Digital Implementation of a Second-order Costas Loop Demodulator

by

David Pérez Feliciano

A Project report submitted in partial fulfillment
of the requirements for the degree of

MASTER OF ENGINERING
in
ELECTRICAL ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
MAY, 2004

Approved by:

[redacted]
Jose Luis Cruz Rivera, Ph.D.
Member, Graduate Committee — 5/17/04 Date

[redacted]
Henric M. Jerkic Vidmar, Ph.D.
Member, Graduate Committee — 5-17-04 Date

[redacted]
Rafael Fernández Sein
President, Graduate Committee — 17 May 2004 Date

[redacted]
Ana Carmen González
Representative of Graduate Studies — 17 May 2004 Date

[redacted]
Ortiz, Jorge L.
Chairperson of the Department — May 17, 2004 Date

Abstract

This project describes the digital implementation of a Second-Order Costas Loop demodulator using simulation software SystemView by Elanix. The digital design of the system is based on direct transformation of every analog component of the Costas Loop to its respective discrete-time and subsequent digital domain. The relation between every component in the analog system that has to do with carrier tracking and synchronization is based on analog phase-locked loop theory. We used Costas Loop theory, which describes the Costas Loop as two phase locked-loops operating in quadrature phase to each other, to develop a linear model practically identical to that of the analog phase-locked loop. The difference between these two models consists of the phase detector gain, which is evident when establishing the phase detector gain of each system in relation to the amplitude of the input and output signals. We then used this model and phase-locked loop theory to develop the analog design of the Costas Loop and its subsequent transformation to the digital domain. The design of the arm filters, however, did not follow the derived linear model. Instead, we used conventional Costas Loop theory and the filter design tools of SystemView to design these two filters. The final digital implementation of the system was then realized using the design and simulation tools of SystemView in addition to fixed-point arithmetic theory to represent the parameters and the values of the signals processed.

Resumen

Este proyecto describe la implementación digital de un Costas Loop de Segundo orden utilizando simulaciones llevadas a cabo con el software SystemView, hecho por Elanix. El diseño digital del sistema está basado en la transformación directa de cada componente análogo a su respectivo dominio discreto o digital. La relación que existe entre cada componente del sistema análogo que controla la adquisición y sincronización de la señal portadora (carrier), está basada en la teoría del Phase-Locked Loop análogo. Para ésto utilizamos la teoría del Costas Loop, que lo describe a éste como un sistema compuesto por dos Phase-Locked Loops que operan con un desfase de noventa grados el uno del otro, para asi desarrollar un modelo lineal prácticamente idéntico al del Phase-Locked Loop análogo. La diferencia entre ambos modelos se encuentra en la ganancia del detector de fase, la cual se puede apreciar cuando establecemos la relación entre la ganancia de los detectores de fase de cada sistema y la amplitud de las señales de entrada y salida. Una vez obtenido el modelo, lo utilizamos en conjunto con la teoría de Phase-Locked Loops análogos para desarrollar el diseño análogo del Costas Loop y su transformación al dominio digital. El diseño de los filtros laterales (Arm Filters) no se llevó a cabo utilizando este modelo lineal. En lugar de ésto, utilizamos la teoria convencional del Costas Loop y las herramientas de diseño de filtros de SystemView. Finalmente, llevamos a cabo la implementación del sistema digital utilizando las herramientas de diseño y simulación de SystemView en conjunto con la teoria matemática fixed-point (fixed-point arithmetic) para representar los parámetros de cada componente del sistema y de las señales procesadas por éste.

Copyright disclosure and consent

I hereby authorize the Library of the University of Puerto Rico at Mayagüez to allow partial or complete copying of this document for research purposes.

© Copyright by

David Pérez-Feliciano

2004

Dedicated to

To my parents David and María, my grandmother Ramonita, and Professor Rafael Fernandez Sein for their guidance, constant support and encouragement that kept me focused and motivated throughout these years, making possible the completion of this project.

To my brothers Elvis, José, Ángel and Luis, my sisters María and Rosín, my nephew Ángel, and nieces Cristina and Jarixa, for being there for me when I needed them the most and to encourage them to persevere and continue working hard to reach their goals.

To Carmen Lugo, for giving me the opportunity to be part of the NASA/UPRM partnership for space telecommunications and education, which led to the summer internship jobs at NASA Goddard Space Flight Center.

To Madeline Butler, Deputy Chief of Engineers at NASA Goddard Space Flight Center for the time she devoted to mentor me during the summer jobs, encouraging me to improve my English and stay focused throughout the completion of my assignment.

To Dave Israel and Scott D. Hoy for the time devoted to being my mentors during the summer jobs I spent at NASA Goddard Space Flight Center.

Acknowledgements

To Professor Rafael Fernandez Sein for his patience, guidance, suggestions, and the time and effort spent correcting this document. Moreover, for giving me the opportunity to work during two consecutive summer internship jobs at NASA Goddard Space Flight Center in Maryland, experience that motivated me to continue graduate studies and helped me select the subject of this project.

To the members of the Graduate Committee, Dr. José L. Cruz Rivera and Dr. Mario Ierkic, for the time and effort spent correcting this document.

To my sister María and my brother Elvis for the time they spent helping me submit this document while I was out of the country.

Table of Content

List of Tables

List of Figures

List of Appendices

1 Introduction

Coherent detection and demodulation requires the utilization of synchronization systems that extract carrier phase and frequency information from the received signal. Phase and frequency are two parameters used by synchronization systems, such as Phase-Locked Loops (*PLL*) to track, acquire and synchronize to the carrier of the received signal. Making use of additional components, PLLs can be used directly to demodulate a signal when the signal contains a positive average energy at its carrier frequency. Old communications systems used to transmit and receive signals that had an average or residual energy at the carrier frequency in order to reduce complexity and cost of the demodulation design. Modern communications systems, on the other hand, do not use this method, as the residual energy is considered to be wasted energy, since it does not transmit any data; however, the carrier is required for the PLL in the receiver to synchronize to. In practice, techniques that conserve power are of interest, hence modern communications systems use suppressed-carrier modulation/demodulation techniques, which do not require a residual energy at the carrier frequency. Using suppressed-carrier modulation techniques, present a problem for PLLs, since in the absence of the carrier, PLLs cannot track, acquire and synchronize to the received signal. Therefore, another synchronization system must be used instead. An example of such a system is the Costas Loop.

The Costas Loop is a synchronization system that was first introduced by John P. Costas in 1956 to achieve phase tracking, acquisition, synchronization and demodulation of double-sideband suppressed-carrier AM signals. The components of the system are the

Arm Filters, the Loop Filter, three multipliers that we will refer to as Phase Detectors, and a Voltage-Controlled Oscillator (VCO). The Costas Loop is able to obtain the phase and frequency information of the modulated carrier and achieve phase tracking, acquisition and synchronization to this extracted carrier while demodulating and extracting the data contained in the received signal. Although the original intent of the Costas Loop was to track and demodulate double-sideband suppressed-carrier AM signals, it can very well be used to demodulate other suppressed-carrier modulation techniques. Another modulation technique for which the Costas Loop is readily used without modifications is Binary Phase Shift Keying (BPSK). The truth of this statement lies in the fact that BPSK signals can be expressed and demodulated as double-sideband suppressed-carrier AM signals. Other applications of the Costas Loop are QPSK (Quadrature Phase Shift Keying), 8-PSK, OQPSK, FSK (Frequency Shift Keying), FM and spread spectrum.

