FPGA Implementation of Impedance-Compensated Phase-locked Loop

by

Yue Yi

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Electrical and Computer Engineering
The University of Manitoba
Winnipeg, Manitoba, Canada
February 2019

Abstract

The Phase-Locked Loop (PLL) plays a key role in HVdc systems. Recently, a new type of PLL called the Impedance-Compensated Phase-Locked Loop (IC-PLL) was introduced to compensate for the voltage drop across the ac network's Thevenin impedance, making the phase locking more robust against transients and harmonics. The IC-PLL has an improved dynamic response as compared with the traditional approaches. However, prior to this thesis, studies of the IC-PLL were based on off-line simulations. In this thesis, an actual IC-PLL is constructed in hardware using a Field-Programmable Gate Array(FPGA). Paralleled structure is implemented on the FPGA to achieve high speed. The IC-PLL's performance is investigated by connecting it to a real-time model of the HVdc system simulated using a Real-Time Digital Simulator (RTDS). Different types of system disturbances such as sudden step change in power, voltage magnitude change and voltage distortion are applied to examine the IC-PLL's dynamic behaviour. Results are compared with the traditional Trans-vector PLL (TV-PLL). The results show the IC-PLL tracks the phase and frequency of the Point of Common Coupling(PCC) voltage in the steady-state and has a minimal error with load changes. Moreover, this thesis investigated potential sources of error when conducting the Controller Hardware-in-Loop (CHIL) simulation of the IC-PLL. The investigation was carried out by building a detailed off-line simulation of the CHIL simulation itself by including models of the interface delays, offsets and finite precision of ADC and DACs. The results show that for this type of interfacing, the constant interface delay and offsets have minimal impact on the accuracy, however, a finite precision of ADC and DACs will have a significant impact on the CHIL simulation.

Acknowledgments

I would like to thank people who helped and supported me during my master program. Firstly, I would like to express my deeply gratitude to my supervisor *Prof. Aniruddha Gole* for his guidance and support during my research studies. *Prof. Gole* is a great advisor who is very nice and knowledgeable. I appreciate his guidance and helpful advices to my research and writing of the thesis, the financial support from *Prof. Gole* is also appreciated. It is a great privilege for me to work under the supervision of *Prof. Gole* and I learnt a lot from him.

Besides, I would like to thank my committee members *Dr. Jenny Zhou* and *Dr. Carson Leung* for their valuable time and feedbacks on my thesis. Also, I would like to thank *Amy*, *Traci*, *Erwin*, *Zoran*, *Shrimal* and other faculty staff members for their help and assistance. Furthermore, I would like to thank my colleges and friends for all the helps they provided. Special thanks to *Dr. Mukesh K. Das*, *Huanfeng Zhao*, *Ajinkya Sinkar*, *Yi Qi*, *Zhiqiang Liu* and *Chen Jiang* who provided valuable suggestions and help on power system simulation. Also I would like to thank *Dong Li* and *King Man Siu* from RIGA lab for their helps on designing and building the hardware.

Finally, I would like to thank my parents *Jianghong Yi* and *Jiangang Luo*, my wife *Binglin Li* and my twin brother *Chao Luo* for their endless support, love and trust over these years. I could not have done this without them.

To my family

Contents

List of Figures

List of Tables

List of Abbreviations

ac Alternate Current

ADC Analog-to-Digital Converter

CORDIC COordination Rotation DIgital Computer

CC Constant Current

CEA Constant Extinction Angle

CHIL Controller Hardware-In-the-Loop

dc Direct Current

DAC Digital-to-Analog Converter

FPGA Field Programmable Gate Array

HVdc High Voltage Direct Current

HIL Hardware-In-the-Loop

IC-PLL Impedance-Compensated Phase-locked Loop

LPF Low Pass Filter

LCC Line-commutated Converter

PLL Phase-locked Loop

PCC Point of Common Coupling

RTDS Real Time Digital Simulator

SCR Short Circuit Ratio

TV-PLL Trans-vector Phase-locked Loop

VSC Voltage Source Converter

IGBT Insulated Gate Bipolar Transistor

VHDL VHSIC hardware description language

VHSIC Very High-Speed Integrated Circuit

Publications

Journal Papers

- Y. Yi, Ajinai, A.M. Gole,"FPGA Implementation of Impedance-Compensated Phase-Locked Loop for HVdc Converters," *The Journal of Engineering*, 2018, DOI: 10.1049/joe.2018.8789, IET Digital Library.

Conference Papers

- Y. Yi, A.D. Sinkar and A.M. Gole,"Effects of Time Delay, DC Offset and Truncation Errors on Interfacing of a Phase-Locked Loop (PLL) with a Real-time Simulator for Controller Hardware-In-the-Loop(CHIL) Simulation," *The 15th IET International Conference on AC and DC Power Transmission*, February 5-7,2019, Coventry, UK

Chapter 1

Introduction

In this chapter, the background and motivation of this research are discussed. Furthermore, the research objectives and the outline of the thesis are listed at the end of this chapter.

1.1 Background and Motivation

The rapid increase in electrical power demand has accelerated the development of High Voltage direct current(HVdc) transmission technology. Traditionally, electrical power was delivered from the generating station to the load through ac transmission lines. However, ac transmission system has some limitations and disadvantages for long distance transmission, such as high transmission losses, lower power transfer capability and possible loss of synchronism when interconnecting two asynchronous networks [1].

The first commercial HVdc system, the Gotland HVdc link began to operate in

1954. Transmitted power between the Swedish mainland and the Swedish island of Gotland[2]. Since then, many HVdc systems were built around the world to deliver power over long distances. In Canada, there are many HVdc projects such as the Nelson River Bipole 1, Bipole 2 and Quebec-New England HVdc system[3].

In HVdc systems, the PLL is used to provide fundamental frequency and phase information of the positive sequence voltage of the ac grid to generate firing pulses for HVdc converter valves. The tracking performance of the PLL will influence the operation of an HVdc system, thus, it is necessary to ensure a good tracking performance of the PLL. The Trans-Vector PLL(TV-PLL) is a widely used PLL topology in HVdc systems. Studies [4][5][6] have shown that the gain of PLL will have significant impact on the system performance in a weak grid. Recently, the Impedance-Compensated Phase-Locked Loop (IC-PLL) [7] was introduced to extend the stability range of weak grids with VSCs. This is achieved by introducing a virtual voltage point where the voltage drop across the ac network's Thevenin impedance was compensated. After that, Ajinai[8] investigated the dynamic performance of the IC-PLL in HVdc systems, the results show that the IC-PLL has improved dynamic performance as compared with traditional approaches such as the TV-PLL[9]. However, earlier studies on the IC-PLL are based on off-line simulations. In this research, the IC-PLL is physically constructed and its performance is studied in a real time environment.

The IC-PLL is implemented using a Field Programmable Gate Array(FPGA). Due to increased requirements for high speed computation, the FPGA is becoming increasingly popular in controller implementation in hardware[10][11][12][13]. The FPGA is a semiconductor device that can be reconfigured to achieve specific func-

tions. In contrast with general purpose DSPs or CPUs where instructions are executed sequentially, the parallel and hard-wired nature of the FPGA allows it to operate on instructions concurrently. Thus, the computation speed can be improved. Furthermore, the hard-wired structure of FPGA makes it more reliable.

Due to above reasons, the main motivation of this research is to construct a fast and reliable IC-PLL prototype on the FPGA platform and investigate its performance in real time environment.

1.2 Research Objectives

The objectives of this thesis are to design, implement and test an IC-PLL prototype on a FPGA platform. The following goals need to be achieved:

- Interfacing modules should be developed to allow the FPGA to communicate with external Analog-to-Digital Converter(ADC) and Digital-to-Analog Converters(DACs) to sample and output data for calculation and testing purposes. Moreover, the FPGA-based IC-PLL should be able to handle at least 12 I/Os (PCC voltages and currents, Estimated Voltages, frequency output, etc.). This requirement is related to the structure of the IC-PLL.

- The FPGA-based IC-PLL should be able to handle arithmetic calculations and complex algorithms in a fast way where parallel operations are utilized.

- Transfer functions should be implemented on the FPGA and be able to run in real-time

- The FPGA-based IC-PLL should be easily interfaceable to an actual or real-time model of an HVdc system, so that detailed performance analysis can be carried out. This includes both open loop test and Controller Hardware-in-the-Loop(CHIL) test.

1.3 Outline of the Thesis

The main part of the thesis contains six chapters and are organized as following:

- **Chapter 2**: In this chapter, fundamentals of HVdc transmission system are reviewed which include the overview of HVdc transmission system, converter station basics and different HVdc transmission configurations. In addition to these, operating principles and control strategies of LCC-HVdc and VSC-HVdc system are discussed.
- **Chapter 3**: In this chapter, the basic operating principle of the Phase-Locked Loop(PLL) which includes the Trans-Vector PLL(TV-PLL) and Impedance-Compensated PLL(IC-PLL) are discussed. Typical application of the IC-PLL in an LCC-HVdc system is explained.
- **Chapter 4**: FPGA features and programming techniques such as concurrent and sequential operations, the pipelining technique are discussed in this chapter. The data representation used to implement the IC-PLL inside FPGA is explained.
- **Chapter 5**: Chapter 5 presents the FPGA implementation of the IC-PLL which includes the I/O interface design as well as the IC-PLL module design. In the IC-PLL module design, the implementation of phase detector, loop filter and

voltage-controlled oscillator are discussed

- **Chapter 6**: In this chapter, both open loop test and CHIL test are performed to verify the performance of the FPGA-based IC-PLL. The experiment hardware setup and system models which are used to verify the FPGA-based IC-PLL functionality are explained. Test results are presented and the effect of various parameters with respect to the CHIL simulation are discussed.
- **Chapter 7**: The important contributions of this thesis are summarized. The possible extensions that can be made are discussed. Several suggestions for future work are presented.

Chapter 2

HVdc Transmission System Fundamentals

In this thesis, the FPGA-based IC-PLL is constructed which is used for HVdc converters. The performance of the FPGA-based IC-PLL is investigated in HVdc transmission system. It is necessary to understand the basic concepts and operation principles of the HVdc transmission system. In this chapter, a brief overview of HVdc transmission system are explained.

2.1 Overview of HVdc Transmission System

An HVdc transmission system consists of three main components: the ac network, the converter station and the transmission line which is shown in Fig. 2.1. The power is generated from one side of the ac network and converted from ac to dc at the rectifier station. Then the dc power is transmitted through the HVdc transmission line and

converted back into ac form to provide power to the ac network on the other side.

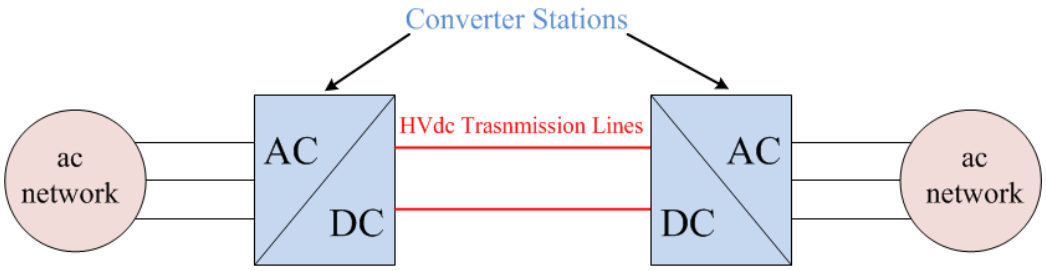


Figure 2.1: Basic Structure of HVdc Transmission System

Some advantages of using HVdc system are[1][14][15][16]:

- For long distance power transmission, the HVdc transmission system has lower power loss as compared to the HVac transmission system, moreover, the cost of dc line is less than that of ac line since HVdc transmission requires fewer conductors.
- It is possible to interconnect systems with different frequencies using HVdc.
- With the same insulation level, dc lines can carry more power than ac lines. For ac lines, the effective voltage is $\frac{1}{\sqrt{2}}$ of the peak ac voltage whereas, for HVdc lines, the effective voltage is same as the actual dc voltage.
- Dc lines are more suitable for long distance power transmission. Unlike ac lines, the current and voltage are always in phase (i.e.. unity power factor) while ac lines would require reactive compensation.
- Underwater cables have high capacitance that would increase transmission losses while using ac lines, however, it has minimal effect on dc lines.

2.2 Converter Station Basics

In an HVdc transmission system, the structure of converter stations at sending and receiving end are similar. A typical converter station has the following components[14][15][16][17]:

- **Converter Valves**: The conversion from ac to dc (Rectifier) and dc to ac (Inverter) is done by semiconductor switches. The converter topology varies based on the type semiconductor switches or the converter configuration selected. At present, the two dominant converter technologies are LCC and VSC. The LCC converter uses thyristor technology. Thyristors are semiconductor devices which can only be turned on, however, turn off only happens when the ac voltage applies a reverse bias. Hence, its operation is greatly affected by the ac voltage. The VSC converter uses IGBT switches which are fully controllable(can be turned on as well as turned off), therefore, the ac voltage is not critical.
- **Converter Transformers**: The main function of the converter transformer is to adjust the ac bus voltage to a designated voltage level for the converter. In additional to that, the transformer provides isolation between ac and dc systems. In LCC systems, the converter transformer has a higher leakage reactance compared to conventional transforms to limit the short circuit current generated from faulted thyristors. A on-load tap changer is often seen in an LCC system for voltage regulation purposes. In a VSC system, the converter transformer is also known as interface transformer which links the ac system with the VSC converter and to adjust the ac voltage level and limit rate of rise of fault current[15].

- **Harmonic Filters**: HVdc converters generate harmonics on both ac and dc sides during normal operation, and these harmonics will have negative impact on the ac side system(eg. ac voltage distortion,telephone interference). Therefore, it is essential to have harmonic filters to filter out these harmonics. In a 12-pulse LCC-HVdc system, the characteristic harmonic currents generated on the ac side are of order $12n \pm 1$(where $n = 1, 2, 3...$). The ac filter is designed to absorb these characteristic harmonics. Furthermore, the LCC converter consumes reactive power which is normally around 50-60% of the real power. The reactive power compensation is mainly supplied by the ac filter together with other compensator(eg. shunt capacitors, static variable capacitors, synchronous condensers)[15]. In VSC-HVdc systems, with the use of IGBTs, harmonics generated by the converter are often at higher frequency according to its switching frequency, which means the size and the cost of the ac filter in VSC-HVdc is less compared to LCC-HVdc system. On the other hand, with the fully controllability of IGBTs, the ac filter in VSC system does not require to provide reactive power compensation. Similarly, the dc filter is designed to reduce harmonics on the dc side of the converter to mitigate problems such as telephone interference.
- **Reactors**: In LCC-HVdc systems, Smoothing reactors are installed to remove ripple on the dc line and to limit the rate of sudden current change during system disturbances to protect converter valves when commutation failure happens. In VSC-HVdc systems, phase reactors are used to control real and reactive power by regulating the current through them. Moreover, phase reactors work along with ac filters to reduce high frequency harmonic penetration into the ac network

produced by the VSC converter during its operation.

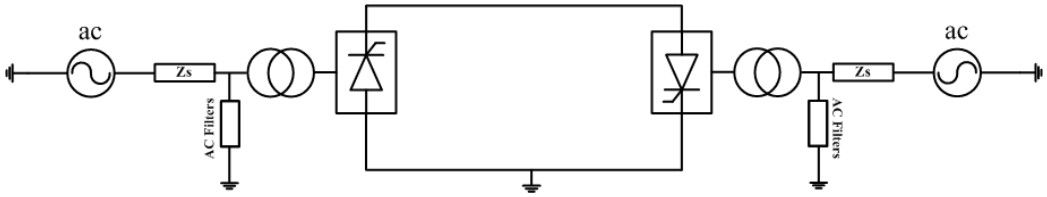
2.3 HVdc Transmission Configurations

An HVdc transmission system can have different configurations based on either the physical location of converters or the operation strategy of the HVdc system. Some examples of HVdc transmission configurations are shown in Fig. 2.2:[1][14][15]:

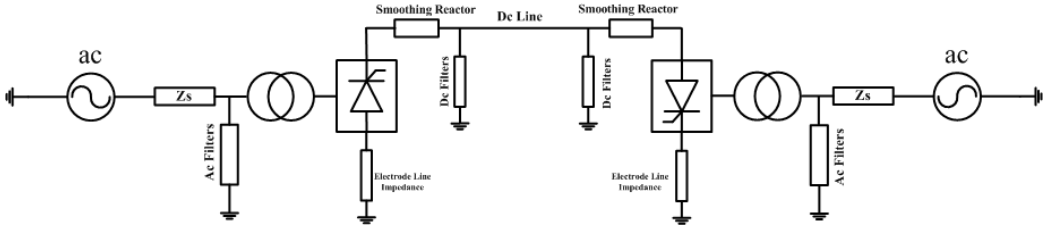
- **Back-to-back Configuration**: Fig. 2.2a illustrates the back-to-back configuration. In this configuration, the rectifier and inverter are located at the same converter station and because of that, dc side filters are sometimes not necessary in this configuration since the dc line is very short(eg. few meters) and coupling of harmonic into adjacent circuits is unlikely to happen. This configuration is often used to connect two ac grids that are asynchronous and to improve the stability of the system. The dc link voltage rating of a back-to-back system is relatively lower compared to point-to-point HVdc configurations. On the other hand, the current rating can be as high as permitted by device ratings. The HVdc configuration is called the point-to-point configuration(eg. Fig. 2.2b and 2.2c) if rectifier and inverter are located at different places. This configuration is often used for long distance transmission where two converters are connected through the dc link.
- **Monopolar Configuration**: In monopolar configuration, converters are connected by a single pole line which can be either positive or negative depending on the direction of the power flow. The earth and sea can be used as return path, however, it is currently more common to use a metallic return path due

to environment constrains or other reasons. Fig. 2.2b shows a monopolar configuration with earth return.

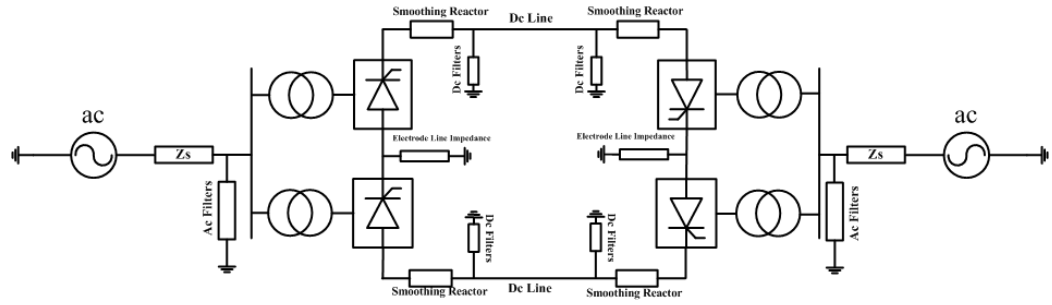
- **Bipolar Configuration**: A bipolar configuration with earth return is shown in Fig. 2.2c. A bipolar configuration can be seen as combination of two monopolar system with shared return path where one pole has positive polarity and the other one has negative polarity. The power transfer capability of this configuration is typically double that of the monopolar configuration. Moreover, each pole can operate independently, if one pole is under fault condition, the rest of the system can operates as a monopolar system. The bipolar configuration is the most widely used configuration in existing HVdc systems.
- **Multi-terminal Configuration**: In the multi-terminal configuration, three or more converter stations are connected together. The multi-terminal configuration is more complicated than the point-to-point system. The multi-terminal configuration can be either series or parallel. A parallel multi-terminal configuration is shown in Fig. 2.2d.



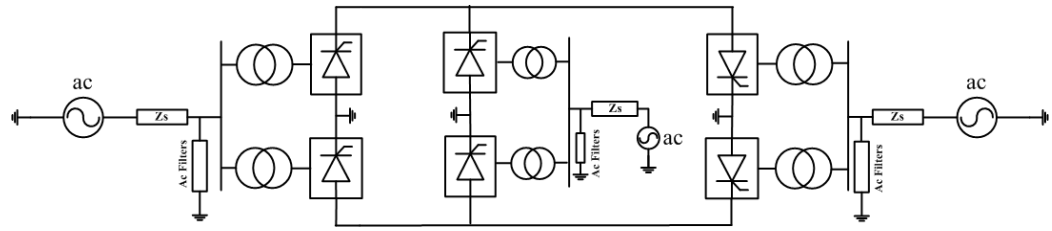
(a) Back-to-back Configuration



(b) Monopolar Configuration



(c) Bipolar Configuration



(d) Multi-Terminal Configuration

Figure 2.2: Examples of HVdc Transmission Scheme

2.4 LCC-HVdc system

2.4.1 LCC-HVdc Operating Principle

The Line-Commutated Converter(LCC), also known as the Current Source Converter(CSC) is a more mature technology compared with other converter technologies[18]. The most commonly used LCC converter is often consists of six-pulse thyristor bridges namely *Graetz Bridges*[14][18].

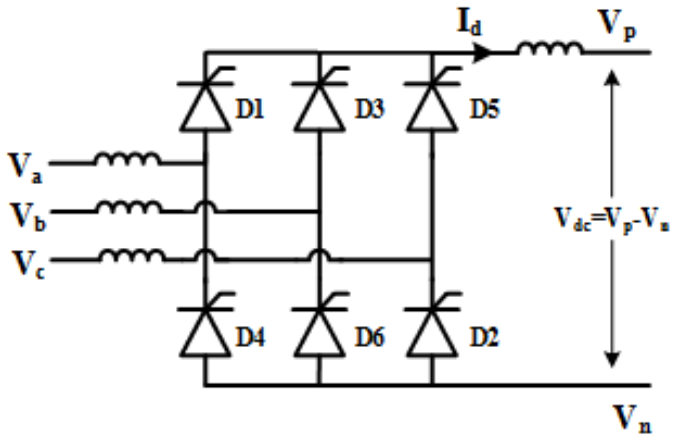


Figure 2.3: Example of a 6-pulse Converter

Fig. 2.3 illustrates an example of a 6-pulse converter with transformer leakage inductance taken into account. The bridge consists of six thyristors where two thyristors are connected in series to form a set and three sets are connected in parallel to represent three phase. During normal operation, the positive terminal voltage V_p is equal to the maximum voltage among three phase voltages(V_a, V_b and V_c), and the negative voltage V_n is equal to the minimum voltage among three phase voltage. The dc voltage V_{dc} is the difference between V_p and V_n. The thyristor conducts when the firing pulse is applied and there is forward bias voltage between thyristor terminals.

The thyristor can only be turned off when the current through it drops below holding current. The angle between the instant at which the thyristor forward bias appears and the time instance that firing pulse is generated is called the firing angle α. Due to the leakage inductance of the transformer, it takes time for current to make transition from one phase to another, the overlap interval is described as overlap angle μ. The LCC operates in rectifier mode with α less than 90^o and in inverter mode with α greater than 90^o. In inverter mode, the extinction angle γ is often used which satisfies the condition $\gamma = 180^o - \mu - \alpha$[14]. The voltage waveform of an 6-pulse converter is illustrated in Fig. 2.4

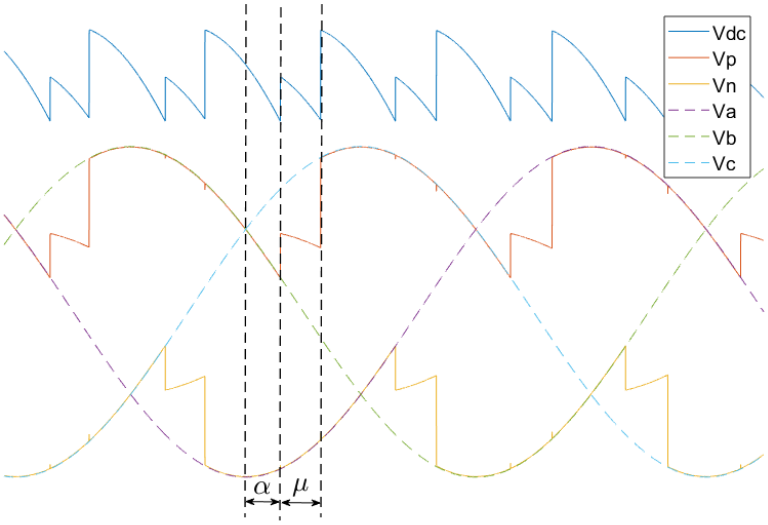


Figure 2.4: Voltage Waveform of a 6-pulse Rectifier

2.4.2 LCC-HVdc Control Strategy

Fig. 2.5 illustrates an equivalent circuit of the LCC-HVdc system. In the LCC-HVdc system, the current can only flow in one direction, The control of voltages on both converter side(V_{dr} and V_{di}), the dc current(I_{dc}) and the power (P_{dc}) are necessary during normal operation.

