BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING

LINEAR SWEEP CONTROLLER DESIGN AND FPGA IMPLEMENTATION FOR FMCW RADAR APPLICATIONS

BURAK DURSUN

MASTER OF SCIENCE THESIS

ANKARA

FEBRUARY, 2012

LINEAR SWEEP CONTROLLER DESIGN AND FPGA IMPLEMENTATION FOR FMCW RADAR APPLICATIONS

FMCW RADAR UYGULAMALARINA YÖNELİK DOĞRUSAL TARAMA DENETİMCİSİ TASARIMI VE FPGA GERÇEKLEŞTİRİMİ

BURAK DURSUN

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Department of Electrical and Electronics Engineering
at Başkent University

FEBRUARY, 2012

This thesis has been approved in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN ELECTRICAL AND ELECTRONICS ENGINEERING** on 24/02/2012 by,

Member (Supervisor)	:…………………………………..
	Yrd. Doç. Dr. Mustafa Doğan
Member	:…………………………………..
	Doç. Dr. Şimşek Demir
Member	:…………………………………..
	Doç. Dr. Hamit Erdem

APPROVAL

...../02/2012

Prof. Dr. Emin AKATA

DIRECTOR

INSTITUTE OF SCIENCE AND ENGINEERING

To my Wife (Zeynep Işıl IŞIK DURSUN) and Family.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my Supervisor Assist. Prof. Dr. Mustafa DOĞAN for his guidance and trust on me throughout the thesis. I thank to my wife for her unlimited support during the long study period.

ABSTRACT

LINEAR SWEEP CONTROLLER DESIGN AND FPGA IMPLEMENTATION FOR FMCW RADAR APPLICATIONS

Burak DURSUN

Başkent University Institute of Science and Technology

Department of Electrical and Electronics Engineering

Frequency Modulated Continuous Wave (FMCW) radar systems rely on linear frequency sweep and Voltage Controlled Oscillators (VCO) are being in use for the sweep generation. However, developing VCO with linear transfer functions at microwave frequencies is a challenging study and availability of such components is limited. Especially for the long-distance FMCW radar applications, the non-linear frequency sweep due to the VCO insufficiency, results in range resolution degradation. In order to develop a solution for this problem; a case study is completed for an FMCW Radar Altimeter at 4.3 GHz center frequency with 62.5 Hz triangle wave modulated 152.6 MHz frequency sweep. Open-loop control solution like voltage pre-distortion method are not adequate for the application like FMCW Radar Altimeter because of the noisy and non-stationary structure of the system due to environmental condition. In these situations closed-loop control systems like Phase Lock Loop (PLL) provide more adequate results. In this research; a PLL structure is implemented on an Field Programmable Gate Array (FPGA) by a Direct Digital Synthesizer (DDS) as the Reference Signal Generator, an improved Phase-Frequency Detector as the Phase Detector and a Kalman Filter as the Loop Filter. For design verification and analysis FPGA-In-the-Loop (FIL) simulations are generated by the use of Mathworks Simulink and Xilinx System Generator.

Keywords: FMCW Sweep Linearization, PLL, Kalman Filter, Model Based FPGA Design, FPGA-In-the-Loop

Advisor: Assist. Prof. Dr. Mustafa DOĞAN, Control Engineering Department, Doğuş University

ÖZ

FMCW RADAR UYGULAMARINA YÖNELİK DOĞRUSAL TARAMA DENETİMCİSİ TASARIMI VE FPGA GERÇEKLEŞTİRİMİ

Burak DURSUN

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Elektrik ve Elektronik Mühendisliği Bölümü

Sıklık Kiplenimli Sürekli Dalga (FMCW) radar sistemleri doğrusal frekans taramasına dayanmaktadır ve frekans taraması Voltaj Kontrollü Osilatörler (VCO) kullanılarak gerçekleştirilmektedir. Ancak mikrodalga frekanslarda doğrusal transfer fonkisyonuna sahip VCO geliştirmek oldukça zorlu bir çalışmadır ve bu komponentler yaygın değillerdir. Özellikle uzak mesafe FMCW radar uygulamalarında, frekans taramasının VCO yetersizliği nedeni ile doğrusal olmaması, menzil çözünürlüğünü olumsuz yönde etkilemektedir. Bu soruna çözüm üretmek amacı ile; 4.3 GHz merkez frekansında, 62.5 Hz üçgen dalga kiplenimli olarak 152.6 MHz frekans taraması gerçekleştiren bir FMCW Radar Altimetreye yönelik çalışma gerçekleştirilmiştir. Voltaj Ön-Bozulma yöntemi gibi açık döngü kontrol cözümleri, FMCW Radar Altimetre uygulamalarıgibi, çevresel koşullar nedeni ile gürültülü ve durağan olmayan sistem yapıları için yetersiz kalmaktadır. Bu şartlar altında Evre Kenetleme Döngüleri (PLL) gibi kapalı döngü kontrol sistemleri ile daha verimli neticeler elde edilmektedir. Bu çalışmada Alanda Programlanabilir Kapı Dizileri (FPGA) ile gerçekleştirimi yapılmış olan PLL yapısında; Referans İşaret Üreteci olarak Doğrudan Sayısal Sentezleyici (DDS), Evre Algılayıcı olarak geliştirilmiş bir Evre-Sıklık Algılayıcı (PFD) ve Döngü Süzgeci (LF) olarak Kalman Süzgeci kullanılmıştır. Tasarım doğrulama ve sonuçların analizi için MathWorks Simulink ve Xilinx System Generator kullanılarak Döngü-İçinde-FPGA (FIL) simülasyonları gerçekleştirilmiştir.

Anahtar Kelimeler: FMCW Tarama Doğrusallaştırma, PLL, Kalman Süzgeci, Model Tabanlı FPGA Tasarımı, Döngü-İçinde-FPGA

Danışman: Yrd. Doç. Dr. Mustafa DOĞAN, Kontrol Mühendisliği Bölümü, Doğuş Üniversitesi

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Frequency Modulated Continuous Wave (FMCW) radar sytems are widely in use from automotive sensors to avionic altimeters. These systems rely on linear frequency sweep and Voltage Controlled Oscillators (VCO) are being in use for the sweep generation. However, developing VCO with linear transfer functions at microwave frequencies is a challenging study and availability of such components is limited. Especially for the long-distance FMCW radar applications, the non-linear frequency sweep due to the VCO insufficiency, results in range resolution degradation. There are several post-compensation hardware methods described in literature as solutions for this problem which are introducing additional circuitry. On the other hand; in addition to the RF front end, FMCW radar systems include digital signal processing and data processing units in order to output the detection to an interface unit. This interface unit may be a display or a communications protocol that is being used in the top level system which is including the FMCW radar sub-system. This thesis work is concentrated on developing an embedded solution for sweep linearisation of an FMCW Radar Altimeter which can also be integrated with the digital signal processing and the data processing blocks of the whole system in a single platform like an Field Programmable Gate Array (FPGA).

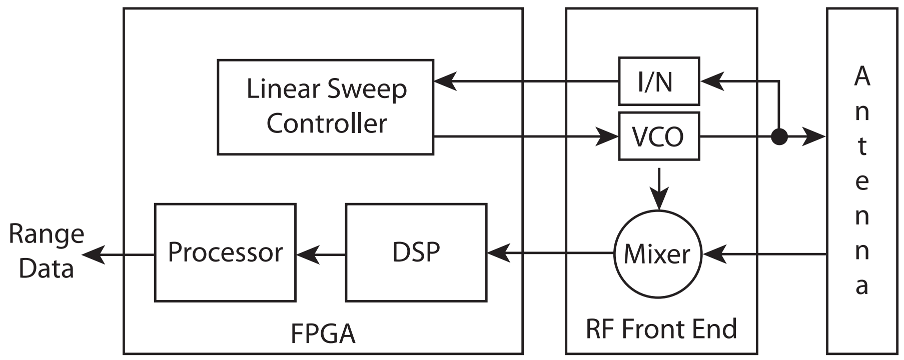


Figure 1.1: Proposed System Block Diagram

This thesis is divided into seven chapters. After introducing the problem that is being worked on briefly in this chapter, several conventional solutions are described in Chapter 2. In Chapter 3, the developed architecture is explained by discussing on parameters of each design block. Chapter 4 deals with model based implementation of the architecture on FPGA. Chapter 5 demonstrates the various simulations of the design while Chapter 6 is indicating the improvement with the summary of work. Finally in Chapter 7, a discussion is made about the architecture, followed by some recommendation for some future work. VHDL codes are given in Appendix A and Matlab codes are given in Appendix B.

1.1 FMCW Principles

FMCW radar systems, rely on the principle of transmitting a modulated signal with a triangular frequency envelope to the target and recieving back the reflected signal. The difference between the transmitted and received signals which is called the beat signal, projects the ranging information. In Figure 1.2, the time variations of the frequency modulated transmitted signal $x(t)$ and the received signal reflected back from the target ($y(t)$) are shown.

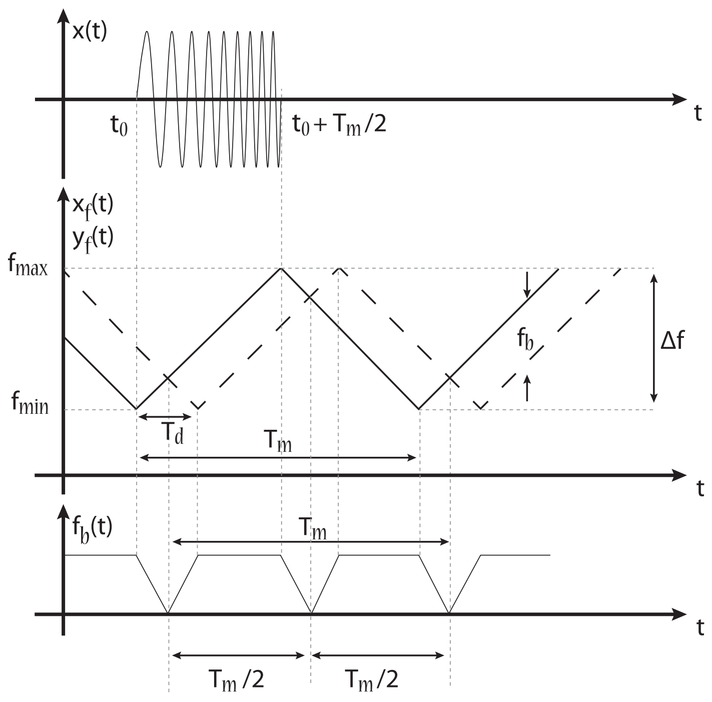


Figure 1.2: FMCW Radar Signals

$$x_f(t) = f_{min} + kt, \qquad \left[t_0, t_0 + \frac{T_m}{2}\right] \tag{1.1}$$

$$x(t) = \sin\left[2\pi\left(f_{min} + \frac{k}{2}(t - t_0)\right)(t - t_0)\right], \qquad \left[t_0, t_0 + \frac{T_m}{2}\right] \tag{1.2}$$

Modulation frequency:

$$f_m = \frac{1}{T_m} \tag{1.3}$$

Frequency Deviation:

$$\Delta f = f_{max} - f_{min} \tag{1.4}$$

Center Frequency:

$$f_0 = \frac{f_{min} + f_{max}}{2} \tag{1.5}$$

Time Delay:

$$T_d = \frac{2R}{c} \tag{1.6}$$

Beat Frequency:

$$f_b = \left| x_f - y_f \right| \tag{1.7}$$

The Slope of the Sweep:

$$k = \frac{\Delta f}{\frac{T_m}{2}} = 2\, f_m \Delta f = \frac{f_b}{T_d} \tag{1.8}$$

The beat frequency f_b is generated by the Fast Fourier Transform (FFT) of the beat signal (f_b) and the distance to the target (R) is measured with a resolution of A (1.10).

$$f_b = 2 f_m \Delta f\, T_d = 2 f_m \Delta f\, \frac{2R}{c} = A \cdot R \tag{1.9}$$

The Range Sensitivity:

$$A = \frac{4\, f_m \Delta f}{c} \tag{1.10}$$

1.2 VCO Non-Linearity

For generation of high frequency $x(t)$ signals LC oscillator based VCO structures are common.

$$C_{eff} = C_0 - \frac{1}{2} C_2 \tag{1.11}$$

where C_0 is the time average capacitance value and C_2 is the second order Fourier Coefficients of the nonlinear varactor driven by the oscillation.

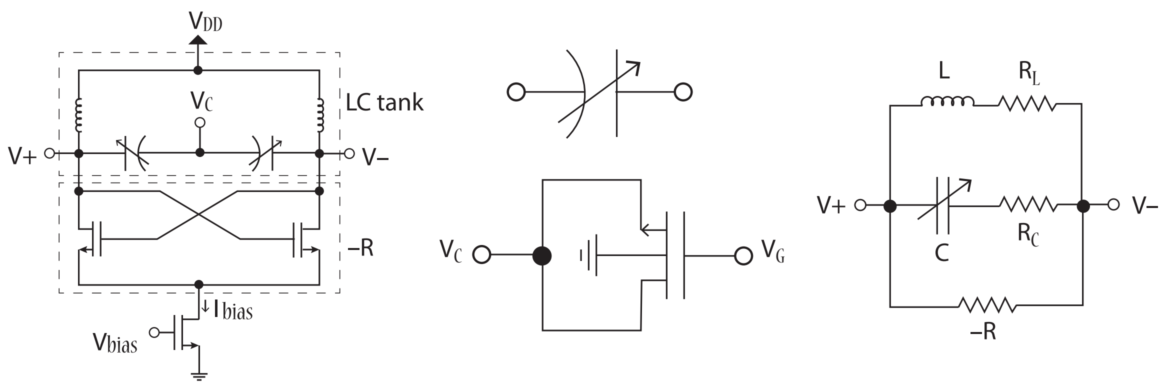
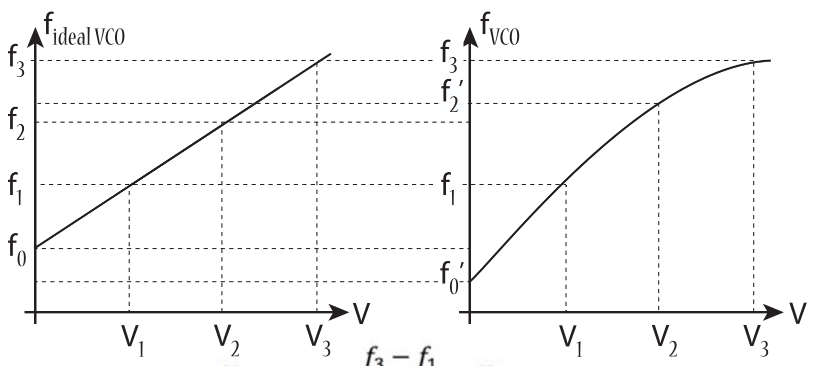


Figure 1.3: LC Oscillator Schematic and Model



$$K_{ideal\,VCO} = \frac{f_3 - f_1}{V_3 - V_1} \neq K_{VCO}$$

$$K_{ideal} = \frac{f_3 - f_1}{V_3 - V_1} \neq K_{VCO}$$

Figure 1.4: VCO Transfer Functions

$$V_{eff} = V_G - V_C - V_T \tag{1.12}$$

$$K_{VCO} = \frac{\delta f}{\delta V_c} = \frac{1}{2}\frac{f}{C_{eff}}\frac{\delta C_{eff}}{\delta V_{eff}} \tag{1.13}$$

The varactors in the oscillator structures are not operating linearly as it is given in (1.13) and described in [1]. Thus transfer functions of VCO are nonlinear as shown in Figure 1.4 [2].

CHAPTER 2

VCO LINEARIZATION METHODS

There are various methods in the literature dealing with the VCO non-linearity problem. Some of them are working on the oscillator structures or the varactors which are the main reason of non-linearity, while the others are working on post-compensation methods with the knowledge of operating with a non-linear component. There are open-loop control solutions like Voltage Pre-Distortion by which VCO transfer function (F) analysis is done for designing a compensator with the inverse transfer function of the VCO F^{-1} [3]. However in this thesis, FMCW Radar Altimeter is selected as the case study which is not suitable to apply an open-loop control design because of the non-stationary environmental conditions. Thus it is concentrated on closed loop structures like Phase Lock Loop (PLL) in this research.

2.1 Voltage Pre-Distortion

In this approach, a measurement of the VCO voltage–frequency characteristic is required. The inverse of the function may then be used to compute the necessary pre-distorted voltage to be applied to the VCO tuning node by a waveform generator or equivalent circuitry.

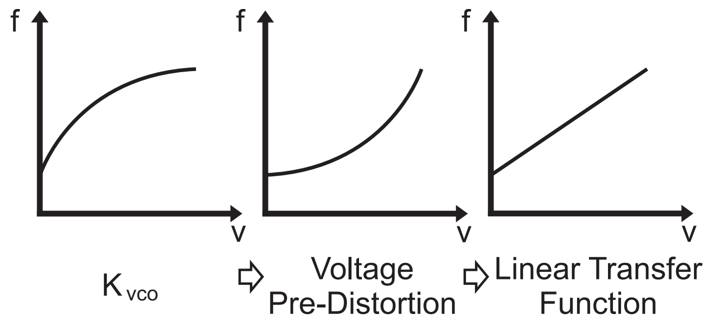


Figure 2.1: Voltage Pre-Distortion

The major drawback of this method is that typically the VCO characteristic is not constant but varies with environmental conditions, such as ambient temperature. This results in a poor phase noise behavior and phase deviations. Semiconductor oscillators therefore generally need some means of stabilization, thus a phase-locked loop is a better alternative to the simple predistortion approach [4], [5], [6].

2.2 Phase Lock Loop

A PLL is a control loop used to synchronize its output signal which is generated by a voltage or numerical controlled oscillator, with an input or reference signal in frequency and in phase. Several variations of PLL exists, as the common ones are analog phase-locked loop (APLL) which is also referred to as a linear phase-locked loop, digital phase-locked loop (DPLL) and all digital phase-locked loop (ADPLL). These classifications are done according to the internal blocks of the structure as they are implemented in digital or analog. Considering the significant advantages of digital systems over their analog counterparts such as superiority in performance, speed, reliability and reduction in size and cost, DPLLs attract much attention compared to APLLs. However, the analog APLLs are still widely used [7], [8], [9], [10].

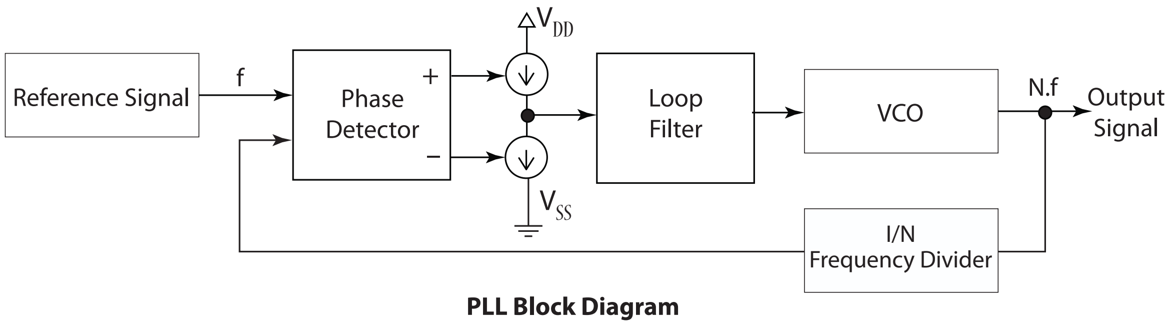


Figure 2.2: Typical Charge-Pump PLL Block Diagram

In Figure 2.2 the block diagram of a typical PLL circuit is shown. As seen in the figure, the PLL consists of a Reference Signal Generator, a VCO, a Frequency Divider, a Phase Detector, a Charge-Pump which can be defined as a part of Phase Detector and a Loop Filter.

