPGA design flow mainly consists of three design stages, namely, design entry, synthesis, and implementation, as well as three verification stages, namely, behavioral simulation, timing analysis, and hardware verification.

- **FPGA Design**

  Design Entry - A design entry file is used to represent the design. The design entry can be based on schematic or textual programming methods. Textual programming is done in hardware description language (HDL) such as Very High Speed Integrated Circuit HDL (VHDL) or Verilog HDL. This is the first stage to define design architecture.

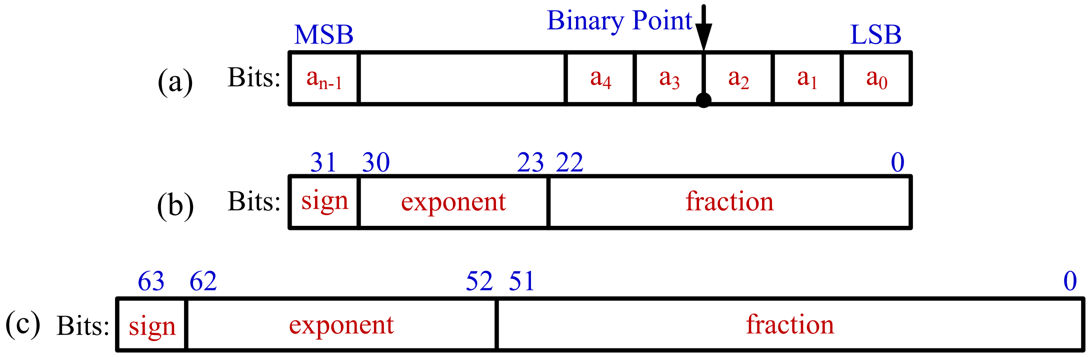


Figure 2.6: Floating-point number representation. (a) single-precision, and (b) double-precision.

Synthesis - This technology is a process by which a design entry file is turned into a netlist file. The netlist file contains both the logical design data and constraints.

Implementation - This step comprises four sub-process. 1) *Translate*: this sub-process translates netlist file into a FPGA design file. 2) *Map*: in this stage, the design is fit into available FPGA hardware resources. 3) *Place and route*: this sub-process assigns the design to physical device and selects wires and switches for interconnections. 4) *Generate programming file*: a bitstream file is created in this stage to be downloaded to the device.

- **FPGA Verification**

Behavioral Simulation - Functional or behavioral simulation is usually performed before synthesis process. This simulation is usually carried out to verify the behavioral code or to confirm that the design is functioning as intended. Isim® or Vivado® simulation tools or Modelsim® can be employed for this simulation.

Static Timing Analysis - Timing Analyzer is used to perform a detailed analysis of the FPGA design after implementation process. This analysis is carried out to ensure that the specified timing constraints are properly met by the implemented design.

Hardware Verification - After (re)programming the device, the hardware signals can be monitored by ChipScope™ Analyzer for debugging before sending digital signals to output ports of FPGA platform.

2.4 Number Representation and Operations

For computational programming, the first step is to define the data format and arithmetic operations. In spite of the dependency of data format and the accuracy of computation, number representation also affects the hardware resource utilization in the FPGA design and programming. Basically there are two types of number systems:

1. **Fixed-point number:**

   A fixed-point number is characterized by their word size in bits, binary point, and their sign. A common representation of a binary fixed-point number, either signed or unsigned, is shown in Fig. 2.6 (a).

   As can be seen, a_is are the binary digits, n is the word length in bits, a_{n-1} is the most significant bit (MSB), and a_0 is the least significant bit (LSB), and the binary point is depicted three bits to the left of the LSB.

   It is worth mentioning that the preferred representation of signed fixed-point numbers is Two's complement method.

2. **Floating-point number:**

   According to *IEEE* standard 754, floating-point numbers may be represented in either single- or double-precision format.

   - *Single-precision floating-point number:*

     Any value stored in this representation has 32 bits. It is formatted as depicted in Fig. 2.6 (b).

     A 32-bit floating-point number consists of 1 *sign* bits, 8 *exponent* bits, and 23 *fraction* bits. The single-precision floating-point number is expressed as follows:

     $$d = (-1)^{sign} \times 2^{(exponent-127)} \times (1.fraction) \tag{2.1}$$

     The value in this format is approximately in the range of -10^{38} to 10^{38}.

   - *Double-precision floating-point number:*

     Any value stored in this representation has 64 bits. It is formatted as depicted in Fig. 2.6 (c).

     A 64-bit floating-point number consists of 1 *sign* bits, 11 *exponent* bits, and 52 *fraction* bits. The double-precision floating-point number is expressed as follows:

     $$d = (-1)^{sign} \times 2^{(exponent-1023)} \times (1.fraction) \tag{2.2}$$

     The value in this format is approximately in the range of -10^{308} to 10^{308}.

In this thesis, single-precision floating-point is employed for computational programming of FPGA. It actually offers high speed computation, dynamic range of data, and acceptable accuracy. These are the main reasons for choosing single-precision format for real-time emulation.

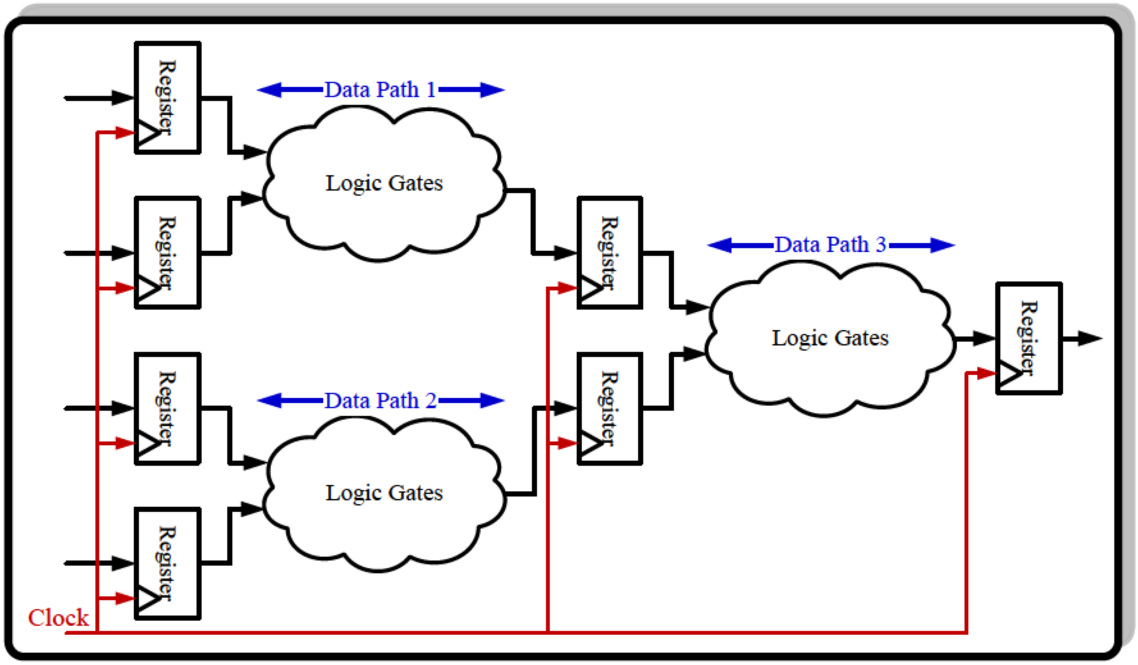


Figure 2.7: Pipeline and parallel pathes of FPGA Design.

2.5 FPGA Design Schemes

Parallel and pipeline paradigms are two interesting features for FPGA programming to improve computational speed and throughput. In this section, these two FPGA design techniques will be explained in more details.

2.5.1 Pipeline Design Architecture

In the pipeline technique, a function is divided into the several stages and the registers are inserted between the stages in the data path. Thus, data can march through the registers at every clock cycle. Although pipelining increases the number of clock cycles per operation, corresponding to the number of stages in a pipeline path, it improves the computational throughput by increasing the number of operations per unit of time and also conserves FPGA hardware resources. As can be seen from Fig. 2.7, the pipeline path of FPGA design has three stages. The first valid results become available after 3 clock cycles and the next results will be ready only after one clock cycle instead of waiting for another 3 clock cycles, consequently, the data throughput is one result per clock cycle.

2.5.2 Parallel Design Architecture

Unlike CPUs or DSPs that are sequential computational engines, an FPGA enjoys its intrinsic parallel architecture for high speed operations and computations. Basically an algorithm can be partitioned into several independent circuits and computed simultaneously on an FPGA. As can be seen from Fig. 2.7, the operations in the first and second paths are executed concurrently on FPGA, whereas these two operations must be performed sequentially on CPU.

2.6 Summary

This chapter presented the fundamentals of FPGA architecture and design briefly. The issues involved in the hardware realizations were shortly explained. The design procedure can be followed for real-time emulation of a system on FPGA. Moreover, taking advantage of parallel and pipeline programming allows FPGA designers to efficiently realize digital emulation with high speed, improved data throughput, and optimized hardware resource utilization.

3
Real-Time *qd*-Type Model of Electrical Machines

3.1 Introduction

This chapter [1] presents a unified framework for FPGA-based real-time emulation of *qd*-type machine models. Off-line transient simulations have been used as a successful method to evaluate the performance of electrical machines for a long time. Although execution time is still a matter of concern in such off-line tools, it is not as critical as in a real-time digital simulator, which has to interact with external devices in a HIL scenario. The main application of real-time emulation of electrical machines is to evaluate the behavior of newly designed machines, drive systems, controllers, and protective devices in a HIL configuration in an effective and economic approach before applying them in a real system [5, 22, 55–57]. Such testing allows the system components to be subjected to extreme conditions in a non-destructive environment and an expedited manner. To reproduce electrical machine transients with high fidelity, an accurate modeling of the machines with a small simulation time-step is a necessity for the real-time simulator. To meet strict real-time step-size constraints, a compromise is usually made between the accuracy and complexity of the system model [5].

Owing to the rapid developments and dramatic advances in digital hardware technology, the FPGA is becoming the fastest, most reliable, and preferred computational engine for digital hardware realization of complex systems without sacrificing the accuracy [39, 43, 44]. Today, FPGA has gained a crucial role in the HIL simulation and rapid

[1] Material from this chapter has been published: N. R. Tavana, and V. Dinavahi, “A general framework for FPGA-based real-time emulation of electrical machines for HIL applications”, *IEEE Trans. on Industrial Electronics*, vol. 62, no. 4, pp. 2041-2053, April 2015.

control prototyping (RCP) of electrical machines and drive systems employed in the industrial applications. According to their paralleled hardwired architecture, reconfigurability, large amount of logic resources, full-custom DSP units, and storage elements, currently available FPGA devices are able to satisfy the accuracy demands of machine models adequately by providing nanosecond computational clock cycle within the simulation time-step in real-time.

A number of studies have been conducted in this area with different objectives [15, 16, 27, 30, 35, 58]. Most methodologies adopted in the literature for the real-time simulation of machines on FPGA are based on fixed-point calculations [15, 16, 27, 30, 35] and a 32-bit floating-point hardware emulation of a synchronous generator used in the nodal analysis for power system transient simulation is presented in [58]. Since the need for a comprehensive comparison between fixed- and floating-point implementation, deeply pipelined and parallelled architecture, as well as schematic and textual programming language method have not so far been met, in this work the focus is to evaluate FPGA-based real-time emulation of the machine models in a general framework. This chapter explains the mathematical modeling and numerical techniques for digital realization of electrical machines. Implementation of machine model by different approaches are presented thereafter. The various designed architectures in terms of real-time simulation time-step sizes and hardware resource consumption as well as accuracy are then evaluated. Finally, the usefulness of FPGA-based real-time emulated machine models are assessed and highlighted by providing a number of case studies (induction motor, synchronous generator, line start-permanent magnet synchronous motor, and DC motor).

3.2 FPGA Design Approaches for Electrical Machine Realization

The techniques to emulate a system on FPGA can be broadly classified into two groups: (a) textual programming language (TPL) such as hardware description language (HDL), and (b) schematic method from vendor specific blocksets. The textual programming method by means of HDL such as VHDL or Verilog HDL is a powerful method to develop a digital hardware design without any restrictions. However, it can be very complex and cumbersome to debug. Even experts in machine modeling and simulation may find programming in HDL a daunting task. The schematic or the model-based method relies on a library of basic combinatorial and sequential building blocks offered in the Altera DSP Builder®, Xilinx System Generator® blocksets, *etc.* within the Matlab/Simulink® environment. This method allows users to go from system simulation using the industry-standard Mathworks simulation tools to hardware implementation in a short time. This method is user-friendly, easier to troubleshoot, easily understandable, and specifically useful for novice users of embedded digital systems. Expert users will of course be able to save development time by adopting Intellectual Property (IP) core blocks already available in such blocksets with no HDL hard-coding. However, this approach is still limited to applications in which no complex sequencers and deeply pipelined structures are required [51].

Furthermore, the FPGA is a space-oriented logic device that enables full hardwired parallelism to be achieved to the extent permitted by the implemented user model and algorithm. A large number of customized parallel processing units can be easily configured. Thus, a highly parallel implementation is the best realization according to the FPGA architecture (one data per hardware module per time-step) for the algorithms where a lower amount of hardware resources for computational processing is required such as fixed-point arithmetic based designs. The integrated massive memory blocks can be partitioned into many independent types such as RAM, ROM, FIFO, single port, or dual-port user memory units through which multiple data can be accessed simultaneously. Although multiple data can be processed in parallel on an FPGA, due to resource limitation, it is difficult to achieve the ideal parallelism for large systems and not area-optimized for the algorithms that utilize massive amount of hardware resources such as floating-point calculation based designs. In such cases, the pipelining technique has to be used. In a pipelined scheme, a function is divided into several stages by inserting registers between stages, allowing multiple data to be processed at different stages at once, resulting in a high computational throughput (multiple data per hardware module per time-step).

As a consequence of the above reasons, the seamless and user-friendly schematic method is the best technique for a highly parallel implementation of real-time machine model based on fixed-point operations, while on the other hand, the TPL method is the most appropriate approach for the pipeline realization of floating-point calculation based real-time system, resulting in an area-optimized hardware architecture acceptable in the industrial applications.

In this chapter, floating-point number calculations in the deeply pipelined scheme are employed for an FPGA-based real-time emulation of commonly used electrical machines to support a wide range of machine specifications and characteristics. Moreover, fixed-point algorithm of the models are implemented for the sake of a complete comparison. The implementation is carried out by state-space approach to provide a unified framework. Therefore, not only the real-time emulation of electrical machines can be realized by this approach, but also the presented methodology can be used for the implementation of other systems such as mechatronics, aerospace, and control systems that can be expressed in terms of state-space equations.

3.3 State-Space Representation of Machine Models

The governing equations describing magnetically coupled stator and rotor circuits in an electrical machines, whose windings are identical, and symmetrically placed and has constant parameters, can be written as follows:

$$\begin{bmatrix}\mathbf{V}_{abc_s}\\ \mathbf{V}_{abc_r}\end{bmatrix} = \begin{bmatrix}\mathbf{r}_s & \mathbf{0}\\ \mathbf{0} & \mathbf{r}_r\end{bmatrix}\begin{bmatrix}\mathbf{I}_{abc_s}\\ \mathbf{I}_{abc_r}\end{bmatrix} + p\begin{bmatrix}\boldsymbol{\lambda}_{abc_s}\\ \boldsymbol{\lambda}_{abc_r}\end{bmatrix},$$

$$\begin{bmatrix}\boldsymbol{\lambda}_{abc_s}\\ \boldsymbol{\lambda}_{abc_r}\end{bmatrix} = \begin{bmatrix}\mathbf{L}_{abc_{ss}} & \mathbf{L}_{abc_{sr}}\\ \mathbf{L}_{abc_{rs}} & \mathbf{L}_{abc_{rr}}\end{bmatrix}\begin{bmatrix}\mathbf{I}_{abc_s}\\ \mathbf{I}_{abc_r}\end{bmatrix}. \tag{3.1}$$

where $\mathbf{V}$, $\mathbf{I}$, $\boldsymbol{\lambda}$, $\mathbf{r}$, and $\mathbf{L}$ denote voltage, current, flux linkage, resistance, and inductance matrices, respectively. Additionally, subscript letters including abc, r, and s are phase domain, rotor, and stator indexes, respectively.

The equations representing the dynamic behavior of machines consist of an inductance matrix as a function of rotor position. Thus, a change of frame is used to reduce the complexity of this matrix by referring the machine variables to a reference frame that rotates at an arbitrary velocity given by [7]:

$$\mathbf{f}_{qd0} = \mathbf{K}(\theta)\ \mathbf{f}_{abc},$$

$$\mathbf{K}(\theta) = \frac{2}{3}\begin{bmatrix}\cos\theta & \cos(\theta-\frac{2\pi}{3}) & \cos(\theta+\frac{2\pi}{3})\\ \sin\theta & \sin(\theta-\frac{2\pi}{3}) & \sin(\theta+\frac{2\pi}{3})\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{bmatrix}, \tag{3.2}$$

where **f** represent voltage, current, or flux linkage vector of stator and rotor circuits. Also, $\mathbf{K}(\theta)$ is a transformation matrix and θ is arbitrarily selected.

Using state-space approach in the orthogonal *qd* axis model, the simulation of electrical side of various machines can be expressed as:

$$\begin{cases}\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}\\ \mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}\end{cases}, \tag{3.3}$$

where $\mathbf{x} \in \mathbb{R}^h$ is the state vector, $\mathbf{u} \in \mathbb{R}^p$ is the input vector, and $\mathbf{y} \in \mathbb{R}^q$ is the output vector. Plus, $\mathbf{A}_{h\times h}$, $\mathbf{B}_{h\times p}$, $\mathbf{C}_{q\times h}$, and $\mathbf{D}_{q\times p}$ denote state, input, output, and feedback matrices in state-space equations. Flux linkages ($\boldsymbol{\lambda}_{qd}$), supply voltages ($\mathbf{V}_{qd}$), and output currents ($\mathbf{I}_{qd}$) are selected as the state, input, and output variables, respectively. The matrices and vectors in (3.3) are defined in Appendix A for different types of machines.

The second set of equations is for the mechanical side of machines for which electromagnetic torque (T_e), and rotor speed (ω_r) are the input and state vectors, represented by

$$T_e = \left(\tfrac{3}{2}\right)\left(\tfrac{P}{2}\right)\left(\lambda_{ds}i_{qs} - \lambda_{qs}i_{ds}\right), \tag{3.4}$$

$$\dot{\omega}_r = \tfrac{P}{2J}T_{effective} = \tfrac{P}{2J}\left(T_e - T_{mech} - T_{damp}\right). \tag{3.5}$$

where T_e, T_{mech}, and T_{damp} represent electromagnetic, mechanical, and damping torques. Moreover, rotor inertia, and number of pole pairs are denoted by J, and P, respectively.

For the digital hardware realization, the differential equations of machine model should be discretized. The implicit techniques for discretization are more expensive due to a root finding procedure at any given time-step (T_s). The important observation regarding explicit methods are that the unknown quantities at each time-step are given in terms of history parameters. Thus, the discretized equations with explicit technique can be computed within the shorter elapsed time compared with the implicit one with the same order, resulting in a reduced local truncation error at every time-step, as well as a reduced global error, and a higher simulation accuracy. However, the drawback arises from the limitation on the time step-size in the explicit method to ensure numerical stability. Since the FPGA can provide nanosecond computation clock cycles, a very small simulation time-step, which is much smaller than electrical and evidently mechanical time constants of machines can be achieved to capture all transients of machine behavior. Thus, the numerical stability is of no concern for the FPGA-based real-time emulation of machines.

The explicit Adams-Bashforth (A-B) method can be employed as a good choice to discretize the machine's equations based on $s = \frac{z-1}{T_s}$ for first-order or $s = \frac{2}{T_s}\frac{z^2-1}{3z-1}$ for second-order Adams-Bashforth transformation [59]. In this chapter, first-order Adams-Bashforth method (Forward Euler) is chosen for the discretization of state-space equations and several implementations in the literatures have also used the Forward Euler method [42, 45] as follows:

$$\begin{cases} \mathbf{x}(t) = \mathbf{x}(t - T_s) + T_s \times [\mathbf{A}(t - T_s)\mathbf{x}(t - T_s) + \mathbf{B}(t - T_s)\mathbf{u}(t - T_s)], \\ \mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t). \end{cases} \tag{3.6}$$

3.4 System Configuration on FPGA

3.4.1 Number Representation

Choosing either a fixed- or floating-point number representation is the first step for any hardware design. To provide a comparison for hardware resource utilization and achieved accuracy, a 32-bit single precision floating-point format (*IEEE* Standard 754) by TPL method (VHDL) and 32-bit fixed-point format (binary point is located where the integer part of the maximum or minimum value of flowing data is effectively fitted in the underlying integer part of fixed-point number for the highest accuracy) by schematic method are used for real-time emulation in this work.

3.4.2 Realization of Machine Models

The FPGA design procedure basically involves the design entry step using TPL or Schematic method to configure the system model and the implementation step to generate the downloadable bitstream.

In this work, the realization approach can be applied to any type of FPGA devices such as Altera®, Xilinx®, Lattice®, *etc.* In the developed machine blocks for real-time emulation, TPL (HDL coding) and schematic method for different hardware platforms remain the same and the only difference is between the IP cores of one FPGA type to another designed just for the implementation of basic arithmetic operations.

a) Floating-point implementation by VHDL

The hardware modules that assist the real-time emulation to be executed include: `Main Control Module`, `Source Module`, `Timer & Switch Module`, and `Electrical Machine Module`.

Fig. 3.1 shows the overall procedure in a simulation time-step in the proposed hardware. First, `Source Module` generates input voltages for the three phase machine terminals, whereas, `Switch Module` checks the switch states to apply or remove faulty conditions and load torque. Then `Electrical Machine Module` starts to solve machine's equations. It is obvious that the parallel processing exists in each stage, while preserving the necessary sequential stages in the overall simulation algorithm.

The `Main Control Module` coordinates the operation of the whole emulator to carry out the algorithm. It sends out control signals (`Simu_on`, `Socmd`, `EMcmd`) to each module to perform the required functions. Meanwhile, it receives the acknowledged signals (`Sodone`, `EMdone`, `dtOver`, `Sw`) to judge if the functions are done or the switches are on or off.

Using the `rst` function on FPGA board, the digital hardware emulator starts executing real-time simulation. At first, `Main Control` sends out a command signal (`Socmd`) to the `Source Module` for generating supply voltages while it checks the switch status signals (`Sw0`, `Sw1`, `Sw2`, ...) to apply or remove voltage source to electrical machine terminals or mechanical torque to the machine shaft. Once acknowledge signals are received from `Source` and `Switch Modules`, main control sends out a command signal (`EMcmd`) to the `Electrical Machine Module` to compute the state variables of the machine. Then, it waits to be given acknowledge signal (`EMdone`) from machine module and transmits real-time signals to output ports of emulator. Finally, `Main Control` checks whether or not real-time procedure is performed within the simulation time-step (`dtOver`), and decides to go to the next simulation time-step or send the error signal to output ports and terminate the real-time emulation procedure.

The voltage source in the `Source Module`, as shown in Fig. 3.2, are represented using sinusoidal function (cos function). The look-up table (LUT) is the most commonly used method to evaluate this nonlinear function. Since cos is a periodic function only half cycle of cos function values need to be stored in the LUT in order to save the memory space of the LUT. The accuracy of the cos value is determined by the length of the LUT. In this design, 4096 (2^{12}) cos values for half cycle are stored in the LUT; thus, the $Resolution$ of the LUT is $d\theta = \pi/4096$. The calculation of the source values begins with the updating of the `phase` angle. The `new_phase` is obtained by adding the previous `phase` by ωT_s, where ω denotes angular frequency. Since the LUT has only half cycle of cos functions values, the calculated

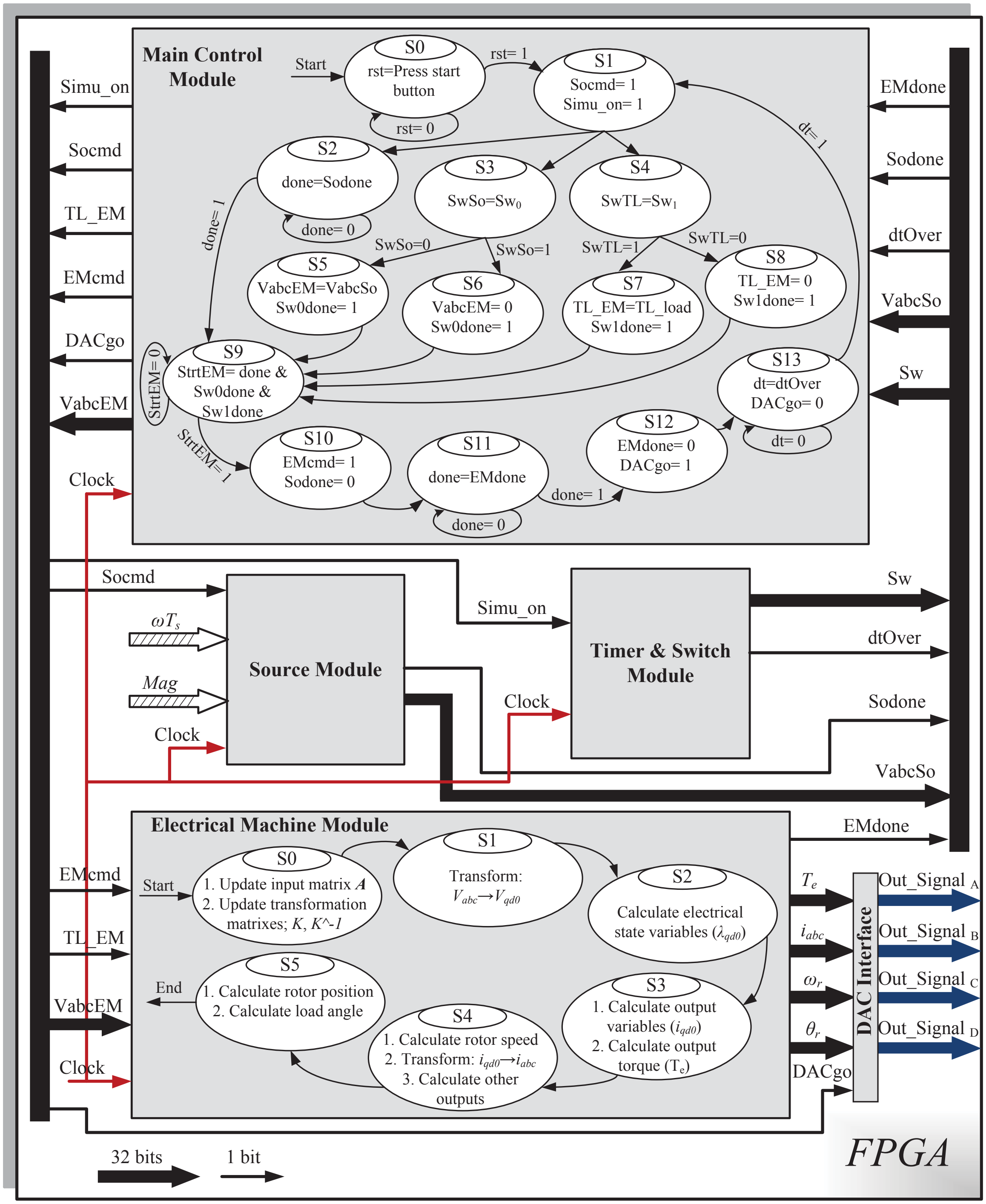
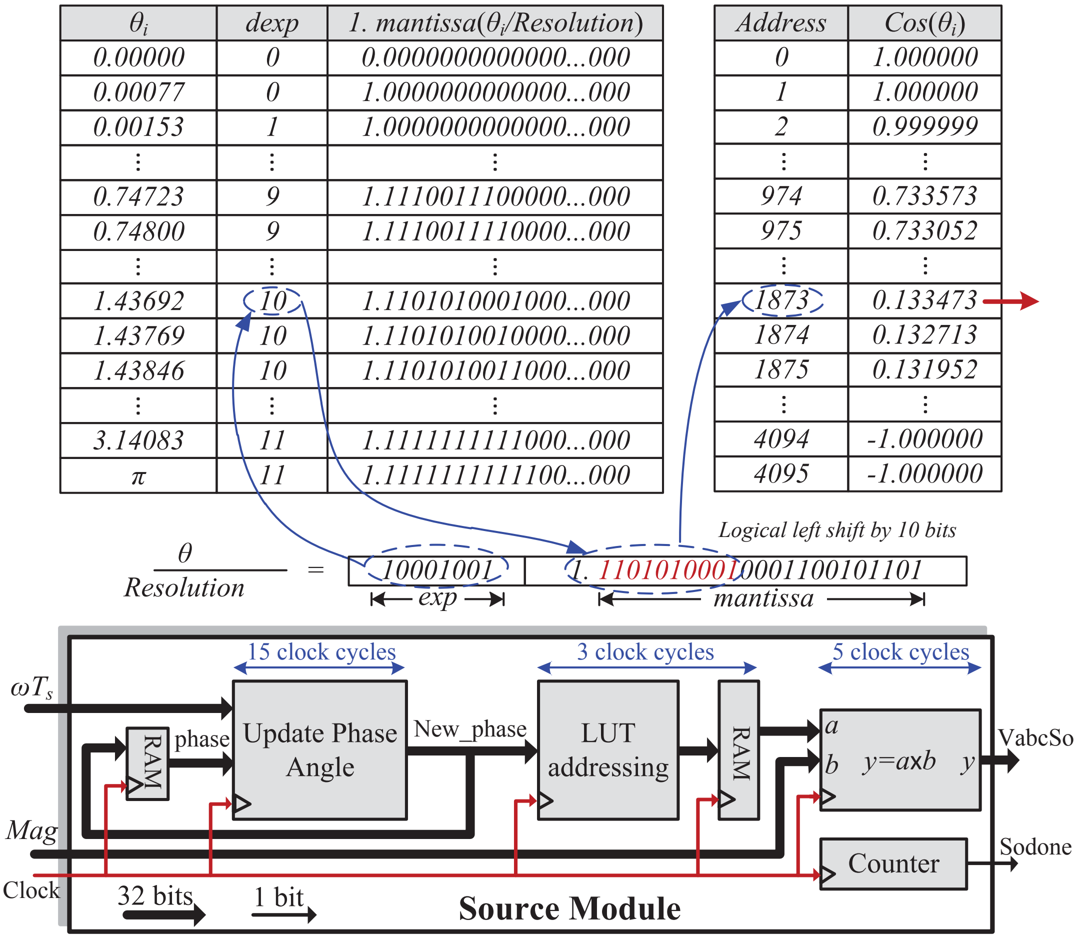


Figure 3.1: Finite-state machine for paralleled and deeply pipelined real-time algorithm for FPGA implementation of electrical machines.



θ_i	*dexp*	*1. mantissa*(θ_i/*Resolution*)
0.00000	0	0.0000000000000...000
0.00077	0	1.0000000000000...000
0.00153	1	1.0000000000000...000
⋮	⋮	⋮
0.74723	9	1.1110011100000...000
0.74800	9	1.1110011110000...000
⋮	⋮	⋮
1.43692	10	1.1101010001000...000
1.43769	10	1.1101010010000...000
1.43846	10	1.1101010011000...000
⋮	⋮	⋮
3.14083	11	1.1111111111000...000
π	11	1.1111111111100...000

Address	*Cos*(θ_i)
0	1.000000
1	1.000000
2	0.999999
⋮	⋮
974	0.733573
975	0.733052
⋮	⋮
1873	0.133473
1874	0.132713
1875	0.131952
⋮	⋮
4094	-1.000000
4095	-1.000000

Figure 3.2: Pipelined configuration for calculating `Source Module`.

`phase` needs to be checked with π; if greater than π, it is subtracted by π and the sign of the result is inverted. Then, the `new_phase` is converted to the address of the LUT. Finally, the retrieved *cos* value is multiplied by the magnitude `Mag`. The exponent and mantissa of the input floating-point number are used directly to access the LUT when the step length is always a power of 2. Assume the floating-point input is θ, the `LUT addressing` unit in Fig. 3.2 outputs the addresses of the point θ_i making $\theta - \theta_i < Resolution$, where $Resolution$ is the interval of the LUT. This is done by left shifting the leading '1' and mantissa of $\theta/Resolution$ by *dexp* bits, where the *dexp* is the exponent of $\theta/Resolution$ (without bias). An example is shown in Fig. 3.2. In this example, input θ is 1.437 whose *dexp* is 10 (without bias). Left shifting the leading 1 and mantissa 10 bits gives 1873 which is the address of $\theta_i = 1.43692$ and $cos(\theta_i) = 0.133473$ [81].