The Costas Loop can be considered to be a variation of a PLL system. Costas Loop theory describes the system as two phase-locked loops operating in phase quadrature to each other. We used this theory to develop a linear model of the Costas Loop (analog system) that is practically identical to that of the analog phase-locked loop. The difference between these two models is established at the phase detector gain term for each system. When the linear modeled was derived, we noticed that there was a publication by (Kamperman 1994 [12]) that attempted to do the same. As the Costas Loop has two multipliers that independently detect phase differences between the input and output signals, and a third multiplier that adds these two phase differences and eliminates the modulation effects attached to them (that is, in

addition to increasing the gain requirements), we end up with a gain term equal to the square of the phase detector gain in the PLL. This difference is evident when establishing the relation between the phase detector gain of each system to the amplitude of their respective input and output signals. We then used this model and phase-locked loop theory to develop the analog design of the Costas Loop and its subsequent transformation to and implementation on the digital domain. The design of the arm filters, however, did not follow the derived linear model. Instead, we used the Costas Loop theory and the filter design tools of SystemView to design two low-pass filters of the Bessel-type. A Bessel-type filter was chosen for the arm filters because the phase response of these filters is constant at 0dB inside the designated bandwidth area. Once all the parameters of the analog system were defined, we proceeded to define those of the digital system. Then we used the design and simulation tools of SystemView, in addition to fixed-point arithmetic theory, to represent the parameters of the digital system and the values of the signals processed.

This project describes the digital implementation of a Second-Order Costas Loop demodulator using simulation software SystemView by Elanix. The digital design of the system is based on direct transformation of every analog component of the Costas Loop to its respective discrete-time and subsequent digital domain. In order to provide thorough details of the design process, this document also includes the theory of analog phase-locked loop and Costas Loop systems, derivation of the Costas Loop linear model used in the design, transformation of the analog PLL and Costas Loop systems to a discrete-time and subsequent digital system, fixed-point arithmetic theory, implementation and simulation of the analog and digital Costas Loop using SystemView, and the final discussion of the results.

2 Revision of Literature

The Costas Loop is a synchronization system introduced by John P. Costas in 1956 to address some of the problems and concerns facing coherent communications systems at the time. Back then, most commercial and military communications systems were using amplitude modulation techniques to transmit information. Double-sideband transmitted carrier was the preferred amplitude modulation technique employed due to the simplicity of the systems that could be used to implement it. Although inefficient, energy wise, transmitting the carrier along with the modulated information allowed the use of phase-locked loop technology to achieve coherent demodulation of the data, greatly improving the demodulation process. But the increasing need for more efficient (in terms of the energy and bandwidth used) and reliable communications systems laid down the grounds for new modulation and demodulation techniques to be proposed; this was the case for the Costas Loop. The Costas Loop significantly improved the amplitude modulation/demodulation techniques by allowing the efficient demodulation of double-sideband suppressed-carrier signals. Doing so, it made possible a more efficient transmission of double-sideband AM signals by allowing the removal of the additional carrier signal formerly transmitted with the modulated information, and thereby reducing the amount of energy needed during the transmission process. In addition to demodulating double-sideband suppressed-carrier AM signals, Costas Loops are used to demodulate BPSK, QPSK, 8-PSK, OQPSK, FSK (Frequency Shift Keying), and FM signals, for example. This characteristic of the Costas Loop makes it suitable for many applications. Some examples are satellite communications, spread spectrum, and CDMA (used in cellular telephone communications).

The Costas loop is a synchronization system that could be analyzed as two phase-locked loops that operate in phase quadrature to each other. Consequently, the operation of any Costas Loop system is influenced by the operation of these two PLLs to a great extent. For this reason, this project devotes a great deal of time and effort to PLL theory and to establish its relation to Costas Loop systems. The following paragraphs are allotted to establishing the progress of phase-locked loop technology and its relation to other technological advances as well as Costas Loop systems.

The first publications about Phase-Locked Loop (PLL)-like systems appeared in 1923 and 1932. In 1923, E. V. Appleton published his paper "Automatic synchronization of triode oscillators", and H. de Bellescize published "La reception synchrone" in 1932. However, it is Henri de Bellescize, a French engineer, who is considered to be the inventor of coherent communications after he designed a vacuum tube based synchronous demodulator for an AM receiver, (Best 1999 [1]). Although the concept was readily available, the use of PLLs did not become as important until the 1940's when engineers at the Jet Propulsion Laboratories were faced with the problem of providing telemetry and radio guidance for short-range ballistic missiles and producing reliable communications under conditions of heavy interference, (Lindsey et al 1991 [8]). To solve these problems, a great deal of study and expansion was given to the earlier work done by C. Shannon, N. Wiener and the staffs of MIT Radiation and Lincoln Laboratories. Following these efforts, the theory and operation of servomechanisms functioning in the presence of noise was developed. Then in the early fifties, E. Rechtin and R. Jaffe applied this theory to develop phase-locked loop receivers, and other PLL

applications such as automatic gain control systems and coherent two-way velocity and range measuring systems. Since then, PLLs have contributed to coherent communications systems and other technological advances.

Another important application, and one of the firsts for PLLs, also occurred during the 1950s: a PLL was used to recover the color sub-carrier in television, (Best 1999 [1]). Originally, PLLs were mostly used for military applications and did not find broader industrial and commercial use until they became available as an integrated circuit. The first PLL ICs appeared around 1965 and were purely analog devices, (Best 1999 [1]). Signetics and RCA developed the first analog PLL ICs: the NE565 and CD4046, (Hsieh et al 1996 [16]). The PLL integrated in those chips included a sinusoidal phase detector, a loop filter, and a voltage-controlled oscillator. The phase detector was a four-quadrant multiplier that had a phase-lock range that extended from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$. In 1970 these PLL ICs were being used on speed control systems for synchronous and DC motors, (Hsieh et al 1996 [16]). As IC and digital technology was improving, it was possible to extend the phase error detection range of the PLL by using a sawtooth comparator as the phase detector. The phase error detection range for this phase detector was $-\pi$ to π. Later in 1972, Motorola manufactured the sequential phase/frequency detector (PFD) chip MC4044, capable of detecting phase error changes that extended from -2π to 2π. This phase detector was combined with the analog components of the NE565 analog phase-locked loop to achieve the realization of a hybrid PLL system used to control the speed of a synchronous motor. This combination allowed a speed regulation of 0.002%, (Hsieh et al 1996 [16]).