2.2.1 Reference signal

Due to the difficulties of developing a high frequency oscillator with a stable output and good noise properties, PLL circuits are widely in use by generating the required output by available high quality low frequency oscillators. Crystal oscillators have adequate properties as reference signal sources but without a PLL structure it is not possible to use them because of low frequency outputs and limited frequency tuning [8]. The transfer function of a PLL is analysed in Appendix D.

2.2.2 Phase detector

The Phase Detector compares the phases of the input and output signals, and generates an error signal proportional to the phase deviation between them. There are several ways of implementing a phase detection unit like analog solutions that are using mixers or digital solutions that are using exclusive-or gates. But due to the advantages like false lock condition avoidance, Phase-Frequency Detector (PFD) implementation is the most popular way for phase detection. Figure 2.3 shows a typical block diagram of a PFD with D-Type Flip-Flop structure.

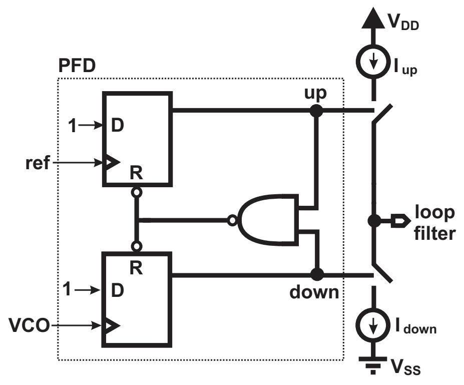


Figure 2.3: Implementation of PFD and Charge-Pump

The PFD is triggered by the edges of the reference and VCO signals and it is reset once both edges have been detected. In Figure 2.4, typical waveforms in the phase-frequency detector are shown. In the case that the VCO runs too slow, the edges of the VCO signal come after the edges of the ref signal. This condition generates an up pulse, increasing the frequency of the VCO. Consequently, in the case that the edges of the VCO signal come before the ref signal, a down pulse is generated which decreases the frequency of VCO.

The switchable current sources in the Charge-Pump are controlled by the up and down signals. As the switches close, charge is sinked from Loop Filter or sourced to it.

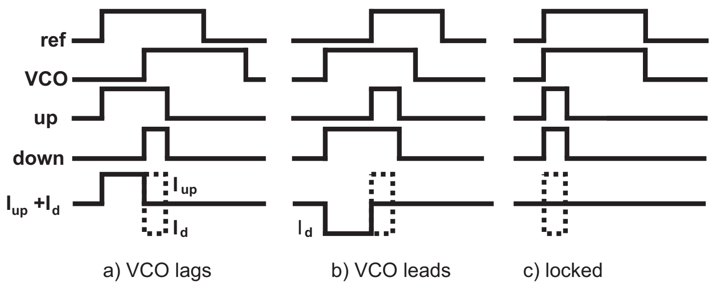


Figure 2.4: VCO Feedback Signal Lags, Leads and Signals in Phase

2.2.3 Loop filter

The output of the phase detector includes a dc term that is proportional to the phase error, and some high frequency components. The higher frequency terms are filtered out by the low pass in order to avoid disturbance. Both active and passive filters can be used as Loop Filter. Main topologies of these filters are given in Figure 2.5. Transfer functions of the basic and most commonly used passive loop filter topologies are analysed in Appendix C.

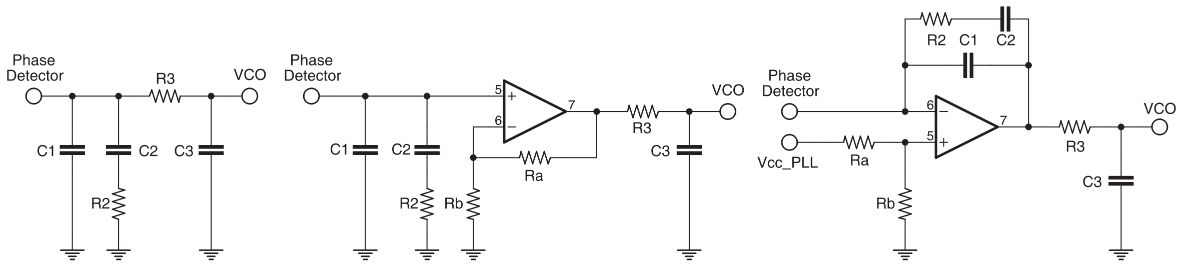


Figure 2.5: Passive and Active Loop Filters

In APLLs active filters are mainly used for the improvement of lock-in and hold-in ranges. Hold-in range is the maximum frequency step of the reference signal that will not disturb the stability of the locked system while the lock-in range is the range of frequencies within the system that the reference signal can be acquired and tracked. Since there are no lock-in or hold-in range problems in PFD, passive filters are satisfactory. Thus there is no need for active filters since active devices increase cost, complexity and noise [7].

2.2.4 Voltage controlled oscillator

A Voltage Controlled Oscillator (VCO) generates a periodic oscillator with a transfer function given in Figure 2.6.

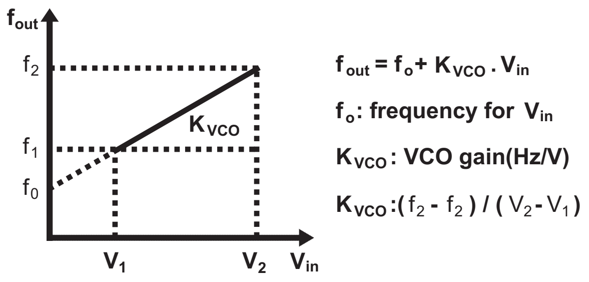


Figure 2.6: VCO Transfer Function

Figure 5.5 shows the model of a VCO. According to the Barkhousen stability criteria if the loop gain (GH) is equal to unity then the output oscillation will be a sinusoidal wave.

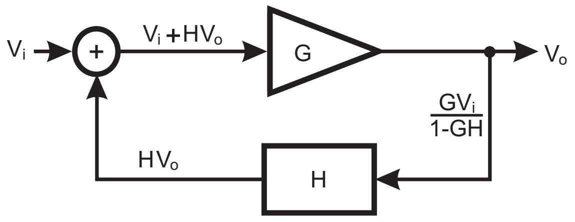


Figure 2.7: VCO Model

The actual clock which is generated by a PLL, comes from the VCO which generates a periodic oscillation. The frequency of this periodic oscillation can be controlled using the modulation of some control voltage. The control voltage corresponds to some filtered form of the phase error, in a PLL. In response to this correspondence, the VCO adjusts its frequency.

2.2.5 Frequency divider

Since reference signal is in low frequency range, in order to check the phase difference of VCO output and the reference signal, a Frequency Divider should be in use. The frequency range of the PLL circuit is also limited by the digital divider in these circuits. A practical limit to the frequency range is put by the number of bits in the divider control logic. The counter implementation will limit both the minimum and maximum achievable division ratio in divider implementation. Fundamental limits to the possible number of bits in the divider do not exist. However, in order not to increase circuit complexity and power consumption, divider should be implemented with adequate number of bits for a correct PLL operation.

CHAPTER 3

ARCHITECTURE OF THE SYSTEM

Conventional types of PLL structures are described in Section 2.2. However this thesis work does not deal exactly with one of them but demonstrates a PLL structure between DPLL and ADPLL. Since in the case study that is being worked on operates at high frequencies, using a Numerically Controlled Oscillator (NCO) is not possible. So even the motivation is designing a system by digital implementations of components; due to the operational frequency, system should involve a high frequency VCO and a high frequency divider. As conventional DPLL, this design also includes a Phase-Frequency Detector but with some improvements that are described in Section 3.2. On the other hand, digital designs of reference signal sources and Loop Filter structures are not common. These topics are also detailed in this chapter.

3.1 Reference Signal Generation

The reference signal source is implemented by a Numerically Controlled Oscillator (NCO). The architecture of this structure has a great influence on the performace of the loop. It is possible to implement a NCO by a divide-by-n counter which divides the high clock frequency by n in order to generate a lower frequency output. Since Direct Digital Synthesis (DDS) method provides much better frequency resolution then such a simple counter, NCO is designed by a DDS structure.

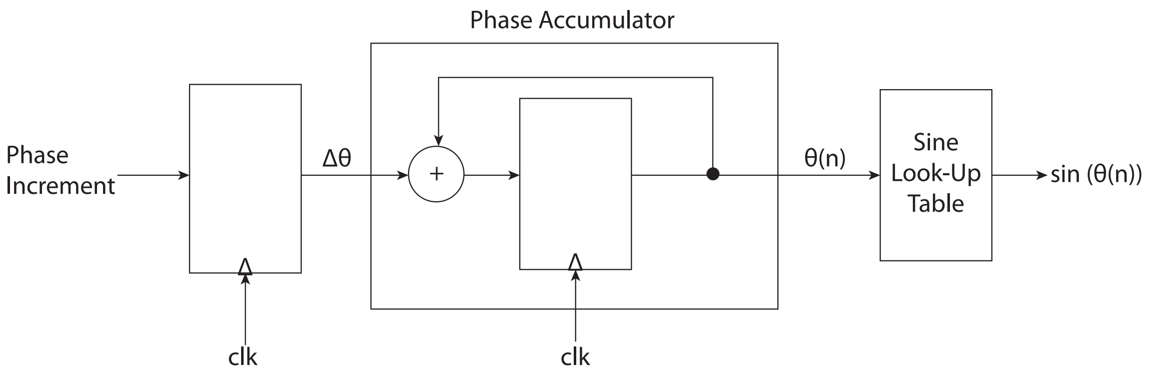


Figure 3.1: DDS Block Diagram

As shown in Figure 3.1 a DDS block typically consists of a Phase Increment block for determining the output frequency, a Phase Accumulator which maps the instantaneous phase of output signal and a Look-Up Table (LUT) in order to generate a sinusoidal output signal. In the case that is being worked on, since a sinusoidal reference signal is not required, LUT is not implemented. So, the output generates as a ramp signal at a frequency of Phase Increment ($\Delta\theta$) times the resolution as shown on Equation 3.1. Frequency resolution can be define as frequency over two to the power of Phase Accumulator bits.

$$f_{out} = \frac{f_{clk}\Delta\theta}{2^{bit}} \tag{3.1}$$

If the reference signal is generated by a crystal oscillator as in the conventional cases; then the beat frequency signal ($f_b(t)$) will be a smooth DC level during transmitted signal ($x(t)$) and received signal ($y(t)$) are both ramping or de-ramping. However, a digitally generated reference signal will have a stair-like shape. Thus beat frequency signal will also have same steps with the reference signal as it is shown in Figure 3.2.

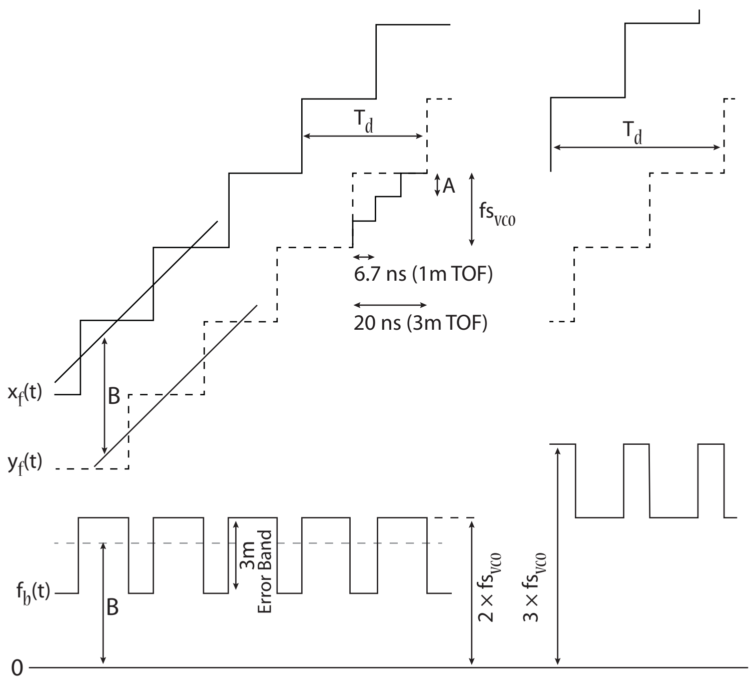


Figure 3.2: Effects of DDS to Beat Signal

Since the beat signal frequency projects the target distance, FFT algorithm is extremely important for high resolution frequency detection which defines higher resolution in target distance measurement. As the FFT algorithm may cause a lower resolution measurement independent from the Range Sensitivity (A) parameter of FMCW radar principles, DDS effect on the beat frequency also causes Range Resolution degradation. Thus while choosing the Range Resolution of a design; equation (1.10), FFT algorithm and DDS resolution should be considered together. On the other hand, since the clock frequency is selected as 50 MHz in the design, time interval of each frequency step of reference signal will be 20 ns. So in this case, since the time of flight (TOF) of a microwave signal to a target in 1 meter distance, accuracy of the measurement may only be 3 meters. For these reasons, Modulation Frequency (f_b) and Frequency Deviation (Δf) parameters of Range Sensitivity (A) are associated with the number of bits and clock rate of DDS during the design process.

Most FMCW Radar Altimeters operate at approximately 150 MHz bandwidth in 4.3 GHz center frequency with a modulation frequency between 50 Hz and 300 Hz. Consequently, the Range Sensitivity is mostly between 100 Hz/m and 600 Hz/m. Even though it seems that 600 Hz/m provides a higher Range Resolution; since the main goal of this research is the linear frequency sweep control, low frequency steps have been preferred. The main reason for this decision is that the Loop Filter used after the Phase Detector will smooth the frequency steps, as described in [11]. As the low frequency steps are filtered better, a linearised VCO transfer function has been obtained. Since the smoothing deformed both time intervals and frequency steps, the measurement accuracy is improved as well as the Range Resolution is converged to the Range Sensitivity.

When the DDS resolution is chosen as 27-bits, the reference signal frequency steps become 0.327529 Hz, as indicated on equation (3.2). The VCO output signal frequency is scaled down to MHz level by the Frequency Divider with the ratio of 1/1024 ($N = 2^{10}$) in order to be comparable with reference frequency. So as it is shown in equation (3.2), VCO output signal frequency steps become 381.47 Hz. Modulation frequency is selected as 62.5 Hz while frequency bandwidth is defined as 152.588 MHz. According to these selections, Range Sensitivity is calculated as 127.245 Hz/m in equation (3.5). If DDS resolution was selected higher, then Range Sensitivity will be under 100 Hz/m limit. For example, 28-bits DDS resolutions corresponds to 63.2223 Hz/m Range Sensitivity. Thus 27-bits resolution for the DDS architecture is ideal for this design.

$$fs_{DSS} = \frac{f_{clk}}{2^{bit}} = \frac{50\,\mathrm{MHz}}{2^{27}} = 0.327529\,\mathrm{Hz} \tag{3.2}$$

$$fs_{VCO} = fs_{DSS} \cdot N = 381.47\,\text{Hz} \tag{3.3}$$

$$\Delta f = fs_{VCO} \overbrace{\frac{1}{2 f_m \frac{1}{f_{clk}}}}^{\text{step count}} \tag{3.4}$$

When we place the equation (3.2), (3.3) and (3.4) in equation (1.10), we will get equation (3.2).

$$A = \frac{4 f_m \frac{f_{clk}}{2^{bit}} N \frac{1}{2 f_m \frac{1}{f_{clk}}}}{c} = \frac{1.70785\, e^{10}}{2^{bit}} \xRightarrow{27-bit} 127.245\,\text{Hz/m} \tag{3.5}$$

3.2 Phase Detection

In order to avoid false lock condition, the phase detector has been designed like a Phase-Frequency Detector as described in Section 2.2.2. But this design includes some improvements over a conventional PFD. Since a PFD outputs the phase difference in $(0, 2\pi)$ range related with the signal at lower frequency, for some initial conditions as shown in Figure 3.3, phase locking may take longer.

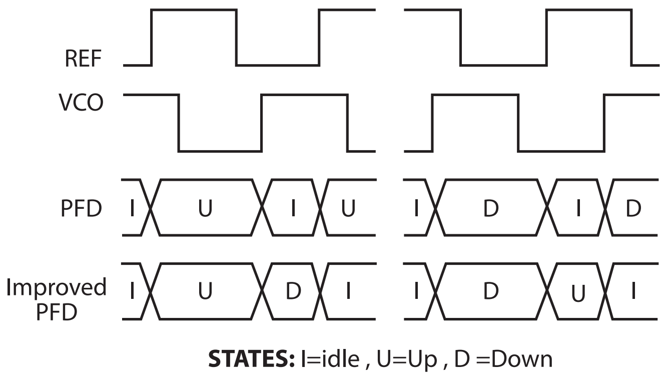


Figure 3.3: Time Diagrams of Conventional and Improved PFD

Figure 3.4 shows a PFD with D-Type Flip-Flop structure and the three-state system model of this structure. Furthermore, the contribution of the structure that is developed in this study is indicated with bold lines in this figure. As shown on the model, the direct transition between states up and down in PFD architecture. However, in the developed structure, if one of the input signals rises when the other is low, a transition is occur between the states up and down.

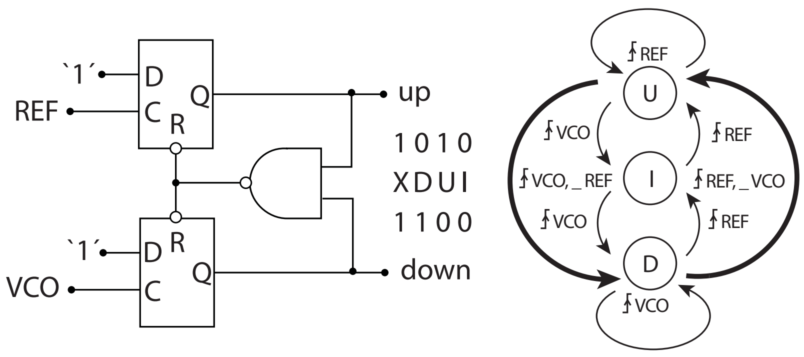


Figure 3.4: D-Type Flip-Flop PFD and PFD Models

The Charge-Pump is implemented by an up-down counter which acts identically with the common transistor topology. It is possible to model transistor widths by implementing a variable count rate counter but it is also possible to multiply the output of counter with a constant that will symbolize the transistor width.

3.3 Loop Filtering

Loop Filters are basically low pass filters, however corner frequency decision is extremely critical. If there is a modulation on the reference signal, it may not be efficient to use a filter with a fixed bandwidth. Therefore a filter with an adaptive bandwidth in behaviour has been designed. The designed filter is actually a Kalman Filter that operates independently from the frequency domain characteristics.

The Kalman Filter is one of the most well known and widely used mathematical tool for stochastic estimation from noisy sensor measurements. In the case that is being worked throughout the thesis, the noisy measurement is Phase Detector output which projects lots of different system noise parameters that are tending to behave as a gaussian noise all together. The Kalman Filter is a set of mathematical equations implementing a predictor-corrector type estimator which is optimal in the sense that minimizes the estimated error covariance. In the following sub-sections computational and probabilistic approach of the filter and the algorithm is explained [12] [13].