A hardware module of `Timer & Switch` is designed to simulate switches. Fig. 3.3 shows the details of this module and its input/output signals. Since the switches are time controlled, the core of this module is a real-time clock generator. The best achievable clock

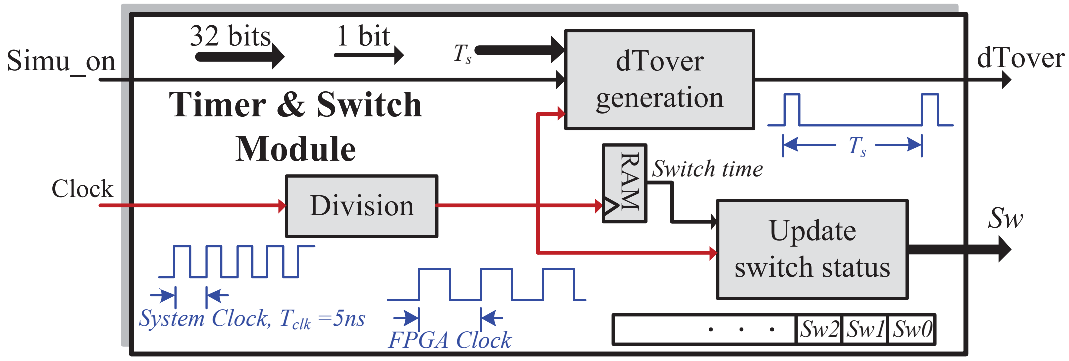
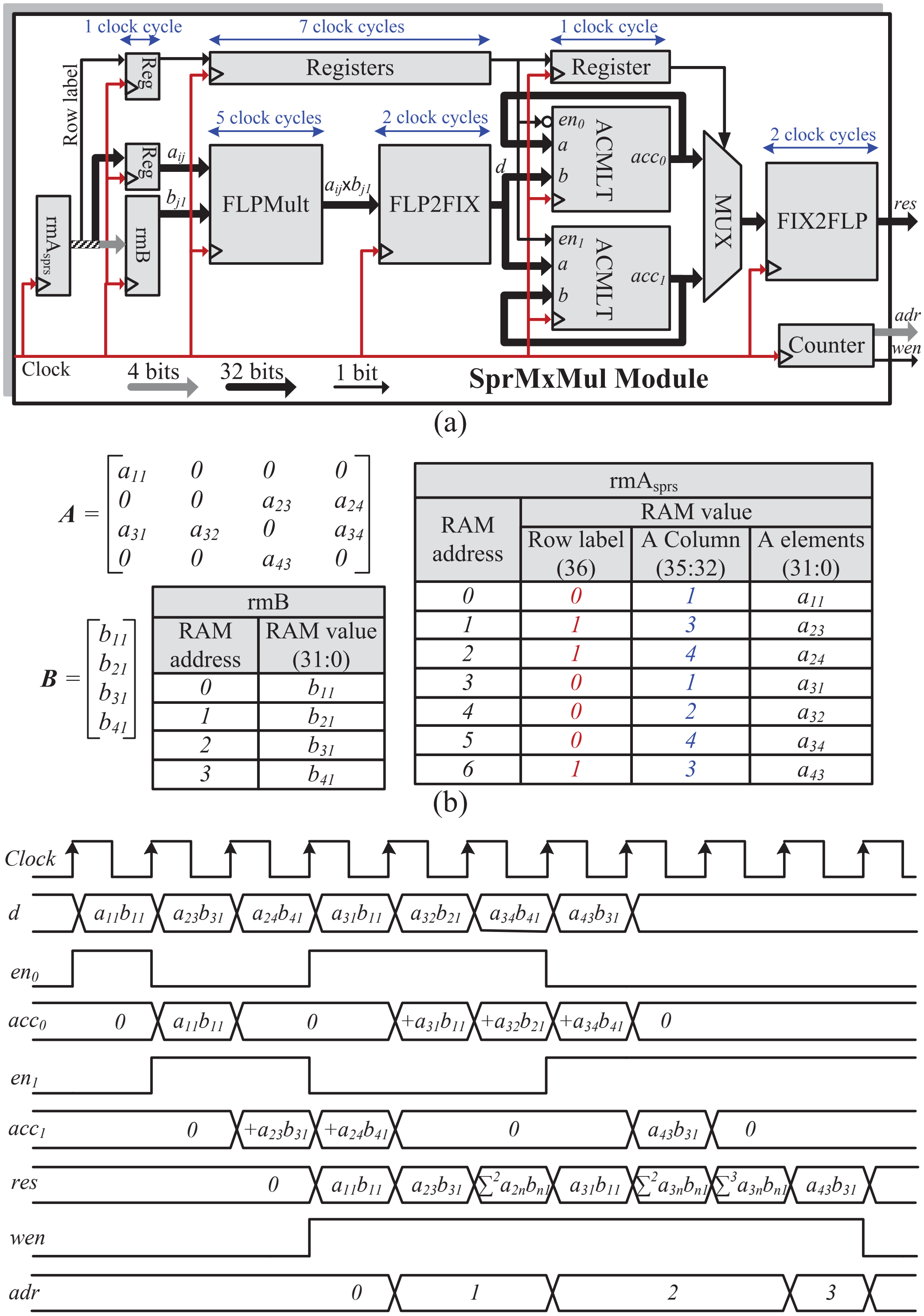


Figure 3.3: Functions realized in the `Timer & Switch Module` [47].

frequency is generated by the input system clock frequency (200MHz). This clock signal is counted and compared with the switch operation times saved in a RAM. Once the switch times are reached, the corresponding switch state bit in `Sw` register (`Sw0`, `Sw1`, `Sw2`, ...) is inverted using ′0′/′1′ for switch open/closed. Another important function of the `Switch Module` is to generate the `dtOver` signal which indicates the end of the real simulation time-step T_s. When a simulation step is finished, the T_s-over signal is checked. If it is not ′1′, the simulation step is finished within T_s, thus, the real-time simulation is achieved. Otherwise, the simulation step takes longer time than T_s, and the real-time constrain is not met [60].

The finite-state machine (FSM) diagram for the hardware realization of machine models is depicted in Fig. 3.1. The procedure starts with the update of input Matrix ($\mathbf{A}$) using the calculated speed from the previous time-step in parallel with the computation of transformation matrices ($\mathbf{K}$ and $\mathbf{K}^{-1}$) using sinusoidal function look-up table (LUT) based on information about rotor and reference frame position from the passed time-step in state S_0. Then, in state S_1, $\mathbf{V}_{abc}$ is transformed to an arbitrary reference frame (qdo) to provide input vector for state-space equations. Once $\mathbf{V}_{qd}$ is available, the electrical state variables ($\boldsymbol{\lambda}_{qd}$) are calculated in state S_2. The output state variables ($\mathbf{I}_{qd}$) and output torque (T_e) are computed concurrently in state S_3. The $\mathbf{I}_{qd}$ is transformed back to abc frame in state S_4, meanwhile, rotor speed (ω_r) and other required outputs are obtained. Finally, rotor position and load angle are calculated simultaneously in state S_5. Obviously, since the implementation involves 6 sequential steps, it is time-consuming. To improve the hardware computational efficiency of the deeply pipelined modules, all possible parallel processing paths have been taken into account, although the overall procedure is sequential.

The hardware realization of electrical machines is designed and performed by arrangement and manipulation of sparse matrix multiplication (`SprMxMul`) and floating-point multiply-add/subtract (`FLPMAS`) units. These submodules can be chained to realize many functions and computations such as: simple addition/subtraction, multiplication, conversion between floating- and fixed-point numbers, $a{\times}b{\times}c$, $a{\times}b{\pm}c{\times}d{\pm}e{\times}f$, $(a{\pm}b){\times}c$, $\mathbf{A}_{n\times1}{\pm}c{\times}\mathbf{B}_{n\times1}$,


1 clock cycle
7 clock cycles
1 clock cycle
Reg
Registers
Register
Row label
5 clock cycles
2 clock cycles
a_{ij}
Reg
b_{j1}
rmB
rmA_{sprs}
FLPMult
a_{ij}xb_{j1}
FLP2FIX
d
en_0
a
b
ACMLT
acc_0
en_1
a
b
ACMLT
acc_1
MUX
2 clock cycles
FIX2FLP
res
adr
Counter
wen
Clock
4 bits
32 bits
1 bit
SprMxMul Module
Clock
d
$a_{11}b_{11}$ $a_{23}b_{31}$ $a_{24}b_{41}$ $a_{31}b_{11}$ $a_{32}b_{21}$ $a_{34}b_{41}$ $a_{43}b_{31}$
en_0
acc_0
0 $a_{11}b_{11}$ 0 $+a_{31}b_{11}$ $+a_{32}b_{21}$ $+a_{34}b_{41}$ 0
en_1
acc_1
0 $+a_{23}b_{31}$ $+a_{24}b_{41}$ 0 $a_{43}b_{31}$ 0
res
0 $a_{11}b_{11}$ $a_{23}b_{31}$ $\sum^2 a_{2n}b_{n1}$ $a_{31}b_{11}$ $\sum^2 a_{3n}b_{n1}$ $\sum^3 a_{3n}b_{n1}$ $a_{43}b_{31}$
wen
adr
0 1 2 3

(a)

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & 0 & a_{23} & a_{24} \\ a_{31} & a_{32} & 0 & a_{34} \\ 0 & 0 & a_{43} & 0 \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{41} \end{bmatrix}$$

rmB	
RAM address	RAM value (31:0)
0	b_{11}
1	b_{21}
2	b_{31}
3	b_{41}

rmA_{sprs}			
RAM address	RAM value		
	Row label (36)	A Column (35:32)	A elements (31:0)
0	0	1	a_{11}
1	1	3	a_{23}
2	1	4	a_{24}
3	0	1	a_{31}
4	0	2	a_{32}
5	0	4	a_{34}
6	1	3	a_{43}

(b)

(c)

Figure 3.4: `SprMxMul` unit. (a) Hardware design, (b) RAM initialization, (c) Timing diagram.

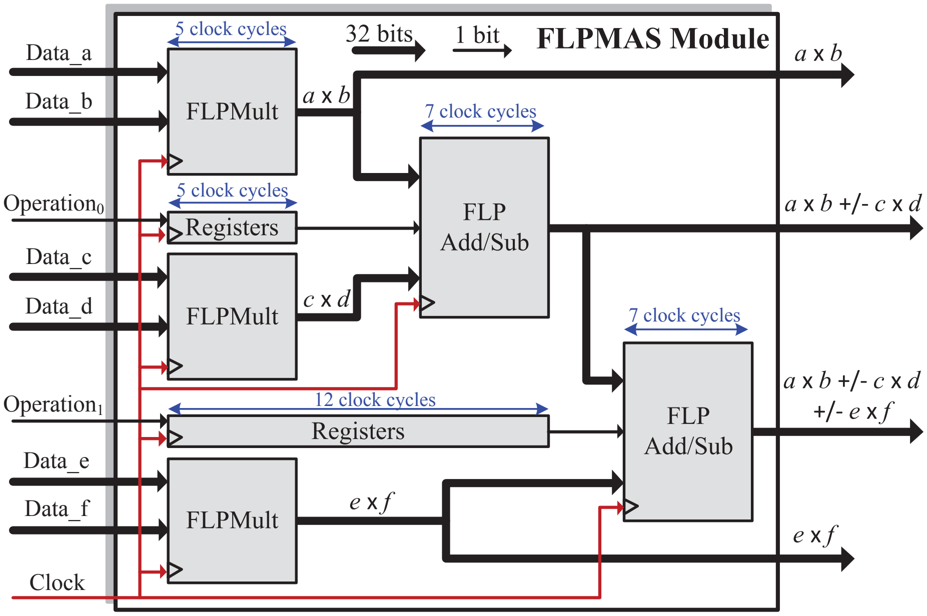


Figure 3.5: Pipelined realization of `FLPMAS Module`.

$\mathbf{A}_{n\times n}\times\mathbf{B}_{n\times 1}\pm c\times\mathbf{D}_{n\times 1}$, and *etc*.

A fast sparse matrix multiplication submodule where a compact sparse matrix storage format (Fig. 3.4) which uses only one vector is defined for the realization of $\mathbf{A}_{n\times n}\times\mathbf{B}_{n\times 1}$. Each entry in this format has the following: 1) a 32-bit value to store the subsequent nonzero value of matrix $\mathbf{A}_{n\times n}$ in row order; 2) a 4-bit column number to identify column index of this nonzero value; and 3) a 1-bit row index to label all non-zero values in the same row with '0' or '1'. Fig. 3.4(b) shows an example sparse matrix and its storage format. In the `SprMxMul` submodule, the accumulation is done in fixed-point format. The reason is that floating-point accumulator has a much longer latency and requires much more logic resources to implement. The fixed-point accumulator needs only one adder with one clock cycle latency. The used fixed-point number format is 40.100, which has 40 integer bits and 100 fraction bits to guarantee both the range and precision [60]. As shown in Fig. 3.4(a), the `SprMxMul` submodule contains one floating-point multiplier, one floating-to-fixed-point converter, two fixed-point adders, and one fixed-point converter. The elements of sparse matrix $\mathbf{A}_{n\times n}$ are retrieved from RAM A_{sprs}, while 4-bit column indexes are used to access the $\mathbf{B}_{n\times 1}$ matrix stored in RAM B. The registers are inserted for synchronization in the computation. The realized matrix-vector multiplication is fully pipelined and fast because there is no stall between two consecutive matrix row-vector multiplications. This is achieved by the two parallel fixed-point adders (accumulators) acc_0 and acc_1 with opposite enable inputs en_0 and en_1 controlled by the row label information. Fig. 3.4(c) shows the logic timing diagram for the `SprMxMul` unit based on the example in this figure. As can be seen, the accumulation of the first matrix row-vector multiplication is processed in the acc_0, while the acc_1 is reset to zero, which makes it ready for the accumulation of the next matrix row-vector multiplication.

Moreover, the basic floating-point functions including addition/subtraction and multiplication are combined to build a basic arithmetic unit. Fig. 3.5 presents the data flow of various outputs of this submodule. The latency of the longest path is 19 clock cycles. It can be used for floating-point arithmetic operations of both scalar and vector quantities, for example the realization of Eqs. (3.4)-(3.6). The `FLPMAS` unit is pipelined to achieve high data throughput.

b) Fixed-point implementation by schematic method

The schematic method is performed under Matlab/Simulink software environment. As shown in Fig. 3.6, the procedure is started by the development of a functional model of a system using basic Simulink continuous-time blocks or M-file coding. The behavior of the developed model is verified by the performance of the machine model from SimPowerSystem toolbox and the fixed-point formats are then defined for all coefficients, variables, and data paths inside the model. In addition, the number of required clock cycles and suitable real-time simulation time-step size are determined to compute all governing equations of the system model.

At this stage, the emulation is realized by the development of a digital fixed-point model by using Xilinx System Generator (XSG). The final step is linked with FPGA-based implementation. The HDL code of the design architecture is automatically generated. The main control and interface files as well as pin assignments suitable for the FPGA platforms or HIL test setups are written and organized. The modifications and adjustments are performed in this step to get a balance in terms of area/time performances. Synthesis, map, place and route processes and analysis of the static timing performances are done in the Xilinx® ISE software. The binary files produced are then transferred to FPGA via a serial interface (JTAG) for its reconfiguration.

The generated FPGA digital and analog I/O signals, which transmit low-level voltages and currents, can be interfaced with an amplifier that generates and absorbs high-level power where an actual real-time emulated machine works in the HIL configuration.

A generic digital machine model implementation is shown in Fig. 3.7. It mainly consists of five functional steps. The algorithm starts from the calculation of voltage source in the three-phase domain. Phase angle is incremented over each time-step to feed the Xilinx DDS Compiler block, which implements high performance, optimized phase to sinusoidal circuits. The core sources sinusoidal waveforms and consists of a SIN/COS look up table. Moreover, the appropriate voltage magnitude can be achieved by a multiplication unit at the output port of this module [61].

The transformation module is in charge to change *abc* to an arbitrary reference frame. As can be seen in the associated unit for *abc* to stationary transformation, to obtain V_q,

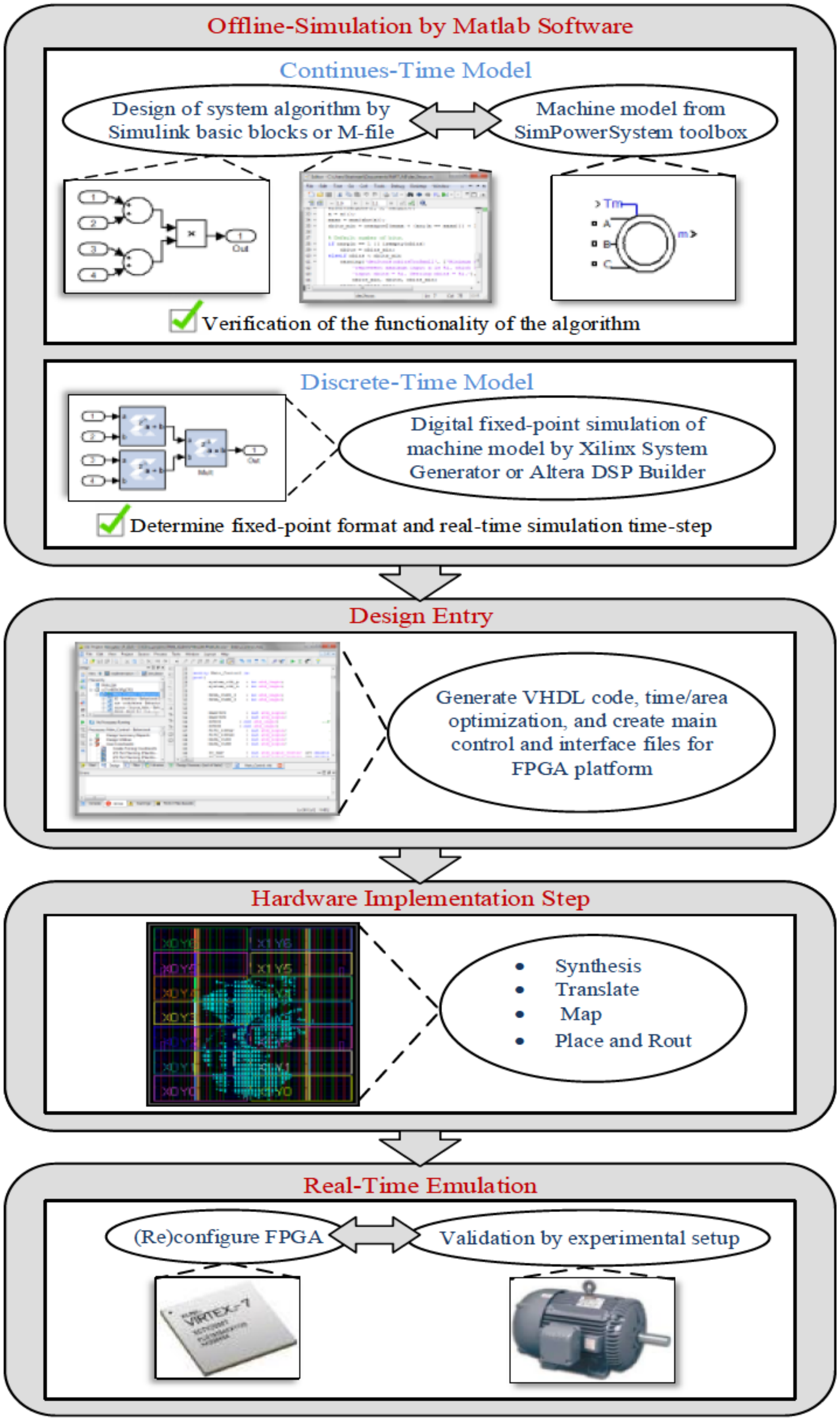

Offline-Simulation by Matlab Software
Continues-Time Model
Design of system algorithm by Simulink basic blocks or M-file
Machine model from SimPowerSystem toolbox
Verification of the functionality of the algorithm
Discrete-Time Model
Digital fixed-point simulation of machine model by Xilinx System Generator or Altera DSP Builder
Determine fixed-point format and real-time simulation time-step
Design Entry
Generate VHDL code, time/area optimization, and create main control and interface files for FPGA platform
Hardware Implementation Step
Synthesis
Translate
Map
Place and Rout
Real-Time Emulation
(Re)configure FPGA
Validation by experimental setup
VIRTEX-7

Figure 3.6: FPGA design flow by schematic method.

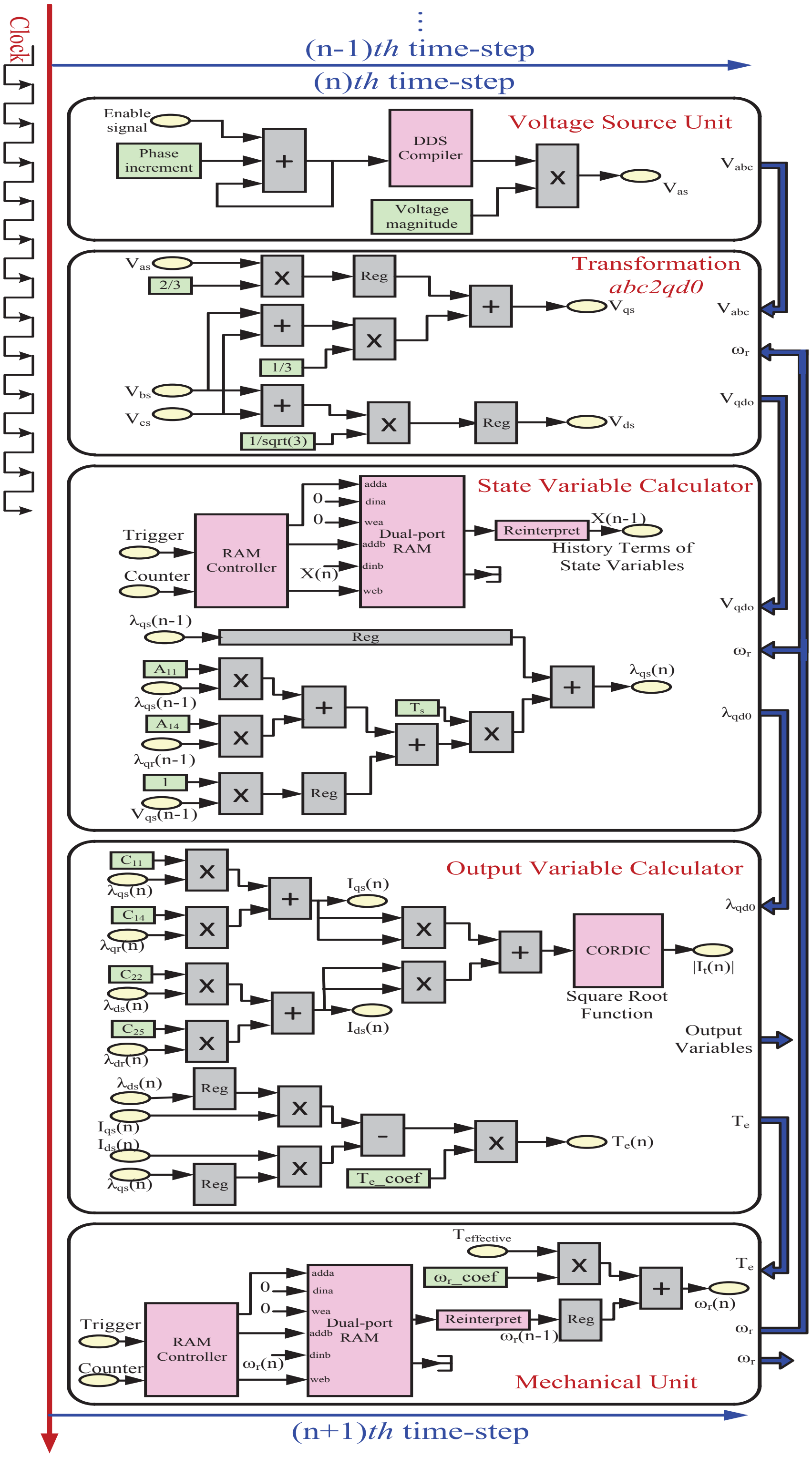


Figure 3.7: Design configuration for one emulation step of induction motor by Schematic method.

two sets of calculation (*1.* Production of $2/3$ and V_a, and *2.* Production of $1/3$ with the summation of V_b and V_c) must be performed in parallel mode with the same latency before the final addition happens. This is why a register is inserted in the first path to synchronize the data flow with the second parallel path. Consequently, the final addition is only done when the operations at the two input ports are completed at the same time. A similar strategy is applied to compute V_d and also can be used for the transformation of other reference frames.

The state-space variable calculator, depicted in Fig. 3.7, includes a memory controller that schedules the reads and writes to memory, and a highly parallel processing unit that computes state variables. To show how to implement a state-space system without losing the generality of the problem, the first row of the induction motor state equations is expanded and realized. All potential parallelism and data synchronism by registers are considered in the data flow graph of the following equation:

$$\begin{aligned}\lambda_{qs}\left((n)T_s\right) = \lambda_{qs}\left((n-1)T_s\right) + T_s \times [A_{11} \cdot \lambda_{qs}\left((n-1)T_s\right)\\ + A_{14} \cdot \lambda_{qr}\left((n-1)T_s\right) + V_{qs}\left((n-1)T_s\right)].\end{aligned} \tag{3.7}$$

In this step, the dual-port RAMs are employed to store the history terms of the state variables in the $(n-1)th$ time-step and call them for the integration procedure in the $(n)th$ time-step.

Once the parallel computation of the state variables are performed, the required output variables are calculated. Most of output variables such as stator and rotor currents, electromagnetic torque can be obtained by using elementary operators including adders, multipliers, multiplexers, and registers (calculation of stator current and electromagnetic torque are provided in the corresponding module in Fig. 3.7). However, to obtain some output variables, for example, the magnitude of a space vector, a square-root function is required, for which the generalized CORDIC function has been used.

Additionally, the structure of blocks arranged to calculate rotor speed is realized based on A-B integration technique. The RAM controller is responsible to manage the data flow of history information of rotor speed synchronized by the incoming current speed difference for an accumulation over each time-step. The same pattern is applied to calculate rotor position based on currently computed rotor speed information.

3.5 Evaluation of Designed Architectures

3.5.1 Real-Time Emulation Time-Step Size and Accuracy Assessment

Emulation time-step is an important criteria from practical point of view. In reality, currently available FPGA-based HIL test setups are not able to accommodate real-time models

Table 3.1: Latencies and Time-Step Sizes of Digital Hardware Realization.

Real-Time	Fixed-Point		Floating-Point	
Emulator	Latency	Time-Step	Latency	Time-Step
Induction Motor	46	230ns	234	1.755μs
Synchronous Generator	55	275ns	256	1.920μs
Line Start-PMSM	52	260ns	240	1.800μs
DC Motor	29	145ns	126	0.945μs

with a time-step more than a few microseconds due to practical limitations to communicate and transfer data to amplifiers and actual devices under real-time tests.

It is worth mentioning that although in the off-line simulation, an increase in the order of discretization method or in the precision of numbers and operations results undoubtedly in an increase in the accuracy of final results, in the real-time emulation it does not necessarily lead to an increase of accuracy.

As will be explained later about all case studies, with the same discretization method, single-precision floating-point implementation of machine models expends more time, listed in Table 3.1, to finish all calculations within the time-step compared with fixed-point. It means that the executed real-time emulation of floating-point realization is carried out with longer time-step size than that of fixed-point. Therefore, although floating-point calculations mathematically offer higher level of accuracy in comparison with fixed-point, an increase in the associated simulation time-step size of digital hardware implementation increases the truncation error at each time-step. Consequently, it may reduce the accuracy of the results in the time-marching or time-stepping digital real-time simulation. That is why in real-time emulation, an increase of discretization order and precision of calculations may not guarantee more accurate results, whereas in the off-line simulation they can.

It should be noted that the real-time hardware emulation of machine models in this work is realized by $5ns$ and $7.5ns$ of FPGA clock frequency for fixed- and floating-point implementation, respectively, resulting in smaller real-time emulation time-step sizes compared with [30, 58] (confirms better real-time implementation).

3.5.2 Off-line and Experimental Validation

The first validation step is an off-line simulation of design architecture according to the best case achievable frequency obtained at the end of place and rout report. This step can be performed by using ISim®, ModelSim®, or Matlab® software tools. The good functionality and accuracy of the design algorithm written by TPL or Schematic method can be verified in this step.

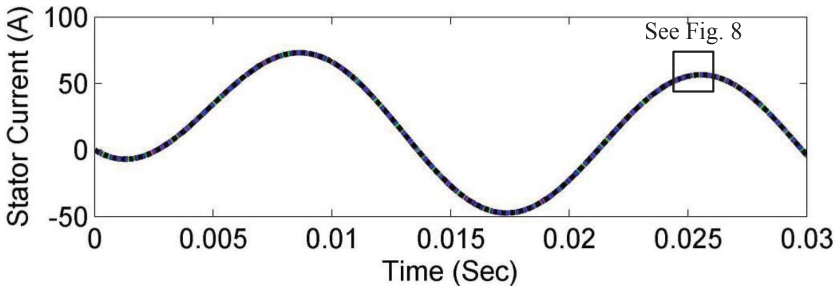


Figure 3.8: Stator current of induction machine using different methods presented in the caption of Fig. 3.9.

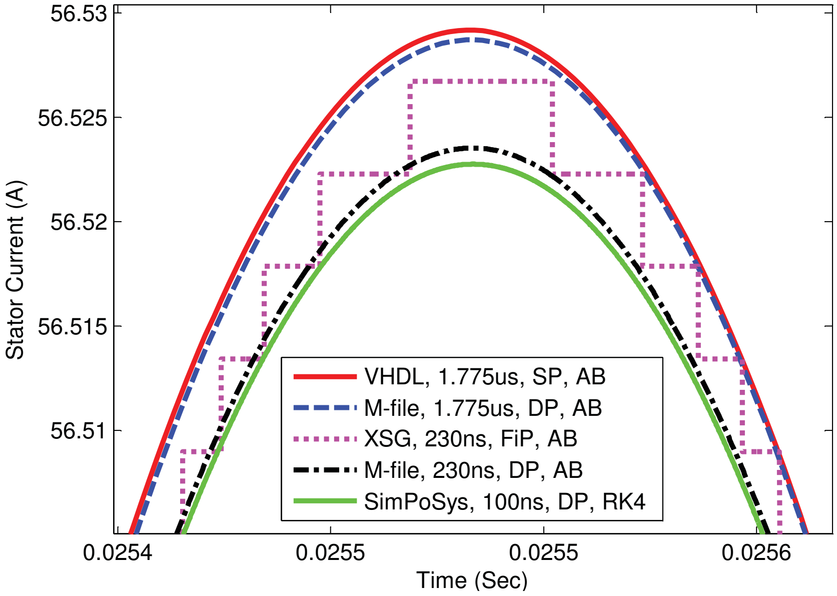


Figure 3.9: Zoomed in view of Fig. 3.8 identified by a legend (realization technique, time step-size, number representation, discretization method). Red solid line: (digital hardware realization by VHDL in ISim, $1.755\mu s$, SP, A-B), blue dashed line: (off-line simulation by M-file, $1.755\mu s$, DP, A-B), magenta dotted line: (digital hardware realization by XSG, $230ns$, FiP (32 bits), A-B), black dash-dot line: (off-line simulation by M-file, $230ns$, DP, A-B), and green solid line: (off-line simulation by SimPowerSystem tools, $100ns$, DP, RK4).

In the following transient off-line study, the first two oscillations of stator current of a Baldor induction machine, whose specifications are listed in Appendix A, by the various approaches are plotted in Fig. 3.8. The induction machine is started from stall. The reference solution is obtained using the qd induction machine model of SimPowerSystem toolbox and solved with Runge-Kutta fourth-order (RK4) method and double-precision (DP) operations using a small time-step of $100ns$. The simulation results obtained by Xilinx System Generator®, Matlab® M-file, and ISim® software are overlaid with the reference solution. As can be observed in Fig. 3.8, the transient responses produced by all

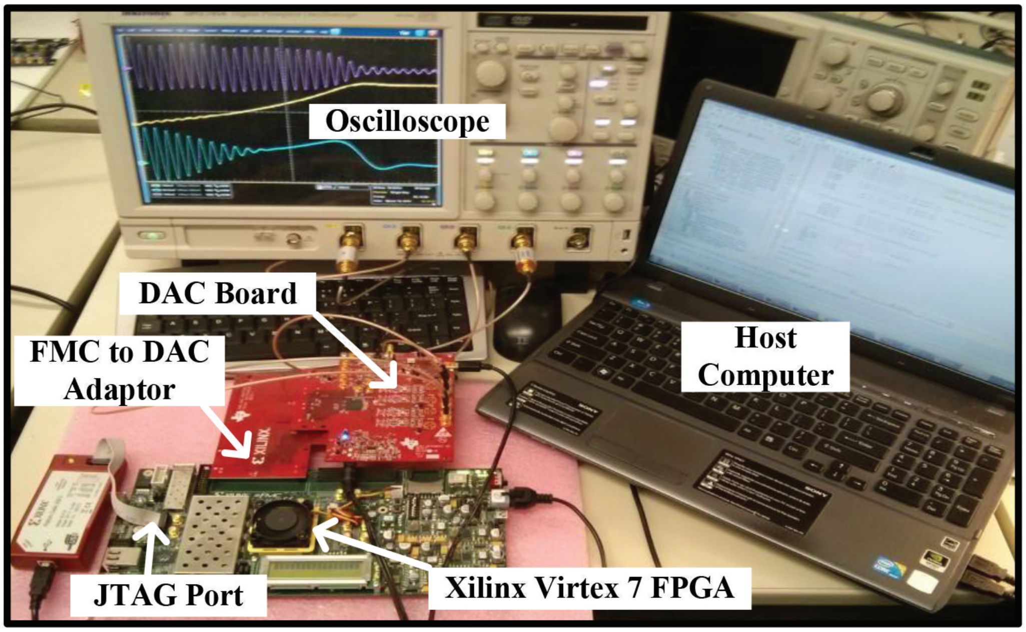
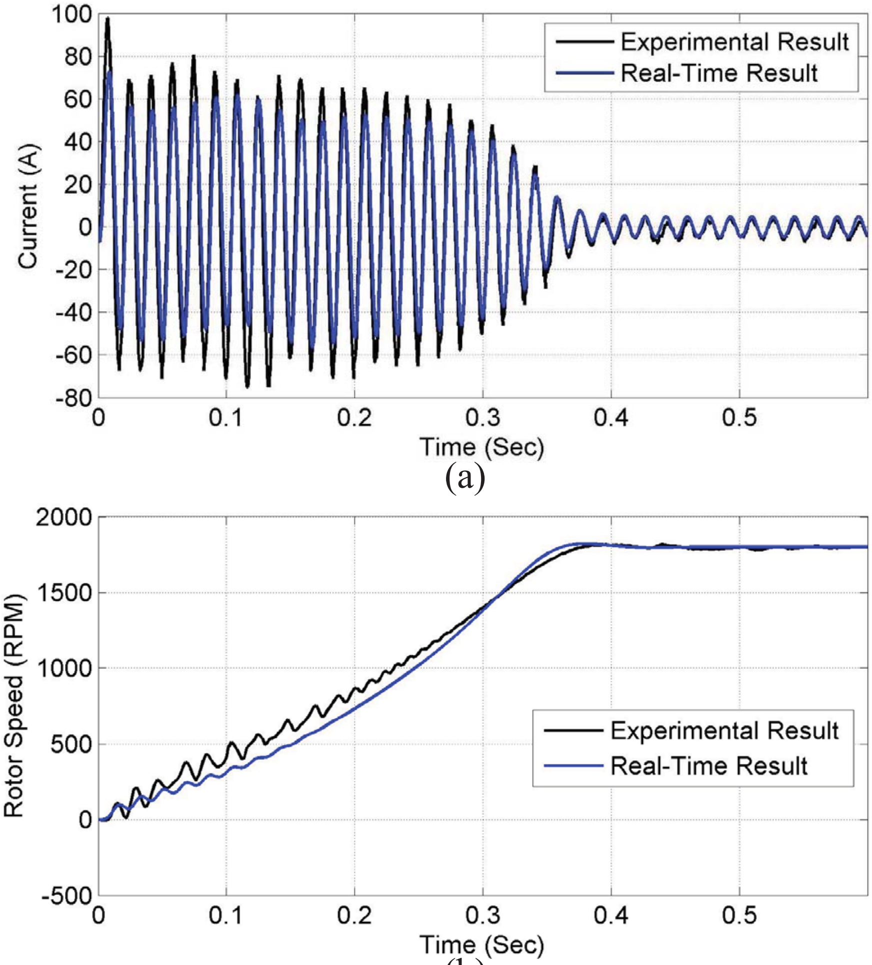


Figure 3.10: FPGA-based real-time emulator of induction machine.

methods coincide and converge to the reference solution. This clearly demonstrates that all approaches predict the machine behavior with the acceptable accuracy. A magnified fragment of Fig. 3.8 is also shown in Fig. 3.9 for better comparison. As seen in Fig. 3.9, the behavior of deeply pipelined digital hardware implementation of machine model by VHDL coding and single-precision (SP) floating-point operations obtained by ISim$^{\circledR}$ (red solid line) with time-step equal to $1.755\mu s$ gives the most deviation from the reference (green solid line), while a minor improvement can be achieved by double-precision calculations with the same algorithm and time-step (blue dashed line) obtained by Matlab M-file coding. The result produced by a digital fixed-point (FiP) hardware implementation (32 bits) of induction motor with $230ns$ time-step (magenta dotted line) by Schematic method (XSG) provides more accurate results compared with previous ones. The most noticeable difference of this waveform from others is its stair step shape. The reason is attributed to the errors that occur when a value lies outside the representable range and when the number of fractional bits is insufficient to represent the fractional portion of a value in fixed-point architecture. Moreover, double-precision calculated stator current by M-file coding (black dash-dot line) with the same condition of fixed-point simulation is also provided for further comparison.

The second validation step is done via a comparison between the FPGA-based real-time emulated machine model shown in Fig. 3.10, and an actual induction machine to make sure the FPGA-based model can truly duplicate the behavior of the machine in the virtual environment of HIL tests.



(a)

(b)

Figure 3.11: FPGA-based real-time emulated and experimentally measured results.

A $3hp$ Baldor induction machine, which is mechanically coupled to a DC generator is employed for experimental test. The experiment is carried out to capture the stator current and rotor speed when the test motor is started directly from three-phase power supply. During the test a current probe is used to capture the current waveform on the oscilloscope and rotor speed is measured by the use of an encoder mounted on the induction motor shaft.

