UNIVERSITY OF TECHNOLOGY, SYDNEY
Faculty of Engineering and Information Technology

DEVELOPMENT OF FPGA BASED CONTROL ARCHITECTURE FOR PMSM DRIVES

by

Quang Nguyen Khanh

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Sydney, Australia

2015

Certificate of Authorship/Originality

I certify that the work in this thesis has not been previously submitted for a degree nor has it been submitted as a part of the requirements for other degree except as fully acknowledged within the text.

I also certify that this thesis has been written by me. Any help that I have received in my research and in the preparation of the thesis itself has been fully acknowledged. In addition, I certify that all information sources and literature used are quoted in the thesis.

Production Note:
Signature removed prior to publication.

Quang Nguyen Khanh

ABSTRACT

DEVELOPMENT OF FPGA BASED CONTROL ARCHITECTURE FOR PMSM DRIVES

by Quang Nguyen Khanh

The rapid advancement of the very large scale integration (VLSI) technology and electronic design automation techniques in recent years has made a significant impact on the development of complex and compact high performance control architecture for industrial motion systems.

Specific hardware with the field programmable gate array (FPGA) technology is now considered as a promising solution in order to make use of the reliability and versatility of controllers. Indeed, FPGAs have been successfully used in many control applications such as power converter control and electrical machines control. This is because such an FPGA-based implementation can offer an effective reprogrammable capability and overcome disadvantages of microprocessor-based or digital signal processor-based embedded systems.

This thesis aims to provide a proof-of-concept for the control-system-on-chip and a prototype for a fully-implemented FPGA control architecture for permanent magnet synchronous motor (PMSM) drives. In this thesis, a special focus is given on analytical effects, design procedure, and control performance enhancement for PMSM drives under sensor/sensorless vector control using a number of control techniques.

The control schemes include FPGA-based intelligent control and robust cascade control for single axis and multiple axis tracking with PMSMs. An important contribution of this thesis rests with a convincing demonstration of high performance estimation schemes, using sliding mode observers and extended Kalman filters, in terms of accuracy and robustness against noisy and/or perturbed currents for sensor-

less PMSM control based on the FPGA technology. In addition, a sequential finite state machine is developed in this work to result in less logic gate resources, leading to a faster processing time.

Significance of this thesis contribution includes in providing a feasible and effective solution for the implementation of complex control strategies to fully exploit the FPGA advantages in power electronics and drive applications.

List of Publications

1. **Nguyen Khanh Quang**, Nguyen Trung Hieu, Q. P. Ha (2014), *FPGA-Based Sensorless PMSM Speed Control Using Reduced-Order Extended Kalman Filters*, IEEE Transactions on Industrial Electronics, vol.61, no.12, pp.6574-6582.

2. Q. P. Ha, Ying-Hao Yu and **Nguyen Khanh Quang** (2012), *FPGA-based cooperative control of indoor multiple robots*, International Journal of Advanced Mechatronic Systems, vol.4, Nos. 5/6, pp.248-259.

3. **Nguyen Khanh Quang**, Doan Duc Tung and Q. P. Ha (2015), *FPGA-Based Sensorless PMSM Speed Control using Adaptive Extended Kalman Filter*, The 11th IEEE Int. Conf. on Automation Science and Engineering (CASE2015), Gothenburg, Sweden, to be published 2015.

4. **Nguyen Khanh Quang**, Nguyen Trung Hieu and Q. P. Ha (2014), *FPGA Sensorless PMSM Drive with Adaptive Fading Extended Kalman Filtering*, The 13th Int. Conf. on Control, Automation, Robotics and Vision (ICARV2014), Singapore, pp.295-300.

5. **Nguyen Khanh Quang**, Doan Quang Vinh, Nguyen D. That and Q. P. Ha (2013), *Observer-based Integral Sliding Mode Control for Sensorless PMSM Drives using FPGA*, The 2nd International Conference on Control, Automation and Information Sciences (ICCAIS2013), Nha Trang, Vietnam, pp.218-223.

6. **Nguyen Khanh Quang**, Doan Quang Vinh (2013), *Sensorless FPGA-Based PMSM Drives using Improved Sliding Mode Observer*, The 1st Vietnam conference on Control and Automation (VCCA2013), Da Nang, Vietnam, pp.164-170.

7. Tri Tran and **Nguyen Khanh Quang** (2013), *Distributed Model Predictive Control with Receding-Horizon Stability Constraints*, The 2nd Int. Conf. on Control, Automation and Information Sciences (ICCAIS2013), Nha Trang, Vietnam, pp.137-141.

8. Nguyen D. That, **Nguyen Khanh Quang**, Pham Thanh and Q. P. Ha (2013), *Robust Exponential Stabilization of the Pendubot in the Presence of Bounded External Disturbances Using Sliding Mode Control*, The 2nd International Conference on Control, Automation and Information Sciences (ICCAIS2013), Nha Trang, Vietnam, pp.137-141.

9. **Nguyen Khanh Quang**, Nguyen D. That, Nguyen Hong Quang and Q. P. Ha (2012), *FPGA-based Fuzzy Sliding Mode Control for Sensorless PMSM Drive*, The 8th IEEE International Conference on Automation Science and Engineering (CASE2012), Seoul, Korea, pp.172-177.

10. **Nguyen Khanh Quang**, N. T. Hieu, G. P. Hunter, and Q. P. Ha (2012), *FPGA-based Sensorless PMSM Drive Using Parallel Reduced-Order Extended Kalman Filter*, The 1st Int. Conf. on Control, Automation and Information Sciences (ICCAIS2012), Ho Chi Minh, Vietnam, pp.164-169.

11. **Nguyen Khanh Quang**, Tri Tran (2012), *Dissipativity Criterion for Linear Systems with a Coupling Delay and Asymptotically Positive Realness Constraint*, The 1st Int. Conf. on Control, Automation and Information Sciences (ICCAIS2012), Ho Chi Minh, Vietnam, pp.60-65.

12. Nguyen D. That, **Nguyen Khanh Quang**, Raja M. T. Raja Ismail, Phan T. Nam and Q. P. Ha (2012), *Improved reachable set bounding for linear systems with discrete and distributed delays*, The 1st Int. Conf. on Control, Automation and Information Sciences (ICCAIS2012), Ho Chi Minh, Vietnam, pp.137-141.

13. **Nguyen Khanh Quang**, Y.-S. Kung, and Q. P. Ha (2011), *FPGA-Based Control Architecture Integration for Multiple-Axis Tracking Motion Systems*, IEEE/SICE International Symposium on System Integration (SII2011), Kyoto, Japan, pp.591-596.

14. Y.S. Kung, **Nguyen Khanh Quang** and Le Thi Van Anh (2009), *FPGA-based neural fuzzy controller design for PMLSM drive*, The 8th Int. Conf. on Power Electronics and Drive Systems (PEDS2009), Taipei, Taiwan, pp.222-227.

Acknowledgements

First of all, I would like to express my sincerely thanks to my PhD supervisor, Associate Professor Quang Phuc Ha, for his guidance, advice, encouragement and support in the course of my doctoral works.

Special thanks to Professor Ying-Shieh Kung, Nguyen Trung Hieu who not only helped me to extend my research horizon, but rigorously correct my errors, from which I have made accountable improvements.

I would like to take this opportunity to thank the Faculty of Engineering and Information Technology of UTS and the Vietnam International Education Development (VIED) for the financial support for my research and study at UTS. Several friends whose concerns and support have helped me overcome obstacles and concentrate on my study; I greatly appreciate their friendship and assistance.

Most importantly, my thesis could not have been completed without the encouragement and support from my family. I would like to dedicate my thesis to my parents, Nguyen Van Kiem, Tran Thi Kim Dinh, Le Xuan Lai, Ton Nu Thu Dong and my beloved wife, Le Xuan Dong Phuong and son, Nguyen Quang Minh who are a source of care, love, support and strength during my graduate study.

Quang Nguyen Khanh

Sydney, Australia, 2015.

Contents

List of Figures

List of Tables

Nomenclature and Notation

List of abbreviations

- ADC : Analog digital converter
- ASIC : Application-specific integrated circuit
- CCCT : Current controller and coordinate transformation
- EDA : Electronic design automation
- EKF : Extended Kalman filter
- EMF : Electromotive force
- EPROM : Programmable read-only memory
- FC : Fuzzy control
- FPGA : Field programmable gate array
- FSM : Finite state machine
- HF : High frequency
- IDE : Integrated development environment
- IP : Intelligent property
- IR : Interrupt service routine
- LE : logic element
- LPM : Library parameterized module
- LUT : Look up table
- PLD : Programmable logic device
- PMSM : Permanent magnet synchronous motor
- PMLSM : Permanent magnet linear synchronous motor
- PI : Proportional integral controller
- QEP : Quadrature encoder pulse

- RBF : Radial basis function
- SoPC : System-on-programmable-chip
- SRAM : Static random access memory
- SVPWM : Pulse-width-modulation
- VHDL : Very high speed integrated hardware description language
- VLSI : Very large scale integration

List of symbols

- θ : Rotor position
- F_e : Motor thrust force
- F_L : External load force
- B_m : Viscous friction coefficient
- K_t : Force constant
- J : Inertia
- λ : Permanent magnet flux linkage
- p : Pairs of poles of a motor
- τ : Pole pitch
- ω_e : Electrical speed (rad/s)
- r_s : Stator resistance
- v : Voltage
- i : Current
- T_s : Sampling period
- x : State space vector
- u : Input vector
- y : Output vector
- $\nu; \xi$: The discrete forms of system and measurement noise
- K : Kalman matrix
- $P; P0$: State error covariance matrix, Initial state error covariance matrix
- $Q; R$: Covariance state noise and covariance measurement noise matrices

- $F; H; \Phi$: Jacobian, output matrix, and state transition matrices

- s : Laplace operator

Indexes

- $d - q$: Rotating reference frame indexes

- $\alpha - \beta$: Stationary reference frame indexes

- $a; b; c$: Three phase reference frame indexes

- $*$: Reference quantity

- $\hat{}$: Estimated quantity

- n : Sampling index

- $n/n-1$: Predicted quantity

- n/n : Optimal estimated quantity

Chapter 1

Introduction

1.1 Motivation

Electrical drive controls are playing an important role in a wide range of industrial control applications, commonly used in transportation systems, material handling, precision machining, and many automation processes. The rising user demand in terms of dynamic response, precision and flexibility forced by technological advancement and energy conservation requires enhanced control. During recent decades, controlled electrical drives have performed in various configurations and have become a mature technology with an already substantial and continuously increasing worldwide market. This is due to greater innovations in the semiconductor and microelectronics industry in the form of efficient power-electronic devices and digital signal processors. Mechanical loads in industry are mainly driven by AC motor drive due to its free from the drawbacks of mechanically commutated DC drives. The stator and rotor of an AC motor are the only contact bearing components. The spinning of the rotor is caused by the stators magnetic field, and needs more complicated control technology (such as the field oriented control) to implement its motion systems. Thanks to the development of semiconductor control devices and the course of development of different types of AC motor control algorithms, the computation requirements for AC motor control can be met easily. This advantage has renewed variable-speed AC motor drives and has contributed actively to the field of control engineering.

The fast progress of the very large scale integration (VLSI) technology and electronic design automation (EDA) techniques in recent years has made a significant impact on the development of complex and compact high performance control architecture for industrial motion systems. Design verification and validation methods were adopted in EDA tools to ensure correctness of the system design with very high confidence in the right prototypical operation of the final product. For instance, EDA Simulator Link allowed us to perform hardware verification on the FPGA board using FPGA-in-the-loop simulation. Therefore, FPGA technology is now considered as a promising solution in order to make use of the reliability and versatility of controllers. Indeed, FPGAs have been successfully used in many control applications such as power converter control (for example, pulse-width-modulation inverters, matrix converters) and electrical machines control (motion axis control, multi-machines systems, neural network control of induction machine drives, fuzzy logic control of power generators, and speed measurement). This is because such an FPGA-based implementation can offer an effective reprogrammable capability and overcome disadvantages of microprocessor-based or digital signal processor-based embedded systems.

In this background, the motivation for this thesis is based on recent growing research interest in the evaluation of FPGA-based AC drives for electrical applications.

1.2 Research Objectives

The goal of this thesis is the development of a fully-implemented FPGA control architecture for permanent magnet synchronous motor (PMSM) drives. The main emphasis of this research is laid on:

i. A feasible development of an effective design procedure for FPGA-based control for PMSM drives with a substantial decrease in the resource usage, execution time

and control performance enhancement due to a sequential finite state machine design method followed by computer simulation and experimental validation.

ii. Development of a hadware/software co-design of an FPGA-based intelligent control and robust cascade control for single axis and multiple axis positioning and tracking in dealing with unmodelled dynamics and cross-axis interferences of control system.

iii. Development of the full hardware of the estimation and control paradigm on a single FPGA chip. The high performance estimation schemes, using improved sliding mode observer, adaptive extended Kalman filter, adaptive fading EKF and reduced-order EKFs, in terms of accuracy and robustness against noisy and/or perturbed currents for sensorless PMSM drives.

1.3 Thesis Structure

This thesis consists of nine chapters:

Chapter 1 provides the motivation of thesis. Next, the research objectives and contributions of the thesis, are outlined.

Chapter 2 deals with FPGA technology and ModelSim/Simulink co-simulation method. It starts with an overview of FPGA technology and issues related to digital hardware implementation. Then, a sequential finite state machine method is presented to realize the mathematical operations, and the very high speed integrated circuit-hardware description language is adopted to describe the circuit of the FSM. Next, the chapter presents the ModelSim/Simulink co-simulation method. The co-simulation work in ModelSim and Simulink environment, how two different softwares can co-work with each other is also presented. Finally, a simulation work is introduced to illustrate the ability of the co-simulation method.

In **Chapter 3**, a brief survey of PMSM drives is presented. It begins with an introduction of PMSM and then FPGA based realization of current vector control of a PMSM drive is described. Next, this chapter gives a brief review of control techniques for PMSM drives, which relates to studies in this thesis, that are intelligent control and estimation schemes for PMSM drives.

Chapter 4 addresses the integration of a multi-loop PI and neural fuzzy control system for multiple-axis motion positioning and tracking via the use of the FPGA technology. In this chapter, a radial basis function neural network (RBF NN) is used to identify the plant dynamic and provide more accurate plant information during parameter tuning of fuzzy control (FC).

Chapter 5 presents the design and evaluation of an observer-based integral sliding mode controller for a sensorless PMSM drive based on the FPGA technology. The system performance can be substantially enhanced by integrating the observer-based and integral sliding mode control techniques into speed control of a PMSM drive.

Chapter 6 addresses the design and implementation of an adaptive fading extended Kalman filter (EKF) for the sensorless PMSM on an FPGA chip. In this chapter, the adaptive fading EKF has been developed to recover the estimation results in events of frequent and sharp state jumps in the control system.

Chapter 7 presents the design and implementation of an adaptive extended Kalman filter for the sensorless PMSM on an FPGA chip. In this chapter, the improved EKF versions can be obtained by incorporating an adjustment mechanism of the noise covariances into the filter.

Chapter 8 presents the design and implementation of an FPGA-based architecture for the speed control of sensorless PMSM drives using a reduced-order extended Kalman filter. For the reduction of computation resources, as well as accuracy im-

provement in the rotor position estimation, a parallel reduced-order EKF is proposed in this chapter.

Finally, **Chapter 9** presents the summary of the results obtained in this work, conclusion reached as well as contributions made to advance the knowledge, and the future works of research. The last section is a bibliography.

Chapter 2

FPGA Technology, Finite State Machine, and ModelSim/Simulink Co-simulation: An Overview

2.1 Introduction

Due to the growing complexity of control algorithms and the increasing of industrial requirements, power electronics and drive control systems are nowadays becoming more sophisticated. These requirements are not just limited to new algorithm issues or high control performances. Indeed, the compactness, flexibility and low cost advantages are also increasing with the adoption of the controller-on-programmable-chip concept. Among novel digital signal processing technologies, FPGA technology not only is an attractive solution for a wide range of applications, but also has brought forward the concept of system-on-programmable-chip (SoPC) to implement a highly complex control algorithm.

In this chapter, an overview of FPGA technology and methods for designing and implementing the controller-on-programable-chip are presented. Firstly, the basic concepts of FPGA are presented in section 2.2. Then, a sequential finite state machine (FSM) method is proposed and described in section 2.3 to show how to realize the mathematical operations and functions computation on FPGA. This method will be applied for VHDL (very high speed integrated circuit-hardware description language) design for whole control algorithms in this thesis. Next, section 2.4 presents ModelSim/Simulink co-simulation method, which is introduced and applied for the

computer simulation work of this research. Co-simulation method is used to confirm the effectiveness of VHDL code of control IP (Intelligent property) before realizing this code in the FPGA based experimental system.

2.2 FPGA Technology

Field programmable gate arrays are digital integrated circuits (ICs), which are programmed by the customer or design engineers after manufacturing to perform a very wide variety of tasks. The FPGAs technology is a middle ground between programmable logic devices (PLDs) and ASICs because their functionality can be customized in the field like PLDs, but they can contain millions of logic gates and are dedicated to implement extremely large and complex combinational functions using a hardware description language (HDL), similar to that which could be realized using only ASICs.

2.2.1 FPGA Programming Technologies

As mentioned, FPGA is similar to a PLD but, whereas PLDs are generally limited to hundreds of gates, FPGAs support millions of gates. FPGAs offer the possibility to design the prototyping integrated circuit. Once the design is set, hardwired chips are produced for faster performance. FPGAs are classified into two major categories:

- Reprogrammable: EPROM, EEPROM, Flash and SRAM based FPGAs (volatile).

- One time programmable: Fuses (destroy internal links with current), PROM and Anti-fuse based FPGAs (grow internal links).

In the following, three different major technologies are in use today for programming FPGAs: antifuse, SRAM, and FLASH EPROM.

- **SRAM technology**: It is mainly manufactured by Altera, Lucent and Xilinx

vendors. The advantages are that such devices can be reprogrammed rapidly and easily, and SRAM uses a standard fabrication technology that is constantly being improved upon. Moreover, these circuits can be dynamically reconfigured partially or entirely. The configurations are stored in the static SRAM. However, the main drawback of SRAM-based FPGA is that each cell consumes a significant amount of silicon real estate because the cells are formed from four or six transistors configured as a latch. Another disadvantage is that the device's configuration data will be lost when power is removed from the system, so these devices always have to be reprogrammed when the system is powered up. Therefore, the use of this technology is limited in the case of critical safety applications such as aircraft and automotive fields [3],[96].

- **Antifuse technology**: It is proposed by Actel and QuickLogic vendors. This technology is created by making physical connections, which are based on the injection of a high current or a laser that heats and then melts the silicon layer between endpoints. Hence, the configuration is maintained even after power is on and this technology tends to be faster and requires lower power. However, it cannot be re-programmed. This makes it impractical in the case of prototyping environments [3],[96].

- **Flash technology**: This technology grew out of an earlier technology known as erasable programmable read-only memory (EPROM) that allows devices to be programmed, erased, and reprogrammed with new data. However, the main disadvantages of this technology are the limitation of the available internal resources and the limited number of reconfiguration cycles [2].

Novel FPGAs that include microprocessors are used as the System-on-Chip in computer architecture. They are typically described in a high level language, such as VHDL, VeriLog, SystemC. Tool support, to turn the design and verify the layout, is

provided by the device manufacturers.

2.2.2 Architecture of FPGA

FPGAs, as shown in Figure 2.1, generally consist of a two-dimensional array of configurable logic blocks (CLBs) of potentially different types, including general logic, memory and multiplier blocks, and linked to each other by an interconnection network which is entirely reprogrammable. The memory cells control the logic blocks as well as the connections so that the component can fulfill the required application specifications [54].

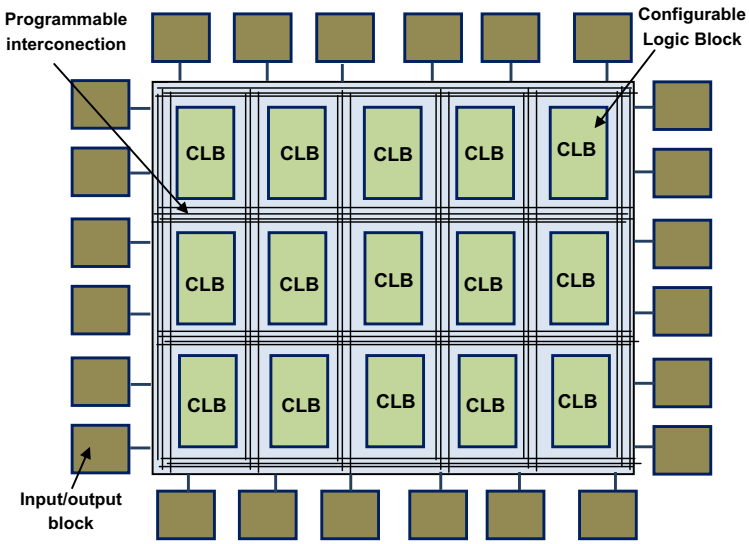


Figure 2.1 : General structure of FPGA

CLBs are designed with the programmable logic of FPGA. Their structures include two, four, or more logic cells, also called logic elements. A logic element usually consists of one or more RAM based on input look-up tables (LUTs), and one or more flip-flops. LUTs are used to implement combinational logic. A typical logic element

(LE) is shown in Figure 2.2. There may also be additional hardware support in each logic element to enable other high speed arithmetic and logic operations.

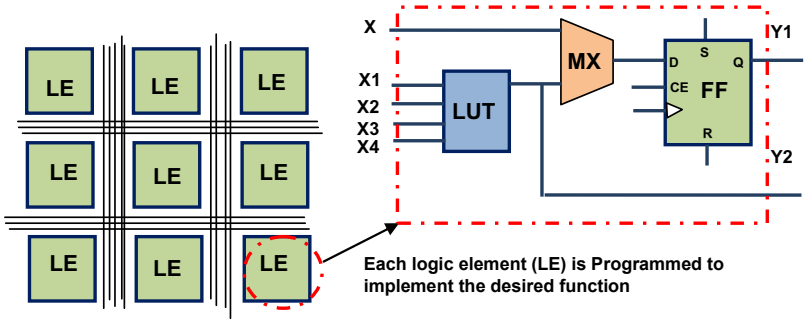


Figure 2.2 : Configurable logic block

2.3 Computation using Finite State Machine Method

Most on-chip programmable systems have relied on systolic arrays to deal with the complicated computation of control algorithms [13],[25],[33],[34],[35],[57],[67],[69]. Although systolic arrays, possessing parallel processing capability, are ideal to implement on FPGAs with a fast execution time, their parallel architecture is quite complicated and requires much gate array resources.

The implementation of control algorithms on a FPGA is still at a primary level. There is no fixed methodology for the algorithm development on these devices because of their generic nature. Therefore, design engineers have greater flexibility to develop strategies for implementation [67],[69],[90]. This section presents and demonstrates that sequential finite state machine design method can solve the issue of optimal usage of the limited resources on a FPGA. For a demonstration the computations of some mathematical operations and special functions are considered and their detailed implementation is described.

2.3.1 Sum of Product Computation

Design a VHDL to compute the sum of product (SOP) as follows

$$y = a_1x_1 + a_2x_2 + a_3x_3. \tag{2.1}$$

Solution: Two kinds of design method shown in Figure 2.3 are presented to realize the computation of SOP . There are parallel processing method and sequential execution method.

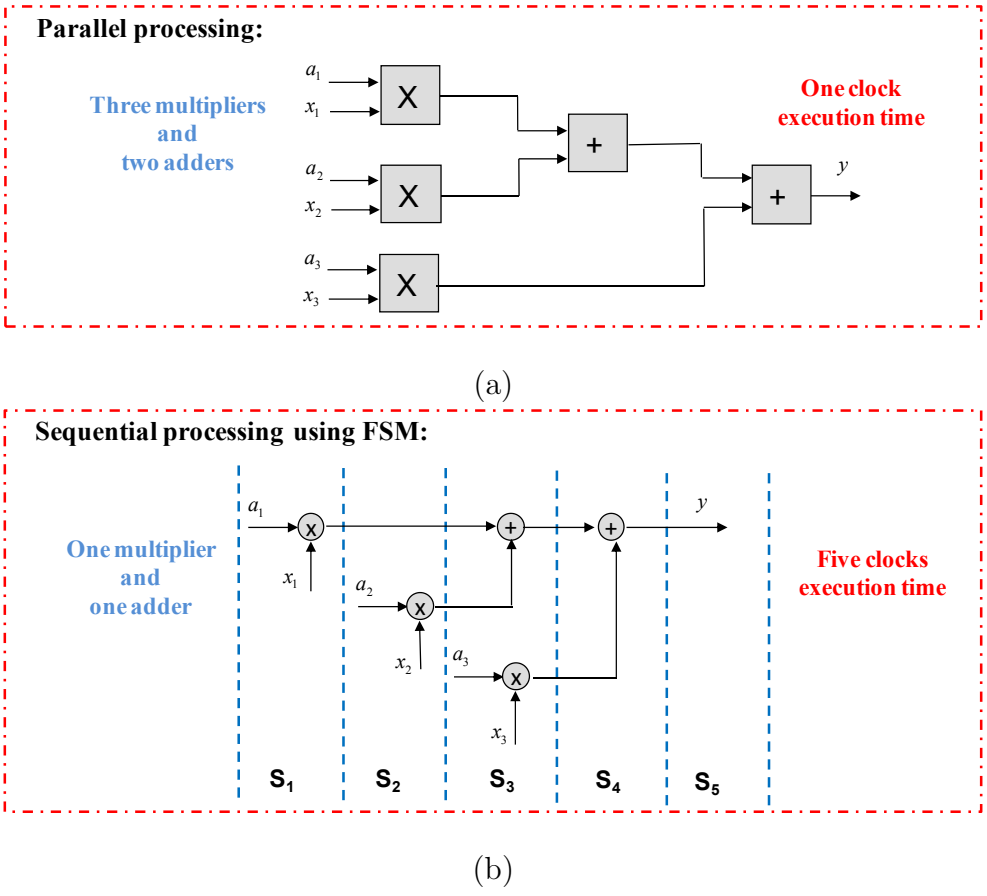


Figure 2.3 : Sum of product computation

Parallel processing with the designed SOP circuit is shown in Figure 2.3(a), which will operate continuously and simultaneously. The SOP circuit requires 2 adders,

3 multipliers, and merely near one clock time to complete the overall computation. With the advantage of fast computation ability, the parallel processing method, however, consumes much more FPGA resources. To solve this problem, a sequential execution method using FSM to model the SOP circuit is adopted and is shown in Figure 2.3(b). The FSM method uses one adder, one multiplier and manipulates 5 steps (or 5 clocks time) machine to carry out the overall computation of SOP. Compared to the parallel processing method, the FSM method requires more operation time (if one clock time is 80 ns, 5 clocks needs 0.4 μs) in executing SOP circuit; nevertheless, it does not lose any computation power. As a result, the more complicated computation in algorithm, the more FPGA resources will be saved by applying FSM method. Besides, the state diagram in Figure 2.3(a) is easy to be described by VHDL.

For the design of a VHDL to compute sum of product in equation (2.1), let us transfer all coefficients and variables to the data type 16 bits length with Q15 format with testing values as follows,

$a_0 = 0.1 \leftrightarrow 0.1 * 32768 = 3277 = X"0CCD"$; $a_1 = 0.2001 \leftrightarrow 0.2001 * 32768 = 6557 = X"199D"$; $a_2 = -0.4525 \leftrightarrow -0.4525 * 32768 = -14828 = X"C614"$;

$x_0 = 0.2001 \leftrightarrow 0.2001 * 32768 = 6557 = X"199D"$; $x_1 = 0.2001 \leftrightarrow 0.2001 * 32768 = 6557 = X"199D"$; and $x_2 = 0.1 \leftrightarrow 0.1 * 32768 = 3277 = X"0CCD"$.

Therefore $y = 0.01479 \leftrightarrow 0.01479 * 32768 = 484 = X"1E4"$;

2.3.2 Computation of the Polynomial Equation

Write a VHDL code to compute the following polynomial equation:

$$y = a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0, \tag{2.2}$$

with $a_5 = 2, a_4 = 3, a_3 = 5, a_2 = 1, a_1 = 12, a_0 = 3,$ and $x = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.$

Solution: If we use the parallel processing method, it requires 5 adders, 5 multipliers, and one clock time to complete the overall computation. However, for easy sequential computation, the equation (2.2) is transferred in the following form

$$y = (x(x(x(xa_5 + a_4) + a_3) + a_2) + a_1) + a_0. \tag{2.3}$$

FSM is employed to model the polynomial form in equation (2.3) and it is shown in Figure 2.4, which uses one adder, one multiplier and needs 10 steps machine to do the computation. The multiplier and adder apply Altera LPM (library parameterized modules) standard and FSM can be easily described by VHDL. Moreover, the operation of each step in Figure 2.4 can be completed within 80 ns (12.5 MHz clock); therefore total 10 steps only need 0.8 μs operational times.

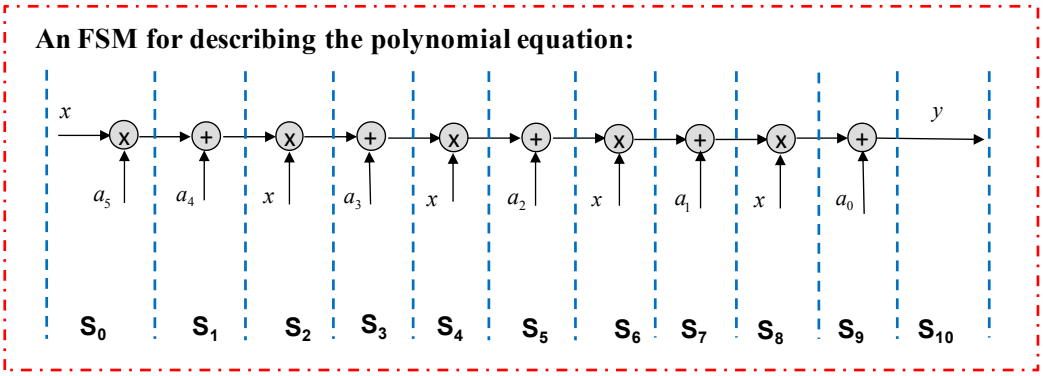


Figure 2.4 : State diagram of an FSM for describing the polynomial equation

Figure 2.5 shows the result of simulation of polynomial equation computation in Quartus II software, if $x = 0.1$ then output is $y = 0.137$. Using Q15 format with 16 bits length, 0.1 corresponds to X"$0CCD$" $(0.1 * 32768 = 3277 = X$"$0CCD$"$)$ and 0.137 corresponds to X"1188" .

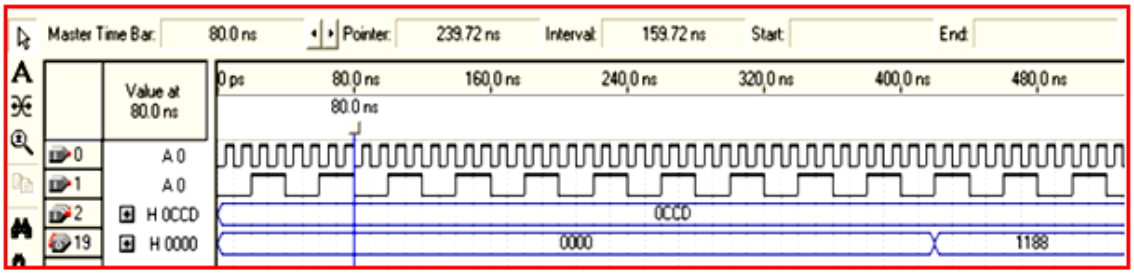


Figure 2.5 : Simulation result of polynomial equation computation in Quartus II

2.3.3 Exponential Function Computation

If a digital signal processing algorithm is implemented with FPGAs and the algorithm uses a nontrivial algebraic function such as square root or trigonometric functions or exponential function, we can always use the Taylor series to approximate this function [42],[69],[90]. For example, let us write a VHDL code to compute the exponential function as,

$$y = e^{-x^2} \triangleq e^{-u}. \tag{2.4}$$

Solution: In order to simplify the computation, the input of exponential function is limited within $0 - 4$ because if $u \geq 4$ the output $y \leq e^{-4} = 0.0183 \approx 0$ so y will be approximated to zero, otherwise if $0 \leq u < 4$, (2.4) can be computed by using Taylor series approximation:

$$y = e^{-u} = \sum_{r=1}^{\infty} (-1)^n \frac{u^n}{n!}. \tag{2.5}$$

The bigger the order of the Taylor series is, the higher the degree of accuracy of exponential function estimated. By MATLAB computation, a good approximation of exponential function is shown in Figure 2.6 with 9th order. Therefore, we select 9th order in (2.5) and define $r = u/4$ to normalize the input value, equation (2.5)

becomes,

$$y = e^{-4r} = \sum_{n=1}^{12}(-1)^n \frac{4^n r^n}{n!}$$
$$= 1 - 4r + 8r^2 + 10.6667r^3 + 10.6667r^4 - 8.5333r^5 + ... + 0.035r^9. \quad (2.6)$$

To avoid the numerical overflow condition during computation, (2.6) is divided by 16 and it becomes,

$$y = 16\sum_{n=1}^{9}\frac{1}{16}(-1)^n \frac{4^n r^n}{n!} = 16\sum_{n=1}^{9}(-1)^n \frac{4^{n-2}}{n!} r^n \triangleq 16\sum_{n=1}^{9} a_n r^n, \quad (2.7)$$

where $a_n = \sum_{n=1}^{9}(-1)^n \frac{4^{n-2}}{n!} \in [-1, 1]$. For easy sequential computation, we transfer the form of (2.7) as follows,

$$y = 16(r(r(r(r(r(r(ra_9 + a_8) + a_7) + a_6) + a_5) + a_4) + a_3) + a_2) + a_1) + a_0. \quad (2.8)$$

From (2.8) we can easily use the sequential FSM method to write a VHDL code for exponential function computation like the polynomial equation calculation above.

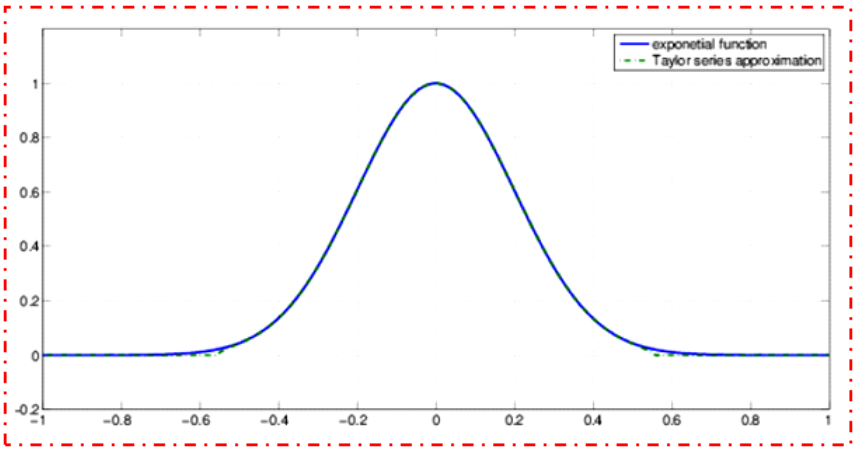


Figure 2.6 : The result of exponential function with the ninth order in MATLAB

The result of exponential function computation is, if $x = 0.3$ (X"2666": Q15

format with 16 bits length) then output is $y = 0.9139$ (X"$74FB$") as shown in Figure 2.7.

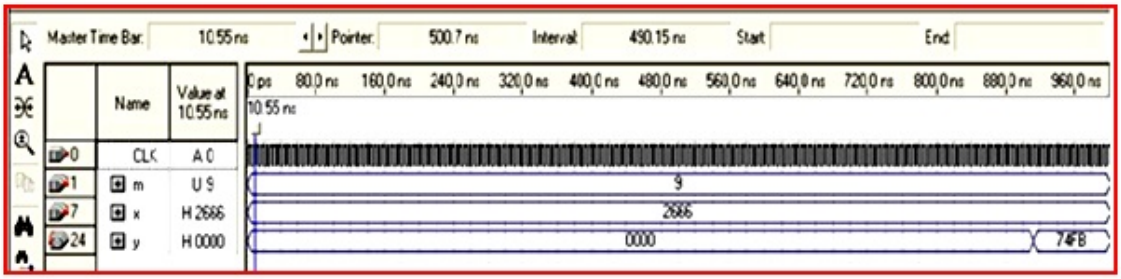


Figure 2.7 : Simulation result of exponential function computation in Quartus II

2.4 ModelSim/Simulink Co-simulation Method

2.4.1 Introduction of ModelSim/Simulink Co-simulation

Simulation is performed by the Electronic Design Automation (EDA) Simulator link. EDA Simulator Link provides a verification interface between MATLAB or Simulink and the HDL simulator or FPGA board. Using EDA Simulator Link we verified our Verilog HDL design against the Simulink model and MATLAB algorithm by using co-simulation with a Verilog or VHDL simulator, such as Mentor Graphics ModelSim. The EDA Simulator Link also allowed us to perform hardware verification on the FPGA board using FPGA-in-the-loop simulation. With EDA Simulator Link, the MATLAB code and Simulink model developed can be used as a test bench to generate stimulus for our HDL design for analysis of the system responses and design elements [44],[53],[56],[66],[70],[72]. ModelSim and Simulink are two independent softwares running in one computer. Therefore, a communication setup needs to be done to co-simulate two softwares together. ModelSim software is the product of Mentor

Graphics. Mentor Graphics was the first to combine single kernel simulator (SKS) technology with a unified debug environment for Verilog, VHDL, and SystemC. The combination of industry-leading, native SKS performance with the best integrated debug and analysis environment make ModelSim an excellent choice for both ASIC and FPGA design. The best standards and platform support in the industry make it easy to adopt in the majority of process and tool flows [56]. The EDA Simulator Link software consists of MATLAB functions and HDL simulator commands for establishing the communication links between the HDL simulator and The MathWorks products. In addition, a library of Simulink blocks is available for including HDL simulator designs in Simulink models for co-simulation. EDA Simulator Link IN software streamlines FPGA and ASIC development by integrating tools available for:

1. Developing specifications for hardware design reference models
2. Implementing a hardware design in HDL based on a reference model
3. Verifying the design against the reference design.

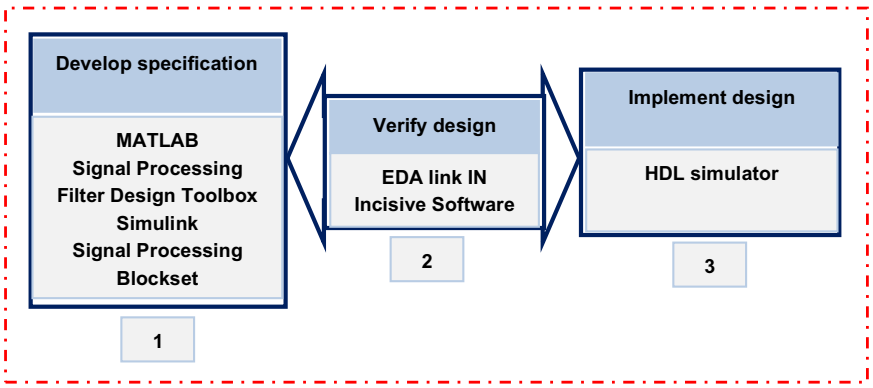


Figure 2.8 : EDA Simulator Link software [56]

Figure 2.9 shows how the HDL simulator and MathWorks products fit into this hardware design scenario. When linked with MATLAB, the HDL simulator functions

as the client. A MATLAB server function waits for service requests that it receives from an HDL simulator session in this scenario.

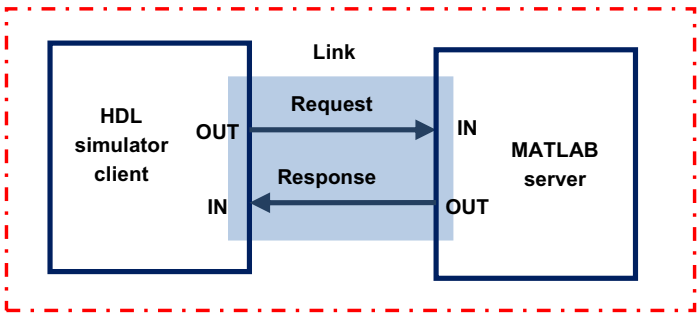


Figure 2.9 : Linking MATLAB with HDL simulator [56]

After receiving a request, the server establishes a communication link and invokes a specified MATLAB function that computes data for, verifies, or visualizes the HDL module (coded in VHDL or Verilog) that is under simulation in the HDL simulator. Figure 2.10 shows how a MATLAB test bench function wraps around and communicates with the HDL simulator during a test bench simulation session.