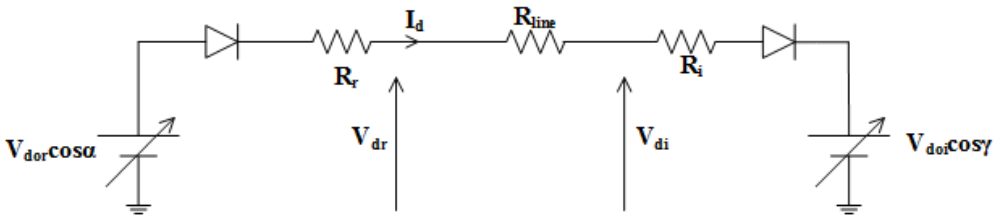


Figure 2.5: Equivalent Circuit of LCC-HVdc System

The voltage and current (V-I) characteristic is often used to determine the operating point for converters and it varies with respect to different control methods been applied. Fig. 2.6 shows a basic control strategy in LCC-HVdc systems. Equation 2.1 describles the V-I relationship for a fixed firing angle α. Controlling α with a feedback loop allows one to add the desired functionality to the control. For example, on the rectifier side, the normal operating mode is Constant Current(CC) control and α can be adjusted to change the dc voltage so that the dc current remains constant, as shown in the current control loop in the control diagram of Fig.2.7. Likewise, on the inverter side, the operating mode is Constant Extinction Angle(CEA) control, where the extinction angle γ is controlled to typically 15-18 degrees so as to reduce the possibility of Commutation Failure(CF). Again, this is done by suitably changing α in equation 2.1 to increase α when γ is too large and decrease α when γ is too small. Equation 2.2 describes the resulting V-I characteristic. The resulting control charac-

teristic for rectifier and inverter are as shown in Fig. 2.6. The current margin(ΔI_d) is generally 10% of the current order.

$$V_{dr} = \frac{3\sqrt{2}}{\pi} V_l \cos\alpha - \frac{3}{\pi} X_{cr} I_{dr} \tag{2.1}$$

$$V_{di} = \frac{3\sqrt{2}}{\pi} V_l \cos\gamma - \frac{3}{\pi} X_{ci} I_{di} \tag{2.2}$$

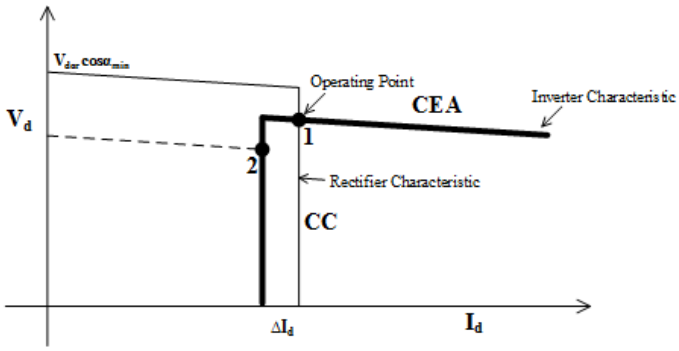


Figure 2.6: Example of LCC-HVdc Control Strategy

There are many other control schemes that can be used to improve the dynamic characteristic and system performance such as Current Error Control(CEC), Voltage Dependent Current Order Limit(VDCOL) control which are not discussed in this section.

Fig. 2.7 shows the control diagram in an LCC-HVdc system when Constant Current(CC) and Constant Extinction Angle(CEA) control are selected. The transition from CEA to CC is achieved by having both control loops in the control system and select the smaller firing angle order α_o among them. The zero crossing of the positive sequence ac voltage is detected by using the Phase-Locked Loop(PLL). The output

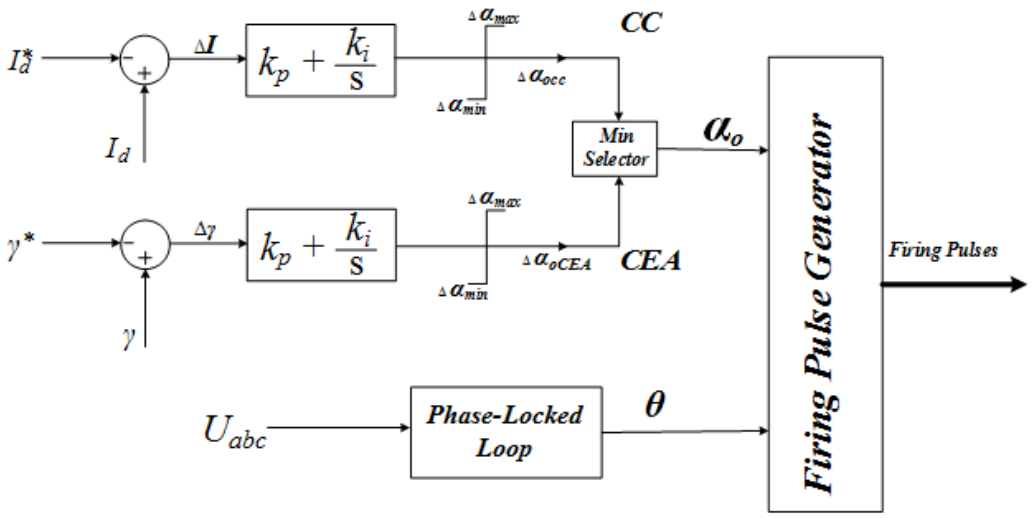


Figure 2.7: Example of Control Blocks in LCC-HVdc System

angle θ_{PLL} from the PLL is compared with the firing angle α_o from the controller to generate firing pulses. Thus, the PLL is very important in LCC-HVdc systems since it provides critical phase and frequency information to the firing pulse generator.

2.5 VSC-HVdc System

2.5.1 2-level VSC-HVdc Operating Principle

The Voltage Source Converter(VSC) is a modern technology which is newer than the LCC and is seeing increasing application. The VSC uses the Insulated Gated Bipolar Transistors(IGBT) device as the basic switch element. Unlike the thyristor, the IGBT can be turned on and off by controlling the gate pulse signal. A single phase 2-level converter is used to demonstrate the operating principle of VSC which is shown in Fig. 2.8. In contrast to the 6-pulse converter discussed earlier, thyristors

are replaced by IGBTs with anti-parallel diode in VSC. The voltage output is $\frac{V_{dc}}{2}$ when the upper IGBT is turned on, and the voltage output is $-\frac{V_{dc}}{2}$ when the lower IGBT is turned on. The gate pulse is usually generated by the Pulse Width Modulation(PWM), Sinusoidal PWM and other techniques to further reduce the low order harmonics in the output voltage. Furthermore, the VSC has ability of controlling real and reactive power[17].

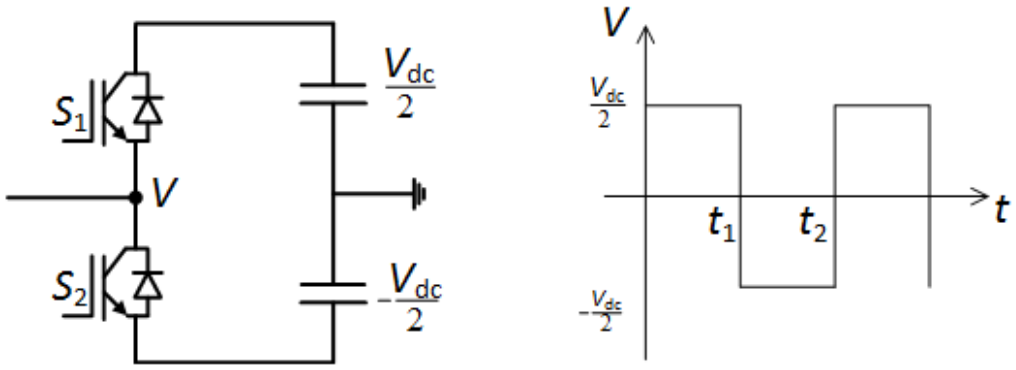


Figure 2.8: 2-level VSC Topology and Waveform

2.5.2 Modular Multi-level Converter Topology

In modern converters, the Modular Multi-level Converter (MMC) topology is fast replacing the 2-level topology discussed above. As shown in Fig. 2.10, the MMC has three phase legs, one for each phase. Each leg consists of an upper and a lower arm. Each arm contains N number of sub-modules in series. The sub-module can be either full bridge or half bridge[19]. With the half bridge sub-module shown in the Fig. 2.10. The sub-modules are all controlled to have the same capacitor voltage v_{cap}. If switch $S1$ is on, the sub-module is inserted and contributes to the arm voltage. And

if $S2$ is on, the sub-module is bypassed and the output voltage is zero. By controlling the inserted or bypass status of all sub-modules, a staircase waveform as shown in Fig. 2.9 results. This waveform is closely made to follow the desired output sine wave reference, and becomes the ac side voltage. An advantage of this topology is that, with a large number of sub-modules, the harmonic content of the ac voltage becomes very small and there is no need for additional filtering. However, the higher level control loops for either case are the same,i.e., in both cases, an reference voltage ac waveform is provided for each phase as will be discussed in the next section(Fig. 2.11), and so for the purpose of this thesis, no distinction is made as to the type of VSC converter.

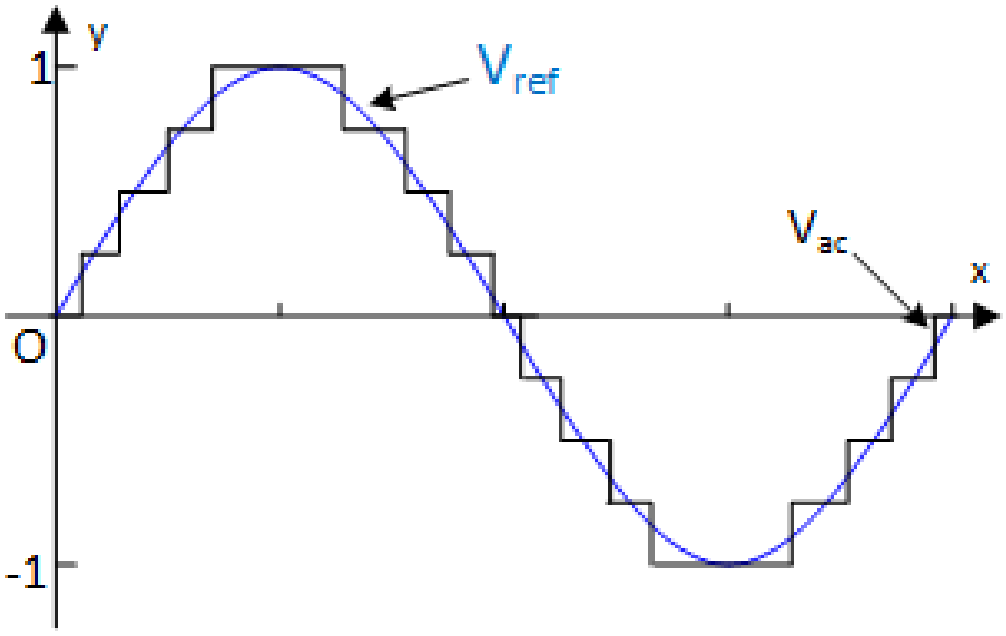


Figure 2.9: Module Multi-level Converter Waveform

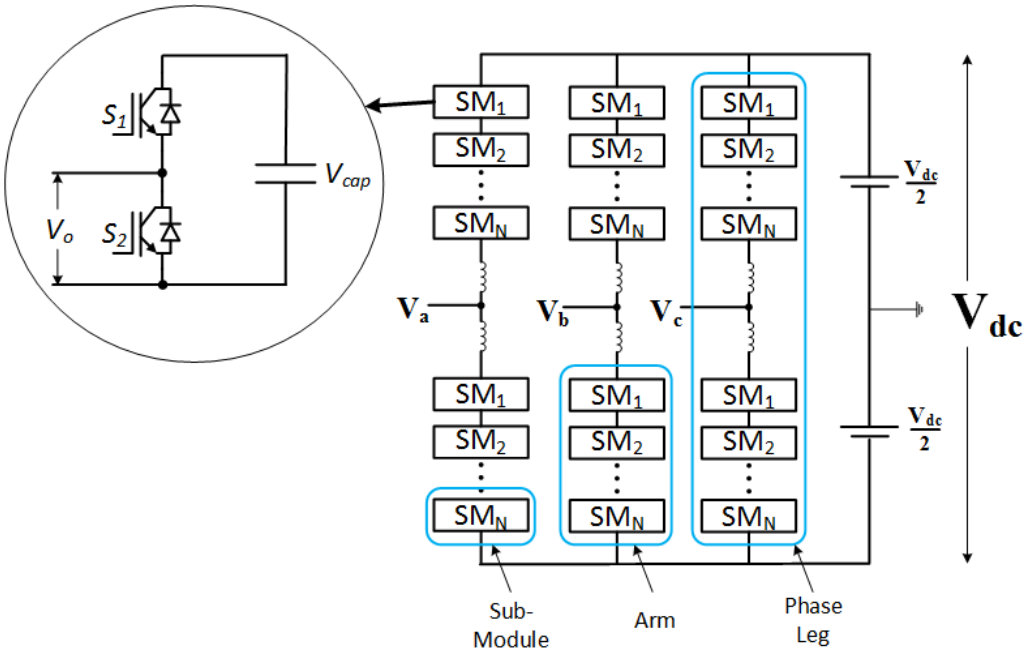


Figure 2.10: Module Multi-level Converter Topology

2.5.3 VSC-HVdc Control Strategy

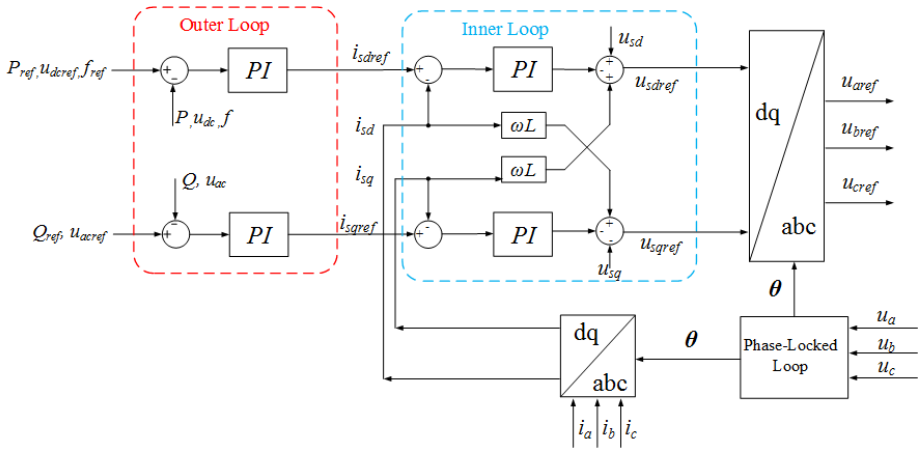


Figure 2.11: A Control Diagram of VSC

In the VSC-HVdc system, the commonly used control scheme is the decoupled

control. This control scheme consists of two loops which are the outer loop and inner loop. The real and reactive power of the VSC can be controlled by adjusting the current order reference on dq axis since $P = V_d I_d$ and $Q = V_d I_q$ [14]. The control diagram is shown in Fig. 2.11. The current reference signals i_{sdref} and i_{sqref} are generated in the outer loop. Since P is proportional to i_d and Q is proportional to i_q, the i_d, i_q references can be generated using signals which are related to real power(P_{ref},u_{dcref} or f_{ref}) and reactive powers(Q_{ref} or u_{acref}). In the inner loop, the current reference signals are compared with the measured current signals(i_{sd} and i_{sq}). The $\omega L i_{sd}$ and $\omega L i_{sq}$ feedback signals ensure decoupled control of the d and q axis currents. For example, when a step change is made in i_{sd}, i_{sq} remains unchanged and vice versa. The reference voltage signals in dq domain are produced and converted into individual phase reference voltage(u_{aref},u_{bref} and u_{cref}) which are then implemented by the VSC. Note that the PLL is an essential element that provides the timing reference for the phase angle of the ac voltage and is used in the abc to/from dq transformation($abc - dq$ transformation is explained explicitly in section 3.2)[20].

Chapter 3

The Phase-Locked Loop Fundamentals

The Phase-Locked Loop(PLL) is a feedback control system whose output is a signal that tracks the phase angle of the input voltage. The PLL contains three parts: the Phase Detector(PD), the Loop Filter(LF) and the Voltage Controlled Oscillator(VCO) which are shown in Fig. 3.1.

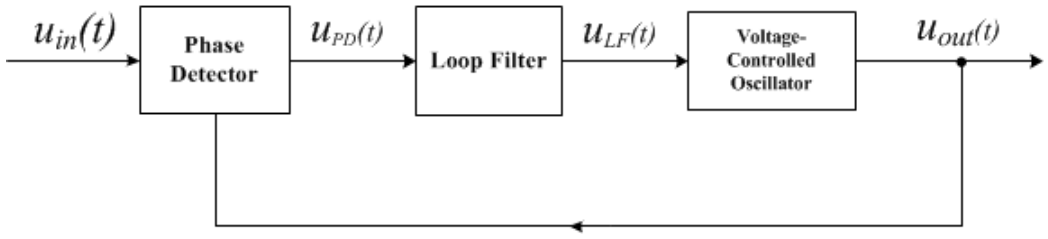


Figure 3.1: Basic Structure of PLL

The PD compares the input $u_{in}(t)$ and output signal $u_{out}(t)$ then generates a phase error signal $u_{PD}(t)$ which is proportional to the phase difference between $u_{in}(t)$

and $u_{out}(t)$. Then the LF filters out the high frequency components(ac components) inside the phase error signal $u_{PD}(t)$ and produce a corresponding dc signal $u_{LF}(t)$ that will be passed to the VCO. In addition, the LF often includes a Proportional-Integral (PI) element which reduces the steady-state phase angle error to zero. This ensures the output of the VCO tracks the phase of the input voltage. The VCO takes the dc signal and generates an output signal which contains the frequency and phase information of the input signal [21].

3.1 The Phase-Locked Loop Operating Principle

To show the operating principle of the PLL, a multiplier is used as the PD and the PLL is initially unlocked. Then the input and output of the PLL is expressed as shown in equation 3.1 and 3.2 [22]:

$$u_{in}(t) = U_{min}\cos(\omega_{in}t + \theta_{in}) \tag{3.1}$$

$$u_{out}(t) = U_{mout}\cos(\omega_{out}t + \theta_{out}) \tag{3.2}$$

After passing the PD with a gain of k_{pd}, we obtain equation 3.3 :

$$\begin{aligned} u_{PD}(t) = \frac{1}{2}k_{pd}U_{min}U_{mout}\Big\{ &\cos\left\{(\omega_{in} - \omega_{out})t + (\theta_{in} - \theta_{out})\right\} \\ &+ \cos\left\{(\omega_{in} + \omega_{out})t + (\theta_{in} + \theta_{out})\right\}\Big\} \end{aligned} \tag{3.3}$$

The higher frequency component in equation 3.3 is filtered out by the LF, the remaining part in equation 3.3 becomes the output of LF

$$u_{LF}(t) = \frac{1}{2}k_{pd}U_{min}U_{mout}\Big\{\cos\left\{(\omega_{in} - \omega_{out})t + (\theta_{in} - \theta_{out})\right\}\Big\} \tag{3.4}$$

After the transient, the output signal becomes in synchronization with the input signal which can be expressed as [22]:

$$u_{out}(t) = U_{mout}\cos(\omega_{in}t + \phi_{out}) \tag{3.5}$$

During the transient period, by comparing equation 3.2 and 3.5, the phase angle θ_{out} can be obtained as shown in equation 3.6 [22]:

$$\theta_{out} = (\omega_{in} - \omega_{out})t + \phi_{out} \tag{3.6}$$

then the LF output in 3.4 becomes [22]:

$$u_{LF}(t) = \frac{1}{2}k_{pd}U_{min}U_{mout}\cos(\theta_{in} - \phi_{out}) \tag{3.7}$$

It is clearly seen from equation 3.7 that the LF output is a dc signal.

The VCO is often treated as a linear time-invariant system, the instantaneous angular frequency ω_{inst} is controlled by the filtered signal $u_{LF}(t)$ around the central angular frequency ω_{out}, the relationship can be express in equation 3.8 [22]

$$\omega_{inst}(t) = \frac{d}{dt}(\omega_{out}t + \theta_{out}) = \omega_{out} + k_{vco}u_{LF}(t) \tag{3.8}$$

where k_{vco} is the sensitivity of VCO. From equation 3.6, 3.7 and 3.8 we obtain:

$$\omega_{in} - \omega_{out} = \frac{1}{2}k_{pd}U_{min}U_{mout}k_{vco}\cos(\theta_{in} - \phi_{out}) \tag{3.9}$$

then

$$\phi_{out} = \theta_{in} - \cos^{-1}\frac{\omega_{in} - \omega_{out}}{\frac{1}{2}k_{pd}U_{min}U_{mout}k_{vco}} \tag{3.10}$$

By substituting equation 3.10 into 3.7 we have:

$$u_{LF} = \frac{\omega_{in} - \omega_{out}}{k_{vco}} \tag{3.11}$$

Rewrite 3.11 in form of 3.12:

$$\omega_{in} = \omega_{out} + k_{vco} u_{LF} \tag{3.12}$$

It is clearly seen from equation 3.12 that the dc signal u_{LF} changes the VCO output frequency from its central angular frequency ω_{out} to the input angular frequency ω_{in} in a linear fashion [22].

Moreover, from equation 3.10, if the angular frequency difference $(\omega_{in} - \omega_{out})$ is much smaller than the loop gain $(\frac{1}{2} k_{pd} U_{min} U_{mout} k_{vco})$, we have $\theta_{in} - \phi_{out} \approx \cos^{-1} 0 = \frac{\pi}{2}$ which indicates that the input signal is in quadrature with the VCO output signal.

let

$$\theta_{out} = \phi_{out} + \frac{\pi}{2} \tag{3.13}$$

then equation 3.7 becomes:

$$u_{LF}(t) = \frac{1}{2} k_{pd} U_{min} U_{mout} \sin(\theta_{in} - \theta_{out}) \tag{3.14}$$

where $\theta_{in} - \theta_{out}$ is the phase error between input and output signals. If $\theta_{in} - \theta_{out}$ is small enough, we have:

$$u_{LF}(t) \approx \frac{1}{2} k_{pd} U_{min} U_{mout} (\theta_{in} - \theta_{out}) \tag{3.15}$$

Therefore, the VCO adjusts the output frequency and phase based on the filtered dc signal u_{LF} whose value is proportional to the phase error between the input signal u_{in} and the output signal u_{out} .

3.2 Trans-Vector Phase-Locked Loop

The Trans-Vector type PLL(TV-PLL) [9] is widely used in HVdc system due to its excellent immunity from harmonic distortion or loss of ac synchronizing voltage. The basic topology of TV-PLL is shown in Fig. 3.2:

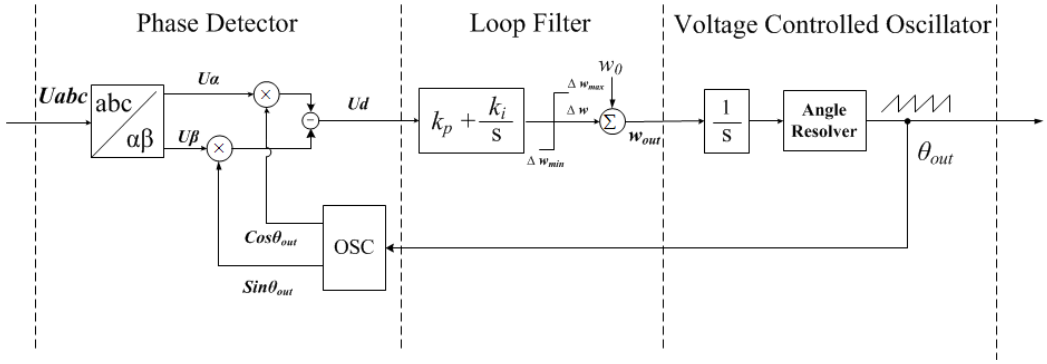
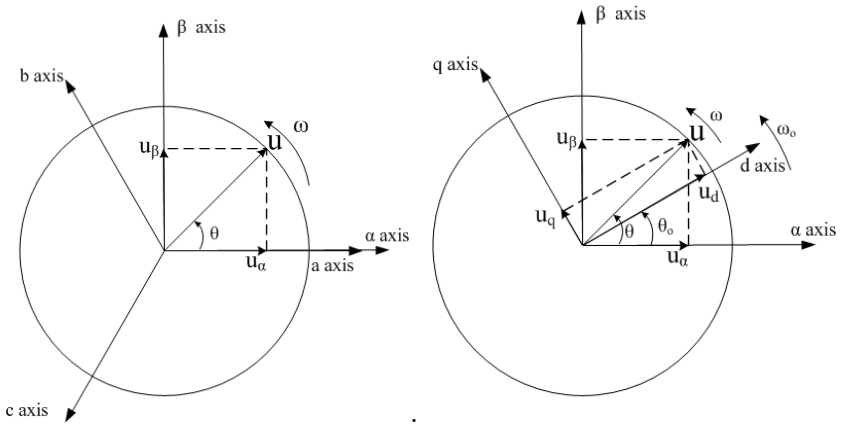


Figure 3.2: Basic Structure of TV-PLL

In the TV-PLL, the Phase Detector(PD) function is achieved by using the $abc - \alpha\beta - dq$ transformation [20][23] which is illustrated in Fig. 3.3:



(a) Clarke Transformation (b) Park Transformation

Figure 3.3: *abc*-$\alpha\beta$-dq Transformation

As shown in Fig.3.3a, the $abc - \alpha\beta$ transformation(also known as the Clarke transformation) projects the resultant vector u of instantaneous three phase quantities onto two axis stationary frame($\alpha\beta$ frame) where u has an angular frequency of ω. The simplified Clarke transformation matrix is shown in equation 3.16[23]:

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} \tag{3.16}$$

After the Clarke transformation, the $\alpha\beta - dq$ transformation(Park Transformation) projects the two phase quantities(u_α and u_β) onto a rotating reference frame(dq frame) where the rotating frame has an angular frequency of ω_o. The $\alpha\beta - dq$ transformation matrix is expressed as in equation 3.17[20]:

$$\begin{bmatrix} u_d \\ u_q \end{bmatrix} = \begin{bmatrix} \cos(\omega_o t) & \sin(\omega_o t) \\ \sin(\omega_o t) & -\cos(\omega_o t) \end{bmatrix} \begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} \tag{3.17}$$

From Fig. 3.3b, when $u_q > 0$, u leads u_d, when $u_q < 0$, u lags u_d and when $u_q = 0$, u and u_d are in phase, therefore, the phase error can be detected by using $abc - \alpha\beta - dq$ transformation.