To plot the experimental and real-time emulated results in the same figure with the high precision, the real-time data of output signals generated from FPGA-based emulator are exported by ChipScope$^{®}$ analyzer and laid over the real values captured by using a current probe and an encoder mounted on induction machine shaft.

As can be seen from Fig. 3.11, there is a good agreement between FPGA based emulated and experimentally measured induction motor current and rotor speed. Both currents decay to steady-state over almost $0.4s$. Furthermore, it is found that the simulated

Table 3.2: FPGA Resource Utilization of Real-Time Machine Emulator using Fixed-Point Arithmetic Operations by Schematic method.

Real-Time Emulator	Fixed-Point by Schematic method		
	Number of Slice Registers (607,200 available)	Number of Slice LUTs (303,600 available)	Number of DSPs (2,800 available)
Induction Motor	25,127	36,567	26
Synchronous Generator	42,287	61,024	43
Line Start-PMSM	36,915	53,298	28
DC Motor	9,886	14,551	9

Table 3.3: FPGA Resource Utilization of Real-Time Machine Emulator using Floating-Point Arithmetic Operations by TPL.

Real-Time Emulator	Floating-Point by TPL		
	Number of Slice Registers (607,200 available)	Number of Slice LUTs (303,600 available)	Number of DSPs (2,800 available)
Induction Motor	6,689	29,115	38
Synchronous Generator	7,670	30,493	67
Line Start-PMSM	7,253	29,540	59
DC Motor	13,102	6,585	27

speed follows the measured one closely. The difference between the current amplitude and speed fluctuation especially in the lower speed region or transient period is mainly due to the backlash between the induction machine shaft and the DC machine, which magnifies the effect of torque pulsations on the experimentally measured current and speed. Another reason for the discrepancies is due to the lumped *qd* method not being able to model all distributed and spatial effects inside the actual machine. Still, the *qd* model remains the commonly used approach for modeling of electrical machine in the industry and its accuracy has been verified by Matlab/Simulink in Figs. 3.8 and 3.9.

3.5.3 Hardware Resource Utilization

The electrical machine hardware designs were targeted to Xilinx Virtex-7 XC7VX485T FPGA. This FPGA is mainly composed by configurable logic blocks, which contain a pair of logic slices that are configured by 6-input LUTs and storage elements, configurable input/output blocks, and programmable interconnections. It has the following features: 1955k logic cells, 68 Mb block RAM, 2800 DSP48E1, and 1200 I/O pins [61].

Table. 3.2 and Table. 3.3 show the FPGA hardware resource utilization of various types of machines. It can be observed that although fixed-point operators are thrifty, massively parallel fixed-point implementation of machine models consumes more hardware

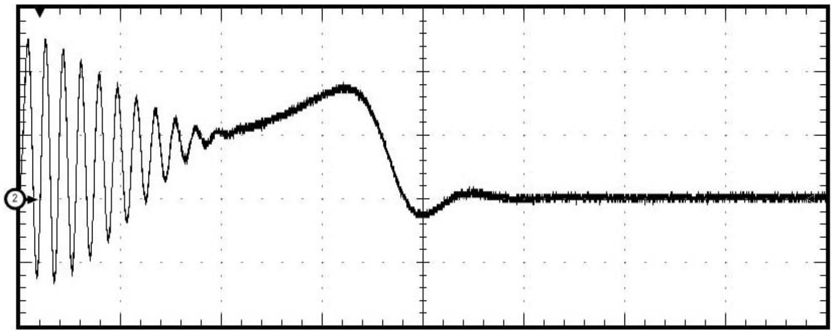


Figure 3.12: FPGA-based real-time trace of torque response of induction motor. [Time: 100ms/div, T_e: 12.8(N.m)/div].

resources in terms of registers and LUTs compared with deeply pipelined floating-point realization containing extravagant floating-point calculations.

3.6 Real-Time Emulation Case Studies

This section presents the real-time performances from the developed hardware machine models. The results were captured on a 500-MHz 4-channel oscilloscope that was connected to the DAC card on the Xilinx Virtex-7 VC707 FPGA development board. The results show the details of the electrical machine transients under normal and disturbed conditions. To demonstrate the usefulness of proposed hardware designs for real-time emulated electrical machines in HIL environment, the models of induction motor, synchronous generator with field windings, line start-permanent magnet synchronous motor (PMSM), and shunt-connected DC motor are tested and evaluated. The parameters and specifications of the machines are listed in Appendix A.

3.6.1 Case I: Induction Motor Transients

The dynamic performance of the induction motor is shown in Fig. 3.11 and also the associated real-time electromagnetic torque waveform is depicted in Fig. 3.12. The oscilloscope traces of real-time emulator corresponding to induction motor transients during free acceleration from stall are captured in this case. Moreover, the corresponding off-line and experimental results are presented and discussed in Section 3.5. Good agreement between off-line, real-time, and experimental results confirms the effectiveness of proposed approach for real-time emulation of the induction machine.

3.6.2 Case II: Synchronous Generator Transients

This case focuses on the real-time emulation of dynamic performance of the synchronous generator during a 3-phase fault at the machine terminals. The stability of synchronous

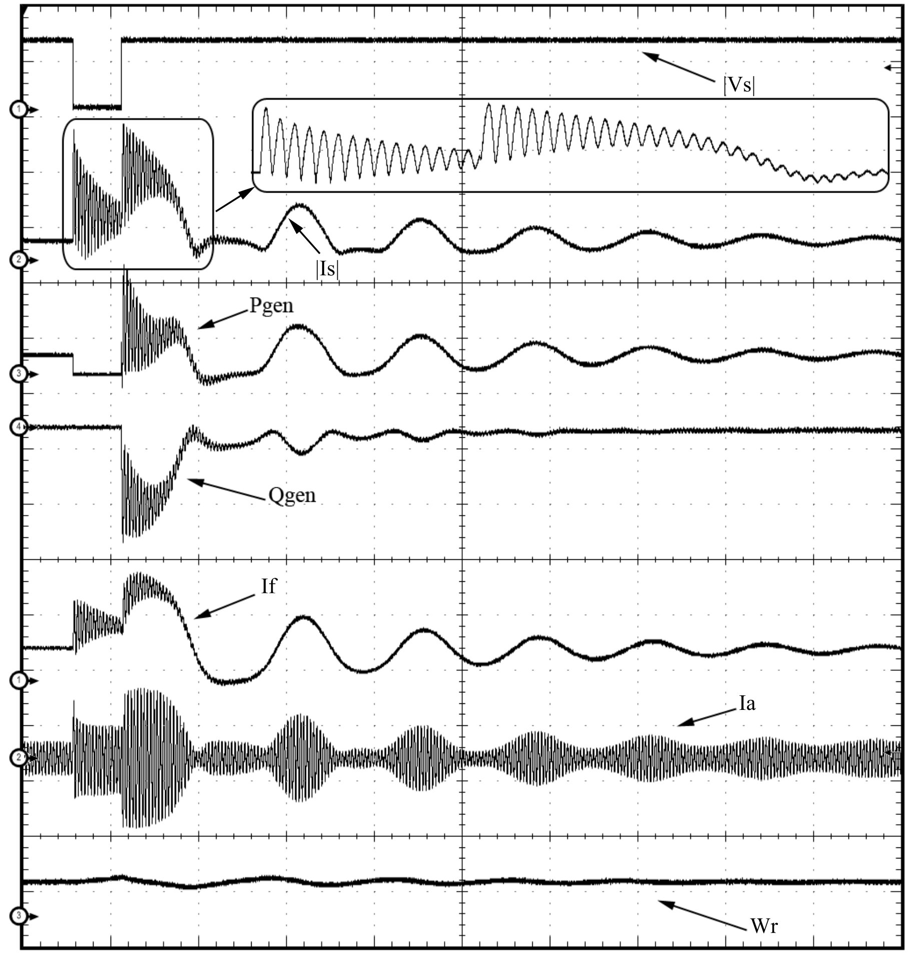


Figure 3.13: Real-time traces of synchronous generator dynamic behavior under normal and faulted condition. [Time: 450ms/div, v_s (stator voltage phasor): 11.3kV/div, i_s (stator current phasor): 104.4kA/div, P_{gen}: 2660MW/div, Q_{gen}: 2660MVar/div, I_f: 98.5kA/div, I_a: 166.5kA/div, ω_r: 590rps/div].

machines in a power system following a fault is of importance to determine line loading limits. The real-time oscilloscope traces shown in Figs. 3.13 and 3.14(a) illustrate the dynamic behavior of the synchronous generator during and following a 3-phase fault. The machine is initially connected to an infinite bus delivering rated apparent power at nominal power factor. The input torque and field voltage are held constant. With the machine operating in steady-state, a 3-phase fault occurs at the machine terminals at $t = 0.25$s. During the fault, the terminal voltage is zero, the machine is unable to transmit power to system. Hence, all of the input torque, with the exception of the ohmic losses, accelerates the rotor. The fault is cleared at $t = 0.5$s and machine returns to its original operating condition after experiencing a transient condition. Plus, Fig. 3.14(b) shows the impact of this disturbance on torque-rotor angle characteristics by a real-time oscilloscope Lissajous curve.

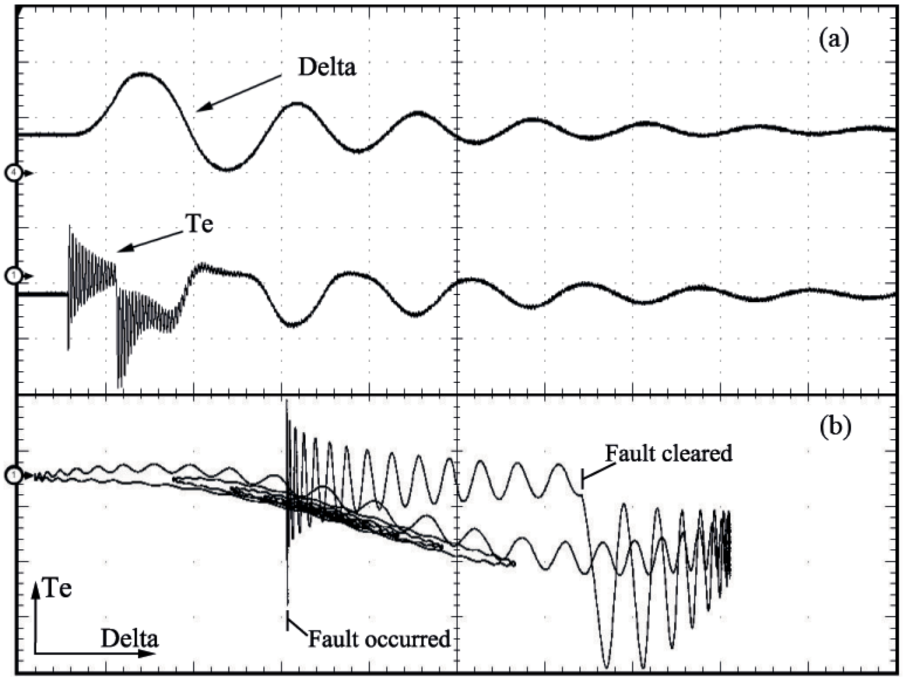


Figure 3.14: (a) Real-time traces of torque and rotor angle of synchronous generator vs. time. [Time: 450ms/div, δ: 1.4rad/div, T_e: 12.8M(N.m)/div]. (b) Real-time X-Y mode trace of torque vs. rotor angle of synchronous generator. [T_e: 7.9M(N.m)/div, δ: 0.33rad/div].

3.6.3 Case III: Line Start-Permanent Magnet Synchronous Motor Transients

Line start-permanent magnet synchronous motor (LSPMSM) is a high efficient alternative to replace induction motors in the constant speed operations with load variations. The dynamic performance of real-time emulated LSPMSM is depicted in Fig. 3.15 by oscilloscope traces for applied three-phase supply voltages. The motor accelerates from stall. Once the steady-state operation is established, the load torque is suddenly stepped to 80% of nominal torque at $t = 1s$. Motor increases load angle (δ) to maintain steady-state operation and then three phase fault is occurred at the motor terminal after $t = 1.5s$ whereupon the motor loses its synchronism and reestablishes the steady-state condition when the fault is cleared at $t = 1.65s$. Induction component (T_{ind}) of electromagnetic torque (T_e) significantly contributes to accelerate the rotor during starting and faulty condition while excitation component (T_{exc}) plays a considerable role in the steady-state operation where reluctance torque (T_{rel}) has a small effect in this case.

3.6.4 Case IV: DC Motor Transients

The results from real-time emulator of DC machine are assessed in this case study. A shunt-connected DC motor is selected for evaluation due to its desirable features and characteristics in the industrial and adjustable speed drive applications. The starting transients of the

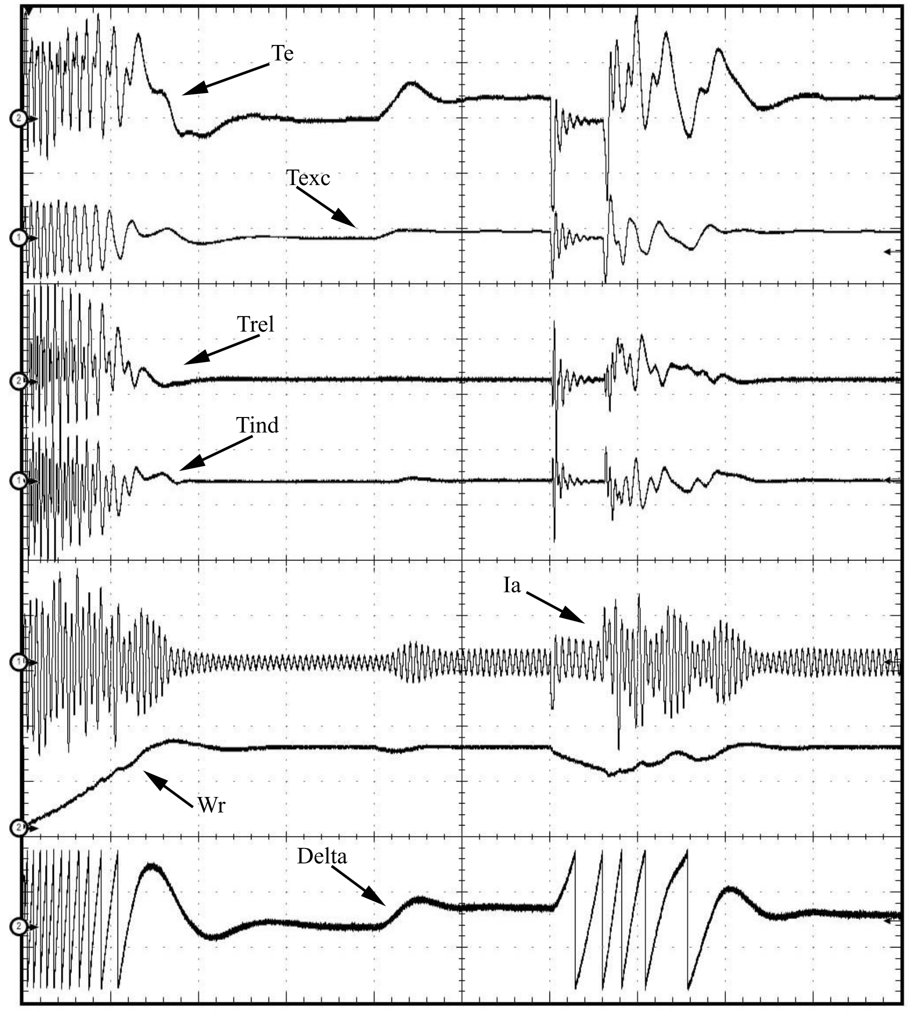


Figure 3.15: Real-time traces of LSPMSM performance. [Time: 250ms/div, T_e: 18.2(N.m)/div, T_{exc}: 39.5(N.m)/div, T_{rel}: 91.7(N.m)/div, I_a: 46.1A/div, ω_r: 269.3rps/div, δ: 2.6rad/div].

motor are represented in Fig. 3.16, when it is started with a resistance starter switched at fixed armature current level to keep it within the safe limit. The armature and field winding current (I_a and I_f), internal emf (E_a), rotor speed (ω_r) and torque (T_e) are captured in the results. During no-load starting, the switching of the four resistor segments is being triggered by the crossing of I_a below the threshold value. After a transient condition, steady-state operation is achieved at $t = 2$s approximately. Then, the rated load torque is suddenly applied at $t = 2.7$s. There is an increase in the armature current and a decrease in speed for this increase in load torque.

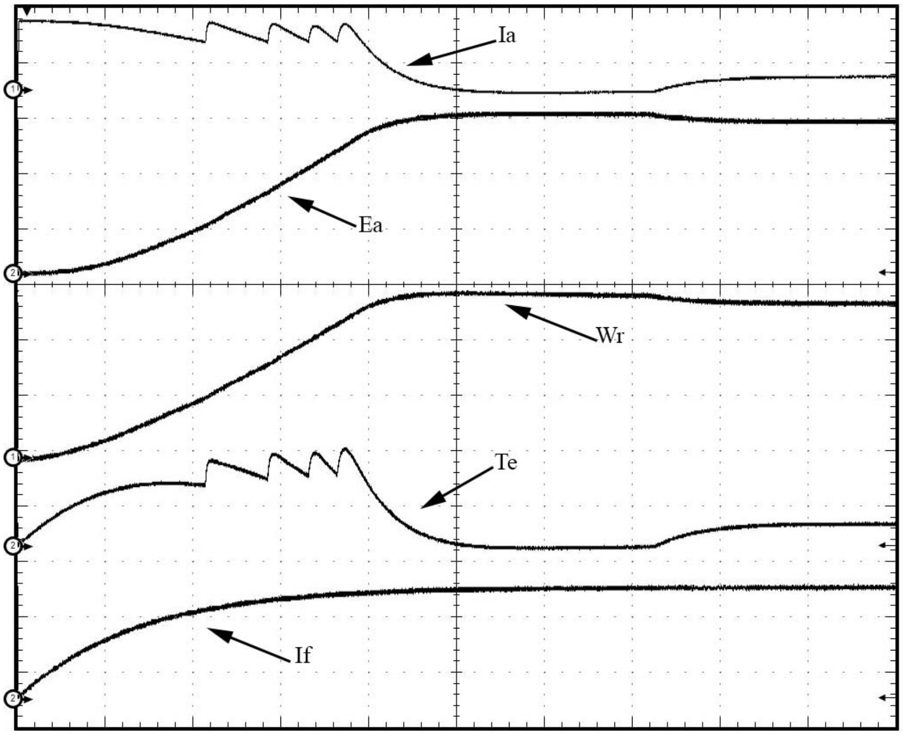


Figure 3.16: Real-time trace of DC motor behavior. [Time: 375ms/div, I_a: 62.5A/div, E_a: 86.1V/div, ω_r: 45.5rps/div, T_e: 72(N.m)/div, I_f: 0.5A/div]

3.7 Summary

In this chapter, a unified framework for FPGA based digital hardware emulation of electrical machines by different approaches was presented and a comprehensive comparison was provided. Hardware designs were developed for induction motor, synchronous generator, line start-permanent magnet synchronous motor, and DC machine. Although the overall procedure of the realization of machine models was sequential, all parallel pathes have been taken into account in order to fast computational processing of real-time simulation. The close agreement between the real-time emulated performances, off-line simulation, and experimental measurements confirmed the effectiveness of the proposed approaches. Such a real-time emulation of electrical machine can be used in HIL simulations to test new controller and drive systems against a virtual model of the machine. Furthermore, the general framework for digital hardware implementation of state-space model presented in this work not only can be applied to machine equations but also can be used for real-time emulation of any system such as power convertors, controls, mechatronic systems, *etc* that can be described in terms of state-space equations. Moreover, in this study the results demonstrated that although floating-point arithmetic utilizes more hardware resources compared with fixed-point calculations, deeply pipelined implementation by floating-point number representation can consume less hardware resources than massively parallel realization by fixed-point operations. In addition, the presented results showed in spite of the fact that floating-point calculation leads to more accurate results in off-line simulation compared with fixed-point, in real-time simulation floating-point computation may not guarantee such a basis. Future work would include modeling of the spatial and nonlinear phenomena inside the machine structure to emulate the machine performances with the higher accuracy.

4
Real-Time Nonlinear Magnetic Equivalent Circuit Model of Induction Machine

4.1 Introduction

This chapter [1] focuses on FPGA-based real-time emulation of the induction machine by means of magnetic equivalent circuit (MEC). The objectives of this work are twofold: (1) A nonlinear MEC suitable for real-time studies is presented for induction machine, and (2) FPGA is chosen as a computational engine to solve the nonlinear equations of MEC in real-time.

As HIL technology is increasingly becoming the preferred, reliable, cost- and time-effective alternative in virtual scenarios for testing real devices in many industrial applications [55, 62–65], a geometry-based real-time emulated machine model in HIL configuration can play an important role in the test procedure allowing engineers to evaluate the behavior of newly designed controllers, drive systems, and protection devices under a wide range of situations and extreme conditions against a real-time emulated machine model in a non-destructive HIL environment. It will also help in the design procedure letting engineers optimize the machine performance by running the emulated machine model in real-time using different geometrical and material properties as many times as they need to reach the desired objectives, thus enabling statistical and sensitivity analyses.

To reproduce the realistic behavior of the machine for HIL simulation, a detailed and accurate modeling technique is required. Electric-equivalent lumped-parameter method such as *qd* machine model [66] and numerical approaches such as finite-element method

[1] Material from this chapter has been submitted: N. R. Tavana, and V. Dinavahi, "Real-Time Nonlinear Magnetic Equivalent Circuit Model of Induction Machine on FPGA for Hardware-in-the-Loop Simulation", *IEEE Trans. on Energy Conversion*, pp. 1-9, submitted on February, revised on May, and revised on August, 2015.

(FEM) [14] are two options for this purpose. However, the qd-type machine models suffer from limited accuracy due to their inability of including the spatial and local nonlinear effects as well as distributed phenomena inside the machine [67–69]. On the other hand, although FEM offers excellent accuracy and correctness, it is computationally intensive, time-consuming even for moderately complex systems, and cannot be performed within the simulation time-step even with a fast computational engine to satisfy the essential and strict execution time-constraint of real-time simulation [70]. Magnetic equivalent circuit is another candidate for the modeling of electromagnetic devices that can provide acceptably accurate solutions with reasonable computational effort [71–80]. Therefore, MEC can be taken into consideration as a compromise between numerical techniques and electric-equivalent approaches, and selected as the best realizable method for real-time emulation. Owing to dramatic developments in VLSI technologies and shrinking transistor sizes, the FPGA is becoming an advanced computational engine for digital hardware realization of complex industrial system models by providing nanosecond computational clock cycle within the simulation time-step in real-time [4,47,81].

This chapter explains the necessary steps to develop a nonlinear MEC for induction machine. The complete details of the FPGA-based real-time hardware realization of the associated nonlinear equations are then presented. The experimental measurements, and finite-element results are finally used to confirm the correctness of proposed approach for real-time emulation of induction machine.

4.2 Nonlinear MEC Model of Induction Machine

An MEC model is presented for rotary induction machine in this section. It is intended to retain much of the salient features of extensive and complex models [73–79] and yet meet the computation time constraint of real-time simulation.

4.2.1 Magnetic Circuit

Similar to meshing in FEM, the MEC model requires a permeance element network. A carefully selected permeance network not only improves the accuracy of final results but also minimizes the complexity and accelerates the necessary computations of corresponding nonlinear equations which must be solved within the simulation time-step in real-time.

According to a general 2-D structure of rotor and stator sectors, the permeance elements are created by a polar grid with several angles (φ_0 to φ_5) and radiuses (R_0 to R_8) outlined by the blue lines in Fig. 4.1 in such a way that the flux tubes are coincident with the main flux paths. Generally, the generated permeance network is similar to the one proposed in [76] in terms of number of nodes and structure. The reason why the minimum number of nodes (two nodes per tooth) is considered for permeance network is that any extra node leads to an extra equation in the system model, which dramatically increases

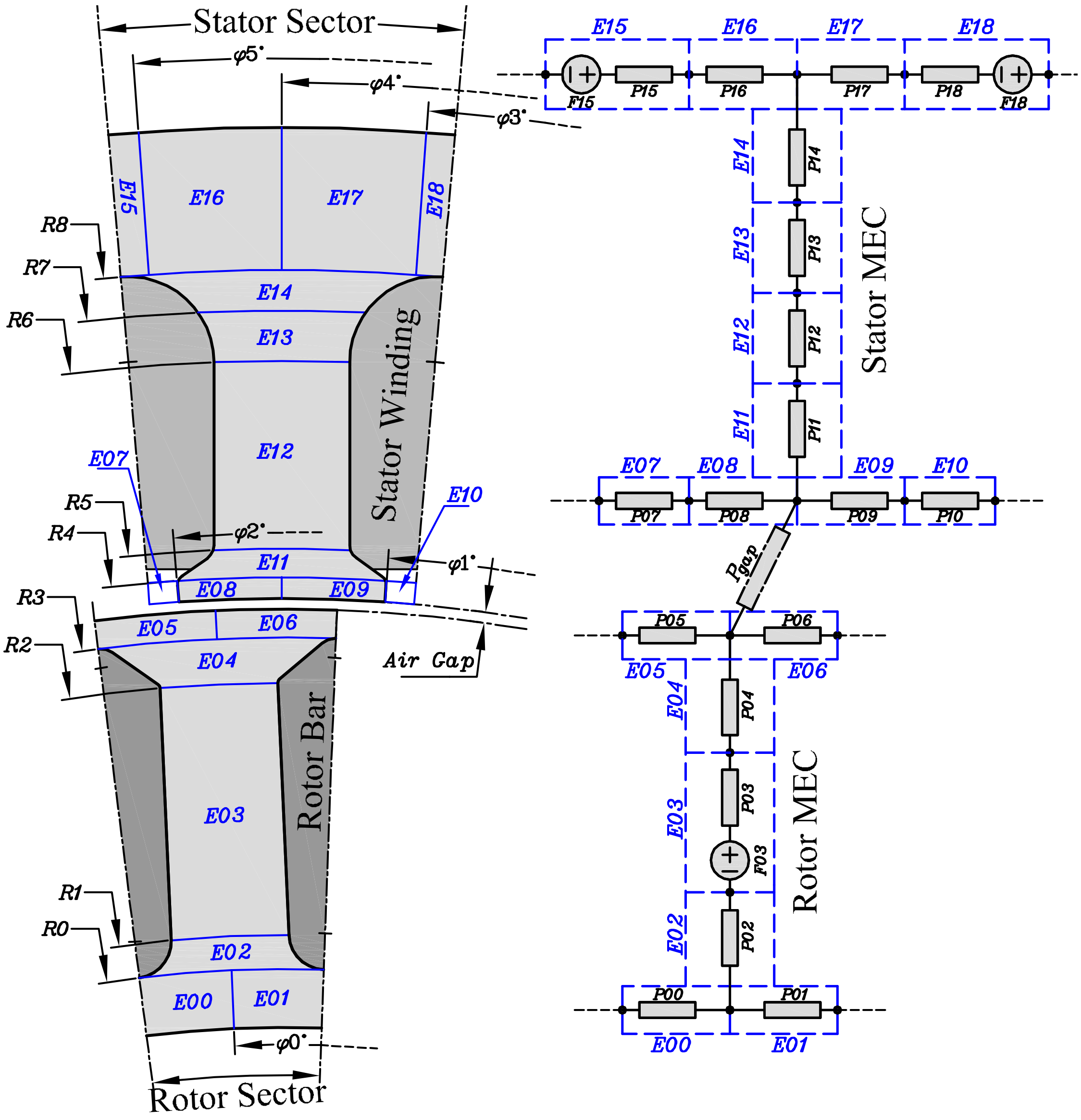


Figure 4.1: One sector of MEC of induction machine.

the computational time. However, it enjoys finer permeance element sizes to define flux tubes in the MEC model more precisely, while the parallel hardware architecture of FPGA allows computing a large number of permeance elements simultaneously. In comparison with MEC in [79], in the proposed MEC the corresponding permeances for polar direction of flux paths in stator and rotor tooth (E_{02} to E_{04}, and E_{11} to E_{14}) are neglected, and additionally permeance elements for the radial direction of flux paths in the stator and rotor slot closure (E_{07} to E_{10}, and E_{05} to E_{06}), stator yoke (E_{15} to E_{18}), and rotor central elements (E_{00} and E_{01}) are eliminated to reduce the real-time computational efforts significantly with only a slight reduction in the results accuracy due mainly to the fact that

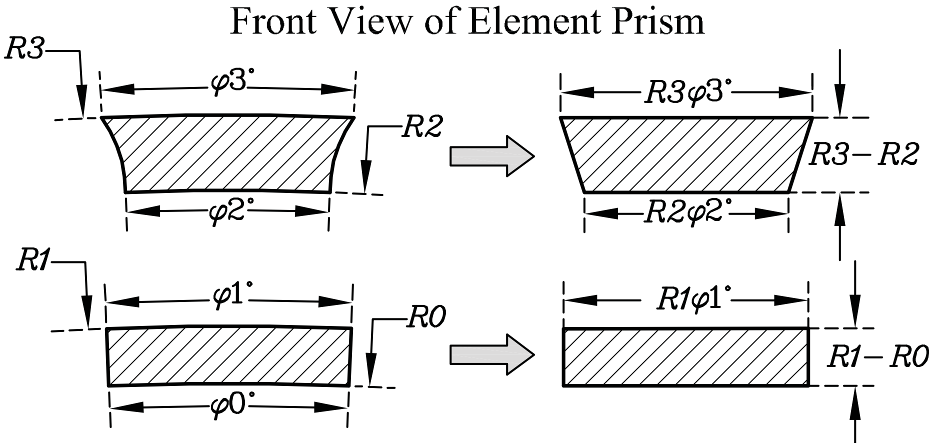


Figure 4.2: Simplification of magnetic flux prism.

the main flux path does not flow anywhere except in the direction of the path of magnetic materials.

In the proposed MEC, the elements of E_{02}, E_{04}, E_{11}, and E_{13} to E_{18} are simplified by trapezoidal flux prisms and the rest by rectangular prisms. The corresponding permeances, as shown in Fig. 4.2, can be approximated as follows [74]:

$$P = \begin{cases} \mu L \left(\frac{R_3\varphi_3 - R_2\varphi_2}{R_3 - R_2}\right)\left(ln\left(\frac{R_3\varphi_3}{R_2\varphi_2}\right)\right)^{-1}, \\ \quad \text{Element Number} : E_{02}, E_{04}, E_{11}, E_{13}, E_{14}. \\ \mu L \left(\frac{R_3 - R_2}{R_3\varphi_3 - R_2\varphi_2}\right) ln\left(\frac{R_3\varphi_3}{R_2\varphi_2}\right), \\ \quad \text{Element Number} : E_{15}, E_{16}, E_{17}, E_{18}. \\ \mu L \left(\frac{(R_1+R_0)(\varphi_1+\varphi_0)}{4(R_1-R_0)}\right), \text{ Element Number : Others.} \end{cases} \tag{4.1}$$

where μ and L are magnetic permeability and machine length, respectively.

The air-gap permeances are calculated based on the area of overlap between the stator and rotor teeth. The air-gap permeance between ith rotor tooth and jth stator tooth with instant polar angle difference of φ can be obtained by

$$P_{gap}(i,j) = \begin{cases} P_{max} = \frac{\mu_0 A_{st} L}{g}, \\ \quad \text{if } |\varphi| < \varphi_{ovr}; \text{ Region I,} \\ \frac{\varphi_{linr} - \varphi}{\varphi_{linr} - \varphi_{ovr}} P_{max}, \\ \quad \text{if } \varphi_{ovr} \leq |\varphi| \leq \varphi_{linr}; \text{ Region II,} \\ \quad 0, \qquad O.W. \end{cases} \tag{4.2}$$

where Region I defines the area of overlap and Region II is where the overlapped area is changing linearly when the motor is rotating. Additionally, g and A_{st} denote mechanical air-gap length and stator tooth face width, respectively.

Moreover, the stator and rotor MMF sources are determined by Ampere-turn formulae corresponding to the associated winding turns in the stator and rotor slots.

Once the permeances and MMF sources are calculated, nonlinear equations of MEC can be formulated based on standard circuit nodal analysis given by [75]:

$$\mathbf{P}\left(\mathbf{M},\mathbf{i}\right)\cdot\left[\mathbf{M},\mathbf{i}\right]^{T}=\mathbf{0}. \tag{4.3}$$

where $\mathbf{P}\left(\mathbf{M},\mathbf{i}\right)$ represents the permeance matrix which is a function of magnetic scalar potentials of nodes ($\mathbf{M}$) as well as stator and rotor loop currents ($\mathbf{i}$).

4.2.2 Interfacing of Magnetic and Electric Circuits

MMF sources are the elements by which magnetic and electric circuit are connected to each other. The magnetic flux generated by stator winding (electric circuit) traveling through the MEC can be expressed by

$$\left(\mathbf{W}\cdot\times\mathbf{P}_{st}^{T}\right)\cdot\left(\mathbf{M}_{st}-\mathbf{M}_{sy}\right)=\boldsymbol{\lambda}_{s}. \tag{4.4}$$

wherein $\mathbf{W}$ is the winding function for the three-phase stator system [66], and $\cdot\times$ represents element-wise multiplication of two vectors. Eq. (4.4) illustrates that adding up the number of turns times the magnetic flux entering the stator teeth with the permeance of $\mathbf{P}_{st}$ is almost equal to the stator flux linkages ($\boldsymbol{\lambda}_{s}$). In this equation, $(\mathbf{M}_{st}-\mathbf{M}_{sy})$ defines the difference of magnetic scalar potentials across the permeance of the stator tooth ($\mathbf{P}_{st}$). Additionally, $\mathbf{M}_{st}$ and $\mathbf{M}_{sy}$ denote magnetic scalar potentials of stator tooth and stator yoke, respectively.

It is worth mentioning that in Eq. (4.4) the small amount of stator leakage flux crossing inside stator slot and also fringing flux for outside and inside faces of stator core are neglected due mainly to the real-time computation while significant amount of this flux crossing from stator tooth tip to the adjacent tooth tip, and the flux traveling from stator to rotor and again back to stator without inducing rotor current are taken into account in the MEC.

Furthermore, the rotor flux linking the end ring (λ_e) and nth current loop (λ_r^n) can be given by:

$$\lambda_r^n=-P_b^{n-1}i_b^{n-1}+P_b^n i_b^n+2P_e^n i_r^n-P_e^n i_e+\phi_r^n. \qquad n=1,2,...N_r, \tag{4.5}$$

$$\lambda_e=\sum_{i=1}^{N_r}P_e^n\left(i_e-i_r^n\right). \tag{4.6}$$

and

$$i_b^n=i_r^n-i_r^{n+1}. \qquad n=1,2,...N_r, \tag{4.7}$$

where P_b^n is the permeance of the magnetic flux path inside nth rotor bar, and P_e^n is the permeance of nth segment of end ring of rotor cage, which can be found in [76]. Also, the current of nth rotor loop, nth rotor bar, and end ring are denoted by i_r^n, i_b^n, and i_e, respectively.

In Eq. (4.5), ϕ_r^n represents the magnetic flux flowing through nth rotor tooth in the rotor structure with N_r slots.

4.2.3 Electric Circuit

The three-phase voltage equations of the stator electric circuit of induction machine can be defined by:

$$\frac{d}{dt}\boldsymbol{\lambda}_s = \mathbf{v}_s - \mathbf{r}_s\mathbf{i}_s. \tag{4.8}$$

where $\mathbf{v}_s$, $\mathbf{r}_s$, and $\mathbf{i}_s$ represent applied voltages, resistances, and currents of stator windings, respectively.