3.3.1 Computational and probabilistic approaches

The Kalman Filter is used to try to estimate the state $x \in \Re^n$ of a discrete-time controlled process. The following linear stochastic difference equation determines the mentioned process (3.6).

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{3.6}$$

The $x \in \Re^m$ measurement of the process is

$$z_k = Hx_k + v_k \tag{3.7}$$

The process and measurement noise is respectively presented by the random variables w_k and v_k. Each of these variables are assumed to be independent than other variables, carry normal probability distributions and are white in characteristics. In another words, each of these variables are assumed to be independent than other variables, they are characteristically white, and with normal probability distributions;

$$p(w) \sim N(0, Q) \tag{3.8}$$

$$p(v) \sim N(0, R) \tag{3.9}$$

It is possible that the process noise covariance Q and measurement noise covariance R matrices to change according to each measurement or time step. Hence, in this process we assume that they are constant.

The state at the previous time step $k - 1$ is related to the state at the current step k by the nxn matrix A in the difference equation (3.6), in any possible absence of a driving function or process noise. It should be kept in mind that in practice A might change according to each time step, hence in this process we assume it to be constant. The optional control input $u \in \Re^l$ is related to the state x by the $nx1$ matrix B. The state is related to the measurement z_k by the mxn matrix H in the measurement equation (3.7). It is possible that H may change according to each time step or measurement, hence in this process we assume that it is constant.

$\hat{x}_k^- \in \Re^n$ is defined as a priori (before) state estimate at step k, as the process prior to step

k is known, and given the measurement z_k, $\hat{x}_k \in \Re^n$ is the posteriori (after) state estimate at step k. Accordingly, the priori and the posteriori estimate errors can be defined as in equations (3.10) and (3.11).

$$e_k^- \equiv x_k - \hat{x}_k^- \tag{3.10}$$

$$e_k \equiv x_k - \hat{x}_k \tag{3.11}$$

Then, the priori estimate error covariance is

$$P_k^- = E\left[e_k^- e_k^{-T}\right] \tag{3.12}$$

and the posteriori estimate error covariance becoming

$$P_k = E\left[e_k e_k^T\right] \tag{3.13}$$

Trying to find an equation that computes the posteriori state by using a linear combination of the priori estimate $\hat{x}_k^-$ and a weighted difference between an actual measurement z_k and a prediction $H\hat{x}_k^-$ of the measurement is the beginning of the derivation of the Kalman Filter.

$$\hat{x}_k = \hat{x}_k^- + K\left(z_k - H\hat{x}_k^-\right) \tag{3.14}$$

The result of the difference $(z_k - H\hat{x}_k^-)$ in equation (3.14) is named as the measurement innovation or residual. This difference (residual) reflects the discrepancy between the actual measurement z_k and the predicted measurement $H\hat{x}_k^-$. Zero residual indicates that the predicted measurement and the actual measurement are in complete harmony.

The nxm matrix K in equation (3.14) is the blending factor or Kalman gain, which minimizes the posteriori error covariance which is given in equation (3.13). In order to accomplish this minimization, first, the equation (3.14) must be substituted into the above definition for e_k. Then, one must substitute that into equation (3.13). Then, the indicated expectations must be performed with taking the derivative of the trace of the result with respect to K. Then, that result must be set equal to zero, and then solving for K we get the equation (3.15).

$$K_k = P_k^- H^T \left(H P_k^- H^T + R\right)^{-1}$$
$$= \frac{P_k^- H^T}{H P_k^- H^T + R} \tag{3.15}$$

When we look at equation (3.15), it is seen that as the measurement error covariance R approaches zero, the gain weights the residual more heavily.

$$\lim_{R_k \to 0} K_k = H^{-1} \tag{3.16}$$

As the a priori estimate error covariance P_k^- approaches zero, the gain K weights the residual less heavily.

$$\lim_{P_k^- \to 0} K_k = 0 \tag{3.17}$$

Another way of thinking about the weighting by K is that as the measurement error covariance R approaches zero, the actual measurement z_k becomes more and more trusted while the predicted measurement $H\hat{x}_k^-$ becomes less and less trusted. On the other hand, as the a priori estimate error covariance Pk approaches zero the actual measurement k is trusted less and less, while the predicted measurement $H\hat{x}_k^-$ is trusted more and more.

According to Bayers' Rule the probability of the a priori estimate $\hat{x}_k^-$ depends on the conditions of the previous measurements z_k. The Kalman Filter maintains the first two moments of the state distribution as shown in equations (3.18) and (3.19).

$$E\left[x_k\right] = \hat{x}_k \tag{3.18}$$

$$E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right] = P_k \tag{3.19}$$

The a posteriori estimate covariance equation reflects the variant of the state distribution (the second non-central moment).

$$p(x_k|z_k) \sim N\left(E[x_k], E[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T]\right)$$
$$= N(\hat{x}_k, P_k) \tag{3.20}$$

3.3.2 Algorithm

The Kalman filter estimates the process state at some time and then obtains feedback in the form of noisy measurements. Relying on this, the equations for the Kalman Filter is divided into two groups as the time update equations and the measurement update equations. Time update equations project the state and the error covariance ahead to obtain the priori estimates for the next time step. These equations can be considered as predictor equations. Measurement update equations use the measurement as a feedback in order to obtain an improved posteriori estimate. These equations can be considered as corrector equations. As shown in Figure 3.5, the estimation algorithm can be defined as a cycle of predictor-corrector algorithm.

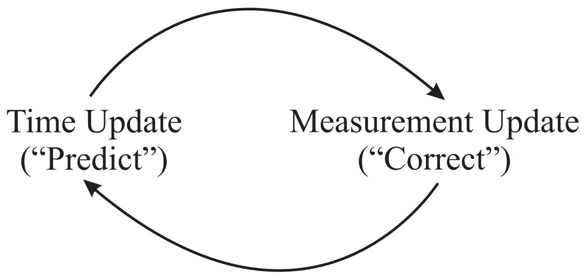


Figure 3.5: Discrete Kalman Filter Cycle

The specific equations for the time and measurement updates are given in equations (3.21) to (3.25)

$$\hat{x}_k^- = A\,\hat{x}_{k-1} + B\,u_k \tag{3.21}$$

$$P_k^- = A\,P_{k-1}\,A^T + Q \tag{3.22}$$

$$K_k = P_k^- H^T \left(H\,P_k^-\,H^T + R\right)^{-1} \tag{3.23}$$

$$\hat{x}_k = \hat{x}_k^- + K_k\,(z_k - H\,\hat{x}_k^-) \tag{3.24}$$

$$P_k = (1 - K_k H)\,P_k^- \tag{3.25}$$

As seen in equations (3.21) and (3.22), the time update equations project the state and the covariance estimates forward from the step $k - 1$ to step k. On the measurement update, computing the Kalman gain is the first step. Then by incorporating z_k measurement, using equation (3.24) generates a posteriori state estimate. Finally, a posteriori error covariance estimate should be obtained via equation (3.25). After each time and measurement update pair, the process is repeated with the previous posteriori estimates used to project or predict the new priori estimates.

$$\hat{x}_k^- = \hat{x}_{k-1} + u_k \tag{3.26}$$

$$P_k^- = P_{k-1} + Q \tag{3.27}$$

$$K_k = \frac{P_k^-}{P_k^- + R} \tag{3.28}$$

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - \hat{x}_k^-\right) \tag{3.29}$$

$$P_k = (1 - K_k)\, P_k^- \tag{3.30}$$

For the case of this thesis equations (3.21) to (3.25) were translated to equations (3.26) to (3.30).So the algorithm that was implemented for the Kalman Filter appears as shown in Figure 3.6.

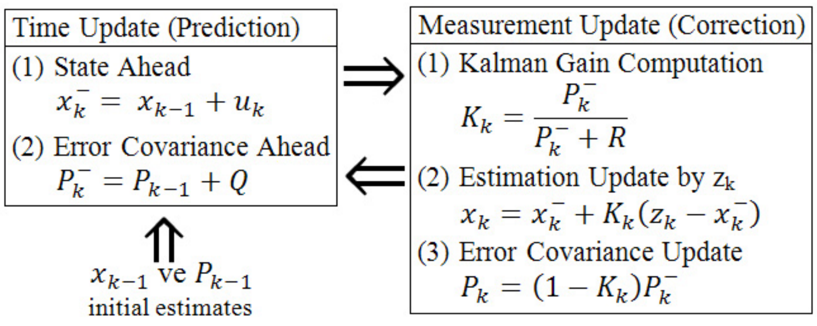

Time Update (Prediction)
(1) State Ahead
$x_k^- = x_{k-1} + u_k$
(2) Error Covariance Ahead
$P_k^- = P_{k-1} + Q$
x_{k-1} ve P_{k-1}
initial estimates
Measurement Update (Correction)
(1) Kalman Gain Computation
$K_k = \frac{P_k^-}{P_k^- + R}$
(2) Estimation Update by z_k
$x_k = x_k^- + K_k(z_k - x_k^-)$
(3) Error Covariance Update
$P_k = (1 - K_k)P_k^-$

Figure 3.6: Kalman Algorithm

CHAPTER 4

FPGA IMPLEMENTATION

Model based design methodology is used in this thesis. Xilinx ISE Design Suite is used for implementation and by the use of Xilinx System Generator, Mathworks Simulink is used as the main platform for top level system design. Several sub-blocks of system are designed in Xilinx ISE Project Navigator with VHDL (Very High Speed Integrated Circuit Hardware Description Language. In this chapter, the linear frequency sweep controller design is described by investigating each sub-block implementation. As it is indicated on Chapter 3, except the high frequency components of the system that are VCO and the Frequency Divider, all other system components are designed digitally inside the FPGA. The main blocks of controller includes a Direct Digital Frequency Synthesizer (DDFS) as the reference signal generator, a Phase-Frequency Detector for phase detection and the Kalman Filter as the loop filter.

4.1 Direct Digital Frequency Synthesizer

For digital oscillation at various frequencies a Numerically Controlled Oscillator (NCO) is designed in this block. Also a counter is implemented in order to generate a triangle waveform for the modulation of the digital signal that is generated by NCO.

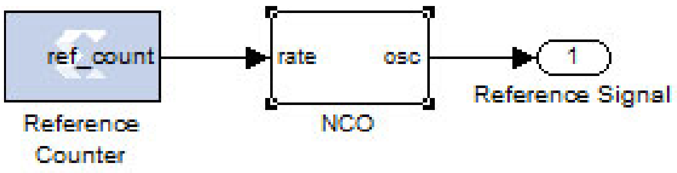


Figure 4.1: Direct Digital Frequency Synthesizer Block

The counter which is called the Reference Counter is designed with VHDL (Appendix A.1) and a configuration m-function (Appendix B.1) is prepared at MATLAB in order to integrate this sub-block to the rest of system at Simulink. Reference Counter is a 27-bits 50MHz up-down counter which changes the counting direction as it reaches xAF0D48 or xA8F2C8. The NCO block is implemented by the DDS Compiler 4.0 Xilinx Intellectual Property (IP) with 27-bits resolution in order to generate frequency steps at 0.372529

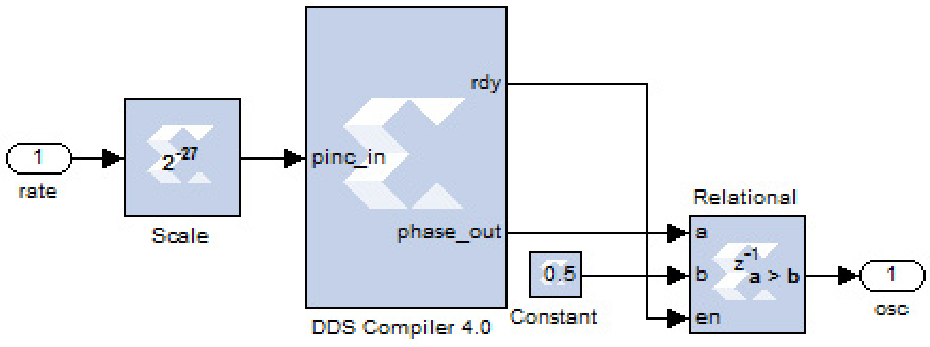


Figure 4.2: Numerically Oscillator Block

which is specified in Section 3.1. It is also calculated in Section 3.1 that the reference signal should modulate between the frequencies 4.12471 MHz and 4.27372 MHz. Thus moreover the Reference Counter upper and lower limits are specified, a Scale Block of Xilinx System Generator is used in order to replace the binary point of the data which will be the input of DDS. In hardware implementation, Scale block actually costs nothing. Since only the phase generator of the DDS Compiler 4.0 is implemented, DDS output is a periodic ramp signal. A Relation Block of Xilinx System Generator is used for generating a two-state digital oscillation as the NCO output.

4.2 Improved Phase-Frequency Detector

Phase detection is consisting both the phase detectors and the Charge-Pumps. Phase detector of this design is implemented with as a state machine in VHDL (Appendix A.2) in a Phase-Frequency Detector architecture with the improved capabilities described in Section 3.2. Charge-Pumps are implemented again in VHDL (Appendix A.3) as a 16-bits 50MHz up-down counter and called Phase Counter. M-functions that are required for Simulink integration are also given in Appendix B.2 and Appendix B.3 respectively. Also a Constant Multiplier Block of Xilinx System Generator is used for modelling transistor widths of the Charge-Pump.

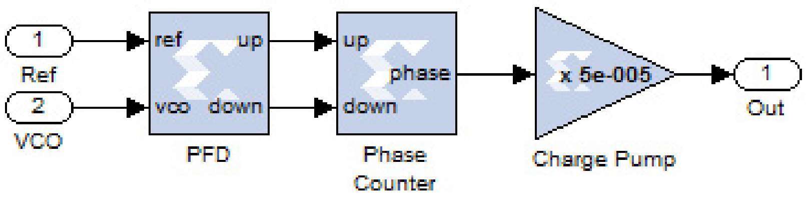


Figure 4.3: Improved Phase-Frequency Detector Block

4.3 Kalman Filter

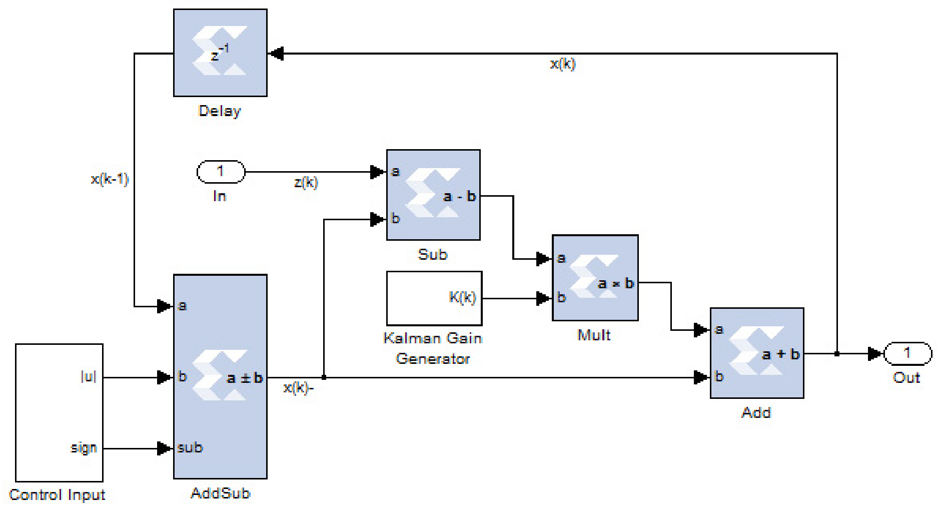


Figure 4.4: Kalman Filter Block

Kalman Filter implementation is completed according to the algorithm defined in Section 3.3.1.2. This block consists 2 Delay, 4 Add, 2 Sub, 1 Add/Sub, 1 Constant Multiplier and 2 Constant Blocks of Xilinx System Generator with a Divider Generator 3.0 Xilinx IP and a Control Input sub-block written in VHDL. As it is shown in Figure 4.4, filter includes two sub-blocks given in Figure 4.5 and Figure 4.6.

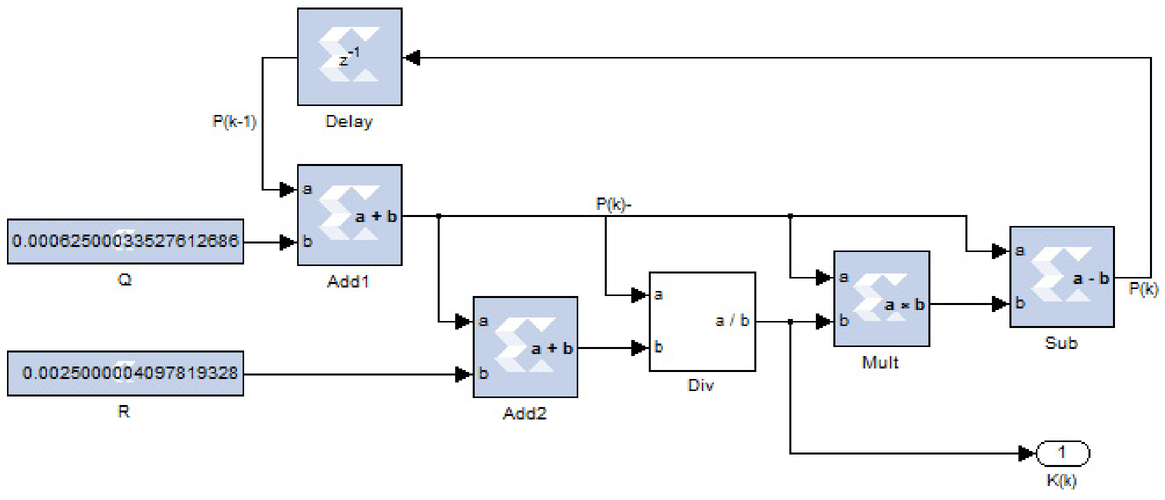


Figure 4.5: Kalman Gain Generator Block

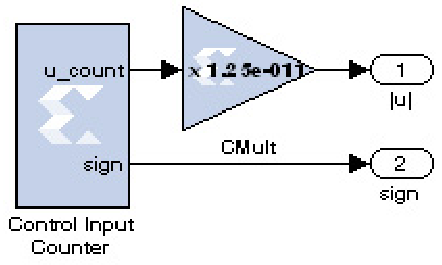


Figure 4.6: Control Input Block

Even though the input of the filter is unsigned 27-bits with the binary point at bit 27, operations with in the filter takes place at 32-bits and the output of filter is in signed 32-bits with the binary point at 30. But Division sub-block of Kalman Gain block is an exception which inputs two 32-bits and outputs 64-bits for precision. This block has a latency of 68 cycles while 2 Scale blocks are used in order to put the data in a form that is available for Divider Generator 3.0 and there is an Add block for addition of quotient and fractional outputs.