The operating principle of LF and VCO are similar to the multiplier-based PLL which is describe in section 3.1 except that using information from all three phases further reduces the ripple in the phase difference signal and so the Low-Pass Filter(LPF) can even be eliminated. Of course, any imbalance in the phase voltages will cause some ripple, but it is usually small. The PI controller is used as the LF which eliminates the steady-state phase(and hence also frequency) error. It can be shown that the TV-PLL actually tracks the positive sequence phase A component of

the three phase voltage. Then the VCO generates the output angle θ_{out} based on the corrected signal ω_{out}.

3.3 Impedance-Compensated Phase-Locked Loop

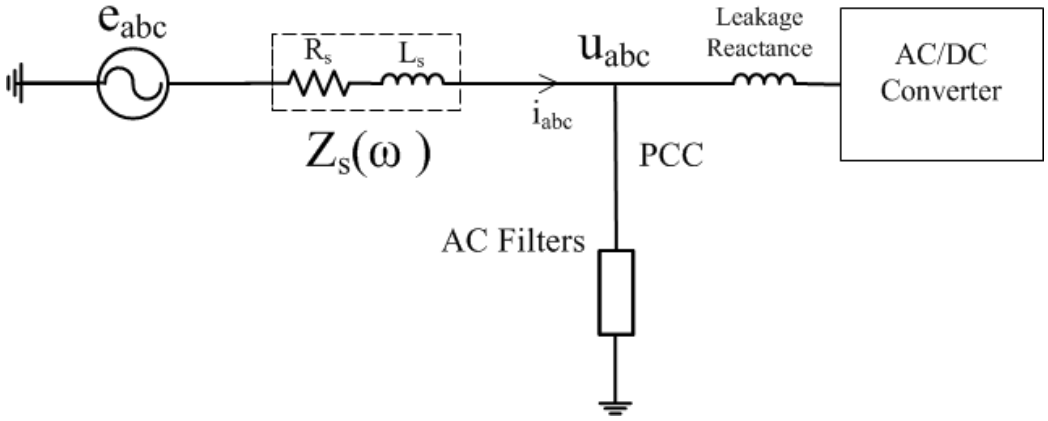


Figure 3.4: Representation of AC Network in the HVdc System(Rectifier Side)

An equivalent circuit of the ac network in the HVdc system is shown in Fig. 3.4. The ac grid is represented as a Thevenin voltage source e_{abc} in series with the Thevenin equivalent impedance Z_s(in general, the Thevenin equivalent is not a simple R-L circuit, but it can be shown[8] that using an R-L representation based on the fundamental frequency impedance still gives good results). The voltage at the Point of Coupling(PCC) is u_{abc} and the current is i_{abc}. The TV-PLL tracks the frequency and the phase of the positive sequence(phase A) voltage at PCC. However, when the PCC voltage undergoes a transient(eg. due to load change in the converter), the phase of the PCC voltage suddenly changes and the PLL needs to re-synchronize with the new phase. Also, with unbalances and other transients, the PCC voltage

may get distorted or unbalanced, adding increase tracking difficulty for the PLL. The Impedance-Compensated Phase-Locked Loop(IC-PLL)[7] was introduced to make the phase locking more robust compared to the TV-PLL. Compared with the TV-PLL, instead of locking onto the voltage at PCC, an extra term is added to compensate the voltage drop across the ac network impedance, by doing so, the IC-PLL is locked onto a virtual estimated voltage point(ideally the fixed Thevenin voltage) which is potentially less distorted than the PCC voltage.

By using Kirchhoff's Voltage Law (KVL) on the circuit shown in Fig.3.4, equation 3.18 can be obtained:

$$\begin{bmatrix} e_a \\ e_b \\ e_c \end{bmatrix} = \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} + L_s \frac{d}{dt} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + R_s \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} \tag{3.18}$$

To estimate the source voltage e_{abc}, the fundamental frequency components need to be extracted. This is carried out in the dq domain. By applying Appendix C equation C.2 and C.1, the expression of 3.18 in dq domain is:

$$\begin{bmatrix} e_d \\ e_q \end{bmatrix} = \begin{bmatrix} u_d \\ u_q \end{bmatrix} + L_s \frac{d}{dt} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} R_s & \omega_0 L \\ -\omega_0 L & R_s \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} \tag{3.19}$$

where ω_0 is the fundamental angular frequency we are interested in.

For a perfectly balanced three phase sinusoidal signal, the projection of the fundamental frequency component on the rotating dq frame is a dc value. However, if the signal contains harmonics, the output will have ripple. A Low-Pass Filter (LPF) is added to remove the ripple.

3.4 IC-PLL in the LCC-HVdc system

Generally, the output signal of the PLL is in synchronization with the reference voltage(PCC voltage) which is used as a reference signal to generate the firing pulse for converter thyristors at the ordered firing angle. Different from the TV-PLL, the output of the IC-PLL is synchronized with the estimated voltage e_{abc}. Therefore, tracking e_{abc} does not immediately give the phase of the PCC voltage as there is a phase shift. Note that for the converter valves, we need the latter. To compensate the phase difference, an angle correction module is added to ensure that the phase angle delivered to the firing pulse generator is in synchronization with the PCC voltage while using the IC-PLL.

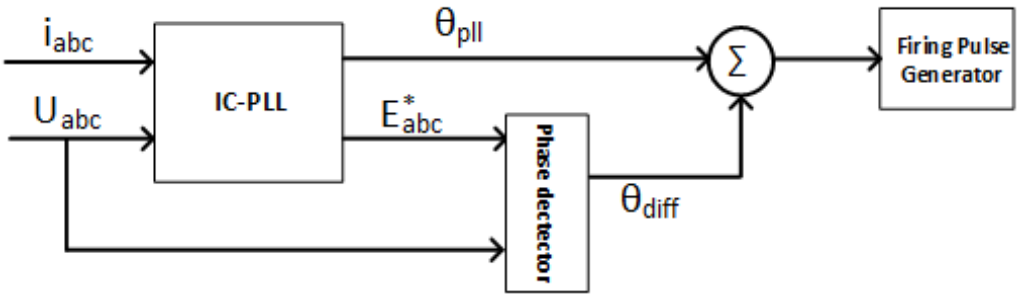


Figure 3.5: Angle Correction Module

As shown in Fig. 3.5, the IC-PLL outputs the estimated three phase voltage e^*_{abc} and the phase angle θ_{PLL} to the angle correction module. The instantaneous phase difference is generated from a phase detector by comparing the e^*_{abc} with the PCC voltage u_{abc}. By adding the instantaneous phase difference, the corrected phase angle is obtained and delivered to the firing pulse generator.

Chapter 4

FPGA Features and Programming Techniques

The Field Programmable Gate Array(FPGA) was used in this thesis to implement the IC-PLL in hardware. This was then connected to the HVdc power system as modelled on a Real-Time Digital Simulator(RTDS). This chapter introduces some key features and programming techniques of the Xilinx FPGA which are relevant to the actual implementation of the IC-PLL described later in Chapter 5. The architecture of Xilinx FPGAs, design flow and programming languages are discussed explicitly in Appendix E.

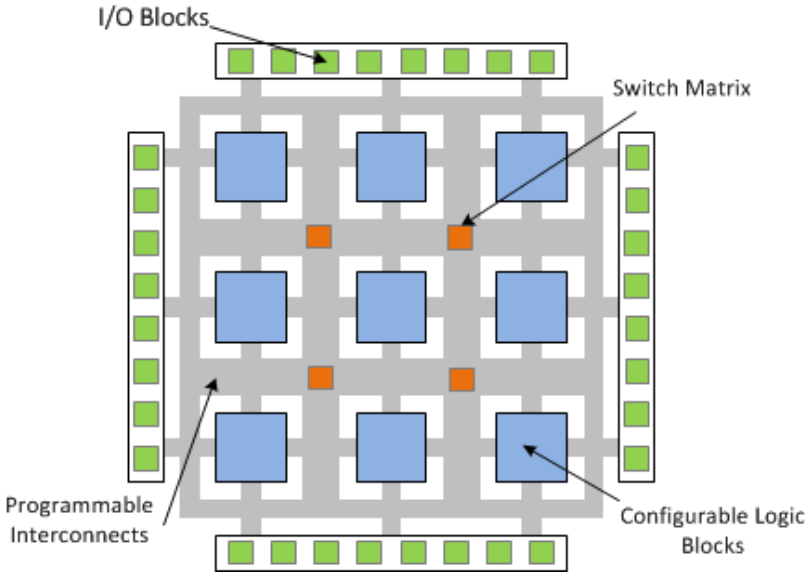


Figure 4.1: Basic Structure of FPGA

Fig. 4.1 shows three basic components inside the FPGA: The Configurable Loigc Blocks(CLBs), the I/O Blocks(IOBs) and the programmable interconnects.

- **CLBs**: Main logic resource inside the FPGA for implementing logic functions.
- **IOBs**: Responsible for interfacing the FPGA with external devices, input and output functions are implemented with IOBs.
- **Programmable Interconnects**: The programmable interconnects are wires that connect IOBs and CLBs together by configuring the switch matrix.

Besides these basic components, the Artix-7 FPGA also contains other components such as DSP slices, Block RAMs, configurable analog interface etc.(Appendix E). All these elements work together to provide the FPGA with a reconfiguration ability and allows the FPGA to perform different combinational logics.

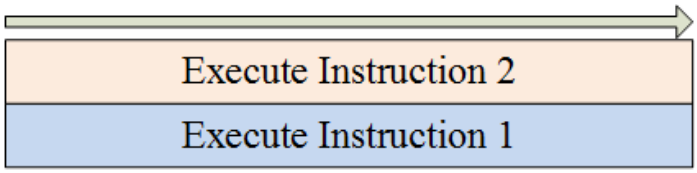
4.1 Concurrent and Sequential Operation

In contrast with CPUs or microprocessors, the hard-wired natural of the FPGA allows FPGA perform both sequential and concurrent operations. For CPUs or microprocessors, instructions are executed sequentially.

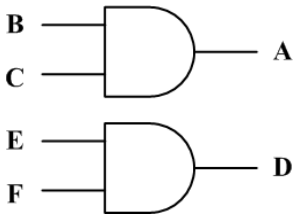
Instruction number	C code	VHDL code
1	`A = B & C`	`A <= B and C`
2	`D = E & F`	`D <= E and F`

Taking the above instructions as an example, the **AND** operation is performed on operands 'B' and 'C', 'E' and 'F', and results are assigned to 'A' and 'D' respectively. The basic procedure for the CPU or microprocessor to execute instructions is to repeat the instruction cycle which contains four steps: fetch the instruction, decode the instruction, read the memory and execute the instruction. Therefore, to execute instruction 1 and 2, the CPU has to follow the steps as shown in Fig. 4.2a. However, in the FPGA implementation, these two instructions are synthesised and realized by hardware resources inside FPGA (CLBs, IOBs, etc.) which can be seen as two AND gates as shown in Fig. 4.2c. Therefore, compared with the CPU execution, there are no fetch and decode steps in the FPGA implementation, two instructions are executed at the same time due to the natural of the hardware circuit.

(a) CPU Excution



(b) FPGA Excution



(c) FPGA RTL Function

Figure 4.2: Sequential and Concurrent Operation

4.2 Data Representation in FPGA

Calculations that involve real numbers are commonly seen in digital systems. The two most widely used data formats are the fixed point format and the floating point format. For the fixed point format, the integer and fractional parts are separated by the decimal point. The accuracy of the fixed point format is determined by the Least Significant Bit(LSB). On the other hand, the floating format represents the number using *Sign*, *Mantissa* and *Exponent*. Compared with the fixed point format, the floating point format is more flexible and has a much larger dynamic range. However, the fixed point calculation is faster than the floating point calculation, and it requires

less hardware resources to implement compared to floating point calculation. In this thesis, the fixed point format is employed due to following reasons:

- Normally,the fixed point calculation is faster than the floating point calculation which results in potentially higher processing speed for real-time applications.
- The fixed point calculation requires less hardware resources and is easier to implement on the FPGA, which makes it suitable for the FPGA board used in this thesis.
- The IC-PLL implementation on the FPGA is based on a per-unit system. Thus, the dynamic range and accuracy of fixed point format would be sufficient.

The general representation of fixed point is shown in Fig. 4.3. The number x can be calculated using equation 4.1.

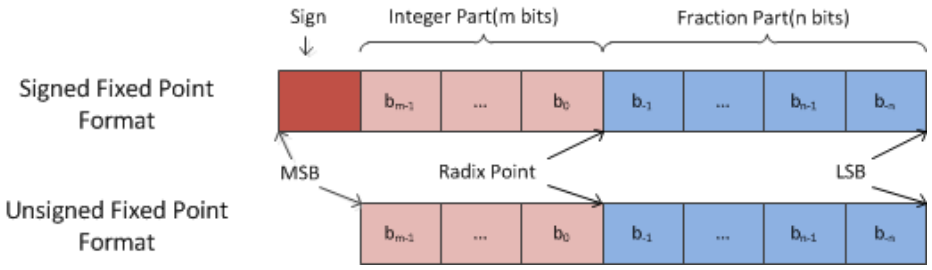


Figure 4.3: Fixed Point Data Representation

$$x = -2^m * Sign + \sum_{i=-n}^{m-1} b_i * 2^i \tag{4.1}$$

One thing to notice is that, since the location of the radix point does not change in fixed point format, a loss of precision may occur when the calculation result has more bits than its operands. In order to maintain the precision of the calculated result, certain sizing rules must be followed.

In this thesis, the *ieee.fixed_pkg* package available in VHDL libraries is used to perform the fixed point calculation and its corresponding sizing rules are shown in Table 4.1. However, if we want to keep the bit length of the result as same as the operands, then truncation or rounding operations must be performed on the calculated result. In this case, the choice of which bit to keep is importance since it can have significant impact on the data precision.

Table 4.1: Fixed Point Sizing Rules[24]

Operation	Result Range
$A + B$	$Max(A'left, B'left)$+1 downto $Min(A'right, B'right)$
A - B	$Max(A'left, B'left)$+1 downto $Min(A'right, B'right)$
A * B	$A'left$+$B'left$+1 downto $A'right$+$B'right$
A rem B	$Min(A'left, B'left)$ downto $Min(A'right, B'right)$
Signed A / B	$A'left$-$B'right$+1 downto $A'right$-$B'left$
Unsigned A / B	$A'left$-$B'right$ downto $A'right$-$B'left$-1

Two data types can be defined in the *ieee.fixed_pkg* package which are the $sfixed$ and $ufixed$ types. The $sfixed$ type is for signed fixed point and $ufixed$ type for unsigned fixed point. The code below shows an example of declaring signed and unsigned fixed point signal. The unsigned_number has 'a' decimal bits and 'b' fractional bits. The signed_number has 'a+1' decimal bits and 'b' fractional bits where the Most Significant Bit(MSB) indicates the sign of the number.

```
signal signed_nubmer :  sfixed(a downto b);
signal unsigned_nubmer :  ufixed(a downto b);
```

Chapter 5

FPGA Implementation of IC-PLL

The hardware implementation of the IC-PLL was done using a Field Programmable Gate Array(FPGA). The nature of the FPGA allows the IC-PLL implementation with a fast execution time and higher reliability compared with the microprocessor implementation. In this thesis, an actual IC-PLL is constructed on the *Nexys 4 DDR Artix-7* FPGA board due to following reasons:

- The FPGA I/O flexibility allows us to define I/O functions for specific use. In this implementation, the IC-PLL interfaces with external devices through at least 12 I/Os (6 inputs and 6 outputs), the number of specified I/Os required is unlikely to be handled by a single microprocessor but can be easily handled by a FPGA.
- The parallel execution ability of the FPGA allows it to run complex algorithms in a faster way and increases the throughput of the system as compared with microprocessors.
- The hard-wired structure of the FPGA makes it more reliable. Unlike mi-

croprocessors, the FPGA realizes different functions based on interconnections between logic blocks and I/Os, no operating system or software is involved once the FPGA is configured.

- The FPGA technique provides fast prototyping capabilities.

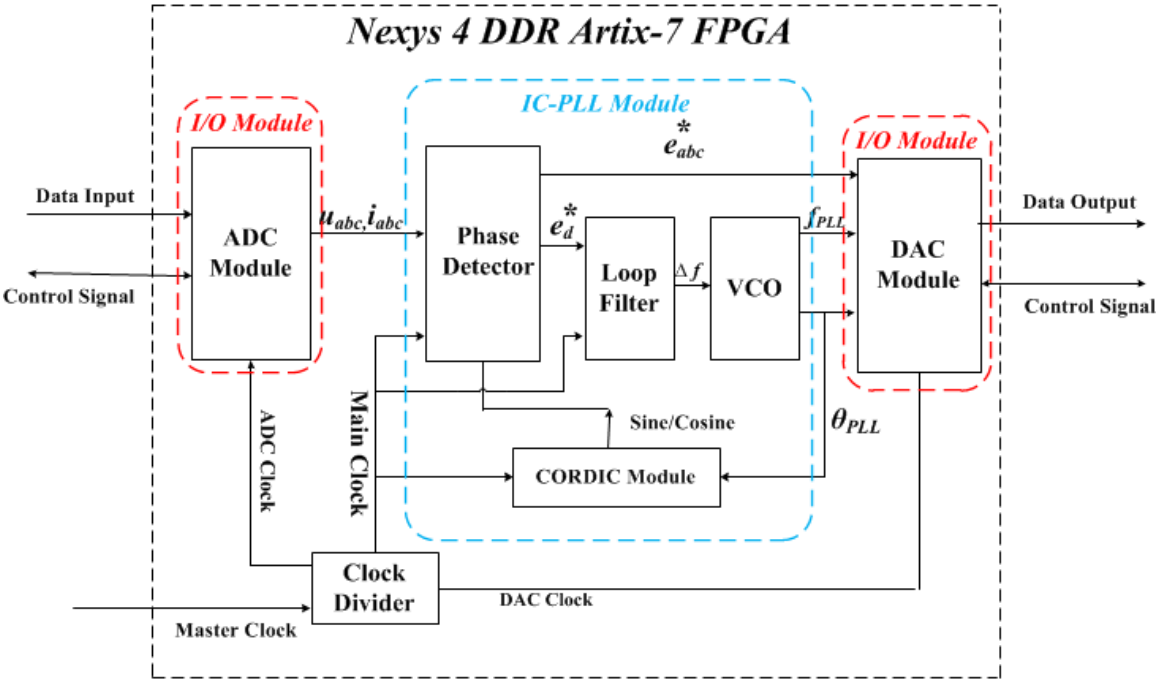


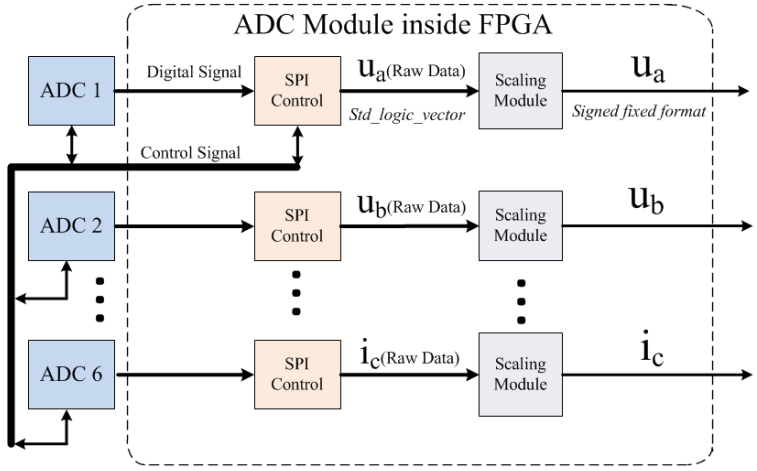
Figure 5.1: IC-PLL Structure Overview

As shown in Fig. 5.1, the FPGA implementation of the IC-PLL consists of two main modules: the I/O interface module and the IC-PLL module. The I/O interface module is comprised of DAC module and ADC module which are designed to communicate with external devices(ADCs and DACs) to acquire/output necessary data for the IC-PLL. In the IC-PLL module, the Phase Detector(PD) block, Loop Filter(LF) block, Voltage-Controlled Oscillator(VCO) block and COordinate Rotation DIgital Computer(CORDIC) block collaborate with each other to process the acquired data

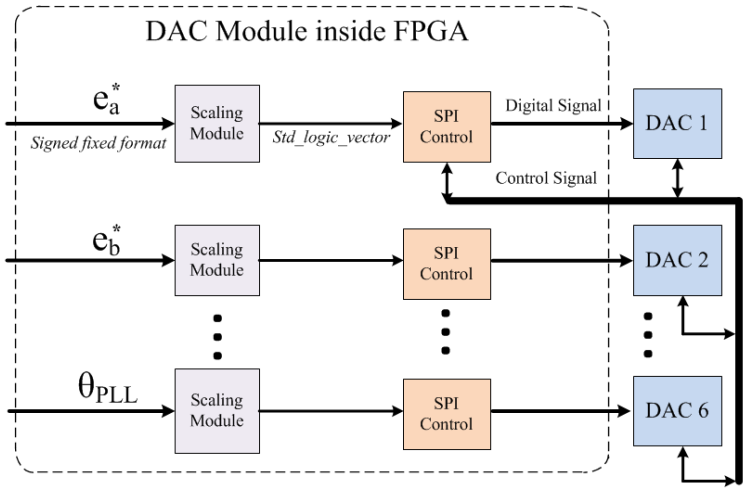
and perform the IC-PLL computations. In this implementation, all the calculations are done in per-unit (p.u) and the fixed point format is applied in most of the modules to make implementation simpler as compared to the floating point format. Moreover, the execution speed of using fixed point format is faster than floating point format. Six input signals (u_{abc} and i_{abc}) are sampled at the same time due to the parallel execution of the ADC module, and the sampled data are converted into p.u values with respect to the LCC-HVdc system parameters. In the PD block, equation 3.19 is applied to compensate the voltage drop across the ac network impedance as mentioned in section 3.3. Then the LF block and VCO block work together to generate the corresponding phase angle θ_{PLL} which contains the fundamental frequency and phase information of the estimated voltage e^*_{abc}, after that, necessary signals of the IC-PLL are generated by the DAC module for control, debug and observation purposes. The CORDIC module is implemented to generate the sine and cosine values which are used for the $abc - dq0(dq0 - abc)$ transformation. The clock module generates clocks with different frequencies according to different modules. All clocks are synchronized, i.e., all clock signals are derived from the same internal clock, which is also the fastest clock. Thus the transition of a slow clock only happens when the internal clock makes a transition. The reason for having different clocks is that each module takes different number of clock cycles to finish its computation process. For example, the sine and cosine values generated by the CORDIC module should be ready to be used by the PD block before the next PCC data arrives, which means the clock frequency of the CORDIC module needs to be higher than that of the clock goes into the PD block. The maximum clock frequency allowed in the module depends

on the latency of other modules. In this thesis, all modules are written based on the VHSIC Hardware Description Language (VHDL).

5.1 I/O Interface Module Design



(a) ADC Module



(b) DAC Module

Figure 5.2: I/O Interface Module

The I/O interface module consists of an ADC module and a DAC module which are designed to sample and pre-process the data so they can be used for IC-PLL calculation. The detailed structures of the ADC and DAC module are shown in Fig. 5.2.