Furthermore, the electrical equations for the rotor side are as follows [73]:

$$\begin{aligned} \frac{d}{dt}\lambda_r^n &= r_b^{n-1}i_b^{n-1} - r_b^n i_b^n + 2r_e^n i_r^n + r_e^n i_e. \\ &\qquad\qquad\qquad\qquad n = 1, 2, ...N_r, \end{aligned} \tag{4.9}$$

$$\frac{d}{dt}\lambda_e = -\sum_{i=1}^{N_r} r_e^n \left(i_e - i_r^n\right). \tag{4.10}$$

where r_b^n denotes the resistance of nth rotor bar, and r_e^n is the resistance of nth segment of end ring.

4.2.4 Nonlinear Solution of Detailed MEC

In the governing equations of the developed MEC, the permeance elements are defined as a nonlinear function of magnetic scalar potentials due to iron saturation effect. Thus, the combination of Eqs. (4.3)-(4.10) results in a system of nonlinear algebraic equations $\mathbf{f}\left(\mathbf{x}\right)$ with a large dimension rewritten as

$$\mathbf{f}\left(\mathbf{x}\right) = \mathbf{A}\left(\mathbf{x}\right)\mathbf{x} - \boldsymbol{\lambda} = \mathbf{0}. \tag{4.11}$$

where $\mathbf{x}$ and $\boldsymbol{\lambda}$ are the unknown and known vectors in each simulation time-step, respectively.

The time-marching transient simulation of MEC of induction machine is performed for a 230 V, three-phase, 3-hp squirrel cage Baldor induction machine, whose geometry and specifications are presented in Appendix B.

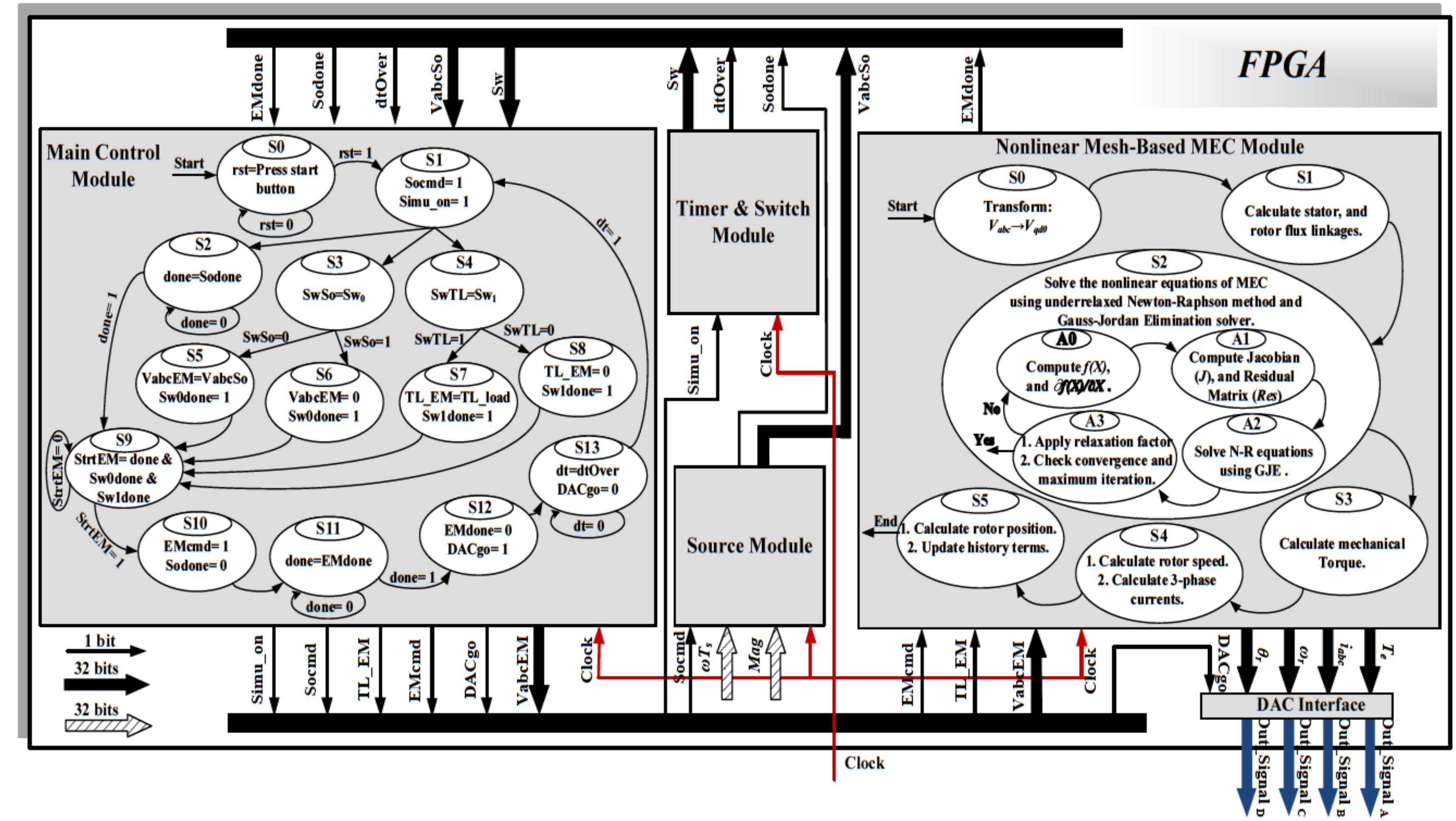


Figure 4.3: Finite-state machine for paralleled and deeply pipelined real-time algorithm for FPGA implementation of the nonlinear MEC model of induction machine.

The motor has 36 stator slots and 28 rotor bars. The system of nonlinear algebraic equations for the presented MEC has a total of 160 unknowns in this case which is excessively time-consuming for real-time simulation.

To cope with this problem, an anti-periodicity and moving band technique, which is widely used in FEM, is applied to the MEC model to reduce the domain of study to one pole-pitch (a quarter) and increase the computational speed. Since the geometry of a squirrel-cage IM is normally composed of a repetitive section of domain and the MEC model is distributed across the geometry of the machine, the method allows to reduce the number of stator slots from 36 to 9 and the number of rotor slots from 28 to 7. Thus, the total number of equations in the augmented magnetic system decreases from 160 to 43.

The Forward Euler method is chosen as a numerical integration technique to discretize the differential equations of electric circuits in each simulation time-step and Newton-Raphson (N-R) method is employed to solve the system of nonlinear equations. In order to improve convergence, the nonlinear iteration is decelerated by applying an under-relaxation factor (α), as follows [82]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \mathbf{J}_k^{-1} \mathbf{f}(\mathbf{x}_k). \tag{4.12}$$

where $\mathbf{J}$ is the Jacobian matrix. Additionally, $\mathbf{x}_{k+1}$ and $\mathbf{x}_k$ are the solutions of nonlinear procedure at $(k+1)$th and (k)th iteration, respectively.

In this work, a constant relaxation factor of 0.32 is considered for all iterations and the maximum number of N-R iterations is limited to 15. Consequently, hardware realization of proposed approach for an MEC model of induction machine on FPGA can be executed within a time-step of 400μs for real-time emulation according to the best case achievable for FPGA clock frequency obtained at the end of place and route process of hardware design architecture.

4.3 Real-Time Hardware Emulation of Nonlinear MEC on FPGA

In this work, a 32-bit single-precision floating-point format (*IEEE* standard 754) is employed for digital hardware implementation. As shown in Fig. 4.3, finite-state machine for real-time emulation of MEC for the induction machine consists of four main modules: `Main Control Module`, `Source Module`, `Timer & Switch Module`, and `MEC Module`.

The `Main Control Module` coordinates the operation of the whole emulator to carry out the algorithm by sending control signals (`Simu_on`, `Socmd`, `EMcmd`) to each module to perform the required functions and receiving the acknowledged signals (`Sodone`, `EMdone`, `dtOver`, `Sw`) to judge if the functions are done.

The three-phase voltage source is created by the `Source Module`. Furthermore, a hardware module of `Timer & Switch` is designed to simulate switch status, and generate `dtOver` signal to check whether or not the simulation is executed within the real-time step-size [83].

The finite-state machine (FSM) diagram for the hardware realization of MEC model is depicted in Fig. 4.3. The procedure starts with the transformation of voltage sources from 3-phase domain to stationary reference frame by the help of floating-point matrix multiplication. The required flux linkages are then computed by dedicated floating-point arithmetic units and Forward Euler integration method. The under-relaxed N-R is responsible to compute unknown variables of the MEC model and mechanical performance of the machine are then predicted in the next sates. The realization of N-R is the core of hardware design which is organized by using the three main computational submodules:

1. Parallel Gauss-Jordan Elimination (GJE) module;
2. Residual vector calculator;
3. Nonlinear function evaluator of $\mu(H)$ and $d\mu(H)/dH$;

The global control schedules all sequential and parallel operations in the MEC model. The sequential operations can be seen in Fig. 4.3 by state transitions, and all possible parallel calculations, which take the advantages of hardware parallelism in the FPGA, are taken into account for the realization. For example, the elements of $f(M,i)$, $\partial f(M,i)/\partial M$, and $\partial f(M,i)/\partial i$ are calculated in parallel in state S_0, and the evaluation of Jacobian and residual vector are carried out concurrently in state S_1. The detailed architecture of main submodules inside the MEC Module are described forthwith.

1) *Parallel Gauss-Jordan Elimination Unit:*

Parallel GJE is employed in this design which is faster than the Gauss-Seidel iterative method and the LU decomposition. In addition, it is easier and simpler for hardware implementation [81]. As can be seen from Fig. 4.4 (a), the hardware architecture consists of 43 elimination units (Eli1, Eli2, ..., Eli43) and one factorization unit. First, the Jacobian matrix ($\mathbf{J}$) and residual vector ($\mathbf{Res}$) are combined together to make a 43×44 matrix whose rows are stored in RAMs and then GJE starts computing unknown vector. In the factorization module, the *i*th row is retrieved from the corresponding elimination unit, and the diagonal element is registered in the `Factorization unit` and the remaining elements within the row are divided by the registered diagonal element and returned to the elimination units. The elimination is carried out concurrently in all elimination units. For the *i*th row, no elimination is performed and the factorized row is saved in RAM for the next call. However, for the *j*th row ($j \neq i$), the *i*th element is identified and registered as Eli_j. The elimination is then processed by using the elements of *i*th factorized row, registered Eli_j, and the corresponding elements of *j*th row [81].

2) *Parallel Computational Unit for Residual Vector:*

As the direct multiplication of $\mathbf{A}$ matrix with a 43×43 dimensions to 43-element $\mathbf{x}$ vector takes long time in each iteration of N-R algorithm within the simulation time-step,

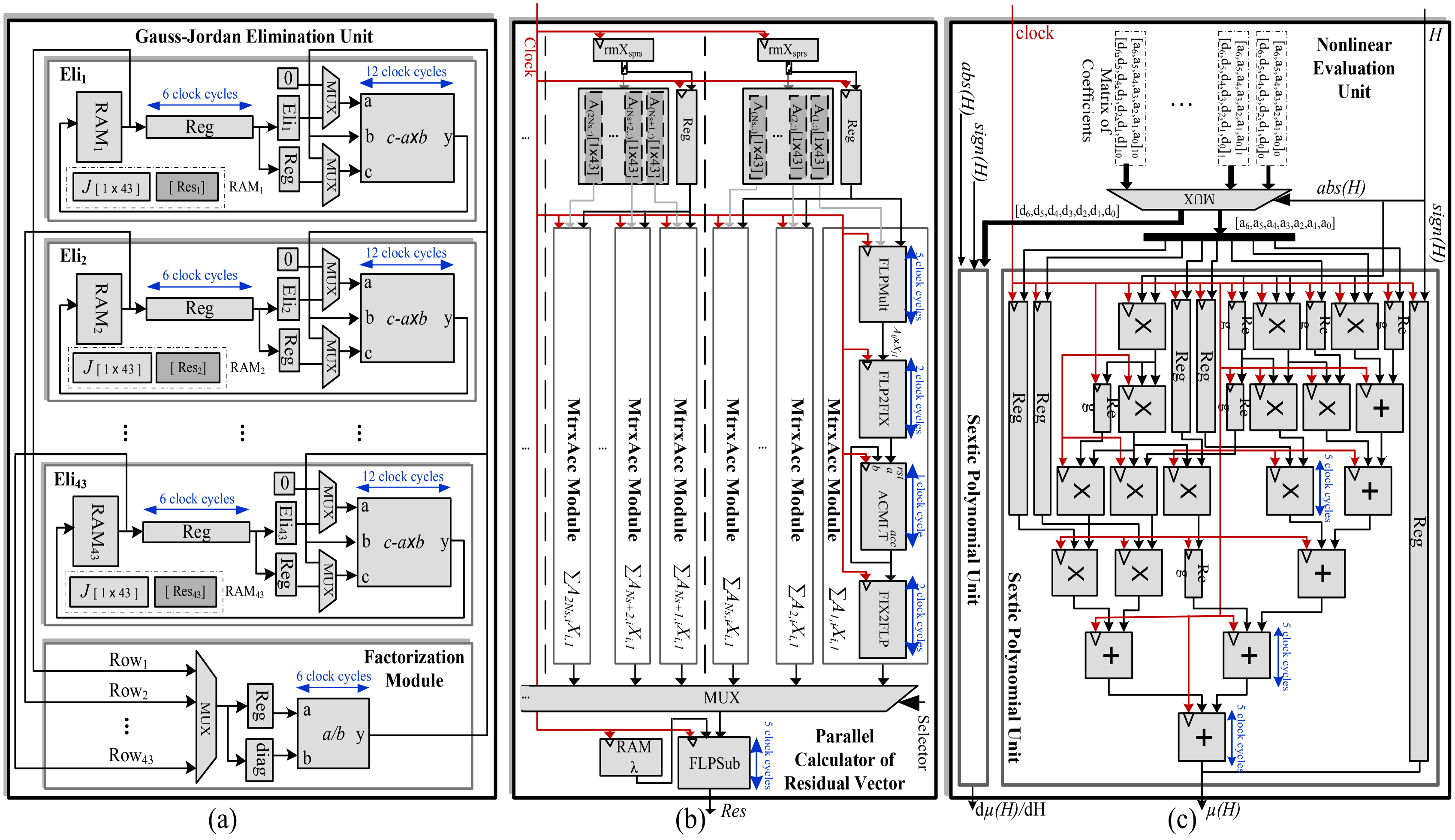


Figure 4.4: (a) Massively parallel Gauss-Jordan Elimination module [81]. (b) Parallel implementation of residual vector calculator. (c) Fully pipelined realization of nonlinear function evaluator.

the massively parallel hardware architecture of sparse matrix vector multiplications is designed to compute the residual vector. It can be observed from Fig. 4.4 (b) that a RAM (rmXsprs) is employed to store the contents of $\mathbf{x}_{43\times1}$ vector and the address of nonzero elements of each row of $\mathbf{A}_{43\times43}$ matrix. Such a hardware realization splits $\mathbf{A}_{43\times43}$ matrix to 43 individual rows, and computes sparse row-vector multiplications simultaneously, for instance $\sum\limits_{j=1}^{43} A_{ij}x_{j1}$ of ith row. The `MatrxAcc` submodule is in charge to provide the element-wise multiplication of two input vectors in a pipelined paradigm, convert the obtained results to fixed-point format (`FLP2FIX`), accumulate them in every single clock cycle (`ACMLT`), and again convert to floating-point representation (`FIX2FLP`). This is the fastest way to perform the matrix multiplications with large dimensions, although its digital realization consumes a large amount of hardware resources.

3) *Nonlinear Evaluation Unit:*

Due to saturation effect, the permeability of iron core (μ) is a nonlinear function of magnetic field intensity (H) presented in Appendix B. In this work, the eleven-segment sextic polynomial functions are used to closely predict the nonlinearity of $\mu(H)$ an $d\mu(H)/dH$ for the calculation of permeances and Jacobian matrix elements in each N-R iteration. As can be seen in Fig. 4.4 (c), the deeply pipelined structure of the unit is implemented by simple arithmetic operations such as adder, multiplier, multiplexer. Additionally, registers are inserted on the data paths to synchronize the data flow in the hardware architecture.

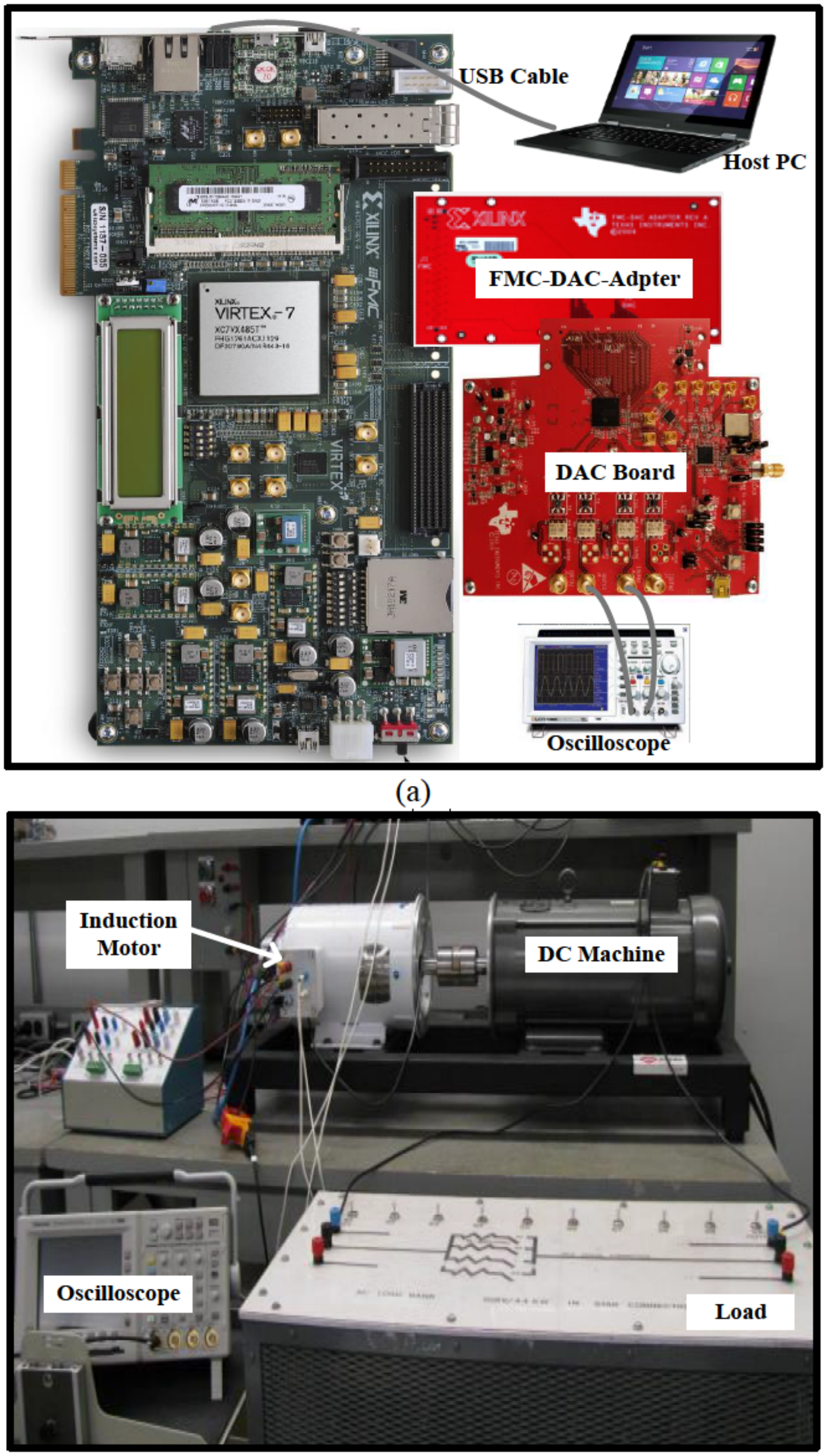
It is worth mentioning that the shape of nonlinear function defines the convergence rate and the number of iterations in the Newton-Raphson method in each time-step regardless of how the nonlinear functions are called for the computation of residual vectors or Jacobian matrix in the iterative solution procedure. In other words, more or less detailed representation of a nonlinear function does not impact the rate of convergence. However, the only deference between less and more detailed representation of a nonlinear function is the speed of nonlinear function call in each iteration, especially when a sequential processor (CPU) is used for computation.

Unlike the CPU, using the advantage of parallel processing on FPGA allows us to represent the nonlinear function with eleven-segment sextic polynomial function with the higher accuracy and fast computational time.

4.4 Investigation of FPGA-based Real-Time Emulation of Induction Machine

Hardware design of the induction machine was targeted to Xilinx® Virtex-7 XC7VX485T FPGA shown in Fig. 4.5(a).

Table. 4.1 shows the FPGA hardware resource utilization of the MEC model of induction machine. It should be noted that the real-time hardware emulation of machine model in this work is realized by 10ns FPGA clock frequency for the computational processing



(a)

(b)

Figure 4.5: (a) FPGA-based real-time emulator of induction machine, and (b) actual induction machine setup.

Table 4.1: FPGA resource utilization of real-time emulator.

Real-Time Emulator	Number of Slice Registers (607,200 available)	Number of Slice LUTs (303,600 available)	Number of DSPs (2,800 available)
Main Control	11	10	0
Switch Module	56	86	0
Source Module	767	697	7
MEC Module			
N-R Unit	123,372	215,772	1,346
Others	6,590	12,052	74
Total	130,796 (22%)	228,617 (76%)	1,427 (51%)

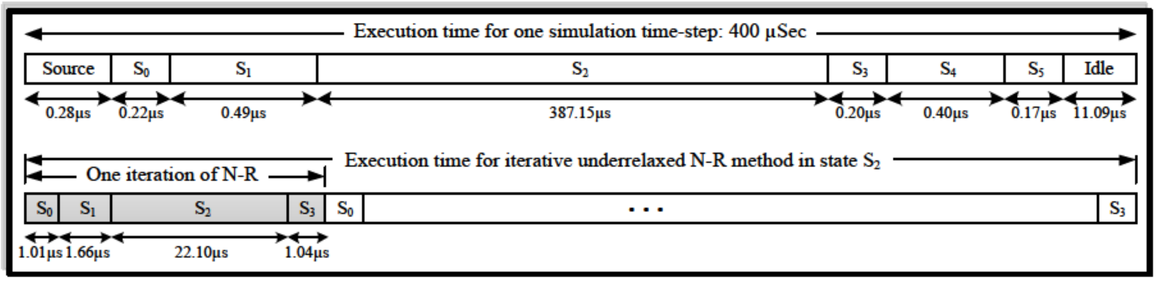


Figure 4.6: Detailed execution time for one simulation time-step and one iteration of under-relaxed N-R method.

within the simulation time-step of 400μs. The iterative under-relaxed N-R realization takes 387.15μs in which one iteration spends 25.81μs.

The detailed execution times according to the states of the nonlinear MEC model presented in Fig. 4.3 for one simulation time-step, and one iteration of under-relaxed N-R method for the computation of unknown variables within the simulation time-step are shown in Fig. 4.6. As the MEC model for real-time emulation of induction machine is obtained by some simplifications, it is necessary to evaluate the extent of accuracy of the real-time model. Thus, a squirrel cage Baldor® induction machine with closed rotor slots mechanically coupled with a DC machine is utilized as an experimental setup, as depicted in Fig. 4.5(b), to assess the performance of FPGA-based real-time emulation of induction machine. Moreover, in this work a 2-D nonlinear time-stepping transient finite-element method by JMAG® software is employed for validation as well. Fig. 4.7 shows the flux density distribution and flux lines inside the analyzed machine computed by FEM. In the FEM simulation, a relaxed N-R technique is used to obtain the nonlinear solution and the Incomplete Cholesky Conjugate Gradient method is chosen as the linear solver.

The transient performance of the proposed real-time method during a free acceleration from stall of the induction machine is investigated by using a no-load test where the machine is started directly from a 208 V three-phase supply. During the test, AC current clamp

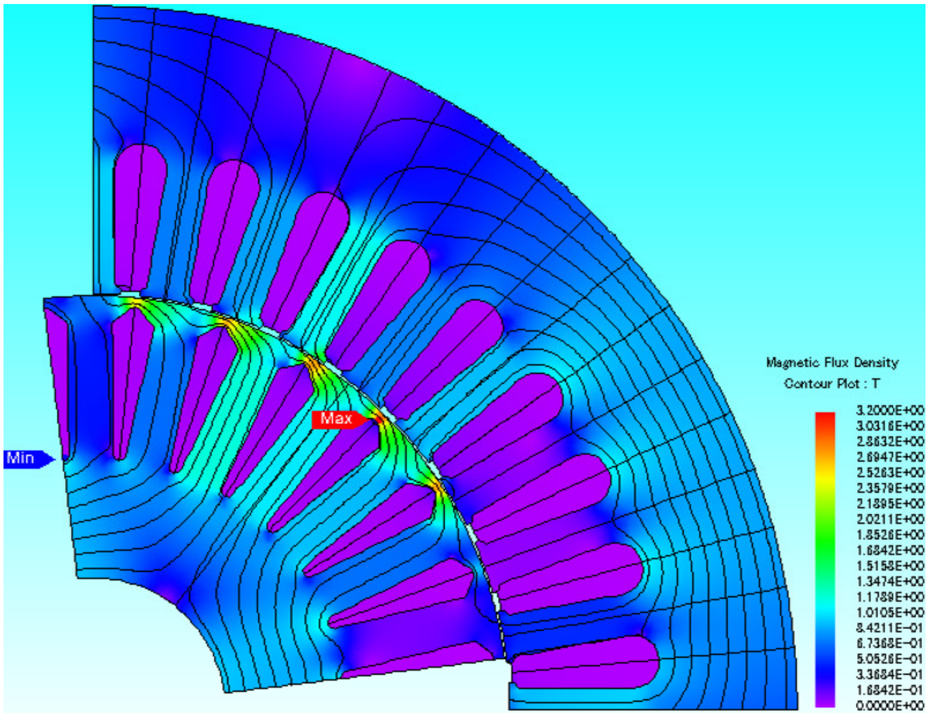
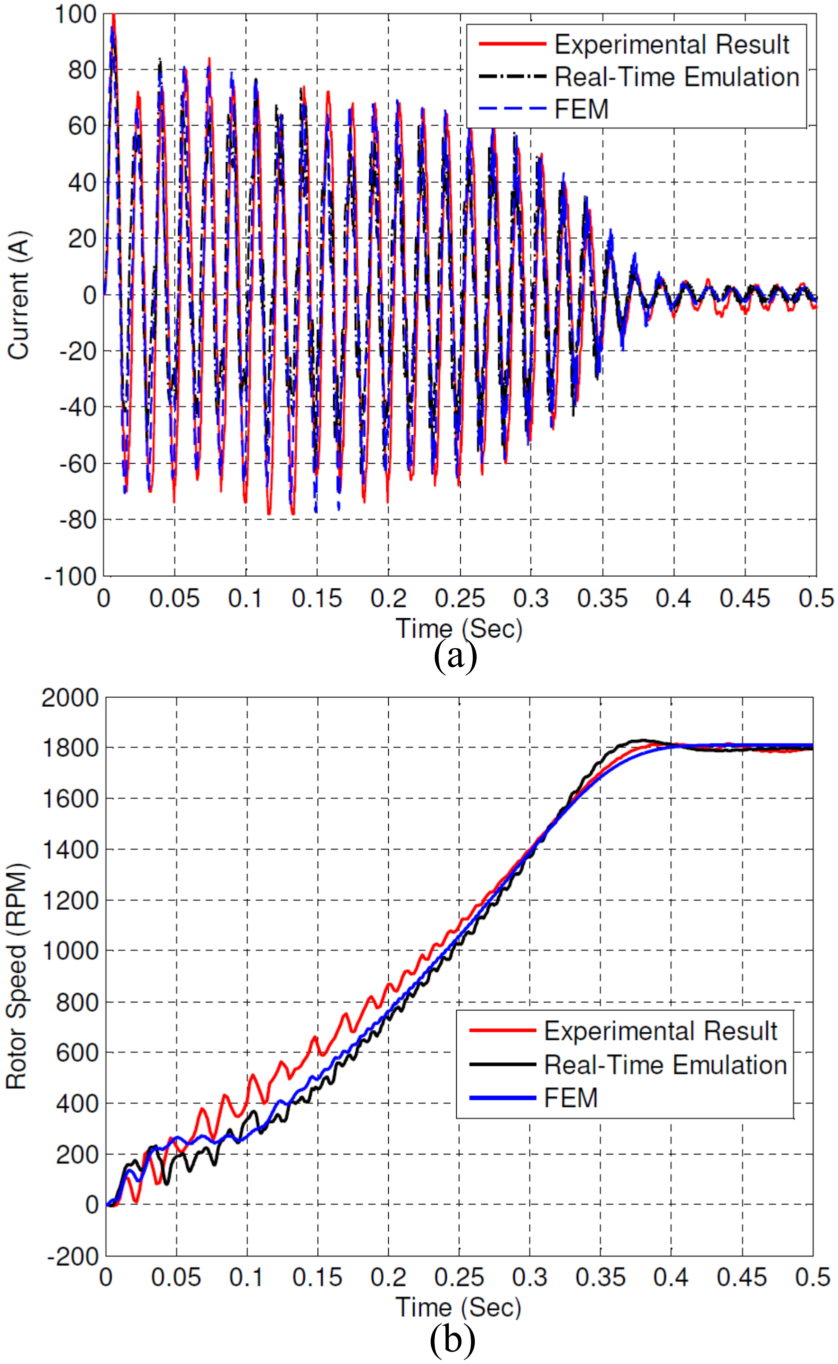


Figure 4.7: Magnetic flux density distribution and flux lines of the analyzed machine by *JMAG® Designer*.

is used to capture inrush current waveform and plotted in Fig. 4.8 (a). The FPGA-based real-time emulated stator current are exported via Xilinx ChipScope® Analyzer and laid over the experimental measurements. For further comparison, the FEM prediction of the current is also provided in Fig. 4.8 (a). Moreover, the rotor speed of the induction machine obtained by different approaches is presented in Fig. 4.8 (b). It is found that there is good agreement between the real-time emulated, finite-element calculated, and experimentally measured performance of the motor. All currents decay to steady-state over almost 0.4s and the speeds follow the similar trajectory closely. However, discrepancies are observed between the current amplitudes and speed fluctuations especially in the lower speed region or transient period due mainly to the backlash between the induction machine shaft and the DC machine which magnifies the effect of torque pulsations on the experimentally measured current and speed. Another reason is that although real-time induction machine model is based on a distributed element approach, the model is still lumped and not being able to include all distributed circuit effects perfectly for the prediction of transient behavior resulting in a limited accuracy in this region.

In the next assessment, the machine performance is investigated during steady-state operating points with different rotational speeds. Figs. 4.9 and 4.10 illustrate the real-time emulated, finite-element calculated, and measured torque and current values. It can be seen that real-time MEC predicts the stator current of steady-state performance with



(a)

(b)

Figure 4.8: Start-up transient of (a) stator current, and (b) rotor speed.

acceptable accuracy. The discrepancy of the results can be attributed in part to neglecting the core loss in the real-time MEC. Another reason could be the neglecting of skin effects in the rotor parameters for real-time computation. Furthermore, simplifying the elements of machine structure with trapezoidal and rectangular prisms, neglecting the skew effect in the rotor bars can be taken into account as other reasons for the differences.

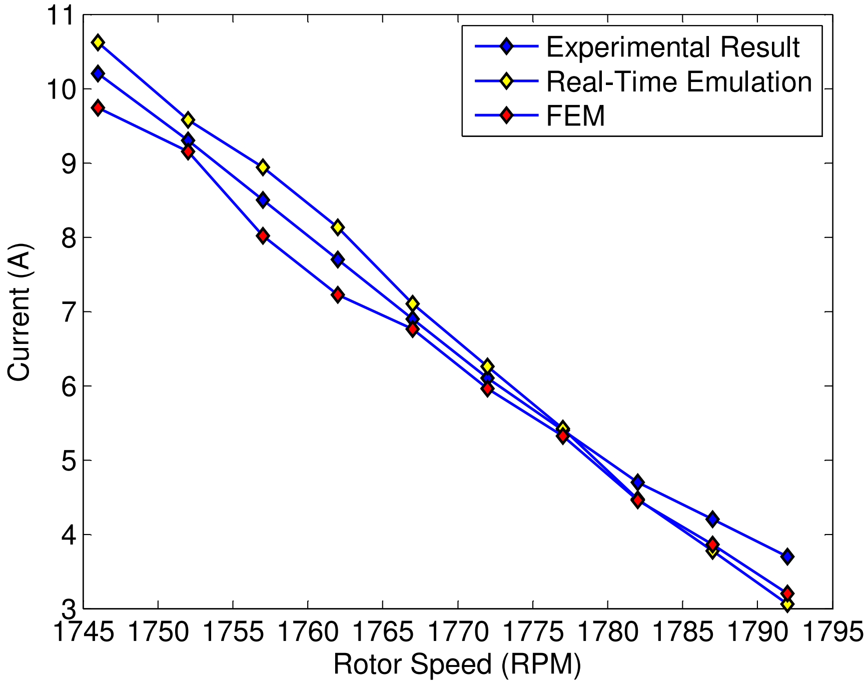


Figure 4.9: Steady-state performance of stator current at different rotor speeds.

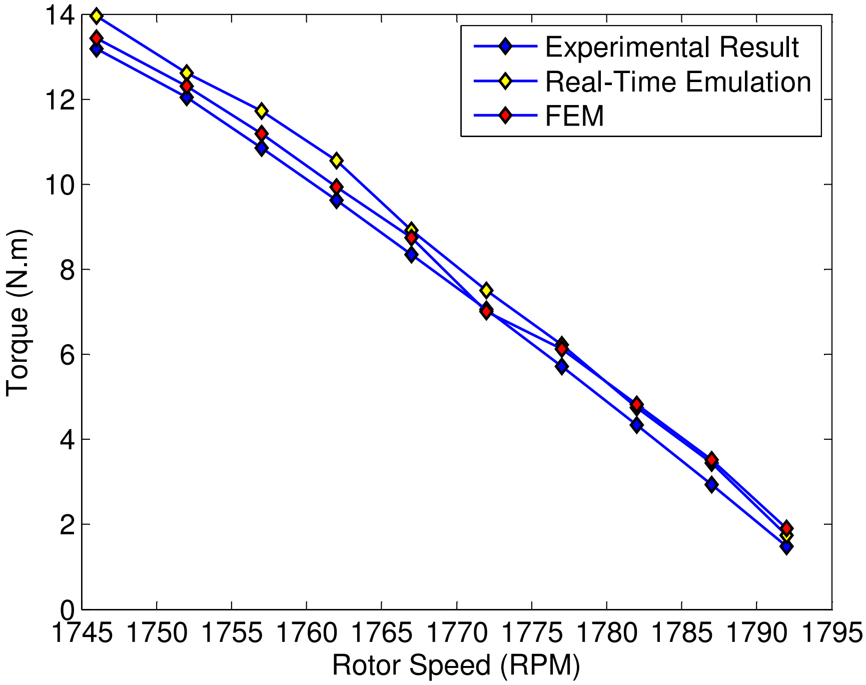


Figure 4.10: Steady-state performance of electromagnetic torque at different rotor speeds.