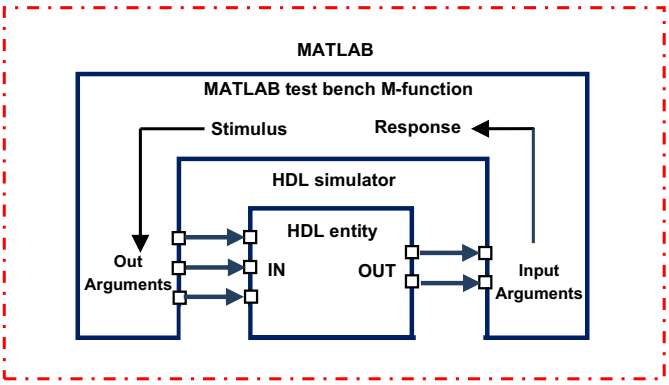


Figure 2.10 : MATLAB and ModelSim co-simulation structure [56]

ModelSim emulates the VHDL codes based on the output arguments from MATLAB/Simulink as the input of designed entity and sends the result back to it. Thus,

two softwares need to communicate and transfer data to each other. The mode of communication that is used for a link between the HDL simulator and Matlab or Simulink depends on whether the simulation application runs in a local, single-system configuration or in a network configuration. If the HDL simulator and The MathWorks products can run locally on the same system and the application requires only one communication channel, we have the option of choosing between shared memory and TCP/IP socket communication. In this work, communication via TCP/IP using port 4449 is used as shown in Figure 2.11,

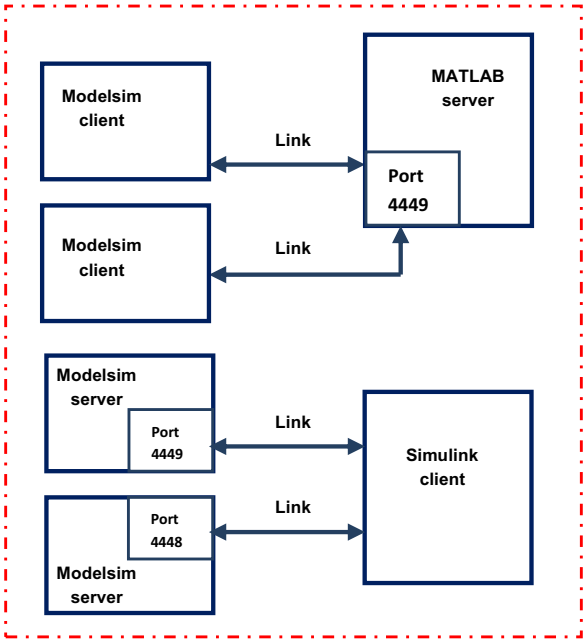


Figure 2.11 : Connection between ModelSim and Simulink via network port 4449 [56]

2.4.2 Simulation Work

Figure 2.12 shows ModelSim/Simulink co-simulation architecture of sensorless speed control for the whole thesis simulation works. The PMSM, inverter and speed commands are implemented in Simulink and the sensorless speed controller is described

by the VHDL code in ModelSim with five modules. Module 1, Module 2 in Figure 6 perform the function of speed controller, the function of current controller and coordinate transformation (CCCT) and SVPWM, and Module 3 to Module 5 perform the function of rotor flux position estimation using parallel reduced-order EKF, full order EKF and SMO, respectively. The sampling frequency of current and speed control is designed with 16 kHz and 2 kHz, respectively. The clocks of 50 MHz and 12.5 MHz supply all modules of ModelSim [66],[67],[68],[70],[71],[73].

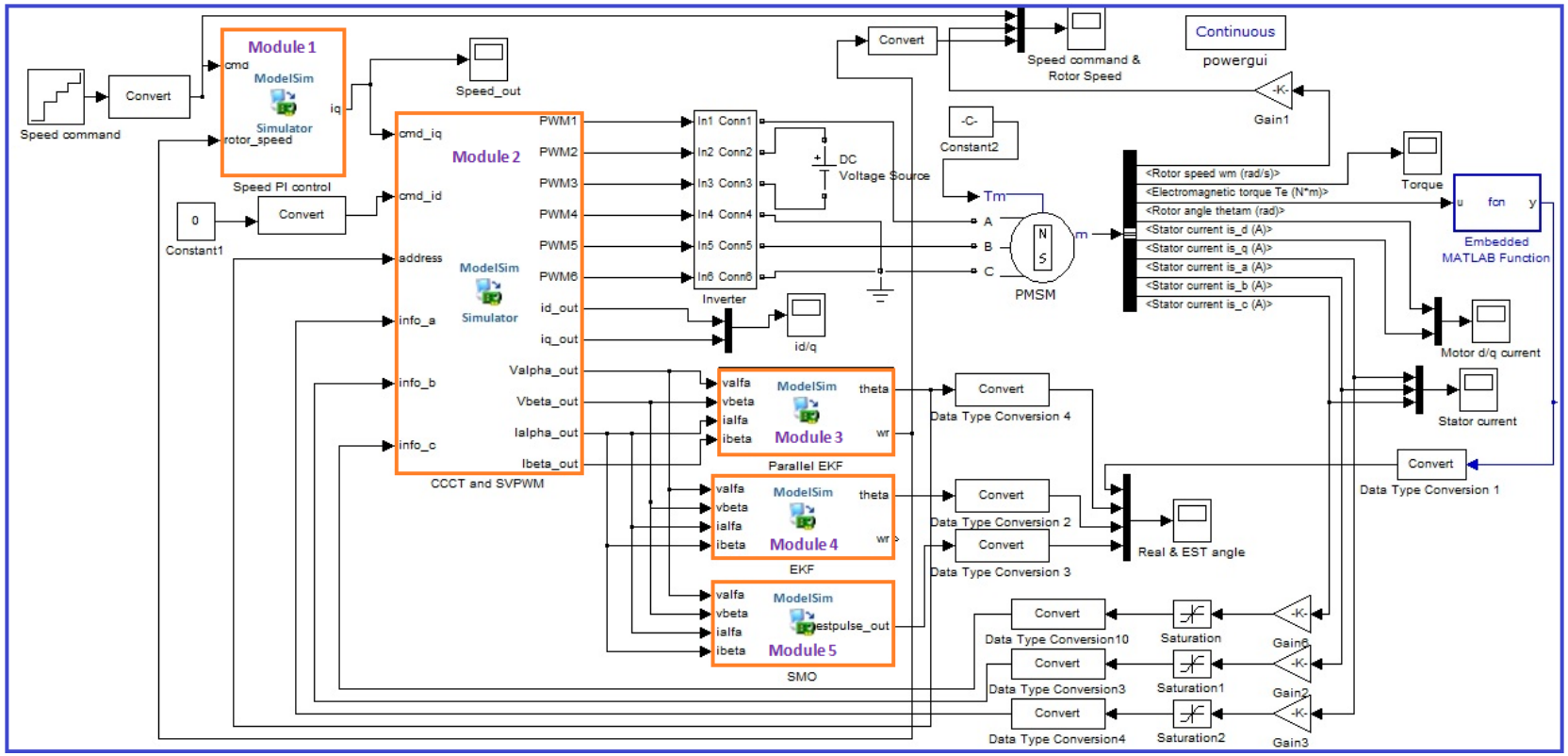


Figure 2.12 : The Simulink/ModelSim co-simulation architecture for sensorless speed control system

Chapter 3

A Brief Survey on PMSM Drives

3.1 Introduction

Innovations in magnetic materials, semiconductor control devices, and control strategies keep on increasing the popularity of permanent magnet synchronous motors in drive applications.

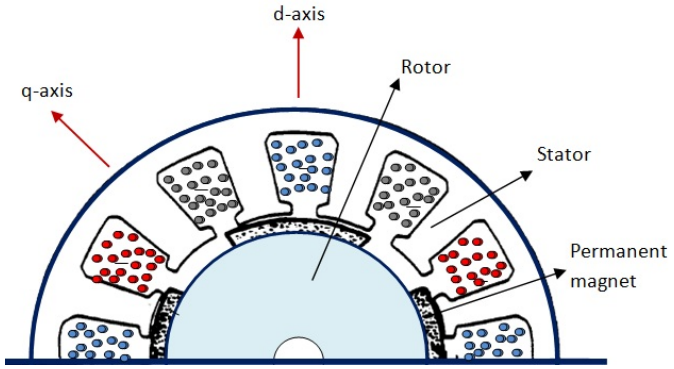
In this chapter, a brief survey of PMSM drives is presented. It begins with an overview of PMSM in section 3.2 and then FPGA based realization of current vector control of PMSM drive is described in section 3.3 and 3.4. The chapter concludes with a brief review on control techniques for PMSM drives in section 3.5, which related to the main studies in this thesis; these are control schemes for motion axis control systems and sensorless estimation methods for PMSMs.

3.2 Overview of PMSM [82]

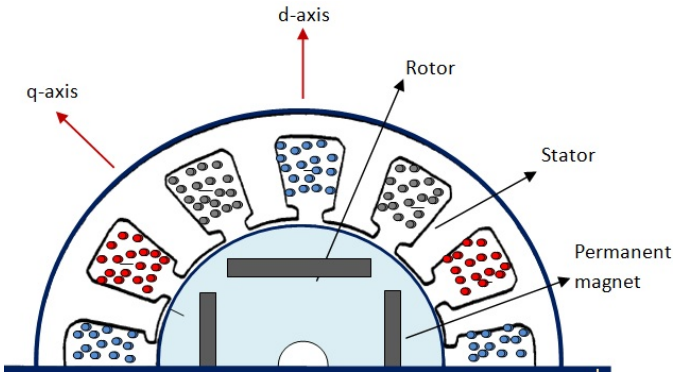
With the advantages of superior power density, high performance motion control with fast speed and enhanced accuracy, the PMSM has been increasingly used in robotics, precision machining and many automation processes. These drives are the best alternative for high-performance applications and are expected to see expanded use as manufacturing costs decrease.

In permanent magnet motors, the magnets can be placed on the rotor surface, the so-called surface-mounted permanent magnet (SPM) motors. Alternatively, they can be

buried inside the rotor, as in the case of an interior permanent magnet (IPM) motor (see Figure 3.1, which depicts the cross section of a SMP motor and an IPM motor). The role of the magnets is to produce a permanent flux in the air-gap between the rotor



(a)



(b)

Figure 3.1 : Cross-section of PMSM (a) Surface mounted PMSM, and (b) Interior PMSM.

and the stator. The SPMs have a smooth air gap, whereas saliency arises in the IPMs. The utilization of permanent magnets in the rotor rather than by a DC field winding facilitates efficiency, eliminates the need for brushes and slip rings, and eliminates the electrical rotor dynamics that make control difficult especially vector control. The permanent magnets have the disadvantages of adding significant capital cost to the drive, although the long term cost can be less through improved efficiency. The PMSM

also has the disadvantages of requiring rotor position feedback by either direct means or by a suitable estimation system. Since many other high performance drives utilize position feedback, this is not necessarily a disadvantage. Another disadvantageous aspect of the PMSM is cogging torque, which depends on the rotor position and exists when there are no stator currents resulting from the interaction of the magnets and the stator teeth. There is also another pulsating torque due to space harmonics and time harmonics. However, this cogging and pulsating effect can be virtually eliminated either by appropriate design of the machine or by electronic mitigation. There are variations of the stator design that are possible, particularly in regard to slot skewing and tooth shape, that is, the stator slots or the magnet can be skewed by one slot pitch, shaping of the magnets, coordinating the design of the number of stator slots, slot opening and the magnet dimensions. There is a wide variety of motor designs, each of which has its own performance and cost considerations.

Regarding the materials to retain magnetism, there are a number of diverse magnet materials that are commonly used. Ferrite and cobalt-samarium (SmCos, Sm2Coi7) magnets have previously been used in permanent magnet machines. Ferrite is a reasonably priced but of less magnetic potential material that is frequently used. The rare-earth (Neodymium-Iron-Boron - NdFeB) magnets have higher energy density and coercivity that enable permanent magnet synchronous machines to have wider applications. Consequently the manufacturing process is costly and complex. The field strength of magnets decreases with an increase in temperature. The field strength of rare earth magnets may be up to 1 T (Tesla) but it can saturate in the teeth and increase iron losses. SmCo magnets are particularly resistant to temperature but are comparably very expensive. Sintered NdFeB magnets have a stronger residual field and lower cost than SmCo magnets, but are less temperature resistant. Bonded

NdFeB magnets are not quite as strong as SmCo, but are less expensive and are more easily shaped. Ferrite magnets are very common for lower-performance motors. Both radial and parallel magnetization are commonly used, depending on the application. The particular choice of magnets and other design factors is important, but does not directly influence the basic principles of power converter control.

3.3 Current Vector Control of PMSM Drives

Vector control is one of the earliest closed loop control schemes developed to control electric machine drives. It is widely used in industry for many electrical machines such as permanent magnet, induction and switched reluctance machines. Current/torque controlled and speed controlled vector drives are some of the commonly used vector drives for practical applications [67],[69]. Details of the vector controlled PM machine drive implementation are fully elaborated in this section .

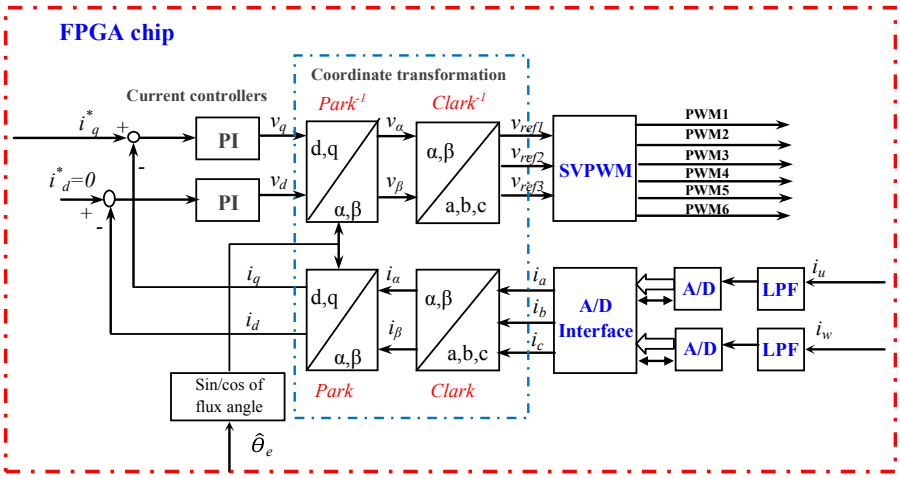


Figure 3.2 : Block diagram of current loop

The configuration of the current loop using vector control for a PMSM, including a PI current controller, $Park$, $Park^{-1}$, $Clark$, $Clark^{-1}$, SVPWM coordinate trans-

formation, detection of current and encoder signals, is depicted in the central block of Figure 3.2. Under vector control, i_d=0, the decoupled characteristics of the motor allows for control of only the q-axis current.

3.3.1 Coordinate Transformations

The rotor speed of PMSM is synchronous with the stator electrical speed, i.e. $\omega_e = (p/2)\omega_r$ and $\theta_e = (p/2)\theta_r$, where θ_e is the electrical angle and p is the number of magnetic poles. The coordinate transformations between the stationary $a-b-c$ frame, stationary $\alpha-\beta$ frame and synchronously rotating $d-q$ frame [70], shown in Figure 3.3, are described as follows.

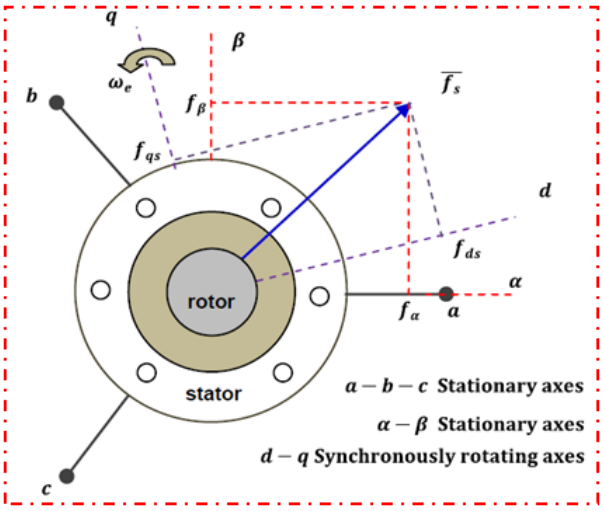


Figure 3.3 : Transformations between stationary and synchronously rotating axes

-*Clark*: stationary $a-b-c$ frame to stationary $\alpha-\beta$ frame

$$\begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{-1}{3} & \frac{-1}{3} \\ 0 & \frac{-1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix}. \tag{3.1}$$

-$Clark^{-1}$: stationary $\alpha - \beta$ frame to stationary $a - b - c$ frame

$$\begin{bmatrix} f_a \\ f_b \\ f_c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} \\ \frac{-1}{2} & \frac{-\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix}. \tag{3.2}$$

-$Park$: stationary $\alpha - \beta$ frame to rotating $d - q$ frame

$$\begin{bmatrix} f_{ds} \\ f_{qs} \end{bmatrix} = \begin{bmatrix} \cos\theta_e & \sin\theta_e \\ -\sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix}. \tag{3.3}$$

-$Park^{-1}$: Rotating $d - q$ frame to stationary $\alpha - \beta$ frame

$$\begin{bmatrix} f_\alpha \\ f_\beta \end{bmatrix} = \begin{bmatrix} \cos\theta_e & -\sin\theta_e \\ \sin\theta_e & \cos\theta_e \end{bmatrix} \begin{bmatrix} f_{ds} \\ f_{qs} \end{bmatrix}. \tag{3.4}$$

3.3.2 Space Vector Pulse Width Modulation

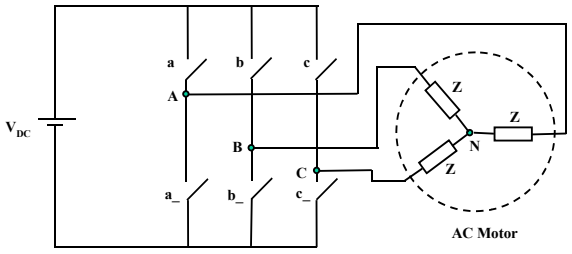


Figure 3.4 : 3-phase power converter and AC motor

Space vector pulse width modulation (SVPWM) is a special switching scheme of a 3-phase power converter with the six power transistors. According to the ON-OFF switching of upper transistors in Figure 3.4, there are eight possible combinations. The eight vectors are called basic space vectors and they are denoted by $U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}, O_{000}$ and O_{111}, which are shown in Figure 3.5.

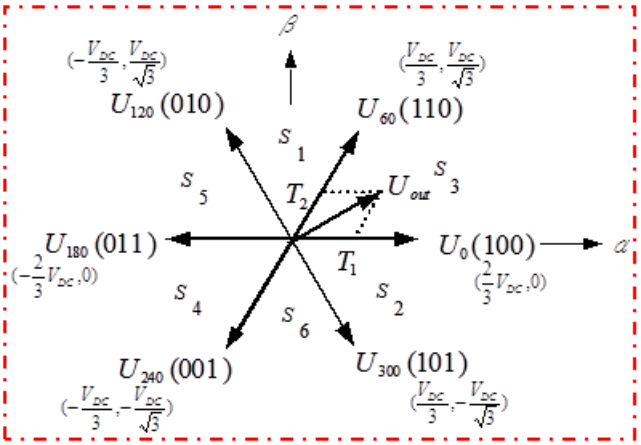


Figure 3.5 : Basic vector space and switching patterns

Using the $Clark$ transformation, the project values in $\alpha - \beta$ axis for six basic space vectors can be obtained. The SVPWM technique is applied to approximate the reference voltage and it combines the switching pattern with the basic space vectors. Therefore, the motor-voltage vector will be located at one of the six sectors $(S_3, S_1, S_5, S_4, S_6, S_2)$ at any given time. Thus, for any PWM period, it can be approximated by the vector sum of two vector components lying on the two adjacent basic vectors, as the following:

$$U_{out} = \frac{T_1}{T}U_x + \frac{T_2}{T}U_{x+60} + \frac{T_0(O_{000} or O_{111})}{T}, \tag{3.5}$$

where $T = T_0 - T_1 - T_2$ and T is half of PWM carrier period.

Computation procedures of the SVPWM design:

Step 1: Determination of the sector according to the following rule. We first modified the $Clark^{-1}$ transformation as follows,

$$\begin{bmatrix} V_{refx} \\ V_{refy} \\ V_{refz} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{-1}{2} & \frac{\sqrt{3}}{2} \\ \frac{-1}{2} & \frac{-\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_\beta \\ v_\alpha \end{bmatrix}, \tag{3.6}$$

where V_{refx},V_{refy}, V_{refz} are the input signals of the SVPWM block circuit in Figure 3.2. Then we can determine the sector according to the following rules:

$$If \;\; V_{refx} > 0 \;\; then \;\; a = 1 \;\; else \;\; a = 0, \tag{3.7}$$

$$If \;\; V_{refy} > 0 \;\; then \;\; b = 1 \;\; else \;\; b = 0, \tag{3.8}$$

$$If \;\; V_{refz} > 0 \;\; then \;\; c = 1 \;\; else \;\; c = 0, \tag{3.9}$$

$$Sector = a + 2b + 4c. \tag{3.10}$$

Step 2: Calculation of T_x, T_y and T_z from (3.11)

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \frac{\sqrt{3}T}{V_{dc}} \begin{bmatrix} V_{refx} \\ V_{refy} \\ V_{refz} \end{bmatrix}. \tag{3.11}$$

Step 3: Determination of T_1 and T_2 from Table 3.1.

Table 3.1 : T1 and T2 in all specific sectors

	S_3	S_1	S_5	S_4	S_6	S_2
T_1	$-T_Z$	T_Z	T_X	$-T_X$	$-T_Y$	T_Y
T_2	T_X	$-T_Y$	$-T_Y$	T_Z	$-T_Z$	$-T_X$

Step 4: Determination of the duty cycle $T_{a_{on}}$, $T_{b_{on}}$, $T_{c_{on}}$ from (3.12) to (3.14).

$$T_{a_{on}} = (T - T_1 - T_2)/2 = T_0/2, \tag{3.12}$$

$$T_{b_{on}} = T_{a_{on}} + T_1, \tag{3.13}$$

$$T_{c_{on}} = T_{b_{on}} + T_2. \tag{3.14}$$

Step 5: Assignment of duty cycles to CMPR1, CMPR2 and CMPR3 from Table 3.2.

Table 3.2 : Assigning duty cycle to CMPx in any sector

	0^o-60^o	60^o-120^o	120^o-180^o	180^o-240^o	240^o-300^o	300^o-360^o
	S_3	S_1	S_5	S_4	S_6	S_2
CMPR1	$T_{a_{on}}$	$T_{b_{on}}$	$T_{c_{on}}$	$T_{c_{on}}$	$T_{b_{on}}$	$T_{a_{on}}$
CMPR2	$T_{b_{on}}$	$T_{a_{on}}$	$T_{a_{on}}$	$T_{b_{on}}$	$T_{c_{on}}$	$T_{c_{on}}$
CMPR3	$T_{c_{on}}$	$T_{c_{on}}$	$T_{b_{on}}$	$T_{a_{on}}$	$T_{a_{on}}$	$T_{b_{on}}$

3.4 Digital Circuit Design of Current Vector Control

Current loop in Figure 3.2 includes current controllers and coordinate transformation (CCCT), SVPWM generation, ADC read-in and transformation. The sampling frequency of current control is designed with 16 kHz.

3.4.1 Current Controller and Coordinate Transformations

In order to calculate park transformation and inverse park transformation, the calculation of the sine and cosine value is needed for obtaining electrical angle. Sine table and cosine table are stored in ROM in advance by MIF (Memory Initialization File) file for sine/cosine look up. The data type is 12-bit length with Q11 format and $2's$ complement operation, so $\frac{1}{\sqrt{3}}$ in equation (3.1) and $\frac{\sqrt{3}}{2}$ in equation (3.2) are corresponding to:

$$\frac{1}{\sqrt{3}} \Rightarrow scaling(Q11) \Rightarrow 10010011110 \tag{3.15}$$

$$\frac{1}{\sqrt{3}} \Rightarrow normalization \Rightarrow \frac{1}{2} + \frac{1}{16} + \frac{1}{128} + \frac{1}{256} + \frac{1}{512} + \frac{1}{2048} = 0.577148437 \tag{3.16}$$

$$\frac{\sqrt{3}}{2} \Rightarrow scaling(Q11) \Rightarrow 11011101101 \tag{3.17}$$

$$\frac{\sqrt{3}}{2} \Rightarrow normalization \Rightarrow \frac{1}{2}+\frac{1}{4}+\frac{1}{16}+\frac{1}{32}+\frac{1}{64}+\frac{1}{256}+\frac{1}{512}+\frac{1}{2048} = 0.866210937 \tag{3.18}$$

FSM Implementation of CCCT

To simplify FPGA resource while implementing the CCCT, the FSM is used to design the circuit of $Park$, $Park^{-1}$, $Clark$, $Clark^{-1}$, and PI controllers in Figure 3.2. All hardware calculations are divided into 24 states shown in Figure 3.6. The internal circuit of CCCT performs the function of two PI controllers, table look-up for sin/cos function and the coordinate transformation for $Clark$, $Park$, inverse $Park$, modified inverse $Clark$. The CCCT circuit designed by FSM is shown in Figure 3.6, which uses one adder, one multiplier, a one-bit left shifter, a look-up-table and manipulates 24 steps machine to carry out the overall computation. The data type is 12-bit length with Q11 format and 2s complement operation. In Figure 3.6, steps $S_0 - S_1$ is for the look-up sin/cos table; steps $S_2 - S_5$ and $S_5 - S_8$ are for the transformation of $Clark$ and $Park$, respectively; steps $S_9 - S_{14}$ is for the computation of $d-$ and $q-$ axis PI controller; and steps $S_{15} - S_{19}$ and $S_{20} - S_{23}$ represent the transformation of the inverse $Park$ and the modified inverse $Clark$, respectively. The operation of each step in FPGA can be completed within 40 ns (25 MHz clock); therefore total 24 steps need 0.96 μs operation time. Although the FSM method needs more operation time than the parallel processing method in executing CCCT circuit, it does not lose any control performance in the overall system because the 0.96 μs operation time is much less than the designed sampling interval, 62.5 μs (16 kHz) of the current control loop. To prevent numerical overflow and alleviate windup phenomenon, the output values of PI controller are both limited within a specific range.

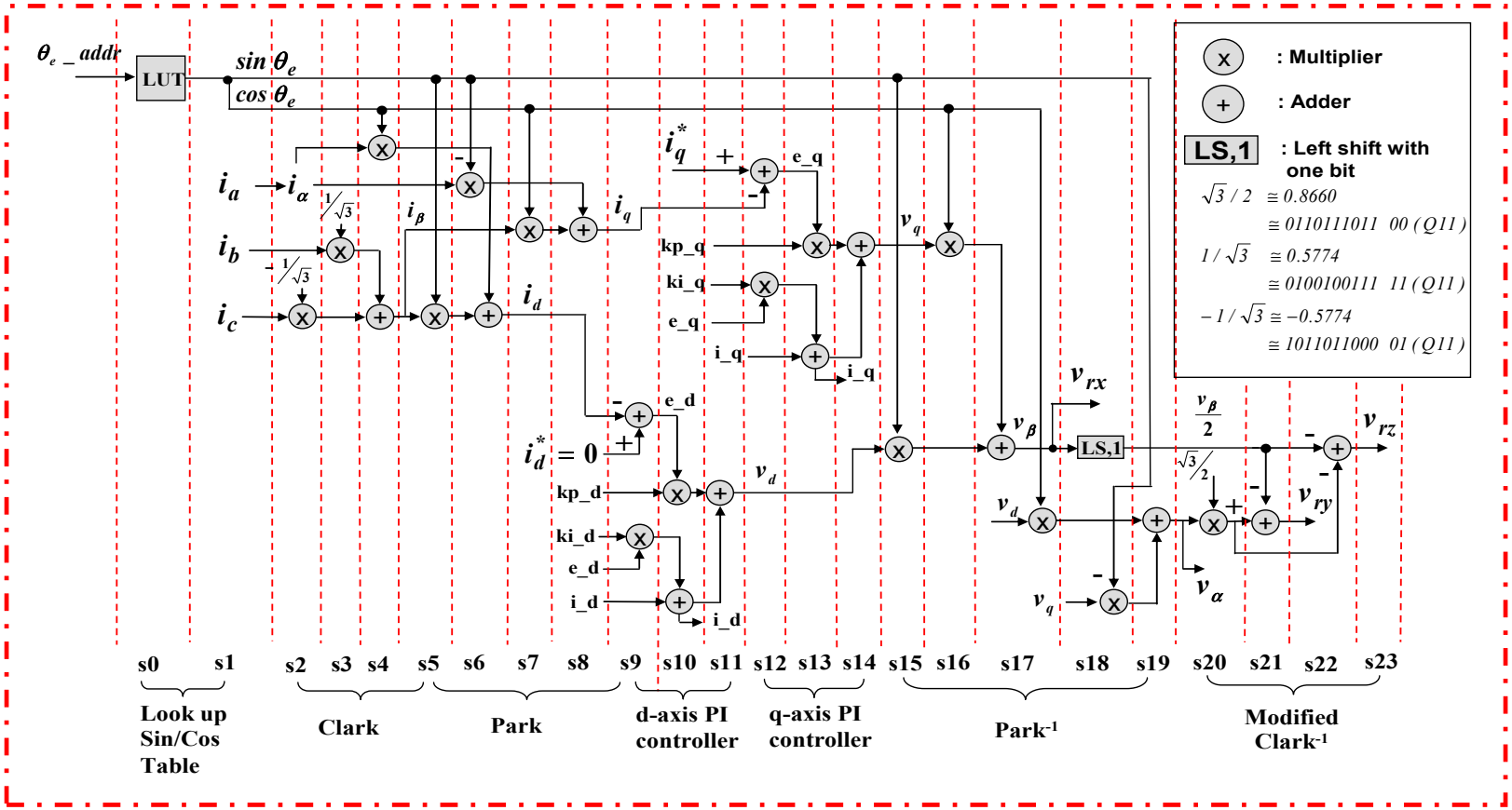


Figure 3.6 : State machines for describing the CCCT

3.4.2 Circuit of SVPWM Generation

Figure 3.7 illustrates the process of generating IGBT control signals. After conversing from two axis coordinate to three axis coordinate (modify $Clark^{-1}$), the three-phase reference output voltages (v_{ref1}, v_{ref2}, v_{ref3}), are inputs of the SVPWM block. After five steps of the calculation of S1 - S5 program shown in Figure 3.8, the outputs are CMPR1, CMPR2, and CMPR3. The value of triangular wave is compared with these outputs. Each comparison has a pair of results; for example, comparator (1) in Figure 3.8 has PWMEA-1 and PWMEA-2 which join with Dead-band unit to create PWM1 and PWM2. Similarly, PWM3, PWM4, PWM5, and PWM6 are created.

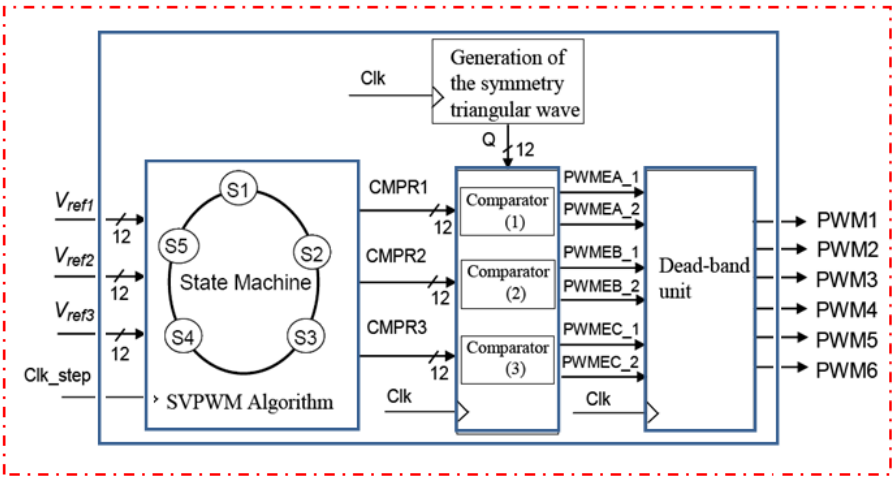
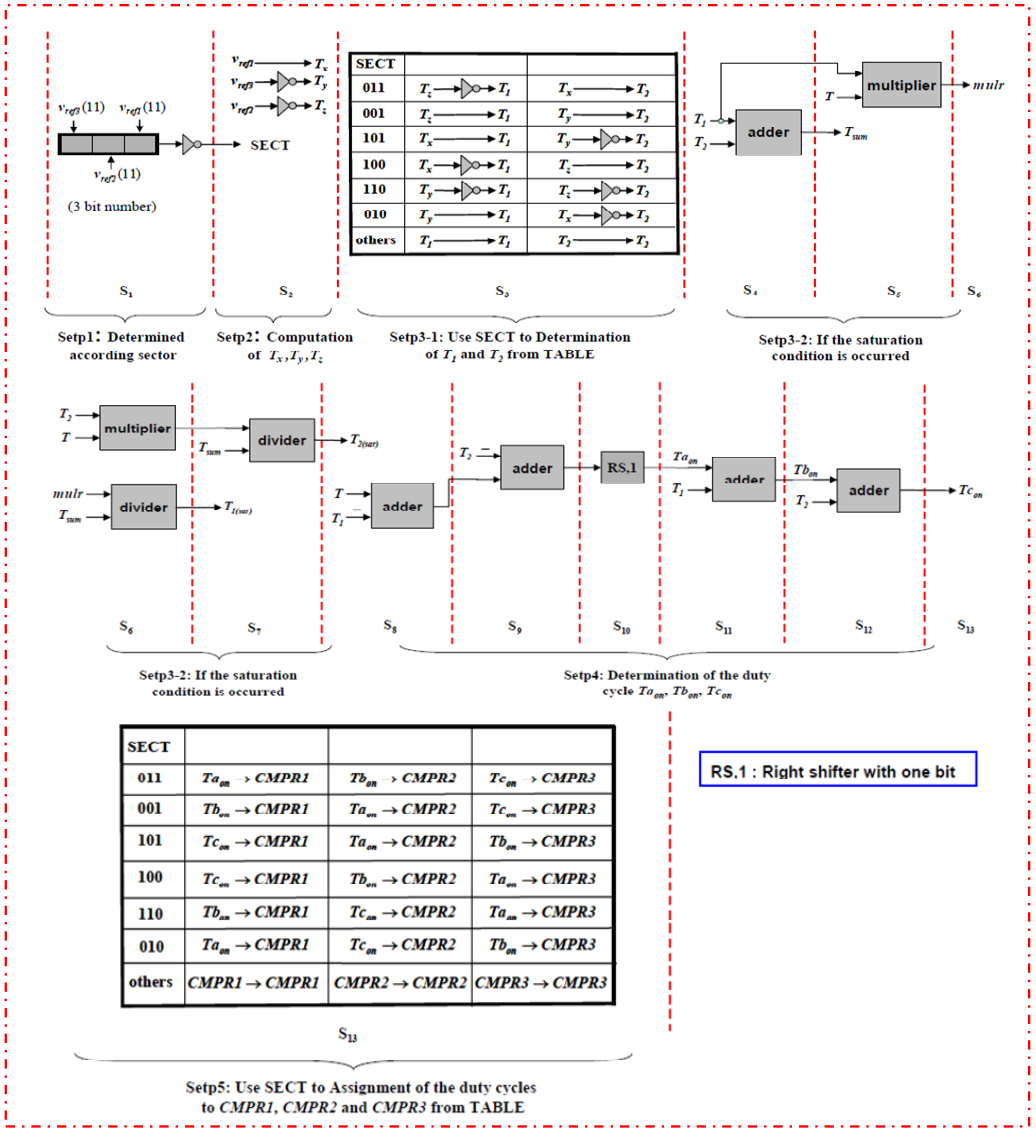


Figure 3.7 : Circuit of SVPWM generation



SECT			
011	$T_z \to T_1$	$T_x \to T_2$	
001	$T_z \to T_1$	$T_y \to T_2$	
101	$T_x \to T_1$	$T_y \to T_2$	
100	$T_x \to T_1$	$T_z \to T_2$	
110	$T_y \to T_1$	$T_z \to T_2$	
010	$T_y \to T_1$	$T_x \to T_2$	
others	$T_1 \to T_1$	$T_2 \to T_2$	

SECT			
011	$Ta_{on} \to CMPR1$	$Tb_{on} \to CMPR2$	$Tc_{on} \to CMPR3$
001	$Tb_{on} \to CMPR1$	$Ta_{on} \to CMPR2$	$Tc_{on} \to CMPR3$
101	$Tc_{on} \to CMPR1$	$Ta_{on} \to CMPR2$	$Tb_{on} \to CMPR3$
100	$Tc_{on} \to CMPR1$	$Tb_{on} \to CMPR2$	$Ta_{on} \to CMPR3$
110	$Tb_{on} \to CMPR1$	$Tc_{on} \to CMPR2$	$Ta_{on} \to CMPR3$
010	$Ta_{on} \to CMPR1$	$Tc_{on} \to CMPR2$	$Tb_{on} \to CMPR3$
others	$CMPR1 \to CMPR1$	$CMPR2 \to CMPR2$	$CMPR3 \to CMPR3$

Figure 3.8 : State machines for describing SVPWM circuit

3.5 Review on Control Techniques for Motion Axis Control Systems

In modern control systems, single or multi-axis motion control systems have become important elements. Motion axis control system is usually driven by linear motors and the servo controllers for coordination of the positioning and trajectory tracking tasks [32],[42],[51],[69],[99],[100]. These are widely used in manufacturing equipment, transportation, and robots due to their high-speed and high-accuracy linear motions. There are three main categories for linear motors; these are induction, permanent magnet, and stepping motors. As is the case with mechanical transmission using rotary motors, linear stepping motors and induction motors not only seriously reduce linear motion speed and dynamic response, but also introduce backlash, large frictional and inertial loads [52]. They are not necessarily of high performance but low cost. However, they will not generate the high dynamic performance that permanent magnet motors will do. Due to the advantages of superior power density, high performance motion control with fast speed and enhanced accuracy, the permanent magnet linear synchronous motor (PMLSM) has been increasingly used in motion control systems. However, the PMLSM does not use conventional gears or ball screws, so uncertainties such as parameter variations, external load disturbance, friction force and unknown dynamics in the drive system greatly affect servo performance and diminish the performance of the predesigned PMLSM driving system. Therefore, many control techniques such as robust control, adaptive control, fuzzy control and neural network control have been developed for the axis control problem to improve tracking performance in machine systems [32],[42],[45],[46],[69],[81],[86],[97]. First, reference [81] presented robustness for the output tracking control problem for transportation

within a manufacturing system. A linearised model for the motor was provided by system identification using a dynamic signal analyser. In [45], a robust position control is designed to reduce the control performance degradation due to system parameter changes and the effect of cogging force. The dynamic model of the motor drive was estimated from measurements. [45] and [81] to simultaneously proposed robust control designs for PMLSM drives, in which the controllers are based on the parameters and model of motor; however, it is not easy to measure the parameters of PMLSM in real world applications. To cope with this difficulty, some studies have used conventional adaptive control for the PMLSM [86],[97]. Conventional adaptive control method does not depend on accurate knowledge of the parameters of the motor. Reference [86] presented the design and realization of a robust adaptive algorithm to suppress the friction and ripple force phenomenon associated with a linear motor. Unfortunately, this research focuses on friction and force ripple compensation but not on position loop. [97] studies the high performance robust motion control of PMLSM, which has negligible electrical dynamics, but this research only focuses on self-tuning algorithm.