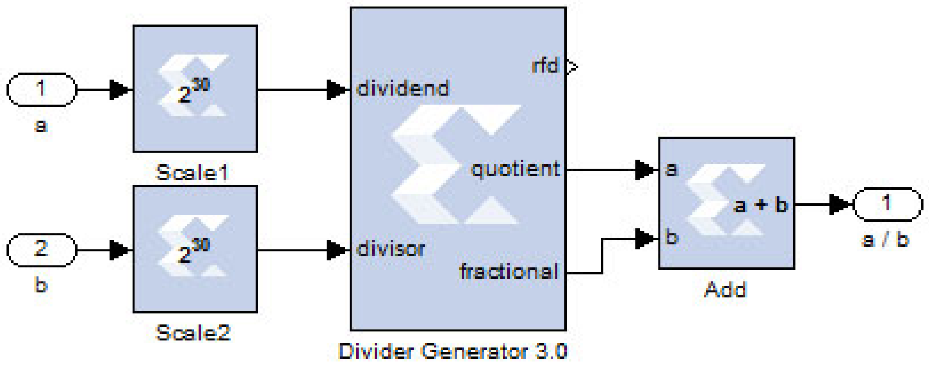


Figure 4.7: Divider Block

CHAPTER 5

SIMULATIONS

Simulations are done by the use of Mathworks Simulink and Xilinx ISE Simulator softwares with Xilinx ML605 Evaluation Kit and Spartan-3E Starter Kit. Since the system includes high frequency hardware components in addition to an FPGA, these parts are simulated in Simulink. On the other hand, the controller design will operate on the FPGA while feedbacks will come from Simulink instead of a real VCO. This hardware-software integrated simulation with FPGA is called FPGA-In-the-Loop (FIL) simulation. While it may be expected to complete simulations in real time, it is not correct since the normal loop operation with a rate of 50 MHz is not possible during FIL simulation. For every cycle FPGA output will be captured from the JTAG chain over USB and after simulating microwave frequencies in Simulink, the software simulation outputs will be sent to FPGA in the same way. For design verification and analysis more than a modulation cycle is being captured and spectrogram of data is investigated. Since the sampling rate used for data collection from VCO output is $1e^{10}$, the number of data that is being collected for 17 ms is $170e^{6}$. More over collecting this data takes a long time, computing the 2^{14} point FFT of the collected data requires a high end computer.

Designed controller architecture of the system is suitable for a Virtex-6 FPGA on the ML605 Evaluation Board but it not possible to operate this Kalman Filter algorithm on a Spartan-3E FPGA at 50 MHz as the one that Spartan-3E Started Kit includes. For this reason, by using the advantages of the platform that is being worked on, Kalman Filter design block is implemented on Simulink for Spartan-3E starter kit and by the rest of controller FIL simulations are completed. The block diagrams of Simulink model of Kalman Filter is given in Figure 5.1, Figure 5.2, Figure 5.3 and Figure 5.4 which are identical with FPGA implementation models except the Divider model. Since Divider model implemented for FPGA includes the Divider Generator 3.0 IP which has 68 cycles of latency, a Delay block is added to Simulink model.

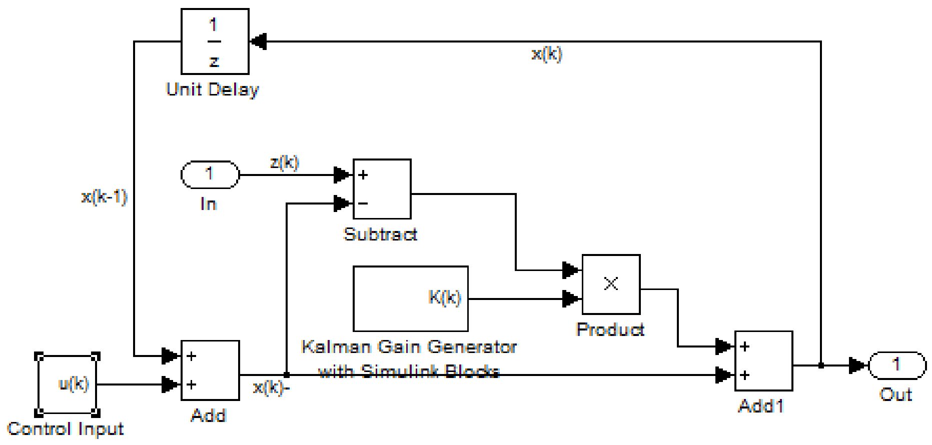


Figure 5.1: Simulink Kalman Filter Block

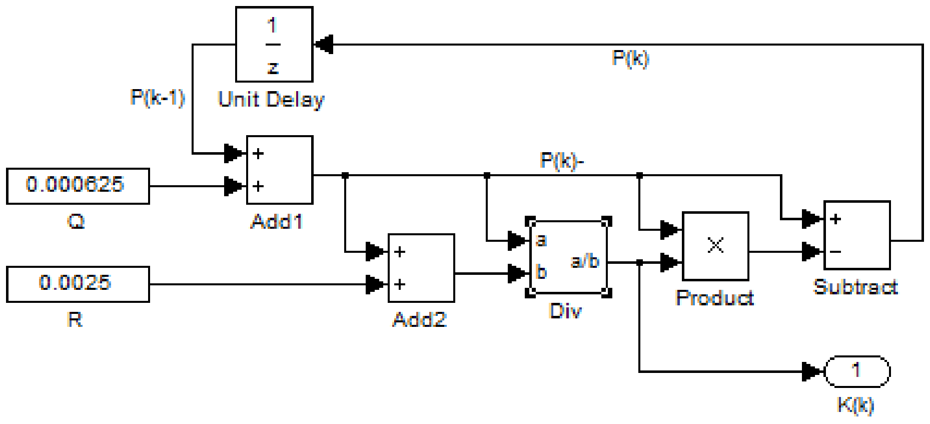


Figure 5.2: Simulink Kalman Gain Generator Block

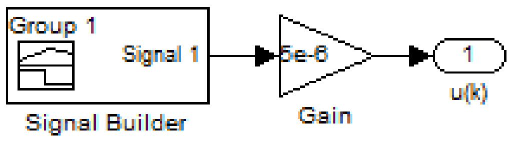


Figure 5.3: Simulink Control Input Block

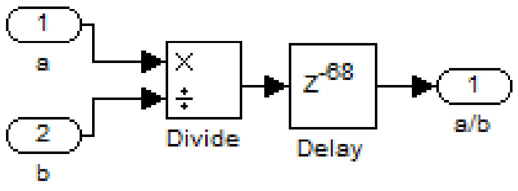


Figure 5.4: Simulink Divider Block

5.1 VCO and Frequency Divider Simulink Models

The VCO that is modelled for simulations is highly non-linear inside the operational bandwidth. Also phase noise is modelled with Gaussian noise as it is shown in Figure 5.5

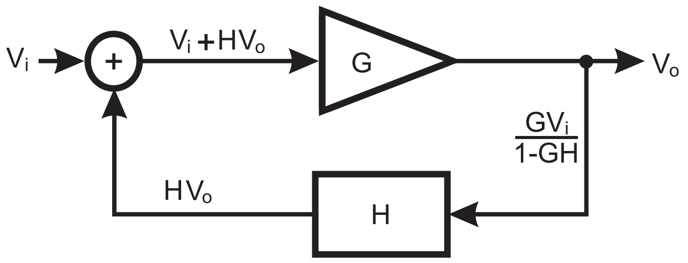


Figure 5.5: Simulink VCO Model

The Frequency Divider model is a simple counter based divider but zero-cross detection is included in order to avoid false frequency division.

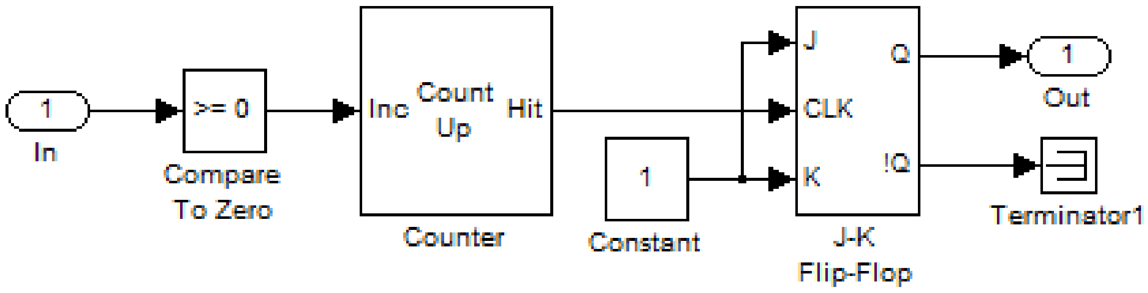


Figure 5.6: Simulink Frequency Divider Model

5.2 Frequency Sweep Analysis of VCO

Figure 5.7 shows the block diagram for linear input sweep of a linear VCO model while the control input of simulation is as given in Figure 5.8 and the output spectrogram of VCO is shown in Figure 5.9.

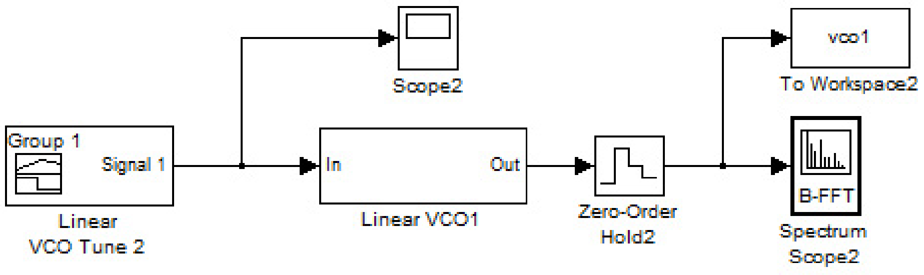


Figure 5.7: Linear Input Sweep of Linear VCO

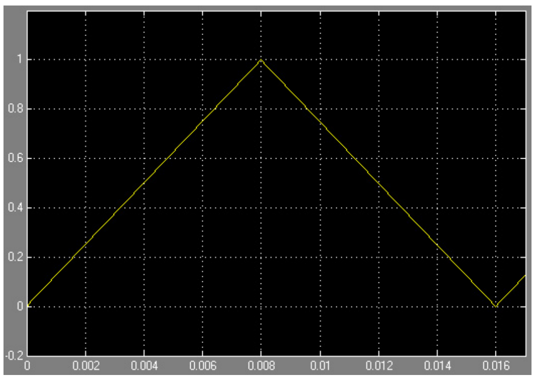


Figure 5.8: Control Input of Simulation shown in Figure 5.7

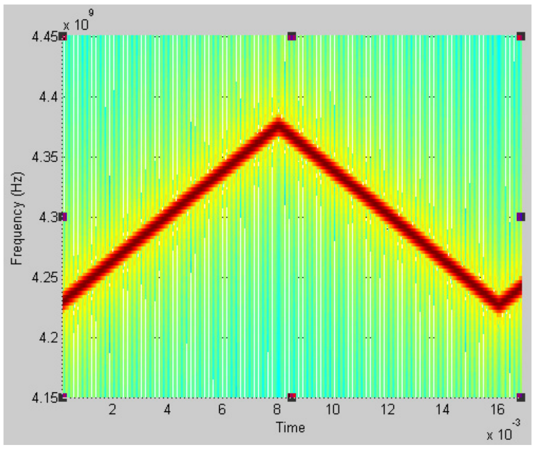


Figure 5.9: Output Spectrogram of VCO for Simulation shown in Figure 5.7

Figure 5.10 shows the block diagram for linear input sweep of a nonlinear VCO model while the control input of simulation is as given in Figure 5.11 and the output spectrogram of VCO is shown in Figure 5.12.

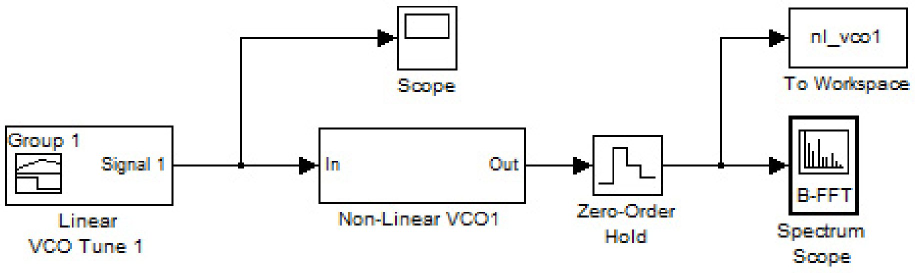


Figure 5.10: Linear Input Sweep of Nonlinear VCO

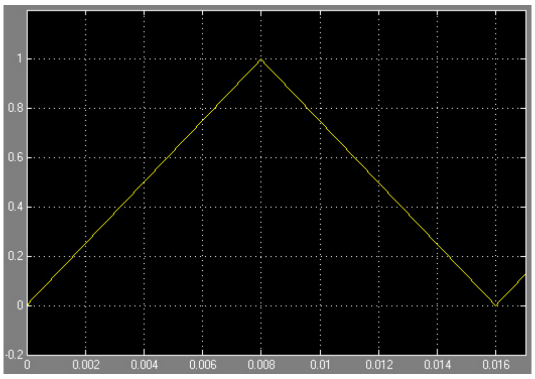


Figure 5.11: Control Input of Simulation shown in Figure 5.10

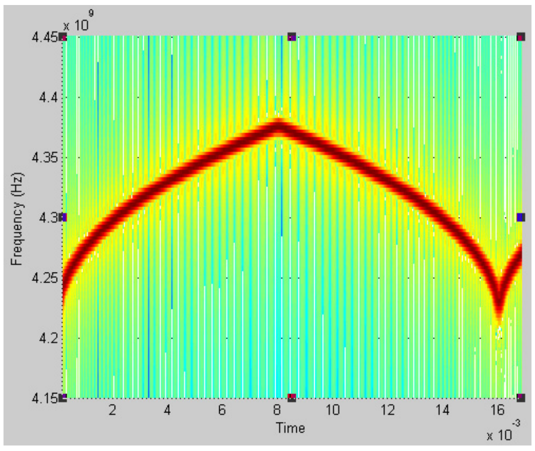


Figure 5.12: Output Spectrogram of VCO for Simulation shown in Figure 5.10

Figure 5.13 shows the block diagram for nonlinear input sweep of a linear VCO model while the control input of simulation is as given in Figure 5.14 and the output spectrogram of VCO is shown in Figure 5.15.

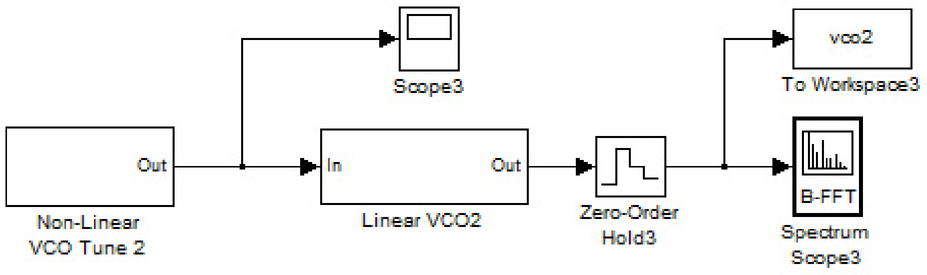


Figure 5.13: Nonlinear Input Sweep of Linear VCO

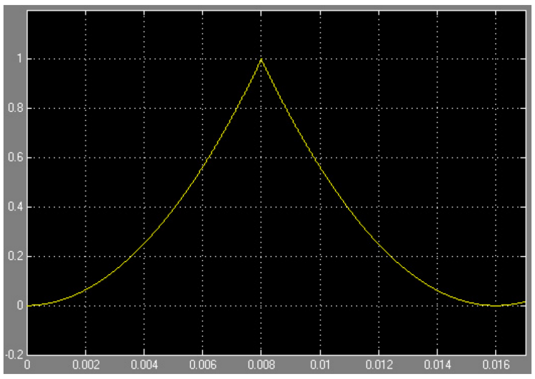


Figure 5.14: Control Input of Simulation shown in Figure 5.13

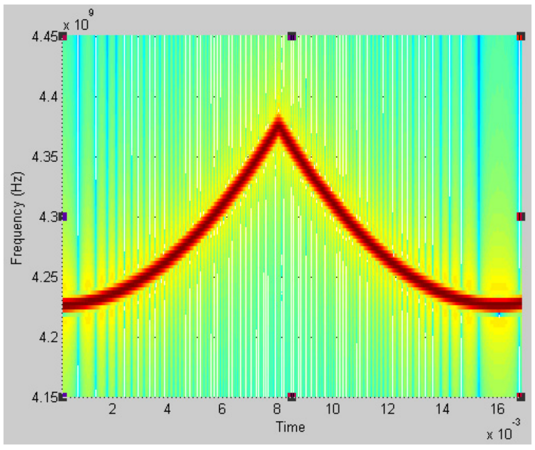


Figure 5.15: Output Spectrogram of VCO for Simulation shown in Figure 5.13

Figure 5.16 shows the block diagram for nonlinear input sweep of a nonlinear VCO model while the control input of simulation is as given in Figure 5.17 and the output spectrogram of VCO is shown in Figure 5.18.

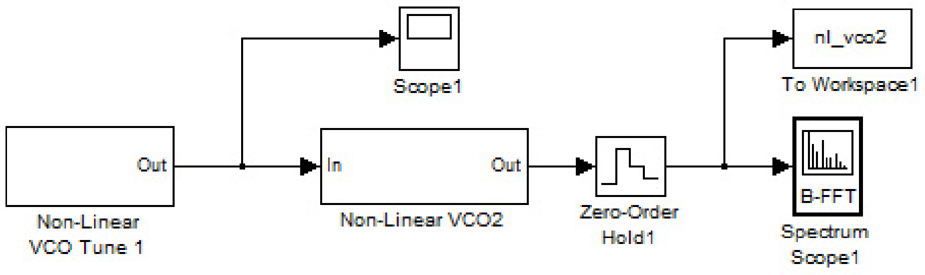


Figure 5.16: Nonlinear Input Sweep of Nonlinear VCO

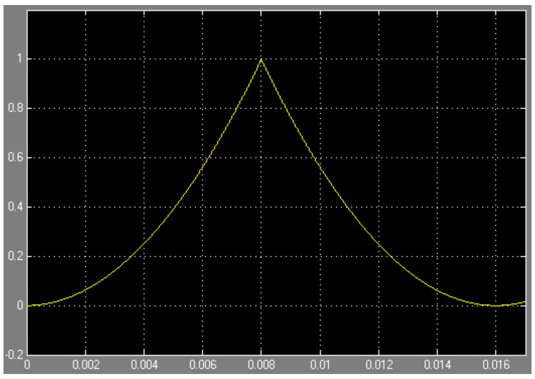


Figure 5.17: Control Input of Simulation shown in Figure 5.16

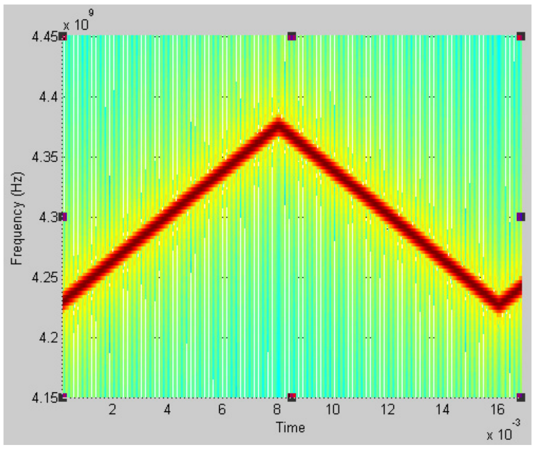


Figure 5.18: Output Spectrogram of VCO for Simulation shown in Figure 5.16

5.3 Frequency Sweep Linearization

Figure 5.19, Figure 5.20, Figure 5.21 and Figure 5.22 show the system block diagrams for the linear sweep controller that is being designed as the output of this thesis.