For further comparison, the phase current waveform obtained by various approaches at 1772 RPM are depicted in Fig. 4.11. The magnitude of the discrete fourier transform

(DFT) of the current is also presented in Fig. 4.12. It can be observed that the real-time MEC estimates reasonably the magnitude and waveshape of the current and its spectrum especially for dominant harmonics. The minor picks in low frequencies (frequency<500Hz), most likely due to spatial harmonics in the air-gap caused by semi-open stator slots, cannot be very closely determined by real-time MEC because of air-gap modeling with the lumped permeances. Another noticeable difference is high frequency distortions on real-time emulated current waveform. The attributed reasons are twofold. First, the maximum number of iteration in the N-R algorithm for real-time emulation on FPGA is limited to 15 iterations to achieve real-time computation within 400μs time-step. Thus, the convergence criterion is not satisfied in a few time-steps and the maximum iteration constraint launches the simulation into the next time-step leading a distortion in the results of those time-steps. What we can elaborate from the results is that in those non-converged iterations the last solutions are acceptably close to the convergence zone, consequently the local error in that time-step does not launch the nonlinear system into unstable region retaining the obtained results with physically realistic behavior, though they experience a high frequency distortion. Second, the emulation is carried out by single-precision floating point operations on the FPGA to satisfy execution time constraint. Therefore, lower accuracy of single-precision arithmetic operation compared with double-precision, which is used in JMAG® software in FEM, can be accounted as a computational error in each time-step and in the high frequency spectrum.

To quantify the difference between the real-time predictions and off-line numerical calculations (FEM) that would be allowed to converge, the squared norm of relative error of steady-state current is computed in this case which is almost 19.85%. This error can be reduced using more detailed and complex MEC models realized on the next generations of FPGA that offer larger amount of hardware resources for massive amount of computations with faster transistors for the implementation of MEC model with the higher clock frequency allowing more iterations in the nonlinear solution within the simulation time-steps.

Overall, the nonlinear real-time MEC model of induction machine can acceptably predict the machine performance in the transient and steady-state conditions. From computational prospective, real-time emulation of MEC on FPGA provides a considerable advantage over the time-consuming FEM and can be employed in the hardware-in-the-loop test setup.

4.5 Summary

The objectives of this work can be categorized in two parts: (1) A nonlinear MEC suitable was proposed for induction machine for real-time simulation, and (2) FPGA as a fastest computational engine was employed to solve the nonlinear equations of MEC in real-time.

The proposed MEC model in this work was similar to the one presented in [76] in terms

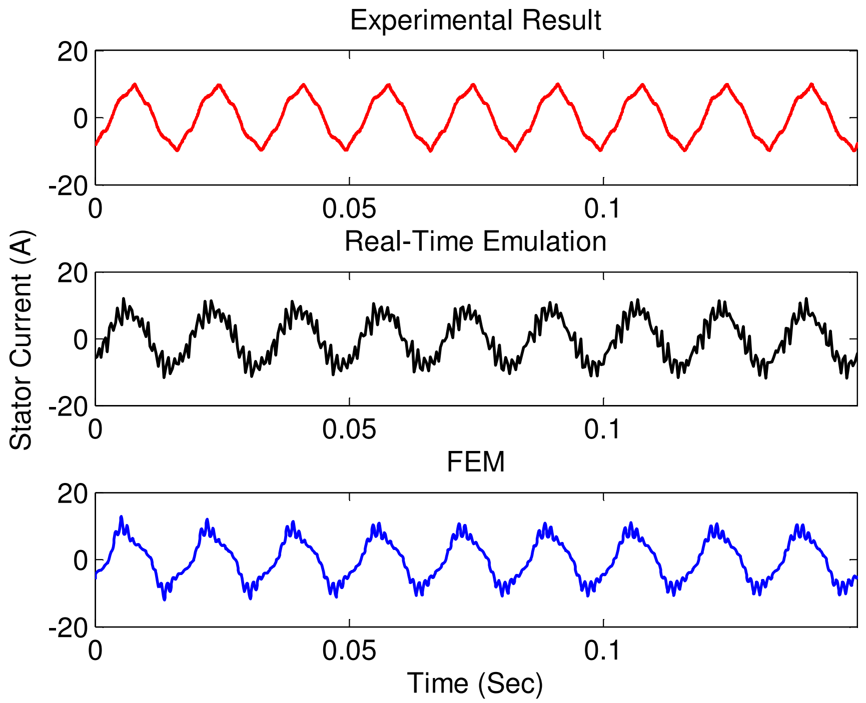


Figure 4.11: Comparison of steady-state stator current waveforms at 1772 RPM.

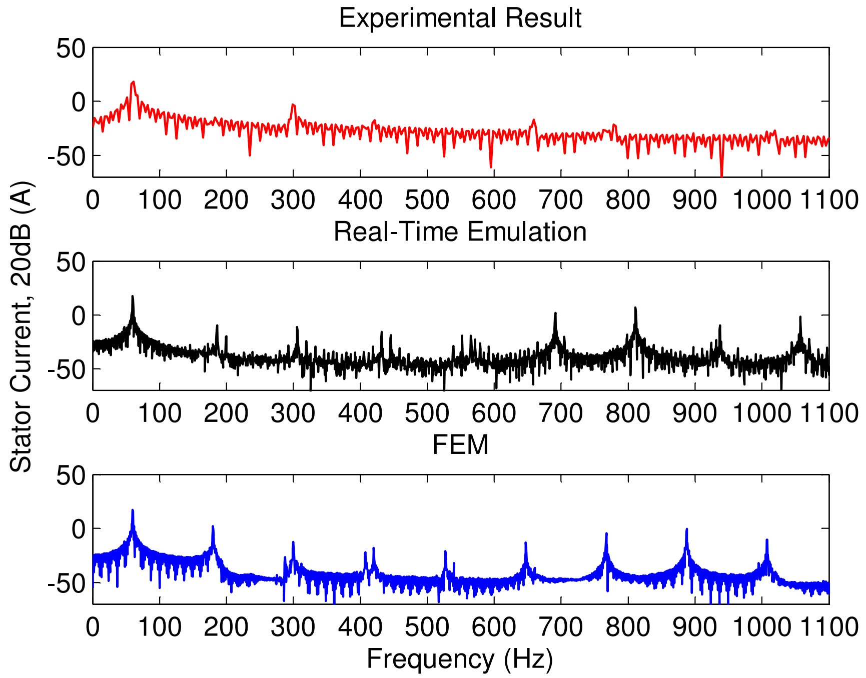


Figure 4.12: Comparison of DFTs of steady-state stator current at 1772 RPM.

of number of nodes. The minimum number of nodes was considered for permeance network to have minimum number of equations in the system model because of reducing the computational efforts for real-time calculations within a very small time-step size. However, it offered finer permeance element sizes in the MEC model to predict the induction machine performance more accurately, while the parallel hardware architecture of FPGA allowed computing a large number of permeance elements concurrently. In comparison with MEC in [79], in the proposed MEC the corresponding permeances for polar direction of flux paths in stator and rotor tooth were eliminated, and permeance elements for the radial direction of flux paths in the stator and rotor slot closure, stator yoke, and rotor central elements were neglected to reduce the real-time computational time significantly with just a slight reduction in the accuracy of the results due mainly to the fact that the magnetic flux flows mainly in the direction of the path of magnetic materials.

This chapter provided a hardware solution for real-time emulation of a nonlinear MEC model of induction machine on FPGA. The steady-state and transient results demonstrated that real-time emulated machine model follows the behavior of the induction machine obtained from experimental measurement and finite-element analysis closely. Consequently, the presented MEC model was suitable candidate for real-time emulation of induction machine which placed stringent execution time constraint while demanding reasonable accuracy. As the proposed MEC predicted the machine performances based on machine structure and geometry in real-time, such a model can be used in the iterative design optimization procedure of the machine to find the best set of machine parameters and dimensions in a large multi-dimensional search space to fulfill the desired objectives in a short time, also can be utilized in hardware-in-the-loop-simulation to test new controllers or drive systems against the virtual model of induction machine.

5 Real-Time Analytical Space Harmonic Model of Permanent Magnet Machines

5.1 Introduction

In this chapter [1] [2], a real-time emulation of an analytical space harmonic model is described to evaluate the performance of an optimized permanent-magnet machine. Due to the advantages of high power density, high efficiency, and low maintenance costs, permanent magnet (PM) machines are widely used in industrial and domestic applications [84, 85]. The flux density distribution of PM poles plays a crucial role in the machine performance since the average values of thrust or torque and their fluctuation especially depend on this distribution. Ideally, for optimal performance, the machine should have sinusoidal air-gap flux density distribution, sinusoidal current waveform, and quasi-sinusoidal distribution of stator conductors [86]. Therefore, different methodologies such as magnet-arc-shaping technique [87–92], pulse-wide modulation (PWM) approach [93, 94], modular PM pole method [95, 96], trapezoidal or sine structure with third harmonic injection shaping technique [97, 98], and stair-step shaped magnetic poles model [99] have been proposed in the literature for the shaping of PM poles to produce a sinusoidal magnetic field and to improve the machine efficiency.

A variety of methods have been employed to model the dynamic performance of PM machines with shaped magnetic poles. A lumped electric or magnetic equivalent circuit

[1] Material from this chapter has been published: N. R. Tavana, and V. Dinavahi, "Real-Time FPGA-Based Analytical Space Harmonic Model of Permanent Magnet Machines for Hardware-in-the-Loop Simulation", *IEEE Trans. on Magnetics*, vol. 51, no. 8, pp. 1-9, August 2015.

[2] Material from this chapter has been published: N. R. Tavana, A. Shoulaie, and V. Dinavahi, "Analytical Modeling and Design Optimization of Linear Synchronous Motor With Stair-Step-Shaped Magnetic Poles for Electromagnetic Launch Applications", *IEEE Trans. on Plasma Science*, vol. 40, no. 2, pp. 519-527, February 2012.

model is the fastest, and the most common technique to predict machine behavior. However, they suffer from the model inaccuracy particularly when the flux paths are complex and and flux leakages are significant [100]. Another candidate is numerical solution such as the finite-element method (FEM). Although FEM offers highly accurate results with due accounting of all nonlinear phenomena and spatial effects, it remains computationally onerous, and cannot be used for real-time prediction of machine behavior even with the aid of the powerful processors to accelerate computations.

To overcome the aforementioned problems, a 2-D analytical space harmonic solution is employed in this chapter for real-time simulation of PM machines with shaped magnetic poles with acceptable accuracy and reasonable computational efforts. The analytical approach offers the mathematical formulas for the prediction of machine performance as a function of machine geometry and design parameters, which would be very useful for real-time dynamic emulation, as well as design and performance optimization. Furthermore, the real-time emulation of the PM machine is carried out entirely in hardware on the FPGA. FPGAs are now becoming the most preferred computational engine for real-time simulation by offering many advantages such as hardwired architecture, programmability with all the algorithms and functions, short development period, and full parallelism [103–105].

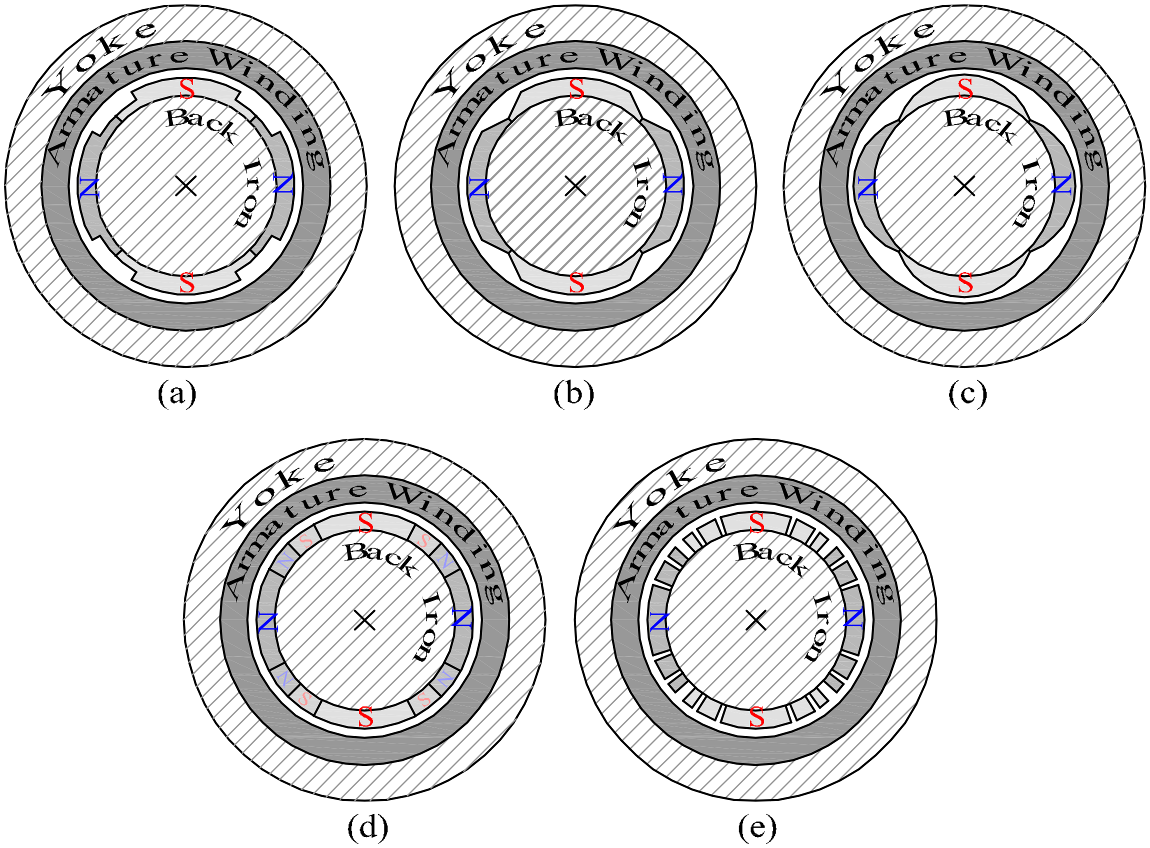
This study presents a unified framework for the real-time FPGA-based analytical space harmonic model of PM machines with the shaped magnetic poles. Such a real-time analytical treatment is useful for the performance evaluation of the PM machines in the HIL configuration. HIL technology provides a cost- and time-effective approach in a virtual scenario for testing real devices in a nondestructive environment [55, 57, 65, 101, 102, 106] and helps engineers to optimize the machine performance by running the emulated machine model in real-time using different geometrical parameters as many times as they need to reach the desired objectives.

This work presents the analytical solution incorporating Maxwell's equations for dynamic simulation of PM machines. A design optimization is then carried out to find the optimal dimensions and parameters of PM machines. Finally, digital hardware realization of the proposed method on FPGA is described in this study. The close agreement between the FPGA-based real-time analytically predicted and finite-element calculated machine performances are demonstrated.

5.2 Dynamic Analytical Space Harmonic Model of PM Machines with Shaped Magnetic Poles for Real-Time Simulation

5.2.1 Dynamic Performance of PM machines

It is necessary to establish the voltage and torque equations of the PM machine that can describe its dynamic performance. Each variable in the governing equations of machine behavior such as induced EMF voltages, and inductance matrix is then determined by the



(a) (b) (c)

(d) (e)

Figure 5.1: PM machines with different pole shapes. (a) Stair-step shaping technique. (b) Trapezoidal or sine with third harmonic injection method. (c) Arc-shaped model. (d) Modular magnetic poles. (e) Pulse-width modulation approach.

analytical space harmonic model.

The three-phase voltage equations in the PM machine variables are given as:

$$\boldsymbol{v}_{abc_s}(t) = \boldsymbol{r}_s \boldsymbol{i}_{abc_s}(t) + \boldsymbol{L}_{abc_s} \frac{d}{dt} \boldsymbol{i}_{abc_s}(t) + \boldsymbol{e}_{abc_s}(t). \tag{5.1}$$

where $\boldsymbol{r}_s$, $\boldsymbol{L}_{abc_s}$, $\boldsymbol{i}_{abc_s}$, and $\boldsymbol{e}_{abc_s}$ denote resistance, inductance, current, and induced electromagnetic force matrices, respectively.

Based on the instantaneous power of three-phase winding, the thrust/torque in a PM synchronous machine (PMSM) is obtained by

$$F(t) = \frac{1}{\nu_s} \boldsymbol{e}_{abc_s}(t) \cdot (\boldsymbol{i}_{abc_s}(t))^T. \tag{5.2}$$

where ν_s is synchronous speed.

For digital simulation of dynamic performance of PMSM, the differential equation (5.1) is discretized as follows

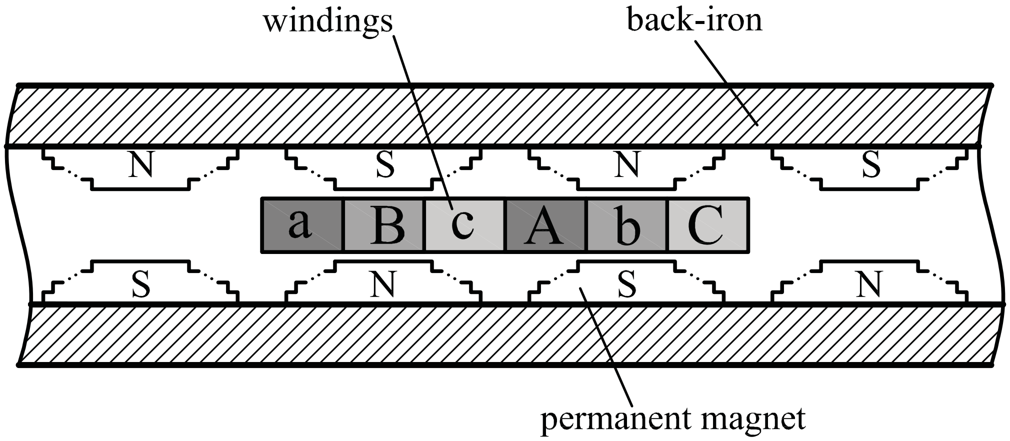


Figure 5.2: Topology of a double-sided PMLSM with stair-step shaped magnetic poles.

$$\boldsymbol{v}_{abc_s}(t) = \boldsymbol{r}_s \boldsymbol{i}_{abc_s}(t) + \frac{1}{\triangle t}\boldsymbol{L}_{abc_s}\boldsymbol{i}_{abc_s}(t) + \boldsymbol{e}_{abc_s}(t) + \boldsymbol{v}_{hist}, \tag{5.3}$$

where $\triangle t$ is the simulation time-step size and $\boldsymbol{v}_{hist}$ is the history term in every time-step, given by

$$\boldsymbol{v}_{hist} = -\frac{1}{\triangle t}\boldsymbol{L}_{abc_s}\boldsymbol{i}_{abc_s}(t - \triangle t). \tag{5.4}$$

5.2.2 Magnet Shaping methods in PM Machines

Since the structure of magnetic poles in the PM machine with arc, trapezoidal, PWM, modular, and stair-step shaping techniques, as depicted in Fig. 5.1, is not uniform, solving the Maxwell's equations for the calculation of magnetic fields and consequently the dynamic performance of the machine becomes more complex. The best way to establish analytical solution in these cases is to split magnetic poles to finitely many uniform and rectangular segments and then apply the layer method to each segment. Additionally, as the soft magnetic parts are assumed to be infinitely permeable and all materials behave linearly in this layer approach, the resultant magnetic field can be obtained using the superposition theorem.

In this chapter, the layer method is applied to a small segment of PM poles and the analytical prediction of magnetic field is presented thereafter. The governing equations can be used for all types of shaped PM poles in either rotary or linear structures. Without losing the generality of solution for the real-time analytical model of PM machines, a double sided air-core permanent magnet linear synchronous motor (PMLSM) with stair-step shaped magnetic poles is chosen in this work as the case study shown in Fig. 5.2.

5.2.3 Magnetic Field Distribution of PM Source

In order to provide analytical solution for the magnetic field distribution of *i*th magnetic segment of a shaped magnetic pole shown in Fig. 5.3(a), the permeability of back iron is assumed to be equal to infinite. Consequently, Maxwell's equations lead to Laplace's and Poisson's equations in Layer I (airspace/winding layer) and Layer II (PM layer), respectively. Fig. 5.3(b) shows the field layers of the *i*th segment. The governing field equations in terms of magnetic vector potential $\boldsymbol{A}$ can be expressed as follows [100, 107]:

$$\begin{cases} \nabla^2 \boldsymbol{A}_{I_i} = 0 & \text{in Layer I,} \\ \nabla^2 \boldsymbol{A}_{II_i} = -\mu_0 \boldsymbol{J}_{M_i} & \text{in Layer II,} \end{cases} \tag{5.5}$$

where $\boldsymbol{J}_{M_i} = \nabla \times \boldsymbol{M}_i$ and $\boldsymbol{M}_i$ is the magnetization vector of the *i*th magnetic segment given by

$$\boldsymbol{M}_i = M_{y_i} \boldsymbol{a}_y, \tag{5.6}$$

where M_{y_i} denotes the component of $\boldsymbol{M}_i$ in y direction. The distribution of M_{y_i} can be expressed as the Fourier series:

$$M_{y_i} = \sum_{n=1,3,\ldots}^{\infty} P_n \sin\left(\frac{m_n w_{m_i}}{2}\right) \cos\left(m_n x_i\right), \tag{5.7}$$

where $P_n = 4B_r/n\pi\mu_0$ and $m_n = n\pi/\tau$. Additionally, B_r and τ represent magnetic remanence and pole pitch, respectively. The boundary conditions to be satisfied by the solution to (5.5) are

$$H_{Ix_i}|_{y_i=l_{g_i}} = 0; \; H_{IIx_i}|_{y_i=-l_{m_i}} = 0,$$
$$H_{Ix_i}|_{y_i=0} = H_{IIx_i}|_{y_i=0}; \; B_{Iy_i}|_{y_i=0} = B_{IIy_i}|_{y_i=0}, \tag{5.8}$$

where H denotes magnetic field intensity, and l_{g_i} is the effective air-gap length of the *i*th segment.

By solving (5.5), the flux density distribution (B) produced by the *i*th segment in the air gap is provided from the curl of $\boldsymbol{A}_{I_i}$ as follows:

$$\begin{aligned} B_{Ix_i}(x_i, y_i) = & \sum_{n=1,3,\ldots}^{\infty} m_n [a_{In_i} \exp\left(m_n y_i\right) \\ & -b_{In_i} \exp\left(-m_n y_i\right)] \sin(m_n x_i), \end{aligned} \tag{5.9}$$

$$\begin{aligned} B_{Iy_i}(x_i, y_i) = & -\sum_{n=1,3,\ldots}^{\infty} m_n [a_{In_i} \exp\left(m_n y_i\right) \\ & +b_{In_i} \exp\left(-m_n y_i\right)] \cos(m_n x_i), \end{aligned} \tag{5.10}$$

where a_{In_i} and b_{In_i} are determined as

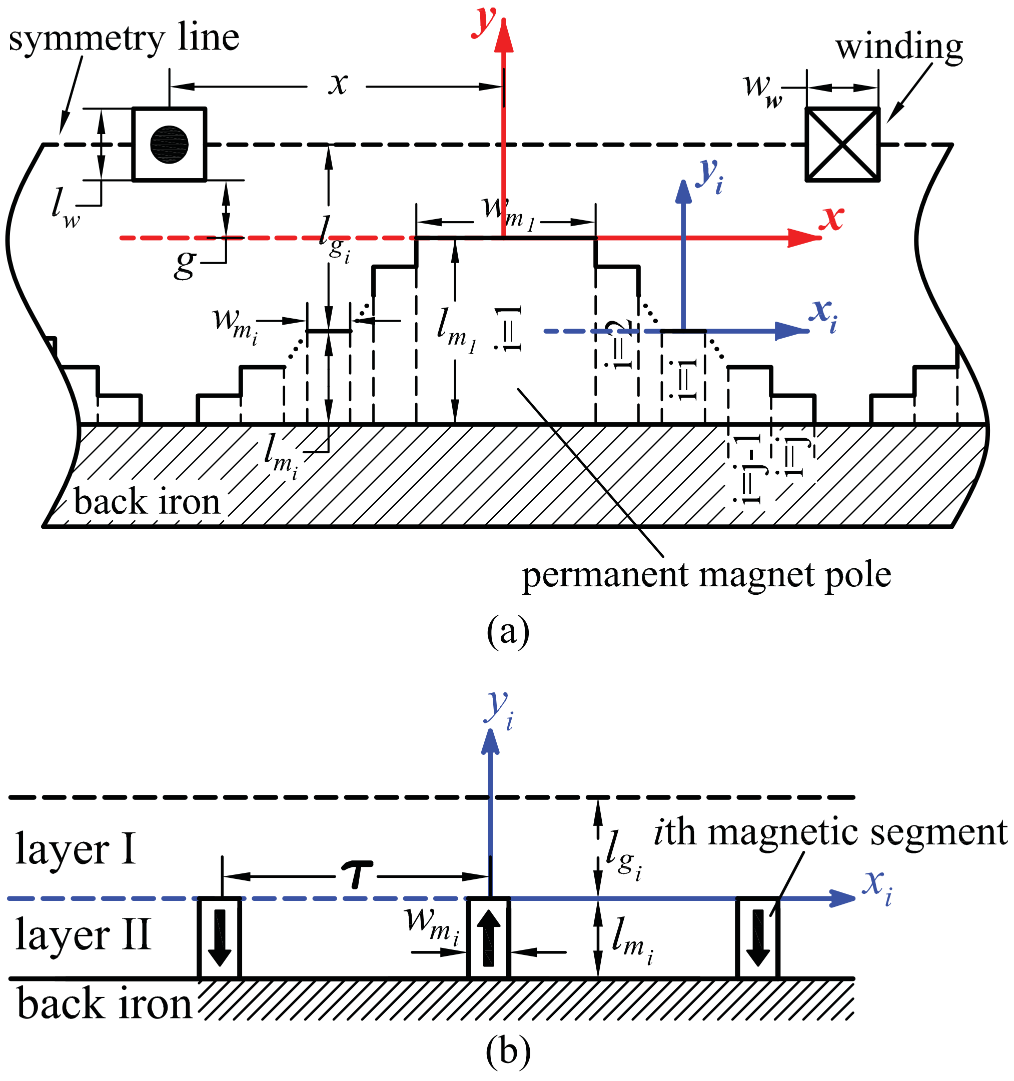


Figure 5.3: Magnetic field analysis model for the PM: (a) Simplified model and (b) field regions.

$$a_{In_i} = \exp\left(-2m_n l_{g_i}\right) b_{In_i}, \tag{5.11}$$

$$b_{In_i} = -\frac{P_n \mu_0}{2m_n} \sin\left(\frac{m_n w_{m_i}}{2}\right) \frac{1-\exp\left(-2m_n l_{m_i}\right)}{1-\exp\left(-2m_n\left(l_{m_i}+l_{g_i}\right)\right)}. \tag{5.12}$$

where l_{m_i} and w_{m_i} are the height and width of ith magnetic segment.

Due to the linear behavior of the model, the total flux density of shaped PM poles is obtained using superposition theorem as

$$B_{Ix}(x,y)=\sum_{n=1}^{N}B_{Ix_i}(x_i,y_i);\;\; B_{Iy}(x,y)=\sum_{n=1}^{N}B_{Iy_i}(x_i,y_i). \tag{5.13}$$

where N is the number of PM segments.

5.2.4 EMF Calculation

The total flux-linkage of a distributed multi-coil phase winding (φ) caused by the *i*th magnetic segments can be obtained as [108]:

$$\begin{aligned}\varphi_i(x_i) &= \oint_s B_{Iy_i}.ds=\int_0^L\int_{x_i-\tau_{wp}}^{x_i}B_{Iy_i}dxdz\\ &= \sum_{n=1,3,...}^{\infty}\phi_{n_i}\cos m_n(x_i-\frac{\tau_{wp}}{2}),\end{aligned} \tag{5.14}$$

and

$$\begin{aligned}\phi_{n_i}=&\frac{8PLp_fk_{wn}N_c}{l_w}\sin(\frac{m_n\tau_{wp}}{2})\\ &\times\int_{g_i}^{l_{g_i}}[a_{In_i}\exp(m_ny_i)+b_{In_i}\exp(-m_ny_i)]dy.\end{aligned} \tag{5.15}$$

where P, p_f, k_{wn}, and N_c denote number of pole pairs, coil packing factor, winding factor of nth harmonic of magnetic field, and number of winding turn. Plus, L, τ_{wp}, and l_w represent motor width, winding pitch, and winding height.

Using Faraday's law, the induced EMF due to the *i*th segment is given by

$$\begin{aligned}e_i\left(x_i\right)=-\frac{d\varphi_i}{dt}&=-\nu_s\sum_{n=1,3,...}^{\infty}\phi_{n_i}m_n\sin m_n\left(x_i-\frac{\tau_{wp}}{2}\right)\\ &=\sum_{n=1,3,...}^{\infty}CoEMF_n\sin m_n\left(x_i-\frac{\tau_{wp}}{2}\right).\end{aligned} \tag{5.16}$$

Thus, the EMF produced by all magnetic segments can be expressed as follows:

$$e(x(t))=\sum_{i=1}^{N}e_i\left(x_i\right). \tag{5.17}$$

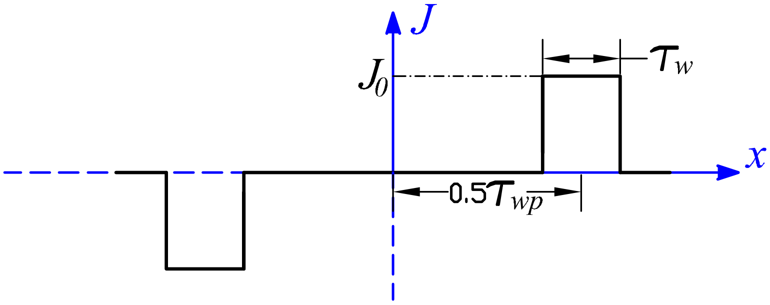


Figure 5.4: Current distribution function of winding layer.

5.2.5 Armature Magnetic Field

The magnetic field caused by armature current is calculated in a similar way to the magnetic field of PM source. The governing equations in the winding layer and other regions are given by

$$\begin{cases} \nabla^2 \boldsymbol{A}_I = -\mu_0 \boldsymbol{J}_c & \text{in layer I,} \\ \nabla^2 \boldsymbol{A}_{II} = 0 & \text{in layer II.} \end{cases} \tag{5.18}$$

The current distribution function of winding layer $\boldsymbol{J}_c$, as shown in Fig. 5.4, is expanded as follows [108]:

$$\boldsymbol{J}_c = -\sum_{i=1,3,\ldots}^{\infty} J_n \sin\left(m_n x\right) \boldsymbol{a}_z, \tag{5.19}$$

and

$$J_n = \frac{4J_0}{\tau m_n} \sin\left(m_n \frac{\tau_{wp}}{2}\right) \sin\left(m_n \frac{\tau_w}{2}\right). \tag{5.20}$$

where τ_w denotes winding width and J_0 is current density.

The magnetic flux density distribution in the winding layer is obtained by

$$\begin{aligned} B_{Ix}(x,y) = & \sum_{n=1,3,\ldots}^{\infty} m_n [a_{In} \exp\left(m_n y\right) \\ & -b_{In} \exp\left(-m_n y\right)] \sin(m_n x), \end{aligned} \tag{5.21}$$

$$\begin{aligned} B_{Iy}(x,y) = & -\sum_{n=1,3,\ldots}^{\infty} m_n [a_{In} \exp\left(m_n y\right) \\ & +b_{In} \exp\left(-m_n y\right) - \frac{\mu_0 J_n}{{m_n}^2}] \cos(m_n x), \end{aligned} \tag{5.22}$$

where

$$a_{In} = \exp\left(2m_n l_w\right) b_{In}, \tag{5.23}$$

$$b_{In} = \frac{\mu_0 J_n}{2{m_n}^2} \frac{\exp\left(2m_n\left(g+l_m\right)\right)-1}{\exp\left(2m_n\left(g+l_m+l_w\right)\right)-1}\ . \tag{5.24}$$

The boundary conditions are fulfilled as:

$$\begin{gathered} H_{Ix}\big|_{y=-l_w} = 0;\ H_{IIx}\big|_{y_i=g+l_m} = 0, \\ H_{Ix}\big|_{y=0} = H_{IIx}\big|_{y=0};\ B_{Iy}\big|_{y=0} = B_{IIy}\big|_{y=0}\,. \end{gathered} \tag{5.25}$$

5.2.6 Self and Mutual Inductances

The flux linkage of a phase winding due to the armature reaction field of another winding is expressed by

$$\begin{aligned} \varphi(x) &= \oint_s B_{Iy}.ds = \int_0^L \int_{x-\tau_{wp}/2}^{x+\tau_{wp}/2} B_{Iy} dx dz \\ &= \sum_{n=1,3,\ldots}^{\infty} \phi_n \cos m_n(x), \end{aligned} \tag{5.26}$$

where

$$\begin{aligned} \phi_n = & \frac{8PLp_f k_{wn} N_c}{l_w} \sin(\frac{m_n \tau_{wp}}{2}) \\ & \times \int_0^{l_w} [a_{In} \exp\left(m_n y\right) + b_{In} \exp\left(-m_n y\right) - \frac{\mu_0 J_n}{{m_n}^2}] dy. \end{aligned} \tag{5.27}$$

Thus, the self-inductance (L_s) and mutual-inductance (L_m) of one slot per phase per pole armature winding can be calculated by

$$L_s = \frac{N_c \phi_a}{i_a}\bigg|_{i_b=i_c=0} = \frac{{N_c}^2 \varphi(0)}{J_0 \tau_w l_w}, \tag{5.28}$$

$$L_m = \frac{N_c \phi_b}{i_a}\bigg|_{i_b=i_c=0} = \frac{{N_c}^2 \varphi(\tau_{wp})}{J_0 \tau_w l_w}. \tag{5.29}$$

5.2.7 Comparison with Finite-Element Method

The validity of the analytical results for the calculation of magnetic field, EMF and thrust force in the PM machines with stair-step shaped magnetic poles (S^{3}MPs) greatly depends on the accuracy of the analytical model. However, the analytical model is based on some

Table 5.1: Parameters of the Initial PMLSMs.

	Motor Types				
Parameter	**Conventional Poles**	**Two-step S^{3}MPs**		**Three-step S^{3}MPs**	
$l_m(mm)$	4.4	l_{m_1} l_{m_2}	5 3.1	l_{m_1} l_{m_2} l_{m_3}	5.8 5.2 3.5
$w_m(mm)$	34	w_{m_1} w_{m_2}	22.6 8.3	w_{m_1} w_{m_2} w_{m_3}	18.8 2.4 4.4
$\tau(mm)$	42				
$l_w(mm)$	6				
$w_w(mm)$	14				
$J(A/mm^2)$	5				
$B_r(T)$	1.13				
$g(mm)$	1				

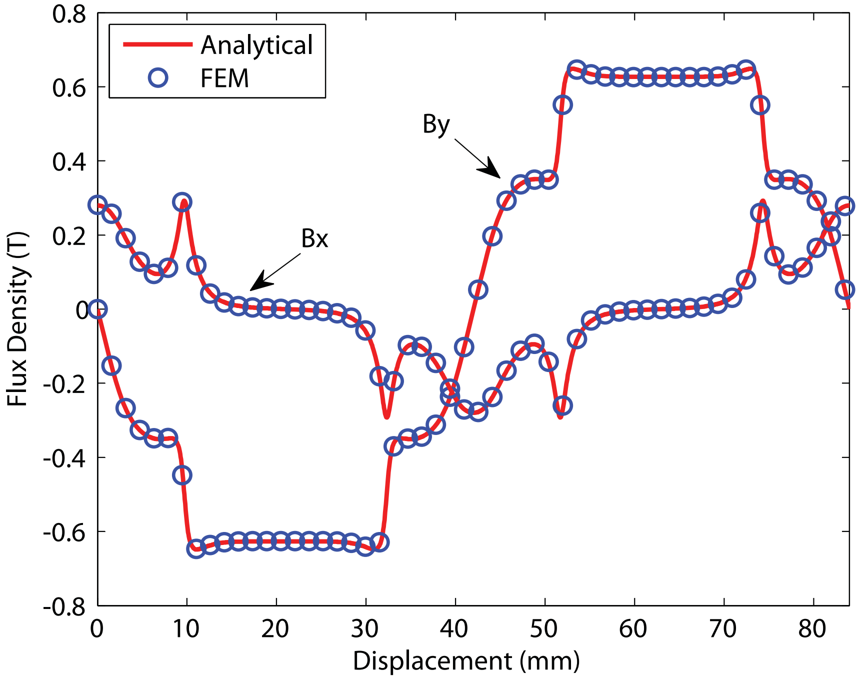


Figure 5.5: Flux density distribution as a function of x at the center of the air gap.

simplifying assumption such as ignoring saturation and considering an infinite motor length. Thus, it is necessary to evaluate the extent of model accuracy. In this section, a 2-D nonlinear time-stepping transient finite-element method is employed to validate the model.