In recent years, intelligent control techniques such as fuzzy control, neural networks control, adaptive fuzzy control, and neural fuzzy controller have been proposed and applied to the position control of servo motor drives to yield high operating performance [32],[42],[46],[48],[50],[51],[69],[99],[100]. Reference [32] applied a fuzzy logic scheme for on-line tuning the PI controller parameters for the speed- and position-loop control of PMLSM drive. Fuzzy control has an advantage in simple implementation from using heuristic inference, but it is not easy to obtain an optimal set of fuzzy membership functions and rules. To address this issue, the concept of incorporating fuzzy logic into a neural network has been developed thanks to the combination of the

learning ability of the neural network and the advantage of the rule-base structure of fuzzy logic [41],[48],[50],[51],[59],[79],[100]. For instance, if the domain knowledge or human expert knowledge of the controller is only partially known, such integration can automatically be incorporated into the adaptive system, in which the remaining unknown control knowledge can then be obtained through the learning process of a neural network [79]. Fixed membership functions with a limited number of fuzzy rules are proposed in [41], where the defuzzifier parameters can be tuned by using an adaptive mechanism. In many neural-control paradigms, the back-propagation algorithm has been popularly used. However, being a gradient descent method, such an algorithm has many problems associated with it which include network paralysis, local minima and slow convergence [59]. Nevertheless, the execution of the intelligent control techniques requires a lot of computations, so implementation of these highly complex control algorithms depends on the PC systems in most studies [48],[50]. In the development of very large scale integration technology, the FPGA has brought more attention than before. Many practical applications in inverter [89],[104] and motor control [47],[105] have been studied. [89] and [104] respectively have presented an SVPWM for three-phase inverters by using FPGA. In [105], an FPGA-realization of a speed servo controller of PMSM has been proposed, in which a conventional PI controller was adopted in the speed loop of a PMSM drive. Reference [47], based on FPGA, has presented an adaptive sliding-mode control for a linear induction motor drive. However, these studies of the servo control system of an inverter or AC motor is only by FPGA hardware implementation with the use of simple computation algorithms.

3.6 Review on Sensorless PMSM Estimation Techniques

In most real world applications, conventional motor control needs a position sensor or an optical encoder to measure the rotor position for feedback to the controller to ensure speed control precision. However, a position sensor presents some disadvantages affecting the cost, reliability, and noise immunity and is subject to such conditions as humidity and corrosion [88]. In recent years, the sensorless control of PMSM drives has become an important topic, and various sensorless control strategies have been investigated [1],[10],[17],[20],[21],[67],[70],[71],[73],[91],[102],[103]. References [17] and [102] have conducted a comprehensive literature review of estimating the rotor position to alleviate the need of physical sensors, which is the key to achieving rotor position and speed of sensorless PMSM drives. There are two main streams of research: Saliency- and model-based methods.

- In saliency-based methods, the detection of rotor position can be carried out by using the additional signals (current [75] or voltage [1],[11]) injected into the motor. The applied signals interact with the rotor saliency the motor, rotor position information is determined by processing the resulting response (voltage or current).

- Model-based methods in which fundamental-frequency models, measured stator currents, and voltages aimed to estimate the rotor position information thanks to the estimation of the back electromotive force (EMF) induced in the motor windings. These methods show good performance for solutions in the medium and high speed range since the magnitude is proportional to the rotor speed [67],[70],[71].

In the following, we also provide a survey of the most recent works focus on model-based estimation methods.

3.6.1 PMSM Drive Model

The typical model of PMSM is expressed in $d-q$ synchronous rotating reference frame as follows,

$$\frac{di_d}{dt} = -\frac{r_s}{L_d}i_d + \omega_e\frac{L_q}{L_d}i_q + \frac{1}{L_d}v_d, \tag{3.19}$$

$$\frac{di_q}{dt} = -\omega_e\frac{L_d}{L_q}i_d + -\frac{r_s}{L_q}i_q + \frac{1}{L_q}v_q - \omega_e\frac{\lambda_m}{L_q}, \tag{3.20}$$

where L_d, L_q are the d and q axis inductance; v_d, v_q are the d and q axis voltages; i_d, i_q, are the d and q axis currents, r_s is the phase winding resistance; ω_e is the rotating speed of magnet flux; and λ_m is the permanent magnet flux linkage.

The coupled and nonlinear model of PMSM under field orientation can be expressed in $d-q$ axes by,

$$\begin{bmatrix} v_d \\ v_q \end{bmatrix} = \begin{bmatrix} r_s + sL_d & -\omega_e L_q \\ \omega_e L_q & r_s + sL_d \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \{(L_d - L_q)(-\omega_e i_d - \dot{i}_q) + \omega_e\lambda_f\} \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{3.21}$$

which can be re-transformed to the $\alpha-\beta$ axes as follows:

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} r_s + sL_d & \omega_e(L_d - L_q) \\ -\omega_e(L_d - L_q) & r_s + sL_d \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \{(L_d - L_q)(-\omega_e i_d - \dot{i}_q) + \omega_e\lambda_f\} \begin{bmatrix} -\sin\theta_e \\ \cos\theta_e \end{bmatrix}. \tag{3.22}$$

This model is useful and can be applied to any type of synchronous motors such as the surface mounted PMSM when ($L_d = L_q$), or interior PMSM when ($L_d < L_q$).

3.6.2 Survey on Model-based Estimation Methods

Back EMF-based Method

The estimators where the estimated quantity is the back-EMF of PMSM model in equation (3.21) or the extended-EMF (EEMF) components in equation (3.22) including the rotor position information. The second term on the right side of (3.22) is defined as the EEMF,

$$E = \begin{bmatrix} E_\alpha \\ E_\beta \end{bmatrix} = \{(L_d - L_q)(-\omega_e i_d - \dot{i}_q) + \omega_e \lambda_f\} \begin{bmatrix} -\sin\theta_e \\ \cos\theta_e \end{bmatrix}. \tag{3.23}$$

The back-EMF which is estimated based of the voltage and current measurements, is supposed to be accurate and then contains the position. The position of magnet can be obtained from its phase and the rotor position estimation generally calculated from the EEMF as,

$$\hat{\theta}_e = tan^{-1}\left(-\frac{E_\alpha}{E_\beta}\right) = tan^{-1}\left(-\frac{v_\alpha - (r_s + sL_d)i_\alpha - \omega_e(L_d - L_q)i_\beta}{v_\beta - (r_s + sL_d)i_\beta - \omega_e(L_d - L_q)i_\alpha}\right). \tag{3.24}$$

The method is presented in [16],[22], which can be suitable to both surface mounted or interior PMSM. In addition, this method is fast and straightforward without using complex observers. However, the EEMF is influenced by stator currents, which vary during motor transient state, so the performance of this method is subject to the accuracy of the sensed current/voltage and machine parameters.

Flux-Based Method [74]

The flux linkage is estimated from measured voltages and currents. At steady state, where $di_\alpha/dt = 0$ and $di_\beta/dt = 0$, the stator and rotor flux vectors rotate synchronously. Therefore, if the position angle of the stator flux can be calculated, the rotor flux angle can also be determined, which is the same as the rotor position angle. The voltage and current components in the stator stationary reference frame in equation (3.22) and the voltage equation of the machine are taken to compute the

stator flux linkage as follows,

$$\begin{cases} \psi_{s\alpha} = \int (v_\alpha - r_s i_\alpha) dt \\ \psi_{s\beta} = \int (v_\beta - r_s i_\beta) dt \end{cases}, \tag{3.25}$$

and then rotor flux linkage is calculated by

$$\begin{cases} \psi_{r\alpha} = \psi_{s\alpha} - L_s i_\alpha \\ \psi_{r\beta} = \psi_{s\beta} - L_s i_\beta \end{cases}, \tag{3.26}$$

where $\psi_{s\alpha}, \psi_{s\beta}$ and $\psi_{r\alpha}, \psi_{r\beta}$ are respectively the stator and rotor flux linkages in fixed coordinates.

Based on the initial position, machine parameters, and relationship between the flux linkage and rotor position, the rotor position can be estimated with

$$\hat{\theta}_e = tan^{-1}\left(\frac{\psi_{r\alpha}}{\psi_{r\beta}}\right) = tan^{-1}\left(\frac{\psi_{s\alpha} - L_s i_\alpha}{\psi_{s\beta} - L_s i_\beta}\right). \tag{3.27}$$

The advantage of this method is that it may work well in the steady state, but the transient performance is usually unsatisfactory. The accuracy of the flux-based methods highly depends on the quality and accuracy of the voltage and current measurements. Since integrators are needed in this method, it has also an error accumulation problem for integration at low speeds [74].

Inductance-Based Method

This method is based on the difference between the $d-$ and $q-$ inductance since it is a function of the rotor position. The phase inductance can be calculated analytically from the instantaneous voltage and current information. Then the rotor position can be found with the calculated phase inductance. In a PMSM control system, if the switching frequency is high enough, then the variation of inductance with the rotor position can be neglected during one switching period. With this assumption the

instantaneous voltage equation for the phase a of the motor can be obtained from

$$v_a = R_a i_a + L_{sa}\frac{di_a}{dt} + e_a, \tag{3.28}$$

where all of the variables are phase a quantities, v_a is the terminal phase voltage, i_a is the phase current, L_{sa} is the synchronous inductance, R_a is the phase resistance, and e_a is the back EMF.

From equation (3.28), the phase inductance can be expressed as

$$L_{sa} = \frac{v_a - R_a i_a - e_a}{s i_a} \ . \tag{3.29}$$

Based on equation (3.29), the rotor position can be obtained from a lookup table method, in which the table of stored data contains the relationship between the rotor position and calculated phase inductance. The accuracy of this method also highly depends on the quality and accuracy of the voltage and current information. Since the current and position derivatives need to be calculated in every switching cycle, the rotor position is subjected to a high level of measurement noise.

Linear State Observer

The EMF or EEMF components can be estimated by using linear state observer, in which the EMF is a kind of disturbance voltage. This method is based on the electrical model of motors and estimates the EMF. For surface mounted PMSM, the circuit equation of PMSMs in (3.22) is re-written as,

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \begin{bmatrix} r_s + sL_s & 0 \\ 0 & r_s + sL_s \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \omega_e \lambda_f \begin{bmatrix} -\sin\theta_e \\ \cos\theta_e \end{bmatrix}, \tag{3.30}$$

where $L_s \triangleq L_d = L_q$ and s is differential operator. In addition, let us define the EMF as

$$e = \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} \triangleq \omega_e \lambda_f \begin{bmatrix} -\sin\theta_e \\ \cos\theta_e \end{bmatrix}. \tag{3.31}$$

The EMF includes the position information from the flux. By estimating the EMF, it is possible to get θ_e from its phase. To observe the EMF, equation (3.30) is rewritten in the state space form by,

$$\frac{d}{dt}\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = -\frac{r_s}{L_s}.\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \frac{1}{L_s}\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} - \frac{1}{L_s}\begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix}. \tag{3.32}$$

In [9], to construct the disturbance observer, they assume that

$$\frac{d}{dt}\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} = -\frac{r_s}{L_s}.\begin{bmatrix} \hat{i}_\alpha \\ \hat{i}_\beta \end{bmatrix} + \frac{1}{L_s}\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} - \frac{1}{L_s}\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix}, \tag{3.33}$$

$$\frac{d}{dt}\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} = G.\frac{d}{dt}\begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \end{bmatrix}, \tag{3.34}$$

where the $\begin{bmatrix} \hat{i}_\alpha & \hat{i}_\beta \end{bmatrix}^T$ is the estimated current on fixed coordinates and $\begin{bmatrix} \hat{e}_\alpha & \hat{e}_\beta \end{bmatrix}^T$ is the estimate of the back EMF, and G is the observer gain matrix, which can be chosen by using the pole placement technique to obtain the desired tracking performance. Using the estimate of the back EMF, the estimated flux angle or rotor magnet position can be derived as $\hat{\theta}_e = tan^{-1}\left(-\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right)$.

This observer has been developed for interior PMSM [9],[24],[27],[58],[60]. In [58], the extended EMF in the rotating reference frame is utilized in order to estimate both position and speed, while the EEMF model in the stationary frame is used in [9].
In reference [9], the structure of the current observer is the same as that in equation (3.33), but the expression for the EMF observer in equation (3.34) is slightly different as,

$$\frac{d}{dt}\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} = A\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} + G.\frac{d}{dt}\begin{bmatrix} \hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta \end{bmatrix}. \tag{3.35}$$

Reference [27] have proposed a state observer, in which the rotor position can be obtained by estimating flux quantities. In this method, the selection of the observer gains is important to assure the stability of the observer. The variations of machine parameters slightly affect the accuracy of the estimation since the machine parameters are needed in the observers' models, e.g. the variation is because of the cross-coupling effect between the $d-$ and $q-$ axis equations. Besides, the quality of voltage and current measurements also affect the performance of the observer.

Sliding Mode Observer [70],[73]

Linear state observers are based on the linear state space equations and use linear state feedback as stated in equations (3.34) and (3.35). By using nonlinear state feedback, non-linear state observers are also good methods for the rotor position estimation.

A sliding mode observer (SMO) is a popular method for non-linear observers. In the literature, the SMO usually is constructed based on the PMSM models in the stationary reference frame [10],[18],[39],[65],[70],[73],[91]. For a surface mounted PMSM, a typical SMO is designed as:

$$\frac{d}{dt}\begin{bmatrix}\hat{i}_\alpha \\ \hat{i}_\beta\end{bmatrix} = -\frac{r_s}{L_s}.\begin{bmatrix}\hat{i}_\alpha \\ \hat{i}_\beta\end{bmatrix} + \frac{1}{L_s}\begin{bmatrix}v_\alpha \\ v_\beta\end{bmatrix} - \frac{1}{L_s}k.sign\begin{bmatrix}\hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta\end{bmatrix}, \tag{3.36}$$

where $sign(.)$ is the sign function; and k is the gain of the switching terms. Here, the sliding surface is defined by $e_i \triangleq \begin{bmatrix}\tilde{i}_\alpha & \tilde{i}_\beta\end{bmatrix}^T = \begin{bmatrix}\hat{i}_\alpha - i_\alpha & \hat{i}_\beta - i_\beta\end{bmatrix}^T$. By choosing the gain k to be large enough, i.e, $k \geq max(|e_\alpha|, |e_\beta|)$, the inequality $e_i^T.\dot{e}_i < 0$ can be reached and the SMO can induce a sliding mode, i.e, $e_i = \dot{e}_i = 0$. In this induced sliding mode, the back EMF can be estimated by:

$$\begin{bmatrix}\hat{e}_\alpha \\ \hat{e}_\beta\end{bmatrix} = k.sign\begin{bmatrix}\hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta\end{bmatrix}, \tag{3.37}$$

then the rotor position can be computed by the estimated EMFs [70],[73].

MRAS-Based Method

The model reference adaptive system (MRAS) based method has the desired state from two different models, which are connected in parallel; one is the reference model and another one is the adjustable model. The error between the two models is used to estimate the unknown parameter. In MRAS only the adjustable model should depend on the unknown parameter, the reference model is independent of the unknown parameter. The error signal is fed into the adaptation mechanism, which provides the estimated quantity which is used to tune the adjustable model. The output of the adjustable model is expected to converge with the output of the reference model under a proper adaption mechanism. In [93], the reference model is written in the following form:

$$\frac{d}{dt}x = A.x + u \ , \tag{3.38}$$

where $x = \begin{bmatrix} \hat{i}_d + \frac{\lambda_m}{L_d} \\ \hat{i}_q \end{bmatrix}$, $u = \begin{bmatrix} \frac{v_d L_d + \lambda_m}{L_d^2} \\ \frac{v_q}{L_q} \end{bmatrix}$, and $A = \begin{bmatrix} -\frac{R_s}{L_d} & \frac{L_q \omega_e}{L_d} \\ -\frac{L_d \omega_e}{L_q} & -\frac{R_s}{L_q} \end{bmatrix}$.

The adjustable model is defined as:

$$\frac{d}{dt}\hat{x} = \hat{A}.\hat{x} + u \ . \tag{3.39}$$

In the adjustable model, the estimated speed information is used as a corrective term in the estimation of matrix A. The adaptive mechanism for the speed update is expressed as:

$$\hat{\omega}_e = \int_0^t k_1[i_d\hat{i}_q - i_q\hat{i}_d - \frac{\lambda_m}{L_d}(i_q - \hat{i}_q)]d\tau + k_2[i_d\hat{i}_q - i_q\hat{i}_d - \frac{\lambda_m}{L_d}(i_q - \hat{i}_q)] + \omega_e(0), \tag{3.40}$$

using the Popov super stability theory to guarantee stability of the MARS and the convergence of the speed estimation [14]. If the tracking errors between the states of the adjustable and reference models are close to zero, the estimated speed obtained by equation (3.40) can be viewed as the actual speed. Then the rotor position can be obtained by using an integrator, i.e., $\theta_e = \int_0^t \hat{\omega}_e dt$.

Extended Kalman Filter [67]

Kalman filtering is an optimal, stochastic approach to state estimation and filtering in linear systems. An EKF [7],[8],[36],[67],[92], which is a nonlinear version of the recursive stochastic optimal Kalman filter, can be used for estimation of joint state and parameters as well as unknown disturbances in noisy environments. The EKF is an optimal estimator which minimizes the cost function $J = \sum_{n=1}^{m} E\{\tilde{x}^2(n)\}$ in the least square sense, in which $\tilde{x}(n)$ is defined by the difference between the system state $x(n)$ and its estimate $\hat{x}(n)$, i.e., $\tilde{x}(n) = x(n) - \hat{x}(n)$. The EKF algorithm can be described in two-step recursive equations: prediction and update. The EKF is a good candidate for the online estimation of the rotor position of PMSM. In the EKF algorithm, the system state variables can be selected in either the rotor reference frame [8] or the stationary reference frame [7], i.e., $x(t) = \begin{bmatrix} i_d & i_q & \omega_e & \theta_e \end{bmatrix}^T$ and $x(t) = \begin{bmatrix} i_\alpha & i_\beta & \omega_e & \theta_e \end{bmatrix}^T$, respectively. Due to the recursive stochastic optimal Kalman filter, it has great advantages in the areas of robustness to measurement noise and the inaccuracy of machine parameters [71].

3.6.3 Remarks

As mentioned, the estimation of the rotor position and speed can be achieved by using an algorithmic estimator to alleviate the need for physical sensors. Available sensorless techniques are mainly classified into two different types: the high frequency

(HF) injection method and the model-based techniques.

These model-based methods are especially effective for medium and high speed range applications. These methods also can be generally classified into two different categories: open-loop calculation (e.g., back EMF-, rotor flux-, or stator inductance-based method) and closed-loop observers (e.g., disturbance observers, SMO, MRAS, and EKF). The open-loop calculation-based estimation methods are straightforward to implement. However, the accuracy of the rotor position obtained by using these methods is restricted by the numerical resolution, which depends on the sampling frequency and control-loop frequency of the control system. The accuracy of these methods strongly depends on the accuracy of the machine parameters and voltage and current measurements. Compared to open-loop speed estimators, closed-loop observers are more robust to motor parameter variations and provide high accuracy over a wide speed range with relative simple implementation.

The HF signal injection method, using the zero crossing of the silent phases back EMF of the motor to detect its rotor position, is more suitable for low-speed operations. In this method, the HF signal may bring noise to the system to degrade its performance, and its real-time implementation would need some special hardware support, which increase the complexity and cost [1],[6],[43],[61]. The rotor position estimation can be achieved via extended Luenberger state observers [64] or nonlinear observers where the estimation error is driven to zero using the principles of, for example, sliding modes, Kalman filtering, or a MRAS. For the sensorless drive technology, while a SMO [18],[38],[65] may suffer from the chattering effect and the MRAS estimator [61] may have a difficulty in the parameter adaptation, an extended Kalman filter [30],[36],[40],[92], which is a nonlinear version for the recursive stochastic optimal Kalman filter, can be used for the estimation of the joint state and parameters as

well as unknown disturbances in noisy environments. However, EKF requires heavy on-line matrix computing and the complex computation [30]. Difficulties exist in the selection or in the inaccuracies of design parameters in the EKF equation. Therefore, an extensive number of works has been published to improve the performance of the extended Kalman filter [66],[67],[71].

Chapter 4

FPGA-based Intelligent Control for Multiple Axis Tracking Motion System

4.1 Introduction

Nowadays, manipulators or service robots are gradually replacing humans in assembly lines with high accuracy or in dangerous tasks that require positioning a tool or moving it along a trajectory. However, most automated manipulators and parallel robots have a very limited range of motion and workspace capabilities which prevent their flexibility and high precision. To overcome this problem, high performance multi-axis control systems are required in the assembly industry. Typical for multi axis control is a linear X-Y table that is driven by linear motors and servo controllers for coordination of positioning and trajectory tracking tasks. Nevertheless, the X-Y table motion is usually subject to unmodelled dynamics, external load disturbance and the cross-axis interferences, which often deteriorate the system performance during a machining process.

On the control implementation aspect, the compactness, flexibility and low cost advantages have recently given rise to the adoption of the controller-on-programmable-chip concept in designing multiple-axis motion control systems. Indeed, the FPGA technology has been widely recognized for its programmable hard-wired feature, fast time-to-market, shorter design cycle, embedding processor, low power consumption, and higher density for digital signal processing implementation with applications to

motion and robotic control systems [37],[69],[76],[77],[80],[87],[101]. In FPGA-based system developments, an essential requirement is to keep the number of logic elements and resource usage as small as possible. Therefore, intelligent control is usually more favorable than rigorous techniques for on-chip controller designs.

Many intelligent control techniques such as fuzzy sliding mode control, fuzzy control, neural network control, and adaptive control have been developed for FPGA-based application to the axis control problem to improve tracking performance in a machining and assembly process (see, e.g. [41],[49],[52] and references therein). Although fuzzy control has an advantage in simple implementation from using heuristic inference, it is not straightforward to obtain an optimal set of fuzzy membership functions and rules. To address this issue, fixed membership functions with a limited number of fuzzy rules are proposed in [41], where the defuzzifier parameters can be tuned by using an adaptive mechanism. In this study, for position control of a permanent magnet linear synchronous motor, parameters of the fuzzy controller are tuned by using a radial basis function neural network whose weights, node centers and spreads are in turn adjusted from the gradient descent method. For further performance improvement and application to robotic assembly, in this work we also propose a PI speed controller in the inner loop, refine the position controller design for the outer loop, and investigate motion performance as well as feasibility of the FPGA-based implementation of the control paradigm for an X-Y table.

This chapter addresses the integration of a multi-loop PI and neural fuzzy control system for multiple-axis motion positioning and tracking via the use of the FPGA technology. The VHDL is adopted to describe advantageous behaviors of the proposed control system. To implement the whole control paradigm, the FPGA chip is developed in the Quartus II and Nios II software environment, provided by Altera for

analysis and synthesis of VHDL designs. Simulation and experimental results of the software/hardware co-design have verified the high performance and effectiveness of the proposed chip-based control system in positioning and trajectory tracking for the X-Y table motion.

4.2 System Description and Neural Fuzzy Controller Design

The proposed axis control system for both X- and Y coordinates is shown in Figure 4.1. The control system comprises two programmable servo-control systems for both axes; each includes a motion planner, reference model, a PI speed controller in the inner loop and a neural fuzzy controller together with its tuning mechanism in the position loop which are all implemented in a single FPGA chip.

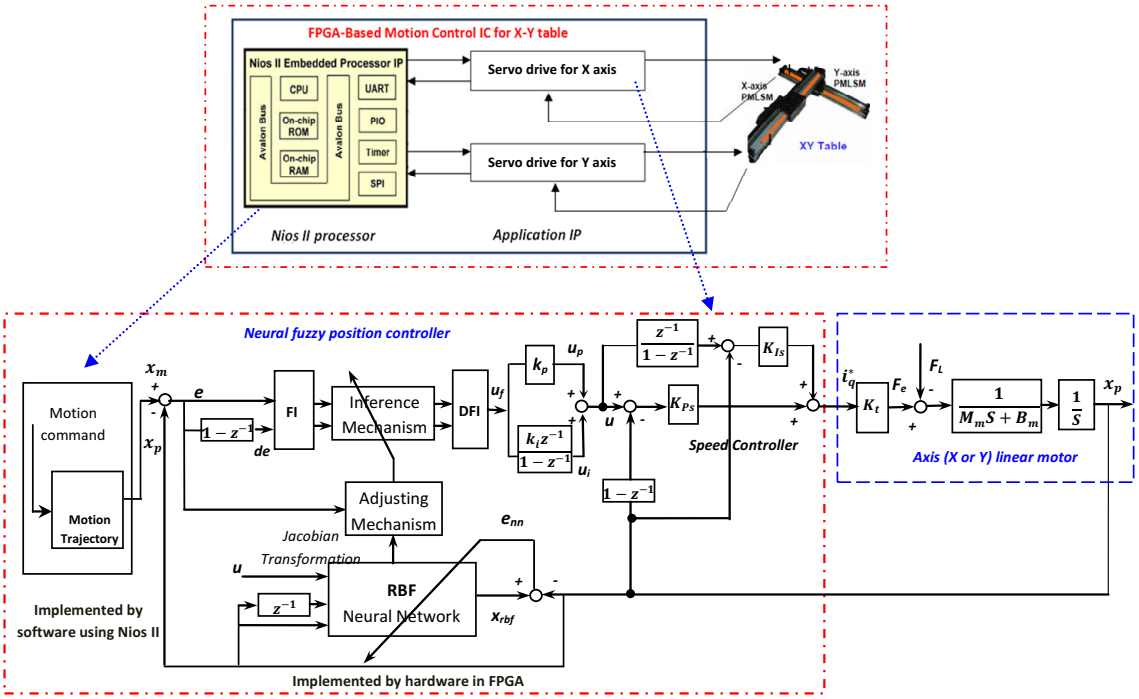


Figure 4.1 : Motion control system for X- and Y-axes

4.2.1 PMLSM Drive Model

The dynamic model of a typical PMLSM can be described in the synchronous rotating reference frame, as follows

$$\frac{di_d}{dt} = -\frac{R_s}{L_d}i_d + \frac{\pi}{\tau}\frac{L_q}{L_d}\dot{x_p}i_q + \frac{1}{L_d}v_d, \tag{4.1}$$

$$\frac{di_q}{dt} = -\frac{\pi}{\tau}\frac{L_d}{L_q}\dot{x_p}i_d - \frac{R_s}{L_q}i_q - \frac{\pi}{\tau}\frac{\lambda_f}{L_q}\dot{x_p} + \frac{1}{L_q}v_q, \tag{4.2}$$

where v_d, v_q and i_d, i_q are respectively the voltages and currents in the motor's $d-q$ frame, R_s is the phase winding resistance; L_d, L_q are the d and q axis inductance; $\dot{x_p}$ is the translator speed; λ_f is the permanent magnet flux linkage; τ is the pole pitch. The developed electromagnetic thrust force is given by

$$F_e = \frac{3\pi}{2\tau}((L_d - L_q)i_d + \lambda_f)i_q. \tag{4.3}$$

The current control of a PMLSM drive is based on a vector control approach. That is, if we control i_d to zero, the PMLSM will be decoupled, so that control of a PMLSM will become as easy as the control of a DC linear motor. After simplification and considering the mechanical load, the model of a PMLSM can be written as the following equations,

$$F_e = \frac{3}{2}\frac{\pi}{\tau}\lambda_f i_q \triangleq K_t i_q, \tag{4.4}$$

where $K_t = \frac{3}{2}\frac{\pi}{\tau}\lambda_f$, and the mechanical dynamic equation of a PMLSM is

$$F_e - F_L = M_m\frac{d^2x_p}{dt^2} + B_m\frac{dx_p}{dt}, \tag{4.5}$$

where F_e, K_t, M_m, B_m and F_L represent respectively the motor thrust force, the force constant, the total mass of the moving element, the viscous friction coefficient and the external load force.

4.2.2 Neural Fuzzy Control Design

The speed control loop has a PI controller, whose gains K_{pS} and K_{iS} can be designed by using any conventional control technique. As positioning and tracking is our focus in this study, the design of the position controller using the proposed NFC in the outer loop will be detailed here.

Fuzzy Controller

Let the tracking error and the change of the error, e, de, be defined as

$$e(k) = x_m(k) - x_p(k), \tag{4.6}$$

$$de(k) = e(k) - e(k-1), \tag{4.7}$$

where x_m are x_p are respectively the model reference and the plant translational displacement. A fuzzy position controller of the PI type is designed as follows,

$$u(k) = u_i(k-1) + (k_p + k_i\frac{T}{2})u_f(k) + k_i\frac{T}{2}u_f(k-1), \tag{4.8}$$

where k_p, k_i are the PI controller gains, u_i is the integral control output, T is the sampling period, and u_f is the output variable of a fuzzy controller (FC) with e and de as the input variables. The FC design can be summarized as:

(a) Take the e and de as the input variables of the FC, and define their linguist variables as E and dE. The linguist value of E and dE are $(A_0, A_1, A_2, A_3, A_4, A_5, A_6)$ and $(B_0, B_1, B_2, B_3, B_4, B_5, B_6)$, respectively. Each linguist value of E and dE is based on the symmetrical triangular membership function which is shown in Figure 4.2.

(b) Compute the membership degree of e and de. Only two linguistic values are excited (resulting in a non-zero membership) in any input value, and the membership

degree $\mu_{A_i}(e)$ can be derived by

$$\mu_{A_i}(e) = \frac{e_{i+1} - e}{2} \quad and \quad \mu_{A_{i+1}}(e) = 1 - \mu_{A_i}(e), \tag{4.9}$$

where $e_{i+1} \triangleq -6 + 2(i+1)$. Similar results can be obtained in computing the membership degree $\mu_{B_j}(de)$.

(c) Select the initial fuzzy control rules by referring to the dynamic response characteristics, such as,

$$IF\ e\ is\ A_i\ and\ \triangle e\ is\ B_j\ THEN\ u_f\ is\ c_{j,i}, \tag{4.10}$$

where A_i and B_j are fuzzy numbers, and $c_{j,i}$ is a real number, $i, j = 0, 1, 2, .., (N_r - 1)$, in which the number of rules, N_r, is usually an odd integer. The consequence $c_{j,i}$ will be tuned by a neural network-based adjusting mechanism. The graph of fuzzification and fuzzy rule table is shown in Figure 4.2.

(d) In Figure 4.2, since only 4 out of the total N_r^2 rules (49 fuzzy rules) are activated at a time, by using the singleton fuzzifier, product-inference rule, and central average method for defuzzification, the fuzzy control signal can be expressed as:

$$u_f(e, de) = \frac{\sum_{n=i}^{i+1}\sum_{m=j}^{j+1} c_{m,n}[\mu_{A_n}(e) * \mu_{B_m}(de)]}{\sum_{n=i}^{i+1}\sum_{m=j}^{j+1} \mu_{A_n}(e) * \mu_{B_m}(de)} \triangleq \sum_{n=i}^{i+1}\sum_{m=j}^{j+1} c_{m,n} * d_{n,m}, \tag{4.11}$$

where $d_{n,m} \triangleq \mu_{A_n}(e) * \mu_{B_m}(de)$ and $\sum_{n=i}^{i+1}\sum_{m=j}^{j+1} d_{n,m} = 1$.

RBF NN-based Adjusting Mechanism of FC

The adjustment of the fuzzy controller parameters $c_{m,n}$ is achieved with a neural network by using the least squared error criterion. For this, the cost function

$$J_e \triangleq \frac{1}{2}e^2 = \frac{1}{2}(x_m - x_p)^2 \tag{4.12}$$

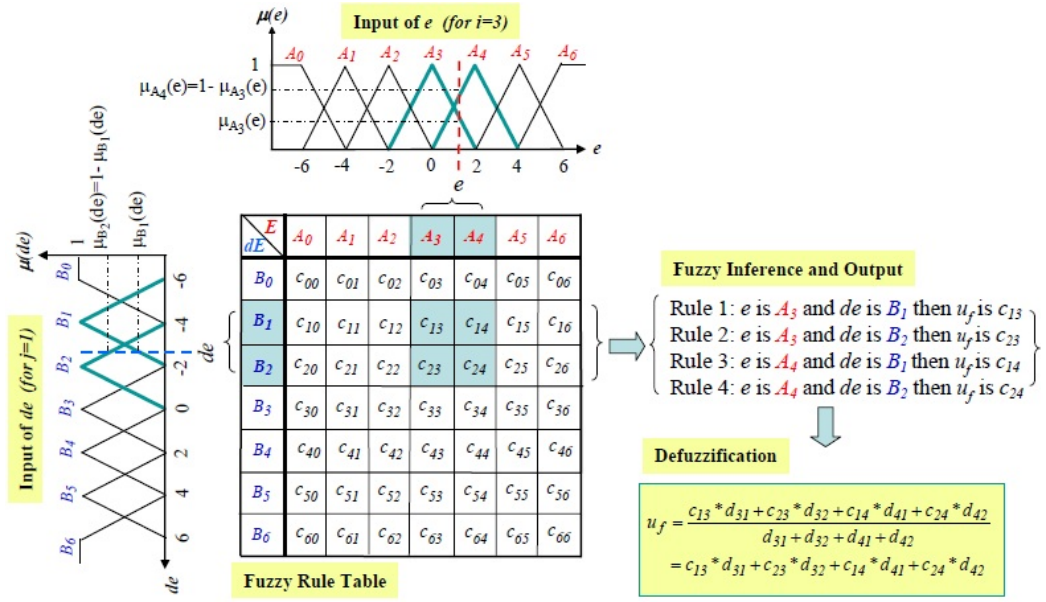


Figure 4.2 : The symmetrical triangular membership function of e and de, fuzzy rule table, fuzzy inference and fuzzification

is minimized with the adaptation law:

$$\triangle c_{m,n} = -\alpha \frac{\partial J_e}{\partial c_{m,n}}, \tag{4.13}$$

where $\alpha > 0$ represents a convergence rate and $\frac{\partial J_e}{\partial c_{m,n}}$ can be obtained from the chain rule and by using (4.12) and (4.11):

$$\frac{\partial J_e}{\partial c_{m,n}} = \frac{\partial J_e}{\partial e}\frac{\partial e}{\partial x_p}\frac{\partial x_p}{\partial u_f}\frac{\partial u_f}{\partial c_{m,n}} = -e d_{n,m}\frac{\partial x_p}{\partial u_f}. \tag{4.14}$$

To realize $\frac{\partial x_p}{\partial u_f}$ and incorporate also the controlled system dynamics, a self-adjusted adaptive RBF NN comprising one input layer, one hidden layer and one output layer is used as shown in Figure 4.3. The network input vector is given by

$$X = [u(k), x_p(k-1), x_p(k-2)]^T, \tag{4.15}$$

and its output is expressed by

$$x_{rbf} = \sum_{r=1}^{q} w_r h_r, \tag{4.16}$$

where w_r and h_r are respectively the weight and output of the r^{th} neuron. The radial basis function is of the Gaussian form:

$$h_r = exp(-\frac{\|X - c_r\|^2}{2\sigma_r^2}),\ r = 1, 2, 3, 4, ...q, \tag{4.17}$$

where $c_r = [c_{r1}, c_{r2}, c_{r3}]^T$ and σ_r are respectively the center and spread, and $\|.\|$ denotes the Euclidean norm. For adjustment of the network parameters, the following instantaneous cost function is defined:

$$J_n = \frac{1}{2}(x_p - x_{rbf})^2 \triangleq \frac{1}{2}e_{nn}^2, \tag{4.18}$$

from which, the network weights, center and spread can be respectively updated according to the gradient descent method and (4.17) as follows:

$$w_r(k+1) = w_r(k) + \eta e_{nn}(k) h_r(k), \tag{4.19}$$

$$c_{rs}(k+1) = c_{rs}(k) + \eta e_{nn}(k) w_r(k) h_r(k) \frac{X_{sr}(k) - c_{rs}(k)}{\sigma_r^2}, \tag{4.20}$$

$$\sigma_r(k+1) = \sigma_r(k) + \eta e_{nn}(k) w_r(k) h_r(k) \frac{\|X - c_r\|^2}{2\sigma_r^3}, \tag{4.21}$$

where $r = 1, 2, ..q$, $s = 1, 2, 3$ and $\eta > 0$ is a learning rate. Now from the chain rule and equation (4.8) we obtain:

$$\frac{\partial x_p}{\partial u_f} = \frac{\partial x_p}{\partial u}\frac{\partial u}{\partial u_f} \cong (k_p + k_i)\frac{\partial x_p}{\partial u}, \tag{4.22}$$

where the Jacobian $\frac{\partial x_p}{\partial u_f}$ can be derived from (4.17) and the relation $x_p = x_{rbf} + e_{nn}$ of the RBF NN shown in Figure 4.3 as:

$$\frac{\partial x_p}{\partial u} \approx \frac{\partial x_{rbf}}{\partial u} = \sum_{r=1}^{q} w_r h_r \frac{c_{r1} - u(k)}{\sigma_r^2}. \tag{4.23}$$

Thus, from (4.13), (4.14), (4.22) and (4.23), the parameters $c_{m,n}$ of the fuzzy controller can be adjusted as follows:

$$\triangle c_{m,n}(k) = \alpha e(k)(k_p + k_i) \sum_{r=1}^{q} w_r h_r \frac{c_{r1} - u(k)}{\sigma_r^2}, \tag{4.24}$$

with $m = j, j+1$ and $n = i, i+1$.

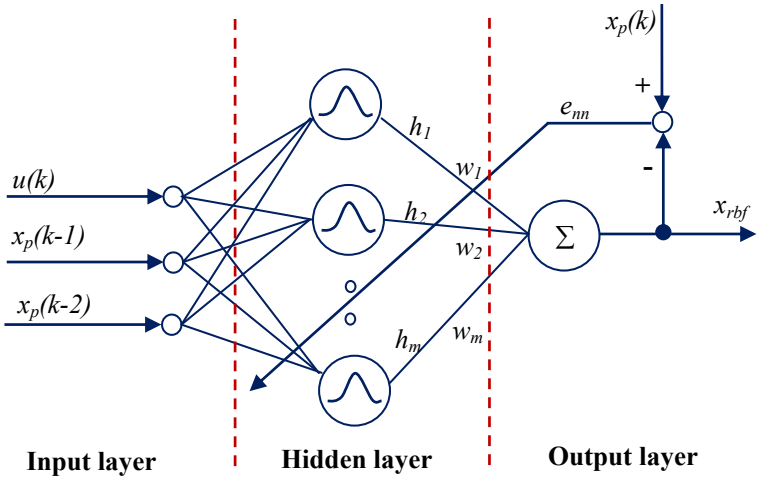


Figure 4.3 : Self-adjusted RBF NN schema

4.3 Hardware/software Co-design of Motion Control

The internal architecture of the proposed FPGA-based motion control IC for the PMLSM drive is shown in Figure 4.4. The FPGA uses Altera Stratix II EP2S60 which has 48,352 ALUTs, maximum 718 user I/O pins, total 2,544,192 RAM bits, and a Nios II embedded processor is downloaded into the FPGA to construct an SoPC environment. The motion control IC which comprises a Nios II embedded processor IP and a position control IP, is designed under the SoPC environment. The position control IP implemented by hardware is adopted to realize the function of a position NFC and speed PI controller, a current controller and coordinate transformation (CCCT), SVPWM generation, a quadrature encoder pulse (QEP) detection and transformation, ADC interface. The sampling frequency of current control is designed with 16 kHz. The operating clock rate of the designed FPGA controller is 50 MHz and the frequency divider generates 50 MHz (*Clk*), 25 MHz (*Clk-step*), 12 kHz (*Clk-cur*) and 2 kHz (*Clk-sp*) clock to supply all module circuits of the position

control IP.

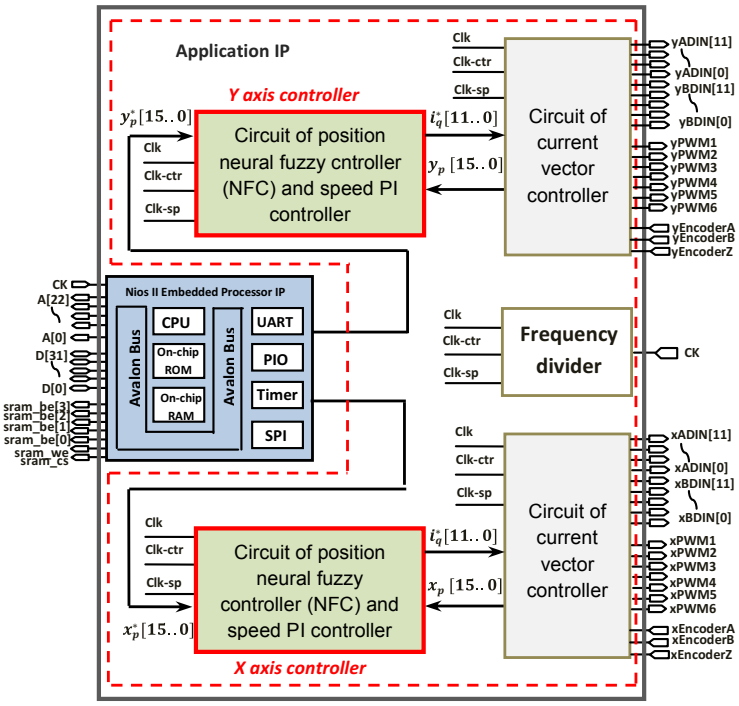


Figure 4.4 : FPGA architecture of the motion control system

A finite state machine (FSM) is employed to model the NFC in position loop and PI controller in speed loop which is shown in Figure 4.5, which uses adders, multipliers and registers, and manipulates 102 steps machine to carry out the overall computation. With the exception of the data type in the reference model are 24-bits, others data types are designed with 12-bits length, 2's complement and Q11 format. Although the algorithm of the NFC is highly complex, the FSM can give a very adequate modeling and can easily be described by VHDL. Furthermore, steps $S_0 - S_5$ execute the computation of the reference model output; steps $S_6 - S_8$ are for the computation of velocity, position error and error change; steps $S_9 - S_{13}$ execute the fuzzification and look-up the fuzzy table; $S_{14} - S_{22}$ are for the defuzzification;

$S_{23} - S_{27}$ are the computation of velocity and current command; $S_{28} - S_{91}$ describe the computation of RBF NN and Jacobian transformation; finally $S_{92} - S_{101}$ execute the tuning of fuzzy rule parameters. The operation of each step in Figure 4.5 can be completed within 40 ns (25 MHz clock) in FPGA; therefore the total of 102 steps need a 4.08 μs operation time.