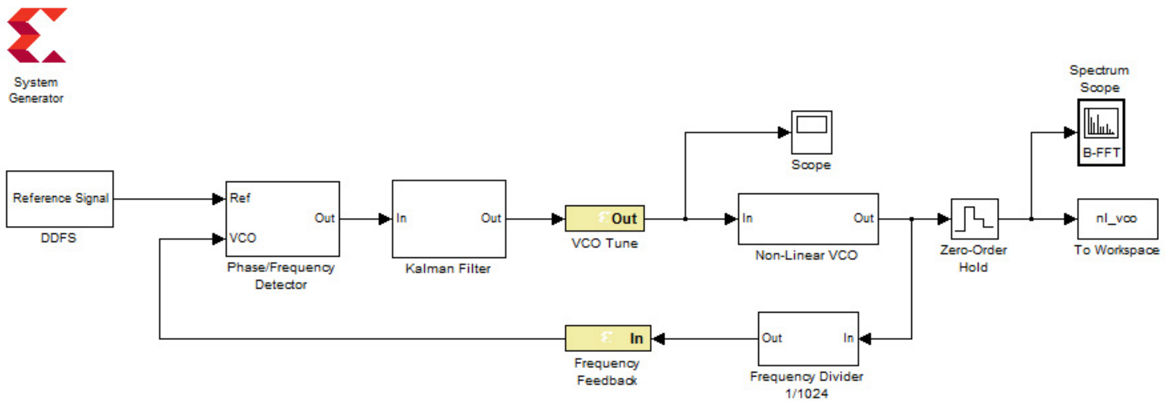


Figure 5.19: System Block Diagram of Complete Controller Implementation

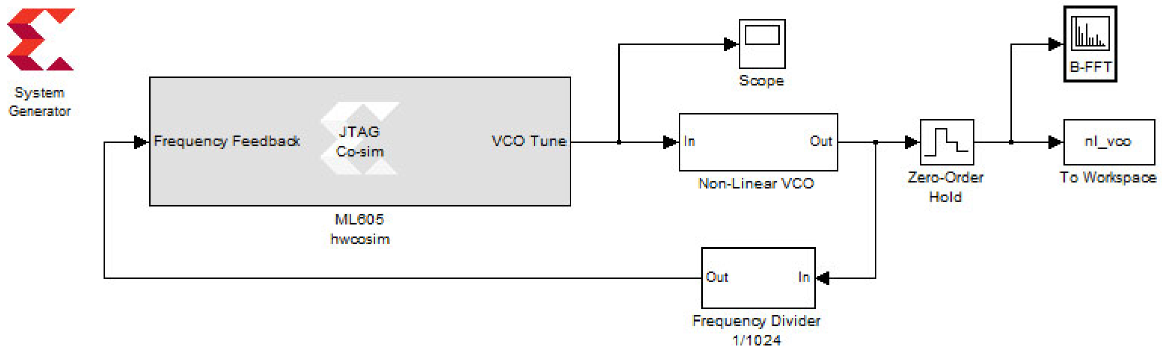


Figure 5.20: FPGA-In-the-Loop Simulation for ML605

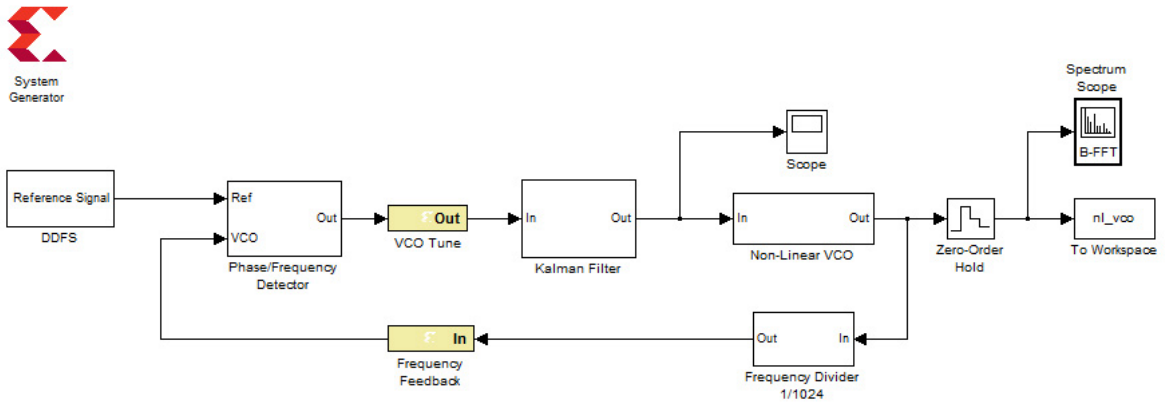


Figure 5.21: System Block Diagram of Partial Controller Implementation

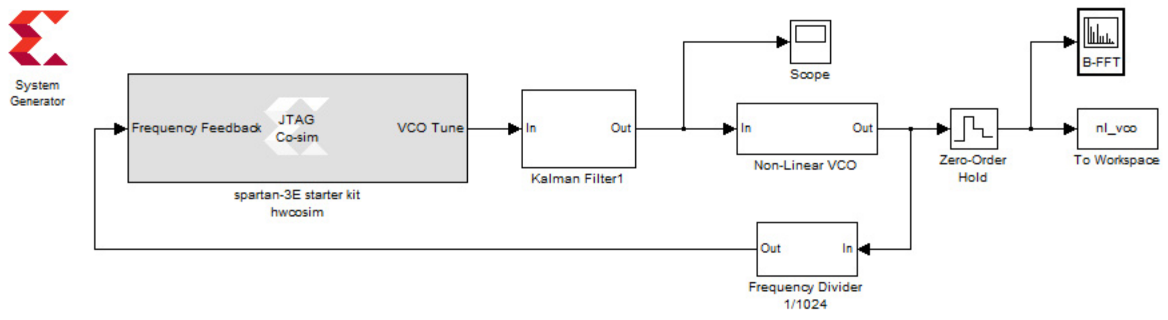


Figure 5.22: FPGA-In-the-Loop Simulation for Spartan-3E Starter Kit

Figure 5.23 shows the control input generated by the designed controller, for the nonlinear VCO model that is analysed in Section 5.2.

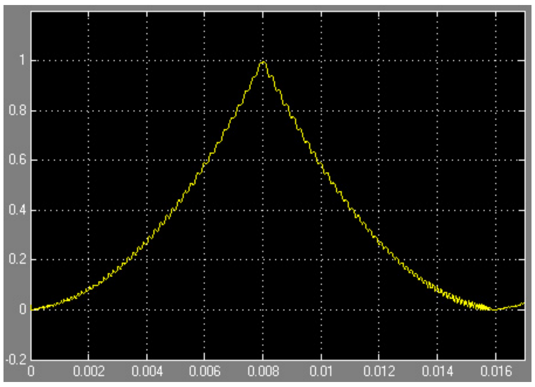


Figure 5.23: Generated Control Input by Controller

Figure 5.24 shows the linearized output spectrogram of nonlinear VCO model that is analysed in Section 5.2 by the use of designed controller.

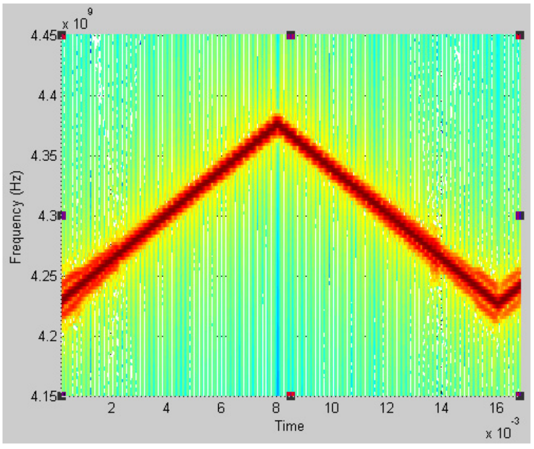


Figure 5.24: Linearized Spectrogram

5.4 Parameter Tuning

During the design verification several simulations conducted for parameter tuning. Figure 5.25 shows the control input generated by the controller as well as the phase detector output signals for the linear VCO model at the early stage of tuning when the Kalman Filter is not integrated yet.

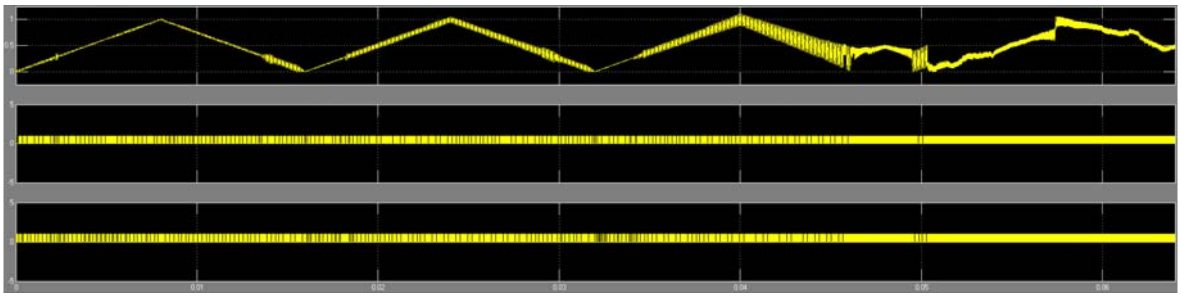


Figure 5.25: Linear VCO Control Input and Phase Detector Outputs Without Kalman Filter, Not Tuned Charge-Pump

Figure 5.26 shows same signals, for the linear VCO model after Charge-Pump tuning by increasing channel width in order to increase the low pass filter effect of this parameter.

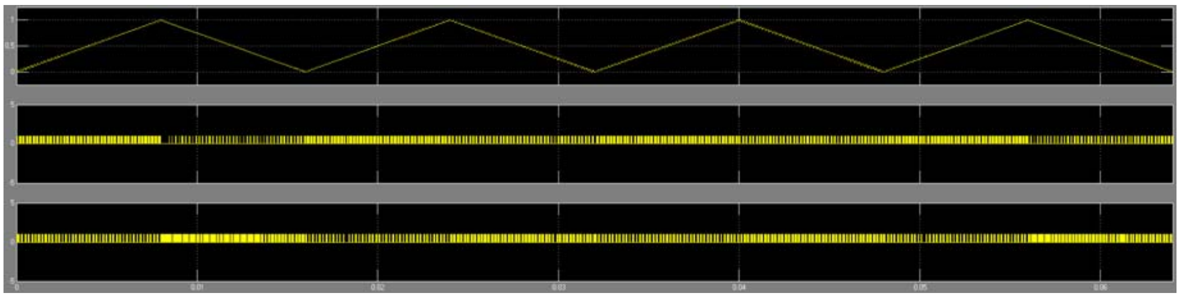


Figure 5.26: Linear VCO Control Input and Phase Detector Outputs Without Kalman Filter, Tuned Charge-Pump (Channel Width = W)

Figure 5.27 shows output signals for the nonlinear VCO model with the same parameters of the controller. As shown in Figure 5.28, increasing channel width is not adequate for more filtration since decreasing the fixed bandwith of the system makes it impossible to track the fast sweep of reference signal. At this point Kalman Filter is integrated to the system, which results control input signal as shown in Figure 5.29 while the system model was as given below in the same figure.

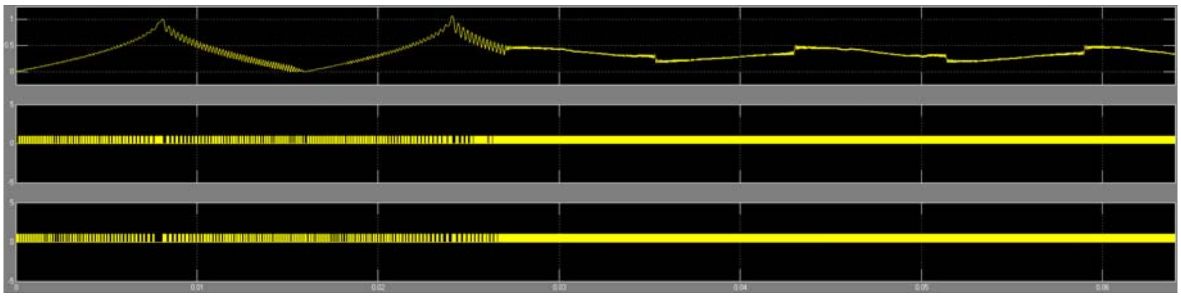


Figure 5.27: Nonlinear VCO Control Input and Phase Detector Outputs Without Kalman Filter, Charge-Pump Channel Width = W

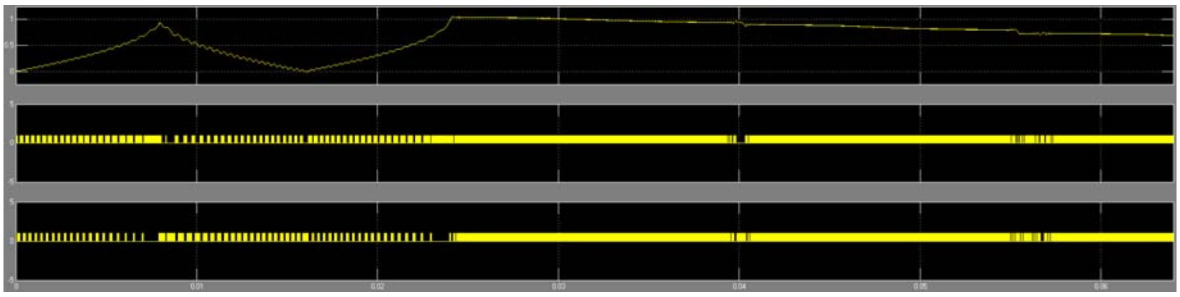


Figure 5.28: Nonlinear VCO Control Input and Phase Detector Outputs Without Kalman Filter, Charge-Pump Channel Width > W

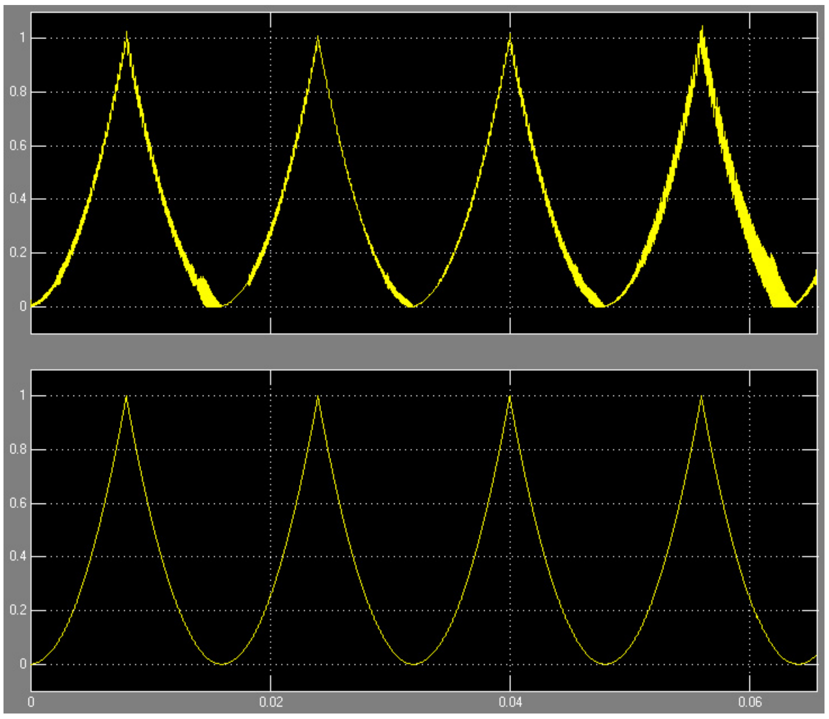


Figure 5.29: Nonlinear VCO Control Input and System Model With Kalman Filter (Q, R), Charge-Pump Channel Width = W

Figure 5.30 shows the VCO spectrogram of the system by these parameters. Due to the high phase noise seen in the spectrogram, Kalman parameters and Charge-Pump are tuned which resulted as given in Figure 5.31 and Figure 5.32.

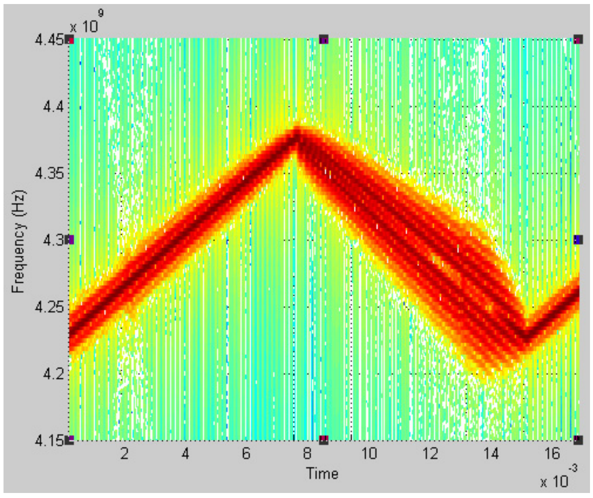


Figure 5.30: Nonlinear VCO Output Spectrogram With Kalman Filter (Q', R'), Charge-Pump Channel Width = W'

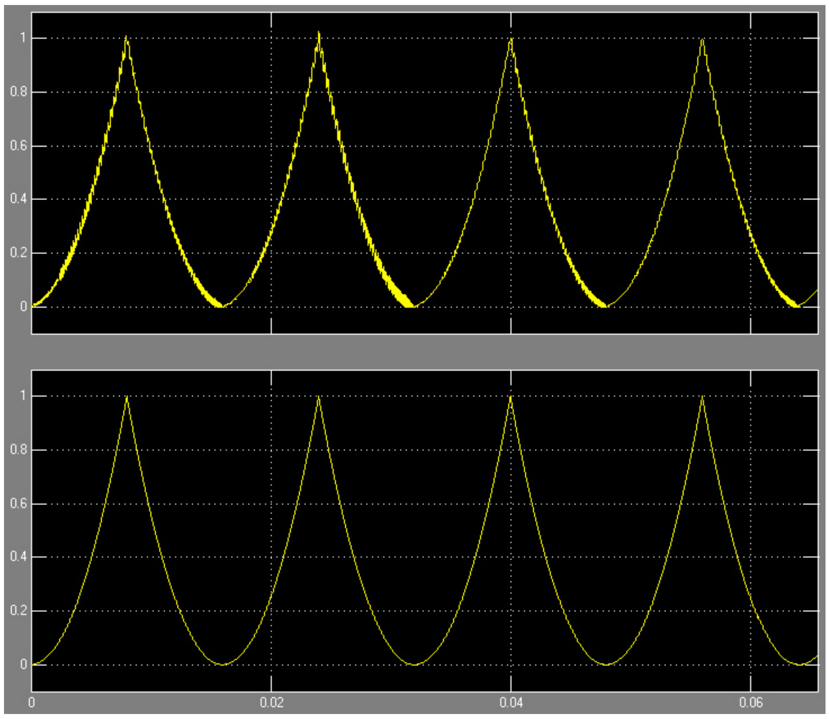


Figure 5.31: Nonlinear VCO Control Input and System Model With Tuned Kalman Filter (Q', R'), Tuned Charge-Pump (Channel Width = W')

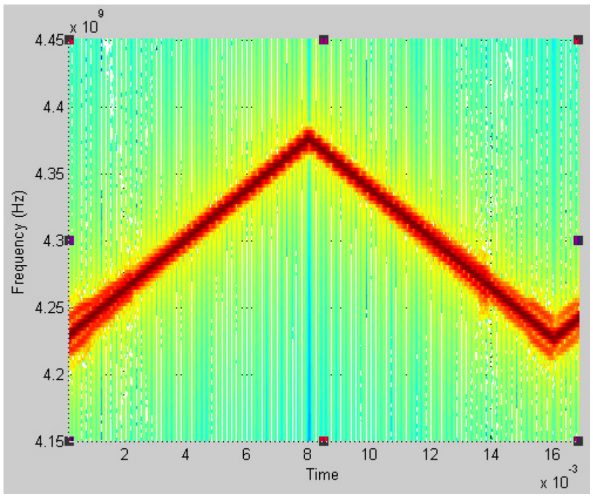


Figure 5.32: Nonlinear VCO Output Spectrogram With Tuned Kalman Filter (Q', R'), Tuned Charge-Pump (Channel Width = W')

Figure 5.33 shows the comparison of the frequency sweep of nonlinear VCO with and without the controller, from left to right respectively.