As many applications of analog PLLs started developing, by 1970 the theoretical description of these systems was very well established, and plenty of information could be found in [A. Blanchard, "Phase-Locked Loops: Application to Coherent Receiver Design", F. M. Gardner, "Phaselock Techniques", and W. C. Lindsey, "A survey of digital phase-locked loops"]. This theoretical description included noise and nonlinear analysis of the loop, variety of phase detectors, threshold performance and determination, modifications to the system based on signal and channel type, and established important design parameters such as bandwidth, steady-state error, mean-square phase error, lock-in range, pull-in range, acquisition time, filter parameters and SNR behavior for low CNR at the input, (Gupta 1975 [13]). Also, there was some work being done to develop other types of phase-locked loops that could be implemented digitally and a variety of systems were proposed. These other phase-locked loops were categorized at the time as Hybrid, Discrete and all-digital phase-locked loops.

The Hybrid phase-locked loop is defined as an analog phase-locked loop where one or more, but not all elements in the loop are digital, (Gupta 1975 [13]). This type of implementation enables the use of more efficient components in the analog system to improve its performance. However, unlike the analog PLL (second-order, for example), system stability has to be analyzed, given that a high system gain could make this loop unstable. The Discrete phase-locked loop used a sampler as the phase detector and the other components were a digital filter and a digital oscillator. This phase-locked loop tracks and synchronizes to the input signal (sine wave) by predicting zero crossing points on this signal. Unfortunately, the architecture of the loop leads to twice as many nonlinearities in the system

equations, compared to the analog phase-locked loop (APLL). These nonlinearities are the result of the non-uniform sampling and sinusoidal nonlinearities. Hence, this phase-locked loop is more difficult to analyze than an APLL. Finally, we have the all-digital phase-locked loop, for which there were various systems proposed. The first all-digital phase-locked loop proposed consisted of replacing all analog components by digital ones. Other digital phase-locked loops were proposed based on specific applications such as harmonic signal filtration and FM demodulation.

Progress on the study of digital phase-locked loops continued and in 1982, (Lindsey et al 1981 [18]) provided an extensive report on digital phase-locked loop technology that covered the period from 1960 to 1980. In this paper, (Lindsey et al 1981 [18]) classified the digital phase-locked loops into four categories, based on the type of phase detector used. These four categories were: Flip-Flop (FF), Nyquist Rate (NR), Zero Crossing (ZC), and Lead/Lag (LL) digital phase-locked loops (DPLL). The FF-DPLL, obtains phase error information from the duration between the set and reset time of the flip-flop triggered by positive zero crossings of the input signal and the local clock. In the NR-DPLL, the input signal is sampled at the Nyquist rate, then the resulting samples are multiplied digitally with the samples of the local reference to generate the required phase error. The ZC-DPLL has two variations. In the first version, the loop samples the input signal in the positive going zero crossings. The second version, however, samples the input signal in both positive and negative going zero crossing points. Finally, for the LL-DPLL the phase detector determines at each cycle of the clock whether the input leads or lags the locally generated clock; if so, the phase is adjusted accordingly. (Lindsey et al 1981 [18]) also provided a thorough analysis

of each system, including steady-state error. It also provided equations in the z-domain for the common excitation signals: phase step, phase ramp and phase parabola; these are the equations used in this project for the analysis of the digital Costas Loop designed.

Another extensive coverage of phase-locked loop theory is provided in, (Best 1999 [1]). In this recent publication, (Best 1999 [1]) covers all aspects of the phase-locked loop theory from a professional engineering point of view, starting with analog phase-locked loops and ending with a category that he describes as the software phase-locked loop. The phase-locked loops covered by (Best 1999 [1]) are essentially those covered by (Gupta 1975 [13]) and (Lindsey et al 1981 [18]), except the categories are named differently. For example, the analog phase-locked loop is described as the linear phase-locked loop, the hybrid PLL is called the classical digital phase-locked loop, and the group composed by the flip-flop, Nyquist rate and zero crossing (including another one that uses a Hilbert transform phase detector) is called all digital phase-locked loops. The category called the software phase-locked loop is essentially a software implementation of all the previous categories already mentioned, including the analog phase-locked loop. This theory developed in this work for a software implementation of the analog PLL that we use to implement the digital Costas Loop. To do so we first used analog Costas Loop and phase-locked loop theory to derive a linear model for the analog Costas Loop. Then we used the results obtained for the software version of the analog PLL to obtain a digital version of the analog Costas Loop. Finally, a design implementation and simulation of the digital Costas Loop was achieved using software SystemView by Elanix.

As an extension to this chapter, chapters three to six have been included to provide a through discussion of the phase-locked loop/Costas Loop subject and its migration to the digital implementation provided in this project.

3 Analog Phase-Locked Loop Theory

3.1 The Analog Phase-Locked Loop: An Overview

A Phase-locked loop is a circuit that generates a signal that is synchronized to the input signal. To achieve synchronization, phase and frequency parameters of both signals are compared and an error signal is produced. This error signal is used to correct the phase and frequency of the output signal in such a way that the error reduces to zero or a minimum. When the error signal reaches or

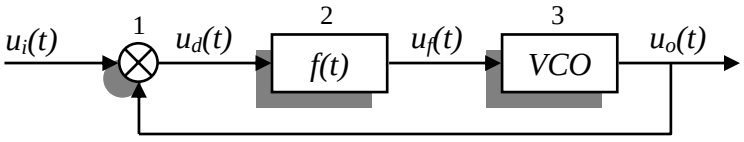

Figure 3-1. Time-Domain Block Diagram of a PLL

settles to a minimum value, we say the signals are synchronized and the phase-locked loop (PLL) is locked. A block diagram of a PLL can be seen in figure 3-1. The diagram shows the three major components of the PLL along with the corresponding signals. The three major components of the loop, from left to right are 1) Phase Detector, 2) Loop filter, and 3) Voltage-Controlled Oscillator (VCO). (A circle with a cross denotes the phase detector and $f(t)$ the loop filter.) PLL signals are defined as follows:

$$u_i(t) = A_i \sin(\omega_i t + \theta_i) \quad (3\text{-}1),$$

$$u_o(t) = A_o \sin(\omega_o t + \varphi_o) \quad (3\text{-}2),$$

$$u_d(t) = \frac{A_i A_o}{2}\cos[(\omega_i - \omega_o)t + \theta_i - \varphi_o] - \frac{A_i A_o}{2}\cos[(\omega_i + \omega_o)t + \theta_i + \varphi_o] \quad (3\text{-}3),$$

$$u_f(t) = \frac{A_i A_o}{2}\cos[(\omega_i - \omega_o)t + \theta_i - \varphi_o] \quad (3\text{-}4).$$