The Nios II embedded processor IP is depicted to perform the function of the position command in the software, which includes the main program and the interrupt service routine (ISR) by 2 ms sampling interval. All programs are coded in the C programming language. Then, through the complier and linker operation in the Nios II IDE (Integrated Development Environment), the execution code is produced and can be downloaded to the external Flash or SDRAM via JTAG interface.

Finally, the FPGA utility of the motion control IC is evaluated. The circuit of a NFC uses 19,225 ALUTs resource and the overall circuits including a Nios II embedded processor IP (4,744 ALUTs and 45,824 RAM bits) as well as a position control IP (22,954 ALUTs and 301,056 RAM bits), use 57.3 % of the ALUTs resource and 13.6 % of the RAM resource of a Stratix II EP2S60.

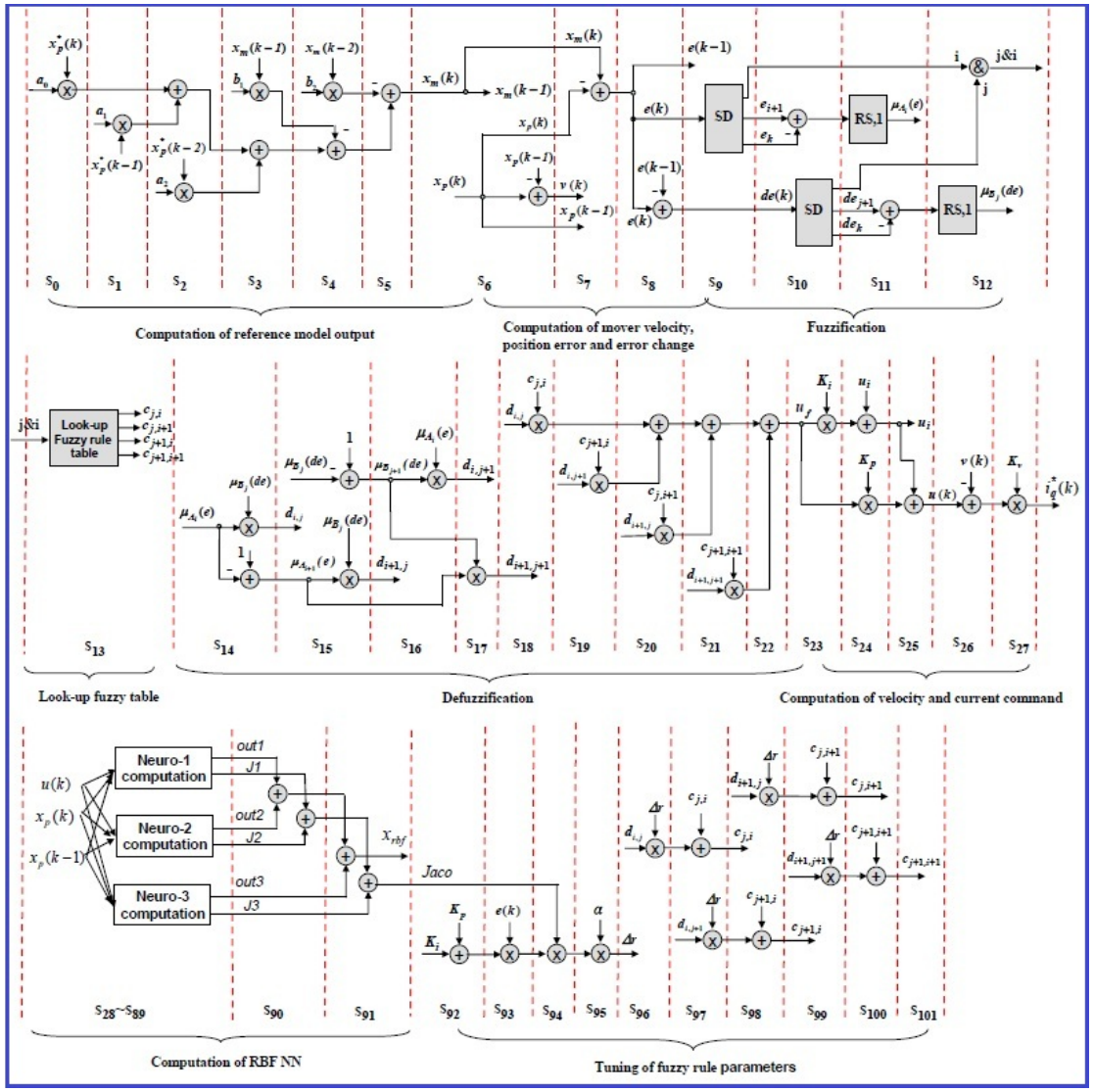


Figure 4.5 : State diagram of an FSM for describing the neural fuzzy controller

4.4 Experimental Results on One Axis of Motion System

4.4.1 Experiment Set-Up

The overall experimental system depicted in Figure 4.6 includes an FPGA (Stratix II EP2S60F672C5), a voltage source IGBT inverter and a PMLSM. The PMLSM is cog-free linear motor and a stroke length with 600 mm. The parameters of the motor are: $R_s = 27\ \Omega$, $L_d = L_q = 23.3\ mH$, $K_t = 79.9\ N/A$. The input voltage, continuous current, peak current (10% duty) and continuous power of the PMLSM are 220 V, 1.6 A, 4.8 A and 54 W, respectively. The maximum speed and acceleration are 4 m/s and 4 g but depend on the external load. The moving mass is 2.5 Kg, the maximum payload is 22.5 Kg and the maximum thrust force is 73 N under continuous operating conditions. A linear encoder with a resolution of 5 μm is mounted on the PMLSM as the position sensor, and the pole pitch is 30.5 mm (about 6100 $pulses$). The inverter has three sets of IGBT power transistors. The collector-emitter voltage of the IGBT is rated 600 V; the gate-emitter voltage is rated $\pm 20\ V$, and the DC collector current is rated 25 A and for a short time (1 ms) is 50 A. Input signals of the inverter are PWM signals from the FPGA device.

4.4.2 Experimental Results

The dynamic performance of the PMLSM drive is evaluated while the NFC is applied in the position control loop of Figure 4.1. The control sampling frequency of the current, speed and position loops are designed as 16 kHz, 2 kHz and 2 kHz, respectively. In the proposed motion control IC, the current controller, the speed controller and the NFC are all realized by hardware in FPGA. The NFC is used in the position loop, the membership function and the initial fuzzy rule table are designed, and the

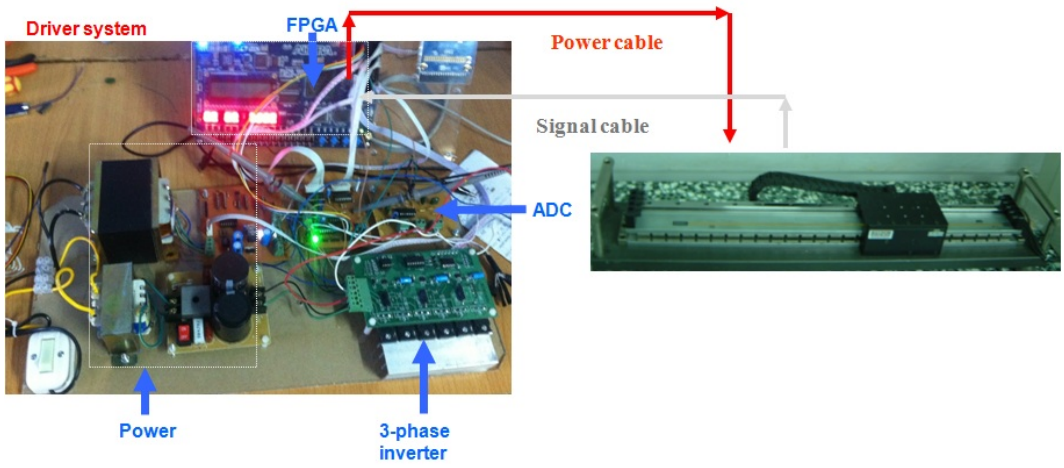


Figure 4.6 : Experiment set-up photograph

PI gains are chosen by $k_p = 0.3$, $k_i = 0.003$. The transfer function of the reference model is a second order system with the natural frequency of 20 rad/s and damping ratio of 1. Figure 4.7 shows the position step responses of the mover using the FC and NFC when the position command is a 0.5 Hz square wave with amplitude varied at $0 - 10$ mm and $20 - 30$ mm. The parameters $c_{j,i}$ of the fuzzy rule table are adequately selected at the 0 kg external load condition, which are presented in Figure 4.7, and the step response shows a good dynamic response with a rising time of 0.2 s, no overshoot and a near-zero steady state in Figure 4.9(a). However, when 11 kg external load is added upon the mover and the same fuzzy control rule table and controller parameters are used, the position dynamic response worsens and exhibits a 19.5 % overshoot in Figure 4.9(b). It reveals that the dynamic performance of the PMLSM is affected by the external load on the mover. Accordingly, a NFC is adopted in Figure 4.1 to solve this problem. When the proposed NFC is used with learning rate being 0.05, the tracking results are highly improved and are presented in Figure 4.9(c). Initially, the mover of the PMLSM tracks the output of the reference model

with overshoot. After one or two square wave commands, the $c_{j,i}$ parameters are tuned to adequate values as shown in Figure 4.8, and the mover can closely follow the output of the reference model. Therefore, the experimental results in Figure 4.9 demonstrate that the proposed FPGA-based NFC for the PMLSM drive is effective and robust.

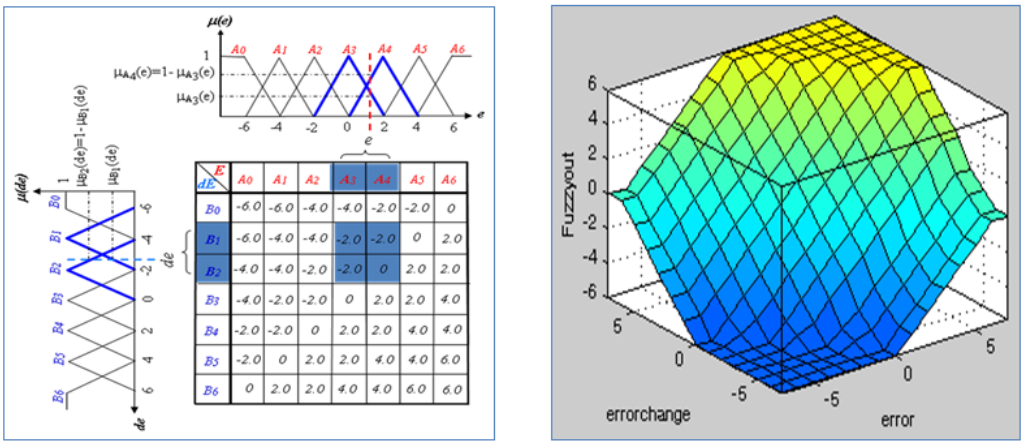


Figure 4.7 : Membership functions, fuzzy rule table and surface for step responses

Rule table after tuning						
-6.000	-6.000	-4.433	-6.332	-2.286	-2.000	0
-6.000	-4.000	-8.000	-8.000	-5.743	0	2.000
-4.000	-4.000	-8.000	-8.000	0	2.000	2.000
-4.000	-2.000	-2.447	0	1.604	2.000	4.000
-2.000	-2.000	0	7.444	2.358	4.010	4.010
-2.000	0	6.433	6.388	4.658	4.011	6.011
0	2.000	3.727	7.166	4.563	6.000	6.000

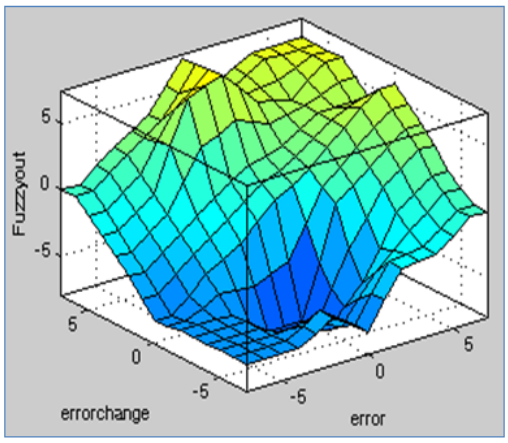
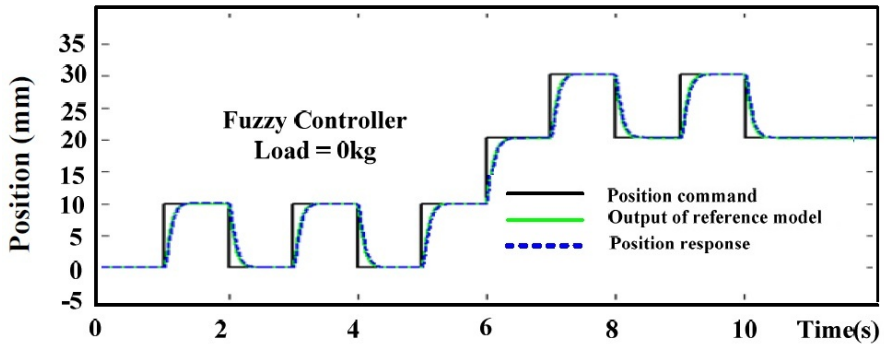
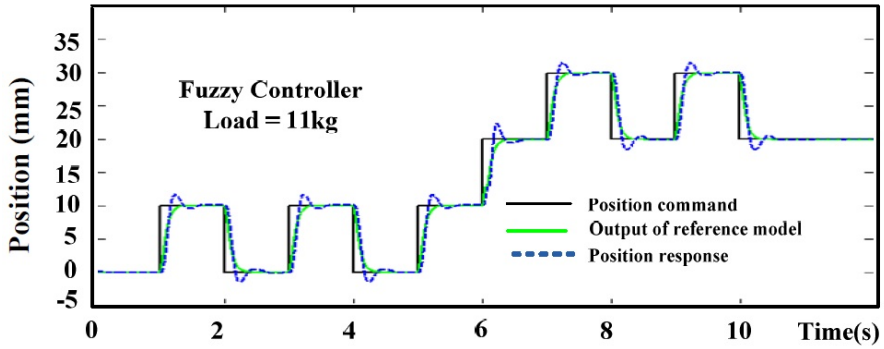


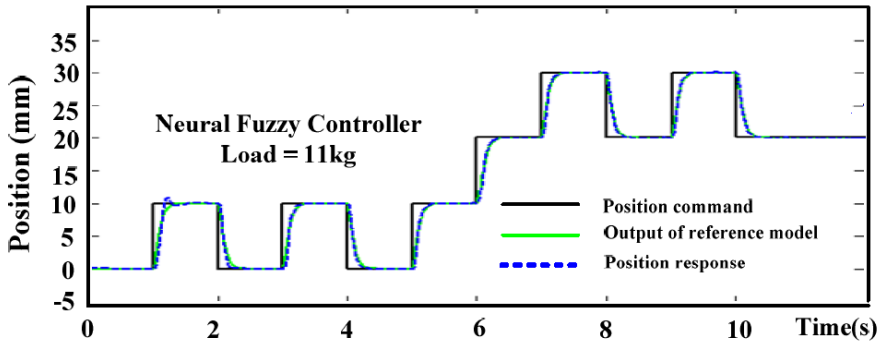
Figure 4.8 : Rule table after adjustment and the control effort surface



(a)



(b)



(c)

Figure 4.9 : Step response at $0-10$ mm to $20-30$ mm square save command under case of (a) FC without external load (b) FC with 11 Kg external load (c) NFC with 11 Kg external load

4.5 Computer Simulation of Multiple Axis Tracking Motion System

To develop FPGA based control architecture integration for a multiple axis tracking motion system, the simulation work is implemented in this section. The simulation results will verify the effectiveness of the proposed on-chip motion control system for a linear X-Y table under window motion trajectories.

4.5.1 Quartus II and Nios II based Simulation

Simulation is performed by using Quartus II and Nios II for a single FPGA chip. The motion trajectory is generated in Nios II integrated development environment and dynamics of two PMLSMs and control algorithms are described in VHDL code and executed in Quartus II software environment.

For simulation work, the discrete-time transfer function of the motion equation (4.5) at no-load is described as

$$\frac{x_p(z^{-1})}{i_q^*(z^{-1})} = \frac{\theta z^{-1}}{(1-\phi z^{-1})(1-z^{-1})}, \tag{4.25}$$

where $\phi = exp(B_m T/M_m)$, $\theta = K_t(1-\phi)/B_m$ with z^{-1} being the backward shift operator. The difference equation for the output displacement is therefore:

$$x_p(k) = \theta i_q^*(k1) + (1+\phi)x_p(k-1) - \phi x_p(k-2), \tag{4.26}$$

and its digital circuit design is implemented by the FSM method. Therein, the FSM uses one 24-bit multiplier, one 12-bit adder, and seven steps machine to carry out computation of the motion equation. With an encoder gain k_e, the displacement of the table to the pulse number is obtained as $x_e = k_e x_p$, where x_e is the pulse number generated from the linear encoder. The FC input is allocated with 16-bit

lengths in Q0 format; and all parameters and computations in the NFC and speed loop controller are designed with a 16-bit length in Q15 format, and the variables in the drive system are designed with a 12-bit length in Q11 format. All numerical operations adopt 2's complement operations. In simulation, the parameters ϕ, θ and k_e in Figure 4.10 are respectively set as 0.3, 0.12 and 100 $pulse/mm$. The travel displacement of x_p is designed within the range of ± 40 mm, and it will be mapped to the $\pm$(Q11) numerical value or binary value that is between -2048 and 2047. Moreover, k_e is 100 $pulses/mm$, the ± 40 mm travel displacement in x_p will generate $\pm$4000(Q0) pulse value in x_e. Thus, the logic circuit that transforms from x_p to x_e in Figure 4.10 is given by

$$x_e(15..0) = \ x_p(11) \ \& \ x_p(11) \ \& \ x_p(11) \ \& \ leftshift \ (x_p(11..0)) * 0.97, 1) \ \& \ '0'. \tag{4.27}$$

The FC rule base and control effort surface for $N_r = 7$ (49 rules) is shown in Figure 4.2. It is followed by a PI controller (4.8) with $k_p = 0.6$ and $k_i = 0.002$. From a desired second order system model, the speed PI controller gain is set at $K_{pS} = 0.9$ and $K_{iS} = 0.01$. The parameters $c_{j,i}$ of the fuzzy-rule table are tuned by using (4.24), whereby the self-adjusted RBFNN has a learning rate chosen as $\eta = 0.5$. The frequency divider generates 50 MHz (*Clk*), 12.5 MHz (*Clk-step*) and 5 MHz (*Clk-sp*). After completing hardware/software co-design, the hardware code is directly downloaded to the FPGA chip (Altera Cyclone II EP2C70), and the software code is downloaded to the external SDRAM. In the Quartus II and Nios II integrated development environment, the simulation work can be done, and the response results of some state variables can be collected. MATLAB can then be used for analysis and plotting.

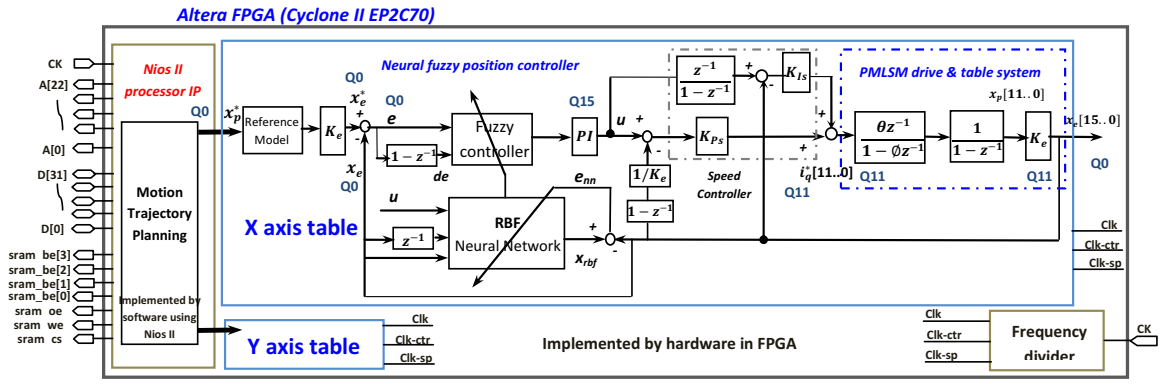


Figure 4.10 : Implementation diagram for on-chip simulation

4.5.2 Trajectory Planning

The trajectory planner, as shown in Figure 4.1, can be programmed to generate the model references and for both axes by Nios II software environment. The window shape trajectory is used in this simulation test, which is presented below.

Window Shape Trajectory

Typically, a window shape trajectory is shown in Figure 4.11, which can be divided into 9 parts from a- segment to i- segment of the trajectory. The window trajectory, from a start point on the negative X- axis, can be described as follows:

a- segment and i- segment:

$$x_i = x_{i-1}, \quad y_i = S + y_{i-1}, \tag{4.28}$$

b- segment:

$$x_i = O_{x1} + r\cos(\theta_i), \quad y_i = O_{y1} + r\sin(\theta_i), \quad \theta_i \in [\frac{3\pi}{2}, 2\pi], \tag{4.29}$$

c- segment:

$$x_i = S + x_{i-1}, \quad y_i = y_{i-1}, \tag{4.30}$$

d- segment:

$$x_i = O_{x2} + r\cos(\theta_i), \quad y_i = O_{y2} + r\sin(\theta_i), \quad \theta_i \in [\pi, \frac{3\pi}{2}], \tag{4.31}$$

e- segment:

$$x_i = x_{i-1}, \quad y_i = -S + y_{i-1}, \tag{4.32}$$

f- segment:

$$x_i = O_{x3} + r\cos(\theta_i), \quad y_i = O_{y3} + r\sin(\theta_i), \quad \theta_i \in [\frac{\pi}{2}, \pi], \tag{4.33}$$

g- segment:

$$x_i = -S + x_{i-1}, \quad y_i = y_{i-1}, \tag{4.34}$$

h- segment:

$$x_i = O_{x4} + r\cos(\theta_i), \quad y_i = O_{y4} + r\sin(\theta_i), \quad \theta_i \in [0, \frac{\pi}{2}], \tag{4.35}$$

where on the four circular corners, the step S and angular increment are constant, depending on the size of the trajectory.

Trajectory Performance Measures

To measure the performance of various controllers, the average tracking error $\bar{e}_{tr}$ and the standard deviation of the tracking error σ_{tr} for the motion tracking are defined as follows:

$$\bar{e}_{tr} = \sum_{k=1}^{N} \frac{e_{tr}(k)}{N}, \tag{4.36}$$

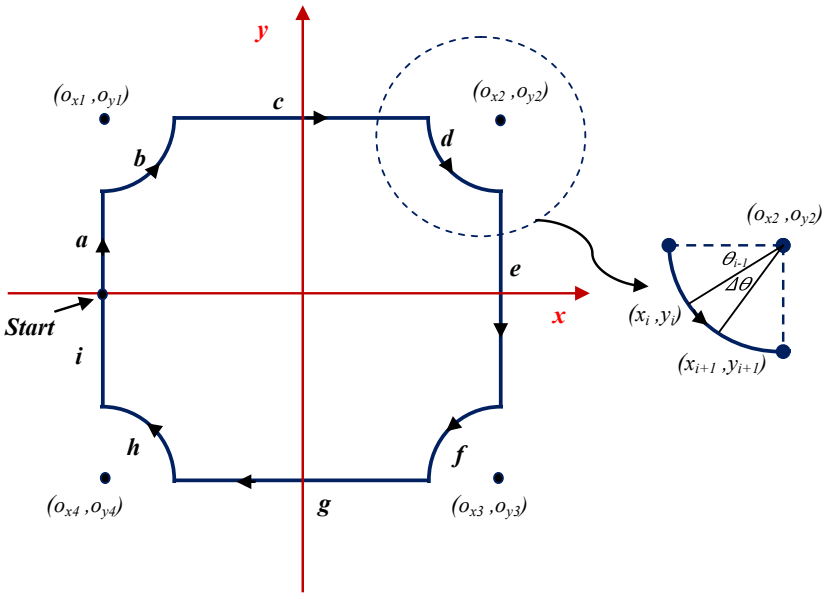


Figure 4.11 : Window motion trajectory

$$\sigma_{tr} = \sqrt{\sum_{k=1}^{N} \frac{(e_{tr}(k) - \bar{e}_{tr})^2}{N}}, \tag{4.37}$$

where N is the total number of sampling periods for a trajectory cycle and $e_{tr}(k) = \sqrt{e_x^2(k) + e_y^2(k)}$ is the tracking error distance in an X-Y coordinate frame.

4.5.3 Simulation Results and Discussion

Simulation is realized to verify the correctness and the effectiveness of the proposed controller for positioning and tracking of the X-Y table. The tracking performance is tested by using a window motion trajectory by using FC and NFC which are evaluated in two cases with parameter variations in (4.26): Case 1: $\theta = 0.3$, $\phi = 0.12$ and case 2: $\theta = 0.3 * 3$, $\phi = 0.12 * 3$.

Corresponding results are shown in Figures 4.12, 4.13 and 4.14. The X-Y table system parameters are initially designed at the normal condition (case 1) with FC

applied. The simulation results, as shown in Figure 4.12 present good tracking responses. When the system is subject to parameter changes (case 2), the results, shown in Figure 4.13, exhibit a deviation in the bends and corners of the window trajectory. To overcome this problem, the NFC is adopted and its simulation results are shown in Figure 4.14. Owing to the capability of adjusting the control parameters in FC the tracking error is significantly reduced. The comparison results between FC and NFC are shown in Table 4.1, where the resource usage is also tabulated. Performance improvement, in terms of average tracking error and standard deviation, is achieved with a slight increase of the resource usage. These together have demonstrated the effectiveness of the proposed NFC for position control of the X-Y table using the FPGA technology.

4.6 Chapter Conclusion

This chapter has presented an FPGA-based motion control system for positioning and window tracking of an X-Y table, driven by permanent magnet linear synchronous motors in vector control. The proposed architecture includes a motion planner, a PI speed control loop and a self-adjusted neural fuzzy PI position control loop.

The work herein is summarized as follows, (1) The functionalities required to build a fully digital motion controller of PMLSM drive for an X-Y table have been integrated in one FPGA chip, and (2) The behavior of a NFC has been successfully described by VHDL.

Tracking performance has been successfully demonstrated through experimental results. The control paradigm using this system-on-programmable-chip not only is energy efficient but also retains high control performance, and hence has great potential in multiple axis tracking motion systems in automation applications.

Table 4.1 : Comparison results between FC and NFC

	Average tracking error $\overline{e}_{tr}(mm)$	Tracking error standard deviation $\sigma_{tr}(mm)$	FPGA resource usage	
			ALUTs	(RAM bits)
FC(case 1)	0.62	0.031	29,701	45,742
FC(case 2)	1.37	0.072		
NFC(case 2)	0.35	0.016	37,704	48,947

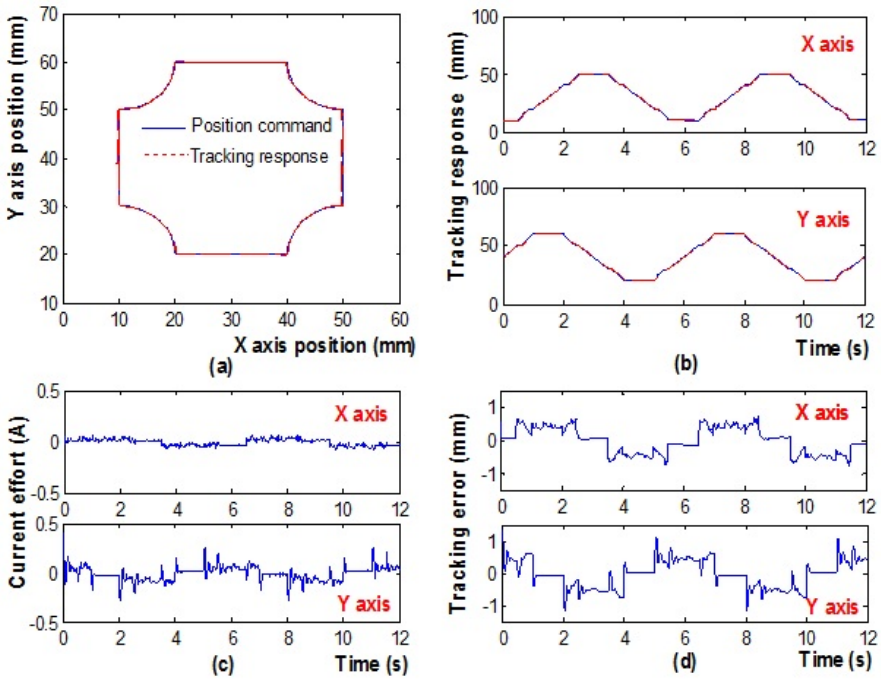


Figure 4.12 : Window trajectory response by using FC for case 1: (a) Window tracking (b) Position tracking (c) Control efforts (d) Tracking errors in X and Y-axis.

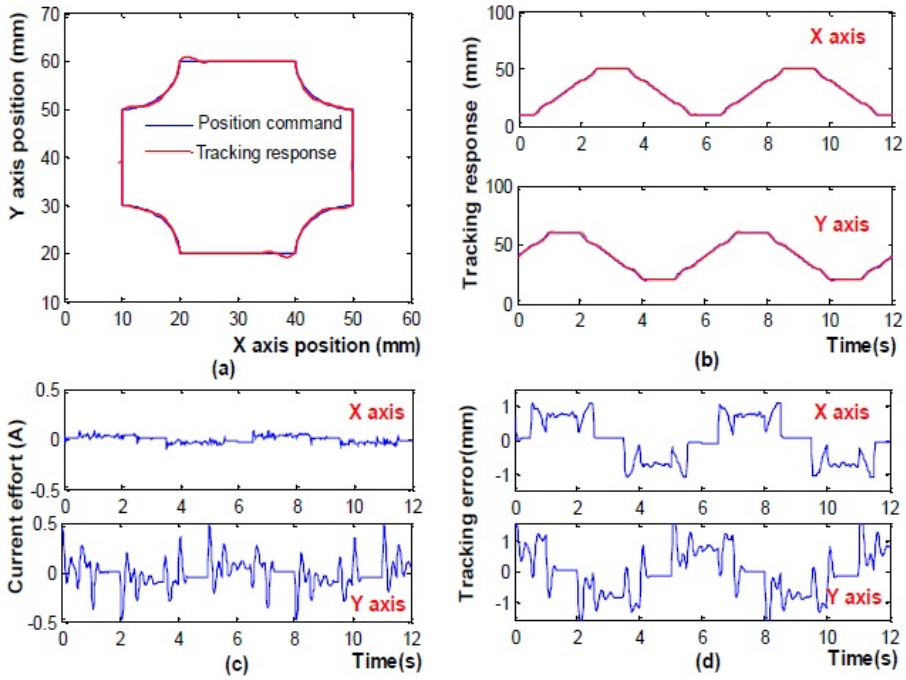


Figure 4.13 : Window trajectory response by using FC for case 2: (a) Window tracking (b) Position tracking (c) Control efforts (d) Tracking errors in X and Y-axis.

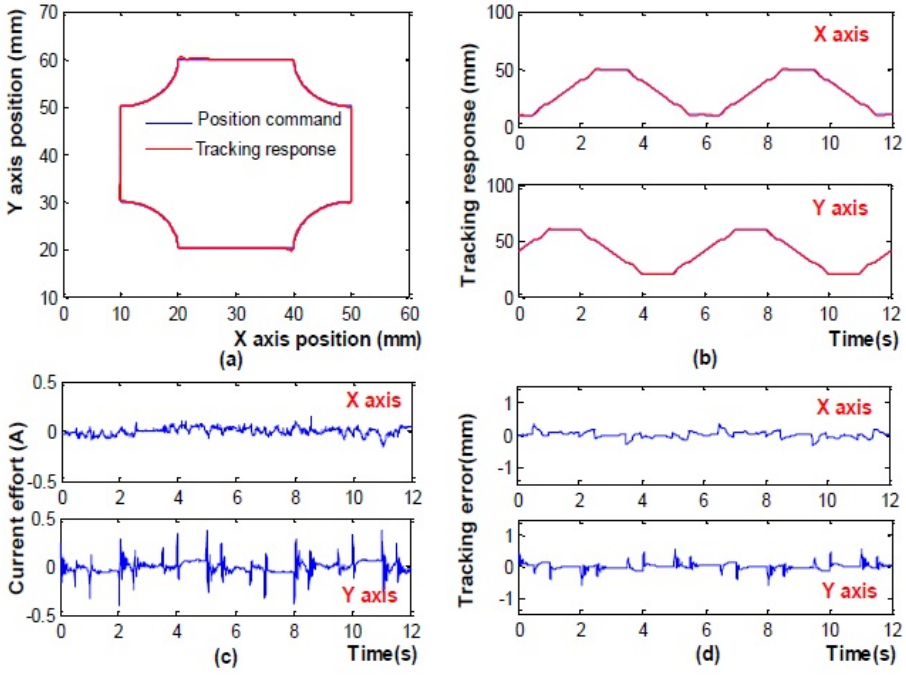


Figure 4.14 : Window trajectory response by using NFC for case 2: (a) Window tracking (b) Position tracking (c) Control efforts (d) Tracking errors in X and Y-axis.

Chapter 5

Observer-based Integral Sliding Mode Control for Sensorless PMSM Drives using FPGA

5.1 Introduction

As mentioned in chapter 3, in recent years, sensorless controllers for PMSM drive become more attractive and various sensorless control techniques have been investigated. Available sensorless methods are mainly based on the back EMF, SMO, EKF, neural network, etc. which require to be implemented by a fix-pointed processor [34],[38],[44],[57],[67],[70],[73]. In industrial applications, existing difficulties such as system parameter variations, external load disturbances, unmodeled uncertainties. always diminish the performance quality of the drive system. Although the proportional-integral (PI) controllers have been widely used in PMSM servo system due to their simple implementation and robustness, it is not easy to obtain a desired performance in the entire operating rage. To this end, intelligent control techniques, such as fuzzy control, neural network control, sliding mode control, have been developed and applied to the speed control of servo motor drives to achieve high operating performance. The sliding mode control (SMC) is a very useful nonlinear control method due to its good robustness for parameter variations, external disturbances, and fast response. To construct a common sliding surface, the sliding mode speed control requires both speed and acceleration signals. However, it is well known that transforming the sensed or the estimated speed into an acceleration sig-

nal is very sensitive to noisy effects and uncertainties of parameters. To cope with this problem, an integral sliding mode control (ISMC) with an integral sliding surface is proposed to regulate the PMSM speed [4],[38],[95]. With rapid developments of the system-on-chip technology, field programmable gate arrays with programmable hard-wired feature, fast computation ability, shorter design cycle, embedding processor, low power consumption and higher density become an alternative solution. FPGA-based controllers have been successfully implemented in many research areas including motion control and PMSM sensorless speed drive.

This chapter presents the design and evaluation of an observer-based integral sliding mode controller for sensorless PMSM drive based on the FPGA technology. For enhancement of robustness, a flux angle estimator using an improved sliding mode observer is proposed to estimate the current and back EMF as well as to derive the flux angle. These estimated values together with the computed rotor speed of the motor are fed back for control purposes in both the current loop and the speed loop. To increase the performance of PMSM speed control, an integral sliding mode control (ISMC) is designed with integral operation to improve steady state accuracy against parameter variations and external disturbances. The developed controller has been implemented in an FPGA-based environment and the VHDL is adopted to show advantages of the proposed control system. The validity of the proposed approach is verified through simulation results based on ModelSim and Simulink co-simulation method [44],[66],[70].

5.2 Observer-based Integral Sliding Mode Control Design

The architecture of the proposed speed control system for a sensorless PMSM drive is shown in Figure 5.1, while the modelling of the PMSM, flux angle and rotor speed

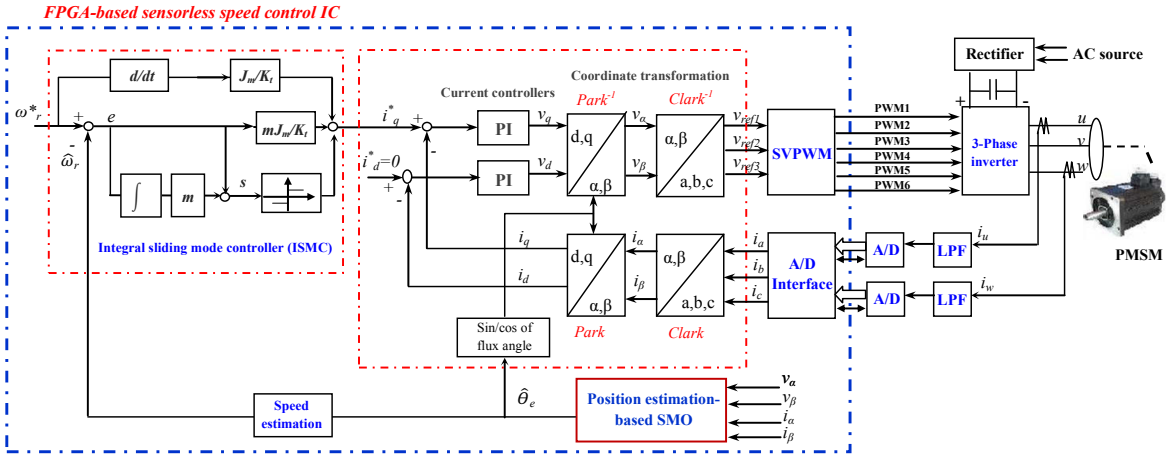


Figure 5.1 : The proposed speed control system for a sensorless PMSM drive using SMO

estimation using improved SMO, and the speed ISMC are described in this section.

5.2.1 Motor Drive Model

The current loop control of the PMSM drive in Figure 5.1 is based on a vector control approach to control i_d to zero and decouple the nonlinear model of the PMSM to a linear system. Therefore, after decoupling, the torque of the PMSM can be written as the following equation,

$$T_e = \frac{3p\lambda_m}{2} i_q \triangleq K_t i_q \ . \tag{5.1}$$

Considering the mechanical load, the overall dynamic equation of PMSM drive system is obtained by,

$$T_e - T_L = J_m \frac{d}{dt}\omega_r + B_m \omega_r \ , \tag{5.2}$$

where T_e is the motor torque, p is pole pairs, K_t is torque constant, J_m is the inertial value, B_m is damping ratio, T_L is the external torque, ω_r is rotor speed.

5.2.2 Improved SMO based Rotor Flux Position Estimation

Based on a conventional SMO, the rotor flux position can be estimated by using a current observer, a on-off controller, a low pass filter and a position calculation, as shown in Figure 5.2. The detailed observer design is described in the following.