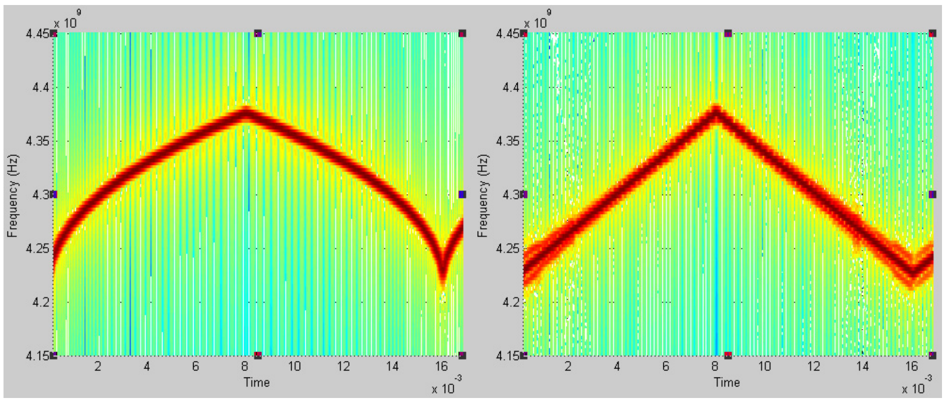


Figure 5.33: Spectrogram Comparison

CHAPTER 6

RESULTS AND CONCLUSIONS

Simulations presented in Chapter 5 prove the method for the linearization of frequency sweep that is proposed in this thesis. Both software based simulations and FPGA-In-the-Loop Hardware co-simulations give satisfactory results for linearization of a VCO model even with high phase noise. Simulation results also show that this method has a better phase noise performance due to the agile behaviour of Loop Filter by the use of Kalman Filtering.

It is shown that, Charge-Pump implementation introduces low pass filter effect. But as it is indicated in Section 3.3, this fixed bandwidth filtering in not adequate for most cases. Loop Filter implementation of this thesis proved that, Kalman Filter is appropriate for PLL applications, regardless the bandwidth requirements.

This thesis shows that, the described method of embedded PLL implementation is applicable for practical usage. Benefiting the advantages of model based design, several modularly implemented sub-blocks of the system that are designed with some new ideas, are also applicable for different applications.

CHAPTER 7

DISCUSSION AND FUTURE WORKS

Since PLL structure is a well known and a conventional method for frequency sweep linearization, it is not obligatory to make a discussion on the main structure of the system, but it is possible to discuss on improved designed blocks which are original to this work.

Due to the digital implementation of the entire system, reference signal generation by a DDS may bring some defects. In this thesis, it is proposed to eliminate these deficiencies by the use of Kalman Filter. Since the implemented filter requires a well described control input related with the system model, the performance of a system designed in this manner depends on the reliability of the model. It is also indicated that the phase detection algorithm is an improved form of conventional phase-frequency detector behaviour. Since this improvement arise in lower frequency reference signal applications, for this work, the contribution of this design is limited. But relying on the experimental results, this thesis suggests this implementation for the cases with higher frequency divider ratio for a better performance.

This thesis was concentrated on developing an embedded implementation of a method. As a future work, digital implementation of a conventional passive Loop Filter, that is deeply analysed in Appendix C, will be compared with the Kalman Filter. Also the Kalman Filter algorithm that is implemented in this work is to be improved in the future works by including time varying parameters of the model. As a side output, this thesis will serve as a base for further research on phase-frequency detection algorithms.

REFERENCES

[1] E. Hegazi and A.A. Abidi. Varactor characteristics, oscillator tuning curves, and am-fm conversion. *Solid-State Circuits, IEEE Journal of*, 38(6):1033 – 1039, june 2003.

[2] A. Buonomo. Nonlinear analysis of voltage-controlled oscillators: A systematic approach. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 55(6):1659 –1670, july 2008.

[3] J. Fuchs, K.D. Ward, M.P. Tulin, and R.A. York. Simple techniques to correct for vco nonlinearities in short range fmcw radars. In *Microwave Symposium Digest, 1996., IEEE MTT-S International*, volume 2, pages 1175 –1178 vol.2, jun 1996.

[4] M. Pichler, A. Stelzer, P. Gulden, C. Seisenberger, and M. Vossiek. Frequency-sweep linearization for fmcw sensors with high measurement rate. In *Microwave Symposium Digest, 2005 IEEE MTT-S International*, page 4 pp., june 2005.

[5] A. Stelzer, S. Scheiblhofer, S. Schuster, and M. Brandl. Multi reader/multi-tag saw rfid systems combining tagging, sensing, and ranging for industrial applications. In *Frequency Control Symposium, 2008 IEEE International*, pages 263 –272, may 2008.

[6] M. Pichler, A. Stelzer, P. Gulden, C. Seisenberger, and M. Vossiek. Phase-error measurement and compensation in pll frequency synthesizers for fmcw sensors mdash;i: Context and application. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 54(5):1006 –1017, may 2007.

[7] M.A. Borucek. Design and implementation of low phase noise phase lock loop based local oscillator. Master's thesis, Middle East Technical University, 2009.

[8] F. Jonsson. Design and calibration of integrated phase lock loop frequency synthesizers. Master's thesis, KTH - Royal Institute of Technology, 2008.

[9] L. Ping. Improving tracking performans of phase lock loop in high dynamic applications. Master's thesis, The University of Calgary, 2004.

[10] Zahir M.H. Saleh, R.A. and A.A. Mahmoud. *Digital Phase Lock Loops - Architectures and Applications*. Springer, 2006.

[11] T. Mitomo, N. Ono, H. Hoshino, Y. Yoshihara, O. Watanabe, and I. Seto. A 77 ghz 90 nm cmos transceiver for fmcw radar applications. *Solid-State Circuits, IEEE Journal of*, 45(4):928 –937, april 2010.

[12] G. Welch and G. Bishop. *An Introduction to Kalman Filtering*, chapter Course 8. SIGGRAPH 2001. University of North Carolina at Chapel Hill, Los Angeles, CA, August 12-17 2001.

[13] D. Simon. *Optimal State Estimation: Kalman, H-Infinity, and Non Linear Approaches*. John Wiley and Sons, 2006.

[14] H. Ruser and V. Magori. Sweep linearization of a microwave fmcw doppler sensor by an ultrasonic reference. In *Frequency Control Symposium, 1997., Proceedings of the 1997 IEEE International*, pages 201 –206, may 1997.

[15] T. Musch, I. Rolfes, and B. Schiek. Fractional divider concepts with phase-lock-control for the generation of precise linear frequency ramps. In *Microwave Conference, 1998. 28th European*, volume 1, pages 451 –456, oct. 1998.

[16] T. Musch and B. Schiek. A fractional ramp generator with improved linearity and phase-noise performance for the use in heterodyne measurement systems. *Advances in Radio Science*, 3:75 –81, 2005.

[17] T. Musch, B. Schulte, and B. Schiek. A fast heterodyne network analyzer based on precision linear frequency ramps. In *Microwave Conference, 2001. 31st European*, pages 1 –4, sept. 2001.

[18] A. Stelzer, E. Kolmhofer, and S. Scheiblhofer. Fast 77 ghz chirps with direct digital synthesis and phase locked loop. In *Microwave Conference Proceedings, 2005. APMC 2005. Asia-Pacific Conference Proceedings*, volume 3, page 4 pp., dec. 2005.

[19] Yun-Taek Im, Jee-Hoon Lee, and Seong-Ook Park. A dds and pll-based x-band fmcw radar system. In *Intelligent Radio for Future Personal Terminals (IMWS-IRFPT), 2011 IEEE MTT-S International Microwave Workshop Series on*, pages 1 –2, aug. 2011.

[20] P.V. Brennan, Y. Huang, M. Ash, and K. Chetty. Determination of sweep linearity requirements in fmcw radar systems based on simple voltage-controlled oscillator sources. *Aerospace and Electronic Systems, IEEE Transactions on*, 47(3):1594 –1604, july 2011.

[21] C.J. Kikkert. Two novel phase-frequency detectors. In *Circuits and Systems, 2006. APCCAS 2006. IEEE Asia Pacific Conference on*, pages 712 –715, dec. 2006.

[22] M. Kozak and E.G. Friedman. Design and simulation of fractional-n pll frequency synthesizers. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 4, pages IV – 780–3 Vol.4, may 2004.

[23] P.J. Burke. Ultra-linear chirp generation via vco tuning predistortion. In *Microwave Symposium Digest, 1994., IEEE MTT-S International*, pages 957 –960 vol.2, may 1994.

[24] Guangming Yu, Yu Wang, Huazhong Yang, and Hui Wang. A fast-locking all-digital phase-locked loop with a novel counter-based mode switching controller. In *TENCON 2009 - 2009 IEEE Region 10 Conference*, pages 1 –5, jan. 2009.

[25] K. De Brabandere, T. Loix, K. Engelen, B. Bolsens, J. Van den Keybus, J. Driesen, and R. Belmans. Design and operation of a phase-locked loop with kalman estimator-based filter for single-phase applications. In *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, pages 525 –530, nov. 2006.

[26] J.I. Statman and W.J. Hurd. An estimator-predictor approach to pll loop filter design. *Communications, IEEE Transactions on*, 38(10):1667 –1669, oct 1990.

[27] G.A. Hirchoren and D.S. Arantes. Optimal phase-locked loop design with kalman predictors for synchronous networks. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 3, pages 1917 –1920 vol.3, apr 1997.

[28] M.H. Izadi and B. Leung. Pll-based frequency discriminator using the loop filter as an estimator. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 49(11):721 – 727, nov 2002.

[29] Wei-Tsen Lin and Dah-Chung Chang. The extended kalman filtering algorithm for carrier synchronization and the implementation. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, page 4 pp., may 2006.

[30] Weibin Li, Shanjian Liu, Chunhui Zhou, Shidong Zhou, and Tingchang Wang. High dynamic carrier tracking using kalman filter aided phase-lock loop. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 673 –676, sept. 2007.

[31] A. Gothandaraman and S.K. Islam. An all-digital frequency locked loop (adfll) with a pulse output direct digital frequency synthesizer (ddfs) and an adaptive phase estimator. In *Radio Frequency Integrated Circuits (RFIC) Symposium, 2003 IEEE*, pages 303 – 306, june 2003.

[32] M.Z. Straayer, A.V. Messier, and W.G. Lyons. Ultra-linear superwideband chirp generator using digital compensation. In *Microwave Symposium Digest, 2006. IEEE MTT-S International*, pages 403 –406, june 2006.

[33] A. Meta, P. Hoogeboom, and L.P. Ligthart. Range non-linearities correction in fmcw sar. In *Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. IEEE International Conference on*, pages 403 –406, 31 2006-aug. 4 2006.

[34] S.O. Piper. Fmcw linearizer bandwidth requirements. In *Radar Conference, 1991., Proceedings of the 1991 IEEE National*, pages 142 –146, mar 1991.

[35] S. Scheiblhofer, S. Schuster, and A. Stelzer. Signal model and linearization for nonlinear chirps in fmcw radar saw-id tag request. *Microwave Theory and Techniques, IEEE Transactions on*, 54(4):1477 – 1483, june 2006.

[36] S. Scheiblhofer, S. Schuster, and A. Stelzer. High-speed fmcw radar frequency synthesizer with dds based linearization. *Microwave and Wireless Components Letters, IEEE*, 17(5):397 –399, may 2007.

[37] Ting Wu, P.K. Hanumolu, K. Mayaram, and Un-Ku Moon. Method for a constant loop bandwidth in lc-vco pll frequency synthesizers. *Solid-State Circuits, IEEE Journal of*, 44(2):427 –435, feb. 2009.

[38] Hong Zhang, Guican Chen, and Ning Li. A 2.4-ghz linear-tuning cmos lc voltage-controlled oscillator. In *Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005. Asia and South Pacific*, volume 2, pages 799 – 802 Vol. 2, jan. 2005.

[39] T. Musch. A high precision 24-ghz fmcw radar based on a fractional-n ramp-pll. *Instrumentation and Measurement, IEEE Transactions on*, 52(2):324 – 327, april 2003.

[40] T. Musch, I. Rolfes, and B. Schiek. A highly linear frequency ramp generator based on a fractional divider phase-locked-loop. *Instrumentation and Measurement, IEEE Transactions on*, 48(2):634 –637, april 1999.

[41] M. Christmann, M. Vossiek, M. Smith, and G. Rodet. Saw based delay locked loop concept for vco linearization in fmcw radar sensors. In *Microwave Conference, 2003. 33rd European*, volume 3, pages 1135 – 1138 Vol.3, oct. 2003.

[42] M. Pichler, A. Stelzer, P. Gulden, and M. Vossiek. Influence of systematic frequency-sweep non-linearity on object distance estimation in fmcw/fscw radar systems. In *Microwave Conference, 2003. 33rd European*, volume 3, pages 1203 – 1206 Vol.3, oct. 2003.

APPENDIX A

VHDL Codes

A.1 VHDL Code for Reference Counter

```
----------------------------------------------------------------------------------
-- Company: YILDIRIM Elektronik R&D Department
-- Engineer: Burak Dursun
--
-- Create Date:    20:59:09 02/06/2012
-- Design Name:    Linear Sweep Controller
-- Module Name:    ref_counter - Reference Counter
-- Project Name:   MSc Thesis Study - VCO Sweep Linearization for FMCW Radar
-- Target Devices: Spartan-3E starter kit & Spartan-6 co-processing kit
-- Tool versions:  ISE Design Suite 13.3 System Edition & MATLAB R2011b
-- Description:    Preliminary Work for Surveillance Radar Project
--
-- Dependencies:   RF-DSP\Research\Implementation\desktop\x61
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments: Revision after desktop\x60 and laptop\xCA
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

use ieee.std_logic_unsigned.all;

entity ref_counter is
    Port ( clk : in  STD_LOGIC;
           ce : in  STD_LOGIC := '1';
           ref_count : out  STD_LOGIC_VECTOR (26 downto 0));
end ref_counter;

architecture Behavioral of ref_counter is

        type state is (up,down);
        signal direction: state;
        signal count: std_logic_vector (26 downto 0) := "000101010001111001011001000";

begin

process (clk)
begin
        if (clk'event and clk='1') then
                case direction is
                        when up =>
                                if (count="000101011110000110101001000") then
                                        count <= count - 1;
                                        direction <= down;
                                else
                                        count <= count + 1;
                                end if;
                        when down =>
                                if (count="000101010001111001011001000") then
                                        count <= count + 1;
                                        direction <= up;
                                else
                                        count <= count - 1;
                                end if;
                        when others =>
                                count <= "000101010001111001011001000";
                                direction <= up;
                end case;
        end if;
end process;

ref_count <= count;

end Behavioral;
```

A.2 VHDL Code for Phase Frequency Detector

```
----------------------------------------------------------------------------------
-- Company: YILDIRIM Elektronik R&D Department
-- Engineer: Burak Dursun
--
-- Create Date:    19:44:27 02/06/2012
-- Design Name:    Linear Sweep Controller
-- Module Name:    pdf - Phase Frequency Detector
-- Project Name:   MSc Thesis Study - VCO Sweep Linearization for FMCW Radar
-- Target Devices: Spartan-3E starter kit & Spartan-6 co-processing kit
-- Tool versions:  ISE Design Suite 13.3 System Edition & MATLAB R2011b
-- Description:    Preliminary Work for Surveillance Radar Project
--
-- Dependencies:   RF-DSP\Research\Implementation\desktop\x61
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments: Revision after desktop\x60 and laptop\xCA
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

use ieee.std_logic_unsigned.all;

entity pdf is
    Port ( clk : in  STD_LOGIC;
           ce : in  STD_LOGIC := '1';
           ref : in  STD_LOGIC;
           vco : in  STD_LOGIC;
           up : out  STD_LOGIC;
           down : out  STD_LOGIC);
end pdf;

architecture Behavioral of pdf is

        type state is (idle,add,sub);
        signal direction: state := idle;
        signal ref_ps: std_logic := '0';
        signal vco_ps: std_logic := '0';

begin

process (clk)
begin
        if (clk'event and clk='1') then
                if (ref/=ref_ps and vco/=vco_ps) then
                        if (ref='1' and vco='1') then
                                direction <= idle;
                        elsif (ref='1' and vco='0') then
                                case direction is
                                        when idle =>
                                                direction <= add;
                                        when add =>
                                                direction <= add;
                                        when sub =>
                                                direction <= idle;
                                        when others =>
                                                direction <= idle;
                                end case;
                        elsif (ref='0' and vco='1') then
                                case direction is
                                        when idle =>
                                                direction <= sub;
                                        when add =>
                                                direction <= idle;
                                        when sub =>
                                                direction <= sub;
                                        when others =>
                                                direction <= idle;
                                end case;
                        end if;
                        ref_ps <= ref;
                        vco_ps <= vco;
                elsif (ref/=ref_ps) then
                        if (ref='1') then
                                if (vco='0') then
                                        direction <= add;
                                else
                                        case direction is
                                                when idle =>
                                                        direction <= add;
                                                when add =>
                                                        direction <= add;
                                                when sub =>
                                                        direction <= idle;
                                                when others =>
                                                        direction <= idle;
                                        end case;
                                end if;
                        end if;
                        ref_ps <= ref;
                elsif (vco/=vco_ps) then
                        if (vco='1') then
                                if (ref='0') then
                                        direction <= sub;
                                else
                                        case direction is
                                                when idle =>
                                                        direction <= sub;
                                                when add =>
                                                        direction <= idle;
                                                when sub =>
                                                        direction <= sub;
                                                when others =>
                                                        direction <= idle;
                                        end case;
                                end if;
                        end if;
                        vco_ps <= vco;
                end if;
        end if;
end process;

process (direction)
begin
        case direction is
```

```
                when add =>
                        up <= '1';
                        down <= '0';
                when sub =>
                        up <= '0';
                        down <= '1';
                when idle =>
                        up <= '0';
                        down <= '0';
                when others =>
                        up <= '0';
                        down <= '0';
        end case;
end process;

end Behavioral;
```

A.3 VHDL Code for Phase Counter

```
----------------------------------------------------------------------------------
-- Company: YILDIRIM Elektronik R&D Department
-- Engineer: Burak Dursun
--
-- Create Date:    20:32:32 02/06/2012
-- Design Name:    Linear Sweep Controller
-- Module Name:    p_counter - Phase Counter
-- Project Name:   MSc Thesis Study - VCO Sweep Linearization for FMCW Radar
-- Target Devices: Spartan-3E starter kit & Spartan-6 co-processing kit
-- Tool versions:  ISE Design Suite 13.3 System Edition & MATLAB R2011b
-- Description:    Preliminary Work for Surveillance Radar Project
--
-- Dependencies:   RF-DSP\Research\Implementation\desktop\x61
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments: Revision after desktop\x60 and laptop\xCA
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

use ieee.std_logic_unsigned.all;

entity p_counter is
    Port ( clk : in  STD_LOGIC;
           ce : in  STD_LOGIC := '1';
           up : in  STD_LOGIC;
           down : in  STD_LOGIC;
           phase : out  STD_LOGIC_VECTOR (15 downto 0));
end p_counter;

architecture Behavioral of p_counter is

        signal counter: std_logic_vector (15 downto 0) := (others => '0');

begin

process (clk)
begin
        if (clk'event and clk='1') then
                if (up='1') then
                        if (counter="1111111111111111") then
                                counter <= counter;
                        else
                                counter <= counter + 1;
                        end if;
                elsif (down='1') then
                        if (counter="0000000000000000") then
                                counter <= counter;
                        else
                                counter <= counter - 1;
                        end if;
                else
                        counter <= counter;
                end if;
        end if;
end process;

phase <= counter;

end Behavioral;
```