The input signal, $u_i(t)$ is a sine waveform with amplitude A_i, radian frequency ω_i, and phase θ_i. $u_o(t)$ is the form of the output signal just before the tracking and acquisition process of the input signal begins. It also is a sine waveform with amplitude A_o, radian frequency ω_o, and phase φ_o. When the tracking and acquisition process is taking place, the PLL will operate in such a way that $\omega_o \to \omega_i$ and the phase error between input and output reduces to zero or a minimum. Since the phase detector is sinusoidal (four-quadrant multiplier), its output signal is as given by $u_d(t)$, which results in a signal composed of two cosine waveforms with frequency components $(\omega_i - \omega_o)$ and $(\omega_i + \omega_o)$. (Think of the four-quadrant multiplier as a device that performs the same type of multiplication operation we would perform in a calculator, or by hand, with any two numbers.) Now let us take a look at each argument of the cosine terms. The argument of the first cosine, $[(\omega_i - \omega_o)t + \theta_i - \varphi_o]$ sustains a phase and frequency difference between the input and output signals, whereas the argument of the second term results in phase and frequency addition. Given this phase and frequency addition, $\frac{A_i A_o}{2}\cos[(\omega_i + \omega_o)t + \theta_i + \varphi_o]$ is considered the high-frequency component of $u_d(t)$, also known as the AC component of $u_d(t)$ because it will always be an oscillating signal of frequency $(\omega_i + \omega_o)$. On the other hand, it is $\frac{A_i A_o}{2}\cos[(\omega_i - \omega_o)t + \theta_i - \varphi_o]$, the first term of $u_d(t)$, that contains information about the phase and frequency error between $u_i(t)$ and $u_o(t)$. When the error between the signals is zero or minimum, this term will tend to a DC value. Moreover, if the error between the signals is large, but the PLL is following

the input signal correctly, $\frac{A_i A_o}{2}\cos[(\omega_i - \omega_o)t + \theta_i - \varphi_o]$ will induce an average DC voltage that takes the loop to the correct lock-in direction (synchronization takes place). For this reason, this term is known as the DC component of $u_d(t)$.

The process continues in the loop filter. Think of the loop filter as a low-pass filter with 3-db bandwidth lower than $(\omega_i + \omega_o)$ but higher than $(\omega_i - \omega_o)$. Consequently, the output signal of the loop filter $u_f(t)$, will be solely composed of $\frac{A_i A_o}{2}\cos[(\omega_i - \omega_o)t + \theta_i - \varphi_o]$, as shown by equation (3-4). (This is true if we assume an ideal filtering process.) In the previous paragraph it was pointed out that only the first cosine term of $u_d(t)$ provided useful information to sustain the proper operation of the PLL. This is the low-frequency component of $u_d(t)$, which justifies selecting a low-pass filter as the loop filter. Finally, $u_f(t)$ frequency modulates the output signal of the VCO, adjusting its phase and frequency until they match those of the input signal, $u_i(t)$. We say $u_f(t)$ frequency modulates the output signal of the VCO because before the error between the signals is close to zero, $u_f(t)$ oscillates. In fact, this is the same process used during frequency modulation (FM).

The frequency of the VCO signal is given by (Best 1999 [1])

$$\omega_o = \omega_{0c} + K_o u_f(t) \qquad (3\text{-}5),$$

where ω_{0c} is its center frequency. It is noticeable that $\omega_o = \omega_{0c}$ when $u_f(t) = 0$. For this reason, it is a design rule to set ω_{0c} equal to the frequency of the signal we expect to receive, ω_i. From the above equations and discussion, a PLL can be seen as a servo control system that establishes synchronization generating an estimated replica of the input signal. (This is true if the input signal is not carrying any type of modulation. However, if the received signal is carrying some type of information (received signal is modulated), then the PLL will lock to its carrier.)

3.2 Time Domain Analysis

Let us analyze the previous equations to derive a more detailed analysis of PLL in the time domain. Assuming the PLL is not locked (the loop is in the process of tracking and acquiring the input signal to achieve synchronization), the equations that govern its operation are as given in the previous section. Furthermore, presume the PLL is operating in the lock-in range and, as a result, synchronization occurs considerably fast. Therefore the synchronization process can be explained as follows.

When the synchronization process starts, both the input and output signals are multiplied in the phase detector generating the signal $u_d(t)$. This signal goes to the loop filter, which eliminates the high-frequency component of $u_d(t)$, producing the signal $u_f(t)$. This signal is fed to the VCO to adjust its output signal in a favorable way as to reduce the difference between its parameters and those of the input. We call this difference the error between the signals. When the error is reduced, synchronization takes place, signifying that the frequency and phase of the output signal match those of

the input signal. Examining $u_f(t)$ we see that $(\omega_i - \omega_o)$ can only be zero if $\varphi_o = [(\omega_i - \omega_o)t + \phi_o]$, which defines φ_o as a function of time. Substituting φ_o into $u_f(t)$ we obtain $u_f(t) = \frac{A_i A_o}{2}\cos[(\omega_i - \omega_o)t + \theta_i - [(\omega_i - \omega_o)t + \phi_o]]$. Simplifying this equation we obtain a representation for $u_f(t)$ totally dependent on the phase of the signals

$$u_f(t) = \frac{A_i A_o}{2}\cos(\theta_i - \phi_o) \qquad (3\text{-}6).$$

Having the PLL eliminated the frequency difference, it will modify the phase of the VCO to equate it to the input until $u_f(t) = 0$. Equation (3-6) establishes that $u_f(t)$ can be equal to zero if the argument of the cosine is $\frac{\pi}{2}$. Hence, defining ϕ_o as $\theta_o - \frac{\pi}{2}$ and placing it into the argument of equation (3-6) we obtain $u_f(t) = \frac{A_i A_o}{2}\cos(\theta_i - \theta_o + \frac{\pi}{2})$. Re-writing this equation yields

$$u_f(t) = \frac{A_i A_o}{2}\sin(\theta_i - \theta_o) \qquad (3\text{-}7).$$

The derivation of equation (3-7) indicates that synchronization occurs when the input and output signals of the PLL are in phase quadrature to each other; that is, they are 90 degrees or $\frac{\pi}{2}$ radians apart. In PLL theory, it customary to think of the $\sin(\theta_i - \theta_o)$ as θ_e, the phase error of the PLL. When the loop is locked, (the difference between θ_i and θ_o is small), we can use the small angle approximation to simplify θ_e to $(\theta_i - \theta_o)$. Moreover, defining a new gain term, K_d, and equating it to $\frac{A_i A_o}{2}$ we obtain a more useful linear approximation for equation (3-7),

$$u_f(t) \approx K_d(\theta_i - \theta_o) \qquad (3\text{-}8),$$

where K_d will represent the gain of the phase detector. Now, having a linear expression for $u_f(t)$, we can rewrite $u_d(t)$ as

$$u_d(t) \approx K_d(\theta_i - \theta_o) + \text{high - frequency terms} \qquad (3\text{-}9).$$

In order to justify equation (3-7), the VCO signal has to be changed from a sine to a cosine waveform. This exemplifies the quadrature phase difference between input and output signals mentioned before. We can use the trigonometric identity about the product of two sinusoids to prove that the VCO signal has to be a cosine waveform when the input to the PLL is a sine wave, $\sin(A)\cos(B) = \frac{1}{2}\left[sen(A-B) + sen(A+B)\right]$. Remember that we started the explanation defining both input and output signals as sine waveforms. Then, after some derivation process, it was shown that they had to be in quadrature for the PLL to lock. Since the input signal is fixed, only the output signal can be varied. Consequently, it is the output signal that changes to a cosine waveform. Now going back to the trigonometric identity, having the input expressed as $A_i \sin(\omega_i t + \theta_i)$ and the output as $A_o \cos(\omega_i t + \theta_o)$, multiplying them and filtering the result we obtain equation (3-7), which can be simplified to equation (3-8) if the phase difference between the signals is small (the loop is locked), proving the relation stated before. This simplifications of $u_d(t)$ and $u_f(t)$ will prove to be useful when deriving a frequency model for the PLL.