As stated in equation (3.31), the EMF includes the position information from the flux. Therefore, it is possible to get position information θ_e from its phase by estimating the EMF. The EMF is observed by using equation (3.32). From this equation, a sliding mode observer is designed by,

$$\frac{d}{dt}\begin{bmatrix}\hat{i}_\alpha \\ \hat{i}_\beta\end{bmatrix} = -\frac{r_s}{L_s}.\begin{bmatrix}\hat{i}_\alpha \\ \hat{i}_\beta\end{bmatrix} + \frac{1}{L_s}\begin{bmatrix}v_\alpha \\ v_\beta\end{bmatrix} - \frac{1}{L_s}Z\ , \tag{5.3}$$

where $\begin{bmatrix}\hat{i}_\alpha & \hat{i}_\beta\end{bmatrix}^T$ is the estimated current on fixed coordinates and $Z \triangleq \begin{bmatrix}z_\alpha & z_\beta\end{bmatrix}^T$ is the output gain of the switching controller. The dynamic estimation error $e_i \triangleq \begin{bmatrix}\tilde{i}_\alpha & \tilde{i}_\beta\end{bmatrix}^T = \begin{bmatrix}\hat{i}_\alpha - i_\alpha & \hat{i}_\beta - i_\beta\end{bmatrix}^T$ is obtained by subtracting equation (5.3) from equation (3.32); we have:

$$\frac{d}{dt}\begin{bmatrix}\tilde{i}_\alpha \\ \tilde{i}_\beta\end{bmatrix} = -\frac{r_s}{L_s}.\begin{bmatrix}\tilde{i}_\alpha \\ \tilde{i}_\beta\end{bmatrix} + \frac{1}{L_s}\begin{bmatrix}e_\alpha \\ e_\beta\end{bmatrix} - \frac{1}{L_s}\begin{bmatrix}z_\alpha \\ z_\beta\end{bmatrix}. \tag{5.4}$$

In conventional SMO, Z is defined as,

$$Z = \begin{bmatrix}z_\alpha \\ z_\beta\end{bmatrix} \triangleq k.sign\left(\begin{bmatrix}\hat{i}_\alpha - i_\alpha \\ \hat{i}_\beta - i_\beta\end{bmatrix}\right), \tag{5.5}$$

To solve the chattering problem, the signum function is replaced by a saturation function in this study and is defined as:

$$sat(\tfrac{e_i}{\xi}) = \begin{cases}\frac{e_i}{\xi} & |\frac{e_i}{\xi}| \le 1 \\ sign(\frac{e_i}{\xi}) & |\frac{e_i}{\xi}| > 1,\end{cases} \tag{5.6}$$

where the constant factor ξ defines the thickness of the boundary layer and the current error e_i. Further, by properly choosing the gain k to be large enough, i.e., $k \geq max(|e_\alpha|, |e_\beta|)$, the inequality $e_i^T.\dot{e}_i < 0$ can be reached and the SMO can induce a sliding mode, i.e, $e_i = \dot{e}_i = 0$. In this induced sliding mode, according to (5.4), Z will approach to the EMF,

$$\begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix} = \begin{bmatrix} e_\alpha \\ e_\beta \end{bmatrix} = \omega_e \lambda_m \begin{bmatrix} -\sin\theta_e \\ \cos\theta_e \end{bmatrix}. \tag{5.7}$$

In order to alleviate the high frequency in switching control, a low-pass filter is applied:

$$\frac{d}{dt}\begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} = -\omega_o. \begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix} + \omega_o \begin{bmatrix} z_\alpha \\ z_\beta \end{bmatrix}, \tag{5.8}$$

where $\omega_o = 2\pi f_o$ and f_o is the cut-off frequency of the filter. Finally, the rotor position $\hat{\theta}_e$ can be computed by,

$$\hat{\theta}_e = tan^{-1}\left(-\frac{\hat{e}_\alpha}{\hat{e}_\beta}\right). \tag{5.9}$$

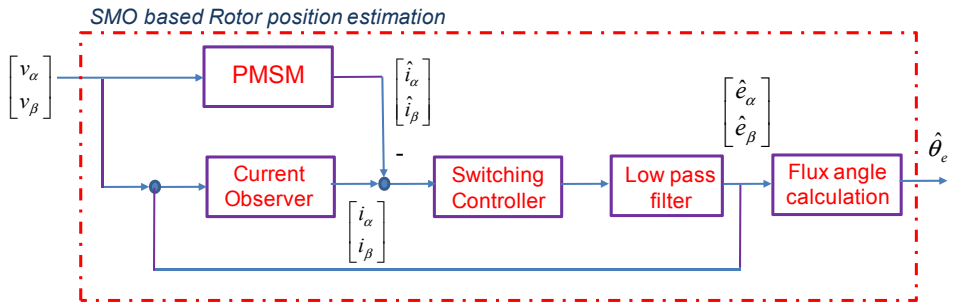


Figure 5.2 : Rotor flux angle estimation based on the conventional SMO

5.2.3 Integral Sliding Mode Control Design in Speed Loop

From equation (5.2), the torque equation of the PMSM system can be derived as:

$$\frac{d}{dt}\omega_r = \frac{3p\lambda_m}{2J_m}i_q - \frac{B_m}{J_m}\omega_r - \frac{T_L}{J_m} = \frac{3p\lambda_m}{2J_m}i_q^* + \mu(t), \tag{5.10}$$

where $\mu(t) = \frac{3p\lambda_m}{2J_m}(i_q - i_q^*) - \frac{B_m}{J_m}\omega_r - \frac{T_L}{J_m}$ is the lumped disturbances. Define the speed error as $e = \omega_r^* - \omega_r$ where ω_r^* is reference speed. Taking the derivative of e and substituting (5.10) into it, we obtain:

$$\dot{e} = \dot{\omega}_r^* - \frac{3p\lambda_m}{2J_m}i_q^* - \mu(t). \tag{5.11}$$

The integral sliding surface is defined as,

$$s = e + m\int_0^t e d\tau, \tag{5.12}$$

and the variable structure speed controller is designed as follows,

$$i_q^* = \frac{2J_m}{3p\lambda_m}m.e + \frac{2J_m}{3p\lambda_m}.\dot{\omega_r}^* + k_s.sign(s), \tag{5.13}$$

where m, $k_s > 0$ and $\left(\frac{2J_m}{3p\lambda_m}m.e + \frac{2J_m}{3p\lambda_m}.\dot{\omega}_r^*\right)$ is the equivalent control.
Assume that the lumped disturbances of the system $\mu(t)$ satisfies $0 \leq |\mu(t)| < d$. Choosing Lyapunov function $V = \frac{1}{2}s^2$, and taking the derivative of it along (5.12), yields:

$$\dot{V} = s.\dot{s} = s[\dot{e} + me]. \tag{5.14}$$

Substituting (5.11) into (5.14), the stability condition can be derived as,

$$\begin{aligned}\dot{V} &= s\left[\dot{\omega}_r^* - \frac{3p\lambda_m}{2J_m}i_q^* - \mu(t) + me\right] < 0 \\ &= -\frac{3p\lambda_m}{2J_m}s\left[i_q^* - \frac{2J_m}{3p\lambda_m}me - \frac{2J_m}{3p\lambda_m}\dot{\omega}_r^* + \frac{2J_m}{3p\lambda_m}\mu(t)\right] < 0.\end{aligned} \tag{5.15}$$

By substituting (5.13) into (5.15), we have:

$$\begin{aligned}\dot{V} &= -\frac{3p\lambda_m}{2J_m}s\left[k_s.sign(s) + \frac{2J_m}{3p\lambda_m}\mu(t)\right] < 0 \\ &= -\frac{3p\lambda_m}{2J_m}\left[k_s.sign(s).s + \frac{2J_m}{3p\lambda_m}\mu(t).s\right] < 0 \\ &= -\frac{3p\lambda_m}{2J_m}\left[k_s|s| + \frac{2J_m}{3p\lambda_m}d.s\right] < 0, \end{aligned} \tag{5.16}$$

where the switching control gain can be derived to satisfy the inequality condition as follows,

$$k_s > \frac{2J_m}{3p\lambda_m}d. \tag{5.17}$$

5.3 FPGA based Sensorless Control Implementation

5.3.1 Control Architecture

Figure 5.3 shows the FPGA-based architecture of the proposed sensorless control system for the PMSM drive. The controller includes an ISMC speed controller, a PI current controller and coordinate transformation (CCCT), a SVPWM generation and SMO-based rotor position estimation. All modules presented in Figure 5.3 are described by Verilog HDL and simulated in ModelSim as well as tested in the Altera Cyclone II EP2C70 board. On the basis of the fact that the velocity of the outer control loop is much slower than the estimated inner loop, the sampling frequency is designed with 16 kHz in the current loop, and 2 kHz in the speed loop. The frequency divider generates 50 MHz (*Clk*), 12.5 MHz (*Clk-step*) and 16 kHz (*Clk-ctr*) clock to supply all circuits.

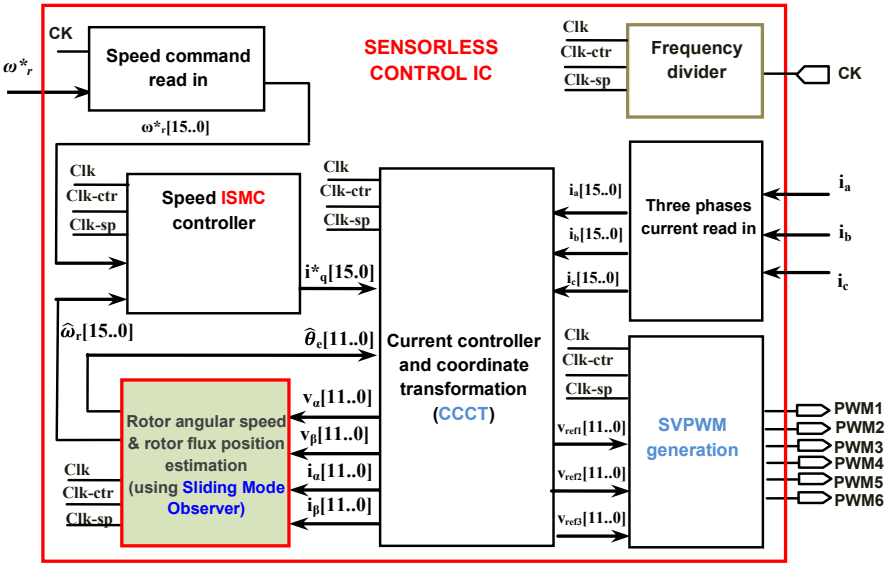


Figure 5.3 : The proposed speed ISMC controller for a sensorless PMSM drive

5.3.2 Design procedure for SMO

A design procedure for the estimation of the rotor position by using SMO is summarized as follows:

Step 1: Measure the values of $i_\alpha(n)$, $i_\beta(n)$, $v_\alpha(n)$, $v_\beta(n)$ from the PMSM drive.

Step 2: Estimate the estimated current by SMO from equation (5.3)

$$\begin{bmatrix} \hat{i}_\alpha(n+1) \\ \hat{i}_\beta(n+1) \end{bmatrix} = \phi. \begin{bmatrix} \hat{i}_\alpha(n) \\ \hat{i}_\beta(n) \end{bmatrix} + \psi \begin{bmatrix} v_\alpha(n) \\ v_\beta(n) \end{bmatrix} - \psi \begin{bmatrix} \hat{e}_\alpha \\ \hat{e}_\beta \end{bmatrix}, \tag{5.18}$$

where $\phi = e^{-\frac{r_s}{L_s}T_s} I$, $\psi = \frac{1}{r_s}(1 - e^{-\frac{R_s}{L_s}T_s})$ and T_s is the sampling time.

Step 3: Calculate the current error by

$$\begin{bmatrix} \tilde{i}_\alpha(n) \\ \tilde{i}_\beta(n) \end{bmatrix} = \begin{bmatrix} \hat{i}_\alpha(n) - i_\alpha(n) \\ \hat{i}_\beta(n) - i_\beta(n) \end{bmatrix}, \tag{5.19}$$

Step 4: Obtain the Z gain of the switching controller from equations (5.5) and (5.6)

$$Z(n) = \begin{bmatrix} z_\alpha(n) \\ z_\beta(n) \end{bmatrix} \triangleq k.sat \left(\begin{bmatrix} \hat{i}_\alpha(n) - i_\alpha(n) \\ \hat{i}_\beta(n) - i_\beta(n) \end{bmatrix} \right), \tag{5.20}$$

where $sat(.)$ is calculated by using look up table (LUT).

Step 5: Estimate the EMF by equation (5.8)

$$\begin{bmatrix} \hat{e}_\alpha(n+1) \\ \hat{e}_\beta(n+1) \end{bmatrix} = \begin{bmatrix} \hat{e}_\alpha(n) \\ \hat{e}_\beta(n) \end{bmatrix} + 2\pi f_o \begin{bmatrix} z_\alpha(n) - \hat{e}_\alpha(n) \\ z_\beta(n) - \hat{e}_\beta(n) \end{bmatrix}, \tag{5.21}$$

Step 6: Obtain the estimated rotor position in equation (5.9)

$$\hat{\theta}_e(n) = tan^{-1} \left(-\frac{\hat{e}_\alpha(n)}{\hat{e}_\beta(n)} \right), \tag{5.22}$$

then set $n = n + 1$ and back to *Step 1*.

5.3.3 Algorithm Implementation

A finite state machine is employed to describe the control algorithm of SMO-based rotor position estimation, as shown in Figures 5.4 and 5.5, respectively. Although the algorithms of ISMC and SMO described are complex, the FSM adequately incorporates the control structure and can be easily described by VHDL. The multiplier and adder apply the Altera LPM (Library Parameterized Modules) standard.

In Figure 5.4, the data type adopts a 16-bit length with Q15 format and 2's complement operations. It manipulates 13 steps machine to carry out the overall computations of the ISMC algorithm. The steps $S_0 - S_6$ are for the computation of the speed error and sliding function; step S_7 executes the signum function; and finally the sliding mode control is realized in steps $S_8 - S_{13}$. The operation of each step can be completed within 80 ns (12.5 MHz); therefore total 13 steps need 1.04 μs for the ISMC operation.

The SMO-based rotor position estimation algorithm is shown in Figure 5.5. The data type adopts a 12-bit length with Q11 format. The steps $S_0 - S_8$ execute the estimation of currents; steps $S_9 - S_{10}$ compute the current errors; S_{11} calculate the saturation function for the switching control; steps $S_{12} - S_{16}$ describe the computation of EMF and $S_{17} - S_{36}$ perform the computation of the rotor position. The implementation of SMO needs 2.88 μs for 36 steps. Other circuit designs such as CCCT and SVPWM, shown in Figure 5.3, were presented in the section on digital circuit design of current vector control, in Chapter 3.

Finally, the resource usage of ISMC, CCCT and SMO are 4,268 Logic Elements (LEs) and 73,728 RAM bits resource for the Cyclone II EP2C70.

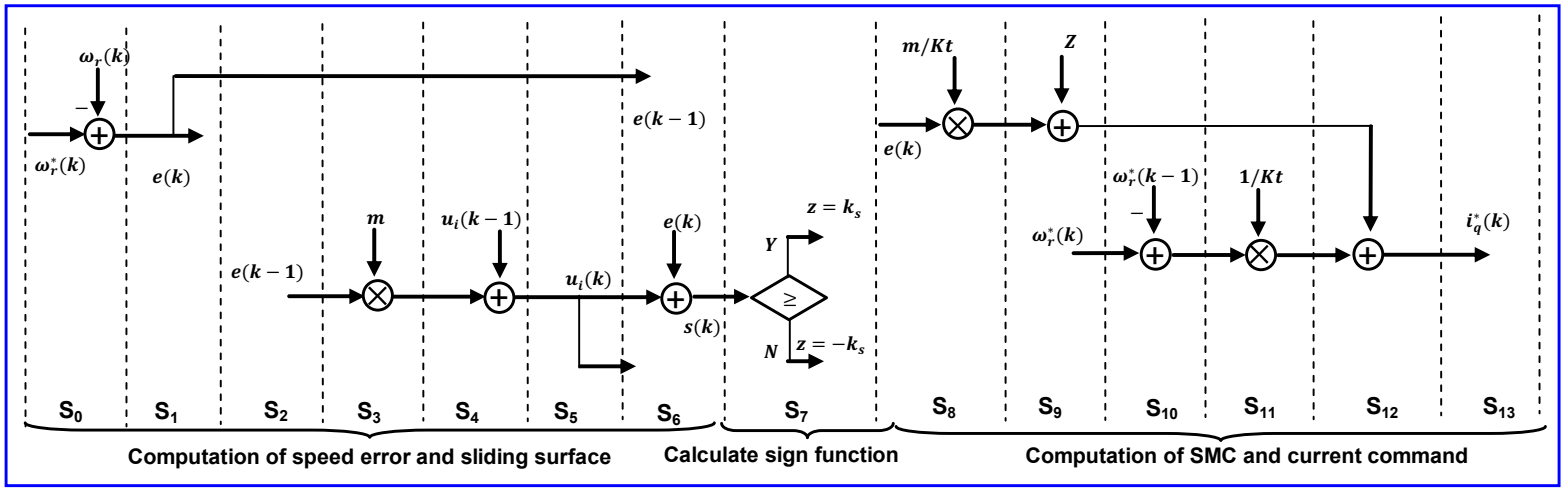


Figure 5.4 : State diagram of an FSM for the speed controller using ISMC

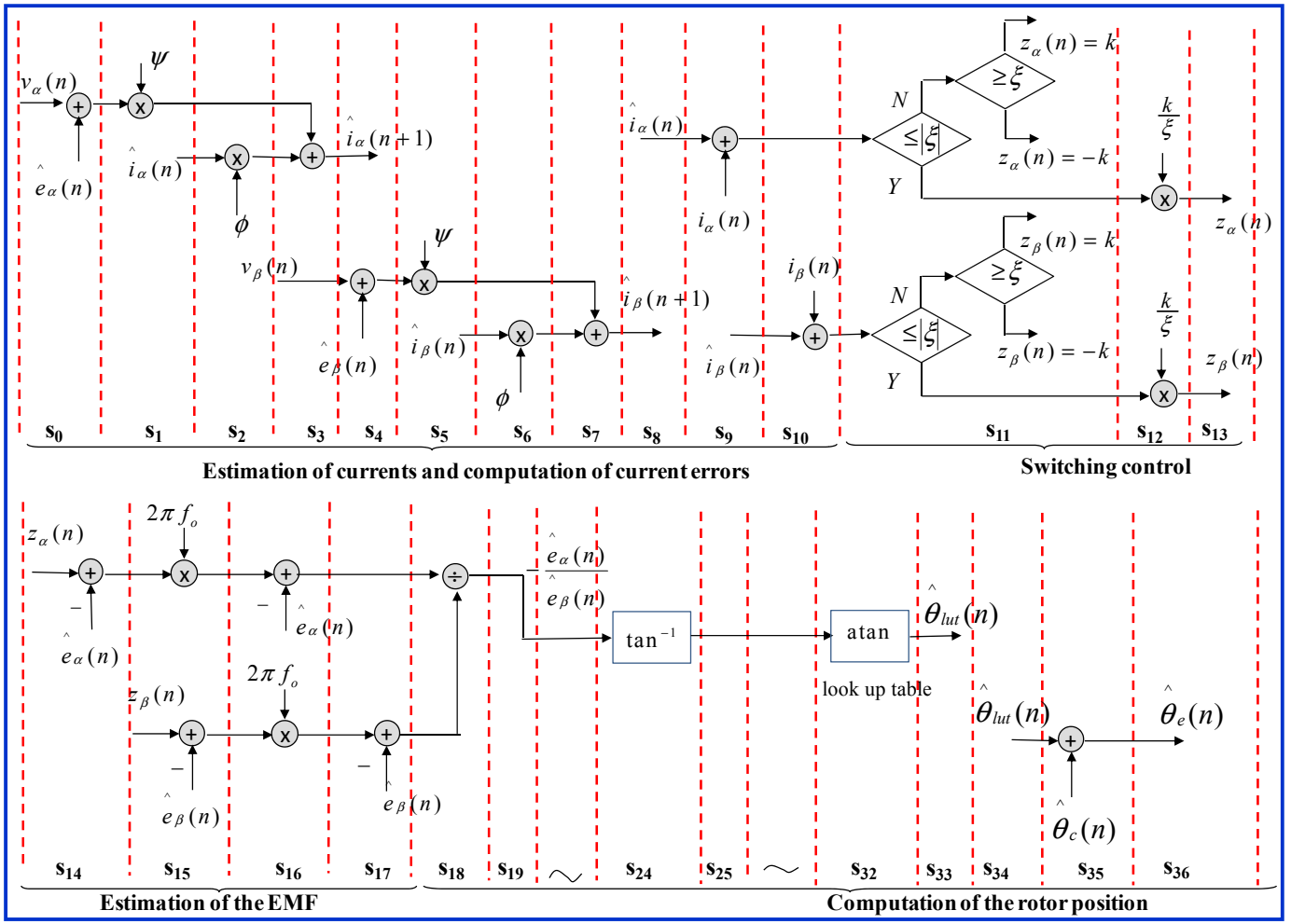


Figure 5.5 : State diagram of an FSM for an improved SMO-based rotor position estimation algorithm

5.4 Results and Discussion

Simulation is performed by using ModelSim/Simulink co-simualtion method. The ModelSim performs the function of ISMC controller, SMO and current vector controller which is described using VHDL code. In the Simulink, the SimPowerSystem blockset can provide the components of PMSM and the inverter and it also can generate stimuli to ModelSim and analyze the simulations responses. The designed PMSM parameters applied in simulation of Figure 5.3 are that pole pairs is 4, stator phase resistance is 1.3 Ω, stator inductance is 6.3 mH, inertia is $J = 0.00011\ kg.m^2$ and friction factor is $F = 0.0014\ N.m.s$.

In the simulation of sensorless PMSM drive, rotor position estimation based on SMO is firstly evaluated. The conventional SMO method with signum function and the improved one with saturation function are tested with PMSM running speed at 500 rpm and these simulation results are presented in Figures 5.6, 5.7. The results show that the response of the estimated rotor flux position can follow with the actual rotor flux position in both methods. On comparing the results of the two methods, it is clearly observed that the chattering of rotor position estimation is reduced and its accuracy is improved significantly in SMO where saturation function is used. Secondly, the performance of sliding mode control using integral operation is verified. Three tested cases are evaluated with parameter variations: Case 1 (Normal-load condition): $J = 0.00011$, $F = 0.0014$; case 2 (Light-load condition): $J = 0.00011/3$, $F = 0.0014/3$; and case 3 (Heavy-load condition): $J = 0.00011 * 3$, $F = 0.0014 * 3$. The 20 Hz square wave with magnitude of 500 rpm is used as the tested command. When the speed controller is adopted by the PI controller only (K_p=0.36, K_i=0.005) and the PMSM parameters are initially designed at the normal load condition (case 1), the simulation result is shown in Figure 5.8 which presents a good following re-

sponse. However, when the running condition is changed to the light-load condition (case 2) and heavy-load condition (case 3), the results in Figures 5.9 and 5.10 show that the step speed response become worse with overshoot in the light-load condition and slow response occurred in the heavy-load condition. It demonstrates that although the sensorless control based on SMO in PMSM drive can give a good speed tracking, it is still easily affected by external load variation. To cope with this problem, an integral SMC is adopted in Figure 5.1. The ISMC has an integral sliding surface to reject the requirement of the acceleration signal, which is usual in traditional sliding-mode speed control techniques. Due to the nature of the sliding control, this control scheme is robust under uncertainties caused by parameter variations or by changes in the external load. Figures 5.11 to 5.13 show the simulation results while it uses the proposed ISMC control in the sensorless PMSM drive. The results show an improvement with the rotor speed responses exhibiting no overshoot and less rising time in both cases 1 and 2. They also indicate that the proposed ISMC can enhance robustness in the sensorless PMSM drive.

5.5 Chapter Conclusion

In this chapter, a FPGA-based sensorless sliding-mode vector control has been presented. The proposed system comprises a sliding mode observer, a field-oriented PI current controller, and an ISMC for the speed control loop. All the control system components can be integrated and realized in one FPGA chip. The SMO is used to estimate the rotor position and speed of the PMSM due to its strong robustness, and the signum function is replaced by the saturation function to reduce system chattering, and the traditional SMO is improved. In addition, a sliding-mode controller with an integral switching surface is proposed. This control scheme is robust under uncer-

tainties. Simulation results validate the feasibility and effectiveness of the proposed control system.

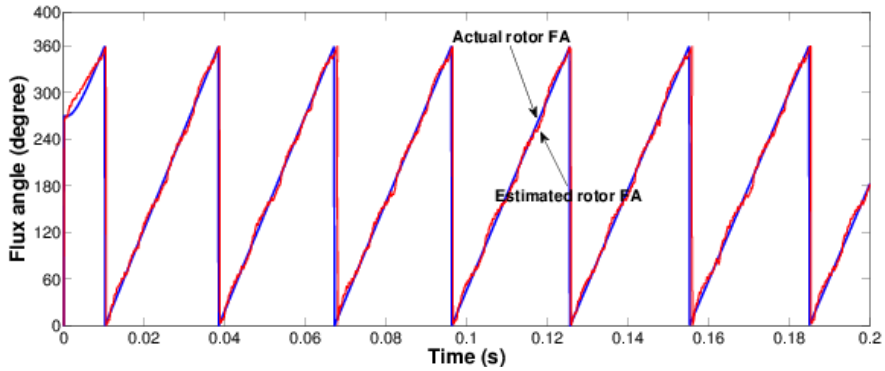


Figure 5.6 : Flux angle (FA) waveforms obtained by the conventional SMO method using the signum function

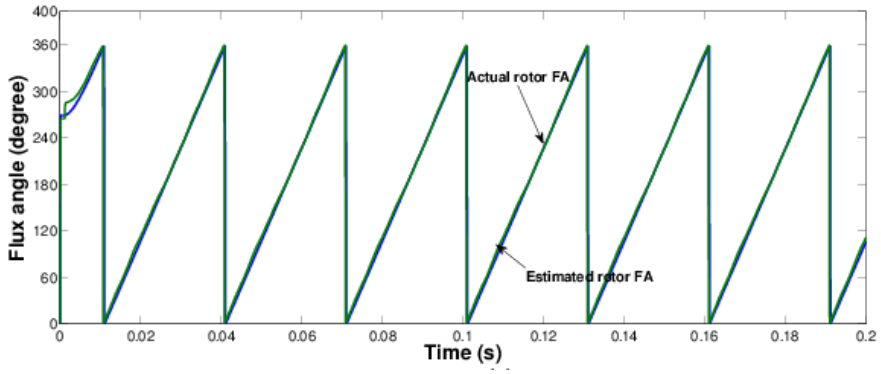


Figure 5.7 : Flux angle (FA) waveforms obtained by the improved SMO method using the saturation function

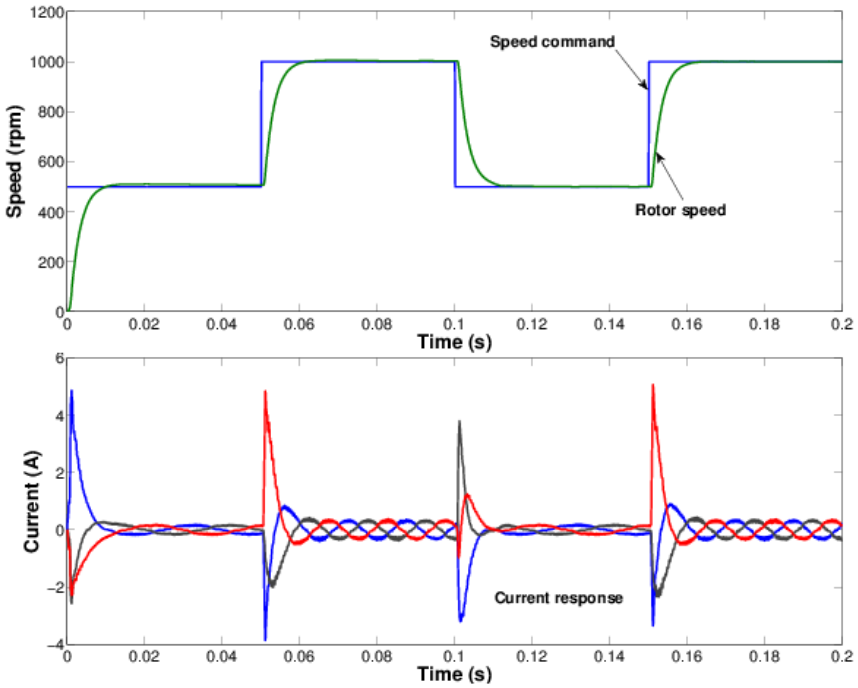


Figure 5.8 : Simulation result when PI controller is used while sensorless PMSM operated at normal load condition

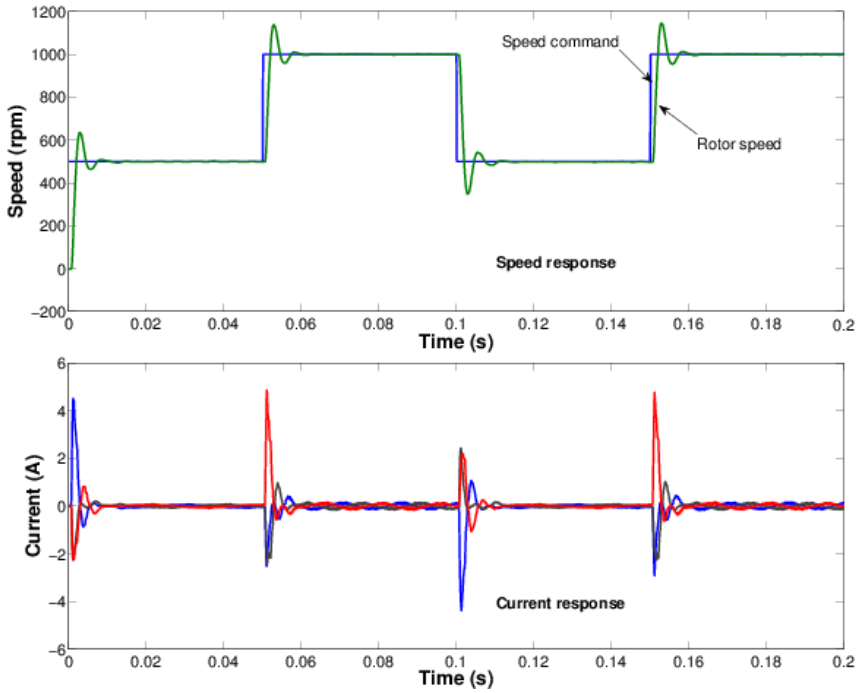


Figure 5.9 : Simulation result when PI controller is used while sensorless PMSM operated at light load condition

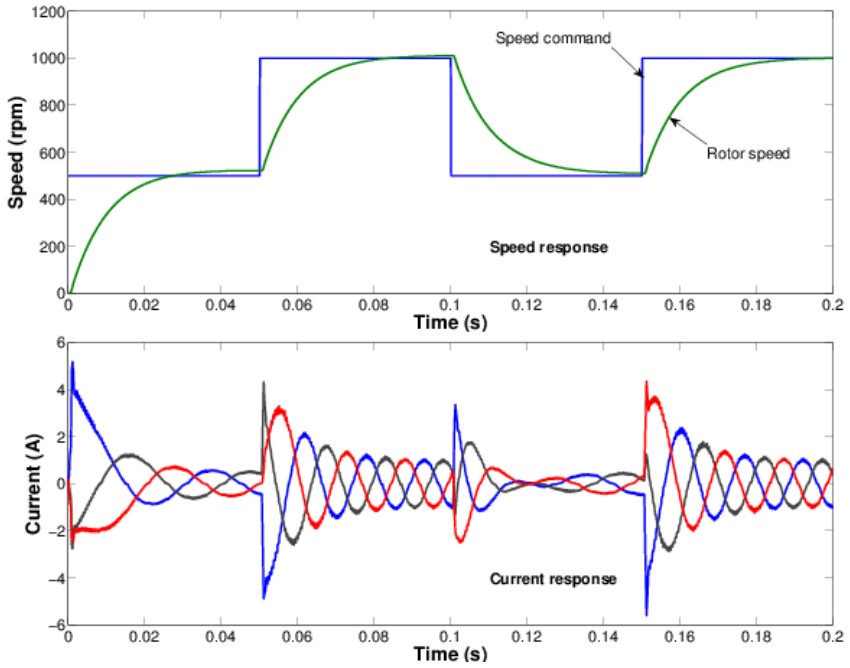


Figure 5.10 : Simulation result when PI controller is used while sensorless PMSM operated at heavy load condition

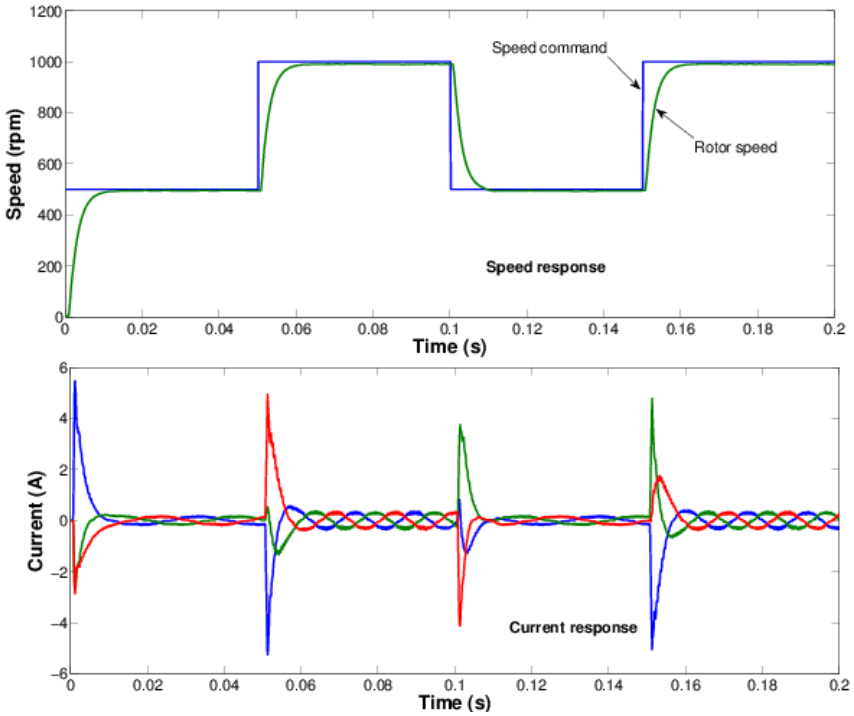


Figure 5.11 : Simulation result when ISMC controller is used while sensorless PMSM operated at normal load condition

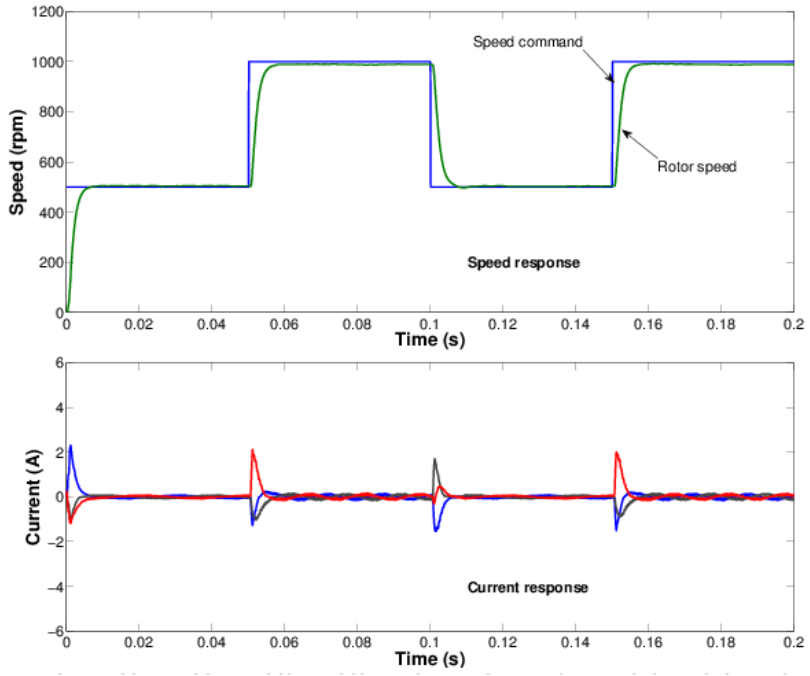


Figure 5.12 : Simulation result when ISMC controller is used while sensorless PMSM operated at light load condition

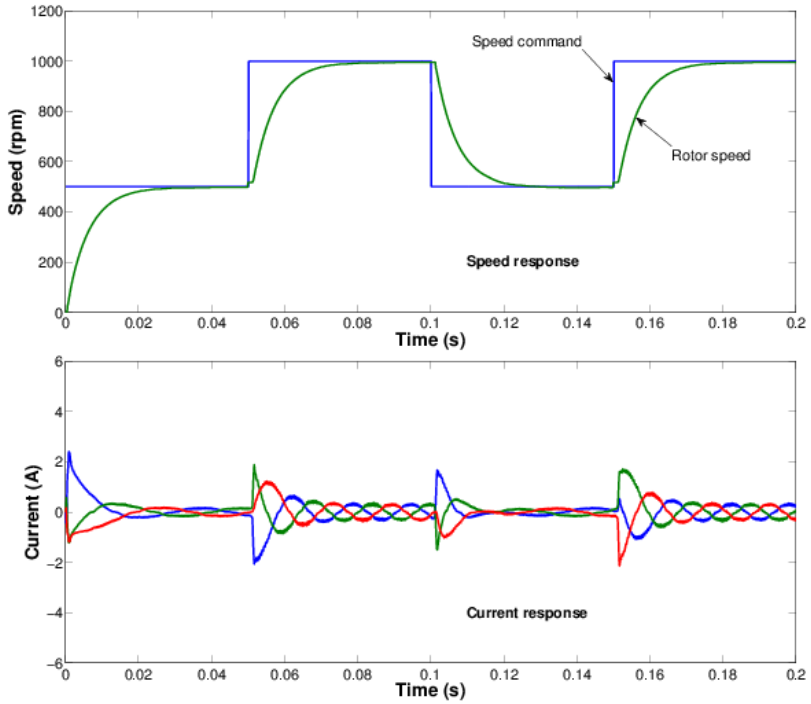


Figure 5.13 : Simulation result when ISMC controller is used while sensorless PMSM operated at heavy load condition

Chapter 6

FPGA Sensorless PMSM Drive with Adaptive Fading Extended Kalman Filter

6.1 Introduction

A variety of filtering and observation techniques have provided promising results for the PMSM rotor position and speed estimation, see, e.g. [6],[12],[13],[20],[88], [103]. It was found, nonetheless, that the estimator may not be able to trace abrupt changes [20], [68]. Further, in most practical systems, the models are simplified from complex dynamics in real-world situations and the statistic characteristics of noise and initial states may not be known accurately. These inaccuracies may seriously degrade performance and generate a divergence of the filter. These may result in a large overall discrepancy in the state estimation. To overcome this problem, fading filtering and its adaptation have been proposed [12],[23],[29],[62],[94]. The adaptive fading extended Kalman filter (AF-EKF) introduced in [63] was based on the weighting of the error covariance equation with a scalar forgetting factor. In addition, a different adaptive fading extended Kalman filter was proposed to improve the filter performance to deal with the case of incomplete information of modelling or measurement [39]. In reference [98], another AF-EKF was suggested, using an adaptive tracking technique with a diagonal matrix forgetting factor to identify time-varying parameters of linear and non-linear dynamics in structural health monitoring.

This chapter addresses the design and implementation of an adaptive fading ex-

tended Kalman filter for the sensorless PMSM on a FPGA chip. In this study, we consider the estimation of PMSM rotor position and speed, in which the reference speed may involve sudden changes during operations or the drive is subject to abrupt loading conditions. To solve this problem, an AF-EKF is proposed to improve the robustness and tracking ability of the estimator and implement a sensorless PMSM control algorithm based the AF-EKF on fully FPGA technology. Here, for realization of the PMSM sensorless control using the system-on-programmable-chip technology, high-speed arithmetic functions and pipelining are employed in the FPGA implementation. The finite state machine method is also used to facilitate the execution timing and chip design. The co-simulation of ModelSim/Simulink shows effectiveness of the proposed chip-based AF-EKF PMSM speed estimation.