The proposed analytical model is applied to a PMLSM with two-step PM poles whose

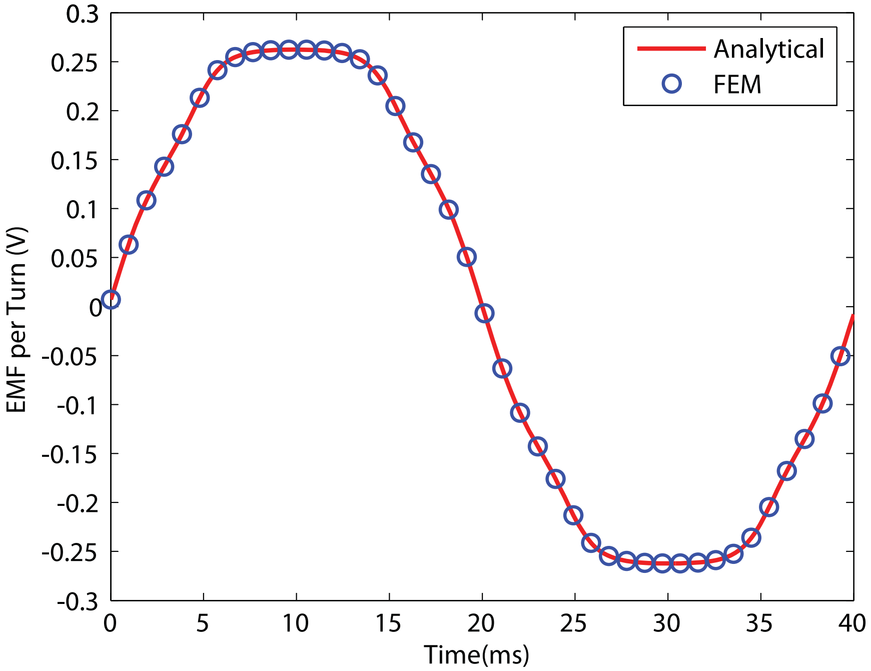


Figure 5.6: EMF per turn at $v_s = 2.1(m/s)$ of the PMLSM.

parameters are listed in Table 5.1. This case is chosen for simplicity without loss of generality. The results can be extended to any other pole configurations with more steps.

Fig. 5.5 shows the comparisons between the normal and tangential components of open-circuit flux density distributions computed by the analytical and finite-element method as functions of x position at the center of the mechanical air gap. It is seen that the FEM accurately verifies the analytical method.

A comparison of the EMF waveform per turn for a constant armature speed of 2.1 (m/s) obtained by the analytical and finite-element method is depicted in Fig. 5.6. Again a close agreement between the results of the analytical method and FEM is observed.

Fig. 5.7 compares analytically predicted and finite-element-calculated thrust force distribution with respect to the mover position. It is seen that the analytical prediction agrees well with the finite-element solution; the slight discrepancy is due mainly to the effect of core saturation, which is neglected in the analytical model.

5.3 Optimal Design of S^{3}MPs

The procedure for any optimization is to find a vector $X = (x_1, x_2, ..., x_n)$, representing a set of n design variables, each of them bounded by $x_{i_{min}} \leq x_i \leq x_{i_{max}}$, $i = 1, 2, ..., , n$, so that the objective function $F(X)$ is maximized (or minimized), subject to a set of k constraints $G_j(X) \leq 0$, $j = 1, 2, ..., k$. As the genetic algorithm is widely used to optimize electromagnetic problems [109]- [111], in this work genetic algorithm is applied to the de-

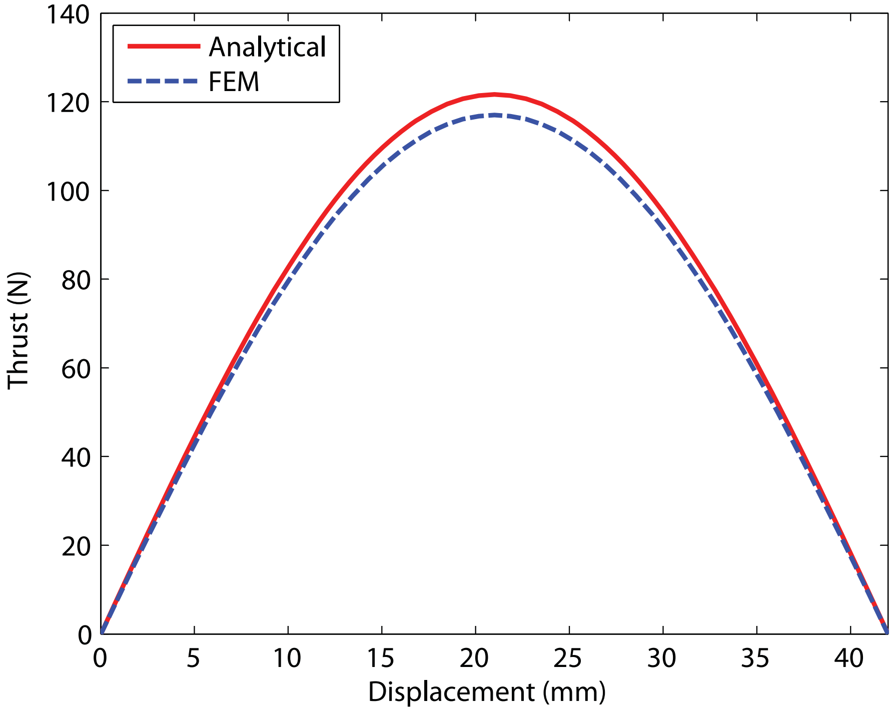


Figure 5.7: Thrust characteristic of the PMLSM.

sign of PMLSM to reach an air-gap flux density distribution close to sinusoidal form. Total harmonic distortion (THD) of flux density distribution is used as the objective function as

$$THD\,(X) = \frac{\sqrt{\sum_{n=3,5,\ldots}^{\infty} B_{yn}^{2}}}{B_{y1}}. \tag{5.30}$$

The optimization procedure is employed to minimize the THD. Magnet dimensions are chosen as the optimization variables.

Fig. 5.8 shows the variation of THD in the analyzed PMLSM with two-step PM poles in terms of w_{m_2} and $l_{m_1} - l_{m_2}$ while l_{m_1} and $w_{m_1} + w_{m_2}$ are fixed. It is seen that an air-gap flux density distribution close to sinusoidal form can be obtained by choosing appropriate width and height of each step in the PM poles.

Figs. 5.9 and 5.10 also show the waveform and THD of the normal flux density component as a function of x position in the conventional, two-step, and three-step magnetic poles, whose parameters are presented in Table 5.1. It is seen that the number of steps effectively influences the THD. The more steps S^{3}MPs have, the more THD decreases, however, the more complexity in the manufacturing of this kind of PMLSM the designer encounters. Therefore, in study PM machines with only two and three steps are selected for investigation.

The results, presented in Figs. 5.8 to 5.10, confirm the need for design optimization to achieve sinusoidal distribution of flux density by choosing appropriate PM dimensions.

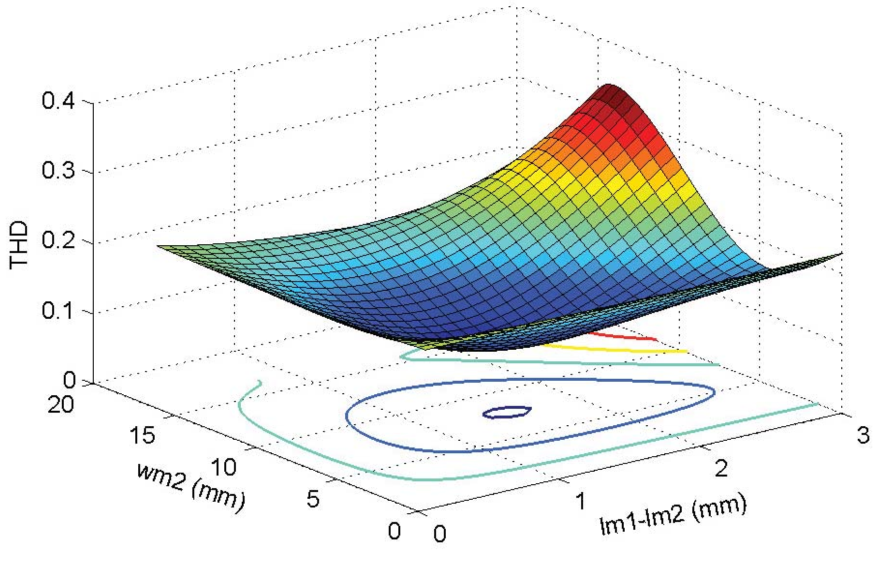


Figure 5.8: THD variation with w_{m_2} and $l_{m_1} - l_{m_2}$ for the PMLSM.

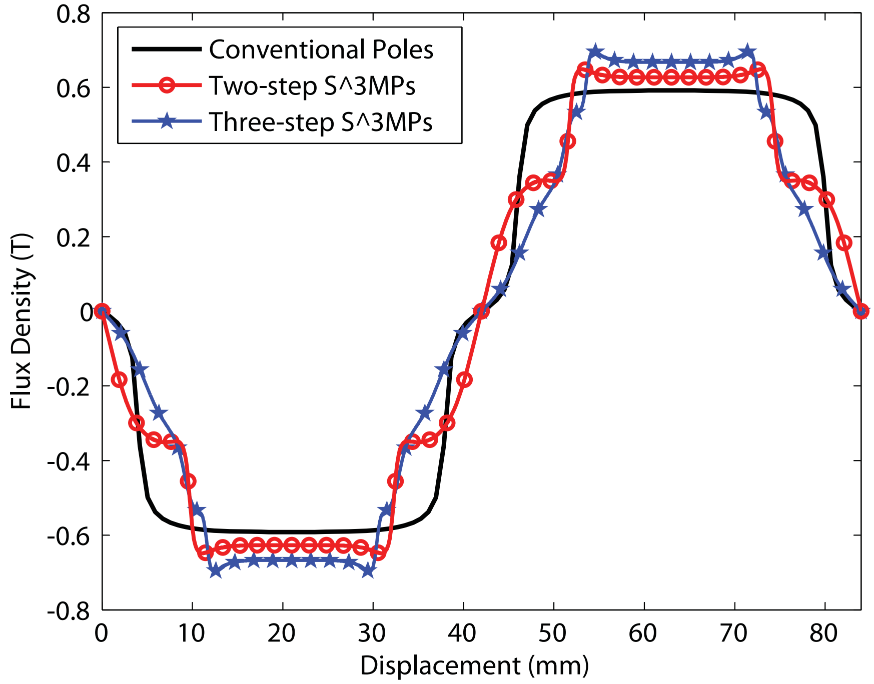


Figure 5.9: Air-gap flux density distribution of the PMLSM.

Finding the optimal dimensions of S^{3}MPs with the direct search method is extremely cumbersome and would take a long time. Therefore, a genetic algorithm has been employed for the optimization procedure.

This technique consists of three basic operators, i.e. selection, crossover and mutation.

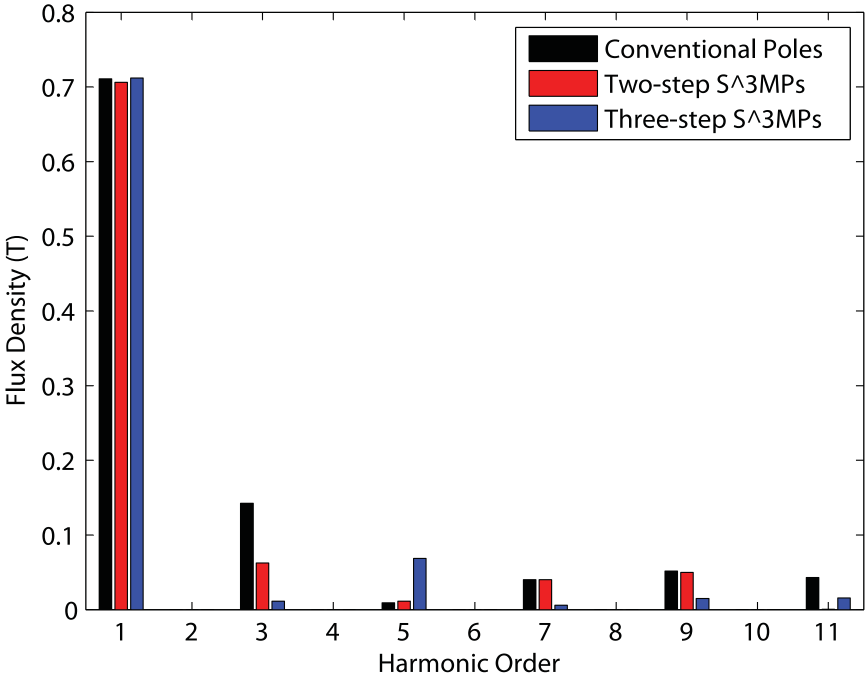


Figure 5.10: Harmonics of air-gap flux density distribution.

Table 5.2: Design Variables and Constraints.

Parameter Change Range
$2(mm) \leq l_{m_j} \leq l_{m_{j-1}} \leq ... \leq l_{m_i} \leq ...$ $... \leq l_{m_2} \leq l_{m_1} \leq 6.5(mm)$
$20(mm) \leq 2w_{m_j} + 2w_{m_{j-1}} + ...$ $... + 2w_{m_i} + ... + 2w_{m_2} + w_{m_1} \leq 42(mm)$
$120(N) \leq F_{avg} \leq 122(N)$

Table 5.3: Parameters of the Optimized S^{3}MPs.

	Motor Types			
Parameter	**Two-step S^{3}MPs**		**Three-step S^{3}MPs**	
$l_m(mm)$	l_{m_1}	5.9	l_{m_1}	6.1
			l_{m_2}	5.4
	l_{m_2}	4.5	l_{m_3}	4
$w_m(mm)$	w_{m_1}	17	w_{m_1}	12.7
			w_{m_2}	4.5
	w_{m_2}	7.7	w_{m_3}	6.2

After initial population is randomly generated, the genetic algorithm operators are applied to the population to reduce their cost gradually. The roulette wheel selection is used for

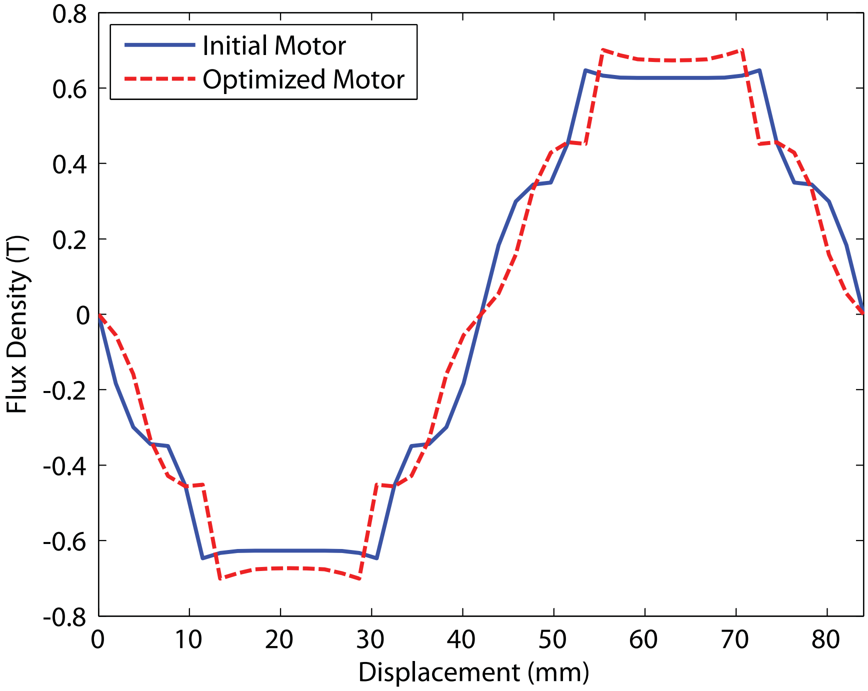


Figure 5.11: Air-gap flux density distribution of two-step S^3MPs in the PMLSM.

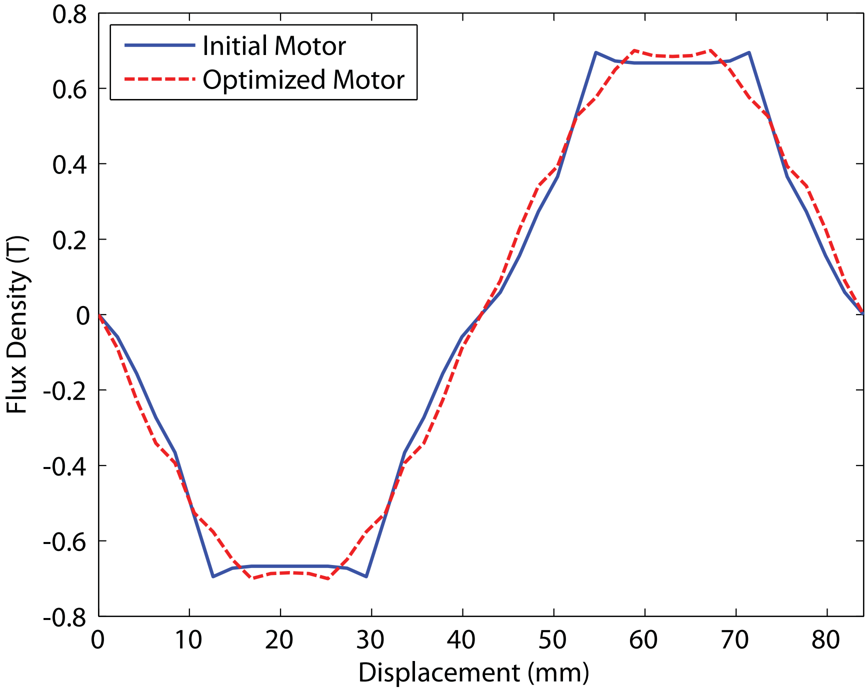


Figure 5.12: Air-gap flux density distribution of three-step S^3MPs in the PMLSM.

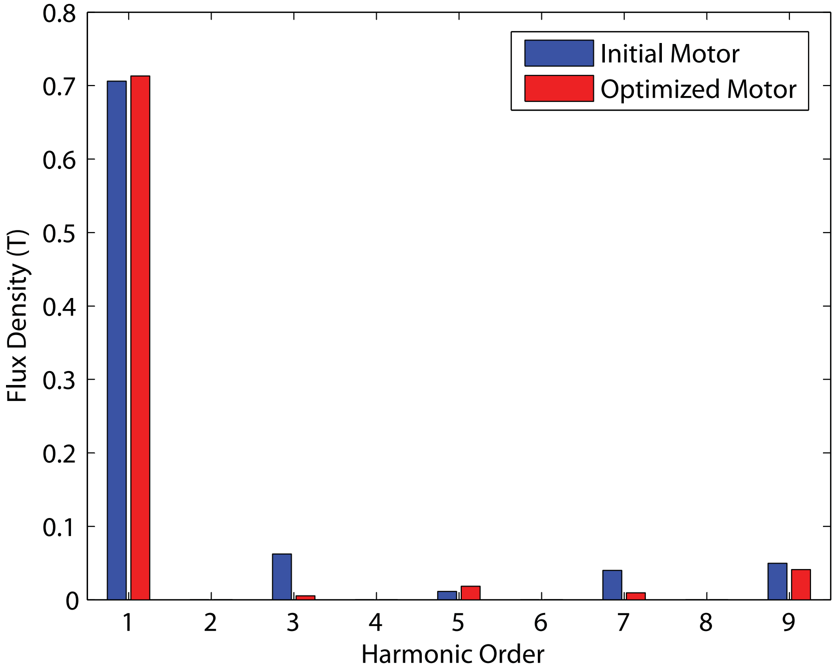


Figure 5.13: Harmonics of air-gap flux density distribution of two-step S^{3}MPs in the PMLSM.

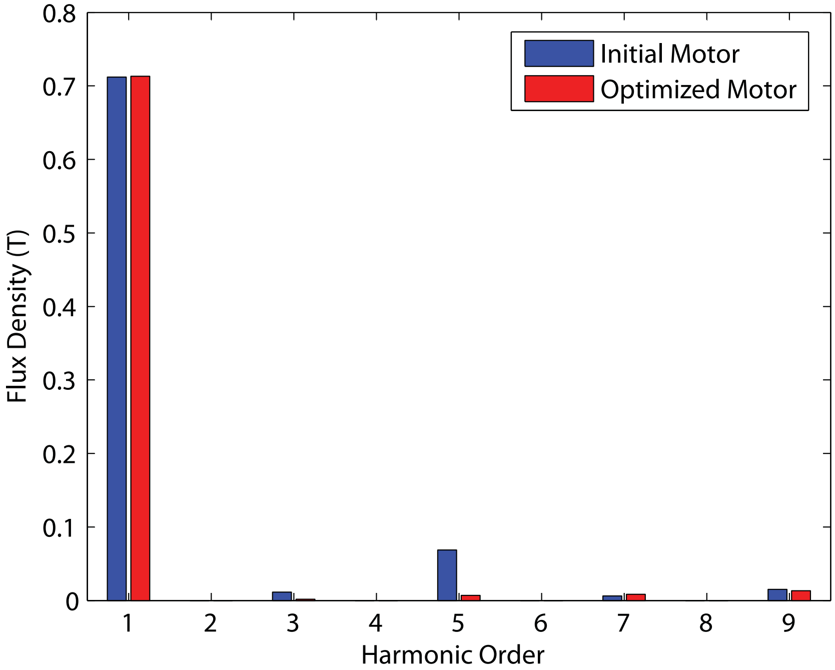


Figure 5.14: Harmonics of air-gap flux density distribution of three-step S^{3}MPs in the PMLSM.

reproducing the population.

The following values of genetic operators are used in the genetic algorithm:

1. initial population: $N = 50$;
2. probability of mutation: $P_m = 0.07$;
3. probability of crossover: $P_c = 0.07$;
4. number of generations: $N_g = 1000$.

To have a more realistic design, some limitations are applied to the optimization procedure as shown in Table 5.2. Finally, in the optimal models, PM dimensions calculated using the genetic algorithm are listed in Table 5.3.

5.4 Design Evaluation

The performance of optimized PMLSMs is investigated and compared with the performance of initial PMLSMs with the specifications listed in Table 5.1. Since an optimization procedure based on the analytical method has been employed to determine the optimal design specifications, another method with the higher accuracy, for example, the FEM should be used to assess the performance of the optimized motors.

The flux density distributions produced by the PM poles of the optimal designs and initial machines in the middle of the mechanical air gap calculated by FEM are presented in Figs. 5.11 and 5.12. The harmonic contents of flux density distribution for both initial and optimized motors are also shown in Figs. 5.13 and 5.14. It can be seen that the harmonics in the optimized motor reduce substantially. In the two-step S^{3}MPs, THD reduces from 14% in the initial motor to 9.1% in the optimal one, while in the three-step S^{3}MPs, THD decreases from 10.6% in the initial motor to 5.2% in the optimized one.

Moreover, the EMF waveforms at the nominal speed obtained by FEM are depicted in Figs. 5.15 and 5.16. The fundamental component amplitudes of EMF curves of the optimized motor are calculated and used as the amplitudes of the standard sinusoidal waveforms. It is also observed that the mismatch between the standard sinusoidal curves and EMF waveforms of optimal motors are approximately zero.

5.5 Real-Time Hardware Emulation of Analytical Space Harmonic Model of PMSM on FPGA

In this chapter, the optimized LPMSM with two-step S^{3}MPs shown in Fig. 5.17 is chosen for real-time study. The real-time emulation of PMLSM is carried out using the single precision floating-point number representation (*IEEE* standard 754). This 32-bit format enjoys the advantages of dynamic range, satisfactory accuracy, and acceptable computational

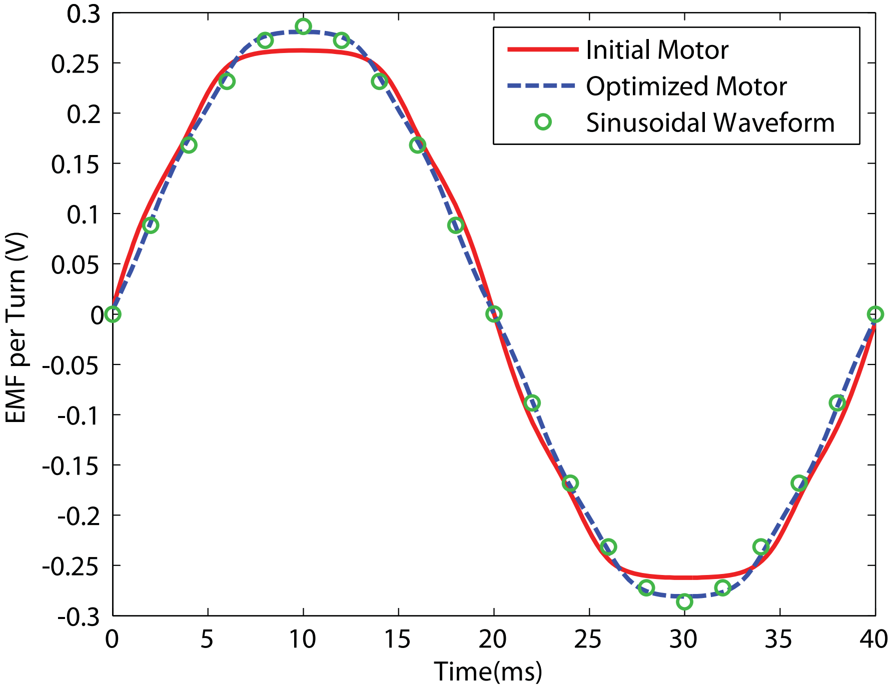


Figure 5.15: EMF of two-step S^{3}MPs in the PMLSM.

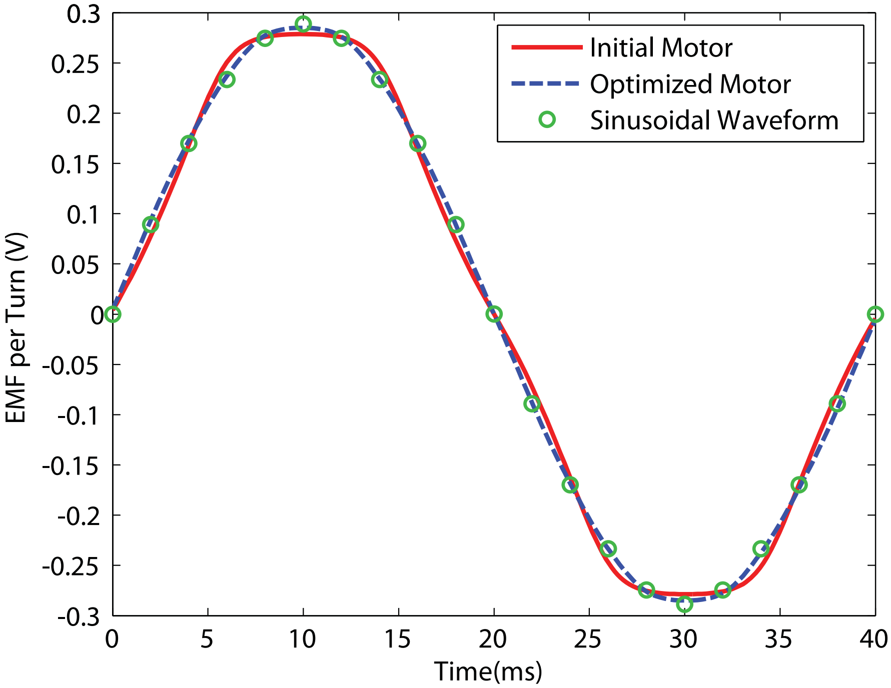


Figure 5.16: EMF of three-step S^{3}MPs in the PMLSM.

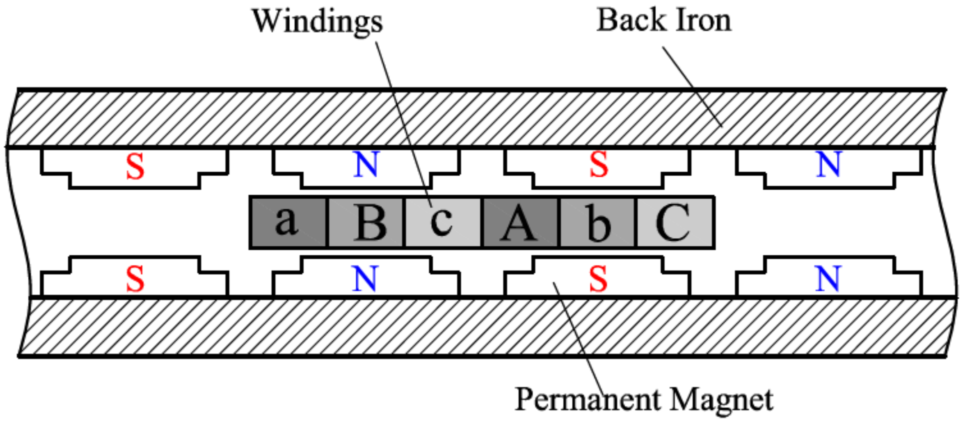


Figure 5.17: Topology of a double-sided PMLSM with two-step S^{3}MPs.

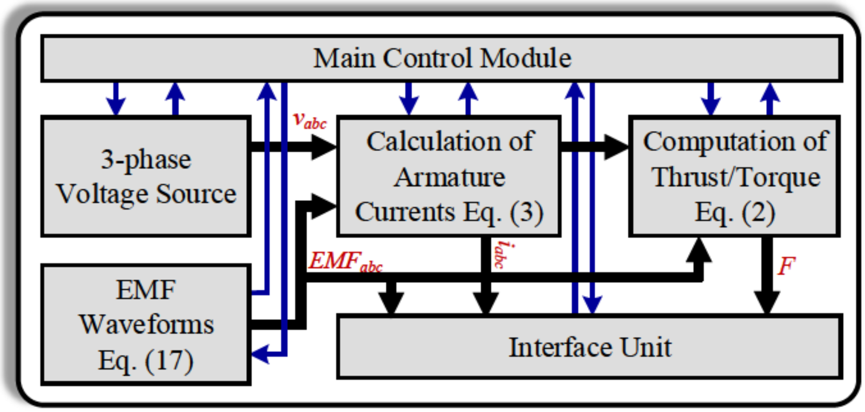


Figure 5.18: Generic machine model implementation on FPGA.

time. A generic hardware realization of PM machine on FPGA is presented in Fig. 5.18. The hardware implementation of the machine performance at synchronous speed mainly consists of three states. The procedure starts with the calculation of three-phase voltage sources and the open-circuit voltages induced in the three-phase stator windings simultaneously in the first state. Then in the second state, the stator current is computed by the history information (5.4) and Forward Euler integration technique. The instantaneous electromagnetic thrust/torque is obtained in the last state, meanwhile the history terms are updated.

The hardware architecture of the machine model is realized by the aid of the source module for generating the three-phase voltage supply, matrix multiplication unit, dedicated floating-point multiply-add/subtract unit, and timer module for checking execution time constraint. All aforementioned modules and their structures have been discussed in details in [47, 81, 83]. However, the main and most complex hardware unit for the implementation of the analytical model of PMSM on FPGA is the EMF module where a

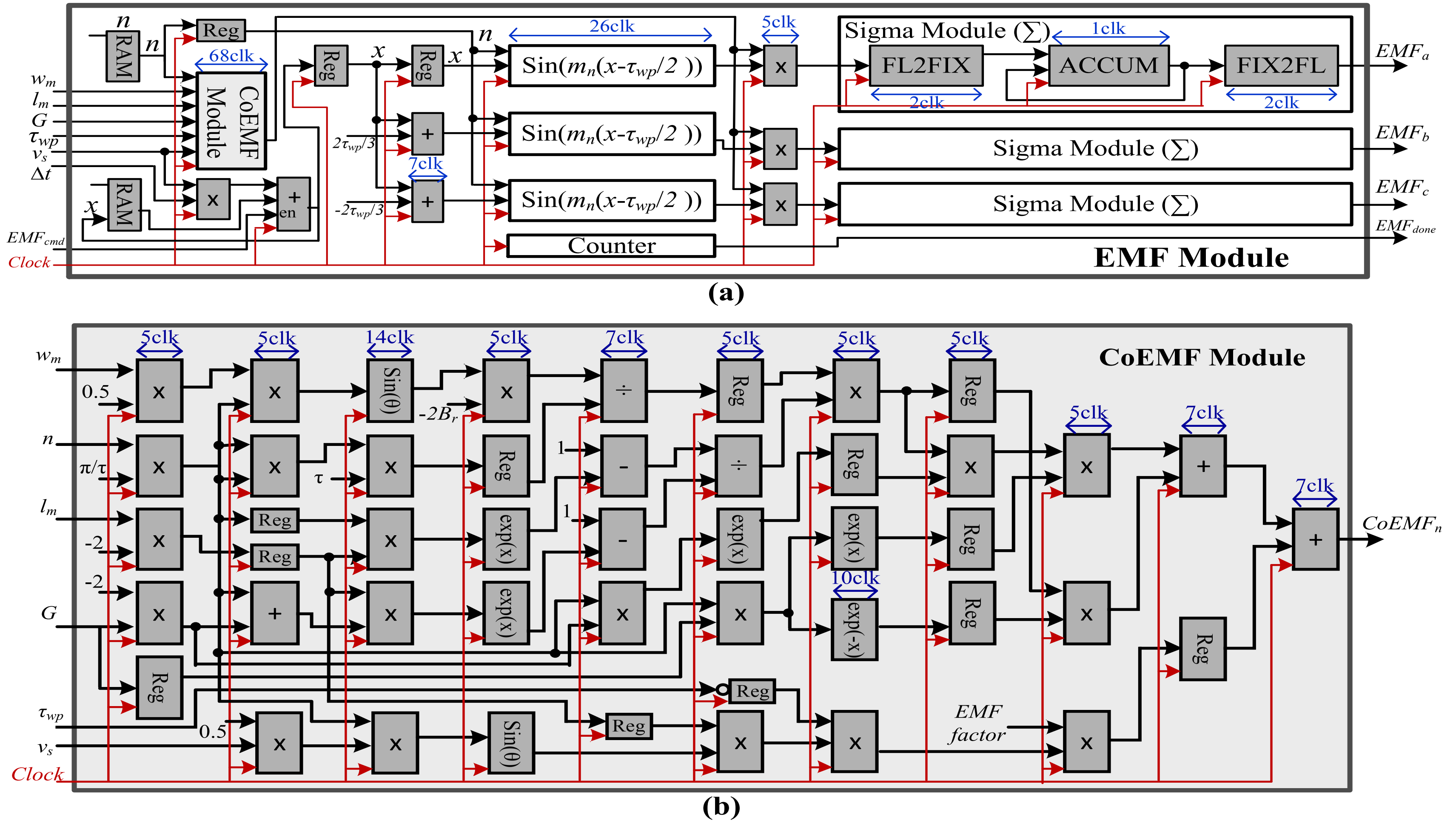


Figure 5.19: Digital hardware realization of EMF module. (a) Deeply pipeline structure of EMF architecture. (b) Massively parallel implementation of EMF coefficient factor.