3.3 Frequency Domain Analysis

Before initiating the frequency domain analysis it would be useful to express the PLL signals in a more compact form. Reconsidering the time domain analysis, we have the signals $u_i(t)$, $u_o(t)$, $u_d(t)$,

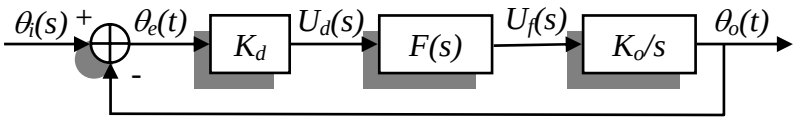

Figure 3-2. Block Diagram of Linear PLL in the Frequency Domain

and $u_f(t)$, representing the input, output, phase detector, and loop filter signal respectively. These signals can be expressed as

$$\theta_e = \theta_i - \theta_o \quad (3\text{-}10)$$

$$u_d(t) = u_i(t) \cdot u_o(t) \approx K_d \cdot \theta_e \text{, (Not including high frequency terms.)} \quad (3\text{-}11),$$

$$u_f(t) = u_d(t) * f(t) \text{, where * denotes convolution.} \quad (3\text{-}12),$$

$$u_f(t) \approx K_d \cdot \theta_e * f(t) \quad (3\text{-}13).$$

Note the relation of the above equations with the phase error, θ_e. We will not have the set of equations complete, until we express $u_o(t)$ in terms of a phase variable. (The idea of these derivations is to establish a relation within every signal of the PLL with a phase variable, given the predisposition of the PLL to keep a phase difference between two signals, input and output, close to zero. PLLs translate every phase or frequency variation to a phase difference, which explains the meaning of its name.)

The frequency of the VCO signal is given by equation (3-5) as $\omega_o = \omega_{0c} + K_o u_f(t)$. For the purpose of this analysis we are assuming the PLL is locked,

therefore the center frequency of the VCO signal equals that of the input signal, that is, $\omega_{0c} = \omega_i$, and the output frequency can be rewritten as $\omega_o = \omega_i + K_o u_f(t)$. By definition, the phase of the VCO is given by the integral of the frequency variation (Best 1999 [1]). Making use of this definition yields:

$$\theta_o = K_o \int u_f(t) dt \qquad (3\text{-}14).$$

Now that we have all the set of equations determined in terms of phase variables, we are able to proceed with the frequency domain analysis. Before we start, it is necessary to make one more assumption. Assume the phase of the input and output signals is changing with time, hence they have to be expressed as $\theta_i(t)$ and $\theta_o(t)$. Now we are able to use the Laplace transform to derive the frequency counterpart of each loop equation.

$$\theta_i(s) = L\{\theta_i(t)\} \qquad (3.15),$$

$$\theta_o(s) = L\{\theta_o(t)\} = L\left\{K_o \int u_f(t) dt\right\} = \frac{K_o}{s} U_f(s) \qquad (3.16),$$

$$V_{co} = \frac{\theta_o(s)}{U_f(s)} = \frac{K_o}{s} \qquad (3.17),$$

$$\theta_e(s) = \theta_i(s) - \theta_o(s) \qquad (3.18),$$

$$U_d(s) = L\{u_d(t)\} = K_d \theta_e(s) \qquad (3.19),$$

$$F(s) = L\{f(t)\} \qquad (3.20),$$

$$U_f(s) = U_d(s) \cdot F(s) \qquad (3.21),$$

where $L\{\ \}$ is the Laplace operator and $V_{co}(s)$ the transfer function of the VCO. It is important to point out that equation (3.16) was obtained assuming that all initial conditions of $\theta_o(t)$ are zero (Best 1999 [1]). Having equations (3-16) to (3-21) we can develop the linear model of the PLL, which will end up with the derivation of its transfer

function (Dorf et al 1995 [10]). The block diagram for this linear model is presented in figure 3-2. Let us now derive the transfer function of the system (PLL).

From equation (3.16) $\theta_o(s) = \frac{K_o}{s} U_f(s)$. Using equation (3.21), $\theta_o(s)$ can be written as $\theta_o(s) = \frac{K_o}{s} \cdot U_d(s) \cdot F(s)$. Substituting the definition of $U_d(s)$ into the previous equation we obtain $\theta_o(s) = \frac{K_o K_d}{s} \cdot \theta_e \cdot F(s)$. Then we use equation (3.18) to find $\theta_o(s) = \frac{K_o K_d}{s} \cdot \left(\theta_i(s) - \theta_o(s)\right) \cdot F(s)$. Solving for $\theta_o(s)$ yields

$$\theta_o(s)\left[1 + K_d K_o \cdot \frac{F(s)}{s}\right] = K_d K_o \cdot \frac{F(s)}{s} \cdot \theta_i(s).$$

Finally, expressing the former equation in the form of $\frac{\theta_o(s)}{\theta_i(s)} = H(s)$ gives us the closed-loop transfer function of the PLL:

$$H(s) = \frac{\theta_o(s)}{\theta_i(s)} = \frac{K_d K_o F(s)}{s + K_d K_o F(s)} \qquad (3.22).$$

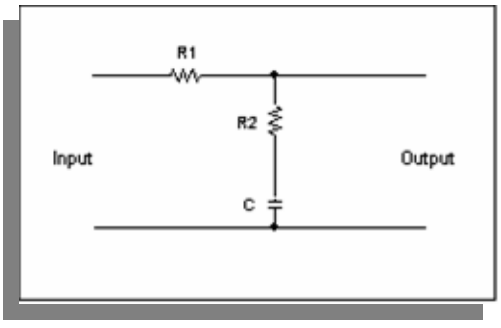

Figure 3-3. Circuit of Passive Lag-Lead Filter

Equation (3.22) is the starting point of the frequency domain analysis that leads to the understanding of the system response to various excitation signals. However, before getting deep into this analysis it would be pertinent to consider the type of loop filter appropriate for the PLL.