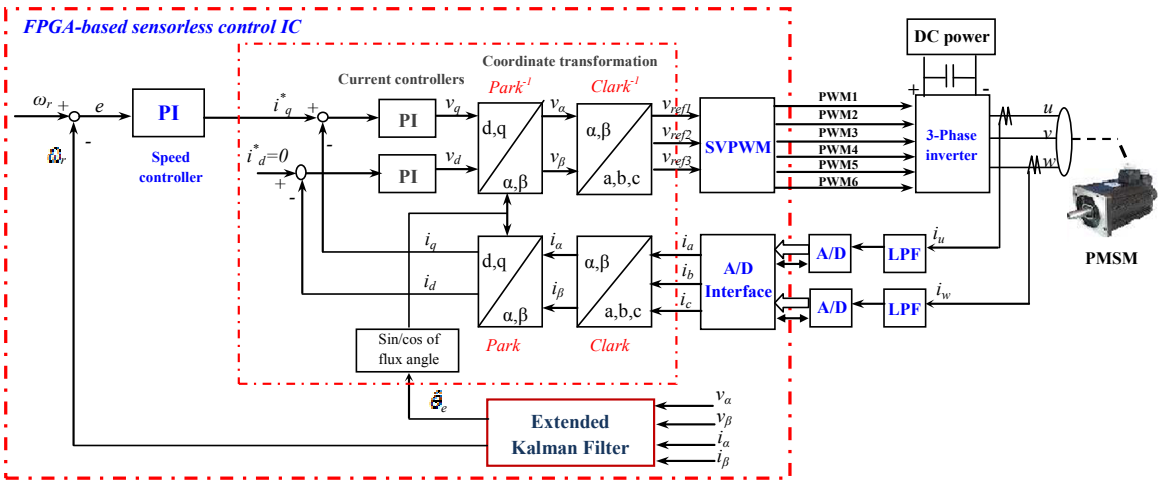


Figure 6.1 : The proposed speed control system for a sensorless PMSM drive using EKF

6.2 Extended Kalman Filter based Rotor Flux Position Estimation

6.2.1 Extended Kalman Filter Algorithm

Kalman filtering is an optimal, stochastic approach to state estimation and filtering in linear systems. For nonlinear systems, the state space equation can be written in the following form:

$$\dot{x}(t) = f(x(t)) + Bu(t) + \sigma(t), \tag{6.1}$$

$$y(t) = h(x(t)) + \mu(t), \tag{6.2}$$

where $x(t)$, $u(t)$ and $y(t)$ are respectively the system's state, input and output. The system noise $\sigma(t)$ and measurement noise $\mu(t)$ are assumed to be zero-mean, white with Gaussian distributions of covariances $Q(t)$ and $R(t)$, respectively. Once a nominal solution to the non-linear equations (6.1)-(6.2) have been obtained, the linearized perturbation equations of the system are

$$\delta\dot{x}(t) = F(x(t))\delta x(t) + B\delta u(t) + \sigma(t), \tag{6.3}$$

$$\delta y(t) = H(x(t))\delta x(t) + \mu(t), \tag{6.4}$$

where the Jacobian and output matrices are defined respectively as follows

$$F(x(t)) = \frac{\partial f}{\partial x}|_{x=x(t)}, \tag{6.5}$$

$$H(x(t)) = \frac{\partial h}{\partial x}|_{x=x(t)}. \tag{6.6}$$

After discretization with a sampling period T_c, (6.3) becomes

$$\begin{aligned} x(t_n) &= \Phi(t_n, t_{n-1}, x(t_{n-1}))x(t_{n-1}) \\ &\quad + (\int_{t_{n-1}}^{t_n} \Phi(t_n, t_{n-1}, x(t_{n-1}))Bd\tau)u(t_{n-1}) + \nu(t_{n-1}), \end{aligned} \tag{6.7}$$

where computation of the state transition matrix for system (6.3), $\Phi(t_n, t_{n-1}, x(t_{n-1}))$, can be simplified by Euler approximation:

$$\Phi(t_n, t_{n-1}, x(t_{n-1})) \cong I + FT_c, \tag{6.8}$$

$$\int_{t_{n-1}}^{t_n} \Phi(t_n, t_{n-1}, x(t_{n-1}))Bd\tau \cong BT_c. \tag{6.9}$$

Therefore, the discrete model of $(6.3) - (6.6)$ becomes approximately,

$$x(t_n) = (I + FT_c)x(t_{n-1}) + BT_c u(t_{n-1}) + \nu(t_{n-1}), \tag{6.10}$$

$$y(t_n) = Hx(t_n) + \xi(t_n), \tag{6.11}$$

where $\nu(t_n)$ and $\xi(t_n)$ are the discrete forms respectively of system noise, with covariance $Q_d(t_n)$, and measurement noise, with covariance $R_d(t_n)$. The EKF is an optimal estimator which minimizes the cost function $J = \sum_{n=1}^{m} E\{\tilde{x}^2(n)\}$ in the least square sense, in which $\tilde{x}(n)$ is defined by the difference between the system state $x(n)$ and its estimate $\hat{x}(n)$, i.e., $\tilde{x}(n) = x(n) - \hat{x}(n)$. Accordingly, the EKF algorithm can be described by the following two-step recursive equations:

(i) Prediction: The predicted estimate is obtained from (6.3) and by using a simple rectangular integration as

$$\hat{x}_{n|n-1} = \hat{x}_{n-1} + (F(\hat{x}_{n-1}) + B.u_{n-1})T_c, \tag{6.12}$$

or from (6.10),

$$\hat{x}_{n|n-1} = (I + FT_c)\hat{x}_{n-1} + BT_c.u_{n-1}. \tag{6.13}$$

The estimate covariance is predicted by

$$P_{n|n-1} = \Phi_{n-1}P_{n-1}\Phi_{n-1}^T + Q_d. \tag{6.14}$$

(ii) Updating: From (6.4) and by using a simple rectangular integration, the updated estimate and correspondingly its covariance are obtained as,

$$\hat{x}_n = \hat{x}_{n|n-1} + K_n(y_n - H\hat{x}_{n|n-1}), \tag{6.15}$$

$$P_n = P_{n|n-1} - K_n H P_{n|n-1}, \tag{6.16}$$

where the Kalman gain is calculated by

$$K_n = P_{n|n-1}H^T(HP_{n|n-1}H^T + R_d)^{-1}. \tag{6.17}$$

6.2.2 Adaptive Fading Extended Kalman Filter Algorithm

The Kalman filter provides the best estimation when the model for the system dynamics and measurement relation are perfect. However, the estimation performance does not maintain when applying to an erroneous model [94]. Since the filter estimation depends highly upon past data, any large deflection from the current system state may cause the state estimation to diverge [84]. To solve this problem, the filter should be capable of eliminating the effect of older data from a current state estimate if these data are no longer meaningful due to modelling errors. In [19], a method was proposed to limit the memory of the Kalman filter by using exponential fading of past data via a forgetting factor, $\lambda(n)$. For applying to our AF-EKF, the equations describing the fading EKF are identical to those of the EKF in equations (6.12)-(6.17) except for the inclusion of the forgetting factor $\lambda(n)$ in the state covariance equation as,

$$P(n|n-1) = \lambda(n)\Phi(n, n-1)P(n-1)\Phi^T(n, n-1) + Q(n), \tag{6.18}$$

where $\lambda(n) \geq 1$. As a result, the influence of the most recent measured data in the state estimation has a higher weight and thus divergence is avoided. The performance of the fading Kalman filter fully depends on the selection of the forgetting factor. In

this study, we adopt the method proposed in [94] to choose the forgetting factor $\lambda(n)$ for our EKF.

In developing the algorithms, we employ an important property, that the residual $r(n)$ defined in the following equation is a white noise sequence when applying the filtering gain

$$r(n) = y(n) - H(n)\hat{x}(n|n-1). \tag{6.19}$$

From Algorithm 3 given in [94], the simplified one step procedure for designing the adaptive fading Kalman filter is applied to compute the forgetting factor of AF-EKF as described in the following:

i) Compute the residual covariance $C_0(n)$ by using on-line measured data as

$$C_0(n) = (1+\rho)^{-1}[\rho C_0(n-1) + r(n)r^T(n)], \tag{6.20}$$

where $C_0(1) = r(1)r(1)^T$ and $0 < \rho < 1$.

ii) Obtain the forgetting factor $\lambda(n)$ as follows. The optimal forgetting factors can be computed by,

$$\lambda(n) = \max\{1, tr[N(n)]/tr[M(n)]\}, \tag{6.21}$$

where

$$N(n) = C_0(n) - H(n)Q(n)H^T(n) - \beta R(n), \tag{6.22}$$

$$M(n) = \Phi(n, n-1)P(n-1)\Phi^T(n, n-1)H(n)H^T(n), \tag{6.23}$$

in which $Q(n)$, $R(n)$ and $P(0)$ are all positive definite matrices, the measurement matrix $H(n)$ is full-ranked, $tr[.]$ is the matrix trace, and β is the weakening factor for avoiding over regulation and smoothing estimated states.

From the circuit equation of PMSM on the $\alpha-\beta$ fixed coordinates system in (3.30), if we choose $x(t) = \begin{bmatrix} i_\alpha & i_\beta & \omega_e & \theta_e \end{bmatrix}^T$ as the vector control of the state variables, then

(6.7) can be expanded to the following formulations

$$\frac{d}{dt}\begin{bmatrix} i_\alpha \\ i_\beta \\ \omega_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} -\frac{r_s i_\alpha}{L_s} + \frac{\omega_e \lambda_f \sin\theta_e}{L_s} \\ -\frac{r_s i_\beta}{L_s} - \frac{\omega_e \lambda_f \cos\theta_e}{L_s} \\ 0 \\ \omega_e \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix}, \tag{6.24}$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ \omega_e \\ \theta_e \end{bmatrix}. \tag{6.25}$$

Thus, from (6.24) – (6.25) the dynamic model of the PMSM in the state variable form can be expressed generally as,

$$\dot{x}(t) = f(x(t)) + Bu(t), \tag{6.26}$$

$$y = Hx(t), \tag{6.27}$$

where $y(t) = \begin{bmatrix} i_\alpha & i_\beta \end{bmatrix}^T$ and $u(t) = \begin{bmatrix} v_\alpha & v_\beta \end{bmatrix}^T$.

To proceed with the filter design, the Jacobian, output matrix, and state transition matrix are obtained in accordance respectively with (6.5), (6.6) and (6.8) as,

$$F = \frac{\partial f}{\partial x}|_{x=x(t)} = \begin{bmatrix} \frac{-r_s}{L_s} & 0 & \frac{\lambda_f \sin\theta_e}{L_s} & \frac{\omega_e \lambda_f \cos\theta_e}{L_s} \\ 0 & \frac{-r_s}{L_s} & -\frac{\lambda_f \cos\theta_e}{L_s} & \frac{\omega_e \lambda_f \sin\theta_e}{L_s} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{6.28}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{6.29}$$

$$\Phi(n|n-1) = I + FT_c$$

$$= \begin{bmatrix} 1 - \frac{r_s T_c}{L_s} & 0 & \frac{\lambda_f T_c \sin\theta_e}{L_s} & \frac{\omega_e \lambda_f T_c \cos\theta_e}{L_s} \\ 0 & 1 - \frac{r_s T_c}{L_s} & -\frac{\lambda_f T_c \cos\theta_e}{L_s} & \frac{\omega_e \lambda_f T_c \sin\theta_e}{L_s} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & T_c & 1 \end{bmatrix} = \begin{bmatrix} G & 0 & \phi_{13} & \phi_{14} \\ 0 & G & \phi_{23} & \phi_{24} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & T_c & 1 \end{bmatrix}. \quad (6.30)$$

The initial values of Q, R and $P(0)$ need to be chosen. Through the recursive calculation, estimated values of the state vector $\hat{x}(t) = \begin{bmatrix} \hat{i}_\alpha & \hat{i}_\beta & \hat{\omega}_e & \hat{\theta}_e \end{bmatrix}^T$ are obtained at each sampling period. Thus, the estimated angular position, $\hat{\theta}_e$, and rotor speed, $\hat{\omega}_e$ are computed directly from these sampled values.

6.3 Control Architecture and Implementation of AF-EKF

Figure 6.1 shows the FPGA-based architecture of the proposed speed control system for the sensorless PMSM drive. The control system includes a PI-speed controller, a CCCT module, a SVPWM generator as well as an AF-EKF. All modules are described by the VHDL language and simulated in ModelSim to test their feasibility before implementation in Altera Cyclone II EP2C70. The sampling frequency for the current and speed controllers is designated at 16 kHz and 2 kHz, respectively. The operating clock rate of the designed FPGA controller is 50 MHz and the frequency divider generates 50 MHz, 12.5 MHz, 16 kHz and 2 kHz clock to supply to all module circuits of the application system-on-chip.

6.3.1 Design Procedure of AF-EKF Algorithm

Step 1: Set the initial values of Q, R, $P(0)$ and $n = 1$.

Step 2: Measure the values of $i_\alpha(n)$, $i_\beta(n)$, $v_\alpha(n)$, $v_\beta(n)$ from the PMSM drive.

Step 3: Predict the temporary state variables by using (6.13). In particular, the scalar form of the prediction equation can be obtained from (6.24) as follows:

$$\hat{i}_\alpha(n|n-1) = (1 - \frac{r_s T_c}{L_s})\hat{i}_\alpha(n-1) + \frac{\omega_e(n-1)\lambda_f \sin\theta_e(n-1)}{L_s} + \frac{T_c}{L_s} v_\alpha(n-1), \tag{6.31}$$

$$\hat{i}_\beta(n|n-1) = (1 - \frac{r_s}{L_s} T_c)\hat{i}_\beta(n-1) - \frac{\omega_e(n-1)\lambda_f \cos\theta_e(n-1)}{L_s} + \frac{T_c}{L_s} v_\beta(n-1), \tag{6.32}$$

$$\hat{\omega}_e(n|n-1) = \hat{\omega}_e(n-1), \tag{6.33}$$

$$\hat{\theta}_e(n|n-1) = \hat{\theta}_e(n-1) + \hat{\omega}_e(n-1)T_c. \tag{6.34}$$

Step 4: Calculate the error covariance $C_0(n)$ from (6.20).

Step 5: After computation of the forgetting factor from (6.21), obtain the temporary covariance matrix $P(n|n-1)$ from (6.14), where $\Phi(n, n-1)$ is given by (6.30). Due to its symmetry, positive definite matrix $P(n|n-1)$ is chosen in the following form:

$$P(n|n-1) = \begin{bmatrix} P_{11} & P_{21} & P_{31} & P_{41} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{23} & P_{33} & P_{43} \\ P_{41} & P_{24} & P_{43} & P_{44} \end{bmatrix}. \tag{6.35}$$

Step 6: Update the present covariance matrix $P(n)$ from (6.16) and compute the Kalman gain from (6.17) for the filter by using(6.29) and (6.35) as,

$$K(n) = \begin{bmatrix} P_{11} & P_{21} \\ P_{21} & P_{22} \\ P_{31} & P_{23} \\ P_{41} & P_{24} \end{bmatrix} \left(\begin{bmatrix} P_{11} & P_{21} \\ P_{21} & P_{22} \end{bmatrix} + R \right)^{-1} = \begin{bmatrix} P_{11} & P_{21} \\ P_{21} & P_{22} \\ P_{31} & P_{23} \\ P_{41} & P_{24} \end{bmatrix} \begin{bmatrix} T_{11} & T_{21} \\ T_{21} & T_{22} \end{bmatrix} \triangleq \begin{bmatrix} k_{11} & k_{21} \\ k_{21} & k_{22} \\ k_{31} & k_{23} \\ k_{41} & k_{24} \end{bmatrix}. \tag{6.36}$$

Step 7: Update the predicted state variables from (6.15) as follows

$$\hat{i}_\alpha(n) = \hat{i}_\alpha(n|n-1) + k_{11}\tilde{i}_\alpha(n) + k_{21}\tilde{i}_\beta(n), \tag{6.37}$$

$$\hat{i}_\beta(n) = \hat{i}_\beta(n|n-1) + k_{21}\tilde{i}_\alpha(n) + k_{22}\tilde{i}_\beta(n), \tag{6.38}$$

$$\hat{\omega}_e(n) = \hat{\omega}_e(n|n-1) + k_{31}\tilde{i}_\alpha(n) + k_{23}\tilde{i}_\beta(n), \tag{6.39}$$

$$\hat{\theta}_e(n) = \hat{\theta}_e(n|n-1) + k_{41}\tilde{i}_\alpha(n) + k_{24}\tilde{i}_\beta(n), \tag{6.40}$$

where

$$\tilde{i}_\alpha(n) = i_\alpha(n) - \hat{i}_\alpha(n|n-1), \tag{6.41}$$

$$\tilde{i}_\beta(n) = i_\beta(n) - \hat{i}_\beta(n|n-1), \tag{6.42}$$

in which k_{ij} are elements of the Kalman gain $K(n)$.

6.3.2 Algorithm Implementation of AF-EKF

A FSM is employed to describe the AF-EKF algorithm, as shown in Figure 6.2, where the data type adopts a 16-bit length with Q15 format and 2's complement operations. Despite the high complexity of the EKF algorithm, the FSM adequately incorporates the control structure and can be described by VHDL. Here, the adder, multiplier and divider apply the Altera LPM standard. It manipulates 149 steps machine to carry out the overall computations.

In our FSM, the steps $S_0 - S_{12}$ execute the computation of the Jacobian matrix, predicted state variables and calculate the prediction error; steps $S_{13} - S_{17}$ are for the calculation of the residual covariance $C_0(n)$; steps $S_{18} - S_{31}$ calculate the forgetting factor $\lambda(n)$; steps $S_{30} - S_{86}$ are the computation of temporary covariance matrix; steps $S_{87} - S_{138}$ update the present covariance matrix $P(n)$ and the calculation of Kalman gain $K(n)$; and finally, steps $S_{138} - S_{148}$ describe the present state tuning,

execute the computation of rotor flux position and rotor speed. Since the execution of each step can be completed within 80 ns corresponding to frequency 12.5 MHz in FPGA, a total of 149 steps needs 11.92 μs for the AF-EKF operation.

Overall, the resource usage of the speed PI controller, CCCT, SVPWM and the proposed adaptive fading EKF are 6,264 LEs and 73,728 RAM bits resource for the Cyclone II EP2C70, as summarized in Table 6.1 below.

Table 6.1 : Utility Evaluation of Sensorless Control IC in FPGA

Module circuits	**Logic elements (LEs)**	**Memory (bits)**
1. CCCT	864	24,576
2. SVPWM generation	1,221	0
3. Speed PI controller	254	0
4. AF-EKF	3,925	49,152
TOTAL	6,264	73,728

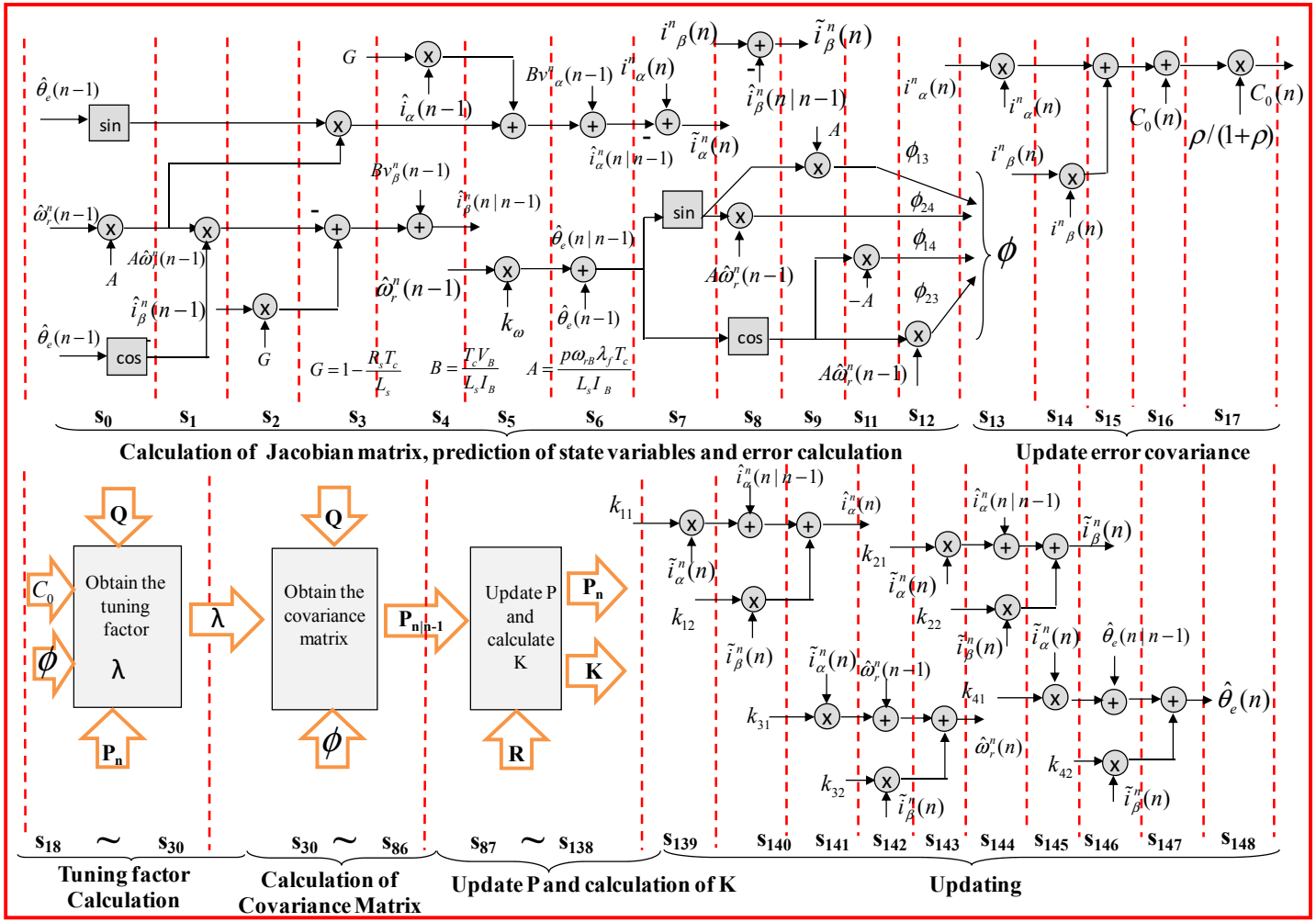


Figure 6.2 : State machine diagram of AF-EKF algorithm

6.4 Results and Discussion

In order to verify effectiveness of the proposed AF-EKF and sensorless control scheme, the rotor flux angle and speed estimation are tested at 100 rpm. Figure 6.3 shows that the estimated flux angle with the proposed AF-EKF almost matches the actual flux angle (1.29^o phase error, 50 μs delay time) and is much better than with the EKF method (3.78^o phase error, 150 μs delay time). We then consider the case when the motor speed command is varied from -100 rpm to 100 rpm. Figure 6.4 presents a good tracking of speed in both methods in the normal load condition.

In addition, to prove the proposed method is an effective estimator, in the case of abrupt load variations of PMSM drives, the system is then evaluated with a sudden loading. In Figure 6.5(a), the simulation result shows that the PMSM is initially running with good tracking responses with a normal load, but when the operation condition is changed abruptly to a heavy load after 0.15 second, the speed estimation becomes worse and loses the speed tracking in the standard EKF case while the proposed AF-EKF still provides a good response. It demonstrates that although the sensorless control based on the standard EKF in the PMSM drive can give good speed tracking, it is still easily affected by external load variations. Responses of $d-q$ axes currents of the proposed AF-EKF are presented in Figure 6.5(b), which show successful vector control performance with the $d-$current component being controlled to zero while the $q-$current consistently following the step speed patterns.

Table 6.2 summarizes the comparison of FPGA sensorless control of PMSMs between SMO in [70], the standard EKF in [68], and AF-EKF proposed in this work, including hardware resource consumption and execution time, the estimated speed error and estimated flux angle error with load variation cases. Although hardware resource usage and execution time with an AF-EKF implementation (7,037 LEs; 49,152

bits; 14.4 μs) are the largest because of the calculation of forgetting factor $\lambda(n)$ and adjusting estimated covariance $P(n|n-1)$, the control performance when using the proposed AF-EKF is quite better than others with fast response and small estimation errors (2.96 rpm speed response and 1.49^o flux angle) under speed variations from 100 rpm to -100 rpm as well as the external load changing abruptly to a heavy load. That is the main advantage of this method which improves the estimation robustness. In addition, for the initial condition matrix $P(0)$, Q and R of the EKF algorithm, it is very difficult to select suitable values while with the proposed AF-EKF, they can be tuned via adjustments of the fading factors.

Table 6.2 : Comparison of Estimation Algorithms in Control Implementation

Sensorless techniques	**LEs**	**Estimation errors**	**Ex. time (μs)**
1. SMO	4,626 (6.8%)	5.75 rpm, 8.63^o	5.84
2. Standard EKF	6,633 (9.7%)	3.35 rpm, 3.78^o	13.36
3. Proposed AF-EKF	7,037 (10.2%)	2.96 rpm, 1.49^o	14.4

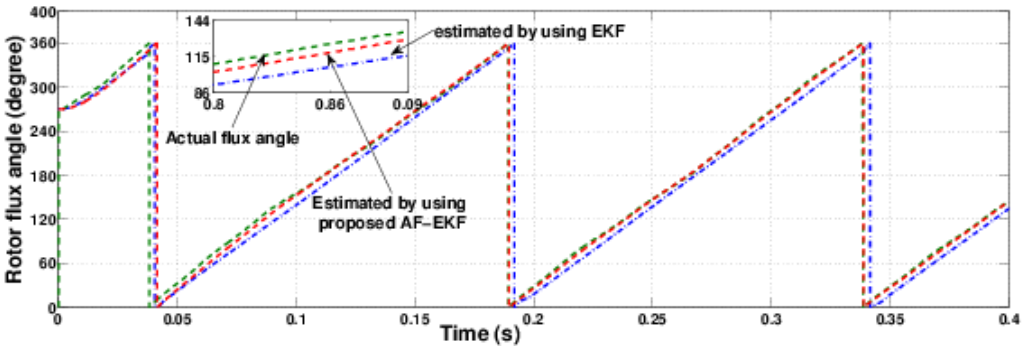


Figure 6.3 : Actual and estimated rotor flux angle from standard EKF and AF-EKF under speed condition at low speed 100 rpm

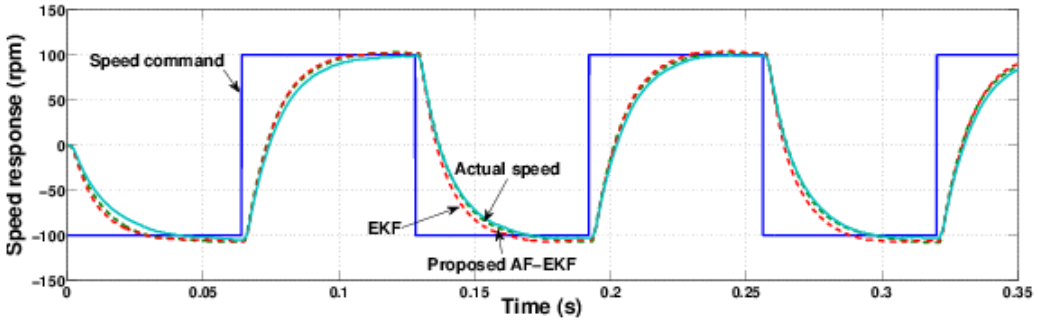
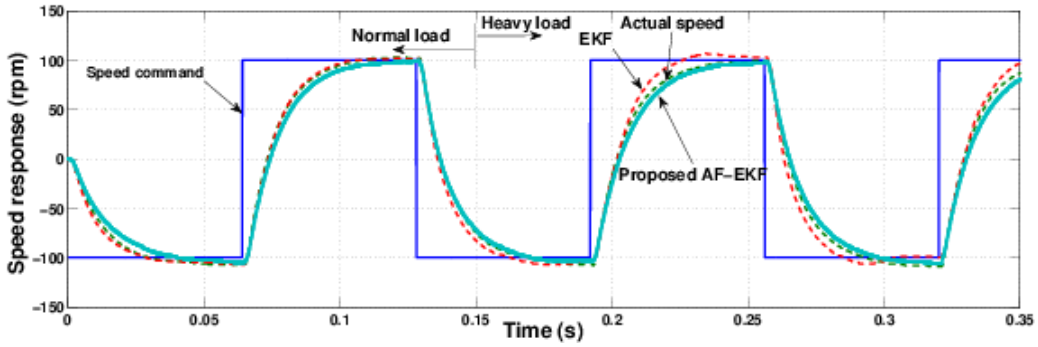
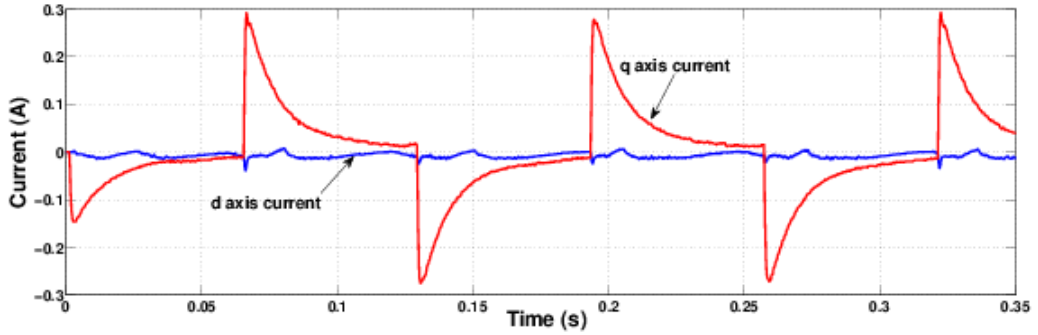


Figure 6.4 : Speed responses by using standard EKF and AF-EKF under speed condition at low speed 100 *rpm* and inverse -100 *rpm*



(a)



(b)

Figure 6.5 : (a) Speed responses by using the standard EKF and proposed AF-EKF under varying external load; (b) Current responses with AF-EKF

6.5 Chapter Conclusion

In this chapter, we have presented the design and implementation of an adaptive fading extended Kalman filter for a FPGA-based sensorless permanent magnet synchronous motor control system. The proposed system comprises a PI speed controller, a field-oriented PI current controller, and a rotor flux angle and rotor speed estimator using an adaptive fading EKF.

The proposed algorithm for estimation of the motor's rotor position and speed is introduced with the incorporation of an adjustable forgetting factor in the conventional EKF. The adaptive fading EKF has a faster response and higher accuracy than the EKF. This is important in such applications that require frequent loading and speed reference variations, as the sensorless estimation offers improvements of the system precision and dynamic responses.

The whole system architecture has been integrated and successfully realized in one FPGA chip. The proposed flux angle and speed estimator algorithms have shown their effectiveness in estimation of the rotor position and speed as verified by the co-simulation method.

Chapter 7

FPGA-Based Sensorless PMSM Speed Control using Adaptive Extended Kalman Filter

7.1 Introduction

In general estimation application of extended Kalman filtering, there remains a difficulty in the selection and adjustment of design parameters in the EKF equations, which appear to affect the filtering performance. As such there has been intensive research efforts paid to the development of adaptive Kalman filtering algorithms [5]. Different adaptive EKF algorithms have been successfully used in many practical systems such as navigation, radar tracking, signal processing, battery usage, and robotics [26],[28],[63],[83],[85]. In the context of sensorless drive and control, a procedure for the offline tuning of covariance matrices in EKF-based PMSM drives has been presented in [6]. Nevertheless, uncertainty of the covariance matrices of the system noise (Q) and the measurement errors (R) appearing in the on-line estimation process may seriously degrade the filtering performance. There is also the need for a feasible implementation of a suitable adaptive EKF, especially in the area of position and speed estimation in electric motor drives.

In this chapter, an adaptive extended Kalman filter for a sensorless PMSM with the combination of an online stochastic modelling and a system noise scaling algorithm, for estimation performance improvement, is designed and implemented in a cacasde control system for sensorless PMSM speed control using the SoPC concept

with the FPGA technology. This is achieved by using a sequential finite state machine of multipliers, adders, comparators, registers and a look-up table, to implement high speed arithmetic functions and pipelining operations for improvement of drive performance and resource usage saving. In order to verify the performance of the proposed on-chip control system, the co-simulation method ModelSim/ Simulink is adopted for extensive simulation.

7.2 Adaptive Extended Kalman Filter

There have been several online estimation of dynamic and measurement noise covariance matrices Q and R. Basically, the adaptation is based on (i) fixing Q and varying R, (ii) fixing R and varying Q, or (iii) varying Q and R simultaneously, to ultimately find the best stable estimate. This chapter is aimed on the one hand to take advantage of online correction of the process noise covariance and on the other hand to guarantee the feasibility of the adaptation algorithm when implemented on an FPGA chip.

7.2.1 Adaptive Algorithm

The updating sequence, representing the difference between the measurement and its predicted value, is defined as:

$$d_n = y_n - H\hat{x}_{n|n-1}. \tag{7.1}$$

Substituting the measurement model (6.11) into (7.1) yields:

$$d_n = H(x_n - \hat{x}_{n|n-1}) + \mu_n, \tag{7.2}$$

where the updating sequences d_n are white Gaussian noise with zero mean when the filter is in an optimal mode.

Adaptive Kalman filtering algorithms can be implemented based on online estimation of the process and measurement noise covariances. One commonly-used technique is based on the covariance matching principle to make the elements of the covariance matrix for sequences d_n consistent with their theoretical values [55]. By taking variances on both sides of (7.2), we obtain:

$$E\{d_n d_n^T\} = H P_{n|n-1} H^T + R_n, \tag{7.3}$$

which can be estimated with $\hat{C}_v$ by using a limited number of innovation samples:

$$\hat{C}_v = \frac{1}{m} \sum_{i=1}^{m} d_{n-i} d_{n-i}^T = H \hat{P}_{n|n-1} H^T + R_n, \tag{7.4}$$

where m is the estimation window size and $\hat{P}_{n|n-1}$ is the estimated prediction of the state covariance.

If R_n and $P_{n|n-1}$ are assumed to be known, Q_n can be corrected online with a scaling factor by calculating the ratio between the estimated updating sequence covariance and the predicted state covariance, defined as

$$\alpha = \frac{trace[H \hat{P}_{n|n-1} H^T]}{trace[H P_{n|n-1} H^T]} = \frac{trace[\hat{C}_v - R_n]}{trace[H P_{n|n-1} H^T]}, \tag{7.5}$$

where $trace[.]$ is the sum of all elements on the main diagonal of a square matrix.

By substituting (6.14) and (7.4) into (7.5), the scaling factor α can be calculated and updated as:

$$\alpha_{n-1} = \frac{trace[H(\Phi_{n-1} \hat{P}_{n-1} \Phi_{n-1}^T + \hat{Q}_{n-1}) H^T]}{trace[H(\Phi_{n-1} P_{n-1} \Phi_{n-1}^T + Q_{n-1}) H^T]}. \tag{7.6}$$

From (7.5) and (7.6), an adaptation rule for the estimated process noise covariance $\hat{Q}_n$ is proposed as [15]:

$$\hat{Q}_n = Q_{n-1} \sqrt{\alpha_{n-1}}. \tag{7.7}$$

If the current estimated $\hat{Q}$ is smaller than its true value Q, the algorithm tends to increase the estimated covariance, and hence the Kalman gain, with a scaling factor α greater than 1, and vice versa. This results in an increase or decrease in the Kalman gain correspondingly to prevent the divergence of the filter.

7.2.2 Design Procedure of Adaptive EKF

Step 1: Set the initial values of Q, R, P and $n = 1$.

Step 2: Measure the values of $i_\alpha(n)$, $i_\beta(n)$, $v_\alpha(n)$, $v_\beta(n)$ from the PMSM drive.

Step 3: Predict the temporary state variables as the same as the implementation of AF-EKF in Chapter 6 from (6.31)-(6.34).

Step 4: Calculation of the estimated updating covariance $\hat{C}_v$ from (7.4) and (7.1).

Step 5: Using $\hat{C}_v$ to adjust Q from (7.5) and (7.7).

Step 6: Obtain the temporary covariance matrix $P(n|n-1)$ from (6.14) with updated $\hat{Q}_{n-1}$, where Φ_{n-1} is given by (6.30). Due to its symmetry, $P(n|n-1)$ is chosen in the form as the same as equation (6.35).

Step 7: Update the present covariance matrix $P(n)$ from (6.16) and compute the Kalman gain from (6.17) for the filter by using (6.29) and (6.35) and we have the gain calculation as in equation (6.36).

Step 8: Update the predicted state variables from (6.15) as in the calculation manner of equations from (6.37) to (6.40).

7.3 FPGA Realization of Sensorless Control Design

7.3.1 Controller Architecture

The control system features the cascade control principle including in the inner loop a PI current controller and CCCT module, and in the outer loop a PI speed controller with a SVPWM generation module and adaptive EKF-based rotor flux angle and rotor speed estimation module. All modules are described by Verilog HDL and simulated in ModelSim as well as tested in Altera Cyclone IV EP4CE115 board. The frequency divider generates 50 MHz (*Clk*), 12.5 MHz (*Clk-sp*) and 16 kHz (*Clk-ctr*) clock to supply for all circuits.

7.3.2 Algorithm Implementation

A finite state machine is employed to describe the adaptive EKF algorithm as shown in Figure 7.1, where the data type adopts a 16-bit length with Q15 format and 2's complement operations. Although the sensorless architecture using the adaptive EKF algorithm described is rather complex, the FSM can sufficiently incorporate the whole control structure and can be described by VHDL. The multiplier and adder apply the Altera LPM standard. The FSM for the adaptive EKF algorithm manipulates 97 steps machine to carry out the overall computations. The steps $S_0 - S_{12}$ are for calculation of the Jacobian matrix and prediction of state variables; steps $S_{13} - S_{21}$ execute the computation of the innovation covariance matrix $\hat{C}_v$; steps $S_{22} - S_{54}$ are for the prediction of process noise matrix Q and the temporary covariance matrix $P_{n|n-1}$; steps $S_{55} - S_{86}$ update the present covariance matrix P_n and calculate the Kalman gain K_n; and finally, steps $S_{87} - S_{96}$ describe the present state tuning, execute the computation of rotor flux position and rotor speed. Since the execution of each step

can be completed within 80 ns corresponding to frequency 12.5 MHz in FPGA, a total of 97 steps needs 7.76 ms for the adaptive EKF operation.

The implementation for the rest of the control architecture has been reported in the previous chapters. Overall, the resource usage of the speed PI controller, CCCT, SVPWM and the proposed adaptive EKF are 5,824 logic elenments (LEs) and 73,728 RAM bits resource for the Cyclone IV EP4CE115.

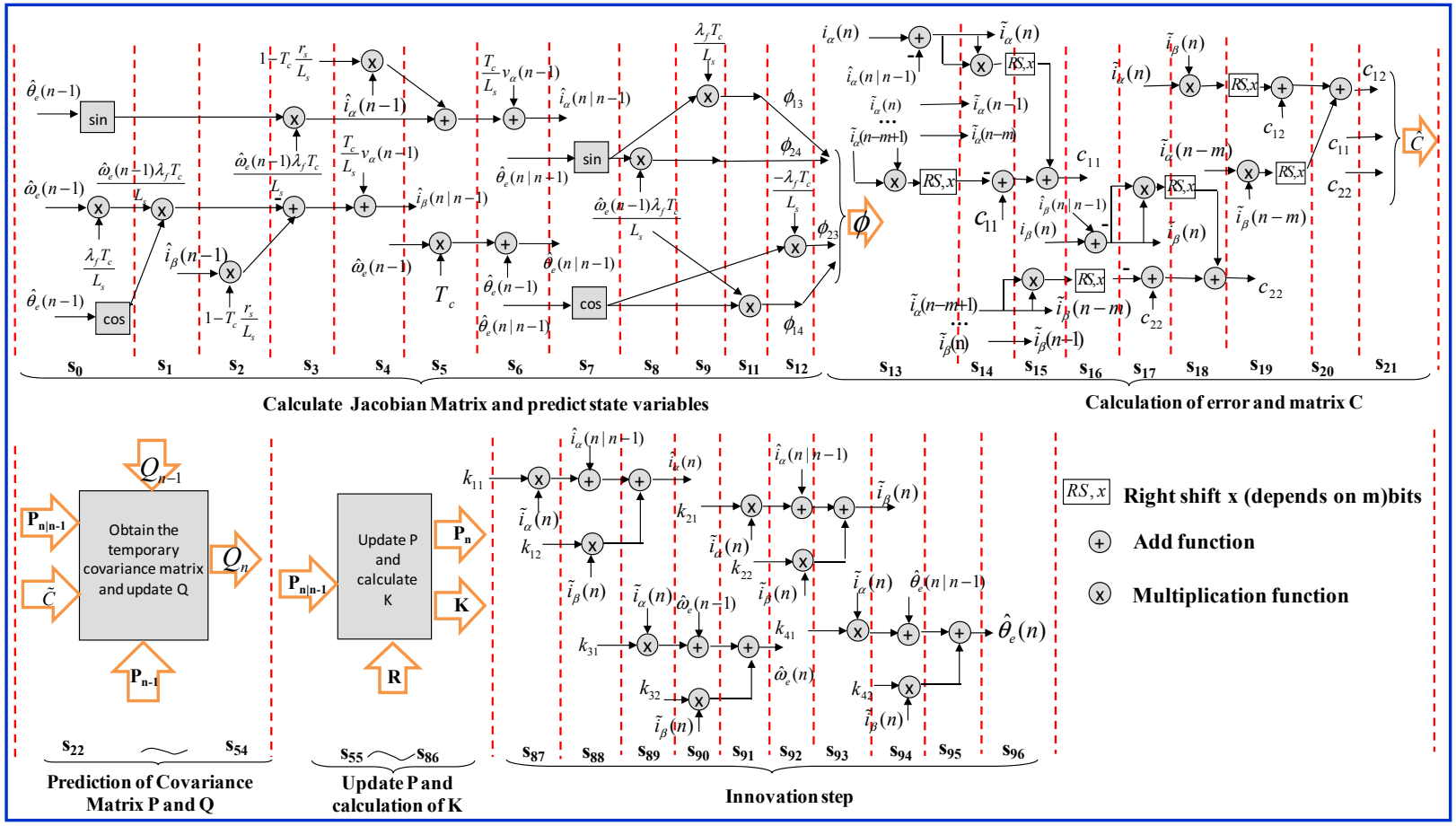


Figure 7.1 : State machine diagram of the adaptive EKF algorithm

7.4 Co-Simulation and Results

Verification of the design is performed in simulation by using the ModelSim and Simulink co-simulation method, which provides an interface environment between MATLAB/Simulink and the VHDL simulator. The PMSM, inverter and speed commands are implemented in Simulink and the sensorless speed control system is described with the VHDL code in ModelSim using five modules. These modules are created for a PI current controller, CCCT and SVPWM units, the rotor flux position estimator using the proposed adaptive EKF and the PI speed controller.