Taking the ADC module as an example, the ADC module contains two sub-modules which are the Serial Peripheral Interface(SPI) control module and the scaling module. The function of SPI control module is to generate control signals to acquire three phase voltage and current data through SPI interface. The sampling frequency is set inside this module. Furthermore, six ADCs share the same SPI control signals(eg. MOSI, SCLK, CS) which means three phase voltage and current signals are sampled at the same time with the same sampling rate. ADCs on the FPGA side have a voltage input range from 0 to 5V which represent ±2 p.u for voltage signals and ±4 p.u for current signals. To satisfy the ADC requirement, dc offset are added to the voltage and current signals before passing them to the ADCs. The function of the scaling module is to remove the dc offset once the data have been sampled and to convert the raw data(*std_logic_vector* format) into corresponding p.u values($sfixed$ format). Inside the FPGA, the sampled data are represented using 10 bits binary code with data range from 0 to 1023. In the ideal case, voltage and current signals are processed according to equation 5.1 and 5.2 respectively[25].

$$u_{p.u} = (\frac{1024 \times V_{in}}{V_{ref}} \times \frac{1}{256}) - 2.0 \tag{5.1}$$

$$i_{p.u} = (\frac{1024 \times V_{in}}{V_{ref}} \times \frac{1}{128}) - 4.0 \tag{5.2}$$

where V_{ref} is 5V and V_{in} is the analog signal goes into the ADC

The scaling and SPI control block inside DAC module are arranged differently from ADC module but have similar functions. As shown in Fig.5.2b, the calculated data are converted into required format for SPI control block, and dc offset are added to some of the data. The SPI control block generates control signals and passes data to DACs. Six analog signals are outputted from the FPGA-based IC-PLL for control and observation purposes. The representation of these output signals with respect to DAC output range of 0 to 5V are shown in table 5.1.

Table 5.1: FPGA-based IC-PLL Output Signal Specification

Signal Name	Signal Description	Represented Values
e_a	Estimated Phase A Voltage	-2 to 2 p.u
e_b	Estimated Phase B Voltage	-2 to 2 p.u
e_c	Estimated Phase C Voltage	-2 to 2 p.u
e_d	Phase Error	-2 to 2 p.u
f_{PLL}	Frequency Output Signal	58.5 to 61.5 Hz
θ_{PLL}	PLL Output Angle	0 to 2π Radians

5.2 IC-PLL Module Design

Fig. 5.3 shows main functions that are implemented on the FPGA. In this implementation, the Phase Detector (PD) function is achieved by performing the Park transformation together with the voltage estimation calculation, then the output is passed through a Low-Pass Filter(LPF) to remove ripple. A Proportional-Integral(PI) controller is adopted as the Loop Filter(LF) which eliminates the steady-state phase(and hence also frequency) error. The VCO takes the signal output from the PI controller ($f_o + \Delta f$) and generates an angle reference θ_{PLL} which contains frequency and

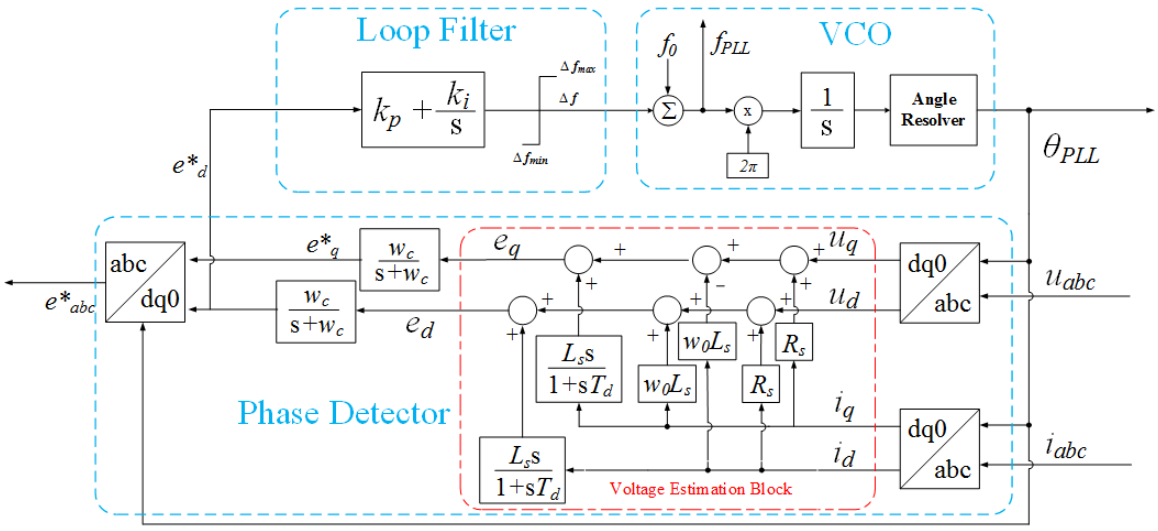


Figure 5.3: IC-PLL Module

phase information of the input signal.

The FPGA module includes the PI controller, VCO, the Park and inverse Park transformation, LPF as well as the derivative function which are present in the s-domain. However, due to the discrete-time form as required in any digital implementation, it is necessary to implement the IC-PLL in Fig. 5.3 in its equivalent discrete-time form.

5.2.1 Phase Detector Implementation

5.2.1.1 LPF and Derivative Function Implementation

A pure derivative function is rarely implemented in the control system due to its propensity to amplify high frequency noise. The output of a derivative function is proportional to the rate of change of the input signal, however, in real application, the negative effect of a derivative function is that it amplifies the high frequency

noise exist in the input signal which could confuse the control system. To avoid this problem, a wash-out filter is used instead of the derivative function. The wash-out function in the s-domain is written as in equation 5.3:

$$H_d(s) = \frac{L_s s}{1 + sT_d} \tag{5.3}$$

where L_s is the Thevenin equivalent inductance of the ac network in p.u and T_d is the wash-out filter time constant. Note that the numerator of $H_d(s)$ is a derivative term and the denominator is a first order LPF which reduces the gain at high frequencies, thereby improving noise immunity. Not that for signals of frequencies significantly lower than the cut off frequency of the LPF, the block behaves as a derivative.

The performance of the derivative function can be adjusted by choosing different T_d values. The larger the T_d value, the better the noise immunity, and the smaller the T_d value, the smaller the bandwidth and hence the closer agreement to the pure derivative function. In this thesis, T_d is set to 0.001. Furthermore, a first order LPF is used to remove the ripple as mentioned in section 3.3. The LPF expression in the s-domain is:

$$H_{LPF}(s) = \frac{\omega_c}{s + \omega_c} \tag{5.4}$$

For simplicity, the LPF and derivative functions are implemented on the FPGA using one module which is called the transfer function module. Consider a typical first order s-domain transfer function form as in equation 5.5:

$$H_{TF}(s) = \frac{Y(s)}{X(s)} = \frac{As + B}{Cs + D} \tag{5.5}$$

To convert to a discrete representation, we use the bilinear transformation and

substitute s with $\frac{2}{\Delta T} \cdot \frac{z-1}{z+1}$ we have:

$$H_{TF}(z) = \frac{Y(z)}{X(z)} = \frac{(2A + B \cdot \Delta T)z - (2A - B \cdot \Delta T)}{(2C + D \cdot \Delta T)z - (2C - D \cdot \Delta T)} \tag{5.6}$$

Multiplying the denominator and numerator of equation 5.6 by z^{-1} and rearranging it, we get equation 5.7

$$Y(z)\cdot(2C+D\cdot\Delta T)-z^{-1}Y(z)\cdot(2C+D\cdot\Delta T) = X(z)\cdot(2A+B\cdot\Delta T)-z^{-1}X(z)\cdot(2A-B\cdot\Delta T) \tag{5.7}$$

Realizing that $X[z]z^{-1}$ is the z transform of $x[n-1]$ and $Y[z]z^{-1}$ is the z transform of y[n-1], we have discrete time function of equation 5.6 below:

$$y[n] = C_A x[n] + C_B x[n-1] + C_C y[n-1] \tag{5.8}$$

$$where \quad C_A = \frac{2A+B\cdot\Delta T}{2C+D\cdot\Delta T} \quad C_B = -\frac{2A-B\cdot\Delta T}{2C+D\cdot\Delta T} \quad C_C = \frac{2C-D\cdot\Delta T}{2C+D\cdot\Delta T}$$

The quation 5.8 gives the value of the output $y[n]$ in the present time step from the past and current inputs $x[n], x[n-1]$ and past history of the output $y[n-1]$. It can therefore be directly implemented on the FPGA as shown in Fig. 5.4.

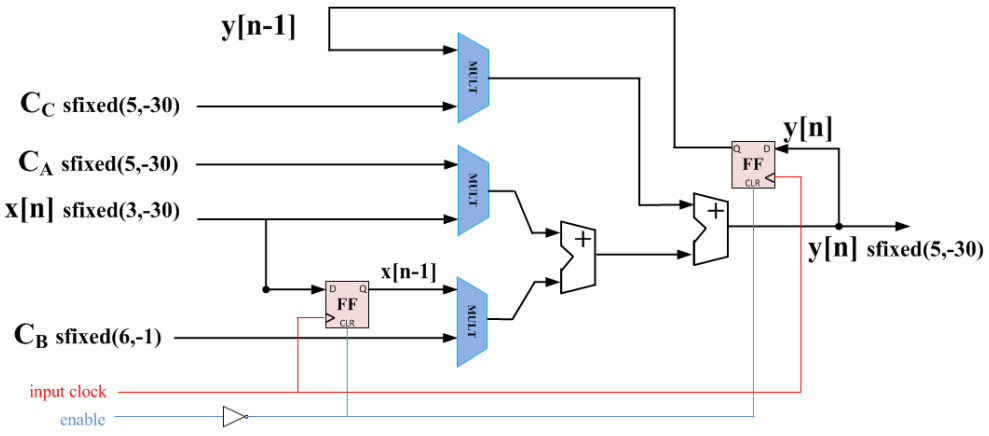


Figure 5.4: Hardware Implementation of Transfer Function Module

The value of C_A, C_B and C_C are pre-calculated and stored in the FPGA memory. The *enable* signal activates the transfer function calculation process and the *input clock* signal controls the output timing. The computation process is divided into 3 stages: During the first stage, $C_A x[n]$, $C_B x[n-1]$ and $C_C y[n-1]$ are calculated concurrently, then $C_A x[n] + C_B x[n-1]$ is calculated, and the last stage, $y[n]$ value is obtained. Unlike sequential processors, the concurrent calculation speeds up implementation considerably, the throughput of the FPGA-based IC-PLL is discussed in section 6.2.2.3.

5.2.1.2 Voltage Estimation Block Implementation

The voltage estimation block calculates the voltage drop across the ac network impedance and use that to generate an estimated voltage which is potentially less distorted as compared to the PCC voltage. Recall from section 3.3, the equation to calculate the estimated voltage is:

$$\begin{bmatrix} e_d \\ e_q \end{bmatrix} = \begin{bmatrix} u_d \\ u_q \end{bmatrix} + L_s \frac{d}{dt} \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \begin{bmatrix} R_s & \omega_0 L \\ -\omega_0 L & R_s \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} \tag{3.19}$$

The voltage estimation block involves addition, subtraction, multiplication as well as derivative calculation. The derivative function is calculated using the transfer function module mentioned in section 5.2.1.1. Estimated voltages e_d and e_q are calculated concurrently.

5.2.1.3 Coordinate Rotation Digital Computer (CORDIC) algorithm

Sine and cosine calculations are required in performing Park and inverse Park transformation($abc-dq/dq-abc$). In a digital system, one fast way of computing sine

and cosine functions is using direct look-up-table(LUT) method. The sine or cosine values are pre-calculated and stored in the system memory, the output is computed by a single read operation regardless of the complexity of the function, however, the disadvantage of this method is that the memory usage increases dramatically as the precision requirement of output increases which makes this method not suitable for the FPGA implementation.

In the FPGA, the sine and cosine functions are implemented using the Coordinate Rotation Digital Computer (CORDIC) algorithm [26]. The CORDIC algorithm is a iterative method which requires less hardware resources and has higher precision compared to the direct LUT method[11][27]. The basic iteration formula of the CORDIC algorithm is shown below [28]:

$$x_{i+1} = x_i - y_i d_i 2^{-i} \tag{5.9}$$

$$y_{i+1} = y_i + x_i d_i 2^{-i} \tag{5.10}$$

$$z_{i+1} = z_i - d_i \tan^{-1}(2^{-i}) \tag{5.11}$$

$$i = i + 1 \tag{5.12}$$

where d_i is the rotation direction (-1 or 1).

It can be seen from above equations that the CORDIC algorithm requires only addition, subtraction, bit-shift ($d_i 2^{-i}$) and table look-ups ($\tan^{-1}(2^{-i})$) which makes it easier to be implemented in hardware.

Fig. 5.5 illustrates an example of computing sine and cosine values using the CORDIC algorithm. Initially, the value of x_i and y_i are set to 1 and 0 respectively and z_i is equal to the input angle ϕ, during the first iteration, the vector v_i rotates

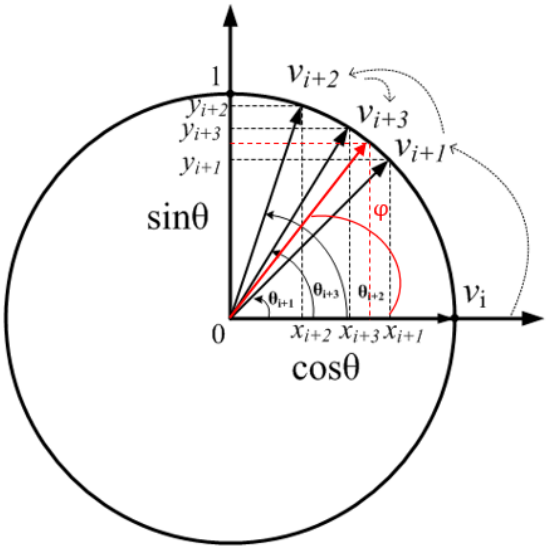


Figure 5.5: CORDIC Rotation Mode

$\tan^{-1}(2^{-i})$ degrees to v_{i+1}, if z_{i+1} is greater than 0 (which means θ_i is smaller than ϕ) then d is set to 1 (next rotation count-clockwise), if z_{i+1} is smaller than 0, d is set to -1. In the second iteration, the vector v_{i+1} rotates to v_{i+2} with angle of $\tan^{-1}(2^{-(i+1)})$ degrees and we repeat the same steps as in the first iteration. This process will repeat n times till the value of z_{i+n} is equal to 0 or within acceptable range. Then the value of x_{i+n} and y_{i+n} are the cosine and sine values of the input angle ϕ.

The accuracy of a fixed-point implementation of the CORDIC algorithm is affected by both the number of iterations and the number of fraction bits of the output value. Increasing number of iterations will reduce the angle approximation error and increase the number of fraction bits will reduce the round off error. These two errors become very close when number of iterations approaches to number of fraction bits, therefore, it is meaningless to have number of iterations greater than number of fraction bits[29][30]. In this thesis, number of fraction bits is 30, thus, the number of

iterations is set to 30, which corresponds to a latency of 36 clock cycle[28]. For the FPGA board with a 100MHz system clock, it takes 0.36 μs to calculate one sine or cosine value.

It should be mentioned that the CORDIC algorithm uses fixed point data presentation (in *std_logic_vector* form as discussed in section 4.2) and the input angle range is from π to $-\pi$, however, the PLL output value is from 0 to 2π, in order to interface with the CORDIC module, an angle conversion module has to be implemented to convert θ_{PLL} into acceptable range of the CORDIC module. In addition to that, the output of the CORDIC module also needs to be converted back into $sfixed$ format so other module can interface with the CORDIC module.

5.2.1.4 Park and Inverse Park Transformation Implementation

The 3 by 3 matrix of the Park and inverse Park transformation are listed in Appendix C.1 and C.2. From Fig. 5.3, only x_a,x_b,x_c, x_d and x_q values are required. To reduce the hardware resource usage on the FPGA, the zero sequence term is ignored as it is not relevant in this implementation. The Park transformation equations are shown below:

$$x_d = \frac{2}{3}\Big\{ \cos(\theta)x_a + \cos(\theta - \frac{2\pi}{3})x_b + \cos(\theta + \frac{2\pi}{3})x_c\Big\} \tag{5.13}$$

$$x_q = \frac{2}{3}\Big\{ \sin(\theta)x_a + \sin(\theta - \frac{2\pi}{3})x_b + \sin(\theta + \frac{2\pi}{3})x_c\Big\} \tag{5.14}$$

Fig. 5.6 shows the FPGA implementation of the Park transformation. Constant values ($\frac{2\pi}{3}$,-$\frac{2\pi}{3}$,$\frac{2}{3}$) are pre-stored on the FPGA. Calculation of equation 5.13 and 5.14 is divided into several steps: first of all, the value of $\theta + \frac{2\pi}{3}, \theta - \frac{2\pi}{3}$ are calculated. And then, θ, $\theta + \frac{2\pi}{3}$ are passed to CORDIC modules to generate their sine and cosine

values. These values are then multiplied by x_a, x_b or x_c according to equation 5.13 and 5.14. After that, $\cos(\theta)x_a + \cos(\theta - \frac{2\pi}{3})x_b + \cos(\theta + \frac{2\pi}{3})x_c$ and $\sin(\theta)x_a + \sin(\theta - \frac{2\pi}{3})x_b + \sin(\theta + \frac{2\pi}{3})x_c$ are calculated concurrently. Finally, the x_d and x_q are obtained by multiplying the results with $\frac{2}{3}$.

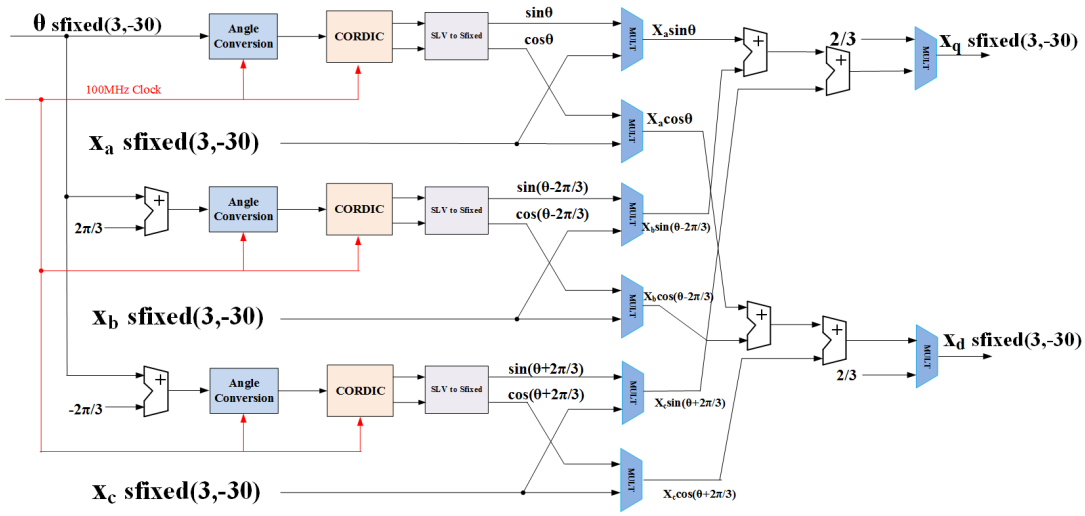


Figure 5.6: FPGA Implementation of the Park Transformation

Alternatively, the Park transformation can be implemented indirectly using the $abc-\alpha\beta-dq$ transformation as mentioned in 3.2[31]. Firstly, the three phase quantities x_a,x_b and x_c are projected onto two axis stationary frame using $abc-\alpha\beta$ (Clarke) transformation, then these two quantities are projected onto the rotating frame using the $\alpha\beta - dq$ transformation. As discussed in section 3.2, the Clarke transformation matrix is

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} \tag{3.16}$$

which can be rewritten as:

$$x_\alpha = x_a^{'} - \frac{1}{2}(x_b^{'} + x_c^{'}) \tag{5.15}$$

$$x_\beta = \frac{\sqrt{3}}{2}(x_b^{'} - x_c^{'}) \tag{5.16}$$

$$\text{where } x_a^{'} = \tfrac{2}{3}x_a,\ x_b^{'} = \tfrac{2}{3}x_b \text{ and } x_c^{'} = \tfrac{2}{3}x_c$$

Then, x_d and x_q values can be obtained by performing $\alpha\beta - dq$ transformation using equations below:

$$x_d = \cos(\theta)x_\alpha + \sin(\theta)x_\beta \tag{5.17}$$

$$x_q = \sin(\theta)x_\alpha - \cos(\theta)x_\beta \tag{5.18}$$

As compared to the direct Park transformation implementation mentioned before, the indirect Park transformation requires only calculation of $\sin(\theta)$ and $\cos(\theta)$, thus, with this implementation, only one CORDIC module is required. Furthermore, the multiplication by $\frac{1}{2}$ can be done using a shifter.

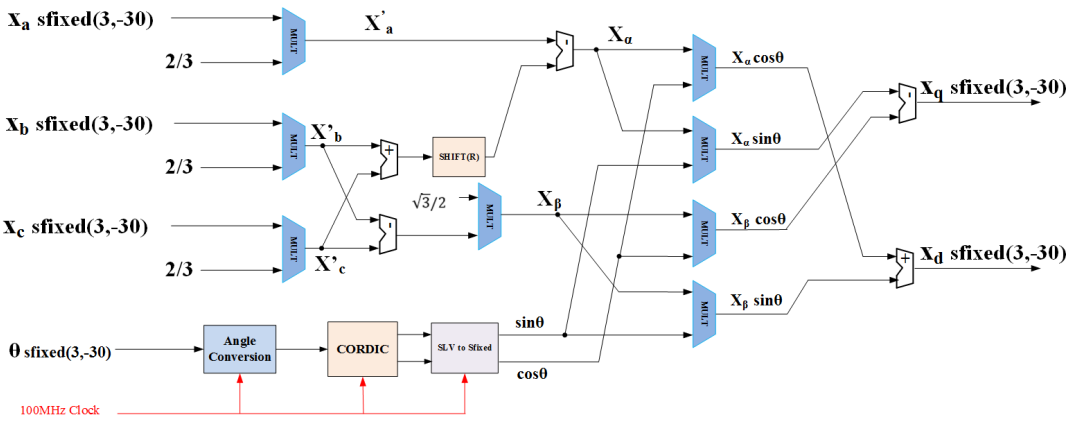


Figure 5.7: FPGA Implementation of the indirect Park Transformation

Fig. 5.7 shows the FPGA implementation of the indirect Park transformation. Similar to the direct Park transformation implementation, the computation of equation 5.15 to 5.18 is divided into several steps: Firstly, $x_a^{'}$, $x_b^{'}$ and $x_c^{'}$ are calculated by multiplying input signals x_a, x_b and x_c with $\frac{2}{3}$. Then x_α and x_β can be obtained using adder,subtracter, multiplier and shifter block. Afterwards, $x_\alpha \cos(\theta)$, $x_\alpha \sin(\theta)$,$x_\beta \cos(\theta)$ and $x_\beta \cos(\theta)$ are calculated in parallel. Finally, x_d and x_q are obtained using adder/subtracter.

Table 5.2: FPGA Resource Usage of direct and indirect Park Transformation

	LUT	FF	BRAM	URAM	DSP Slice
direct Park transformation	13045	11820	0	0	40
indirect Park transformation	5534	4043	0	0	48

The FPGA resource usage for the direct and indirect Park transformation approaches is shown in Table 5.2. It can be seen that the direct Park transformation takes more Look-Up Table(LUT) and flip-flop (FF) resources but less Digital Signal Processing (DSP) slices as compared to the indirect Park transformation. Therefore, the choice of using direct or indirect Park transformation mainly depends on the LUT& FF or DSP resource available on the FPGA. However, if we can rescale x_a,x_b and x_c by $\frac{2}{3}$ before they are sampled through the Analog-to-Digital Converters(eg. using an analog voltage divider), then these three multipliers on the left of Fig. 5.7, are then no longer needed, and this will reduce the use of DSP slices. In that case, it is suggested to use indirect Park transformation since it uses less hardware resources.

Fig. 5.8 shows the timing diagram of the Park transformation with a computation

time step of 2 μs. The input signals x_a,x_b,x_c and θ are concurrently updated and computed every $2\mu s$. The computation process is triggered by the rising edge of the 100MHz clock, the sine and cosine values are calculated after 36 clock cycles, then the output signals x_d and x_q are obtained.