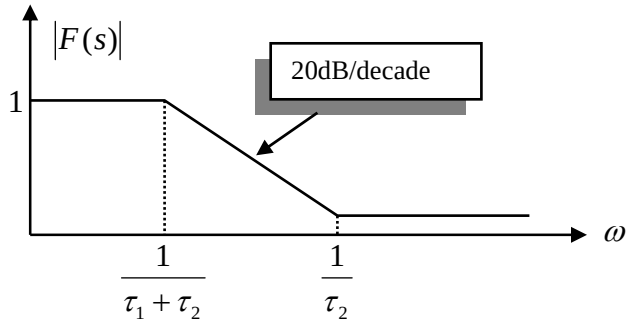

Figure 3-4. Bode Diagram of Passive Lag-Lead Filter

3.3.1 The Loop Filter

The loop filter is the component of the PLL that eliminates unwanted signals (high-frequency signals). In figure 3-2, the loop filter is designated in the frequency domain by $F(s)$, its transfer function. The selection of the loop filter is a very important decision, since it will determine the behavior and characteristics of the PLL under various operating conditions. The type of loop filter employed will determine characteristics such as bandwidth, lock-in range, pull-in range, pull-out range and hold-in range of the PLL during operation.

In control theory, it is the number of poles in the transfer function of a closed-loop system (when the system does not have a feedback path the open-loop transfer function is

used instead) that determines the order of the system. This concept is well suited to designate the order of any system for which its transfer function can be expressed as a rational function.

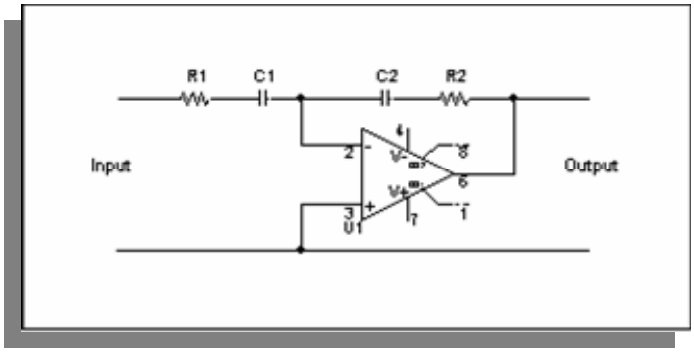

Figure 3-5. Circuit of the Active Lag-Lead Filter

The poles of a system can be defined as the zeros or roots of the denominator in the system transfer function. Since the design presented in this work is a second-order system, we can use this equation as an example to explain the order concept and root calculation: $H(s) = \dfrac{2\varsigma\omega_n s + \omega_n^2}{s^2 + 2\varsigma\omega_n s + \omega_n^2}$. To determine the poles of this system, we should take a look at its denominator $s^2 + 2\varsigma\omega_n s + \omega_n^2$, which is a quadratic equation on s. Now we can determine the values of s that make the denominator of $H(s)$ to become zero. For a quadratic equation, only two values of s can make it happen. They can be obtained using the quadratic formula, $s_1, s_2 = \dfrac{-2\varsigma\omega_n \pm \sqrt{(2\varsigma\omega_n)^2 - 4\omega_n^2}}{2}$.

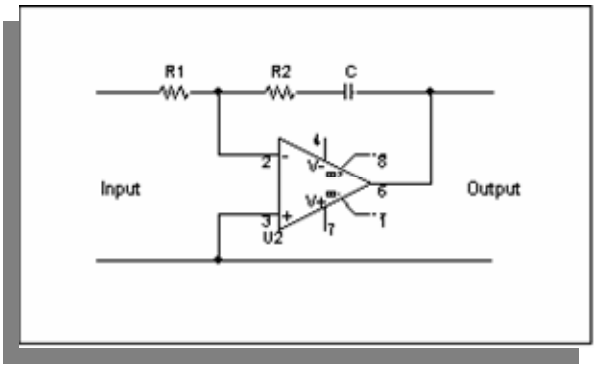

Figure 3-6. Circuit of the PI lag-Lead Filter

Simplifying the equation we obtain the two zeros of $s^2 + 2\varsigma\omega_n s + \omega_n^2$, which are $s_1 = -\varsigma\omega_n + \omega_n\sqrt{\varsigma^2 - 1}$ and $s_2 = -\varsigma\omega_n - \omega_n\sqrt{\varsigma^2 - 1}$. As mentioned previously, these two values of s make the denominator of $H(s)$ to become zero, which therefore make $H(s)$ to become infinite ($\frac{x}{0} \to \infty$, where x can be any number, except zero). When a value of s makes $H(s)$ to become infinite, we call that value a pole of $H(s)$. Now, given this system has only two poles, it is called a second-order system. On the other hand, if $H(s)$ had three poles it would be called a third-order system, and so on.

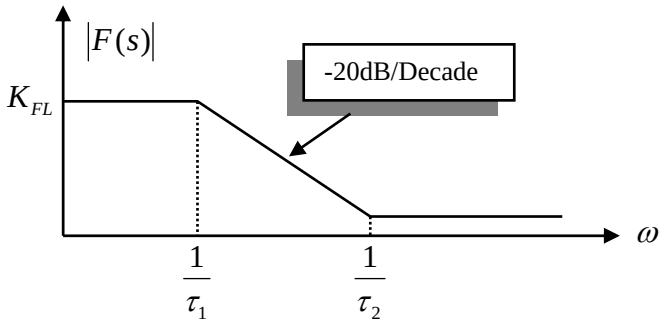

Figure 3-7. Bode Diagram of the Active Lag-Lead Filter

There are PLLs implemented as first-, second-, and third-order systems. However, it is the second-order PLL that is mostly implemented because of its stability, outstanding response, and ease of analysis (although it can be cumbersome too), (Best 1999 [1]). Third- and higher-order loops can be implemented, however, they confront stability problems and their analysis is cumbersome, which makes third-order loops the highest-order PLLs implemented. Yet, its implementation can be achieved using second-order approximations. Taking a look at equation (3.22), which gives the general form of the closed-loop transfer function of a linear PLL, we can see that the lowest-order loop that can be implemented is first-order. To obtain this type of PLL we have to substitute the loop filter by a gain, which we will refer to as K_{LF}. The resulting transfer function is

$$H(s) = \frac{K_d K_o K_{LF}}{s + K_d K_o K_{LF}} = \frac{K}{s + K} \qquad (3.23).$$

Since second-order system is the main concern of this thesis report, we will cover only those loop filters that are commonly used for this type of implementation. (First-order loops were discarded for this work because they do not provide flexibility to adjust system gain and bandwidth separately as second-order systems do)