To be consistent with the hardware implementation, the sampling frequency of current and speed control is selected respectively as 16 kHz and 2 kHz. The clocks of 50 MHz supply to all multipliers, dividers, adders, comparators, registers and the clock of 12.5 MHz is used for each operation step of the finite state machines used. The parameters of an 8-pole PMSM used in this simulation are the same as in the previous chapters. The initial values for $Q(0)$, $R(0)$, $P(0)$ and $\alpha(0)$ are selected respectively as $diag(0.045 \quad 0.045 \quad 3.1.10^{-5} \quad 0.0013)$, $diag(1 \quad 1)$, $diag(0.01 \quad 0.01 \quad 0.01 \quad 0.01)$, and 1, where $diag(d_{ii})$ is a diagonal matrix with entries d_{ii} on its main diagonal.

In the simulation of the flux angle estimation, the PMSM operation conditions are tested in the range of 500 *rpm*, 1000 *rpm* and 2000 *rpm*. To evaluate the correctness of the measured flux angle, the motor speed and the flux angle signals are obtained from the PMSM model for judging the accuracy of the estimated value, as shown in Figure 7.2(a). In addition, in order to illustrate the effectiveness of the proposed adaptive EKF in such FPGA-based sensorless control system, both speed responses by using the proposed adaptive EKF and a full-order EKF are compared with actual speed, as depicted in Figure 7.2(b). The obtained results indicate that not only for speed tracking but also for flux angle (FA) tracking, the flux estimation performance

has been improved by adopting the adaptive EKF method, leading to better speed estimation performance at different command values.

7.5 Chapter Conclusion

In this chapter we have presented an FPGA-based speed control system for a sensorless permanent magnet synchronous motor. The proposed system comprises a PI speed controller, a field-oriented PI current controller, and a rotor flux angle and rotor speed estimator based on adaptive extended Kalman filtering. All the system modules have been integrated and successfully realized in one FPGA chip. The whole control architecture consumes less than 6000 logic elements on a Cyclone IV environment. The proposed flux angle and speed estimator algorithms have proven their effectiveness in the estimation of the rotor's flux angle and speed as verified by the co-simulation method. Responses obtained from this control-system-on-programmable chip have shown better performance when compared with conventional EKF for FPGA-based sensorless speed control in PMSM drives.

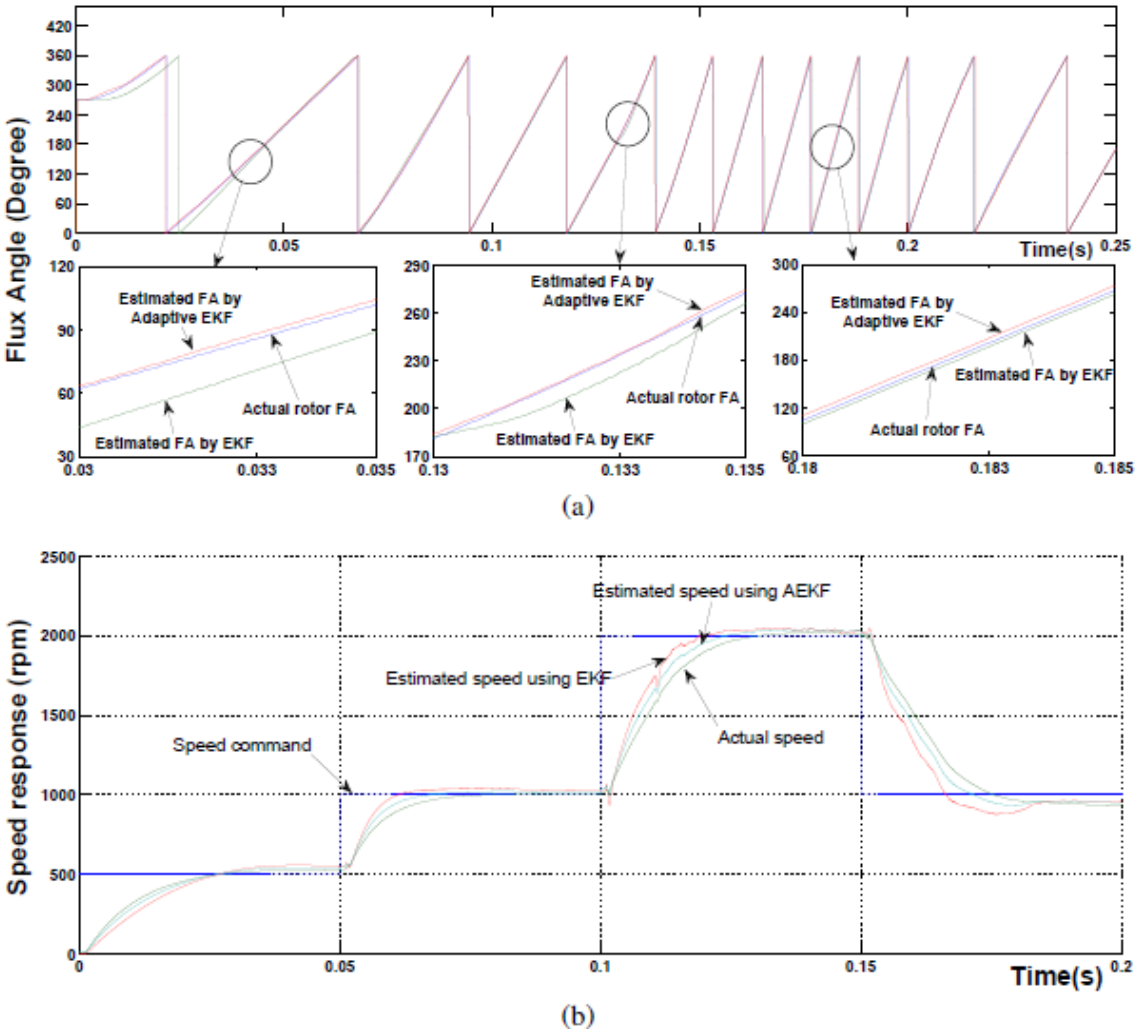


Figure 7.2 : Comparison between adaptive EKF and conventional EKF: (a) Rotor flux angle, (b) Rotor speed.

Chapter 8

Reduced-Order Extended Kalman Filters-based Sensorless PMSM Speed Control using FPGA

8.1 Introduction

Adaptive fading- and adaptive-EKF showed a good estimation and control performance in Chapter 6 and Chapter 7. However, as remarked in [30], an EKF requires heavy online matrix computing, and the complex computation becomes a challenge for a fixed-point processor system, particularly for implementation on an FPGA chip. In addition, the linearization in EKF may cause the filter algorithm to quickly diverge while the computational cost for its realization increases very considerably with the system complexity. A two-stage Kalman estimator could be used to reduce the number of arithmetic operations [31]. Here, motivated by the work in [36], the problem is addressed via a reduction in the system order by using two parallel EKFs [68], in order to meet the limited computational resources and processing time of FPGA implementation.

This chapter aims to design and implement a fully FPGA based sensorless control paradigm for PMSM drives using reduced-order EKFs that can ensure both a significant reduction in the required FPGA resources and a fast execution time. For this, we seek, on the one hand, to lessen the number of matrix calculations in our EKF algorithm [92] and to reduce the EKF order, as generally outlined in [78]. Here, by estimating the back EMF according to the stator current observed as state variables,

the decoupling of the system equation can be obtained to facilitate a reduced-order EKF. On the other hand, a finite-state machine is developed in this work to result in less logic gates used, leading to a faster processing time. In order to verify the performance of the proposed on-chip control system, the co-simulation method ModelSim/Simulink is used for extensive simulation. Then, experimental validation on a PMSM is conducted using an Altera kit and laboratory instrumentation.

8.2 Reduced-Order Extended Kalman Filter for Sensorless PMSM Drive

For the reduction of computation resources, as well as accuracy improvement in the rotor position estimation, a parallel reduced-order extended Kalman filter is proposed in this work. Compared with an EKF, the system order is reduced and the iteration process is greatly simplified, resulting in significant savings of resource utility, while maintaining high estimation performance. The development of the algorithm will be detailed here.

8.2.1 PMSM Decoupled Dynamics

The basic emf model in (3.31) is time-varying in practice and from the assignment of $\alpha - \beta$ coordinates it can be incorporated in (3.30) to form the following dynamics:

$$\frac{d}{dt}\begin{bmatrix} i_\alpha \\ i_\beta \\ e_\alpha \\ e_\beta \end{bmatrix} = \begin{bmatrix} \frac{-r_s}{L_s} & 0 & \frac{-1}{L_s} & 0 \\ 0 & \frac{-r_s}{L_s} & 0 & \frac{-1}{L_s} \\ 0 & 0 & 0 & -\omega_e \\ 0 & 0 & \omega_e & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ i_\beta \\ e_\alpha \\ e_\beta \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} & 0 \\ 0 & \frac{1}{L_s} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix}. \tag{8.1}$$

To achieve the order reduction purpose, model (8.1) can be decoupled according to currents i_α and i_β into:

$$\frac{d}{dt}\begin{bmatrix} i_\alpha \\ e_\alpha \\ e_\beta \end{bmatrix} = \begin{bmatrix} \frac{-r_s}{L_s} & \frac{-1}{L_s} & 0 \\ 0 & 0 & -\omega_e \\ 0 & \omega_e & 0 \end{bmatrix} \begin{bmatrix} i_\alpha \\ e_\alpha \\ e_\beta \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} v_\alpha \end{bmatrix}, \tag{8.2}$$

and

$$\frac{d}{dt}\begin{bmatrix} i_\beta \\ e_\beta \\ e_\alpha \end{bmatrix} = \begin{bmatrix} \frac{-r_s}{L_s} & \frac{-1}{L_s} & 0 \\ 0 & 0 & \omega_e \\ 0 & -\omega_e & 0 \end{bmatrix} \begin{bmatrix} i_\beta \\ e_\beta \\ e_\alpha \end{bmatrix} + \begin{bmatrix} \frac{1}{L_s} \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} v_\beta \end{bmatrix}. \tag{8.3}$$

Thus, the dynamic models of the PMSM in the state variable form can be expressed generally as,

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{8.4}$$

$$y(t) = Cx(t), \tag{8.5}$$

where $x(t) = \begin{bmatrix} i_\alpha & e_\alpha & e_\beta \end{bmatrix}^T$, $y(t) = \begin{bmatrix} i_\alpha \end{bmatrix}$, and $u(t) = \begin{bmatrix} v_\alpha \end{bmatrix}$, or $x(t) = \begin{bmatrix} i_\beta & e_\beta & e_\alpha \end{bmatrix}^T$, $y(t) = \begin{bmatrix} i_\beta \end{bmatrix}$ and $u(t) = \begin{bmatrix} v_\beta \end{bmatrix}$, respectively, in which the system matrices are given by:

$$A = \begin{bmatrix} \frac{-r_s}{L_s} & \frac{-1}{L_s} & 0 \\ 0 & 0 & \mp\omega_e \\ 0 & \pm\omega_e & 0 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{L_s} \\ 0 \\ 0 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.$$

8.2.2 Parallel Reduced-Order EKFs

To proceed with the EKF design, the Jacobian, output matrix, and state transition matrix are obtained in accordance respectively with (6.5), (6.6) and (6.8) as,

$$F = \frac{\partial f}{\partial x}|_{x=x(t)} = A = \begin{bmatrix} \frac{-r_s}{L_s} & \frac{-1}{L_s} & 0 \\ 0 & 0 & \mp\omega_e \\ 0 & \pm\omega_e & 0 \end{bmatrix}, \tag{8.6}$$

$$H = \frac{\partial h}{\partial x}|_{x=x(t)} = C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \tag{8.7}$$

$$\Phi(t_n, t_{n-1}, x(t_n)) \cong I + FT_c = \begin{bmatrix} 1 - \frac{r_s T_c}{L_s} & \frac{-T_c}{L_s} & 0 \\ 0 & 1 & \mp\omega_e T_c \\ 0 & \pm\omega_e T_c & 1 \end{bmatrix} \triangleq \begin{bmatrix} \phi_1 & \phi_2 & 0 \\ 0 & 1 & \phi_3 \\ 0 & -\phi_3 & 1 \end{bmatrix}. \tag{8.8}$$

In our sensorless control scheme, according to the parity of the index n of the sampling time t_n, currents i_α and i_β are predicted/updated in subsequent samples, making use of the time continuity of the inductive current. Then, the previous data of the rotor speed and EMF components e_α, e_β are applied to two corresponding third-order EKFs in the prediction/updating steps to obtain the estimates $\hat{e}_\alpha$, $\hat{e}_\beta$. Hence, operating at a time is only EKF of third order (p=3), instead of fourth order (p=4). This, therefore, reduces the number of arithmetic operations involved and saving the logic elements used. Thus, the estimated angular position, $\hat{\theta}_e$, and rotor speed, $\hat{\omega}_e$, to be used in the next step, are obtained by the following equations:

$$\hat{\theta}_e(n) = tan^{-1}\left(\frac{-\hat{e}_\alpha(n)}{\hat{e}_\beta(n)}\right), \tag{8.9}$$

$$\hat{\omega}_e(n) = \frac{1}{\lambda_f}\sqrt{\hat{e}_\alpha^2(n) + \hat{e}_\beta^2(n)}. \tag{8.10}$$

8.2.3 Design Procedure of Reduced-Order EKF

Step 1: Set the initial values of Q_d, R_d, P_o and $n = 1$.

Step 2: Measure the values of $i_\alpha(n)$, $i_\beta(n)$, $v_\alpha(n)$, $v_\beta(n)$ from the PMSM drive.

Step 3: Predict the temporary state variables by using (6.13). In particular, the scalar form of the prediction equation can be obtained from (8.1) as follows:

$$\begin{aligned}\hat{i}_\alpha(n|n-1) &= (1-\frac{r_s}{L_s}T_c)\hat{i}_\alpha(n-1) - \frac{T_c}{L_s}\hat{e}_\alpha(n-1) \\ &\quad + \frac{T_c}{L_s}v_\alpha(n-1) \quad \text{(if } n \text{ is odd)}, \end{aligned} \tag{8.11}$$

$$\begin{aligned}\hat{i}_\beta(n|n-1) &= (1-\frac{r_s}{L_s}T_c)\hat{i}_\beta(n-1) - \frac{T_c}{L_s}\hat{e}_\beta(n-1) \\ &\quad + \frac{T_c}{L_s}v_\beta(n-1) \quad \text{(if } n \text{ is even)}, \end{aligned} \tag{8.12}$$

$$\hat{e}_\alpha(n|n-1) = \hat{e}_\alpha(n-1) - \hat{\omega}_e(n-1)T_c\hat{e}_\beta(n-1), \tag{8.13}$$

$$\hat{e}_\beta(n|n-1) = \hat{e}_\beta(n-1) + \hat{\omega}_e(n-1)T_c\hat{e}_\alpha(n-1). \tag{8.14}$$

Step 4: Obtain the temporary covariance matrix $P_{n|n-1}$ from (6.14), where Φ_{n-1} is given by (8.8). Due to its symmetry, positive-definite matrix $P_{n|n-1}$ is chosen in the following form:

$$P_{n|n-1} = \begin{bmatrix} P_{11} & P_{21} & P_{31} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{23} & P_{33} \end{bmatrix}. \tag{8.15}$$

Step 5: Compute the Kalman gain (6.17) for the reduced-order filter by using (8.7) and (8.15) as,

$$K_n = \begin{bmatrix} P_{11} \\ P_{21} \\ P_{31} \end{bmatrix} (\begin{bmatrix} P_{11} \end{bmatrix} + R_d)^{-1} = \begin{bmatrix} P_{11} \\ P_{21} \\ P_{31} \end{bmatrix} \begin{bmatrix} T_{11} \end{bmatrix}^{-1} \triangleq \begin{bmatrix} k_{11} \\ k_{21} \\ k_{31} \end{bmatrix}. \tag{8.16}$$

Step 6: Update the predicted state variables from (6.15) for the following cases:

(i) n is odd:

$$\hat{\hat{i}}_\alpha(n) = \hat{\hat{i}}_\alpha(n|n-1) + k_{11}\tilde{\tilde{i}}_\alpha(n), \tag{8.17}$$

$$\hat{e}_\alpha(n) = \hat{e}_\alpha(n|n-1) + k_{21}\tilde{\tilde{i}}_\alpha(n), \tag{8.18}$$

$$\hat{e}_\beta(n) = \hat{e}_\beta(n|n-1) + k_{31}\tilde{\tilde{i}}_\alpha(n), \tag{8.19}$$

(ii) n is even:

$$\hat{\hat{i}}_\beta(n) = \hat{\hat{i}}_\beta(n|n-1) + k_{11}\tilde{\tilde{i}}_\beta(n), \tag{8.20}$$

$$\hat{e}_\beta(n) = \hat{e}_\beta(n|n-1) + k_{21}\tilde{\tilde{i}}_\beta(n), \tag{8.21}$$

$$\hat{e}_\alpha(n) = \hat{e}_\alpha(n|n-1) + k_{31}\tilde{\tilde{i}}_\beta(n), \tag{8.22}$$

where

$$\tilde{\tilde{i}}_\alpha(n) = i_\alpha(n) - \hat{\hat{i}}_\alpha(n|n-1), \tag{8.23}$$

$$\tilde{\tilde{i}}_\beta(n) = i_\beta(n) - \hat{\hat{i}}_\beta(n|n-1), \tag{8.24}$$

in which k_{ij} are elements of the Kalman gain K_n.

Step 7: Update the present covariance matrix P_n from (6.16).

Step 8: Compute the rotor angular speed and rotor flux position from (8.9) – (8.10). Then, set $n = n + 1$, and go back to *Step 2*.

8.3 FPGA Control Architecture and FSM Implementation

8.3.1 FPGA Architecture for Sensorless PMSM Speed Control

Figure 8.1 shows the FPGA-based architecture of the proposed speed control system for the sensorless PMSM drive. The control system includes a PI-speed controller,

a speed command read-in unit, a CCCT module, a SVPWM generator, a quadrature encoder pulse (QEP) detection and transformation module, an analog-to-digital converter (ADC) read-in and conversion, a frequency divider as well as a parallel reduced-order EKF-based rotor flux angle and rotor speed estimation module. All modules are described by the VHDL and simulated in ModelSim to test their feasibility before implementation in Altera Cyclone II EP2C70. The sampling frequency for the current and speed controllers is designated at 16 kHz and 2 kHz, respectively. The operating clock rate of the designed FPGA controller is 50 MHz and the frequency divider generates 50 MHz (*Clk*), 12.5 MHz (*Clk-step*), 16 kHz (*Clk-cur*) and 2 kHz (*Clk-sp*) clock to supply to all module circuits of the application system-on-chip.

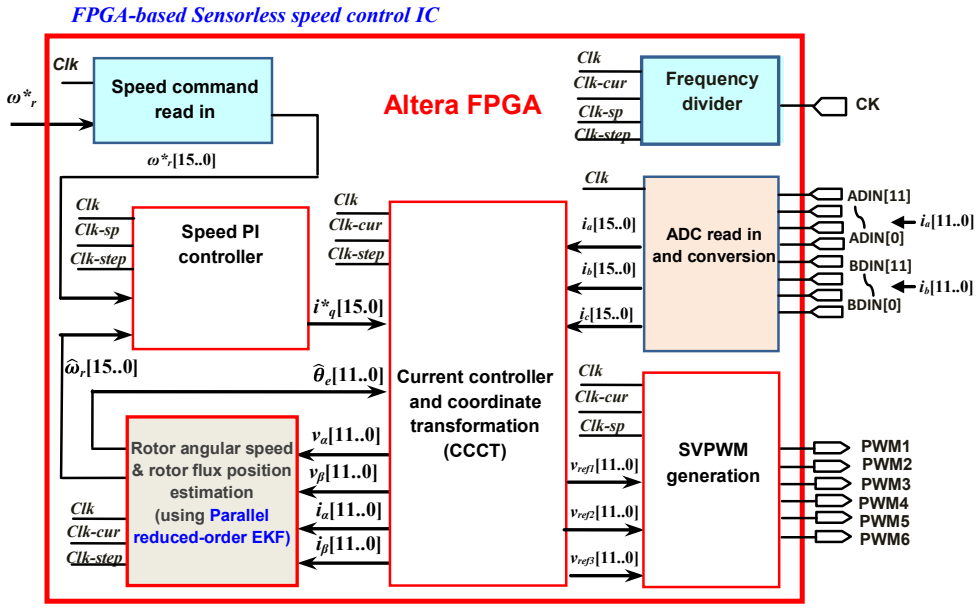


Figure 8.1 : Proposed EKF based sensorless speed control IC circuitry

8.3.2 EKF Complexity and Implementation with FSM

The computational cost for EKF realization increases drastically with the algorithm complexity. In our approach, this has been significantly reduced thanks to a decrease in the system order by using two parallel EKFs. The complexity, in terms of the number of arithmetic operations, for the full-order EKF (p=4) and proposed reduced-order EKF (p=3) are evaluated in Table 8.1.

Table 8.1 : Complexity of EKF Algorithms

Operations	Full-order EKF(p=4)	Reduced-order EKF(p=3)
Multiplications	94	32
Additions	67	21
Subtractions	31	8
Divisions	7	4

For implementation, a sequential FSM is employed to describe the parallel reduced-order EKF algorithm, as shown in Figure 8.2, where the data type adopts a 16-bit length with Q15 format and 2's complement operations. Although the reduced-order EKF algorithm described is highly complex, the FSM adequately incorporates the control structure and can be described by VHDL. Here, the divider, multiplier and adder apply the Altera LPM standard. It manipulates 75 steps machine to carry out the overall computations.

In our FSM, the steps $S_0 - S_1$ select one of the two EKFs; steps $S_2 - S_7$ execute the computation of the Jacobian matrix and predicted state variables; steps $S_8 - S_{31}$ are for the calculation of the temporary covariance matrix; steps $S_{32} - S_{35}$ describe the computation of the state error and the Kalman gain K_n; steps $S_{36} - S_{39}$ perform the present state updating; steps $S_{40} - S_{45}$ update the present covariance matrix P_n;

steps $S_{46} - S_{59}$ compute the rotor flux position; steps $S_{40} - S_{45}$ compute the rotor speed; and finally step S_{75} is for getting back to step 1 for a new iteration loop. Since the execution of each step can be completed within 80 ns corresponding to frequency 12.5 MHz in FPGA, a total of N_{EKF}=75 steps needs 6 μs for the parallel reduced-order EKF operation. Similarly, the numbers of steps for execution of CCCT and SVPWM are respectively N_{CCCT}=24 and N_{SVPWM}=13 steps. The total resource usage in FPGA are 5,996 LEs; 5 hardwired (hw) 18-bit multipliers and 73,728 RAM bits resource for the Cyclone II EP2C70, as summarized in Table 8.2.

Table 8.2 : FPGA Utility Evaluation for Sensorless PMSM Speed Control

Module circuits	LEs	Memory (bits)	hw 18-bit multipliers
1. CCCT	864	24,576	1
2. SVPWM generation	1,221	0	1
3. ADC read-in and conversion	136	0	1
4. Speed PI controller	254	0	1
5. Parallel reduced-order EKF	3,521	49,152	1
TOTAL	**5,996**	**73,728**	**5**

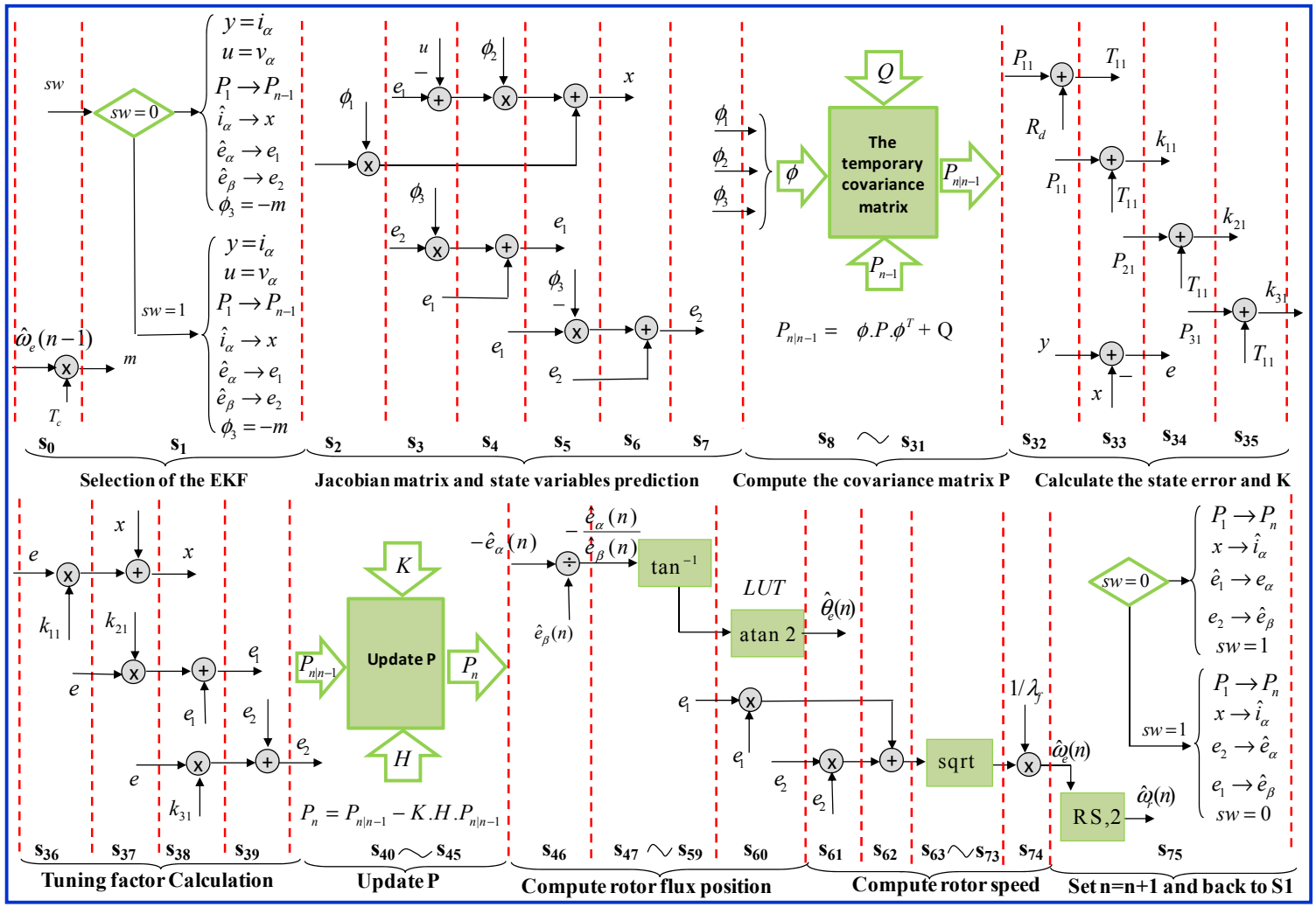


Figure 8.2 : State machine diagram of the parallel reduced-order EKF algorithm

8.3.3 Sensorless Control Timing Diagram

The designed sampling interval is T_c=62.5 μs (16 kHz), the same as the PWM switching period. Since all processes are implemented by using FSM, the execution time t_{ex} is calculated by

$$\begin{aligned} t_{ex} &= t_{EKF} + t_{CCCT} + t_{SVPWM} \\ &= (N_{EKF} + N_{CCCT} + N_{SVPWM}).T_{Clk-step}, \end{aligned} \tag{8.25}$$

where $T_{Clk-step}$, t_{EKF}, t_{CCCT} and t_{SVPWM} are respectively the clock step period, execution time for reduced-order extended Kalman filtering, current control and coordinate transformation, and space vector pulse width modulation. The ADC conversion and the divider, multiplier, adder all use 50 MHz clock (*Clk*) and 12.5 MHz (*Clk-step*) is used for step implementation. Figure 8.3 shows the timing diagram of all blocks in the FPGA-based sensorless control for PMSM using the proposed reduced-order EKFs.

At the beginning of the n^{th} sampling period nT_c, the ADC read-in and conversion block reads and converts the values of the stator currents from two external ADCs (AD574AJN). This block is implemented by 80 steps and at 50 MHz clock, i.e., 1.6 μs operation time. At the same time the EKF estimation is activated and generates the estimated position angle θ_e to CCCT and the estimated rotor speed to the speed control loop. The EKF estimation block is executed in 75 steps with 6 μs in total. The CCCT block including the circuits of $d-, q-$axis PI control, transformation, and cosine/sine look-up table operates sequentially with 24 steps at 12.5 MHz clock, resulting in an execution time of 1.92 μs. Its outputs are three-phase reference voltages (v_{ref1}, v_{ref2}, v_{ref3}), which are input to the SVPWM block, implemented by a state vector machine with a computation time of 1.04 μs for 13 steps.

The total execution time is t_{ex}=8.96 μs. Although the FSM method needs more operation time, it consumes less FPGA resources than the parallel processing method.

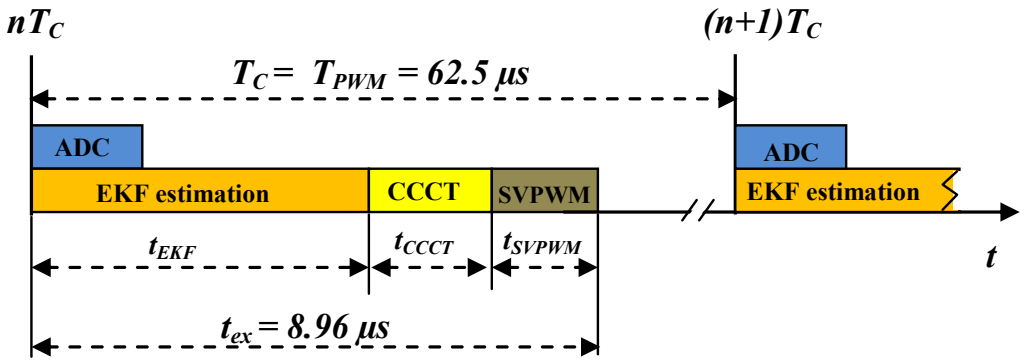


Figure 8.3 : Timing diagram of the sensorless control system

8.4 Computer Simulation

8.4.1 ModelSim/Simulink Co-simulation

Simulation is performed by using the Electronic Design Automation (EDA) Simulator Link, which provides a verification interface between MATLAB/Simulink and the VHDL simulator on an FPGA board.

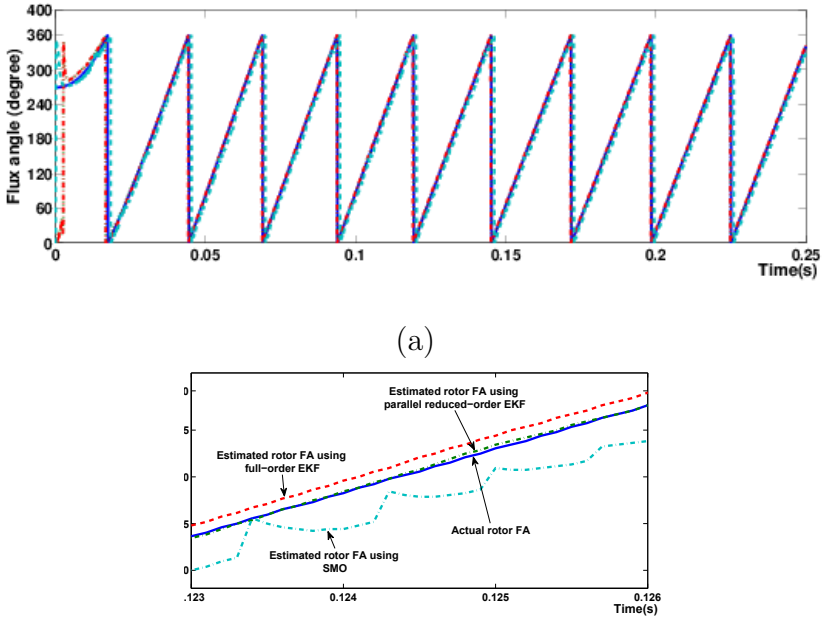
Five modules are created for a PI speed controller, CCCT and SVPWM units, the rotor flux position estimator using either the proposed parallel reduced-order EKFs, full-order EKF or SMO, respectively. To be consistent with the hardware implementation, the sampling frequency of current and speed control is selected respectively as 16 kHz and 2 kHz. The clocks of 50 MHz, 12.5 MHz and 16 kHz supply to all other modules of ModelSim in accordance with the architecture of Figure 8.1. The

parameters of an 8-pole PMSM used in this simulation are the same as in the previous chapters.

8.4.2 Estimation and Speed Control: Simulation Results

To show the effectiveness of the proposed parallel reduced-order EKF technique for sensorless PMSM control, we compare it with cases using a full-order EKF and SMO with the same algorithm and estimator parameters as reported respectively in [34] and [70]. The simulation results are shown in Figures 8.4-8.6. In Figure 8.4(a), the actual rotor flux angle and its estimates are shown for a full-order EKF, SMO and the proposed parallel reduced-order EKFs, typically at 900 rpm. Their zoom-in responses between 0.123 and 0.126 (s) are depicted in Figure 8.4(b) to show the accuracy of the three estimation techniques. The estimated flux angle has a 3^o phase error, about 150 μs delay time under reference speed 900 rpm by using the parallel reduced-order EKF method.

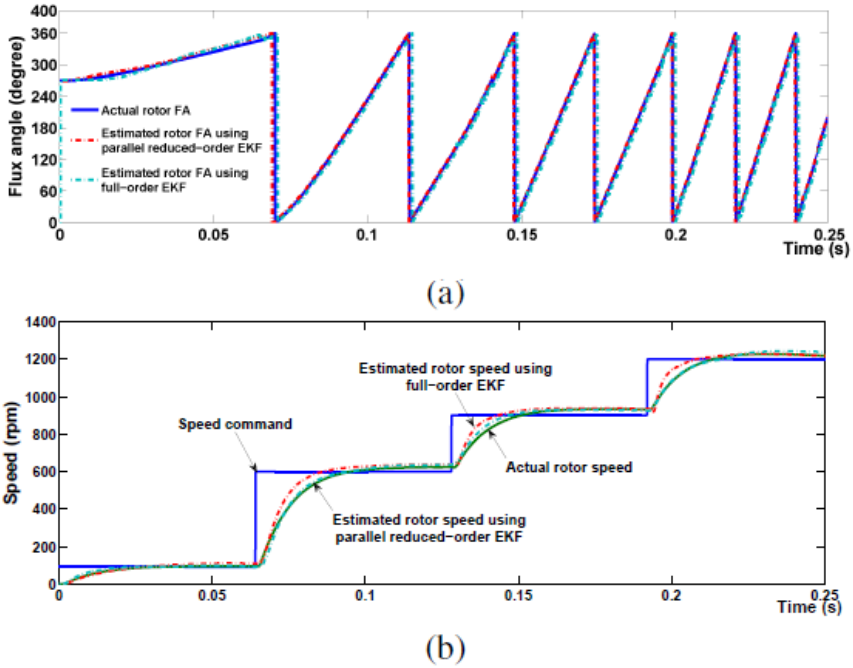
We then consider the case when the motor speed command is varied with a staircase pattern 90→600→900→1200 rpm using the full-order and parallel reduced-order EKFs. Figures 8.5(a) and (b) show respectively the actual and estimated rotor flux angle, and the actual and estimated speed responses for both EKF techniques. A slight improvement in the estimation performance with a rising time of 20 ms and steady-state error near 0 rpm is accounted for by the easy tuning of the filter parameters with a lower order. Responses of the currents are presented in Figure 8.6(a) for the $d-q$ axes and in Figure 8.6(b) for three phases when using the parallel reduced-order EKFs. The current responses are consistent with the staircase speed patterns applied to the sensorless PMSM.



(a)

(b)

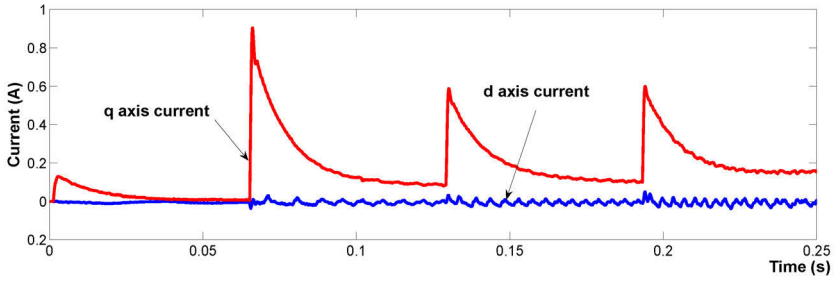
Figure 8.4 : Speed 900 *rpm*: (a) actual and estimated rotor flux angle from SMO, full-order EKF and parallel reduced-order EKFs, (b) zoom-in responses



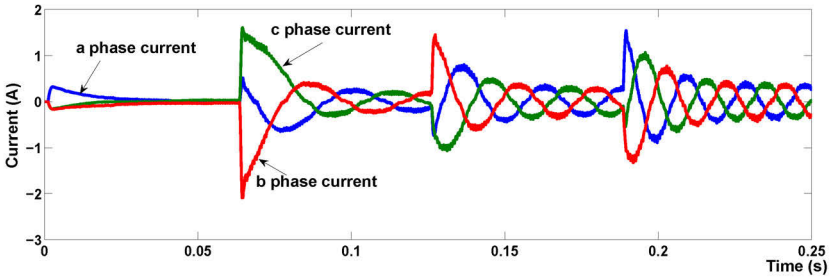
(a)

(b)

Figure 8.5 : Speed pattern 0→90→600→900→1200 *rpm*: (a) actual and estimated rotor flux angle, (b) actual and estimated speed



(a)



(b)

Figure 8.6 : Responses with reduced-order EKFs: (a) $d-q$ currents, (b) three-phase currents

8.4.3 Resource Consumption and Execution Time Analysis

Table 8.3 summarizes the overall hardware resource consumption and execution time for speed control of sensorless PMSM drives using the low cost Altera Cyclone II FPGA (EP2C70F896C6) with three different types of sensorless techniques mentioned above in comparison with another low cost FPGA, the Xilinx Spartan 3E (XC3S1600E), reported in [34]. Hardware resource usage and execution time for the system with an EKF implementation are larger than for a SMO as it requires more matrix computations. However, simulation results show that EKFs yield better performance than SMO. It is clear from Figure 8.4(b) that SMO has a larger error (9^o) and slower response (400 μs) compared to the full-order EKF (5^o, 250 μs) and our proposed method (2^o, 150 μs). Compared with an FPGA-based standard EKF im-

plementation using parallel processing systolic arrays, which are ideal to implement on FPGAs, the systolic array, albeit having a fast execution time, requires much FPGA resources and its parallel architecture is quite complicated. The results given in [34], using Xilinx Spartan 3E (XC3S1600E) with a 44 MHz clock frequency and 22-bit format in the standard EKF, showed good control performance with 6.82 μs for execution thanks to the parallel design methodology, but at a high expense of the computational resource with 7,376 LEs; 18,000 RAM bits and 36 hardwired 18-bit multipliers. The proposed method has thus achieved a very good compromise between the hardware resource consumption (5,996 LEs; 49,152 bits; 5 hw 18-bit multipliers), execution time 8.96 μs (with a 12.5 MHz clock frequency) and overall control performance of the system.