A.4 VHDL Code for Control Input Counter

```
----------------------------------------------------------------------------------
-- Company: YILDIRIM Elektronik R&D Department
-- Engineer: Burak Dursun
--
-- Create Date:    21:31:05 02/06/2012
-- Design Name:    Linear Sweep Controller
-- Module Name:    u_counter - Control Input Counter
-- Project Name:   MSc Thesis Study - VCO Sweep Linearization for FMCW Radar
-- Target Devices: Spartan-3E starter kit & Spartan-6 co-processing kit
-- Tool versions:  ISE Design Suite 13.3 System Edition & MATLAB R2011b
-- Description:    Preliminary Work for Surveillance Radar Project
--
-- Dependencies:   RF-DSP\Research\Implementation\desktop\x61
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments: Revision after desktop\x60 and laptop\xCA
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

use ieee.std_logic_unsigned.all;

entity u_counter is
    Port ( clk : in  STD_LOGIC;
           ce : in  STD_LOGIC := '1';
           u_count : out  STD_LOGIC_VECTOR (18 downto 0);
           sign : out  STD_LOGIC);
end u_counter;

architecture Behavioral of u_counter is

        type state is (up,down);
        signal direction: state := down;
        signal count: std_logic_vector (18 downto 0) := (others=>'0');

begin

process (clk)
begin
        if (clk'event and clk='1') then
                case direction is
                        when up =>
                                if (count="1100001101001111111") then
                                        count <= count;
                                        sign <= '1';
                                        direction <= down;
                                else
                                        count <= count + 1;
                                        sign <= '0';
                                end if;
                        when down =>
                                if (count="0000000000000000000") then
                                        count <= count;
                                        sign <= '0';
                                        direction <= up;
                                else
                                        count <= count - 1;
                                        sign <= '1';
                                end if;
                        when others =>
                                count <= "0000000000000000000";
                                sign <= '0';
                end case;
        end if;
end process;

u_count <= count;

end Behavioral;
```

APPENDIX B

MATLAB Codes

B.1 MATLAB Code for Reference Counter

```
function ref_counter_config(this_block)

  % Revision History:
  %
  %   06-Feb-2012  (22:43 hours):
  %     Original code was machine generated by Xilinx's System Generator after parsing
  %     D:\School\tez\RF-DSP\Research\Implementation\desktop\x61\ref_counter.vhd

  this_block.setTopLevelLanguage('VHDL');

  this_block.setEntityName('ref_counter');

  % System Generator has to assume that your entity  has a combinational feed through;
  %   if it  doesn't, then comment out the following line:
  this_block.tagAsCombinational;


  this_block.addSimulinkOutport('ref_count');

  ref_count_port = this_block.port('ref_count');
  ref_count_port.setType('UFix_27_0');


  % Since the block has no inputs,  assuming output runs at system rate:
  outputRate = 1;
  this_block.addClkCEPair('clk','ce',outputRate);
  % (!) You may wish to modify this behavior.
  %     It is possible to create a black box from which the
  %     output rate is set from the block parameterization GUI;
  %     consult the System Generator documentation for details.
  for i = 1:this_block.numSimulinkOutports
     this_block.outport(i).setRate(outputRate);
  end
    % (!) Set the inout port rate to be the same as the first input
    %     rate. Change the following code if this is untrue.
    uniqueInputRates = unique(this_block.getInputRates);


  % Add addtional source files as needed.
  %  |-------------
  %  | Add files in the order in which they should be compiled.
  %  | If two files "a.vhd" and "b.vhd" contain the entities
  %  | entity_a and entity_b, and entity_a contains a
  %  | component of type entity_b, the correct sequence of
  %  | addFile() calls would be:
  %  |    this_block.addFile('b.vhd');
  %  |    this_block.addFile('a.vhd');
  %  |-------------

  %    this_block.addFile('');
  %    this_block.addFile('');
  this_block.addFile('ref_counter.vhd');

return;
```

B.2 MATLAB Code for Phase Frequency Detector

```
function pdf_config(this_block)

  % Revision History:
  %
  %   06-Feb-2012  (22:45 hours):
  %     Original code was machine generated by Xilinx's System Generator after parsing
  %     D:\School\tez\RF-DSP\Research\Implementation\desktop\x61\pdf.vhd
  %
  %

  this_block.setTopLevelLanguage('VHDL');

  this_block.setEntityName('pdf');

  % System Generator has to assume that your entity  has a combinational feed through;
  %   if it  doesn't, then comment out the following line:
  this_block.tagAsCombinational;

  this_block.addSimulinkInport('ref');
  this_block.addSimulinkInport('vco');

  this_block.addSimulinkOutport('up');
  this_block.addSimulinkOutport('down');

  up_port = this_block.port('up');
  up_port.setType('UFix_1_0');
  up_port.useHDLVector(false);
  down_port = this_block.port('down');
  down_port.setType('UFix_1_0');
  down_port.useHDLVector(false);

  % -----------------------------
  if (this_block.inputTypesKnown)
    % do input type checking, dynamic output type and generic setup in this code block.

    if (this_block.port('ref').width ~= 1);
      this_block.setError('Input data type for port "ref" must have width=1.');
    end

    this_block.port('ref').useHDLVector(false);

    if (this_block.port('vco').width ~= 1);
      this_block.setError('Input data type for port "vco" must have width=1.');
    end

    this_block.port('vco').useHDLVector(false);

  end  % if(inputTypesKnown)
  % -----------------------------

  % -----------------------------
   if (this_block.inputRatesKnown)
     setup_as_single_rate(this_block,'clk','ce')
   end  % if(inputRatesKnown)
  % -----------------------------

    % (!) Set the inout port rate to be the same as the first input
    %     rate. Change the following code if this is untrue.
    uniqueInputRates = unique(this_block.getInputRates);
  % Add addtional source files as needed.
  %  |-------------
  %  | Add files in the order in which they should be compiled.
  %  | If two files "a.vhd" and "b.vhd" contain the entities
  %  | entity_a and entity_b, and entity_a contains a
  %  | component of type entity_b, the correct sequence of
  %  | addFile() calls would be:
  %  |    this_block.addFile('b.vhd');
  %  |    this_block.addFile('a.vhd');
  %  |-------------

  %    this_block.addFile('');
  %    this_block.addFile('');
  this_block.addFile('pdf.vhd');

return;


% ------------------------------------------------------------

function setup_as_single_rate(block,clkname,cename)
  inputRates = block.inputRates;
  uniqueInputRates = unique(inputRates);
  if (length(uniqueInputRates)==1 & uniqueInputRates(1)==Inf)
    block.addError('The inputs to this block cannot all be constant.');
    return;
  end
  if (uniqueInputRates(end) == Inf)
     hasConstantInput = true;
     uniqueInputRates = uniqueInputRates(1:end-1);
  end
  if (length(uniqueInputRates) ~= 1)
    block.addError('The inputs to this block must run at a single rate.');
    return;
  end
  theInputRate = uniqueInputRates(1);
  for i = 1:block.numSimulinkOutports
     block.outport(i).setRate(theInputRate);
  end
  block.addClkCEPair(clkname,cename,theInputRate);
  return;

% ------------------------------------------------------------
```

B.3 MATLAB Code for Phase Counter

```
function p_counter_config(this_block)

  % Revision History:
  %
  %   06-Feb-2012  (22:45 hours):
  %     Original code was machine generated by Xilinx's System Generator after parsing
  %     D:\School\tez\RF-DSP\Research\Implementation\desktop\x61\p_counter.vhd
  %
  %

  this_block.setTopLevelLanguage('VHDL');

  this_block.setEntityName('p_counter');

  % System Generator has to assume that your entity  has a combinational feed through;
  %   if it  doesn't, then comment out the following line:
  this_block.tagAsCombinational;

  this_block.addSimulinkInport('up');
  this_block.addSimulinkInport('down');

  this_block.addSimulinkOutport('phase');

  phase_port = this_block.port('phase');
  phase_port.setType('UFix_16_0');

  % -----------------------------
  if (this_block.inputTypesKnown)
    % do input type checking, dynamic output type and generic setup in this code block.

    if (this_block.port('up').width ~= 1);
      this_block.setError('Input data type for port "up" must have width=1.');
    end

    this_block.port('up').useHDLVector(false);

    if (this_block.port('down').width ~= 1);
      this_block.setError('Input data type for port "down" must have width=1.');
    end

    this_block.port('down').useHDLVector(false);

  end  % if(inputTypesKnown)
  % -----------------------------

  % -----------------------------
   if (this_block.inputRatesKnown)
     setup_as_single_rate(this_block,'clk','ce')
   end  % if(inputRatesKnown)
  % -----------------------------

    % (!) Set the inout port rate to be the same as the first input
    %     rate. Change the following code if this is untrue.
    uniqueInputRates = unique(this_block.getInputRates);


  % Add addtional source files as needed.
  %  |-------------
  %  | Add files in the order in which they should be compiled.
  %  | If two files "a.vhd" and "b.vhd" contain the entities
  %  | entity_a and entity_b, and entity_a contains a
  %  | component of type entity_b, the correct sequence of
  %  | addFile() calls would be:
  %  |    this_block.addFile('b.vhd');
  %  |    this_block.addFile('a.vhd');
  %  |-------------

  %    this_block.addFile('');
  %    this_block.addFile('');
  this_block.addFile('p_counter.vhd');

return;


% ------------------------------------------------------------

function setup_as_single_rate(block,clkname,cename)
  inputRates = block.inputRates;
  uniqueInputRates = unique(inputRates);
  if (length(uniqueInputRates)==1 & uniqueInputRates(1)==Inf)
    block.addError('The inputs to this block cannot all be constant.');
    return;
  end
  if (uniqueInputRates(end) == Inf)
     hasConstantInput = true;
     uniqueInputRates = uniqueInputRates(1:end-1);
  end
  if (length(uniqueInputRates) ~= 1)
    block.addError('The inputs to this block must run at a single rate.');
    return;
  end
  theInputRate = uniqueInputRates(1);
  for i = 1:block.numSimulinkOutports
     block.outport(i).setRate(theInputRate);
  end
  block.addClkCEPair(clkname,cename,theInputRate);
  return;

% ------------------------------------------------------------
```

B.4 MATLAB Code for Control Input Counter

```
function u_counter_config(this_block)

  % Revision History:
  %
  %   06-Feb-2012  (22:47 hours):
  %     Original code was machine generated by Xilinx's System Generator after parsing
  %     D:\School\tez\RF-DSP\Research\Implementation\desktop\x61\u_counter.vhd
  %

  this_block.setTopLevelLanguage('VHDL');

  this_block.setEntityName('u_counter');

  % System Generator has to assume that your entity  has a combinational feed through;
  %   if it  doesn't, then comment out the following line:
  this_block.tagAsCombinational;

  this_block.addSimulinkOutport('u_count');
  this_block.addSimulinkOutport('sign');

  u_count_port = this_block.port('u_count');
  u_count_port.setType('UFix_19_0');
  sign_port = this_block.port('sign');
  sign_port.setType('Bool');
  sign_port.useHDLVector(false);

  % Since the block has no inputs,  assuming output runs at system rate:
  outputRate = 1;
  this_block.addClkCEPair('clk','ce',outputRate);
  % (!) You may wish to modify this behavior.
  %     It is possible to create a black box from which the
  %     output rate is set from the block parameterization GUI;
  %     consult the System Generator documentation for details.
  for i = 1:this_block.numSimulinkOutports
     this_block.outport(i).setRate(outputRate);
  end
    % (!) Set the inout port rate to be the same as the first input
    %     rate. Change the following code if this is untrue.
    uniqueInputRates = unique(this_block.getInputRates);


  % Add addtional source files as needed.
  %  |-------------
  %  | Add files in the order in which they should be compiled.
  %  | If two files "a.vhd" and "b.vhd" contain the entities
  %  | entity_a and entity_b, and entity_a contains a
  %  | component of type entity_b, the correct sequence of
  %  | addFile() calls would be:
  %  |    this_block.addFile('b.vhd');
  %  |    this_block.addFile('a.vhd');
  %  |-------------
  %    this_block.addFile('');
  %    this_block.addFile('');
  this_block.addFile('u_counter.vhd');

return;
```

APPENDIX C

Passive Loop Filter Transfer Function

C.1 Topology-I

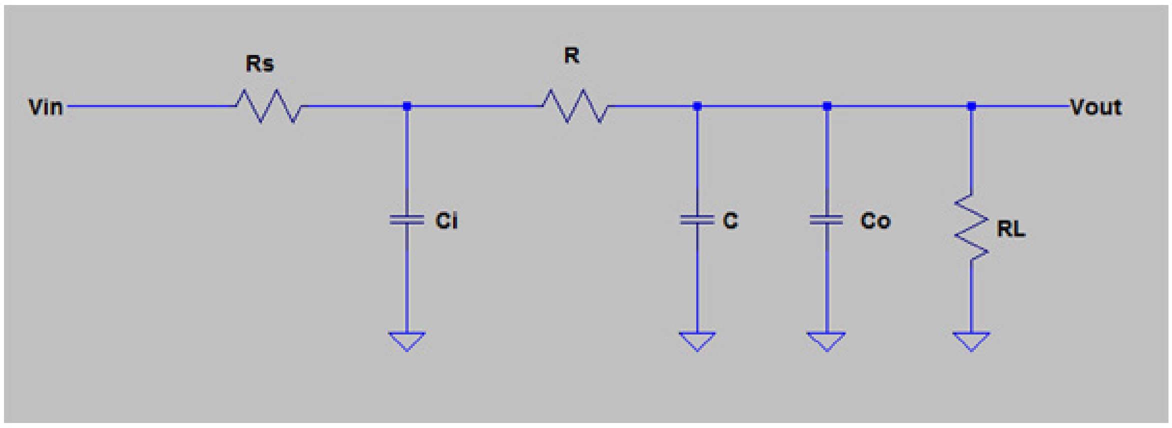


Figure C.1: Original Circuit

Due to equations in (C.1), we can represent the original circuit as a thevenin equivalent circuit.

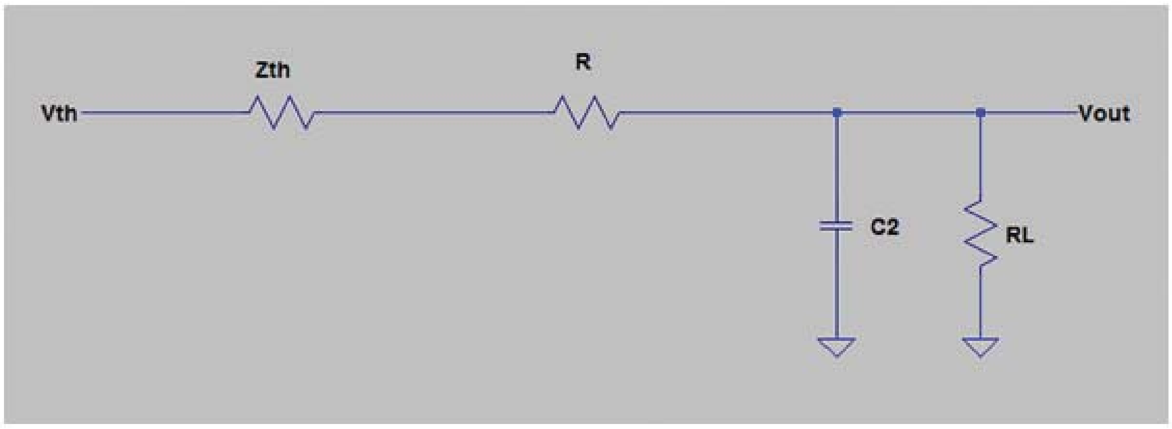


Figure C.2: Thevenin Equivalent of the Original Circuit

$$C_i = C_1 \qquad and \qquad C + C_o = C_2 \tag{C.1}$$

$$V_{th} = \frac{X_{C_1}\ V_i}{X_{C_1} +\ R_S} \qquad Z_{th} = \frac{X_{C_1}\ R_S}{X_{C_1} +\ R_S} \tag{C.2}$$

$$V_o = \frac{R_L X_{C_2} V_{th}}{R_L X_{C_2} + R_L Z_{th} + R_L R + X_{C_2} Z_{th} + X_{C_2} R} \tag{C.3}$$

By substituting (C.1) and (C.2) into (C.3), it is possible to obtain (C.4)

$$V_o = \frac{R_L X_{C_2} \frac{X_{C_1} V_i}{X_{C_1} + R_S}}{R_L X_{C_2} + R_L \frac{X_{C_1} R_S}{X_{C_1} + R_S} + R_L R + X_{C_2} \frac{X_{C_1} R_S}{X_{C_1} + R_S} + X_{C_2} R} \tag{C.4}$$

by simplification (C.4) becomes (C.5)

$$V_o = \frac{R_L X_{C_1} X_{C_2} V_i}{R_L X_{C_2}\left(X_{C_1} + R_S\right) + R_L X_{C_1} R_S + R_L R\left(X_{C_1} + R_S\right) + X_{C_2} X_{C_1} R_S + X_{C_2} R(X_{C_1} + R_S)} \tag{C.5}$$

After dividing both sides by V_i, it is possible to obtain the transfer function in the form of (C.6).

$$\frac{V_o}{V_i} = \frac{R_L X_{C_1} X_{C_2}}{R_L X_{C_2} X_{C_1} + R_L X_{C_2} R_S + R_L X_{C_1} R_S + R_L R X_{C_1} + R_L R R_S + X_{C_2} X_{C_1} R_S + X_{C_2} R X_{C1} + X_{C_2} R R_S} \tag{C.6}$$

$$X_{Ci} = \frac{1}{j\omega C_i} \quad \rightarrow \quad \mathcal{L}\{X_{Ci}\} = \frac{1}{sC_i} \qquad , \quad i = 1,\ 2,\ 3\ldots \tag{C.7}$$

Substitute the corresponding (C.7) equivalents into (C.6) and then take the Laplace Transform of (C.6) to obtain (C.8).

$$\frac{V_o}{V_i} = \frac{\frac{R_L}{s^2 C_1 C_2}}{\frac{R_L}{s^2 C_1 C_2} + \frac{R_L R_S}{s C_2} + \frac{R_L R_S}{s C_1} + \frac{R_L R}{s C_1} + R_L R R_S + \frac{R_S}{s^2 C_1 C_2} + \frac{R}{s^2 C_1 C_2} + \frac{R R_S}{s C_2}} \tag{C.8}$$

By simplification it is possible to obtain (C.9) from (C.8).

$$\frac{V_o}{V_i} = \frac{R_L}{R_L + sC_1 R_L R_S + sC_2 R_L R_S + sC_2 R_L R + s^2 C_1 C_2 R_L R R_S + R_S + R + sC_1 R R_S} \tag{C.9}$$