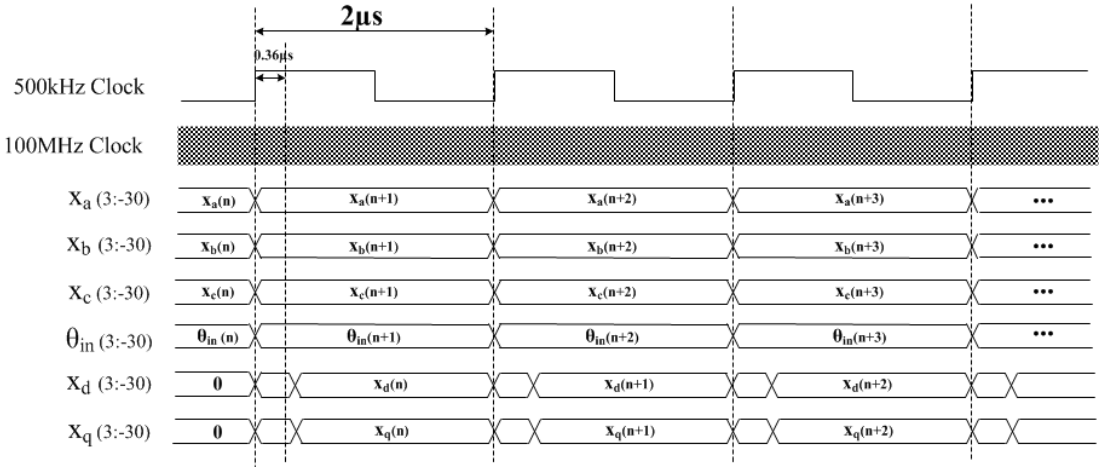


Figure 5.8: Timing Diagram of the Park Transformation

The inverse Park transformation converts quantities in the rotating *dq* frame into the *abc* reference frame. The transformation equations are shown below:

$$x_a = \cos(\theta)x_d + \sin(\theta)x_q \tag{5.19}$$

$$x_b = \cos(\theta - \frac{2\pi}{3})x_d + \sin(\theta - \frac{2\pi}{3})x_q \tag{5.20}$$

$$x_c = \cos(\theta + \frac{2\pi}{3})x_d + \sin(\theta + \frac{2\pi}{3})x_q \tag{5.21}$$

The hardware implementation of the inverse Park transformation is shown in Fig. 5.9. These equations are implemented similar to the Park transformation. Three inputs θ, x_d, x_q are passed to the inverse Park transformation module, the value of

$\theta - \frac{2\pi}{3}$ and $\theta - \frac{2\pi}{3}$ are calculated firstly, after that, sine and cosine values of these angles are computed. Then x_a, x_b and x_c are calculated in parallel.

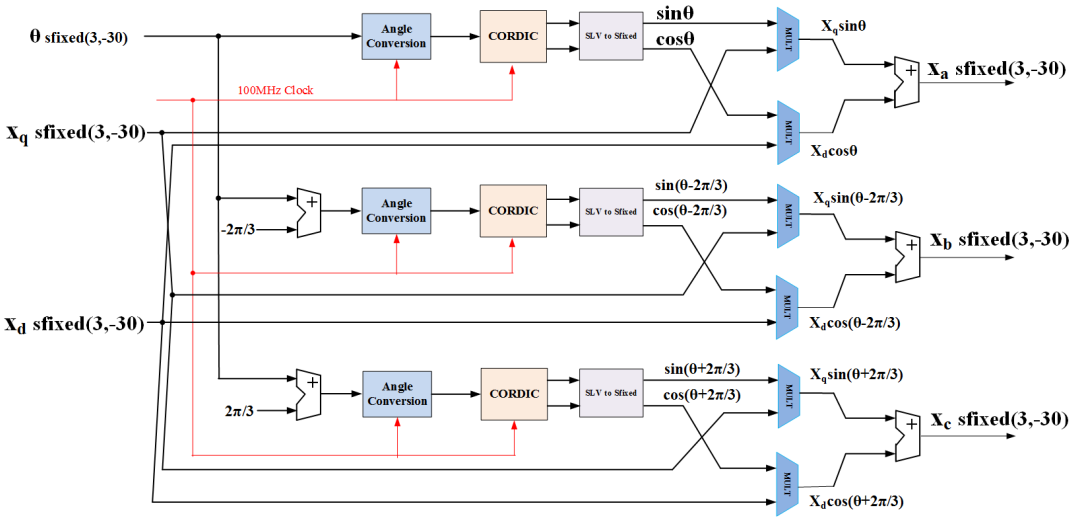


Figure 5.9: Hardware Diagram of the Inverse Park Transformation

5.2.2 Loop Filter Implementation

The Loop Filter (LF) determines the dynamic response of the PLL and filters the high frequency components contained in the Phase Detector(PD) output signal. In the IC-PLL implementation, the LF is realized using a PI controller whose s-domain expression is:

$$G_{PI}(s) = k_p + \frac{k_i}{s} \tag{5.22}$$

After performing the bilinear transformation and inverse z-transformation, the discrete time equation of the PI controller implemented on the FPGA is expressed as:

$$y[n] = (k_p + \frac{k_i \Delta t}{2})x[n] - (k_p - \frac{k_i \Delta t}{2})x[n-1] + y[n-1] \tag{5.23}$$

Fig. 5.10 describes the simplified hardware implementation of the PI controller based on equation 5.23.

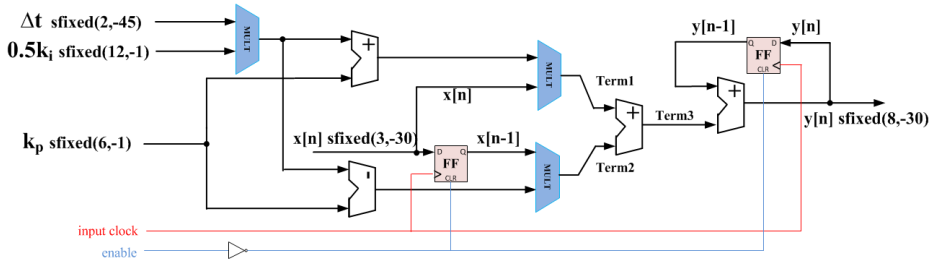


Figure 5.10: Hardware Implementation of the PI Controller

The LF module takes the phase error e_d^* generated from the PD module and outputs the frequency deviation Δf. Constant values(Δt, k_i and k_p) are pre-stored in FPGA memory. Since the FPGA-based IC-PLL operates in real-time, to obtain correct PI output result at right time instance, the Δt must be equal to the input clock period(eg. $2\mu s$ time step corresponds to $500kHz$ input clock signal) in this case. The $x[n-1]$ and $y[n-1]$ values are stored in Flip-Flops(FFs) which are triggered by the rising edge of the input clock. As shown in Fig.5.3, the user can set the maximum and minimum limit on the PI Controller output. However, data overflow may happen when using fixed point format, which may lead to unpredictable result. For example, if $y[n-1]$ goes beyond the data range, the incorrect $y[n-1]$ value will be stored(i.e., the most significant bit of the result will be dropped), and the rest of calculation results will be wrong. To avoid that problem, the output is made to saturate at the minimum or maximum value if the limits of the data range are exceeded.

Fig. 5.11 shows the flow chart of the PI controller implementation. The PI controller calculation process(equation 5.23) is divided into serial stages which contains both concurrent and sequential execution. The concurrent execution blocks are identified with a blue color. The calculation process is triggered by the rising edge of the input clock signal once the IC-PLL is enabled. The calculation of equation 5.23 is done through stage 1 to 4. In stage 5, the current $y[n]$ value ($CurrentY$) is compared with the PLL frequency limit. The $CurrentY$ value is assigned to the PI controller output $y[n]$ if the $CurrentY$ value is within the PLL frequency limit, otherwise, the $CurrentY$ value is set to the PLL frequency limit. The $y[n-1]$ value is set slightly lower or higher than the data range limitation when $y[n-1]$ approaches to the fixed point data range limitation, otherwise, the $y[n-1]$ value remains the same. At the beginning(t=0), the $y[n]$, $x[n]$, $y[n-1]$ and other internal variables are set to 0.

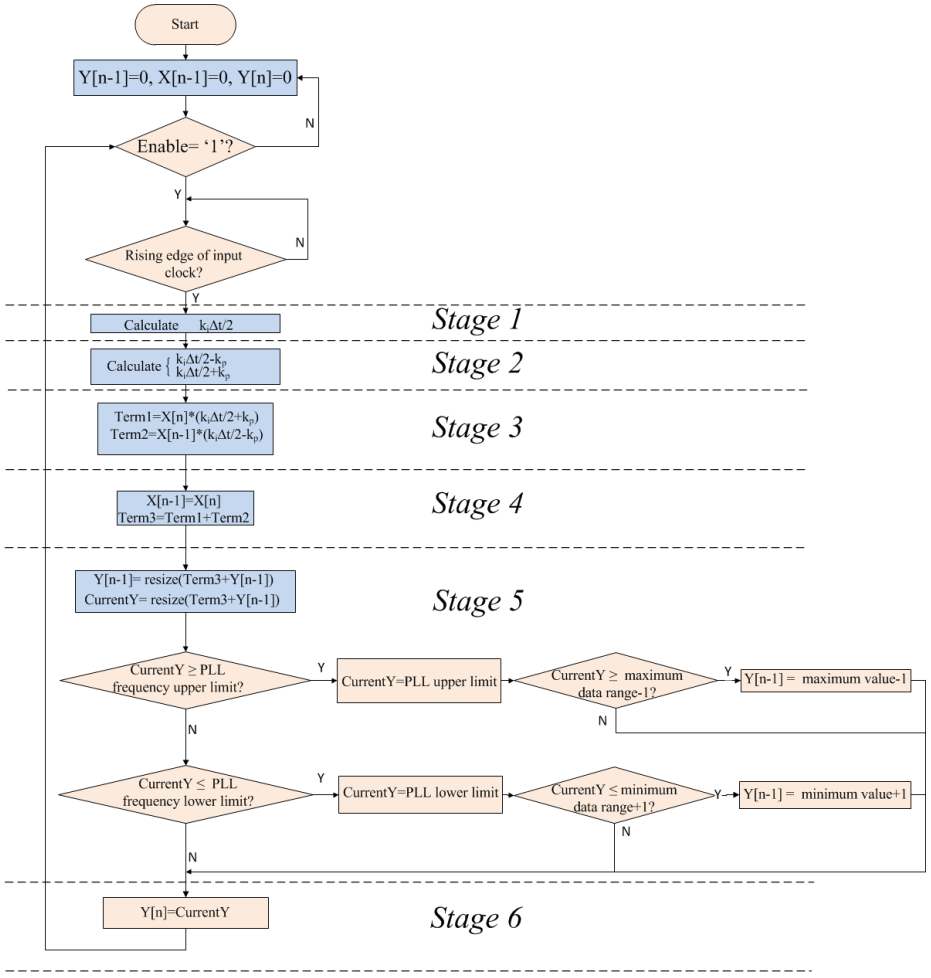


Figure 5.11: Flow Chart of the PI Controller Implementation

5.2.3 Voltage-Controlled Oscillator Implementation

The VCO generates an output signal whose frequency is proportional to the input signal magnitude(which is assumed to vary slowly with time). Using rectangular integration, the discrete time function to implement the integral block is shown in equation 5.24. The rectangular integration is adequate because input varies slowly.

$$\theta[n] = \theta[n-1] + \omega[n]\Delta t \tag{5.24}$$

The FPGA implementation of VCO block is shown below in Fig. 5.12: The calculation has four stages: Initially, the input frequency $f_o+\Delta f$ is calculated, then it is converted to angular frequency $\omega[n]$ by multiplying $f_o + \Delta f$ with a constant value of 2π. After that, $\omega[n]$ is passed to an integral block, the angle $\theta[n]$ is calculated based on the input angular frequency. Finally, to generate the sawtooth waveform that will be used for firing pulse generator, an angle resolver module is added to convert the output angle in range of 0 to 2π.

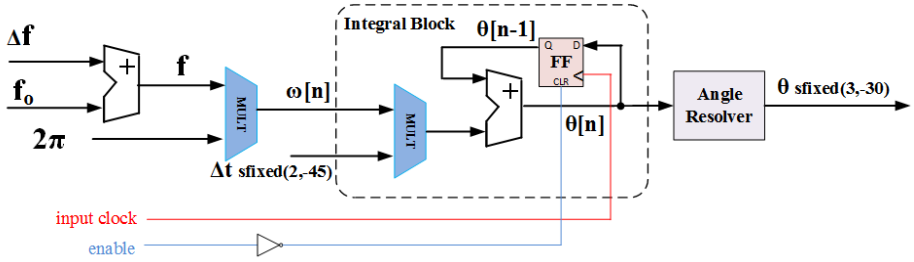


Figure 5.12: Hardware Implementation of the VCO Block

Chapter 6

Validation of FPGA-based IC-PLL

Both open loop test and Hardware-In-the-Loop(HIL) test are conducted to validate the performance of the FPGA-based IC-PLL, following studies are made in this chapter:

1. Comparison of the FPGA-based IC-PLL by itself with the IC-PLL simulated on the RTDS.

2. Comparison of the FPGA-based IC-PLL controlling a simulated HVdc system against complete model of the IC-PLL and HVdc system both modelled in RTDS.

3. Comparison of a TV-PLL simulated on RTDS with FPGA-based IC-PLL and IC-PLL simulated on RTDS.

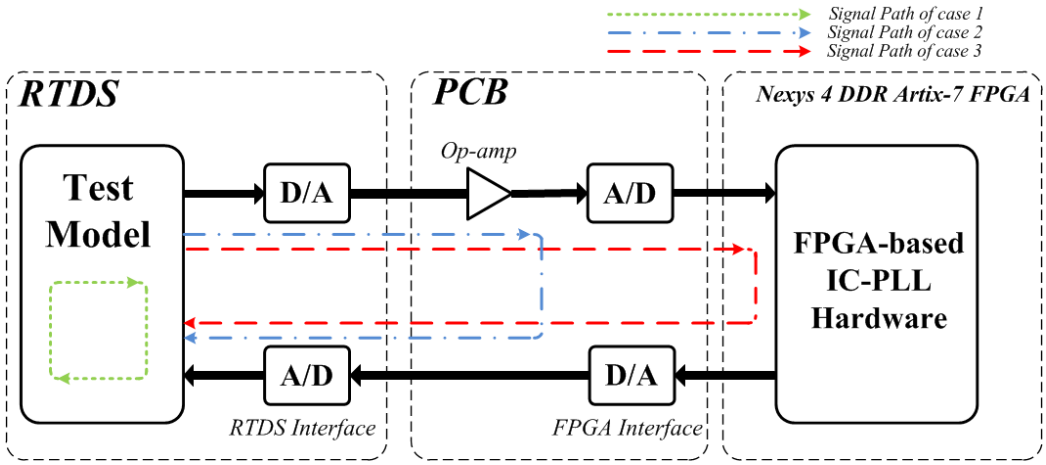
4. Investigation of interfacing issues by simulating the simulated system with PSCAD/EMTDC.

The first three comparisons are made to validate the performance of the FPGA-based IC-PLL and to compare its performance with the TV-PLL and IC-PLL simulated on the RTDS. Then simulations were done using PSCAD/EMTDC to study the affect of sources of error in a HIL test when interfacing the PLL to a real-time digital simulator.

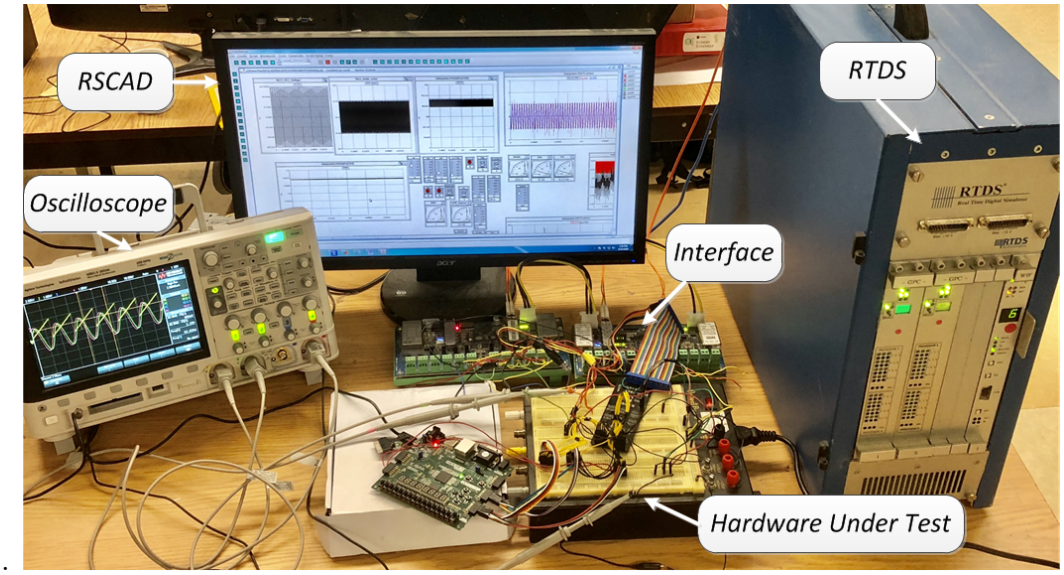
6.1 Experiment Hardware Setup

The experiment hardware setup consists of three parts: Real Time Digital Simulator(RTDS), the PCB board and the *Nexys 4 DDR Artix-7* FPGA board. The digital IC-PLL is implemented on the FPGA, the RTDS models the test circuit and generates analog signals through the Giga-Transceiver Analogue Output Card(GTAO) card to test the digital IC-PLL and the PCB board acts as an interface between the RTDS and the FPGA. Moreover, the FPGA-based IC-PLL output signals can be observed on either the oscilloscope or $RSCAD$. To observe signals on $RSCAD$, they are sampled through the Giga-Transceiver Analogue Input Card(GTAI) card. One advantage of displaying signals on $RSCAD$ is that it makes the comparison between simulation results and FPGA-based IC-PLL results much easier(instead of exporting data from oscilloscope and import them to $RSCAD$, $RSCAD$ can directly sample the data through GTAI card and generates plots). The actual hardware setup is shown in Fig. 6.1b.

Three testing cases and their corresponding signal paths are shown in Fig. 6.1a. In case 1, all circuits are modelled inside RTDS, therefore, we can obtain theoretical results by avoiding possible effect of interfacing in real world. For case 2, signals



(a) Block Diagram of the Hardware



(b) Actual Hardware Setup

Figure 6.1: Experiment Setup

acquired for the IC-PLL calculation(u_{abc},i_{abc},etc.) are passed through the I/O interface and op-amps, then signals are looped back to RTDS by which the effect of possible signal delay due to the interface will be presented in the experiment result. In case 3, the actual FPGA-based IC-PLL is incorporated with the test cir-

cuit simulated inside RTDS, signals are sampled by RTDS for either control and/or observation purpose. The reason of having first two cases is to obtain reference results that can be compared with the FPGA-based IC-PLL result to investigate the performance of the FPGA-based IC-PLL. The source of error that may have impact on the CHIL result are discussed in section 6.4. Furthermore, the actual control signal coming from the FPGA-based IC-PLL is the reference angle θ_{PLL} where $\theta_{PLL} = \int 2\pi(f_o + \Delta f)dt = 2\pi f_o + \int 2\pi\Delta f dt$. It is difficult to observe small variations in θ_{PLL}. Instead, we look at Δf, which is the contributor to small changes in phase.

6.2 Open Loop Test

In the open loop test, the firing signals for the HVdc valves are taken from the internal fully simulated model. The output of the FPGA-based IC-PLL is merely observed to see if it tracks the internal PLL. I did this because I was testing the FPGA-based IC-PLL only and did not want to look at the interfacing issues or inaccuracies that occur when interfacing the output of the FPGA-based IC-PLL to the RTDS. The following comparisons are made:

1. Comparison of estimated phase A voltages obtained from the IC-PLL modelled inside the RTDS and the physical constructed IC-PLL on FPGA platform.

2. Comparison of frequency output signals obtained from the TV-PLL and the IC-PLLs.

3. Comparison of frequency output signals obtained from the FPGA-based IC-PLL

with use of different time steps.

The first two comparisons are made to test the functionality of the FPGA-based IC-PLL and to compare the dynamic performance of the TV-PLL with the performance of the IC-PLL. The aim of the third comparison is to identity the maximum operating speed of the FPGA-based IC-PLL.

6.2.1 The Blackwater HVdc system

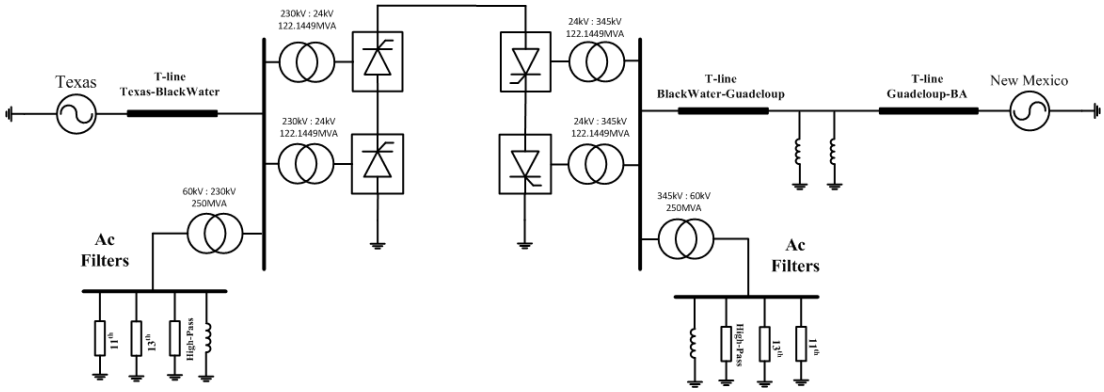


Figure 6.2: Blackwater Back-to-Back HVdc Model[32]

The test system model is based on the Blackwater back-to-back HVdc system as described in the *RTDSManual*[32]. The Blackwater HVdc system model is designed to interconnect Texas power system with New Mexico power system which is shown in Fig. 6.2. Two grids are connected to the 12-pulse converter through transmission lines and ac filters are provided on both rectifier and inverter side. Under normal operation, the rectifier side is under constant current(CC) control and the inverter side is under constant extinction angle(CEC) control. Some parameters of the Blackwater HVdc system are listed in table 6.1.

Table 6.1: Basic Parameters of the Blackwater HVdc system[32]

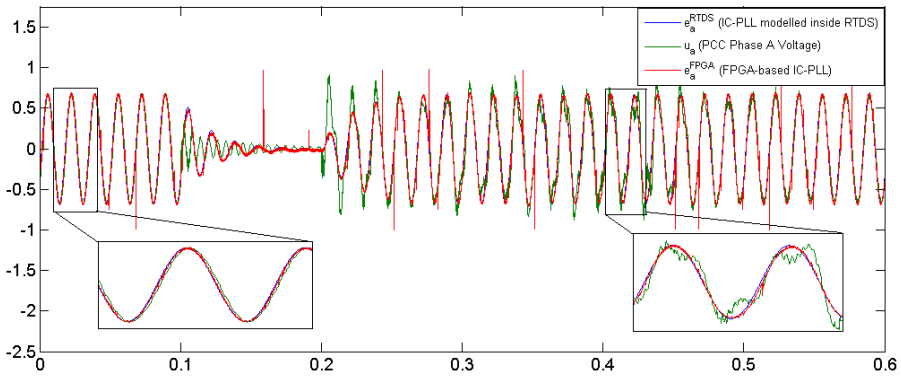
Parameters	Rectifier Side	Inverter Side
Rated Power	200MW	
Rated Dc Current	3.6kA	
Rated Ac Bus Voltage	230kV	345kV
Rated Dc Voltage	56.8kV	55.55kV
Rated Valve Voltage	24kV	24kV

6.2.2 Open Loop Test Results

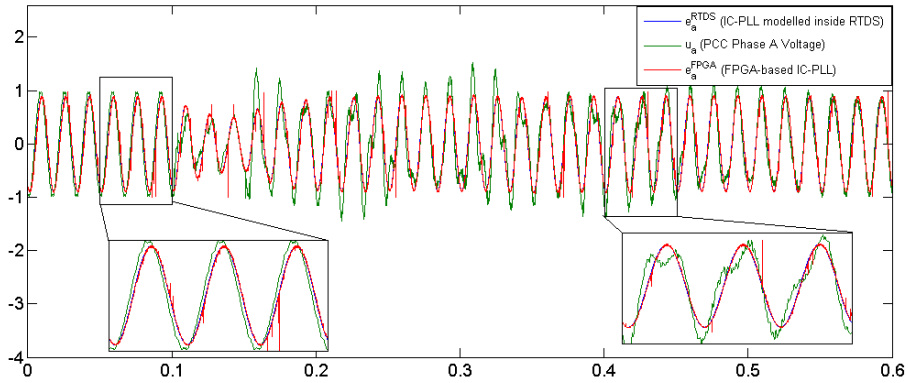
6.2.2.1 Estimated Phase A Voltage from IC-PLLs

Two test cases are performed by applying source fault on the rectifier side ($0p.u$ for $0.1s$) and the inverter side ($0.5p.u$ for $0.05s$) to study the performance of IC-PLL. Fig. 6.3 shows the results for the IC-PLL completely modelled inside the RTDS simulator e_a^{RTDS}, the IC-PLL implemented on the FPGA board e_a^{FPGA} and the PCC phase A voltage u_a. From Fig. 6.3a and 6.3b, we can see that in both cases, e_a^{FPGA} is very close to e_a^{RTDS} before, during and after the source fault. They are less distorted as compared to u_a as they are supposed to track the fundamental components. Also, there is a phase shift between the PCC voltage and the estimated voltage, as there is a phase change caused by the thevenin equivalent impedance between the estimated voltage and PCC voltage which was discussed earlier in section 3.4. Moreover, e_a^{RTDS} and e_a^{FPGA} go back to normal after the fault which make sense because source fault is removed after 0.1 s and 0.05 s for ac grid on rectifier and inverter respectively.