Table 5.4: Hardware setup for the real-time analytical emulation of PMLSM.

Real-Time Emulator	**EMF Module**	**Source Module**	**Others**	**Total**
Logic LUTs	32,427	778	2,352	35,557
(303,600 available)	10.7%	0.3%	0.8%	11.7%
Memory LUTs	1,310	35	179	1524
(130,800 available)	1%	$\simeq$ 0%	0.1%	1.2%
FFs	24,882	724	2,570	28,176
(607,200 available)	4.1%	0.1%	0.4%	4.6%
Block RAMs	163	13	2	178
(2,600 available)	7.9%	0.6%	0.1%	8.6%
DSP48 Blocks	214	6	12	232
(2,800 available)	7.6%	0.2%	0.4%	8.3%

Fourier series is designed and implemented. Fig. 5.19 illustrates the organization of the basic arithmetic operations and the storage elements for the realization of the EMF module. Fig. 5.19(a) shows how pipelining is employed to calculate EMF factors of 25 space harmonics. It can be observed from Fig. 5.19(b) that the sinusoidal, and exponential functions as well as add, multiply, and divide operations are utilized to compute EMF coefficient factor (CoEMF) for each harmonic and registers are inserted into data paths to synchronize data flow in the digital hardware design. The data marches through the `CoEMF` submodule at every clock cycle and takes the advantages of high computational data throughput.

In this architecture, the `Sigma` ($\sum$) submodules, which are used to accumulate the harmonic contents of EMF waveforms, are arranged in a parallel scheme to compute induced open-circuit voltages of three-phase windings simultaneously. It should be noted that the accumulation in the `Sigma` submodule is carried out in the fixed-point format. The reason is that floating-point accumulator has a longer latency for high frequency computation clock cycles and needs more logic resources to implement. The fixed-point accumulation is in the format of 40.100 (1 sign bit, 39 integer, and 100 fractional bits) that can guarantee both the range and precision. In this structure, the input data is accumulated over each clock cycle and then converted to floating-point format. Therefore, the `Sigma` submodule contains one floating-to-fixed point converter (`FL2FIX`), one fixed-point accumulator, and one fixed-to-floating point converter (`FIX2FL`) for the hardware realization.

5.6 FPGA-Based Real-Time Emulation and FEM Validation

5.6.1 Hardware Resource Utilization and Latencies

In this work, the real-time analytical space harmonic model of the PMLSM is targeted to the Xilinx® Virtex-7 development board. Fig. 5.20 presents the final hardware architecture implemented on the chip. It depicts the compactness of the designed circuit for the digital

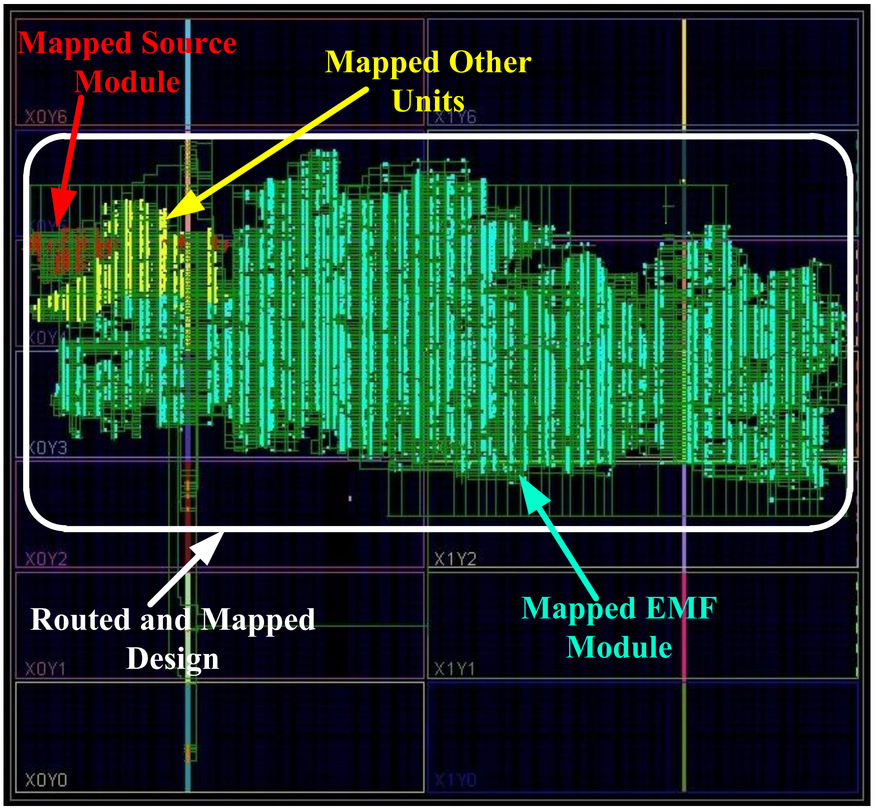


Figure 5.20: Implemented design of PMLSM on the FPGA chip.

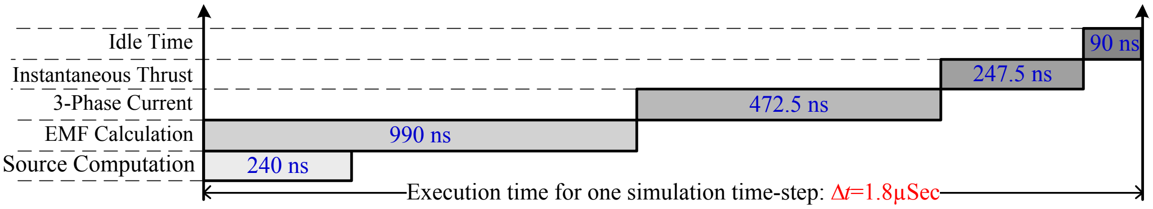


Figure 5.21: Detailed execution time for one simulation time-step of analyzed PMLSM.

hardware realization of PMSM, and also shows the distribution of each hardware unit on FPGA. The yellow, purple, and blue (light gray, gray, and dark gray in grayscale view) dots represent the area of the mapped design of the EMF module, the Source module, and other units, respectively. Table 5.4 lists the resource utilization of FPGA-based real-time emulator. As can be seen from this table, the EMF module consumes more logic resources compared with other parts in the emulator. The full design consumes about 12% and 8% of logic LUTs and DSP48 blocks, respectively. Such resource conservation is a direct consequence of exploitation of deep pipelining in the EMF model computation.

According to the best-case achievable FPGA clock frequency in the static timing analysis obtained at the end of the place and route process of hardware design, a clock frequency

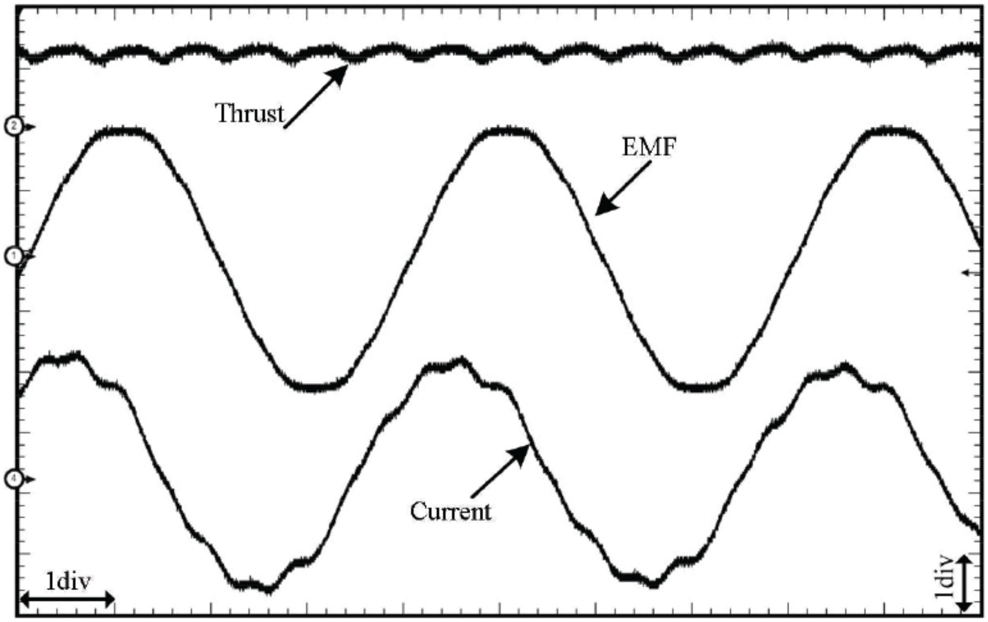


Figure 5.22: Real-time Oscilloscope traces of PMLSM performances at synchronous speed. [Time: 10ms/div, Thrust: 11N/div, EMF: 14.2V/div, and Current: 0.7A/div].

of 133.33MHz was chosen for the implementation. Thus, the FPGA based real-time emulation of the analytical model of PMLSM was executed within a time-step of 1.8μs. The detailed execution time for one simulation time-step is depicted in Fig. 5.21. It can be seen that the 90ns idle time at the end of each time-step can be used for sending the real-time data out of the FPGA based real-time emulator for external interfacing.

The real-time oscilloscope traces of the PMLSM behavior are illustrated in the Fig. 5.22. The motor is fed with the three-phase voltage supply at synchronous speed. The thrust force, EMF, and current waveforms are depicted in this figure, as will be explained with further details in the next section.

5.6.2 Comparison Between the Real-Time Emulated and Finite-Element Calculated Machine behavior

In this section, finite-element analysis is employed to evaluate the performance of FPGA-based real-time emulated PMLSM.

Fig. 5.23 shows a graphical representation of finite-element predicted flux lines and flux density distribution of the analyzed machine in one pole pitch obtained from JMAG$^{®}$ software. The execution time for running 80ms (2 cycles) of finite-element simulation even with time-step size of 50μs for only half of one pole pitch of the PMLSM by using the natural and anti-periodic boundary conditions with 5520 elements and 3052 nodes is 8min and 22s on a PC with Intel$^{®}$ Core^{TM} i7-2600 CPU at 3.4GHz and 8GB of installed memory (RAM), while the proposed analytical method was run in real-time with a 1.8μs time-step.

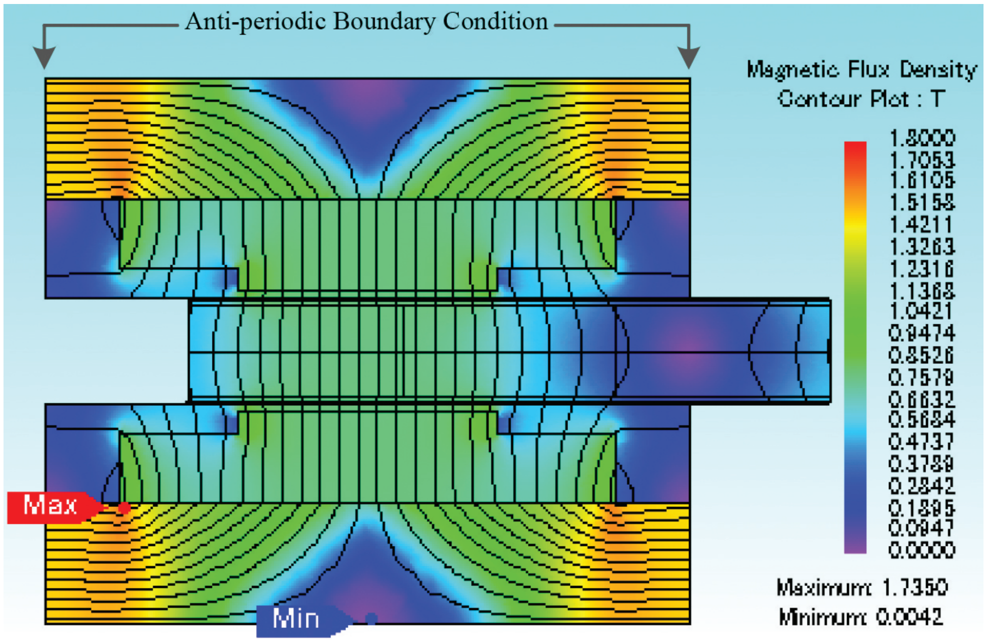


Figure 5.23: Flux lines and flux density distribution in one pole pitch of the PMLSM with stair-step shaped poles.

Although real-time emulation is carried out in a shorter time-step size compared with finite-element analysis in this work, it does not influence significantly on the differences between two sets of results.

In order to provide a detailed comparison, the real-time digital simulation data are exported from the FPGA using ChipScope® Analyzer and laid over the finite-element solutions.

Fig. 5.24 compares the real-time analytically and numerically predicted EMF waveforms of one phase of armature winding at a synchronous speed of 2.1 m/s. It can be seen that both sets of results agree quite well. A comparison of the FPGA-based real-time analytically predicted and finite-element calculated armature current resulting from the balanced three-phase sinusoidal voltage source is also presented in Fig. 5.24. A good agreement is again obtained in both the magnitude and wave shape. One reason for the slight discrepancy can be attributed to a finite and limited number of space harmonics (25 sine waves) which were computed and combined to represent the machine performance in the digital real-time emulation in this case. Moreover, the analytical model was developed based on some simplifications such as ignoring the saturation effects in the iron cores, and assuming a linear behavior for the machine model. The single-precision number calculation was used for hardware implementation due mainly to very fast computational requirement in real-time, while double-precision is employed in the finite-element analysis.

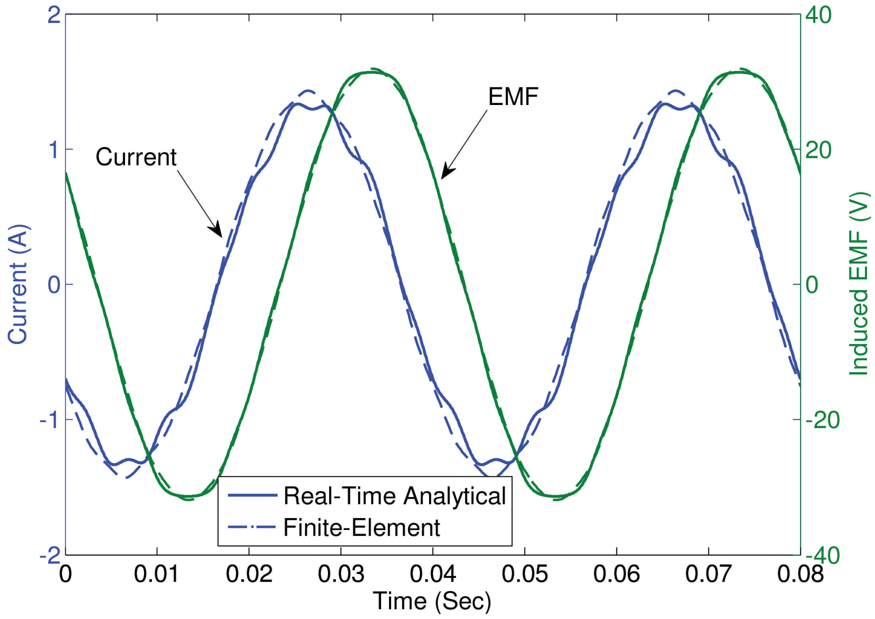


Figure 5.24: Armature current waveform and induced EMF at synchronous speed.

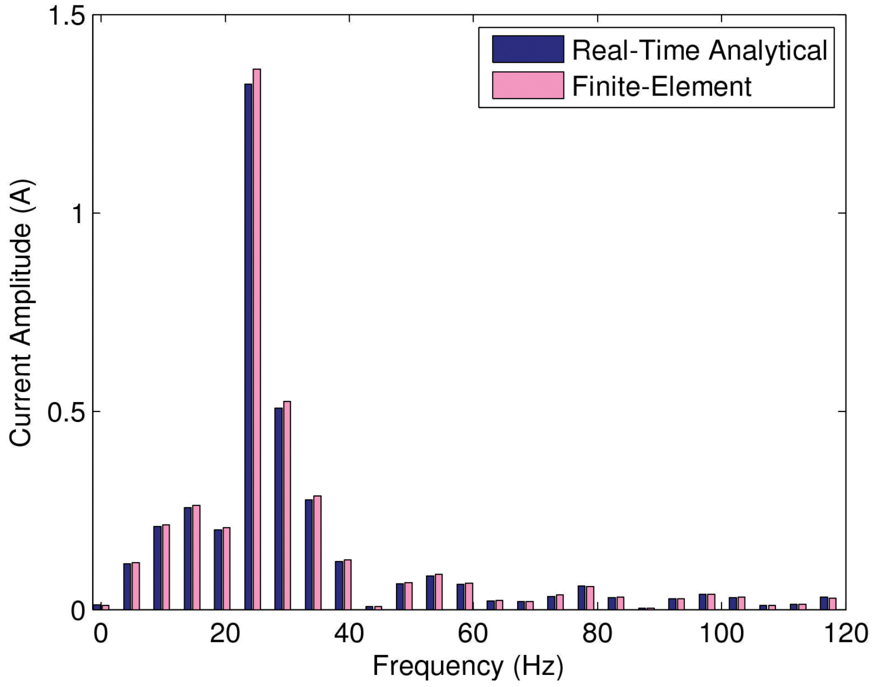


Figure 5.25: Harmonic contents of the steady-state current.

Fig. 5.25 depicts the harmonic contents of the current waveform obtained from real-time analytical and finite-element model. It can be observed that there is a close agreement between the results especially for the dominant harmonics.

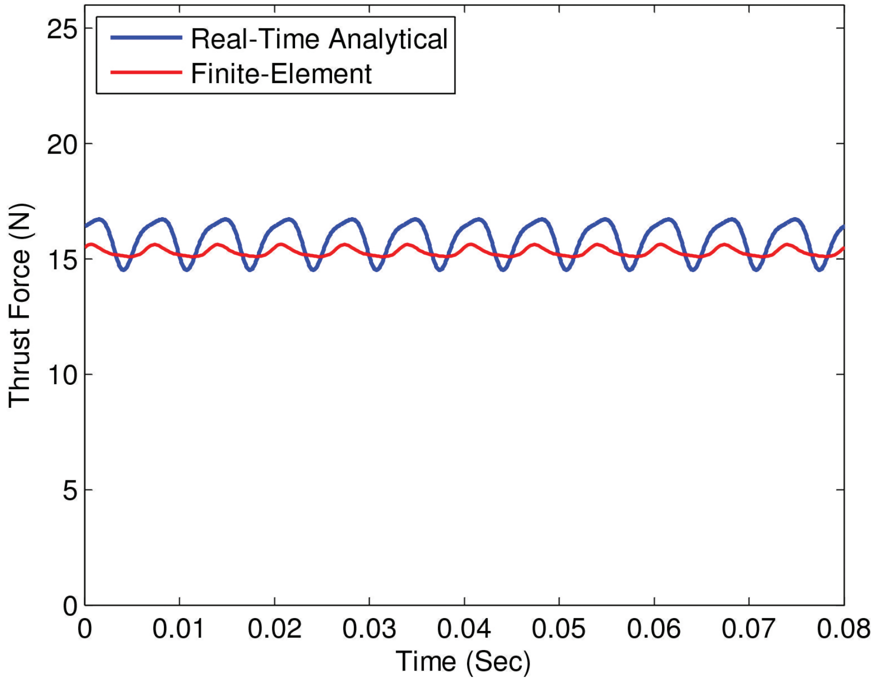


Figure 5.26: Instantaneous thrust force at synchronous speed of PMLSM.

Furthermore, the instantaneous thrust force waveform is presented in the Fig. 5.26. The FPGA real-time emulated result tracks the FEM solution closely. The fluctuations in the real-time prediction is a consequence of the distortions on the analytically computed current waveform.

5.7 Summary

This chapter proposed a new structure for enhancing the performance of PMLSMs. An analytical analysis has been presented to model the air-gap flux density distribution, EMF, and thrust in the PMLSM with stair-step shaped magnetic poles. The analytically calculated results have been evaluated using FEM. The close agreement between the results of two methods confirms the effectiveness of the presented model. Thus, there is no need to apply the time-consuming FEM for design optimization. The pole shape optimization for two PMLSMs with two types of stair-step shaped magnetic poles has been then carried out using the analytical model and genetic algorithm. Finite-element analysis results show that the optimization procedure significantly reduces the flux density as well as the EMF harmonics.

Moreover, real-time digital emulators of electrical machines can play a key role in the industrial applications by enabling fast design optimization procedure and rapid testing of new technologies in drive systems in a HIL scenario. They must however be based on a detailed machine model simulated with significantly small time-step and capable of predicting the machine performances as a function of machine geometry. This study proposes an FPGA-based real-time analytical space harmonic model of PM machines with shaped magnetic poles for HIL applications. Analytically derived formulas allow the prediction of machine performance in closed form as a function of physical dimensions and parameters of the machine. They facilitate the characterization of machine topologies, sensitivity analysis, iterative design optimization procedure, and real-time dynamic modeling of the machine. However, although finite-element analysis determines the magnetic field very precisely, with due account of saturation, *etc.*, it remains time-consuming. Furthermore, while the presented case study in this work addresses a linear PM machine, the model is significantly generalized for application to several types of rotary PM machines with shaped magnetic poles. All possible output variables of real-time emulated PMSM are presented and obtained by feeding the machine with the voltage sources similar to the real machine. Thus, the emulated PMSM can respond to any circumstances and changes that can be imposed by external devices and apparatus in the real-time HIL test. Taking advantages of the inherent parallel architecture of FPGA, the proposed hardware design is paralleled and pipelined to achieve an efficient FPGA implementation. Finite-element method is then employed to evaluate the analytical solution of the machine model from the real-time emulator. A close agreement between real-time analytically calculated and numerically predicted machine performances confirms the effectiveness and accuracy of the proposed approach.

6
Conclusions and Future Works

With the large increase in the electrical machine population in the industry in recent years, their design, test, and simulation are becoming critical and interesting.

As an advanced design and test method, HIL simulation allows the new prototype of an electrical machine or apparatus to be evaluated in a virtual environment under a wide range of realistic conditions repeatedly, safely, and economically. This work attempted to provide a framework for real-time emulation of electrical machines for HIL configuration.

Due to the fast development in digital VLSI technology, FPGAs are increasingly being used as a core computational engine in real-time HIL simulator. FPGAs offer inherent parallel architecture, high clock speed, and high logic resource capacity, enabling parallel computation technique in a very short period of time for a complicated and complex algorithm.

A digital hardware emulation of electrical machine are presented in this thesis with different level of accuracy and details on FPGA. The summary and future works are outline in this chapter.

6.1 Contributions of This Thesis

The main contributions of this thesis can be summarized as follows:

- A unified framework is proposed for FPGA-based real-time emulation of *qd* model of electrical machines. State-space approach is employed to represent the mathematical model of the machines. The step-by-step procedure of digital hardware realization of state-space model is provided in details on FPGA platform. Thus, the presented technique can be not only applied to the emulation of electrical machines, but also used for the real-time modeling of any system that can be described in terms of state-space equations.

- As the seamless and user friendly schematic or model-based method within the Matlab/Simulink environment is the best technique for highly parallel implementation of a system on FPGA using fixed-point operations, and textual programming language method such as VHDL or Verilog HDL is the most appropriate method for the pipeline realization of floating-point computation based real-time system, resulting in an area-optimized hardware architecture accepted for the industrial application, both these approaches are employed to model electrical machine model in real-time for the sake of a complete, and comprehensive comparison in term of accuracy, time-step size, hardware resource utilization.

- The developed general framework of FPGA-based real-time simulation of *qd* model of electrical machine is validated with the experimental measurements to demonstrate its usefulness for digital HIL tests.

- A new magnetic equivalent circuit model is proposed for rotary induction machine. It remains the salient features of available MEC models and meets the computation time constraints of real-time simulation. The governing nonlinear equations of developed MEC model are established and difficulties that exist in solving the equations in a real-time frame are explained.

- The concepts of antiperiodicity and moving-band technique that are used in finite element analysis are employed to reduce the domain of study and help the emulator to finish the required computation of nonlinear equations of MEC model within the simulation time-step.

- An iterative under-relaxed Newton Raphson method is implemented to solve the nonlinear equations of MEC model, although the realization of iterative algorithm is challenging task in nature on inherently parallel architecture of FPGA. A hardware module is developed for nonlinear systems including nonlinear evaluator unit, parallel residual vector calculator module, and linear solver submodule. Such a hardware module makes engineers able to solve a large dimensional nonlinear system within a few hundred microsecond on FPGA.

- A sinusoidal excitation test on induction machine and finite-element solution by JMAG® software are carried out to verify the performance of real-time MEC model in this work.

- As the magnetic pole shape affects the PM machine performance substantially, a design optimization method is proposed to shape the air-gap flux density distribution produced by poles to be as close to a sine waveform as possible for the reduction of thrust ripple and the increase of motor controllability. A pole shape optimization for two PMLSMs with two types of S^3MPs is performed using the analytical model and genetic algorithm. Finite-element analysis results show that the optimization procedure significantly reduces the flux density harmonics as well as the EMF harmonics.

- A real-time FPGA-based emulation of an analytical space harmonic model of a permanent magnet synchronous machine with shaped poles is proposed. The dynamic performance of the PMSM is developed based on analytical method by solving Maxwell equations and applying superposition theorem. The derived analytical formulas based on Fourier series are implemented on FPGA in a pipeline and parallel paradigm for an efficient hardware realization.

- The proposed real-time model of PMSM in HIL scenario is an effective means of evaluating the machine behavior as well as drive systems, and will aid the design process when addressing a given performance specification in a short time.

- In this work, the real-time emulation of analytical model of PMSM is validated by finite element calculations to confirm its utility and accuracy.

6.2 Directions for Future Work

The following topics are proposed for future work:

- With the advances in FPGA technology, real-time data processing is now available on cost-effective platforms to monitor the motor performance in abnormal and faulty condition in real-time. Therefore, FPGA-based real-time fault diagnosis of electrical machines can be considered as a hot topic for future work.

- Real-time estimation of time-varying parameters and variables can be implemented on FPGA. This procedure can be useful for self commissioning drive systems in a wide range of applications. Moreover, another possible utilization can be for the detection of failure.

- Future research can also be done to develop real-time MEC model on FPGA for other types of electromagnetic devices such as synchronous reluctance machines, switched reluctance motors, electromagnetic actuators, transformers, *etc*. The basic method and numerical techniques and required simplifications are fully explained and described in details in this thesis.

- 3-D MEC model of electrical machines in real-time on FPGA needs to be studied in future. Such a model can include 3-D effects inside the machine such as fringing fluxes in both ends of machine, and rotor bar skew effects. Hence, 3-D model can predict the machine performance with higher precision and duplicate its behavior with higher accuracy.

- Similar analytical method can be developed for the switched reluctance motor for design optimization procedure and real-time simulation.

- The developed real-time machine models of this thesis can be accompanied with the interfaces and amplifiers to make a HIL configuration setup, and test the virtual machine against an actual drive system or protection devices in a close loop environment. Such closed loop results in HIL scenario can be compared with the experimental results obtained from the interaction of actual machine and drive system in a closed loop to demonstrate the effectiveness of FPGA-based real-time HIL tests.

Bibliography

[1] A. Boglietti, A. M. El-Refaie, O. Drubel, A. M. Omekanda, N. Bianchi, E.B. Agamloh, M. Popescu, A. Di Gerlando, and J. B. Bartolo, "Electrical machine topologies: hottest topics in the electrical machine research community", *IEEE Industrial Electronics Magazine,* vol. 8, no. 2, pp. 18-30, June 2014.

[2] P. Lok-Fu, M. O. Faruque, Nie Xin, and V. Dinavahi, "A versatile cluster-based real-time digital simulator for power engineering research", *IEEE Trans. on Power Systems,* vol. 21, no. 2, pp. 455-465, May 2006.

[3] E. Monmasson, and M. N. Cirstea, "FPGA design methodology for industrial control systems - a review", *IEEE Trans. on Industrial Electronics,* vol. 54, no. 4, pp. 1824-1842, August 2007.

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

[5] M. O. Faruque, and V. Dinavahi, "Hardware-in-the-loop simulation of power electronic systems using adaptive discretization", *IEEE Trans. on Industrial Electronics,* vol. 57, no. 4, pp. 1146-1158, April 2010.

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

[7] P. C. Krause, O. Wasynczuk, and S. Sudhoff, *Analysis of Electric Machinery and Drive Systems,* New York: IEEE Press, 1994.

[8] V. Ostovic, "A novel method for evaluation of transient states in saturated electric machines", *IEEE Trans. on Power Delivery,* vol. 25, no. 1, pp. 96-100, January/Febryuary 1989.

[9] M. A. Rahman, T. Little, and G. R. Slemon, "Analytical models for interior-type permanent magnet synchronous motors", *IEEE Trans. on Magnetics,* vol. 21, no. 5, pp. 1741-1743, September 1985.

[10] M. Preston, and J. P. Lyons, "A switched reluctance motor model with mutual coupling and multi-phase excitation", *IEEE Trans. on Magnetics,* vol. 27, no. 6, pp. 5423-5425, November 1991.

[11] J. Wang, G. W. Jewell, and D. Howe, "A general framework for the analysis and design of tubular linear permanent magnet machines", *IEEE Trans. on Magnetics,* vol. 35, no. 3, pp. 1986-2000, May 1999.

[12] Z. J. Liu, A. Vourdas, and K. J. Binns, "Magnetic field and eddy current losses in linear and rotating permanent magnet machines with a large number of poles", *IEE Procedings, Science, Measurement and Technology* vol. 138, no. 6, pp. 289-294, November 1991.

[13] D. L. Trumper, W. Kim, and M. E. Williams, "Design and analysis framework for linear permanent magnet machines", *IEEE Trans. on Industry Applications,* vol. 32, no. 2, pp. 371-379, March/April 1996.

[14] N. Bianchi, *Electrical Machine Analysis Using Finite Elements,* Taylor & Francis Group; CRC Press, 2005.

[15] G. G. Parma, and V. Dinavahi, "Real-time digital hardware simulation of power electronics and drives", *IEEE Trans. on Power Delivery,* vol. 22, no. 2, pp. 1-11, April 2007.

[16] C. Hao, S. Song, D.C. Aliprantis, and J. Zambreno, "Dynamic simulation of electric machines on FPGA boards", *Int. IEEE Conf. Electric Machines and Drives,* pp. 1523-1528, 2009.

[17] R. Gregor, G. Valenzano, J. Rodriguez-Pineiro, and J. Rodas, "FPGA-based real-time simulation of a dual three-phase induction machine", *Int. IEEE Conf. Power Electronics and Applications (EPE'14-ECCE Europe),* pp. 1-8, August 2009.

[18] S. Tola, and M. Sengupta, "Real-time simulation of an induction motor in different reference frames on a FPGA platform", *Int. IEEE Conf. Power Electronics, Drives and Energy Systems (PEDES),* pp. 1-6, December 2012.

[19] L. Herrera, and J. Wang, "DFPGA based detailed real-time simulation of power converters and electric machines for EV HIL applications", *IEEE Conf. Energy Conversion Congress and Exposition (ECCE),* pp. 1756-1764, September 2013.

[20] M. Dagbagi, L. Charaabi, L. Idkhajine, E. Monmasson, and I. Slama-Belkhodja, "Digital implementation using delta operator for FPGA-based induction motor emulator", *Int. IEEE Conf. Systems, Signals and Devices (SSD),* pp. 1-6, March 2011.

[21] M. Dagbagi, L. Idkhajine, E. Monmasson, L. Charaabi, and I. Slama-Belkhodja, "Digital implementation using delta operator for FPGA-based induction motor emulator", *Int. IEEE Symposium on Industrial Electronics (ISIE),* pp. 1581-1586, June 2011.

[22] S. Abourida, C. Dufour, J. Belanger, T. Yamada, and T. Arasawa, "Hardware-in-the-loop simulation of finite-element based motor drives with RT-LAB and JMAG", *Int. IEEE Symposium on. Indastrial Electronics,* pp. 2462-2466, 2006.

[23] C. Dufour, J. Belanger, S. Abourida, and V. Lapointe, "FPGA-based real-time simulation of finite-element analysis permanent magnet synchronous machine drives", *Int. IEEE Conf. Power Electronics Specialists,* pp. 909-915, 2007.

[24] C. Dufour, J. Belanger, S. Abourida, and V. Lapointe, "Real-time simulation of finite-element analysis permanent magnet synchronous machine drives on a FPGA card", *Int. IEEE Conf. Power Electronics and Applications,* pp. 1-10, 2007.

[25] C. Dufour, J. Belanger, V. Lapointe, and S. Abourida, "Real-time simulation on FPGA of a permanent magnet synchronous machine drive using a finite-element based model", *Int. IEEE Symposium on. Power Electronics, Electrical Drives, and Automation and Motion,* pp. 19-25, 2008.

[26] C. Dufour, J. Belanger, V. Lapointe, and S. Abourida, "Real-time simulation of permanent magnet motor drive on FPGA chip for high-bandwidth controller tests and validation", *Int. IEEE Conf. Industrial Electronics,* pp. 4581-4586, 2006.

[27] C. Dufour, J. Belanger, and V. Lapointe, "FPGA-based ultra-low latency HIL fault testing of a permanent magnet motor drive using RT-LAB-XSG", *Int. IEEE Conf. Power System Technology,* pp. 1-7, 2008.

[28] G. Jin, S. Shiyang, H. Yang, and H. Surong, "Implementation and test for the semi-physical real-time simulation of IPMSM based on 3-D inductance table", *Int. IEEE Conf. Transportation Electrification Asia-Pacific,* pp. 1-5, September 2014.

[29] G. Jin, H. Yang, S. Shiyang, and Huang Surong, "The semi-physical real-time simulation and test of IPMSM considering the iron loss effect", *Int. IEEE Conf. Transportation Electrification Asia-Pacific,* pp. 1-5, September 2014.

[30] M. Matar, and R. Iravani, "Massively parallel implementation of AC machine models for FPGA-based real-time simulation of electromagnetic transients", *IEEE Trans. Power Delivery,* vol. 26, no. 2, pp. 830-840, April 2011.

[31] T. O. Bachir, J. P. David, C. Dufour,and J. Blanger, "Effective FPGA-based electric motor modeling with floating-point cores", *Int. IEEE Conf. Ind. Electronics Society,* pp. 829-836, 2010.

[32] T. O. Bachir, and J. P. David,; "FPGA-based real-time simulation of state-space models using floating-point cores", *Int. IEEE Conf. Power Electronics and Motion Control,* pp. 26-31, 2010.

[33] T. Schulte, and J. Bracker, "Real-time simulation of BLDC motors for hardware-in-theloop applications incorporating sensorless control", *Int. IEEE Conf. Industrial Electronics,* pp. 2195-2200, 2008.

[34] T. Hanamoto, J. Yano, H. Ikeda, and T. Tsuji, "Hardware real time simulator of synchronous reluctance motor including three phase PWM inverter model", *Int. IEEE Conf. Power Electronics,* pp. 2005-2009, 2010.

[35] P. A. Rajne, and V. Ramanarayanan, "Programming an FPGA to emulate the dynamics of DC machines", *Int. IEEE Conf. Power Electronics,* pp. 120-124, 2006.