Finally, it is possible to rewrite (C.9) as (C.10) by grouping the terms in addition due to the order of s^n type coefficients.

$$\frac{V_o}{V_i} = \frac{R_L}{s^2(C_1C_2R_LRR_S) + s(C_1RR_S + C_2R_LR + C_2R_LR_S + C_1R_LR_S) + (R_S + R + R_L)} \quad \text{(C.10)}$$

$$\text{Equ(C.10)} = \text{Equ(C.10)} \cdot \frac{1/R_L}{1/R_L} \quad \text{(C.11)}$$

Moving from (C.11) it is possible to rewrite (C.10) as (C.12).

$$\frac{V_o}{V_i} = \frac{1}{s^2(C_1C_2RR_S) + s(\frac{C_1RR_S}{R_L} + C_2R + C_2R_S + C_1R_S) + (\frac{R_S}{R_L} + \frac{R}{R_L} + 1)} \quad \text{(C.12)}$$

We assume that;

$$0 < R_S \ll 1 \quad and \quad R_L \gg 1 \quad \text{(C.13)}$$

Thus, the final state of the transfer function is actually an approximation of (C.13), in accordance with the aforementioned magnitudes of R_S and R_L. The final version of the transfer function is shown as (C.14).

$$\frac{V_o}{V_i} = \frac{1}{sRC_2 + 1} \quad \text{(C.14)}$$

C.2 Topology-II

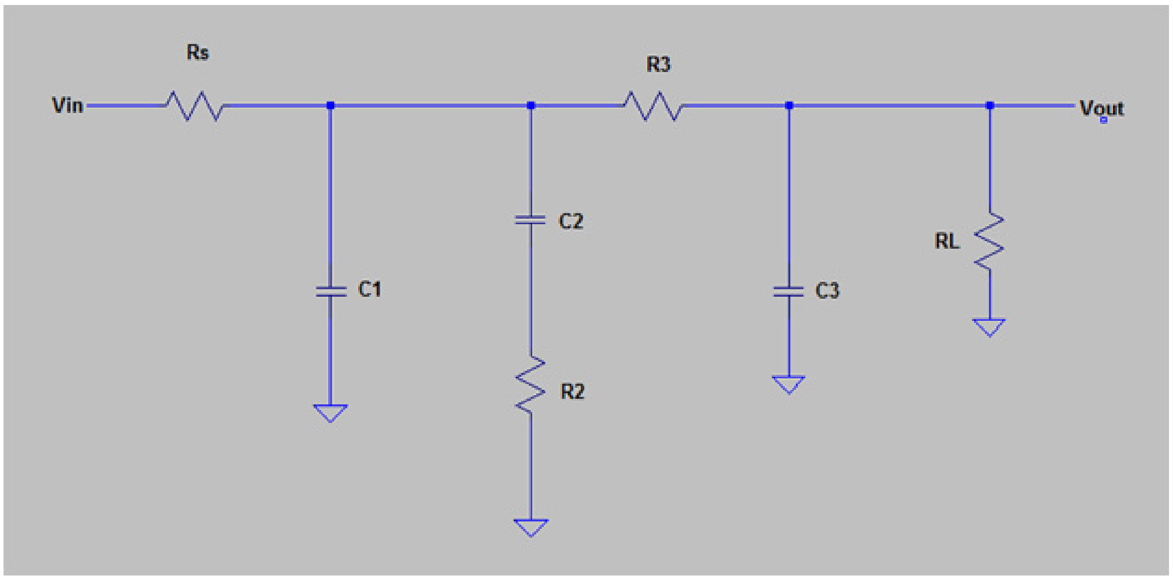


Figure C.3: Original Circuit

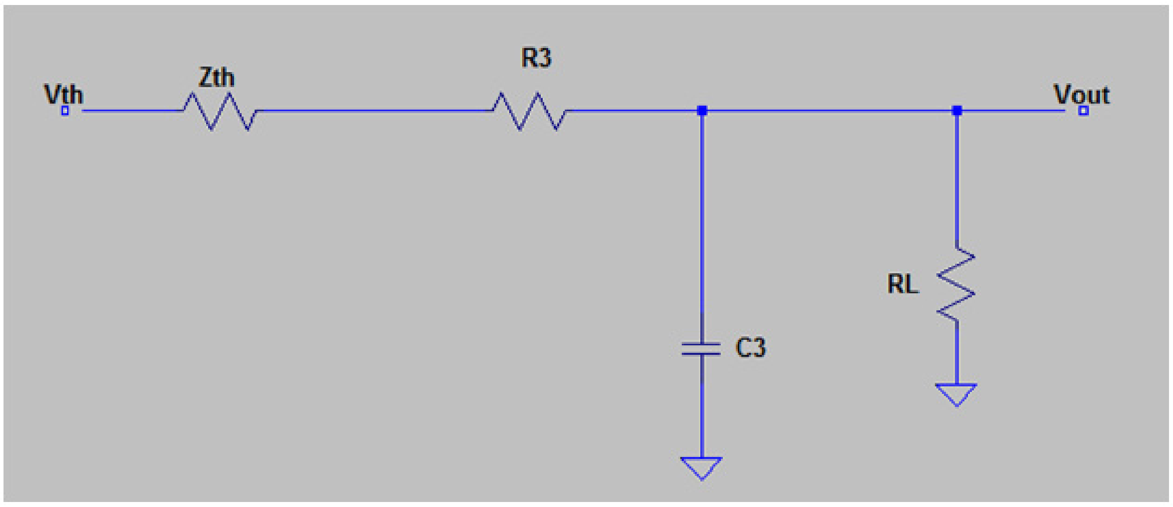


Figure C.4: Thevenin Equivalent of the Original Circuit

$$Z_{th} = \frac{R_S R_2 X_{C1} + R_S X_{C1} X_{C2}}{R_S R_2 + R_S X_{C2} + R_S X_{C1} + R_2 X_{C1} + X_{C1} X_{C2}} \tag{C.15}$$

$$V_{th} = V_{in} \frac{(R_2 X_{C1} + X_{C1} X_{C2})}{R_S R_2 + R_S X_{C2} + R_S X_{C1} + R_2 X_{C1} + X_{C1} X_{C2}} \tag{C.16}$$

$$V_{out} = \frac{R_L X_{C3} V_{th}}{X_{C3} R_L + R_L Z_{th} + R_L R_3 + X_{C3} Z_{th} + X_{C3} R_3} \tag{C.17}$$

Substitute (D.1) and (D.2) into (D.11) and obtain (D.4).

$$V_{out} = V_{in} \frac{\frac{R_L R_2 X_{C1} X_{C3} + R_L X_{C1} X_{C2} X_{C3}}{R_S R_2 + R_S X_{C2} + R_S X_{C1} + R_2 X_{C1} + X_{C1} X_{C2}}}{X_{C3} R_L + \frac{R_L R_S R_2 X_{C1} + R_L R_S X_{C1} X_{C2}}{R_S R_2 + R_S X_{C2} + R_S X_{C1} + R_2 X_{C1} + X_{C1} X_{C2}} + R_L R_3 + \frac{R_S R_2 X_{C1} X_{C3} + R_S X_{C1} X_{C2} X_{C3}}{R_S R_2 + R_S X_{C2} + R_S X_{C1} + R_2 X_{C1} + X_{C1} X_{C2}} + X_{C3} R_3} \tag{C.18}$$

Divide both sides of (D.4) by V_{in} and then simplify the right side to obtain (D.13).

$$\frac{V_{out}}{V_{in}} = \frac{R_L R_2 X_{C1} X_{C3} + R_L X_{C1} X_{C2} X_{C3}}{\begin{gathered} R_S R_L R_2 X_{C3} + R_S R_L X_{C2} X_{C3} + R_S R_L X_{C1} X_{C3} + R_2 R_L X_{C1} X_{C3} + R_L X_{C1} X_{C2} X_{C3} + R_S R_L R_2 X_{C1} + R_S R_L X_{C1} X_{C2} \\ + R_S R_L R_2 R_3 + R_S R_L R_3 X_{C2} + R_S R_L R_3 X_{C1} + R_L R_2 R_3 X_{C1} + R_L R_3 X_{C1} X_{C2} + R_S R_2 X_{C1} X_{C3} + R_S X_{C1} X_{C2} X_{C3} \\ + R_S R_2 R_3 X_{C3} + R_S R_3 X_{C2} X_{C3} + R_S R_3 X_{C1} X_{C3} + R_2 R_3 X_{C1} X_{C2} + R_3 X_{C1} X_{C2} X_{C3} \end{gathered}} \tag{C.19}$$

$$X_{Ci} = \frac{1}{j\omega C_i} \quad \rightarrow \quad \mathcal{L}\{X_{Ci}\} = \frac{1}{sC_i} \qquad , \; i = 1,\ 2,\ 3\ldots \tag{C.20}$$

Substitute the corresponding (C.20) equivalents into (D.13) and then take the Laplace Transform of (D.13) to obtain (D.14).

$$\frac{V_{out}}{V_{in}} = \frac{\frac{R_L R_2}{s^2 C_1 C_2} + \frac{R_L}{s^3 C_1 C_2 C_3}}{\begin{gathered} \frac{R_S R_L R_2}{sC_3} + \frac{R_S R_L}{s^2 C_2 C_3} + \frac{R_S R_L}{s^2 C_1 C_3} + \frac{R_2 R_L}{s^2 C_1 C_3} + \frac{R_L}{s^3 C_1 C_2 C_3} + \frac{R_S R_L R_2}{sC_1} + \frac{R_S R_L}{s^2 C_1 C_2} + R_S R_L R_2 R_3 + \frac{R_S R_L R_3}{sC_1} \\ + \frac{R_L R_2 R_3}{sC_1} + \frac{R_L R_3}{s^2 C_1 C_2} + \frac{R_S R_2}{s^2 C_1 C_3} + \frac{R_S}{s^3 C_1 C_2 C_3} + \frac{R_S R_2 R_3}{sC_3} + \frac{R_S R_3}{s^2 C_2 C_3} + \frac{R_S R_3}{s^2 C_1 C_3} + \frac{R_2 R_3}{s^2 C_1 C_3} + \frac{R_3}{s^3 C_1 C_2 C_3} \end{gathered}} \tag{C.21}$$

Simplify the right side of (D.14) to obtain (C.22).

$$\frac{V_{out}}{V_{in}} = \frac{sR_L R_2 C_3 + R_L}{\begin{gathered} s^2 R_S R_L R_2 C_1 C_2 + sR_S R_L C_1 + sR_S R_L C_2 + sR_L R_2 C_2 + R_L + s^2 R_S R_L R_2 C_2 C_3 + sR_S R_L C_3 + s^3 R_S R_L R_2 R_3 C_1 C_2 C_3 \\ + s^2 R_S R_L R_3 C_2 C_3 + s^2 R_L R_2 R_3 C_2 C_3 + sR_L R_3 C_3 + sR_S R_2 C_2 + R_S + s^2 R_S R_2 R_3 C_1 C_2 + sR_S R_3 C_1 + sR_S R_3 C_2 \\ + sR_2 R_3 C_2 + R_3 \end{gathered}} \tag{C.22}$$

$$\text{Equ(C.22)} = \text{Equ(C.22)} \frac{1/R_L}{1/R_L} \tag{C.23}$$

Moving from (C.23) it is possible to rewrite (C.22) as (C.24).

$$\frac{V_{out}}{V_{in}} = \frac{sR_2 C_3 + 1}{\begin{gathered} s^2 R_S R_2 C_1 C_2 + sR_S C_1 + sR_S C_2 + sR_2 C_2 + 1 + s^2 R_S R_2 C_2 C_3 + sR_S C_3 + s^3 R_S R_2 R_3 C_1 C_2 C_3 \\ + s^2 R_S R_3 C_2 C_3 + s^2 R_2 R_3 C_2 C_3 + sR_3 C_3 + \frac{sR_S R_2 C_2}{R_L} + \frac{R_S}{R_L} + \frac{s^2 R_S R_2 R_3 C_1 C_2}{R_L} + \frac{sR_S R_3 C_1}{R_L} + \frac{sR_S R_3 C_2}{R_L} \\ + \frac{sR_2 R_3 C_2}{R_L} + \frac{R_3}{R_L} \end{gathered}} \tag{C.24}$$

Finally, it is possible to rewrite (C.24) as (C.25) by grouping the terms in addition due to the order of s^n type coefficients.

$$\frac{V_{out}}{V_{in}} = \frac{s\,(R_2C_3) + 1}{\begin{gathered} s^3\,(R_S R_2 R_3 C_1 C_2 C_3) + s^2\left(R_S R_2 C_1 C_2 + R_S R_2 C_2 C_3 + R_S R_3 C_2 C_3 + R_2 R_3 C_2 C_3 + \frac{R_S R_2 R_3 C_1 C_2}{R_L}\right) + \\ s\left(R_S C_1 + R_S C_2 + R_2 C_2 + R_S C_3 + R_3 C_3 + \frac{R_S R_2 C_2}{R_L} + \frac{R_S R_3 C_1}{R_L} + \frac{R_S R_3 C_2}{R_L} + \frac{R_2 R_3 C_2}{R_L}\right) + \left(\frac{R_S}{R_L} + \frac{R_3}{R_L} + 1\right) \end{gathered}} \tag{C.25}$$

We assume that;

$$0 < R_S \ll 1 \quad and \quad R_L \gg 1 \tag{C.26}$$

Thus, the final state of the transfer function is actually an approximation of (C.25), in accordance with the aforementioned magnitudes of R_S and R_L. The final version of the transfer function is shown as (C.27).

$$\frac{V_{out}}{V_{in}} = \frac{s\,(R_2C_3) + 1}{s^2\,(R_2R_3C_2C_3) + s\,(R_2C_2 + R_3C_3) + 1} \tag{C.27}$$

APPENDIX D

PLL Transfer Function

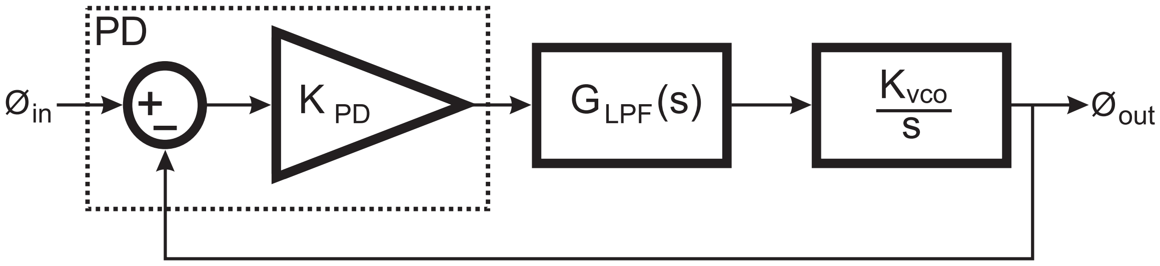


Figure D.1: Linear Model of a PLL

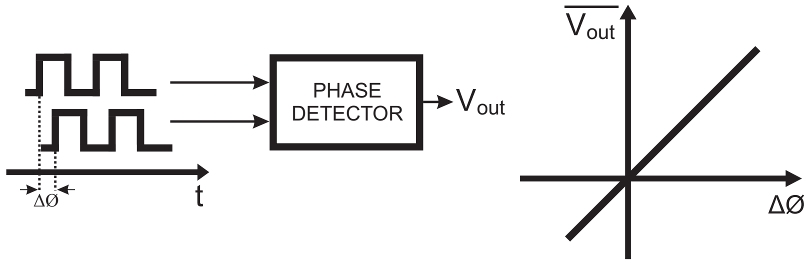


Figure D.2: Characteristic of an Ideal Phase Detector

$$\overline{V_{out}} = K_{PD}\Delta\phi \tag{D.1}$$

The open-loop transfer function of the PLL is therefore equal to:

$$H_o(s) = K_{PD}G_{LPF}(s)\frac{K_{VCO}}{s} \tag{D.2}$$

yielding the following close-loop transfer function:

$$H(s) = \frac{\Phi_{out}(s)}{\Phi_{in}(s)} \tag{D.3}$$

$$= \frac{K_{PD}K_{VCO}G_{LPF}(s)}{s + K_{PD}K_{VCO}G_{LPF}(s)} \tag{D.4}$$

In its simplest form, a low-pass filter is implemented as in Figure (D.3), with

$$G_{LPF}(s) = \frac{1}{1 + \dfrac{s}{\omega_{LPF}}} \tag{D.5}$$

where $\omega_{LPF} = 1/(RC)$. Equation (D.4) then reduces to

$$H(s) = \frac{K_{PD}K_{VCO}}{\dfrac{s^2}{\omega_{LPF}} + s + K_{PD}K_{VCO}} \tag{D.6}$$

The quantity $K = K_{PD}K_{VCO}$ is called the *loop gain*.

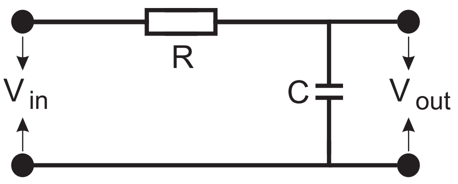


Figure D.3: Simple Low-Pass Filter

ζ is the damping factor and ω_n is the natural frequency

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \tag{D.7}$$

where

$$\omega_n = \sqrt{\omega_{LPF}K}, \tag{D.8}$$

$$\zeta = \frac{1}{2}\sqrt{\frac{\omega_{LPF}}{K}}. \tag{D.9}$$

Phase error transfer function defined as $H_e(s) = \Phi_e(s)/\Phi_{in}(s)$

$$\begin{aligned} H_e(s) &= 1 - H(s) \\ &= \frac{s^2 + 2\zeta\omega_n s}{s^2 + 2\zeta\omega_n s + \omega_n^2} \end{aligned} \tag{D.10}$$

The output excess phase is given by

$$\begin{aligned}\Phi_{out}(s) &= H(s)\Phi_{in}(s) \\ &= \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}\frac{\Delta\omega}{s^2}\end{aligned} \tag{D.11}$$

which is the response of a second-order system to a *ramp input*.

Phase error is:

$$\Phi_e = \Phi_{in} - \Phi_{out} \tag{D.12}$$

therefore in frequency domain, the phase error is:

$$\begin{aligned}\Phi_e(s) &= H_e(s)\Phi_{in}(s) \\ &= \frac{s^2 + 2\zeta\omega_n s}{s^2 + 2\zeta\omega_n s + \omega_n^2}\frac{\Delta\omega}{s^2}\end{aligned} \tag{D.13}$$

whose final value is given by

$$\begin{aligned}\Phi_e(t = \infty) &= \lim_{s\to 0} s\Phi_e(s) \\ &= \Delta\omega\frac{2\zeta}{\omega_n} \\ &= \frac{\Delta\omega}{K}\end{aligned} \tag{D.14}$$

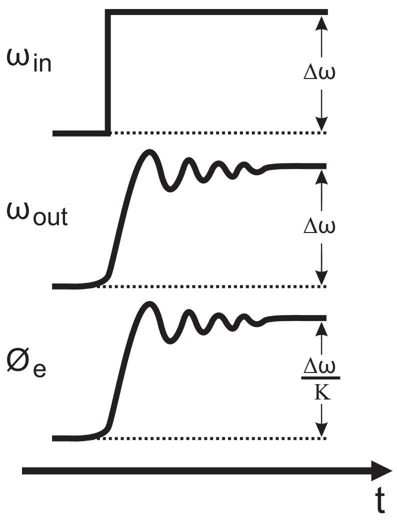


Figure D.4: Response of a PLL to a Frequency Step