One thing to notice is that, in Fig. 6.3, there are small spikes seen in FPGA-based IC-PLL results. The spike duration is as same as the DAC sampling period. That is



(a) Rectifier Side Source Fault



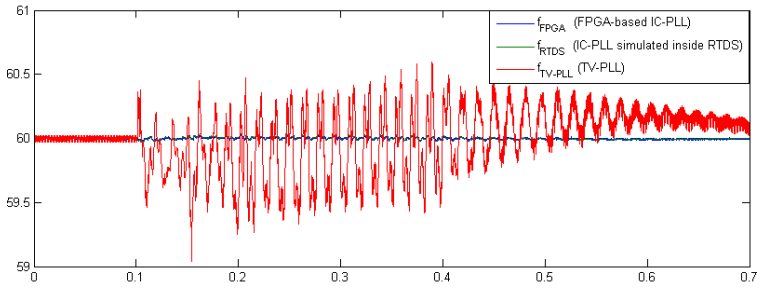
(b) Inverter Side Source Fault

Figure 6.3: Test Results of Source Faults on Rectifier and Inverter Side

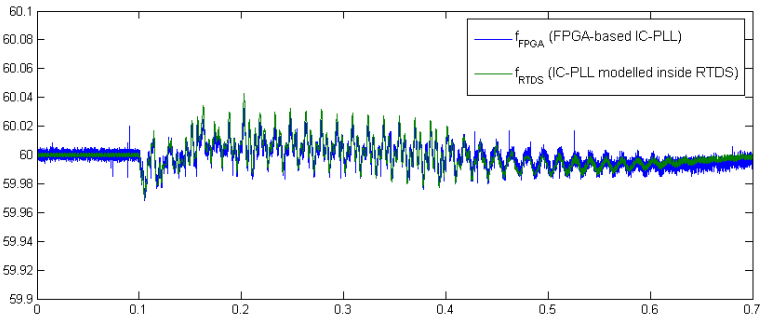
because of the communication error due to the loss of bit. This is clearly not caused by the internal implementation of the IC-PLL because it is a dynamical system, and if the value of the spike is the actual value inside FPGA then that means there is a discontinuity in the system, and the rest of the calculation results would also have been affected. And that is not the case as observed in Fig. 6.3.

6.2.2.2 Frequency Output Signals from FPGA-based IC-PLL and Simulated IC-PLL

By locking on to an estimated voltage point, the tracking performance of IC-PLL is expected to be better than TV-PLL. To verify that, the response of frequency of the TV-PLL and IC-PLL during the source fault on the inverter side are obtained in Fig. 6.4.



(a) TV-PLL vs IC-PLL



(b) RTDS result vs FPGA-based IC-PLL result

Figure 6.4: Response of Frequency during Inverter Side Source Fault

In Fig. 6.4a, when a source fault is initiated at t=0.1 s, the TV-PLL shows considerable error in tracking the frequency. The frequency output of TV-PLL varies

between 59.5 Hz and 60.5 Hz, however, in the same figure, the IC-PLL frequency output remains close to 60 Hz, which means the IC-PLL tracking performance is less affected by the disturbance due to source fault as compared to the TV-PLL. Fig. 6.4b shows a magnified view of the frequency output from IC-PLL simulated inside RTDS and the FPGA-based IC-PLL. The FPGA-based IC-PLL result shows a good match with the RTDS simulation result. The peak to peak frequency excursion is around 0.07 Hz which is better than the TV-PLL result. From above tests, we can see that the IC-PLL has better tracking performance as compared to TV-PLL and the FPGA-based IC-PLL is working.

6.2.2.3 Frequency Output Signals of FPGA-based IC-PLL with different Time Steps

The maximum operating speed of the FPGA-based PLL(i.e its throughput) is determined by the time required to complete the computations described in chapter 5 as well as the sampling rate of the ADC and the DACs. In this section, we investigate the throughput assuming that the sampling rate is fixed.

The simplest way to estimate maximum throughput that the the IC-PLL module can achieve is to reduce the time step till an error occurs. In this case, time steps of $40\mu s$, $4\mu s$, $0.5\mu s$ and $0.4\mu s$ are used and the resulting frequency outputs during the source fault on the inverter side are shown in Fig. 6.5. It can be seen that the frequency outputs are nearly equal with time steps of $40\mu s$, $4\mu s$ and $0.5\mu s$. And these results also match the simulated IC-PLL result obtained from the RTDS simulator in Fig.6.4b. However, when the time step reduces to $0.4\mu s$, the result becomes inaccurate

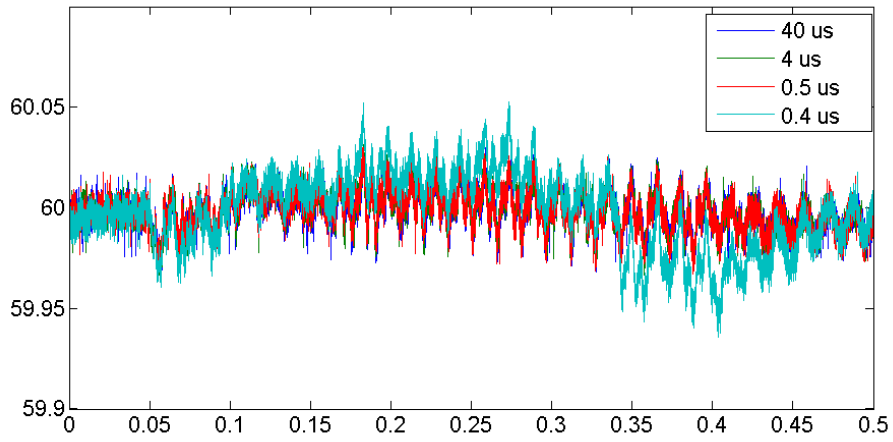


Figure 6.5: Frequency Output of FPGA-based IC-PLL with different Time Steps

which indicates that the computation process is not able to finish within the time-step duration. From the above observation, the execution period for the FPGA-based IC-PLL is between $0.4\mu s$ and $0.5\mu s$ or 40 to 50 clock cycles of the 100 MHz system clock. The results show that the IC-PLL module can operates with a throughput of $2M$ Samples/s (time step of $0.5\mu s$). One thing to notice is that, the throughput obtained in this section corresponds to a FPGA clock frequency of 100 MHz, the throughput can be increased with a faster system clock. On the other hand, the throughput can be further increased at a sacrifice of accuracy by reducing the number of iterations of the CORDIC algorithm.

6.3 Controller Hardware-In-the-Loop test

Low order harmonic instabilities in the HVdc system is a problem, particularly with weak ac systems that have ESCR less than 2[33][34]. The IC-PLL can improve

the propensity for such low order instabilities. One particular case of harmonic instability associated with the inaccurate dynamic tracking of the PLL is the 2^{nd} order harmonic instability. A simplified LCC-HVdc system is built on the *RTDS* simulator to demonstrate the system improved harmonic stability by the accurate tracking of the IC-PLL compare with the TV-PLL. By having the simplified HVdc system, we can run real-time simulation with a smaller time step as compared to the Blackwater HVdc system. In this section, a Controller Hardware-in-the-Loop(CHIL) test is conducted by connecting the FPGA-based IC-PLL to the LCC-HVdc model to further investigate the performance of the FPGA-based IC-PLL.

6.3.1 The Simplified LCC-HVdc Model

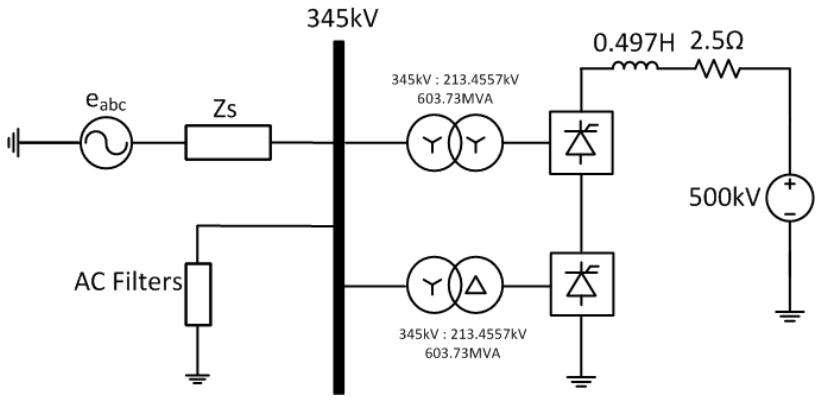


Figure 6.6: Simplified LCC-HVdc System Model(Rectifier Side)

Fig. 6.6 illustrates the LCC-HVdc model for testing the IC-PLL. This model is modified from the CIGRE HVdc benchmark model[35], which is a widely used benchmark model for HVdc studies. The inverter side of the system is replaced with a dc voltage source assuming it is in voltage control mode. On the rectifier side, a

weak ac grid with SCR of 1.8 (1000MW/345kV) is connected to the dc link through a 12-pulse converter. The ac grid is represented with a thevenin equivalent voltage source in series with a thevenin equivalent impedance, ac filters are connected to the ac bus to reduce harmonics and provide reactive power. The rectifier side is under constant current control and the control parameters are tuned to excite the 2^{nd} harmonic on the ac side.

6.3.2 CHIL Test Results

As mentioned in section 6.1, CHIL results are obtained with three testing cases as listed below:

- **Case 1**: All circuits are modelled inside the RTDS.
- **Case 2**: Control signals are passed through the I/O interface and then looped back to the RTDS.
- **Case 3**: FPGA-based IC-PLL is incorporated with the test circuit simulated inside the RTDS.

With these three testing cases, we can study the IC-PLL with or without taking interfacing issues into account.

6.3.2.1 Test Results at Steady-State

Fig. 6.7 shows the comparison of estimated phase A voltages from the FPGA-based IC-PLL $e_{_{FPGA}}$ and the IC-PLL simualted inside the RTDS $e_{_{RTDS}}$ with the PCC voltage u_a at steady-state. As mentioned in section 3.3, the IC-PLL is locked onto

a virtual voltage point by compensating the voltage drop across the ac network impedance. Therefore, we expect a phase different between e_{RTDS}& e_{FPGA} and u_a, which can be seen in the same figure. To obtain the correct timing reference, three phase estimated voltages generated from the FPGA-based IC-PLL are passed through an angle correction module (section 3.4) to provide corrected phase angle that is delivered to the firing pulse generator. The magnitude and phase of estimated voltage between the FPGA-based IC-PLL e_{FPGA} and the RTDS simulation e_{RTDS} are approximately the same which shows that the FPGA-based IC-PLL is working as designed.

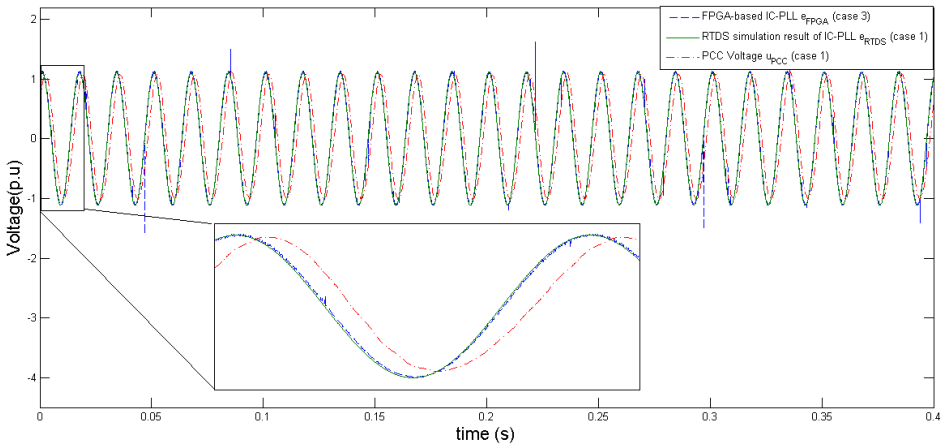
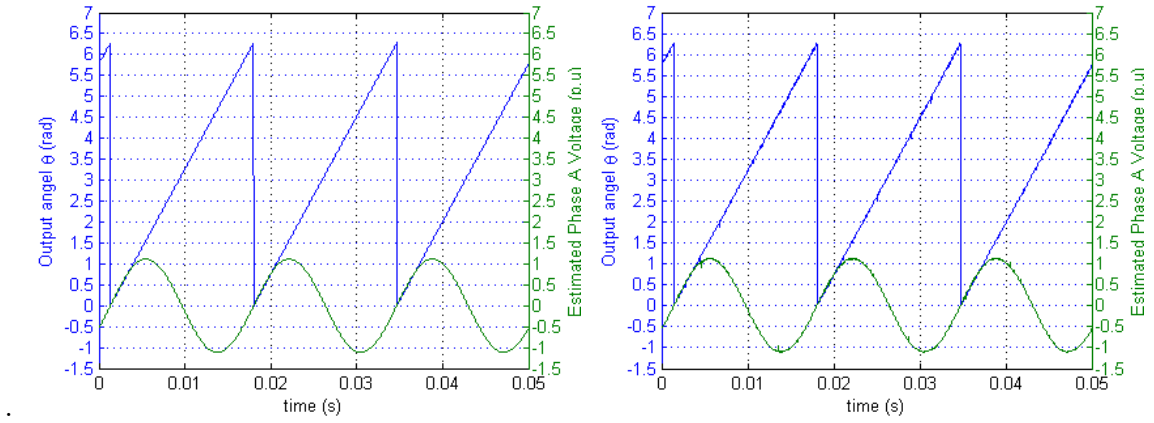


Figure 6.7: Estimated Voltage and PCC voltage at Steady-state

The same is observed in Fig. 6.8 which shows the VCO output starting from the zero crossing of the estimated phase A voltage. For the IC-PLL, the angle output is synchronized with the estimated voltage. Fig. 6.8a shows the RTDS simulation result. When the IC-PLL is locked onto the estimated voltage point, the output angle should be 0 at the zero-crossing of the estimated phase a voltage. Similar conclusion can be obtained from Fig. 6.8b. This indicates the tracking performance of the



(a) RTDS Simulation Result of IC-PLL (case 1) (b) FPGA-based IC-PLL (case 3)

Figure 6.8: Relationship between Output Angle θ and Estimated Voltage

FPGA-based IC-PLL is good. However, there is very slight phase difference between the FPGA-based IC-PLL and the RTDS simulation which can be observed in the zoomed in part in Fig. 6.7. It is because when we perform the HIL test, there is a unavoidable time delay introduced by the interfaces and the FPGA computation process. In addition to that, there are other factors that may affect the HIL result which are explained in section 6.4.

6.3.2.2 Second Order Harmonic Resonance

The LCC-HVdc system parameters are tuned purposely to have resonance near 2^{nd} harmonic on the ac side and this is reflected on the dc current as a harmonic with near-fundamental frequency.

Fig. 6.9 shows the dc current waveforms obtained from the FPGA-based IC-PLL, simulated IC-PLL and TV-PLL using different testing cases as discussed in section

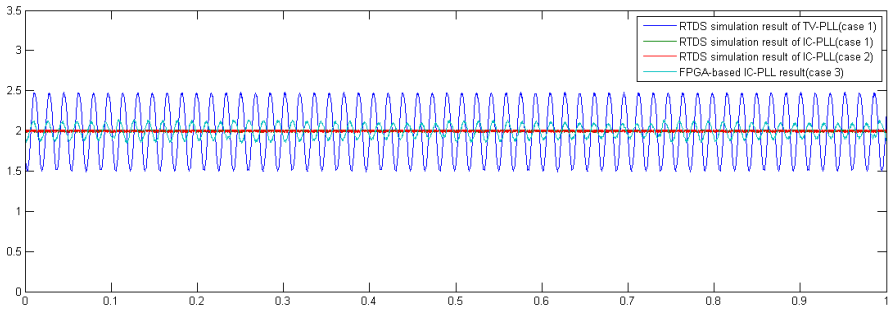
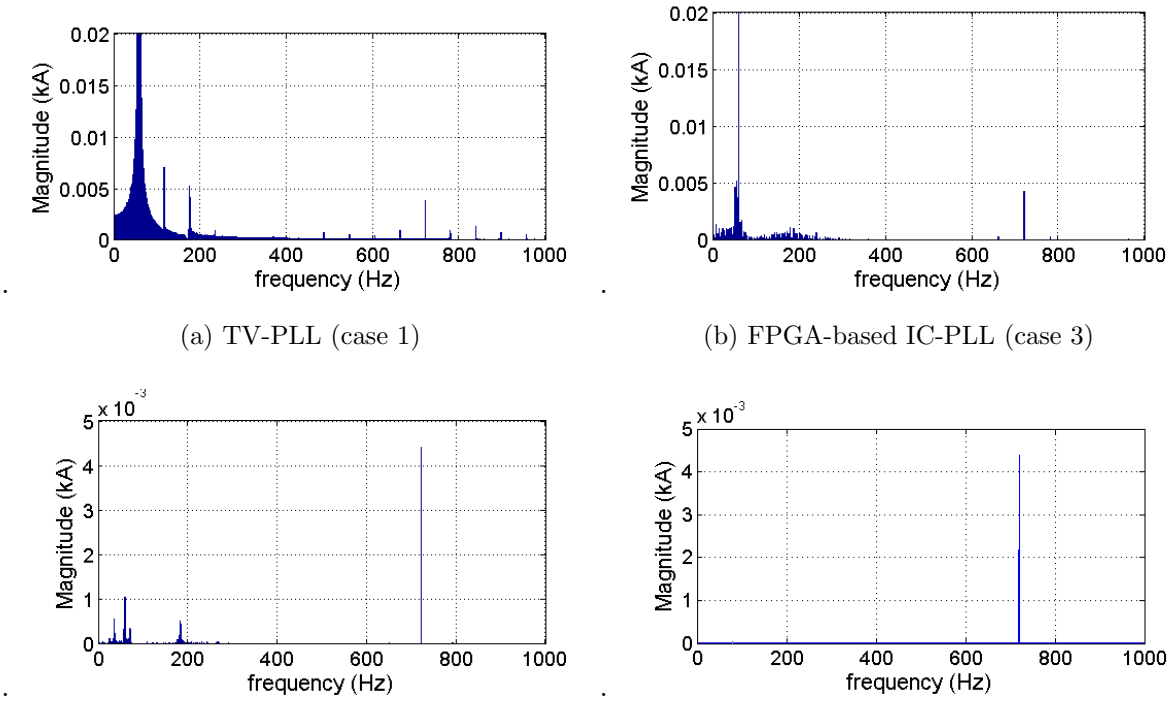


Figure 6.9: Dc Current Waveform with different Testing Cases

6.1. Under steady-state, the dc current contains a harmonic of near-fundamental frequency which is the evidence of 2^{nd} order harmonic on ac side when using TV-PLL. The peak to peak oscillation starts from 1.5 p.u to 2.5 p.u. The dc current waveforms of IC-PLL have better performance as compared to the TV-PLL as expected since they have less oscillation.

Furthermore, by taking the Discrete Fourier Transform(DFT) of dc currents, harmonic components of dc current for each cases are obtained. Fig. 6.10a to 6.10d show the frequency spectrum of TV-PLL and IC-PLL with different testing cases. Fig. 6.10a and 6.10d results are obtained from case 1 where all components are modelled inside the RTDS. It can be seen that the non-characteristic harmonics close to 60 Hz are present while using the TV-PLL, however, with the IC-PLL, only the characteristic harmonic(i.e 12^{th}) is present which means the IC-PLL has better performance as compared to the IC-PLL. Fig. 6.10b to 6.10d show the IC-PLL results obtained from case 3, 2 and 1. Compared Fig. 6.10d with 6.10c, a small amount of near-fundamental harmonic exists when the IC-PLL is looped back to the $RTDS$ which is caused by the I/O interface and the noises which exists in the real world. A similar



(a) TV-PLL (case 1)

(b) FPGA-based IC-PLL (case 3)

(c) IC-PLL (case 2)

(d) IC-PLL (case 1)

Figure 6.10: DFT of Dc Current with different Testing Cases

spectrum can been seen from Fig. 6.10b where the magnitude of near-fundamental frequency harmonics are larger.

6.3.2.3 Test Results During Three Phase AC Fault at PCC

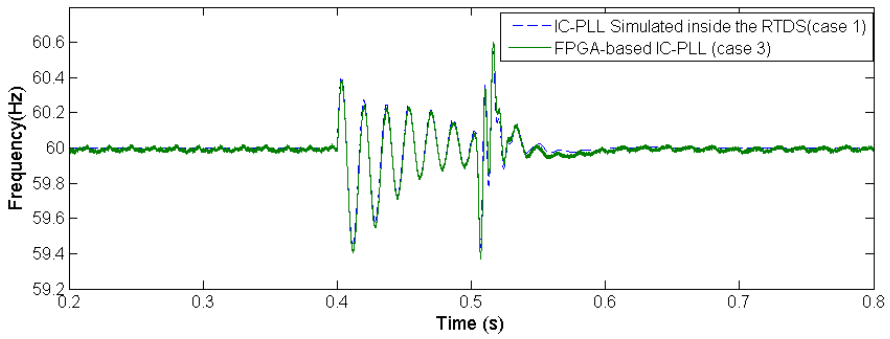


Figure 6.11: Response of Frequency during Three Phase Ac Fault

Fig. 6.11 demonstrates the comparison of the frequency output signals between the RTDS simulation of the IC-PLL and the FPGA-based IC-PLL. After a fault is applied, the magnitude of frequency signals from the RTDS and from the FPGA-based IC-PLL vary between 59.4 Hz to 60.6 Hz in both cases. A small oscillation is presented in the FPGA-based IC-PLL result, but the transient response is quite similar to the IC-PLL transient response.

6.4 Parametric Study of Sources of Error in CHIL

In a CHIL simulation, the errors caused by the interfacing of two separate platforms (in this case, a digital simulator and the FPGA-based IC-PLL) may lead to inaccurate simulation results[36]. Therefore, it is necessary to investigate the effects of various parameters of the interface hardware on the results obtained from the CHIL simulation. From Fig. 6.11 on previous page, the response of frequency f_{PLL} from the RTDS simulator and the FPGA-based IC-PLL shows a good agreement even during

the transient period. However, in Fig. 6.9,the steady-state dc link current obtained from CHIL has a spurious oscillation at fundamental frequency (60 Hz) as compared to the result obtained by RTDS simulation (case 1). The oscillation magnitude has a peak value of about 8% of the average value. Since the control signal f_{PLL} shows a good match which is unlikely to cause such significant oscillation, we conjectured that the interfacing issues are the likely cause of the spurious oscillation. The sources of error in the CHIL simulation of the FPGA-based IC-PLL are investigated which include interface delays, dc offsets and finite precision of ADC and DAC.

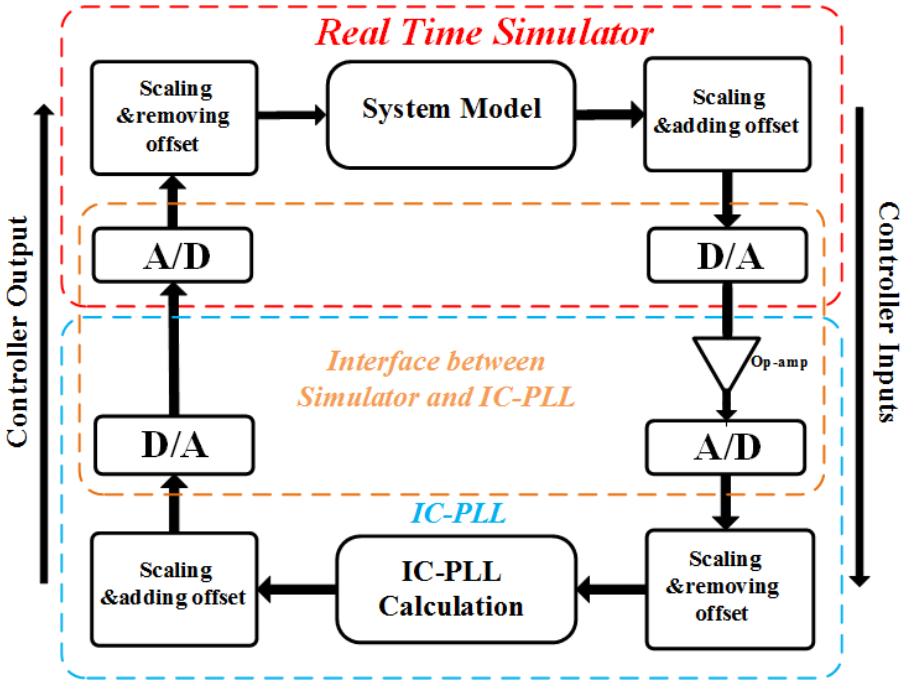


Figure 6.12: Function Diagram of CHIL Simulation

The function diagram of CHIL simulation is shown in Fig. 6.12. The IC-PLL has been implemented in a Xilinx Artix-7 FPGA and interfaced with a real-time simulator. The ADC as well as the DAC on the controller side are single ended and

have a 10-bit precision with a voltage range of 0 to 5 V. The ADC and DAC of the real time simulator have a 16-bit precision with a voltage range of $\pm 10V$. The DAC in this case has a differential output. The Op-Amp is used for converting the differential output to single-ended output. Based on the CHIL simulation setup, the following issues may have an impact on the accuracy of the CHIL simulation:

- **Time Delay**: In this CHIL simulation, the interface hardware contains ADC, DAC and Op-Amp circuits for signal conditioning. These circuits will introduce inevitable time delays and the CHIL simulation results may get affected by these delays. Many studies [36][37][38] show the interface delays could have a significant impact on the HIL simulation results. Therefore, the interfacing delays can be one of the causes.