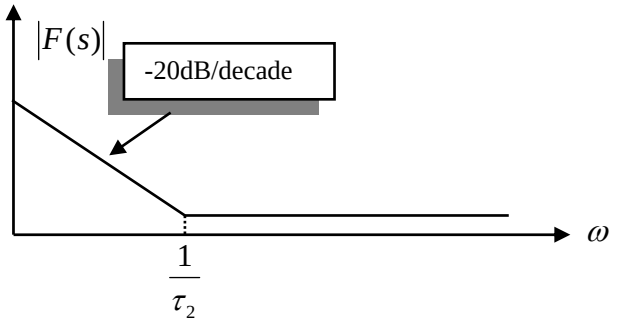

Figure 3-8. Bode Diagram of the PI Lag-Lead Filter

Using equation (3.22) again, it can be seen that the loop filter needed to implement a second-order PLL must add a pole to $H(s)$. Before deciding the type of loop filter, remember that it has to be low-pass. Therefore, combining both conditions it turns out that we need a first-order low-pass filter in order to obtain a second-order PLL. There are three types of filters commonly presented in PLL theory: passive, active, and PI (Proportional + integral) lag-lead filters (also known as lead-lag or simply lag filters), (Best 1999 [1]). The schematic of the passive lag-lead filter is shown in figure 3-3. The input and output to this filter are $u_d(t)$ and $u_f(t)$ respectively; the same holds for the other filter diagrams. The transfer function of this filter is given as

$$F(s) = \frac{1+\tau_2 s}{1+(\tau_1+\tau_2)s} \qquad (3.24),$$

where $\tau_1 = R_1 C$, and $\tau_2 = R_2 C$, (Best 1999 [1]). Figure 3-4 shows the Bode diagram of its frequency response. Once integrated circuit technology became accessible, amplification devices such as the OPAM (Operational Amplifier) were easily available. This made possible the implementation of the high-gain loop filters shown in figures 3-5 and 3-6, respectively.

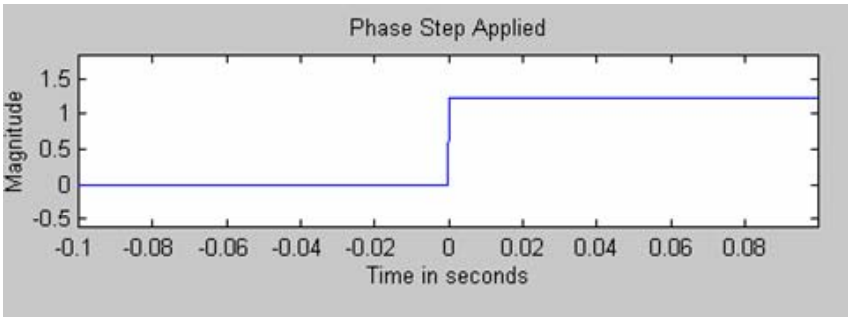

Figure 3-9. Phase Step Applied to Input Signal

These are the active and PI lag-lead filters whose frequency responses are shown in the Bode diagrams of figures 3-7 and 3-8. The transfer function of each filter is given in equations (3-25) and (3-26)

$$F(s) = K_{LF}\frac{1+\tau_2 s}{1+\tau_1 s} \quad (3\text{-}25),$$

$$F(s) = \frac{\tau_2 s + 1}{\tau_1 s} \quad (3\text{-}26).$$

For the active loop filter $\tau_1 = R_1 C_1$, $\tau_2 = R_2 C_2$, and $K_{LF} = -\frac{C_1}{C_2}$, whereas $\tau_1 = R_1 C$ and $\tau_2 = R_2 C$ for the PI filter. Due to the integrator, $\frac{1}{s}$ term in equation (3-26), the PI filter is the most effective loop filter that could be used in a PLL.

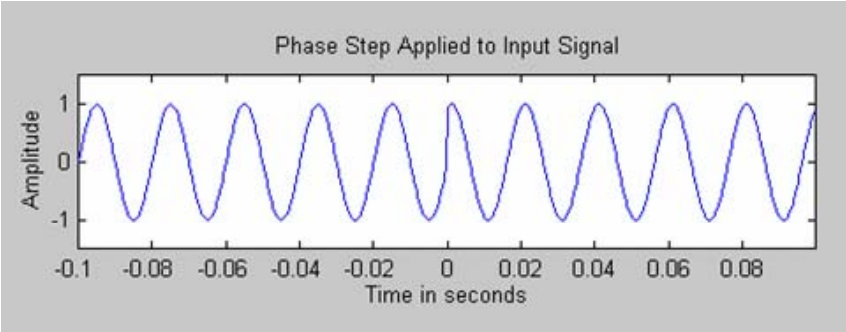

Figure 3-10. Input Signal Having Phase-Step Variations

Some of the characteristics that make this the obvious selection will be explained later, however it is important to mention at least one. Remember from previous explanations that the loop filter is used to eliminate the high frequency component of $u_d(t)$, retaining only the low frequency component. This low frequency component tends to a DC value when the PLL is locked. Now let us take a look at the DC gain of the PI filter, which is

obtained when $s = 0$, $F(0) = \frac{\tau_2(0)+1}{\tau_1(0)} = \frac{1}{0} \to \infty$. It is obvious from this result that the PI filter will emphasize the DC over any other signal, forcing the establishment of synchronization. More about this subject will be covered in subsequent sections. It is important to point out that the transfer function given in equation (3.26) is an approximation of $F(s) = \frac{-A(sCR_2+1)}{sCR_2+1+(1+A)sCR_1}$, which is the real transfer function of the PI filter. The variable A in this equation is the gain of the operational amplifier in the circuit. Because this gain is considered to be high, the expression is simplified to that given in equation (3.26).

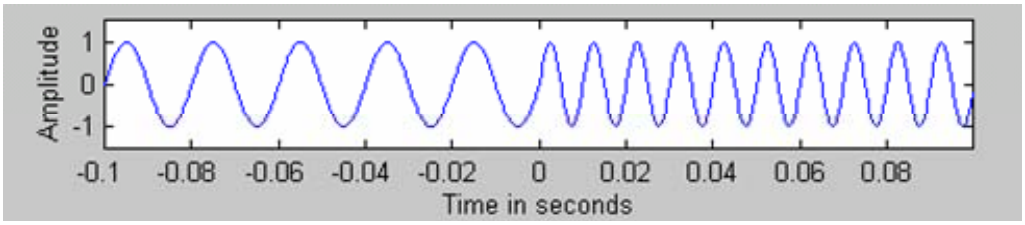

Figure 3-11. Input Signal Having Frequency Step Variations

The important consequence of this result is that the PI filter cannot be implemented in the analog domain, but approximated. However, even when this is true, this statement does not hold for a digital implementation of the PI filter, which results in a filter having the exact infinite DC gain desired and indicated before (this subject will be covered in subsequent sections), (Gardner 1979 [2]). Now that we have some loop filters defined we can proceed with the analysis.

3.3.2 Steady-State Error in PLLs Due to Common Excitation Signals

The response of the Phase-locked loop will be examined using the most common conditions encountered by communication systems: phase step, frequency step, and frequency ramp. These three excitation signals can be seen as an example of phase modulation, frequency modulation, and a signal whose frequency is changing linearly with time.