[36] K. Jayalakshmi, and V. Ramanarayanan, "Real-time simulation of electrical machines on FPGA platform", *Int. IEEE Conf. Power Electronics,* pp. 259-263, 2006.

[37] C. Dufour, J. N. Paquin, H. Blanchette, and J. Belanger, "Specifications for real-time simulation of switched reluctance drives using microprocessors and FPGAs as computational engines", *Int. IEEE Conf. Electric Machines and Drives,* pp. 750-754, 2009.

[38] H. Saad, T. Ould-Bachir, J. Mahseredjian, C. Dufour, S. Dennetiere, and S. Nguefeu, "Real-time simulation of MMCs Using CPU and FPGA", *IEEE Trans. on Power Electronics,* vol. 30, no. 1, pp. 259-267, January 2015.

[39] H. F. Blanchette, T. Ould-Bachir, and J. P. David, "A state-space modeling approach for the FPGA-based real-time simulation of high switching frequency power converters", *IEEE Trans. on Industrial Electronics,* vol. 59, no. 12, pp. 4555-4567, December 2012.

[40] W. Wang, Z. Shen, and V. Dinavahi, "Physics-based device-level power electronic circuit hardware emulation on FPGA", *IEEE Trans. on Industrial Electronics,* vol. 10, no. 4, pp. 2166-2179, November 2014.

[41] A. Myaing, V. Dinavahi, "FPGA-based real-time emulation of power electronic systems with detailed representation of device characteristics", *IEEE Trans. on Industrial Electronics,* vol. 58, no. 1, pp. 358-368, January 2011.

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

[43] M. W. Naouar, E. Monmasson, A. A. Naassani, and I. Slama-Belkhodja, "FPGA-based dynamic reconfiguration of sliding mode current controllers for synchronous machines", *IEEE Trans. on Industrial Informatics,* vol. 9, no. 3, pp. 1262-2171, August 2013.

[44] H. Berriri, W. Naouar, I. Bahri, I. Slama-Belkhodja, and E. Monmasson, "Field programmable gate array-based fault-tolerant hysteresis current control for AC machine drives", *IET. Electric Power Applications,* vol. 6, no. 3, pp. 181-189, March 2012.

[45] L. Idkhajine, E. Monmasson, and A. Maalouf, "Fully FPGA-based sensorless Control for synchronous AC drive using an extended Kalman filter", *IEEE Trans. Industrial Electronics,* vol. 59, no. 10, pp. 3908-3918, October 2012.

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

[47] Y. Chen, and V. Dinavahi, "Hardware emulation building blocks for real-time simulation of large-scale power grids", *IEEE Trans. on Industrial Informatics,* vol. 10, no. 1, pp. 373-381, February 2014.

[48] Y. Chen, and V. Dinavahi, "Multi-FPGA digital hardware design for detailed large-scale real-time electromagnetic transient simulation of power systems", *IET. Generation, Transmission & Distribution,* vol. 7, no. 5, pp. 451-463, May 2013.

[49] S. Kilts, *Advanced FPGA Design: Architecture, Implementation, and Optimization,* Wiley-IEEE Press, 2007

[50] D. Bailey, *Design for Embedded Image Processing on FPGAs,* Wiley-IEEE Press, 2011

[51] E. Monmasson, and M. N. Cirstea, "FPGA design methodology for industrial control systemsa review", *IEEE Trans. on Industrial Electronics,* vol. 54, no. 4, pp. 1824-1842, August 2007.

[52] *7 Series FPGAs Configurable Logic Block,* Xilinx User Guide, November 2014

[53] *7 Series FPGAs Memory Resources,* Xilinx User Guide, November 2014

[54] *7 Series DSP48E1 Slice,* Xilinx User Guide, November 2014

[55] M. Steurer, C. S. Edrington, M. Sloderbeck, W. Ren, and J. Langston, "A megawatt-scale power hardware-in-the-loop simulation setup for motor drives," *IEEE Trans. on Industrial Electronics*, vol. 57, no. 4, pp. 1254-1260, April 2010.

[56] S. C. Oh, "Evaluation of motor characteristics for hybrid electric vehicles using the hardware-in-the-loop concept," *IEEE Trans. on Vehicular Technology*, vol. 54, no. 3, pp. 817-824, May 2005.

[57] B. Lu, X. Wu, H. Figueroa, and A. Monti, "A low-cost real-time hardware-in-the-loop testing approach of power electronics controls," *IEEE Trans. on Industrial Electronics*, vol. 54, no. 2, pp. 919-931, April 2007.

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

[59] G. A. Korn, and T. M. Korn, *Mathematical Handbook for Scientists and Engineers: Definitions, Theorems, and Formulas for Reference and Review.*, McGraw Hill. Inc, 1961.

[60] Y. Chen, "Large-scale real-time electromagnetic transient simulation of power systems using hardware emulation on FPGAs," Ph.D. dissertation, Dept. Elect. Comp. Eng., Univ. Alberta, Edmonton, AB, Canada, 2012.

[61] *Xilinx System Generator,* Xilinx User Guide, October 2012

[62] I. Munteanu, AI. Bratcu, S. Bacha, D. Roye, and J. Guiraud, "Hardware-in-the-loop-based simulator for a class of variable-speed wind energy conversion systems: design and performance assessment," *IEEE Trans. on Energy Conversion*, vol. 25, no. 2, pp. 564-576, June 2010.

[63] Fei Gao, B. Blunier, M. G. Simo es, and A. Miraoui, "PEM fuel cell stack modeling for real-time emulation in hardware-in-the-loop applications," *IEEE Trans. on Energy Conversion*, vol. 26, no. 1, pp. 184-194, March 2011.

[64] O. Vodyakho, M. Steurer, C. S. Edrington, and F. Fleming, "An induction machine emulator for high-power applications utilizing advanced simulation tools With graphical user interfaces," *IEEE Trans. on Energy Conversion*, vol. 27, no. 1, pp. 160-172, March 2012.

[65] D. N. Dyck, T. Rahman, and C. Dufour, "Internally consistent nonlinear behavioral model of a PM synchronous machine for hardware-in-the-loop simulation," *IEEE Trans. on Magnetics*, vol. 50, no. 2, pp. 853-856, February 2014.

[66] P. C. Krause, O. Wasynczuk, and S. Sudhoff, *Analysis of Electric Machinery and Drive Systems*, New York: IEEE Press, 1994.

[67] H. Gorginpour, H. Oraee, and R. A. McMahon, "A novel modeling approach for design studies of brushless doubly fed induction generator based on magnetic equivalent circuit," *IEEE Trans. Energy Convers.*, vol. 28, no. 4, pp. 902-912, December 2013.

[68] M. Amrhein, and P. T. Krein, "Force calculation in 3-D magnetic equivalent circuit networks with a Maxwell stress tensor," *IEEE Trans. Energy Convers.*, vol. 24, no. 3, pp. 587-593, September 2009.

[69] T. A. Lipo, *Introduction to AC Machine Design*, vol. 1, 1996, Power Electronics Research Center, University of Wisconsin.

[70] S. R H. Hoole, A. Mascrenghe, K. Navukkarasu, and K. Sivasubramaniam, "An expert design environment for electrical devices and its engineering assistant," *IEEE Trans. on Magnetics*, vol. 39, no. 3, pp. 1693-1696, May 2003.

[71] V. Ostovic, *Dynamics of Saturated Electric Machines*, New York: Springer-Verlag, 1989.

[72] C. Delforge, and B. Lemaire-Semail, "Induction machine modeling using finite element and permeance network methods," *IEEE Trans. on Magnetics*, vol. 31, no. 3, pp. 2092-2095, May 1995.

[73] H. A. Toliyat, T. A. Lipo, "Transient analysis of cage induction machines under stator, rotor bar and end ring faults," *IEEE Trans. on Energy Conversion*, vol. 10, no. 2, pp. 241-247, June 1995.

[74] J. Perho, *Reluctance network for analysing induction machines*, ActaPolytech. Scand., Electr. Eng. Series, vol. 110, pp. 1-147, Dec. 2002.

[75] H. W. Derbas, J. M. Williams, A. C. Koenig, and S. D. Pekarek, "A comparison of nodal- and mesh-based magnetic equivalent circuit models," *IEEE Trans. Energy Convers.*, vol. 24, no. 2, pp. 388-396, June 2009.

[76] S. D. Sudhoff, B. T. Kuhn, K. A. Corzine, and B. T. Branecky, "Magnetic equivalent circuit modeling of induction motors," *IEEE Trans. on Energy Conversion*, vol. 22, no. 2, pp. 259-270, June 2007.

[77] M. Amrhein, and P. T. Krein, "3-D magnetic equivalent circuit framework for modeling electromechanical devices," *IEEE Trans. on Energy Conversion*, vol. 24, no. 2, pp. 397-405, June 2009.

[78] M. L. Bash, J. M. Williams, and S. D. Pekarek, "Incorporating motion in mesh-based magnetic equivalent circuits," *IEEE Trans. on Energy Conversion*, vol. 25, no. 2, pp. 329-338, June 2010.

[79] M. Amrhein, and P. T. Krein, "Induction machine modeling approach based on 3-D magnetic equivalent circuit framework," *IEEE Trans. on Energy Conversion*, vol. 25, no. 2, pp. 339-347, June 2010.

[80] S. D. Sudhoff, G. M. Shane, and H. Suryanarayana, "Magnetic-equivalent-circuit-based scaling laws for low-frequency magnetic devices," *IEEE Trans. on Energy Conversion*, vol. 28, no. 3, pp. 746-755, September 2013.

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

[82] C. Neagoe, and F. Ossart, "Analysis of convergence in nonlinear magnetostatic finite elements problems," *IEEE Trans. Magn.*, vol. 30, no. 5, pp. 2865-2868, September 1994.

[83] N. R. Tavana, and V. Dinavahi, “A general framework for FPGA-based real-time emulation of electrical machines for HIL applications,” *IEEE Trans. on Industrial Electronics*, vol. 62, no. 4, pp. 2041-2053, April 2015.

[84] Z. Q. Zhu, and D. Howe, “Electrical machines and drives for electric, hybrid, and fuel cell vehicles,” *Proc. IEEE*, vol. 95, no. 4, pp. 746-765, April 2007.

[85] A. M. El-Refaie, “Motors/generators for traction/propulsion applications: A review,” *IEEE Vehicular Technology Magazine*, vol. 8, no. 1, pp. 90-99, February 2013.

[86] M. F. Hsieh, and Y. S. Hsu, “An investigation on influence of magnet arc shaping upon back electromotive force waveforms for design of permanent-magnet brushless motors,” *IEEE Trans. on Magnetics*, vol. 41, no. 10, pp. 3949-3951, October 2005.

[87] Y. Li, J. Xing, T. Wang, and Y. Lu, “Programmable design of magnet shape for permanent magnet synchronous motors with sinusoidal back-EMF waveforms,” *IEEE Trans. on Magnetics*, vol. 44, no. 9, pp. 2163-2167, September 2008.

[88] Y. Li, J. Zou, and Y. Lu, “Optimum design of magnet shape in permanent-magnet synchronous motors,” *IEEE Trans. on Magnetics*, vol. 39, no. 6, pp. 3523-3526, November 2003.

[89] P. Zheng, J. Zhao, J. Han, J. Wang, Z. Yao, and R. Liu, “Optimization of the magnetic pole shape of a permanent-magnet synchronous motor,” *IEEE Trans. on Magnetics*, vol. 43, no. 6, pp. 2531-2533, June 2007.

[90] N. R. Tavana, and A. Shoulaie, “Analysis and design of magnetic pole shape in linear permanent-magnet machine,” *IEEE Trans. on Magnetics*, vol. 46, no. 4, pp. 1000-1006, April 2010.

[91] N. R. Tavana, and A. Shoulaie, “Pole-shape optimization of permanentmagnet linear synchronous motor for reduction of thrust ripple,” *Energy Conversion and Managment*, vol. 52, no. 1, pp. 349-354, January 2010.

[92] K. Min-Mo, J. Seok-Myeong, P. Yu-Seop, P. Hyung-Il, and C. Jang-Young, “Characteristic analysis of direct-drive wind power generator considering permanent magnet shape and skew effects to reduce torque ripple based on analytical approach,” *IEEE Trans. on Magnetics*, vol. 49, no. 7, pp. 3917-3920, July 2013.

[93] S. Chaithongsuk, N. Takorabet, and F. Meibody-Tabar, “On the use of pulse width modulation method for the elimination of flux density harmonics in the air gap of surface PM motors,” *IEEE Trans. on Magnetics*, vol. 45, no. 3, pp. 1736-1739, March 2009.

[94] A. H. Isfahani, “Analytical framework for thrust enhancement in permanent-magnet (PM) linear synchronous motors with segmented PM poles,” *IEEE Trans. on Magnetics*, vol. 46, no. 4, pp. 1116-1122, April 2010.

[95] A. H. Isfahani, S. Vaez-Zadeh, and M. A. Rahman, “Using modular poles for shape optimization of flux density distribution in permanent magnet machines,” *IEEE Trans. on Magnetics*, vol. 44, no. 8, pp. 2009-2015, August 2008.

[96] W. Lijian, and Z. Zi-Qiang, “Analytical modeling of surface-mounted PM machines accounting for magnet shaping and varied magnet property distribution,” *IEEE Trans. on Magnetics*, vol. 50, no. 7, pp. 1-11, July 2014.

[97] J. Seok-Myeong, P. Hyung-Il, C. Jang-Young, K. Kyoung-Jin, and L. Sung-Ho, “Magnet pole shape design of permanent magnet machine for minimization of torque ripple based on electromagnetic field theory,” *IEEE Trans. on Magnetics*, vol. 47, no. 10, pp. 3586-3589, October 2011.

[98] K. Wang, Z. Q. Zhu, and G. Ombach, “Torque enhancement of surface-mounted permanent magnet machine using third-order harmonic,” *IEEE Trans. on Magnetics*, vol. 50, no. 3, pp. 104-113, March 2014.

[99] N. R. Tavana, A. Shoulaie, and V. Dinavahi “Analytical modeling and design optimization of linear synchronous motor with stair-step-shaped magnetic poles for electromagnetic launch applications,” *IEEE Trans. Plasma Science*, vol. 40, no. 2, pp. 519-527, Febryuary 2012.

[100] J. Wang, G. W. Jewell, and D. Howe, “A general framework for the analysis and design of tubular linear permanent magnet machines,” *IEEE Trans. on Magnetics*, vol. 40, no. 2, pp. 1986-2000, May 1999.

[101] W. Ren, M. Steurer, and T. L. Baldwin, “Improve the stability and the accuracy of power hardware-in-the-loop simulation by selecting appropriate interface algorithms,” *IEEE Trans. Ind. Appl.*, vol. 44, no. 4, pp. 1286-1294, July-August 2008.

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

[103] Y. El-Kurdi, D. Giannacopoulos, and W. J. Gross, “Hardware acceleration for finite-element electromagnetics: efficient sparse matrix floating-point computations with FPGAs,” *IEEE Trans. on Magnetics*, vol. 43, no. 4, pp. 1525-1528, April 2007.

[104] W. Zhao, E. Belhaire, C. Chappert, and P. Mazoyer, “Power and area optimization for run-time reconfiguration system on programmable chip based on magnetic random access memory,” *IEEE Trans. on Magnetics*, vol. 45, no. 2, pp. 776-780, February 2009.

[105] Y. Cai, S. Jeon, K. Mai, and B. V. K. Vijaya Kumar, “Highly parallel FPGA emulation for LDPC error floor characterization in perpendicular magnetic recording channel,” *IEEE Trans. on Magnetics*, vol. 45, no. 10, pp. 3761-3764, October 2009.

[106] C. Dufour, and J. Belanger, “On the use of real-time simulation technology in smart grid research and development,” *IEEE Trans. Ind. Appl.*, vol. 50, no. 6, pp. 3963-3970, November-December 2014.

[107] S. Vaez-Zadeh, and A. H. Isfahani, “Multiobjective design optimization of air-core linear permanent-magnet synchronous motors for improved thrust and low magnet consumption,” *IEEE Trans. on Magnetics*, vol. 42, no. 3, pp. 446-452, March 2006.

[108] J. Wang, and D. Howe, “Design optimization of radially magnetized, iron-cored, tubular permanent-magnet machines and drive systems,” *IEEE Trans. on Magnetics*, vol. 40, no. 5, pp. 3262-3277, March 2006.

[109] H. Li, Z. Chen, and H. Polinder, “Optimization of multibrid permanent-magnet wind generator systems,” *IEEE Trans. Energy Convers.*, vol. 24, no. 1, pp. 82-92, March 2009.

[110] H. M. Hasanien, A. S. Abd-Rabou, and S. M. Sakr, “Design optimization of transverse flux linear motor for weight reduction and performance improvement using response surface methodology and genetic algorithms,” *IEEE Trans. Energy Convers.*, vol. 25, no. 3, pp. 598-605, September 2010.

[111] K. T. Chau, Q. Sun, Y. Fan, and M. Cheng, “Torque ripple minimization of doubly salient permanent-magnet motors,” *IEEE Trans. Energy Convers.*, vol. 20, no. 2, pp. 352-358, June 2005.

[112] *7 Series FPGAs Overview*, Xilinx Inc., Product Specification, Oct. 2014.

A

State-space matrices and parameters of electrical machines in Chapter 3

A.1 State-Space Matrices

Based on the following matrix notation, the state-space definitions for various machines are given below.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{C}_{11} & \mathbf{C}_{12} \\ \mathbf{C}_{21} & \mathbf{C}_{22} \end{bmatrix}.$$

A.1.1 Induction Machine

$$\mathbf{x} = \left[\begin{array}{cccc} \lambda_{qs} & \lambda_{ds} & \lambda_{qr} & \lambda_{dr} \end{array} \right]^T. \tag{A1}$$

$$\mathbf{u} = \left[\begin{array}{cccc} v_{qs} & v_{ds} & v_{qr} & v_{dr} \end{array} \right]^T. \tag{A2}$$

$$\mathbf{y} = \left[\begin{array}{cccc} i_{qs} & i_{ds} & i_{qr} & i_{dr} \end{array} \right]^T. \tag{A3}$$

$$\begin{aligned}
\mathbf{A}_{11} &= \left[\begin{array}{cc} \frac{r_s}{L_{ls}} \left(\frac{L_{aq}}{L_{ls}} - 1 \right) & -\omega \\ \omega & \frac{r_s}{L_{ls}} \left(\frac{L_{ad}}{L_{ls}} - 1 \right) \end{array} \right], \\
\mathbf{A}_{21} &= \left[\begin{array}{cc} \frac{r_r}{L_{lr}} \frac{L_{aq}}{L_{ls}} & 0 \\ 0 & \frac{r_r}{L_{lr}} \frac{L_{ad}}{L_{ls}} \end{array} \right], \ \mathbf{A}_{12} = \left[\begin{array}{cc} \frac{r_s}{L_{ls}} \frac{L_{aq}}{L_{lr}} & 0 \\ 0 & \frac{r_s}{L_{ls}} \frac{L_{ad}}{L_{lr}} \end{array} \right], \\
\mathbf{A}_{22} &= \left[\begin{array}{cc} \frac{r_r}{L_{lr}} \left(\frac{L_{aq}}{L_{lr}} - 1 \right) & \omega_r - \omega \\ \omega - \omega_r & \frac{r_r}{L_{lr}} \left(\frac{L_{ad}}{L_{lr}} - 1 \right) \end{array} \right].
\end{aligned} \tag{A4}$$

$$\mathbf{B}= eye(4,4). \tag{A5}$$

$$\mathbf{C}= \frac{1}{D}\begin{bmatrix} L_{rr} & 0 & -L_M & 0 \\ 0 & L_{rr} & 0 & -L_M \\ -L_M & 0 & L_{ss} & 0 \\ 0 & -L_M & 0 & L_{ss} \end{bmatrix}. \tag{A6}$$

$$\mathbf{D}= zeros(4,4). \tag{A7}$$

where

$$L_{ss}= L_{ls} + L_m,\;\; L_{rr}= L_{lr} + L_m,\;\; D= L_{ss}L_{rr} + L_m^2. \tag{A8}$$

$$L_{ad}^{-1} = L_{aq}^{-1}= 1/L_m + 1/L_{ls} + 1/L_{lr}. \tag{A9}$$

A.1.2 Synchronous Machine

$$\mathbf{x}= \begin{bmatrix} \lambda_{qs} & \lambda_{ds} & \lambda_{kq1} & \lambda_{kq2} & \lambda_{fd} & \lambda_{kd} \end{bmatrix}^T. \tag{A10}$$

$$\mathbf{u}= \begin{bmatrix} v_{qs} & v_{ds} & v_{kq1} & v_{kq2} & e_{xfd} & v_{kd} \end{bmatrix}^T. \tag{A11}$$

$$\mathbf{y}= \begin{bmatrix} i_{qs} & i_{ds} & i_{kq1} & i_{kq2} & i_{fd} & i_{kd} \end{bmatrix}^T. \tag{A12}$$

$$\mathbf{A}_{11} = \begin{bmatrix} \frac{r_s}{L_{ls}}\left(\frac{L_{aq}}{L_{ls}} - 1\right) & -\omega_r & \frac{r_s}{L_{ls}}\frac{L_{aq}}{L_{lkq1}} \\ \omega_r & \frac{r_s}{L_{ls}}\left(\frac{L_{ad}}{L_{ls}} - 1\right) & 0 \\ \frac{r_{kq1}}{L_{lkq1}}\frac{L_{aq}}{L_{ls}} & 0 & \frac{r_{kq1}}{L_{lkq1}}\left(\frac{L_{aq}}{L_{lkq1}} - 1\right) \end{bmatrix},$$

$$\mathbf{A}_{21} = \begin{bmatrix} \frac{r_{kq2}}{L_{lkq2}}\frac{L_{aq}}{L_{ls}} & 0 & \frac{r_{kq2}}{L_{lkq2}}\frac{L_{aq}}{L_{lkq1}} \\ 0 & \frac{r_{fd}}{L_{lfd}}\frac{L_{ad}}{L_{ls}} & 0 \\ 0 & \frac{r_{kd}}{L_{lkd}}\frac{L_{ad}}{L_{ls}} & 0 \end{bmatrix},$$

$$\mathbf{A}_{12} = \begin{bmatrix} \frac{r_s}{L_{ls}}\frac{L_{aq}}{L_{lkq2}} & 0 & 0 \\ 0 & \frac{r_s}{L_{ls}}\frac{L_{ad}}{L_{lfd}} & \frac{r_s}{L_{ls}}\frac{L_{ad}}{L_{lkd}} \\ \frac{r_{kq1}}{L_{lkq1}}\frac{L_{aq}}{L_{lkq2}} & 0 & 0 \end{bmatrix},$$

$$\mathbf{A}_{22} = \begin{bmatrix} \frac{r_{kq2}}{L_{lkq2}}\left(\frac{L_{aq}}{L_{lkq2}} - 1\right) & 0 & 0 \\ 0 & \frac{r_{fd}}{L_{lfd}}\left(\frac{L_{ad}}{L_{lfd}} - 1\right) & \frac{r_{fd}}{L_{lfd}}\frac{L_{ad}}{L_{lkd}} \\ 0 & \frac{r_{kd}}{L_{lkd}}\frac{L_{ad}}{L_{lfd}} & \frac{r_{kd}}{L_{lkd}}\left(\frac{L_{ad}}{L_{lkd}} - 1\right) \end{bmatrix}. \tag{A13}$$

$$\mathbf{B}_{11} = eye(3,3),\;\; \mathbf{B}_{12} = \mathbf{B}_{21} = zeros(3,3),\;\; \mathbf{B}_{22} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{r_{fd}}{L_{md}} & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A14}$$

$$\mathbf{C}_{11} = \begin{bmatrix} \frac{(L_{kq1}L_{kq2}-L_{mq}^2)}{Dq} & 0 & \frac{(-L_{mq}L_{kq2}+L_{mq}^2)}{Dq} \\ 0 & \frac{(L_{fd}L_{kd}-L_{md}^2)}{Dd} & 0 \\ \frac{(L_{mq}L_{kq2}-L_{mq}^2)}{Dq} & 0 & \frac{(-L_{mq}L_{kq2}+L_{mq}^2)}{Dq} \end{bmatrix},$$

$$\mathbf{C}_{21} = \begin{bmatrix} \frac{(L_{mq}L_{kq1}-L_{mq}^2)}{Dq} & 0 & \frac{(L_qL_{mq}+L_{mq}^2)}{Dq} \\ 0 & \frac{(L_{md}L_{kd}-L_{md}^2)}{Dd} & 0 \\ 0 & \frac{(L_{md}L_{fd}-L_{md}^2)}{Dd} & 0 \end{bmatrix},$$

$$\mathbf{C}_{12} = \begin{bmatrix} \frac{(-L_{mq}L_{kq1}+L_{mq}^2)}{Dq} & 0 & 0 \\ 0 & \frac{(-L_{md}L_{kd}+L_{md}^2)}{Dd} & \frac{(-L_{md}L_{fd}+L_{md}^2)}{Dd} \\ \frac{(L_qL_{mq}+L_{mq}^2)}{Dq} & 0 & 0 \end{bmatrix},$$

$$\mathbf{C}_{22} = \begin{bmatrix} \frac{(-L_qL_{kq1}+L_{mq}^2)}{Dq} & 0 & 0 \\ 0 & \frac{(-L_dL_{kd}+L_{md}^2)}{Dd} & \frac{(L_dL_{md}-L_{md}^2)}{Dd} \\ 0 & \frac{(L_dL_{md}-L_{md}^2)}{Dd} & \frac{(-L_dL_{fd}-L_{md}^2)}{Dd} \end{bmatrix}. \tag{A15}$$

$$\mathbf{D} = zeros(6,6). \tag{A16}$$

where

$$L_q = L_{ls} + L_{mq},\ L_d = L_{ls} + L_{md},\ L_{kq1} = L_{lkq1} + L_{mq}. \tag{A17}$$

$$L_{kq2} = L_{lkq2} + L_{mq},\ L_{fd} = L_{lfd} + L_{md},\ L_{kd} = L_{lkd} + L_{md}. \tag{A18}$$

$$D_q = L_{mq}^2\left(L_q - 2L_{mq} + L_{kq1} + L_{kq2}\right) - L_qL_{kq1}L_{kq2}. \tag{A19}$$

$$D_d = L_{md}^2\left(L_d - 2L_{md} + L_{fd} + L_{kd}\right) - L_dL_{fd}L_{kd}. \tag{A20}$$

$$L_{aq}^{-1} = 1/L_{mq} + 1/L_{ls} + 1/L_{lkq1} + 1/L_{lkq2}. \tag{A21}$$

$$L_{ad}^{-1} = 1/L_{md} + 1/L_{ls} + 1/L_{lfd} + 1/L_{lkd}. \tag{A22}$$

A.1.3 Line Start-Permanent Magnet Motor

$$\mathbf{x} = \begin{bmatrix} \lambda_q & \lambda_d & \lambda_{kq} & \lambda_{kd} \end{bmatrix}^T. \tag{A23}$$

$$\mathbf{u} = \begin{bmatrix} v_q & v_d + r_s\frac{L_{md}}{L_{ls}}i_m & v_{kq} & v_{kd} + r_{kd}\frac{L_{md}}{L_{lkd}}i_m \end{bmatrix}^T. \tag{A24}$$

$$\mathbf{y} = \begin{bmatrix} i_q & i_d & i_{kq} & i_{kd} \end{bmatrix}^T. \tag{A25}$$

$$\mathbf{A}_{11} = \begin{bmatrix} \frac{r_s}{L_{ls}}\left(\frac{L_{aq}}{L_{ls}} - 1\right) & -\omega_r \\ \omega_r & \frac{r_s}{L_{ls}}\left(\frac{L_{ad}}{L_{ls}} - 1\right) \end{bmatrix},$$

$$\mathbf{A}_{21} = \begin{bmatrix} \frac{r_{kq}}{L_{lkq}}\frac{L_{aq}}{L_{ls}} & 0 \\ 0 & \frac{r_r}{L_{lr}}\frac{L_{ad}}{L_{ls}} \end{bmatrix}, \mathbf{A}_{12} = \begin{bmatrix} \frac{r_s}{L_{ls}}\frac{L_{aq}}{L_{lkq}} & 0 \\ 0 & \frac{r_s}{L_{ls}}\frac{L_{ad}}{L_{lkd}} \end{bmatrix},$$

$$\mathbf{A}_{22}\!=\!\begin{bmatrix} \frac{r_{kq}}{L_{lkq}}\left(\frac{L_{aq}}{L_{lkq}}-1\right) & 0 \\ 0 & \frac{r_{kd}}{L_{lkd}}\left(\frac{L_{ad}}{L_{lkd}}-1\right) \end{bmatrix}. \tag{A26}$$

$$\mathbf{B}\!=eye(4,4). \tag{A27}$$

$$\mathbf{C}\!=\!\begin{bmatrix} -L_{kq}/D_q & 0 & L_{mq}/D_q & 0 \\ 0 & -L_{kd}/D_d & 0 & -L_{md}/D_d \\ L_{mq}/D_q & 0 & -L_q/D_q & 0 \\ 0 & L_{md}/D_d & 0 & -L_d/D_d \end{bmatrix}. \tag{A28}$$

$$\mathbf{D}\!=zeros(4,4). \tag{A29}$$

where

$$L_q\!=L_{ls}+L_{mq},\ D_q\!=L_{mq}^2-L_{kq}L_q. \tag{A30}$$

$$L_d\!=L_{ls}+L_{md},\ D_d\!=L_{md}^2-L_{kd}L_d. \tag{A31}$$

$$L_{aq}^{-1}=1/L_{mq}+1/L_{ls}+1/L_{lkq}, L_{ad}^{-1}=1/L_{md}+1/L_{ls}+1/L_{lkd}. \tag{A32}$$

A.1.4 DC Machine

$$\mathbf{x}\!=\!\begin{bmatrix} \lambda_f & \lambda_a \end{bmatrix}^T,\ \mathbf{u}\!=\!\begin{bmatrix} v_f & v_a \end{bmatrix}^T,\ \mathbf{y}\!=\!\begin{bmatrix} i_f & i_a \end{bmatrix}^T. \tag{A33}$$

$$\mathbf{A}\!=\!\begin{bmatrix} -\frac{r_f}{L_{ff}} & 0 \\ -\omega_r\frac{r_{af}}{L_{ff}} & -\frac{r_a}{L_{aa}} \end{bmatrix},\ \mathbf{B}\!=\!\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{A34}$$

$$\mathbf{C}\!=\!\begin{bmatrix} 1/L_{ff} & 0 \\ 0 & 1/L_{aa} \end{bmatrix},\ \mathbf{D}\!=zeros(2,2). \tag{A35}$$

A.2 Machine Parameters

The parameters of various machines are given below.

Induction Machine:

3hp, 230V, $r_s=0.5\Omega, r_r=0.51\Omega, l_{ls}=4\text{mH}, l_{lr}=4\text{mH}, l_m=89.4\text{mH}$.

Synchronous Generator:

26kV, 802.5MVA, $r_s=0.0048\Omega, r_{fd}=0.58043\Omega, r_{kd}=0.0203\Omega, r_{kq1}=0.0727\Omega, L_{ls}=0.57031\text{mH}, L_{md}=4.2\text{mH}, L_{mq}=3.8\text{mH}, L_{fd}=0.40759\text{mH}, L_{kd}=0.27852\text{mH}, L_{kq1}=0.16605\text{mH}$.

LSPMSM:

$4\text{hp}, 230\text{V}, r_s = 0.017\Omega, r_{kq} = 0.108\Omega, r_{kd} = 0.054\Omega, l_{ls} = 0.172\text{mH}, l_{lkq} = 0.350\text{mH}, l_{lkd} = 0.350\text{mH}, l_{mq} = 2.7\text{mH}, l_{md} = 1.3\text{mH}, i_m = 1.6203\text{A}.$

***DC Machine*:**

$240\text{V}, r_a = 0.6\Omega, r_f = 240\Omega, L_{ff} = 120\text{mH}, L_{af} = 1.8\text{H}, L_{aa} = 0.012\text{H}.$

B

Material Properties and Machine Geometry in Chapter 4

B.1 Induction Machine Specifications

The parameters and dimensions of induction machine are listed in Table. B.1.

B.2 Material Property

The permeability of iron core is an even function of magnetic flux intensity expressed by

$$\mu(H) = \mu(-H) \tag{B1}$$

and

$$\mu = \begin{cases} \quad 0.183, & H = 0; \\ a_6 H^6 + a_5 H^5 + a_4 H^4 + a_3 H^3 + ... & \\ \qquad\qquad ...\ a_2 H^2 + a_1 H + a_0, & H > 0. \end{cases} \tag{B2}$$

where the coefficients of sextic polynomial function are presented in Table. B.2.

Table B.1: Specification of Induction Machine.

Parameter	**Value (Unit)**
Stator outer diameter	195.38 (mm)
Rotor outer diameter	114.9 (mm)
Stator inner diameter	115.54 (mm)
Rotor inner diameter	36.5 (mm)
Stator slot depth	21.1 (mm)
Stator tooth width	5.7 (mm)
Rotor slot depth	22.1 (mm)
Rotor tooth width	6.2 (mm)
Stator tooth face width	7.4 (mm)
Rotor tooth face width	12.7 (mm)
Motor length	107.95 (mm)
Stator tooth face width	7.4 (mm)
Rotor tooth face width	12.7 (mm)
Stator tooth flange thickness	1.2 (mm)
Rotor tooth flange thickness	1.8 (mm)
Mechanical air-gap length	0.31 (mm)
Number of stator slots	36
Number of rotor slots	28
Rated voltage	230 (V)
Frequency	60 (Hz)
Number of poles	4
Rotor inertia	0.025 (Kg.m^2)
Winding connections	Wye
Rotor type	Squirrel cage

Table B.2: Coefficients of eleven-segment sextic polynomial function.

Condition	a_6	a_5	a_4	a_3	a_2	a_1	a_0
0$< H \leq$6	8.0877e-6	-1.7992e-4	0.0017	-0.0084	0.027	-0.0669	0.1832
6$< H \leq$12	1.5357e-8	-1.002e-6	2.8222e-5	-4.4966e-4	0.0045	-0.0299	0.1543
12$< H \leq$20	5.3198e-10	-6.1076e-8	3.0226e-6	-8.42e-5	0.0015	-0.0162	0.1279
20$< H \leq$1e2	3.727e-13	-1.2845e-10	2.2318e-8	-2.0899e-6	1.1457e-4	-0.0038	0.0765
1e2$< H \leq$1e3	2.9884e-19	-1.1222e-15	1.7099e-12	-1.3597e-9	6.0526e-7	-1.5092e-4	0.0205
1e3$< H \leq$8e3	1.6017e-25	-4.9291e-21	6.1873e-17	-4.0799e-13	1.5174e-9	-3.1808e-6	0.0036
8e3$< H \leq$3e4	3.8695e-30	-5.0942e-25	2.8012e-20	-8.3311e-16	1.4477e-11	-1.4773e-7	8.4505e-4
3e4$< H \leq$15e4	1.1805e-34	-7.3192e-29	1.8773e-23	-2.5741e-18	2.0337e-13	-9.2756e-9	2.3158e-4
15e4$< H \leq$5e5	1.1739e-38	-2.6458e-32	2.4967e-26	-1.2774e-20	3.8275e-15	-6.7392e-10	6.6024e-5
5e5$< H \leq$2e6	1.2401e-42	-1.0712e-35	3.8560e-29	-7.4825e-23	8.4452e-17	-5.556e-11	2.0111e-5
2e6$< H$	0	0	0	0	0	0	1.7154e-6