- **Dc Offsets**: On the controller side, the voltage range of the ADC is from 0 to 5V, therefore, offsets are added to three phase voltage and current signals which are coming out from the real-time simulator. To obtain the correct signals, any dc offset that was added previously needs to be removed. Since the interface hardware is not ideal, the static dc offset removal may not be exact. While removing the dc offset, a small amount of dc offset may remain in these signals[39]. On the other hand, the differential Op-Amp gain may be slightly different from the designed values [40] due to different tolerance in the values of the components that have been used. In the interface hardware, the Op-Amp circuits convert multiple differential inputs to single-ended signals from the real-time simulator to the FPGA-based IC-PLL. This can also introduce errors in the simulation results.

- **Finite Precision of ADCs and DACs**: The precision of ADCs and DACs on the controller side are 10 bits while on the real-time simulator side are 16 bits. Inside the FPGA-based IC-PLL, the data outputs use fixed point format with 34 bits precision. Due to the limited precision of ADCs and DACs on the controller side, the data outputs are truncated, and truncation error may have impact on the CHIL result.

In conclusion, three problems that mentioned above have the potential of causing significant errors in the CHIL simulation results shown in Fig. 6.9. A parametric study has been done in this section using the off-line EMT simulation software ($PSCAD/EMTDC^{©}$). New components are introduced to the system model to simulate the CHIL simulation which include delay model, the DAC/ADC model as well as the bit truncation model.

6.4.1 Effect of Time Delay

In CHIL simulation, the time delay consists of ADC delay, DAC delay , Op-Amp delay as well as the computation delay. The total time delay is the summation of all these delays which can be treated as a time varying delay combined with a constant time delay where the time varying delay is caused by the interface between DACs and ADCs with different sampling frequencies.

Fig. 6.13 shows the results obtained from the off-line simulation with constant delays included in both inputs and output of the IC-PLL(after u_{abc}, i_{abc} and θ_{PLL}). By comparing the results of $120\mu s$ delay and a much larger, even unrealistic delay of $500\mu s$ with the no delay(i.e. the ideal case) result, no significant errors are observed in

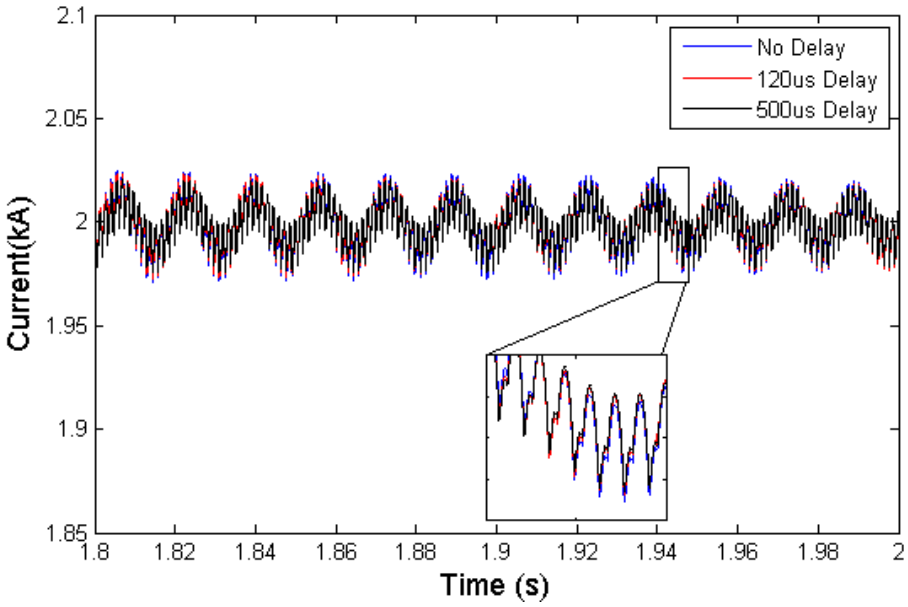
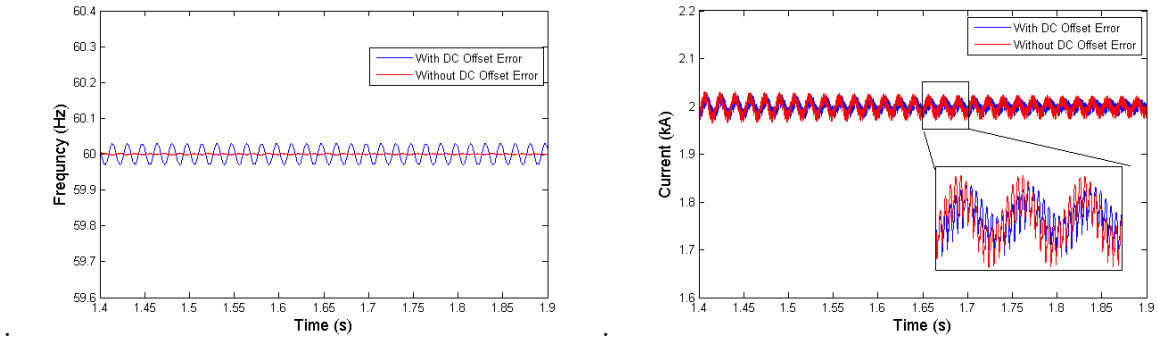


Figure 6.13: Comparison of different Time Delay

the simulation. The firing pulse is generated by comparing the firing angle signal with the IC-PLL output angle signal. Thus, the effect of the constant delay is to increase the firing angle. However, as the firing angle is generated by a feedback controller controlling the dc current (or extinction angle in the inverter case), the firing angle automatically adjusts to compensate the error due to constant delay which results in the dc current drop. Based on above results, the constant time delay is unlikely to cause the oscillation shown in Fig. 6.9.

6.4.2 Effect of Dc Offset

To investigate the impact of dc offset, different dc offsets in a range of ±2% p.u are added to the three phase voltage and current signals(u_{abc} and i_{abc}). Fig. 6.14 shows the effect of dc offsets on the simulation results. The dc offset introduces a



(a) Frequency Output of IC-PLL (b) Dc Link Current

Figure 6.14: Effect of Dc Offset

small oscillation in the frequency output of the IC-PLL as shown in Fig. 6.14a and a small amplitude oscillation in the dc link current which is shown in Fig. 6.14b. However, the oscillation is very small as compared to the oscillation observed from Fig. 6.9. Therefore, the dc offset is not the primary cause of the oscillation. In addition, the oscillation introduced by the dc offset errors can be removed by adding Digital High-Pass Filters(HPFs) after input signals.

6.4.3 Effect of Finite Precision of ADCs and DACs

The truncation introduced by the finite precision of the ADCs and DACs of the controller is modelled in the EMT model where the ADC and DAC hardware use 10 bits precision in the CHIL simulation. To study the impact of the precision of the ADC and DAC, 10 bit,11 bit and 12 bit precision has been simulated. As shown in Fig. 6.15, the oscillation occurs with 10 bits precision where a low frequency envelope can be seen in dc link current. By comparing the 10 bit result with the 11 bit and

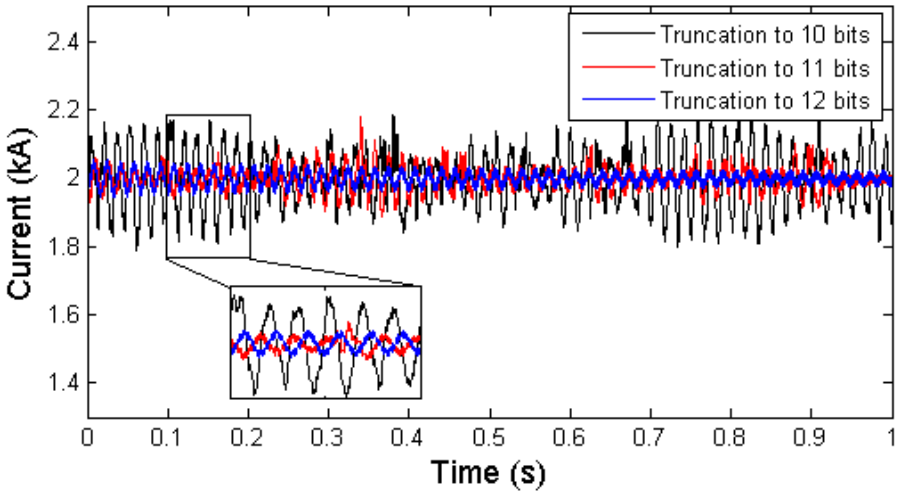
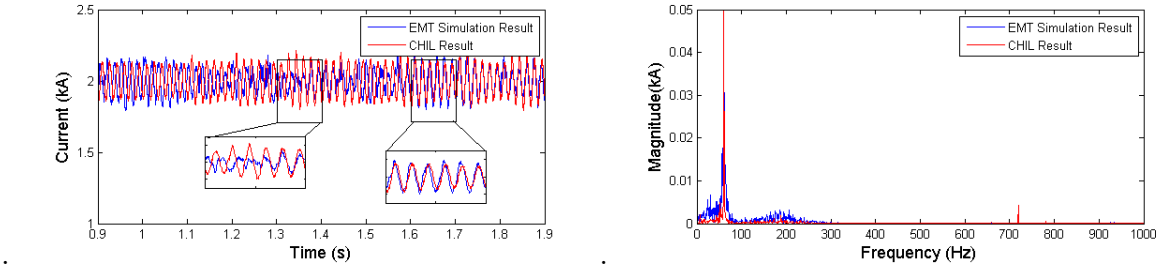


Figure 6.15: Effect of Variation of Number of Bits of Precision on Dc Link Current

12 bit results, the magnitude of oscillation reduces considerably as the number of bits increases. No significant oscillation is observed in the result with 12 bit, and the result with 11 bit precision is intermediate between 10 bit and 12 bit.



(a) Dc Link Current)

(b) DFT of Dc Link Current

Figure 6.16: EMT Simulation Result with 10 Bits Precision vs. CHIL Simulation Result

Fig. 6.16 shows the comparison between EMT simulation result with 10 bit precision and CHIL simulation result. The EMT simulation result shows a good match

with the CHIL simulation result as far as dc link current oscillation is concerned (except for short intervals as shown in Fig. 6.16a from t = 1.3 s to 1.4 s). To confirm this, the DFT of the dc link current is plotted in Fig. 6.16b. The trends in the DFT plot show that there is a match between the two spectra, although their magnitudes are slightly different. The small spike at 720 Hz shown in Fig. 6.16b is the characteristic harmonic (12^{th} harmonic for a 60Hz system). Based on the simulation, we can see the finite precision of ADC and DACs will have an impact on the CHIL simulation in our particular case.

Chapter 7

Conclusion and Future Work

The PLL is widely adopted in HVdc systems to provide fundamental frequency and phase information of the positive sequence voltage of the ac grid. A new type of PLL called the Impedance-Compensated Phase-Locked Loop (IC-PLL) [7][8] was introduced to compensate for the voltage drop across the ac network's Thevenin impedance to make the phase locking more immune to transient and distortion of the system. In this thesis, an actual IC-PLL is designed and implemented on the FPGA platform.

7.1 Contributions of this Thesis

The main contributions of this thesis are listed as follows:

- **FPGA Implementation of the IC-PLL**: In this thesis, an IC-PLL is implemented on an FPGA platform. The FPGA-based IC-PLL consists of two modules: the I/O module and the IC-PLL module. Serial operations involved

in this design are divided into parallel operations as much as possible to take advantage of the nature parallelism of the FPGA to achieve fast execution time. In the I/O interface module, the control signals are shared by 6 ADCs(or 6 DACs) to sample and process the data at the same time. In the IC-PLL module, fixed point format is employed and calculations are separated into parallel calculations based on their dependency. Moreover, a PCB is designed to reduce the size of the hardware(eg: ADCs,DACs), improve reliability and reduce electric noise.

- **Verification of the FPGA-based IC-PLL**: In this thesis, tests are performed to verify the performance of the FPGA-based IC-PLL by interfacing it to the simulation of LCC-HVdc transmission systems on a Real-Time Digital Simulator(RTDS). An open loop test is conducted to test the dynamic performance of the IC-PLL. The test system uses model of the Blackwater back-to-back HVdc system between Texas and New Mexico. The test results show the FPGA-based IC-PLL has better dynamic performance as compare to TV-PLL. Furthermore, a CHIL simulation is also performed. In this case, a simplified LCC-HVdc system is employed which is based on the CIGRE benchmark model[35]. One reason to use the simplified HVdc system is that we can run the real-time simulation with a smaller time step as compared to the Blackwater HVdc system. Although the frequency output obtained from CHIL indicates that the FPGA-based IC-PLL is working properly, however, there is a spurious oscillation at fundamental frequency (60 Hz) in the steady-state dc link current which may be caused by interfacing issues.

- **Parametric study of sources of error in CHIL**: In this thesis, potential sources of error when conducting the CHIL simulation of a PLL are investigated. The investigation was carried out by building a detailed off-line simulation of the CHIL simulation itself by including models of the interface delays, offsets and finite precision of ADC and DACs. The results show that in this particular case, the constant interface time delay and dc offsets have minimal impact on the accuracy, however, the finite precision of ADC and DACs will have significant impact on the CHIL results.

7.2 Recommendations for Future Research

Some recommendations for future work are listed below:

- Optimization on the VHDL code and upgrade hardware to achieve higher accuracy and speed of the FPGA-based IC-PLL
- Study the performance of FPGA-based IC-PLL in a VSC-HVdc system
- Derive an analytical model to study the effect of errors introduced in the CHIL simulation and its impact on the overall system performance.
- Develop compensation methods for all errors in CHIL simulations that involve PLL.

Appendix A

FPGA Pin Assignment of IC-PLL Implementation and FPGA Resource Usage

Table A.1: FPGA Pin Assignment of IC-PLL Implementation

I/O Group	Name	PIN Number	Description
Switches	J15	en	enable FPGA-based IC-PLL
Pmod Header JA	C17	MISO_Ua	Phase A Voltage
	D18	MISO_Ub	Phase B Voltage
	E18	MISO_Uc	Phase C Voltage
	G17	MISO_Ia	Phase A Current
	D17	MISO_Ib	Phase B Current
	E17	MISO_Ic	Phase C Current
	F18	MOSI_ADC	SPI MOSI
	G18	SCLK_ADC	ADC Master Clock
Pmod Header JB	D14	CS_ADC	ADC Chip Select
	F16	Ua_out	Estimated Phase A Voltage
	G16	freq	frequency output signal
	H14	sclk_DAC	DAC system clock
	E16	LDAC	DAC Load Signal
	F13	CS_DAC	DAC Chip Select Signal
	G13	theta	angle θ
	H16	Ed_out	Estimated Ed
Pmod Header JC	J4	Uc_out	Estimated Phase C Voltage
	E6	Ub_out	Estimated Phase B Voltage

Table A.2: FPGA Resource Usage of different Modules on the FPGA-based IC-PLL

	LUT	FF	BRAMS	URAM	DSP
direct Park transformation	13045	11820	0	0	40
indirect Park transformation	5534	4043	0	0	48
direct inverse Park transformation	12999	11921	0	0	24
Loop Filter and Voltage Controlled Oscillator	1506	146	0	0	36
ADC Module	1482	819	0	0	12
DAC Module	608	423	0	0	0
data conversion for outputs	246	10	0	0	0
Low-Pass Filter	525	68	0	0	12
Voltage Estimation	2055	140	0	0	58
Clock Module	26	64	0	0	0

Appendix B

PCB Schematic

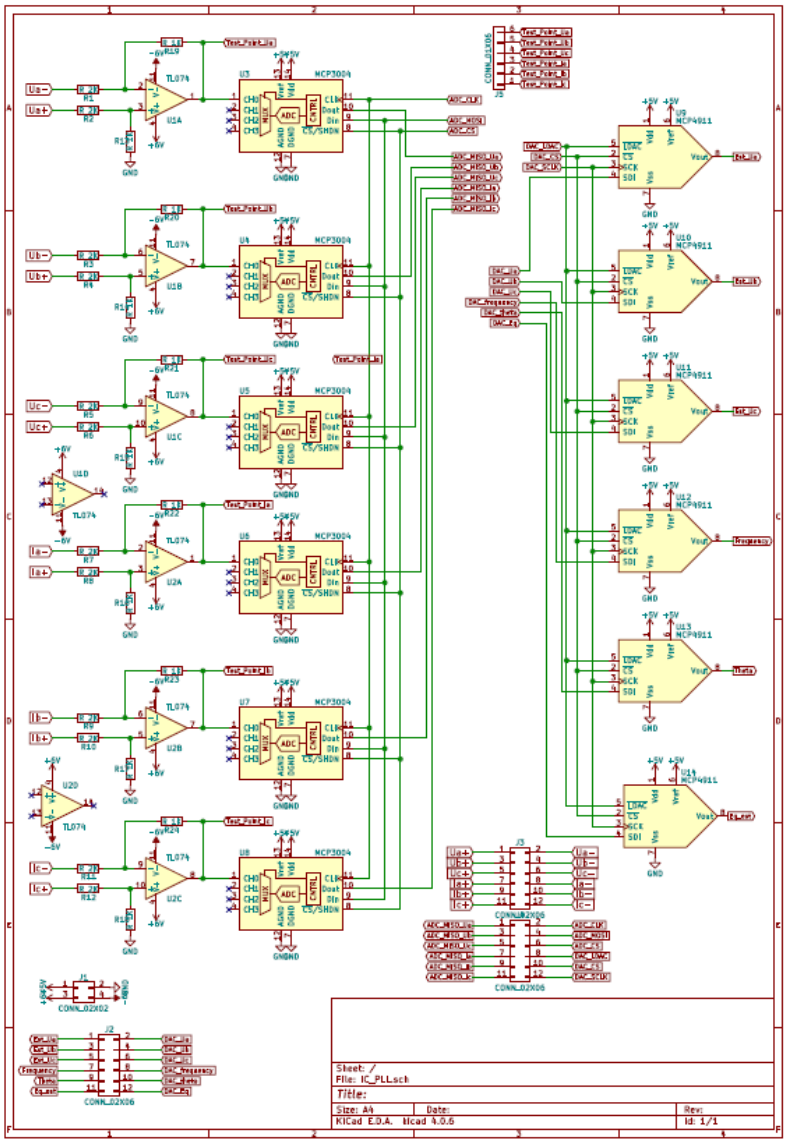


Figure B.1: PCB Schematic

Appendix C

Matrix

Park and Inverse Park Transformation(abc-dq and dq-abc)

$$\begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos\theta & \cos(\theta - 120^o) & \cos(\theta + 120^o) \\ \sin\theta & \sin(\theta - 120^o) & \sin(\theta + 120^o) \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} \tag{C.1}$$

and

$$\begin{bmatrix} u_a \\ u_b \\ u_c \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 1 \\ \cos(\theta - 120^o) & \sin(\theta - 120^o) & 1 \\ \cos(\theta + 120^o) & \sin(\theta + 120^o) & 1 \end{bmatrix} \begin{bmatrix} u_d \\ u_q \\ u_0 \end{bmatrix} \tag{C.2}$$

Appendix D

RTL schematic of the FPGA-based IC-PLL

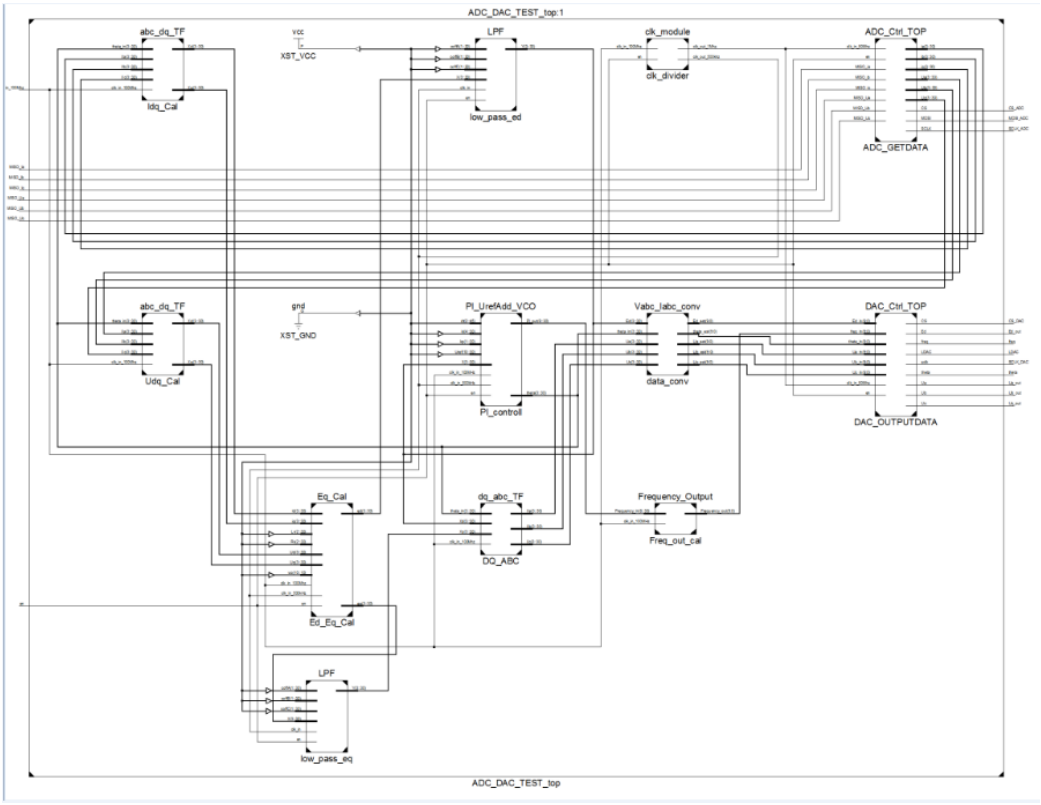
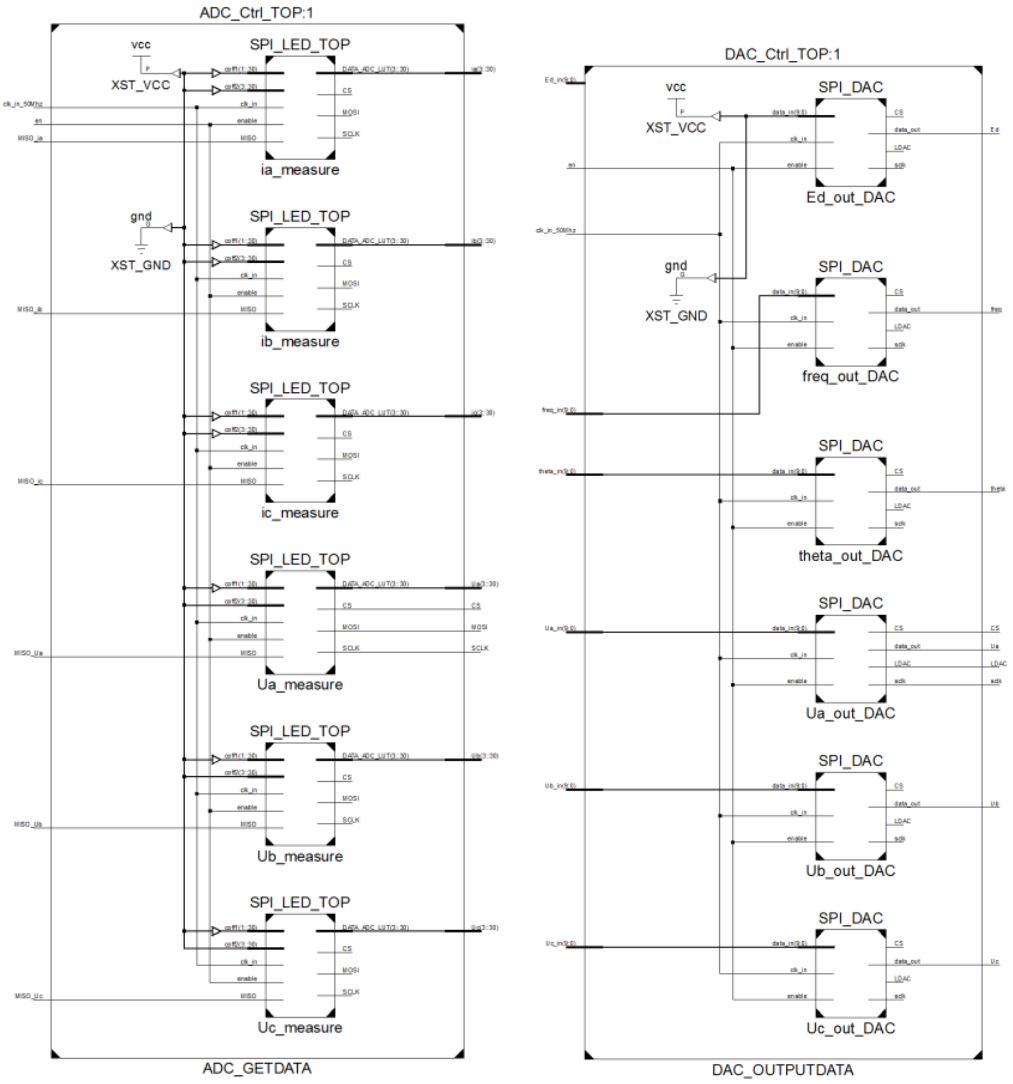