8.5 Implementation and Results

8.5.1 Laboratory Set-Up

The on-chip control system and IGBT inverter voltage source supplied to the sensorless PMSM drive is depicted in a photograph shown in Figure 8.7.

- *Motor*: The PMSM's name-plate specifications are 100 W, 4 $poles$, 100 V, 1.7 A, and 3000 rpm. The brake torque of the motor is 0.32 Nm. It uses an optical encoder of 1000 $pulses/rev$.

- *Supply*: The inverter has 6 sets of power transistors of the IGBT type. The collector-emitter voltage of the IGBT is rating 500 V, the gate-emitter voltage is rating ± 20 V, and the collector current is rating 20 A in DC. Input signals of the inverter are PWM signals from the FPGA.

- *On-chip Control System*: The FPGA chip Cyclone II (EP2C70F896C6), pro-

duced by Altera, has 68,416 LEs, 1,152,000 bits of on-chip RAM, 150 hardwired 18-bit embedded multipliers, 36 DSP blocks and maximum user I/O pins of 622.

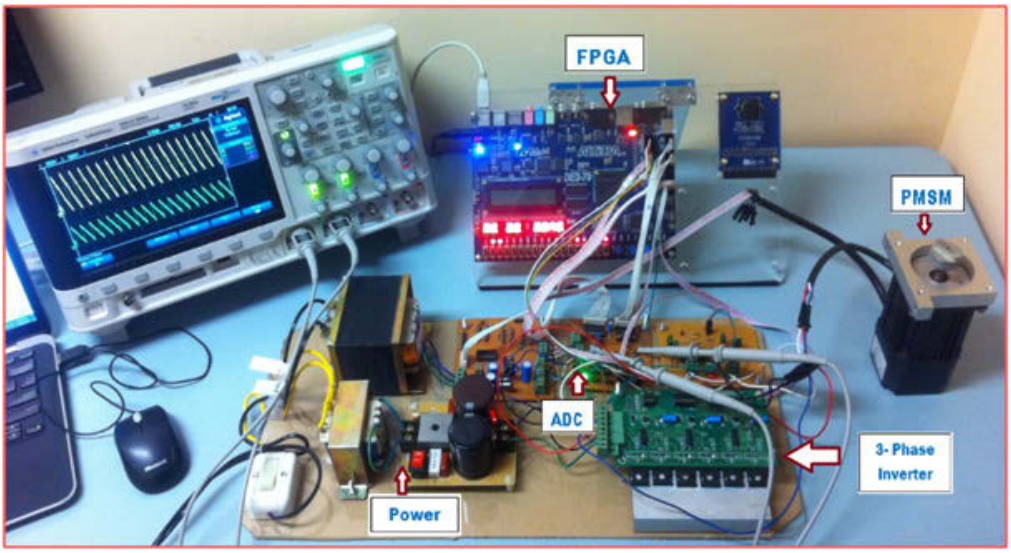


Figure 8.7 : Set-up photograph

8.5.2 Experimental Results

For implementation, the PWM frequency, inverter dead-band interval, current loop sampling frequency, and speed loop sampling frequency are designated respectively at 16 kHz, 1 μs, 16 kHz and 2 kHz. In the proposed sensorless control IC, the modules for current control, speed control, current vector control scheme, SVPWM generation, ADC read-in and conversion, coordinate transformation and parallel reduced-order EKFs for rotor flux angle and rotor speed estimation are all realized by hardware in the FPGA chip. Again, the PMSM is experimentally tested at various speed commands. For verification, the encoder attached to the PMSM is used to obtain the actual rotor flux position. As shown in Figure 8.8, at 200 rpm the estimated flux

angle exhibited around 4° phase error as compared with the measurements from the encoder sensor. When the PMSM reversed from 1000 to -1200 rpm, the estimated flux angle almost matched the actual value obtained from the encoder, as shown in Figure 8.9. Experiment results indicate that the proposed parallel reduced-order EKFs for estimating the flux angle is suitable for a wide speed range of the motor, including a low speed, a moderate or a high speed motor running in both forward and reverse directions. At a low speed of the motor, the weak back emf signal tends to affect the accuracy of the flux position but it is not considerable. At a speed command of 1000 rpm, the PMSM was loaded from a 2 Nm external load, the step response using the proposed estimation method is shown in Figure 8.10, where the rising time and steady-state error value are observed as 250 ms and near 0 rpm. Experimental results show that all the rotor speed tracks the electrical reference speed in different operation conditions of the PMSM. These together have confirmed the effectiveness and correctness of the proposed FPGA-based control architecture for sensorless PMSM drives.

8.6 Chapter Conclusion

In this chapter, we have presented a fully FPGA-based speed control system for a sensorless permanent magnet synchronous motor by using the parallel reduced-order EKFs for estimation and the finite state machine method for algorithm implementation. The proposed system comprises a PI speed controller, field-oriented PI current controllers, a rotor flux angle and rotor speed estimator together with other modules for command read-in, frequency divider, ADC conversion and SVPWM generation. The whole system architecture has been successfully realized in one FPGA chip with a small resource usage as well as a fast execution time. The proposed flux angle

and speed estimator algorithms have proven their effectiveness in the estimation of the rotor speed as verified by the co-simulation method and in extensive experiments. Simulation results have shown better performance when compared with other schemes for FPGA-based sensorless speed control in PMSM drives. Real-time responses obtained from a laboratorial set-up with a Cyclone II FPGA platform are included for performance validation of the control-system-on-programmable chip design.

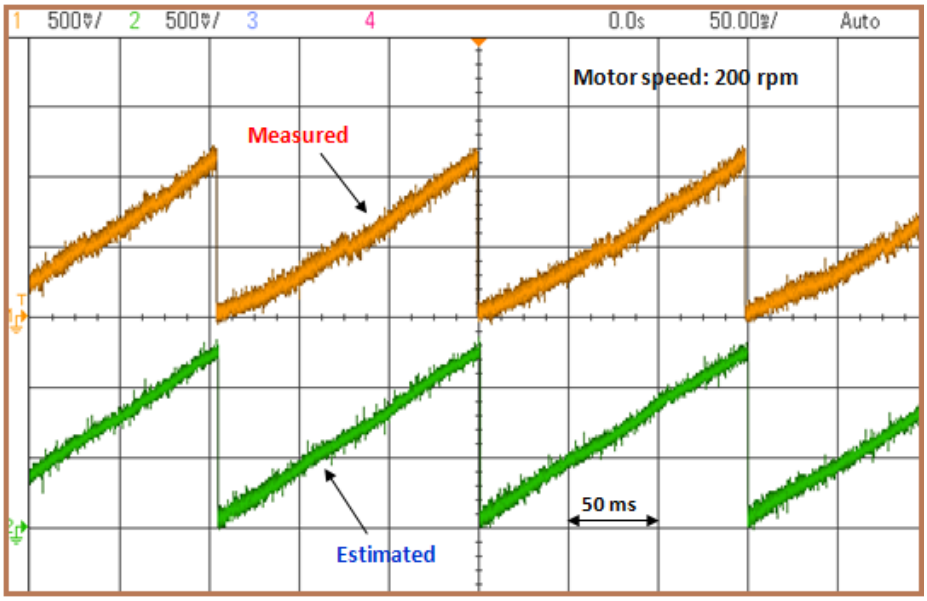


Figure 8.8 : Estimated and measured flux angle when PMSM runs at 200 *rpm*

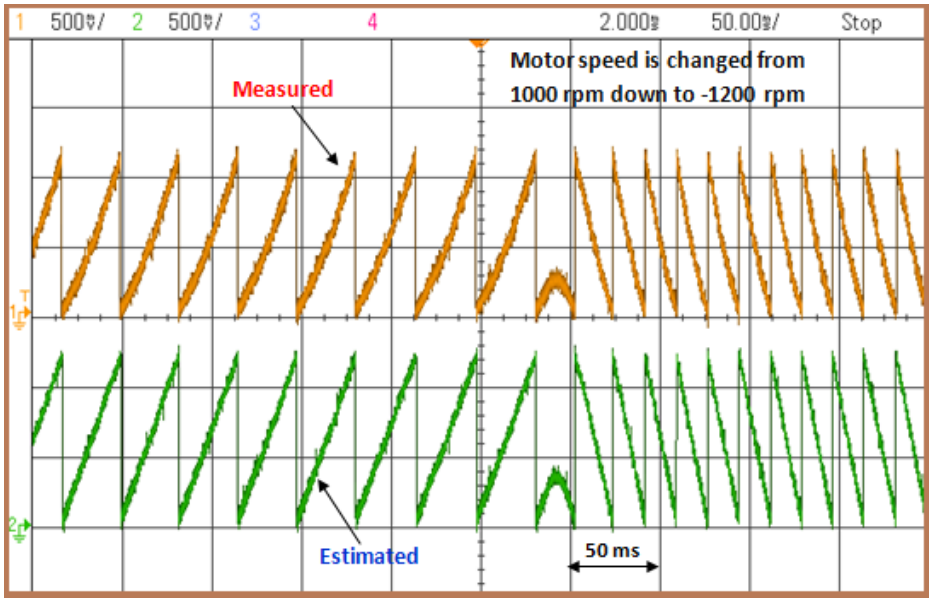


Figure 8.9 : Estimated and measured flux angle with PMSM running from 1000 to -1200 *rpm*

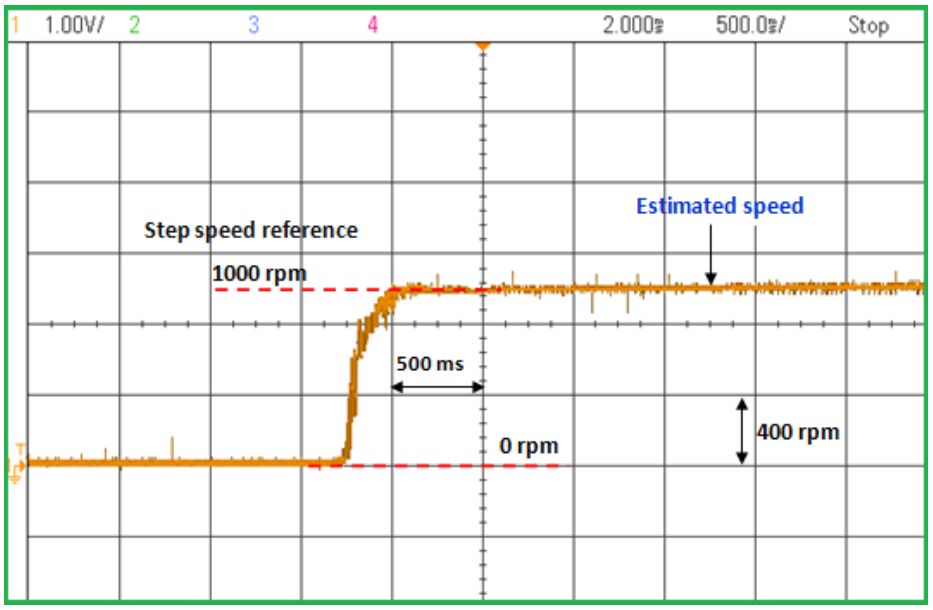


Figure 8.10 : Speed step response of the sensorless PMSM drive

Chapter 9

General Conclusion

9.1 Conclusions

In all the thesis topics, the analysis and design are mostly based on a detailed analytical effects, design procedure, and control performance enhancement along with a sequential FSM design method followed by computer simulation and experimental validation. The ModelSim/Simulink is used for extensive simulation and experimental validation is conducted using an Altera kit and laboratory instrumentation.

To accomplish the thesis objectives, the integration of a multi-loop PI and neural fuzzy control system for multiple-axis motion positioning and tracking via the use of the FPGA technology is firstly presented in Chapter 4. The controlled plant here is an X-Y table driven by permanent magnet linear synchronous motors. The control system comprises two programmable servo control systems for both axes; each includes a motion planner, a PI speed controller in the inner loop and a NFC in the position loop. Here, to increase the tracking performance in dealing with unmodelled dynamics and cross-axis interferences, the NFC is designed by using a radial basis function neural network in combination with a parameter adjusting mechanism.

High performance estimation schemes for sensorless PMSM control based on the FPGA technology, are demonstrated in Chapter 5, Chapter 6, Chapter 7 and Chapter 8, respectively. In Chapter 5, an observer-based integral sliding mode controller is presented. By integrating the observer-based and integral sliding mode control

techniques into speed control of a PMSM drive, the system performance can be substantially enhanced while improving its cost-effectiveness and reliability.

The improvements of EKF algorithm for the sensorless PMSM have been developed in Chapters 6-8. In conventional Kalman filtering, abrupt state changes may not be tracked adequately since sudden variations may seriously affect the auto-correlation Gaussian property of white noise in the filter residuals. For this, the AF-EKF has been developed in Chapter 6 to recover the estimation results in events of frequent and sharp state jumps. The AF-EKF is, therefore, a promising estimator for PMSM drives that are subject to frequently-varying loads speed commands.

Chapter 7 presents the design and implementation of an adaptive extended Kalman filter. This improved EKF versions can be obtained by incorporating an adjustment mechanism of the noise covariances into the filter to tune online the process noise covariance to improve the filtering performance. Therefore, it is also a good solution for sensorless PMSM drives with more accurate estimation features, provided it is feasible in implementation.

In Chapter 8, for reduction of computation resources as well as accuracy improvement in the rotor position estimation, a parallel reduced-order EKF is proposed. Compared with an EKF, the system order is reduced and the iteration process is greatly simplified, resulting in significant savings of resource utility, while maintaining high estimation performance.

In conclusion, the significance of this thesis contribution includes providing a feasible and effective solution for the implementation of complex control strategies to fully exploit the FPGA advantages in power electronics and drive applications.

9.2 Thesis Contributions

In the field of power electronics and drive applications, this research aims to contribute a proof-of-concept for the control-system-on-chip and a prototype for a fully-implemented FPGA control architecture for PMSM drives. Based on a sequential finite state machine design method, this work aims to optimally design the controllers-based FPGA. In the following, thesis contributions are elaborated,

i. A design and validation methodology for FPGA-based digital controller is proposed throughout this thesis based on analytical effects, design procedure, and control performance enhancement along with a sequential FSM design method for PMSM drives under sensor/sensorless vector control using a number of control techniques. The proposed method is first tested by ModelSim/Simulink co-simulation method as well as Altera Cyclone boards and then validated by experimental systems.

ii. An FPGA-based intelligent control and robust cascade control for single axis and multiple axis tracking with PMSMs have been implemented in this study. It is based on the neural fuzzy controller in the position loop which increases the tracking performance in dealing with unmodelled dynamics and cross-axis interferences of multi-axis control systems.

iii. An important contribution of this thesis rests with a convincing demonstration of high performance estimation schemes, using sliding mode observers and extended Kalman filters, in terms of accuracy and robustness against noisy and/or perturbed currents for sensorless PMSM control based on the FPGA technology.

In general, this thesis aims to provide a feasible and effective solution for the implementation of complex control strategies to fully exploit the FPGA advantages in power electronics and drive applications.

9.3 Future Works

Future work in FPGA control architecture for PMSM drives can be explored to complement the findings of this thesis. The tasks suggested for it are listed below:

(1) In the development for neural fuzzy control, the parameters of a fuzzy rule table are tuned to adequate values; however, most adaptive fuzzy control approaches have quite often a fixed membership function. As such, fuzzy membership functions can also be tuned based on a RBF NN identification technique, or a suitable fuzzy membership function can adopt the Gauss function. These proposed algorithms can be applied for servo control of an X-Y or X-Y-Z table.

(2) In the sensorless PMSM estimation control, the chattering is still a challenging problem in SMC. Therefore, the performance of an FPGA based motor drive system using a combination of high order sliding mode observer and control is worth being investigated. Regarding filtering technique, EKF suffers from being sensitive to uncertainties of PMSM parameters. So the combination of EKF and online estimation of motor parameters (such as resistor, inductor, flux linkages) is an open topic. Additionally, a smooth transition scheme between adaptive algorithms and reduced order models can be further developed into new EKFs. Further, Kalman filtering is normally based on the assumption that the uncertainties of the PMSM system follow a Gaussian distribution; however, it may not be the case in practice. Particle filter, a new approach with unity distribution assumption of uncertainties distribution has become a hot research topic in this context for process variables estimation.

(3) In the realization aspect, the comparison of hardware/software co-design method and fully hardware design method will be considered. The proposed FPGA design methodology with developed algorithms can further be investigated for other control and automation applications, or various robotic and mechatronic systems.

Bibliography

[1] A. Accetta, M. Cirrincione, M. Pucci, and G. Vitale, "Sensorless control of pmsm fractional horsepower drives by signal injection and neural adaptive-band filtering," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 3, pp. 1355–1366, March 2012.

[2] Actel, *Actel on-line documentation*. Available in :www.actel.com.

[3] Altera, *Altera on-line documentation*. Available in :www. altera.com.

[4] O. Barambones, A. Garrido, and F. Maseda, "Integral sliding-mode controller for induction motor based on field-oriented control theory," *Control Theory Applications, IET*, vol. 1, no. 3, pp. 786–794, May 2007.

[5] V. A. Bavdekar, R. B. Gopaluni, and S. L. Shah, "Evaluation of adaptive extended kalman filter algorithms for state estimation in presence of model-plant mismatch," in *Dynamics and Control of Process Systems, The 10th IFAC International Symposium on*, Dec 2013, pp. 184–189.

[6] S. Bolognani, S. Calligaro, R. Petrella, and M. Tursini, "Sensorless control of ipm motors in the low-speed range and at standstill by hf injection and dft processing," *Industry Applications, IEEE Transactions on*, vol. 47, no. 1, pp. 96–104, Jan 2011.

[7] S. Bolognani, L. Tubiana, and M. Zigliotto, "Ekf-based sensorless ipm syn-

chronous motor drive for flux-weakening applications," in *Industry Applications Conference, 2002. 37th IAS Annual Meeting. Conference Record of the*, vol. 1, Oct 2002, pp. 112–119 vol.1.

[8] S. Bolognani, L. Tubiana, and M. Zigliotto, "Extended kalman filter tuning in sensorless pmsm drives," *Industry Applications, IEEE Transactions on*, vol. 39, no. 6, pp. 1741–1747, Nov 2003.

[9] Z. Chen, M. Tomita, S. Doki, and S. Okuma, "An extended electromotive force model for sensorless control of interior permanent-magnet synchronous motors," *Industrial Electronics, IEEE Transactions on*, vol. 50, no. 2, pp. 288–295, Apr 2003.

[10] S. Chi, Z. Zhang, and L. Xu, "Sliding-mode sensorless control of direct-drive pm synchronous motors for washing machine applications," *Industry Applications, IEEE Transactions on*, vol. 45, no. 2, pp. 582–590, March 2009.

[11] C. H. Choi and J. K. Seok, "Pulsating signal injection-based axis switching sensorless control of surface-mounted permanent-magnet motors for minimal zero-current clamping effects," *Industry Applications, IEEE Transactions on*, vol. 44, no. 6, pp. 1741–1748, Nov 2008.

[12] W. Chongwu, H. Yuyao, and L. Hong, "The study on the pmsm sensorless control using the sub-optimal fading extend kalman filter," in *Power Electronics and Drive Systems (PEDS), 2013 IEEE 10th International Conference on*, April 2013, pp. 294–297.

[13] V. Colli, R. Di Stefano, and F. Marignetti, "A system-on-chip sensorless control for a permanent-magnet synchronous motor," *Industrial Electronics, IEEE*

Transactions on, vol. 57, no. 11, pp. 3822–3829, Nov 2010.

[14] F. Dezza, G. Foglia, M. Iacchetti, and R. Perini, "An mras observer for sensorless dfim drives with direct estimation of the torque and flux rotor current components," *Power Electronics, IEEE Transactions on*, vol. 27, no. 5, pp. 2576–2584, May 2012.

[15] W. Ding, J. Wang, and C. Rizos, "Improving adaptive kalman estimation in gps/ins integration," *The Journal of Navigation*, vol. 60, no. 8, pp. 517 – 529, 2007.

[16] Y. Driss and D. Yousfi, "Pmsm sensorless control using back-emf based position and speed estimation method," in *Renewable and Sustainable Energy Conference (IRSEC), 2013 International*, March 2013, pp. 436–440.

[17] M. Eskola, "Speed and position sensorless control of pmsms in matrix converter and voltage source converter applications," in *PhD Thesis, Tampere University of Technology*, Finland, 2006.

[18] M. Ezzat, A. Glumineau, and F. Plestan, "Sensorless high order sliding mode control of permanent magnet synchronous motor," in *Variable Structure Systems (VSS), 2010 11th International Workshop on*, June 2010, pp. 233–238.

[19] S. L. Fagin, "Recursive linear regression theory, optimal filter theory, and error analysis optimal system," in *IEEE Int. Convention Record*, Nov 1964, pp. 216–240.

[20] G. Foo and M. Rahman, "Sensorless sliding-mode mtpa control of an ipm synchronous motor drive using a sliding-mode observer and hf signal injection,"

Industrial Electronics, IEEE Transactions on, vol. 57, no. 4, pp. 1270–1278, April 2010.

[21] S. Gadoue, D. Giaouris, and J. Finch, "Sensorless control of induction motor drives at very low and zero speeds using neural network flux observers," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 8, pp. 3029–3039, Aug 2009.

[22] F. Genduso, R. Miceli, C. Rando, and G. Galluzzo, "Back emf sensorless-control algorithm for high-dynamic performance pmsm," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 6, pp. 2092–2100, June 2010.

[23] Y. Geng and J. Wang, "Adaptive estimation of multiple fading factors in kalman filter for navigation applications," *GPS Solutions*, vol. 12, no. 1, pp. 273–279, Jan 2008.

[24] T. Goto, Y. Sato, S. Kondo, M. Tomita, M. Hasegawa, S. Doki, and S. Kato, "Position and velocity sensorless control of ipmsm using full-order observer based on extended electromotive force with a new observer design method," in *Electrical Machines (ICEM), 2014 International Conference on*, Sept 2014, pp. 864–870.

[25] Q. P. Ha, Y. H. Yu, and N. K. Quang, "Fpga-based cooperative control of indoor multiple robots," *International Journal of Advanced Mechatronic Systems*, vol. 61, no. 12, pp. 6574–6582, Dec 2014.

[26] D. Haifeng, W. Xuezhe, and S. Zechang, "State and parameter estimation of a hev li-ion battery pack using adaptive kalman filter with a new soc-ocv concept,"

in *Measuring Technology and Mechatronics Automation, 2009. ICMTMA '09. International Conference on*, vol. 2, April 2009, pp. 375–380.

[27] M. Hasegawa, S. Yoshioka, and K. Matsui, "Position sensorless control of interior permanent magnet synchronous motors using unknown input observer for high-speed drives," *Industry Applications, IEEE Transactions on*, vol. 45, no. 3, pp. 938–946, May 2009.

[28] H. He, R. Xiong, X. Zhang, F. Sun, and J. Fan, "State-of-charge estimation of the lithium-ion battery using an adaptive extended kalman filter based on an improved thevenin model," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 4, pp. 1461–1469, May 2011.

[29] C. Hide, T. Moore, and M. Smith, "Adaptive kalman filtering for low-cost ins/gps," *Journal of Navigation*, vol. 56, pp. 143 – 152, Jan 2003.

[30] M. Hilairet, F. Auger, and C. Darengosse, "Two efficient kalman filters for flux and velocity estimation of induction motors," in *Power Electronics Specialists Conference, 2000. PESC 00. 2000 IEEE 31st Annual*, vol. 2, 2000, pp. 891–896.

[31] M. Hilairet, F. Auger, and E. Berthelot, "Speed and rotor flux estimation of induction machines using a two-stage extended kalman filter," *Automatica*, vol. 45, no. 8, pp. 1819 – 1827, 2009.

[32] S. C. Hsu, C. H. Liu, and N. J. Wang, "Fuzzy pi controller tuning for a linear permanent magnet synchronous motor drive," in *Industrial Electronics Society, IECON '01. The 27th Annual Conference of the IEEE*, vol. 3, 2001, pp. 1661–1666.

[33] L. Idkhajine and E. Monmasson, "Design methodology for complex fpga-based controllers - application to an ekf sensorless ac drive," in *Electrical Machines (ICEM), 2010 XIX International Conference on*, Sept 2010, pp. 1–6.

[34] L. Idkhajine, E. Monmasson, and A. Maalouf, "Fully fpga-based sensorless control for synchronous ac drive using an extended kalman filter," *Industrial Electronics, IEEE Transactions on*, vol. 59, no. 10, pp. 3908–3918, Oct 2012.

[35] L. Idkhajine, E. Monmasson, M. W. Naouar, A. Prata, and K. Bouallaga, "Fully integrated fpga-based controller for synchronous motor drive," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 10, pp. 4006–4017, Oct 2009.

[36] J. S. Jang, B. G. Park, T. S. Kim, D. M. Lee, and D. seok Hyun, "Parallel reduced-order extended kalman filter for pmsm sensorless drives," in *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, Nov 2008, pp. 1326–1331.

[37] S. Jung and S. S. Kim, "Hardware implementation of a real-time neural network controller with a dsp and an fpga for nonlinear systems," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 1, pp. 265–271, Feb 2007.

[38] H. Kim, J. Son, and J. Lee, "A high-speed sliding-mode observer for the sensorless speed control of a pmsm," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 9, pp. 4069–4077, Sept 2011.

[39] K. H. Kim, G. I. Jee, and J. G. Lee, "The stability analysis of the adaptive fading extended kalman filter using the innovation covariance," in *Int. J. Control, Automation, and Systems*, vol. 7, 2009, pp. 49–56.

[40] Y. H. Kim and Y. S. Kook, "High performance ipmsm drives without rotational position sensors using reduced-order ekf," *Energy Conversion, IEEE Transactions on*, vol. 14, no. 4, pp. 868–873, Dec 1999.

[41] Y. S. Kung, C. C. Huang, and M. H. Tsai, "Fpga realization of an adaptive fuzzy controller for pmlsm drive," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 8, pp. 2923–2932, Aug 2009.

[42] Y. S. Kung, N. K. Quang, and L. T. V. Anh, "Fpga-based neural fuzzy controller design for pmlsm drive," in *Power Electronics and Drive Systems, 2009. PEDS 2009. International Conference on*, Nov 2009, pp. 222–227.

[43] R. Leidhold, "Position sensorless control of pm synchronous motors based on zero-sequence carrier injection," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 12, pp. 5371–5379, Dec 2011.

[44] Y. Li, J. Huo, X. Li, J. Wen, Y. Wang, and B. Shan, "An open-loop sin microstepping driver based on fpga and the co-simulation of modelsim and simulink," in *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on*, vol. 6, Aug 2010, pp. 223–227.

[45] C. Liaw, R. Shue, H. Chen, and S. Chen, "Development of a linear brushless dc motor drive with robust position control," *Electric Power Applications, IEE Proceedings* -, vol. 148, no. 2, pp. 111–118, Mar 2001.

[46] W. Limei, Z. Tao, and W. Zhitao, "Pmlsm controller design based on self-constructing feedback fuzzy neural network," in *Control and Decision Conference, 2009. CCDC '09. Chinese*, June 2009, pp. 3146–3150.

[47] F. J. Lin, C. K. Chang, and P. K. Huang, "Fpga-based adaptive backstepping sliding-mode control for linear induction motor drive," *Power Electronics, IEEE Transactions on*, vol. 22, no. 4, pp. 1222–1231, July 2007.

[48] F. J. Lin and P. H. Shen, "Robust fuzzy neural network sliding-mode control for two-axis motion control system," *Industrial Electronics, IEEE Transactions on*, vol. 53, no. 4, pp. 1209–1225, June 2006.

[49] F. J. Lin and P. H. Shen, "Robust fuzzy neural network sliding-mode control for two-axis motion control system," *Industrial Electronics, IEEE Transactions on*, vol. 53, no. 4, pp. 1209–1225, June 2006.

[50] F. J. Lin, P. H. Shen, S. L. Yang, and P. H. Chou, "Recurrent radial basis function network-based fuzzy neural network control for permanent-magnet linear synchronous motor servo drive," *Magnetics, IEEE Transactions on*, vol. 42, no. 11, pp. 3694–3705, Nov 2006.

[51] F. Lin, C. Lin, and P. Huang, "Recurrent fuzzy neural network controller design using sliding-mode control for linear synchronous motor drive," *Control Theory and Applications, IEE Proceedings* -, vol. 151, no. 4, pp. 407–416, July 2004.

[52] T. H. Liu, Y. C. Lee, and Y. H. Crang, "Adaptive controller design for a linear motor control system," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 40, no. 2, pp. 601–616, April 2004.

[53] MathWorks, *Matlab/Simulink Users Guide, Application Program Interface Guide*. The MathWorks, 2004.

[54] C. Maxfield, *The Design Warrior's Guide to FPGAs*. Newes of Elsivier, 2004.

[55] P. S. Maybeck, *Stochastic Models, Estimation, and Control*. New York: Academic Press, 1982.

[56] Modeltech, *ModelSim Reference Manual*, 2004.

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

[58] S. Morimoto, K. Kawamoto, M. Sanada, and Y. Takeda, "Sensorless control strategy for salient-pole pmsm based on extended emf in rotating reference frame," *Industry Applications, IEEE Transactions on*, vol. 38, no. 4, pp. 1054–1061, Jul 2002.

[59] J. Nie and D. Linkens, *Fuzzy-Neural Control: Principles, algorithms and applications*. UK: Prentice Hall, 1995.

[60] S. Nohara, M. Tomita, M. Hasegawa, S. Doki, and S. Kato, "A new design method of full-order extended electromotive force observer for position sensorless control of ipmsm," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, Nov 2013, pp. 2512–2517.

[61] T. Orlowska-Kowalska and M. Dybkowski, "Stator-current-based mras estimator for a wide range speed-sensorless induction-motor drive," *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 4, pp. 1296–1308, April 2010.

[62] L. Ozbek and F. Aliev, "Comments on adaptive fading kalman filter with an application," *Automatica*, vol. 34, pp. 1163 – 1164, 1998.

[63] L. Ozbek and M. Efe, "An adaptive extended kalman filter with application to compartment models," *Communication in Statistics - Simulation and Computation*, vol. 3, no. 1, p. 145158, Jan 2003.

[64] F. Poulain, L. Praly, and R. Ortega, "An observer for permanent magnet synchronous motors with currents and voltages as only measurements," in *Decision and Control, CDC08. 47th IEEE Conference on*, Dec 2008, pp. 5390–5395.

[65] Z. Qiao, T. Shi, Y. Wang, Y. Yan, C. Xia, and X. He, "New sliding-mode observer for position sensorless control of permanent-magnet synchronous motor," *Industrial Electronics, IEEE Transactions on*, vol. 60, no. 2, pp. 710–719, Feb 2013.

[66] N. K. Quang, Q. P. Ha, and N. T. Hieu, "Fpga sensorless pmsm drive with adaptive fading extended kalman filtering," in *Control Automation Robotics Vision (ICARCV), 2014 13th International Conference on*, Dec 2014, pp. 295–300.

[67] N. K. Quang, N. T. Hieu, and Q. P. Ha, "Fpga-based sensorless pmsm speed control using reduced-order extended kalman filters," *Industrial Electronics, IEEE Transactions on*, vol. 61, no. 12, pp. 6574–6582, Dec 2014.

[68] N. K. Quang, N. T. Hieu, G. Hunter, and Q. P. Ha, "Fpga-based sensorless pmsm drive using parallel reduced-order extended kalman filter," in *Control, Automation and Information Sciences (ICCAIS), 2012 International Conference on*, Nov 2012, pp. 164–169.

[69] N. K. Quang, Y. S. Kung, and Q. P. Ha, "Fpga-based control architecture integration for multiple-axis tracking motion systems," in *System Integration*

(SII), 2011 IEEE/SICE International Symposium on, Dec 2011, pp. 591–596.

[70] N. K. Quang, N. That, Q. N. Hong, and Q. P. Ha, "Fpga-based fuzzy sliding mode control for sensorless pmsm drive," in *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, Aug 2012, pp. 172–177.

[71] N. K. Quang, D. D. Tung, and Q. P. Ha, "Fpga-based sensorless pmsm speed control using adaptive extended kalman filter," in *Automation Science and Engineering (CASE2015), 2015 11th IEEE International Conference on*, Aug 2015, to be published.

[72] N. K. Quang and D. Q. Vinh, "Sensorless fpga-based pmsm drives using improved smo," in *Control and Automation (VCCA2013), 2013 The 1st Vietnam conference on*, Nov 2013, pp. 164–170.

[73] N. K. Quang, D. Q. Vinh, N. That, and Q. P. Ha, "Observer-based integral sliding mode control for sensorless pmsm drives using fpga," in *Control, Automation and Information Sciences (ICCAIS), 2013 International Conference on*, Nov 2013, pp. 218–223.

[74] M. Rahman, M. Haque, L. Tang, and L. Zhong, "Problems associated with the direct torque control of an interior permanent-magnet synchronous motor drive and their remedies," *Industrial Electronics, IEEE Transactions on*, vol. 51, no. 4, pp. 799–809, Aug 2004.

[75] L. Ribeiro, M. Degner, F. Briz, and R. Lorenz, "Comparison of carrier signal voltage and current injection for the estimation of flux angle or rotor position," in *Industry Applications Conference, 1998. Thirty-Third IAS Annual Meeting. The 1998 IEEE*, vol. 1, Oct 1998, pp. 452–459 vol.1.

[76] J. Rodriguez-Andina, M. Moure, and M. Valdes, "Features, design tools, and application domains of fpgas," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 4, pp. 1810–1823, Aug 2007.

[77] S. Sanchez-Solano, A. Cabrera, I. Baturone, F. Moreno-Velo, and M. Brox, "Fpga implementation of embedded fuzzy controllers for robotic applications," *Industrial Electronics, IEEE Transactions on*, vol. 54, no. 4, pp. 1937–1945, Aug 2007.

[78] B. Saunders, G. Heins, and F. D. Boer, "Framework for sensitivity analysis of industry algorithms for sensorless pmsm drives," in *Proc. 21st Australasian Conf. Univ. Power Eng*, Sept 2011, pp. 1–6.

[79] T. L. Seng, M. Khalid, R. Yusof, and S. Omatu, "Adaptive neuro-fuzzy control system by rbf and grnn neural networks," *Journal of Intelligent and Robotic Systems*, vol. 23, no. 4, pp. 267–289, June 1998.

[80] X. Shao and D. Sun, "Development of a new robot controller architecture with fpga-based ic design for improved high-speed performance," *Industrial Informatics, IEEE Transactions on*, vol. 3, no. 4, pp. 312–321, Nov 2007.

[81] N. Shieh and P. Tung, "Robust output tracking control of a linear dc brushless motor for transportation in manufacturing system," *Electric Power Applications, IEE Proceedings* -, vol. 148, no. 2, pp. 119–124, Mar 2001.

[82] T. L. Skvarenina, *The Power Electronics Handbook*. USA: CRC Press, 2002.

[83] Q. Song, J. Qi, and J. Han, "An adaptive ukf algorithm and its application in mobile robot control," in *Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on*, Dec 2006, pp. 1117–1122.

[84] D. L. Synder, "Information processing for observed jump processes," *Information and Control*, vol. 22, no. 1, p. 6975, Jan 1973.

[85] K. Szabat and T. Orlowska-Kowalska, "Performance improvement of industrial drives with mechanical elasticity using nonlinear adaptive kalman filter," *Industrial Electronics, IEEE Transactions on*, vol. 55, no. 3, pp. 1075–1084, March 2008.

[86] K. Tan, S. Huang, and T. Lee, "Robust adaptive numerical compensation for friction and force ripple in permanent-magnet linear motors," *Magnetics, IEEE Transactions on*, vol. 38, no. 1, pp. 221–228, Jan 2002.

[87] H. Tanaka, K. Ohnishi, H. Nishi, T. Kawai, Y. Morikawa, S. Ozawa, and T. Furukawa, "Implementation of bilateral control system based on acceleration control using fpga for multi-dof haptic endoscopic surgery robot," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 3, pp. 618–627, March 2009.

[88] P. Tomei and C. Verrelli, "Observer-based speed tracking control for sensorless permanent magnet synchronous motors with unknown load torque," *Automatic Control, IEEE Transactions on*, vol. 56, no. 6, pp. 1484–1488, June 2011.

[89] Y. Y. Tzou and H. J. Hsu, "Fpga realization of space-vector pwm control ic for three-phase pwm inverters," *Power Electronics, IEEE Transactions on*, vol. 12, no. 6, pp. 953–963, Nov 1997.

[90] U.Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*. Springer, 2001.

[91] K. Veluvolu and Y. C. Soh, "High-gain observers with sliding mode for state

and unknown input estimations," *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 9, pp. 3386–3393, Sept 2009.

[92] W. Wenjie, Z. Min, and W. Qinghai, "Application of reduced-order extended kalman filter in permanent magnet synchronous motor sensorless regulating system," in *Digital Manufacturing and Automation (ICDMA), 2010 International Conference on*, vol. 1, Dec 2010, pp. 271–274.

[93] X. Xi, L. Yongdong, Z. Meng, and L. Yan, "A sensorless control based on mras method in interior permanent-magnet machine drive," in *Power Electronics and Drives Systems, 2005. PEDS 2005. International Conference on*, vol. 1, 2005, pp. 734–738.

[94] Q. Xia, M. Rao, Y. Ying, and X. Shen, "Adaptive fading kalman filter with an application," *Automatica*, vol. 30, no. 8, pp. 1333 – 1338, 1994.

[95] W. Xiang and F. Chen, "Sliding mode control strategies for the hyperchaotic mck system," *Express Letters*, vol. 3, no. 3, pp. 283– 288, Nov 2009.

[96] Xilinx, *Xilinx on-line documentation.* Available in :www.xilinx.com.

[97] L. Xu and B. Yao, "Adaptive robust precision motion control of linear motors with negligible electrical dynamics: theory and experiments," *Mechatronics, IEEE/ASME Transactions on*, vol. 6, no. 4, pp. 444–452, Dec 2001.

[98] J. N. Yang, S. Lin, H. Huang, and L. Zhou, "An adaptive extended kalman filter for structural damage identification," *Structural Control and Health Monitoring*, vol. 13, no. 1, pp. 849–867, Jan 2006.

[99] J. Yang, R. Chen, and N. Fa, "A new recurrent fuzzy neural network sliding mode position controller based on vector control of pmlsm using svm," in *Power Electronics and Motion Control Conference, 2006. IPEMC 2006. CES/IEEE 5th International*, vol. 3, Aug 2006, pp. 1–5.

[100] J. Yang, L. Guan, and Y. Zhao, "An intelligent compensation strategy for permanent magnet linear synchronous motors drive," in *TENCON 2009 - 2009 IEEE Region 10 Conference*, Jan 2009, pp. 1–5.

[101] Y. H. Yu, N. Kwok, and Q. P. Ha, "Color tracking for multiple robot control using a system-on-programmable-chip," *Automation in Construction*, vol. 20, no. 6, pp. 669 – 676, 2011, selected papers from the 26th *ISARC* 2009.

[102] Y. Zhao, "Position/speed sensorless control for permanent-magnet synchronous machines," in *PhD Thesis, University of Nebraska*, USA, 2014.

[103] Q. Zhou, P. Shi, S. Xu, and H. Li, "Observer-based adaptive neural network control for nonlinear stochastic systems with time delay," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 71–80, Jan 2013.

[104] Z. Zhou, T. Li, T. Takahashi, and E. Ho, "Design of a universal space vector pwm controller based on fpga," in *Applied Power Electronics Conference and Exposition, 2004. APEC '04. Nineteenth Annual IEEE*, vol. 3, 2004, pp. 1698–1702 Vol.3.

[105] Z. Zhou, T. Li, T. Takahashi, and E. Ho, "Fpga realization of a high-performance servo controller for pmsm," in *Applied Power Electronics Conference and Exposition, 2004. APEC '04. Nineteenth Annual IEEE*, vol. 3, 2004, pp. 1604–1609 Vol.3.