Figure D.1: RTL Schematic of Overall IC-PLL



(a) ADC Control (b) DAC Control

Figure D.2: RTL Schematic of I/O module

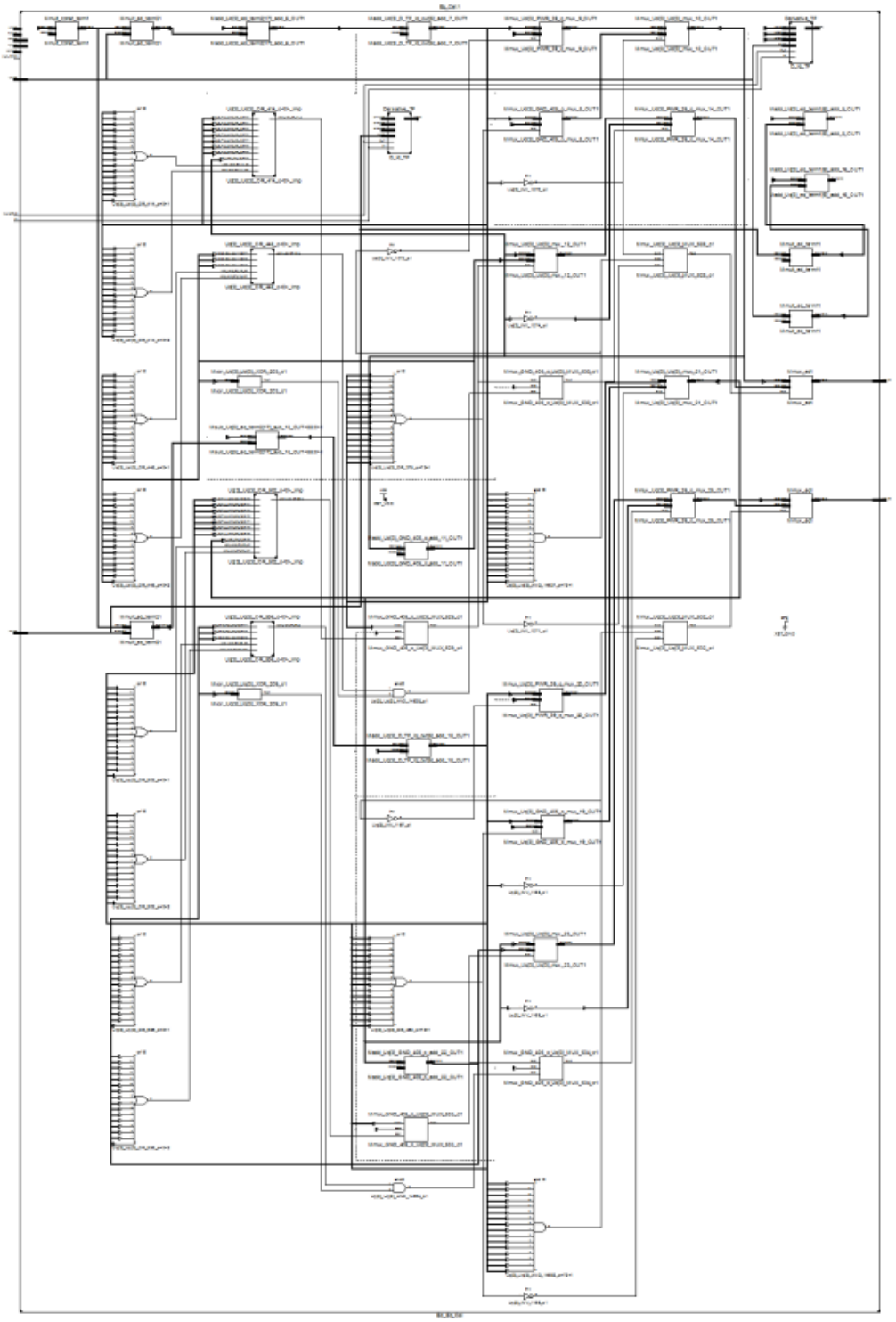


Figure D.3: RTL Schematic of Voltage Estimation Block

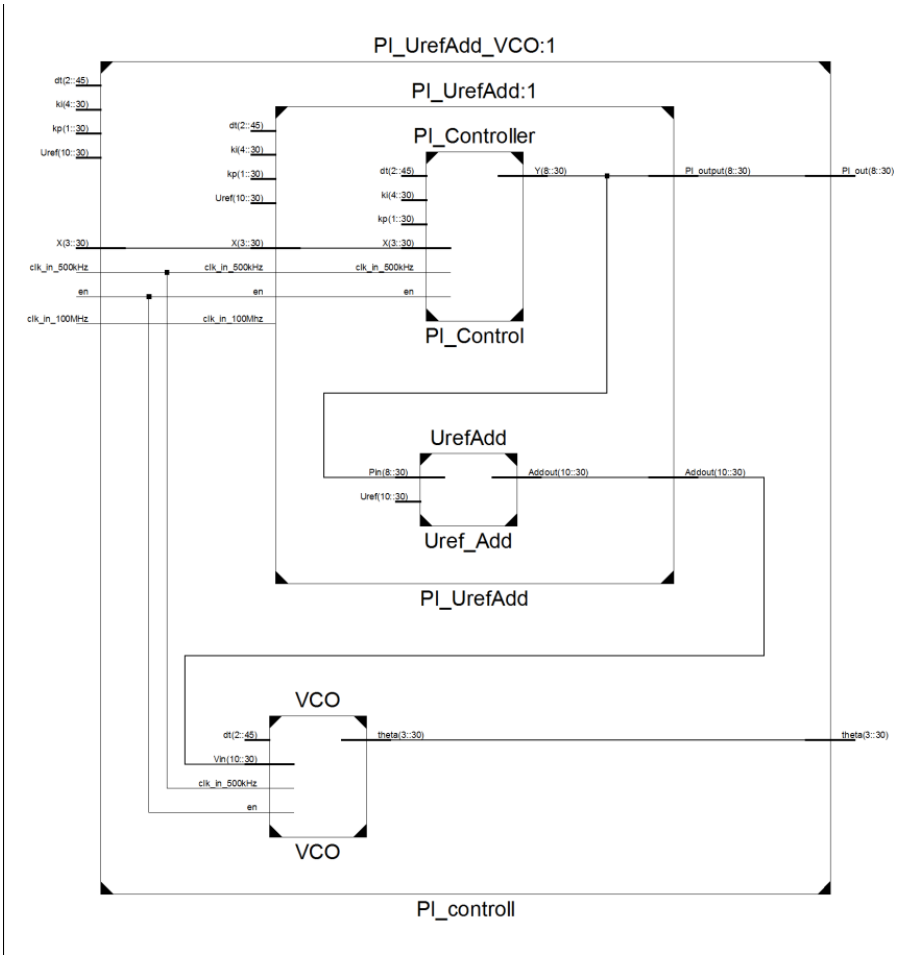


Figure D.4: RTL Schematic of PI Controller and VCO Block

Appendix E

FPGA Fundamentals

A brief introduction of Xilinx® FPGA fundamentals are discussed which include the architecture of Xilinx® FPGAs, design flow and programming techniques.

E.1 Xilinx FPGA Hardware Architecture

E.1.1 The Configurable Logic Blocks

The CLB is the main logic resource inside the FPGA for implementing logic functions. In Xilinx® 7 series FPGAs, each CLB element contains two slices and each slice is consists of four logic-function generators(or look-up tables), eight storage elements, wide-function multiplexers and carry logic. There are two types of slices which are called SLICEL and SLICEM. Both slices can provide logic functions, arithmetic as well as the ROM function. In addition to that, the SLICEM also provides data storage ability using distributed RAMs and data shift ability using 32 bit registers. Each CLB can have two SLICEL or one SLICEM and one SLICEL. Two slices are

not directly connected to each other in one CLB, and each slice is organized as a column with independent carry chain[41].

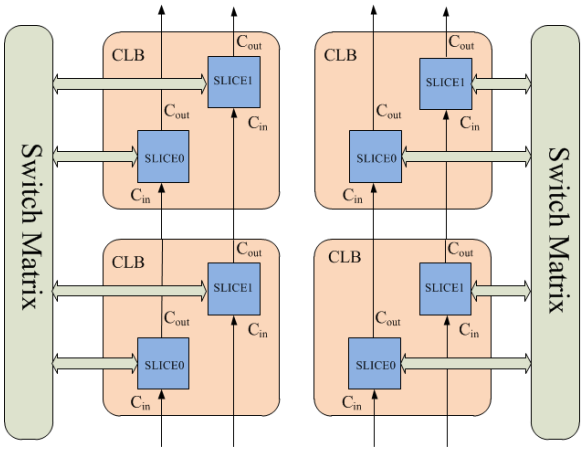


Figure E.1: Connection between CLBs and Slices[41]

As shown in Fig. E.1, two slices are arranged on the top and bottom of each CLB and they are in different columns. The switch matrix is connected with every slice inside the CLB for access general routing matrix. Similarly, other hardware resources such as DSP48E1 slice are connected to the switch matrix. All those element works together to provide the FPGA the reconfigurable ability.

E.1.2 Digital Signal Processing Slices(DSP48E1)

The Xilinx® 7 series FPGA provides dedicated Digital Signal Processing(DSP) slices to implement custom and fully parallel algorithms. Some components inside the DSP48E1 slice are shown in Fig. E.2 which are 25×18 two's-complement multiplier, 48-bit accumulator, power saving pre-adder, Single-instruction-multiple-data(SIMD) arithmetic unit, optional logic unit, pattern detector as well as the optional pipelining and dedicated buses for cascading [42]. The DSP48E1 slice is recommended when use

signed values in HDL source, pipelining, lower power etc. Generally, the DSP48E1 slice are used automatically for most DSP functions and arithmetic functions.

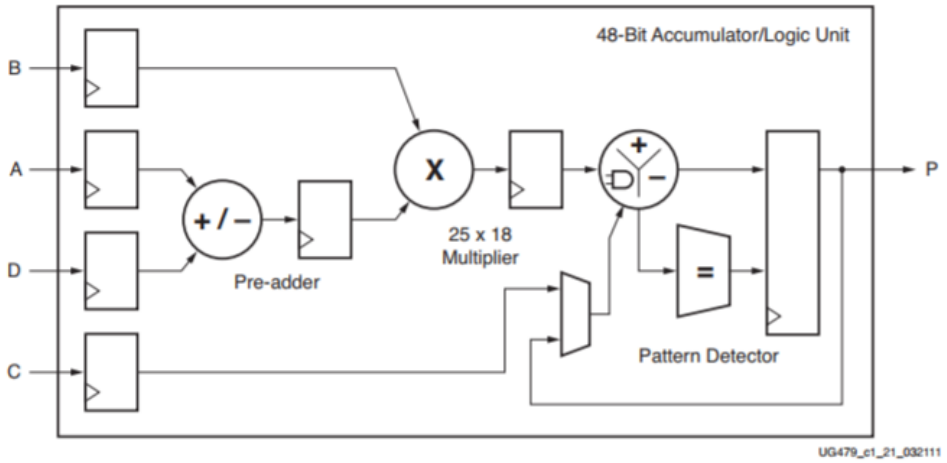


Figure E.2: Basic Function of DSP48E1 Slice [42]

E.2 FPGA Programming Language and Design Flow

The FPGA is configured using Hardware Description Lauguage(HDL) such as Very High-Speed Integrated Circuit Hardware Description Language(VHDL) or Verilog. The HDL targets on the Register-Transfer Level(RTL) coding. The VHDL is a strongly typed HDL compare to the Verilog but more verbose. Two design tools are provided for Xilinx® FPGA design which are the ISE® Design Suite and the Vivado® Design Suite, in this thesis, the ISE® Design Suite is employed. The general FPGA design flow shown in Fig. E.3 is explained below[43]:

- **Design Entry**: The design process starts with design entry using either schematic entry or HDL entry. For schematic entry, we draw gates and wires to describe

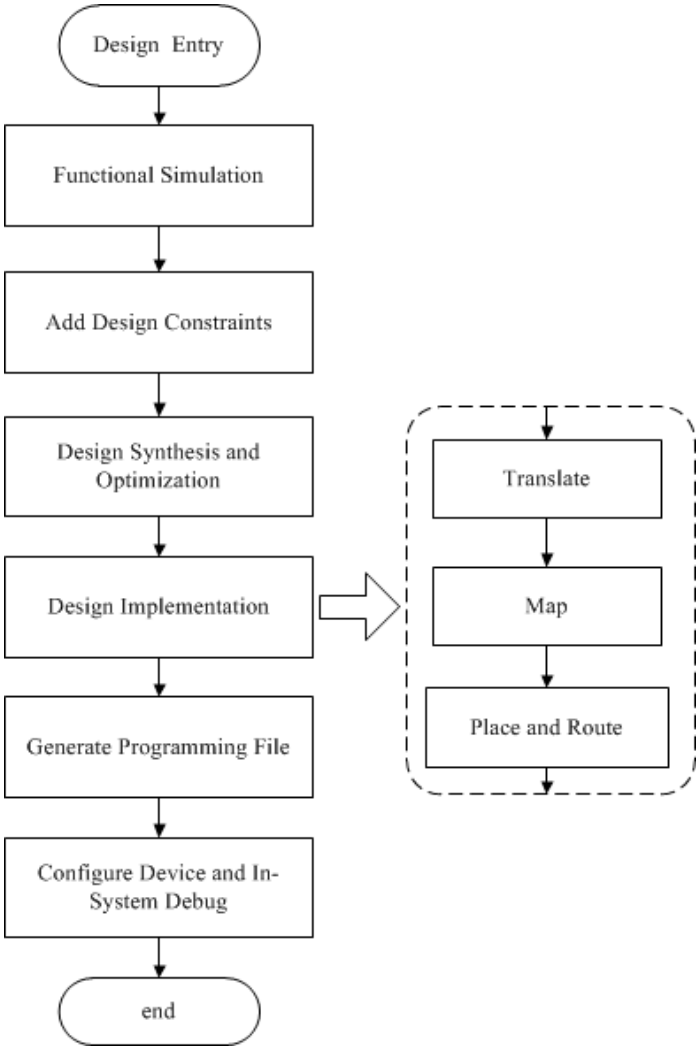


Figure E.3: FPGA Design Flow Chart [43]

the system behaviour, however, if system size gets larger, it is harder to design using schematic entry. The HDL entry uses HDL such as VHDL or Verilog to describe the behaviour of the system. Compared with schematic entry, the HDL entry provides a top-down design approach and it is portable and reusable.

- **Functional Simulation**: After finishing the design draft, a test bench is used to test the behaviour of the design entry. For a large design, we can write

test bench for each module to test their functionality and performance before simulating the entire design. Many tools (Isim®,Modelsim®, Vivado® Simulator,etc) can be used in this stage. In this thesis, Isim® is used.

- **Adding Design Constrains**: Design Constrains indicate the design requirements that the system must be met. Generally, the design constrains are separated into two categories: timing constrains and physical constrains. Timing constrains specifies the maximum allowable delay or skew on data paths or nets while physical constrains specifies the I/O locations, I/O standards,etc[44].
- **Design Synthesis and Optimization**: The synthesis and optimization process checks the syntax of the HDL codes, optimize the design hierarchy and generates a netlist file that contains both logic data design and constrains. The ISE® tool includes Xilinx Synthesis Technology (XST) which performs the synthesis process and generates the netlist file called NGC file.
- **Design Implementation**: The design implementation consists of three procedures: translate, map and place & route. During the translate process, all netlist files and constrain files are merged into a Xilinx® Native Generic Database (NGD) file which describes the logical design using FPGA primitives such as logic gates, LUT, RAMs, etc. The map process maps the logic generated from the NGD file into FPGA elements(CLBs and IOBs) and outputs a Native Circuit Description (NCD) file which physically represent the design that is mapped to the FPGA. During the place and route process, the NCD file is taken as the input, the logic blocks are placed and the internal connections are routed. After this process, a new NCD file is generated.

- **Generate Programming File**: Once the design is completely placed and routed, a bitstream is generated during this process to configure the FPGA.
- **Configure Device and In-System Debug**: In this process, the bitstream is downloaded to the target FPGA device through JTAG interface. Once the FPGA is configured, we can verify the FPGA functions by observing the digital output of the FPGA using digital logic analyser or oscilloscope, moreover, we can convert the digital signal output through a Digital-to-Analog Converter(DAC) to debug the FPGA.

References

[1] Chan-Ki Kim, Vijay K Sood, Gil-Soo Jang, Seong-Joo Lim, and Seok-Jin Lee. *HVDC transmission: power conversion applications in power systems*. John Wiley & Sons, 2009.

[2] P Haugland. Its time to connect: Technical description of hvdc light® technology. *ABB Techincal Report*, 2008.

[3] RL Koropatnick. Hvdc projects listing. *HVDC and Flexible AC Transmission Subcommitee, Tech. Rep*, 2012.

[4] J. Z. Zhou and A. M. Gole. Vsc transmission limitations imposed by ac system strength and ac impedance characteristics. In *10th IET International Conference on AC and DC Power Transmission (ACDC 2012)*, pages 1–6, Dec 2012.

[5] C. Guo, C. Zhao, R. Iravani, H. Ding, and X. Wang. Impact of phase-locked loop on small-signal dynamics of the line commutated converter-based high-voltage direct-current station. *IET Generation, Transmission Distribution*, 11(5):1311–1318, 2017.

[6] J. Z. Zhou, H. Ding, S. Fan, Y. Zhang, and A. M. Gole. Impact of short-circuit

ratio and phase-locked-loop parameters on the small-signal behavior of a vsc-hvdc converter. *IEEE Transactions on Power Delivery*, 29(5):2287–2296, Oct 2014.

[7] J. A. Suul, S. D'Arco, P. Rodrguez, and M. Molinas. Impedance-compensated grid synchronisation for extending the stability range of weak grids with voltage source converters. *IET Generation, Transmission Distribution*, 10(6):1315–1326, 2016.

[8] Ajinai. Improved HVdc dynamic performace via phase locking to virtual Thvenin voltage. Master's thesis, University of Manitoba, 2017.

[9] A. M. Gole, V. K. Sood, and L. Mootoosamy. Validation and analysis of a grid control system using d-q-z transformation for static compensator systems. *Canadian Conference on Electrical AND Computer Engineering*, pages 745–748, Sept 1989.

[10] H. Abu-Rub, J. Guzinski, Z. Krzeminski, and H. A. Toliyat. Advanced control of induction motor based on load angle estimation. *IEEE Transactions on Industrial Electronics*, 51(1):5–14, Feb 2004.

[11] Y. Wang and V. Dinavahi. Low-latency distance protective relay on fpga. In *2014 IEEE PES General Meeting — Conference Exposition*, pages 1–1, July 2014.

[12] V. Dinavahi, R. Iravani, and R. Bonert. Design of a real-time digital simulator for a d-statcom system. *IEEE Transactions on Industrial Electronics*, 51(5):1001–1008, Oct 2004.

[13] I. Urriza, L. A. Barragan, J. I. Artigas, D. Navarro, J. Acero, R. Blasco, and O. Lucia. Word length selection method based on mixed simulation for digital pid controllers implemented in fpga. In *2008 IEEE International Symposium on Industrial Electronics*, pages 1965–1970, June 2008.

[14] Aniruddha Gole. Lecture notes in ECE7990: HVdc Transsmission-I, September 2017.

[15] Colin R Bayliss, Colin Bayliss, and Brian J Hardy. *Transmission and distribution electrical engineering*. Elsevier, 2012.

[16] AG Siemens. High voltage direct current transmission-proven technology for power exchange. *Energy Sector, Germany. Accessed at http://www. siemens. com/sustainability/pool/en/environmental-portfolio/products-solutions/power-transmission-distribution/hvdc_proven_technology. pdf*, 2011.

[17] M. M. Alharbi and M. L. Crow. Modeling of multi-terminal vsc-based hvdc system. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5, July 2016.

[18] Kamran Sharifabadi, Lennart Harnefors, Hans-Peter Nee, Staffan Norrga, and Remus Teodorescu. *Design, control, and application of modular multilevel converters for HVDC transmission systems*. John Wiley & Sons, 2016.

[19] A. Lesnicar and R. Marquardt. An innovative modular multilevel converter topology suitable for a wide power range. In *2003 IEEE Bologna Power Tech Conference Proceedings,*, volume 3, pages 6 pp. Vol.3–, June 2003.

[20] R. H. Park. Two-reaction theory of synchronous machines generalized method of analysis-part i. *Transactions of the American Institute of Electrical Engineers*, 48(3):716–727, July 1929.

[21] S. Gao and M. Barnes. Phase-locked loop for ac systems: Analyses and comparisons. In *6th IET International Conference on Power Electronics, Machines and Drives (PEMD 2012)*, pages 1–6, March 2012.

[22] Guan-Chyun Hsieh and J. C. Hung. Phase-locked loop techniques. a survey. *IEEE Transactions on Industrial Electronics*, 43(6):609–615, Dec 1996.

[23] W. C. Duesterhoeft, M. W. Schulz, and E. Clarke. Determination of instantaneous currents and voltages by means of alpha, beta, and zero components. *Transactions of the American Institute of Electrical Engineers*, 70(2):1248–1255, July 1951.

[24] David Bishop. *Fixed point package users guide*. VHDL Standards Working Group.

[25] The Microchip Technology Inc. *MCP3004/3008 Datasheet*, DS21295D, 2008.

[26] J. E. Volder. The cordic trigonometric computing technique. *IRE Transactions on Electronic Computers*, EC-8(3):330–334, Sept 1959.

[27] Y. H. Hu. Cordic-based vlsi architectures for digital signal processing. *IEEE Signal Processing Magazine*, 9(3):16–35, July 1992.

[28] Xilinx. *LogiCORE IP CORDIC v4.0, DS249*, 2011.

[29] Y. H. Hu. The quantization effects of the cordic algorithm (coordinate rotation digital computer). In *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*, pages 1822–1825 vol.3, April 1988.

[30] Sang Yoon Park and Nam Ik Cho. Fixed-point error analysis of cordic processor based on the variance propagation formula. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(3):573–584, March 2004.

[31] S. A. Allman, A. M. Gole, A. Neufeld, and E. Dirks. Application specific integrated circuits for the transformation from three phase ac to dqz coordinates. *IEEE Transactions on Power Delivery*, 3(4):1523–1529, Oct 1988.

[32] RTDS Technologies. *REAL TIME DIGITAL SIMULATION FOR THE POWER INDUSTRY MANUAL SET(RSCAD TUTORIAL MANUAL)*, July 07, 2017.

[33] J. D. Ainsworth. Harmonic instability between controlled static convertors and a.c. networks. *Electrical Engineers, Proceedings of the Institution of*, 114(7):949–957, July 1967.

[34] Xiao Jiang and A. M. Gole. A frequency scanning method for the identification of harmonic instabilities in hvdc systems. *IEEE Transactions on Power Delivery*, 10(4):1875–1881, Oct 1995.

[35] M. Szechtman, T. Wess, and C. V. Thio. A benchmark model for hvdc system studies. In *International Conference on AC and DC Power Transmission*, pages 374–378, Sep 1991.

[36] W. Ren, M. Sloderbeck, M. Steurer, V. Dinavahi, T. Noda, S. Filizadeh, A. R.

Chevrefils, M. Matar, R. Iravani, C. Dufour, J. Belanger, M. O. Faruque, K. Strunz, and J. A. Martinez. Interfacing issues in real-time digital simulators. *IEEE Transactions on Power Delivery*, 26(2):1221–1230, April 2011.

[37] C. Choi and W. Lee. Analysis and compensation of time delay effects in hardware-in-the-loop simulation for automotive pmsm drive system. *IEEE Transactions on Industrial Electronics*, 59(9):3403–3410, Sept 2012.

[38] S. Bibian and Hua Jin. Time delay compensation of digital control for dc switchmode power supplies using prediction techniques. *IEEE Transactions on Power Electronics*, 15(5):835–842, Sept 2000.

[39] G. Byeon, S. Oh, and G. Jang. A new dc offset removal algorithm using an iterative method for real-time simulation. *IEEE Transactions on Power Delivery*, 26(4):2277–2286, Oct 2011.

[40] Adel S Sedra and Kenneth Carless Smith. *Microelectronic circuits*, volume 1. New York: Oxford University Press, 1998.

[41] Xilinx. *7 Series FPGAs Configurable Logic Block User Guide*, UG474 (v1.8) September 27, 2016.

[42] Xilinx. *7 Series FPGAs DSP48E1 Slice User Guide*, UG479 (v1.10) March 27, 2018.

[43] Xilinx. *Synthesis and Simulation Design Guide*, UG626 (v 11.4) December 2, 2009.

[44] Xilinx. *Vivado Design Suite User Guide Using Constraints*, UG903 (v2014.3) October 31, 2014.