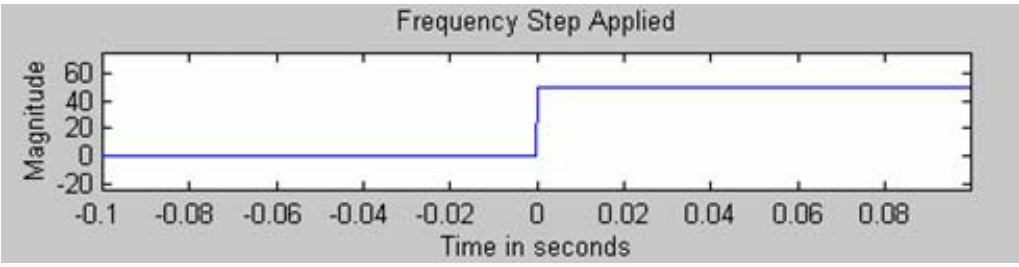

Figure 3-12. Frequency Step Applied to Input Signal

A phase step can also model an abrupt change in the phase of the input signal, whereas the frequency step can model a Doppler shift in the frequency of the incoming signal due to relative motion between transmitter and receiver. A frequency ramp, on the other hand, can model a change in the Doppler rate, due to acceleration in the motion. These excitation signals will be used to determine the steady-state error of the system, which results from the response of the system to these signals. Figures 3-9, 3-11, and 3-14 show a sine wave subject to each variation mentioned.

The steady-state error of a system is defined as the error when the time period is large and the transient response has decayed, leaving only the continuous response. To determine the steady-state error of the PLL due to any of the previously mentioned

signals, let us first obtain an expression for the error of the closed-loop system, also called the error transfer function, $\theta_e(s)$.

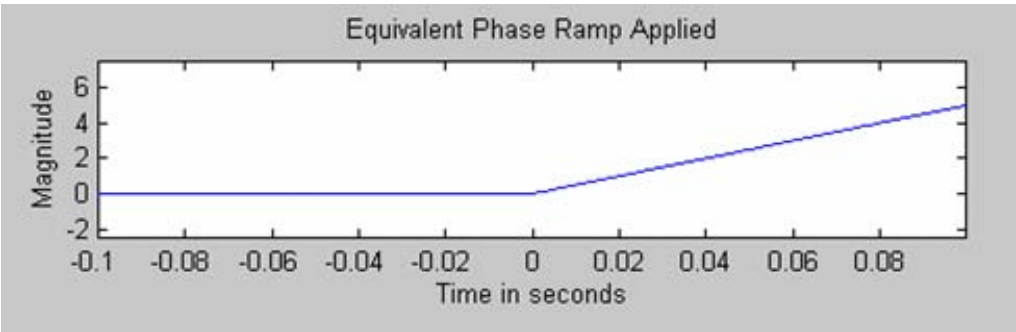

Figure 3-13. Equivalent Phase Ramp Applied to Input Signal

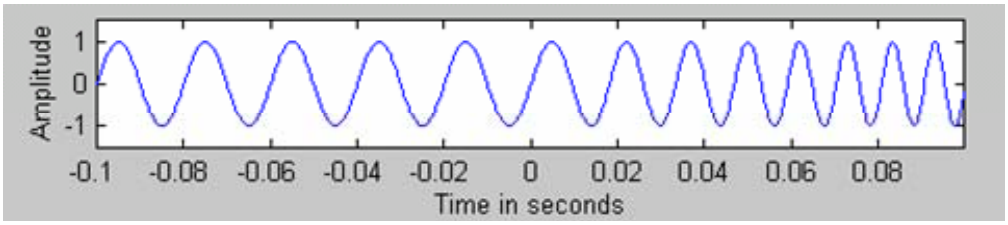

Figure 3-14. Input Signal Having Frequency Ramp Variations

From equation (3-18), $\theta_e(s) = \theta_i(s) - \theta_o(s)$. Equation (3-22) established that $H(s) = \dfrac{\theta_o(s)}{\theta_i(s)} = \dfrac{K_d K_o F(s)}{s + K_d K_o F(s)}$. Rewriting equation (3-22) in terms of $\theta_o(s)$ we obtain $\theta_o(s) = \theta_i(s)H(s) = \dfrac{K_d K_o F(s)}{s + K_d K_o F(s)}\theta_i(s)$. Substituting this result into (3.18) we obtain

$$\theta_e(s) = \left[1 - H(s)\right] \cdot \theta_i(s) = \frac{s}{s + K_d K_o F(s)} \qquad (3\text{-}27).$$

Making use of the definition of steady-state error, it would be necessary to measure the error between the actual and desired response of the system for a large period of time (until transients have died out) to obtain an idea of the system's performance. To understand the meaning of the term actual and desired response of a system, consider the block diagram of the PLL given in figure 3-2. This diagram is a linearized model of the PLL in the frequency domain. From figure 3-2, the PLL has an input signal, having a phase value designated by $\theta_i(s)$, and an output signal, having a phase value designated by $\theta_o(s)$. $\theta_i(s)$, the phase of the input signal, represents the desired response of the system.

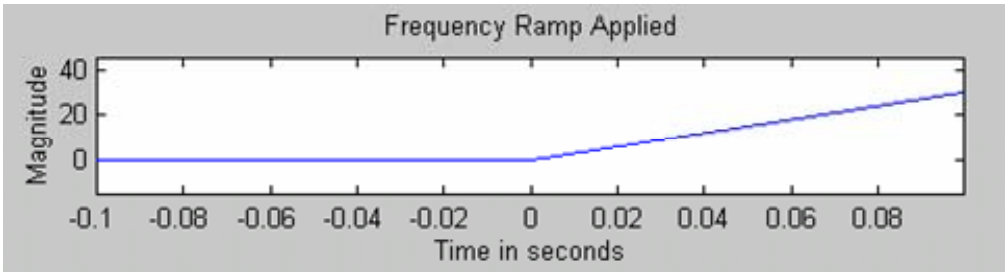

Figure 3-15. Frequency Ramp Applied to Input Signal

It is the phase that the output signal must have to allow correct demodulation of the input signal. $\theta_o(s)$, the phase of the output signal, represents the actual response of the system. It is the phase that the output signal possesses while the system is trying to match (which is really approximating via estimation) the phase of the input signal.

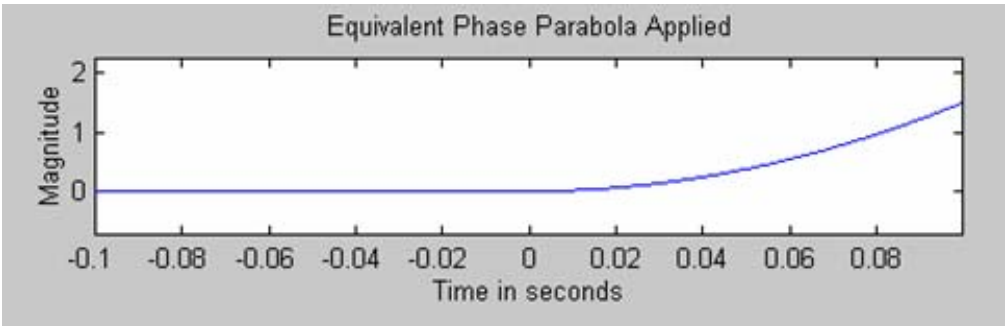

Figure 3-16. Equivalent Phase Parabola Applied to Input Signal

If $\theta_o(s)$ equals $\theta_i(s)$, and remains like that for a long period of time, then the actual response of the system has matched the desired response (real systems can only approximate it closely), which is the phase of the input signal.

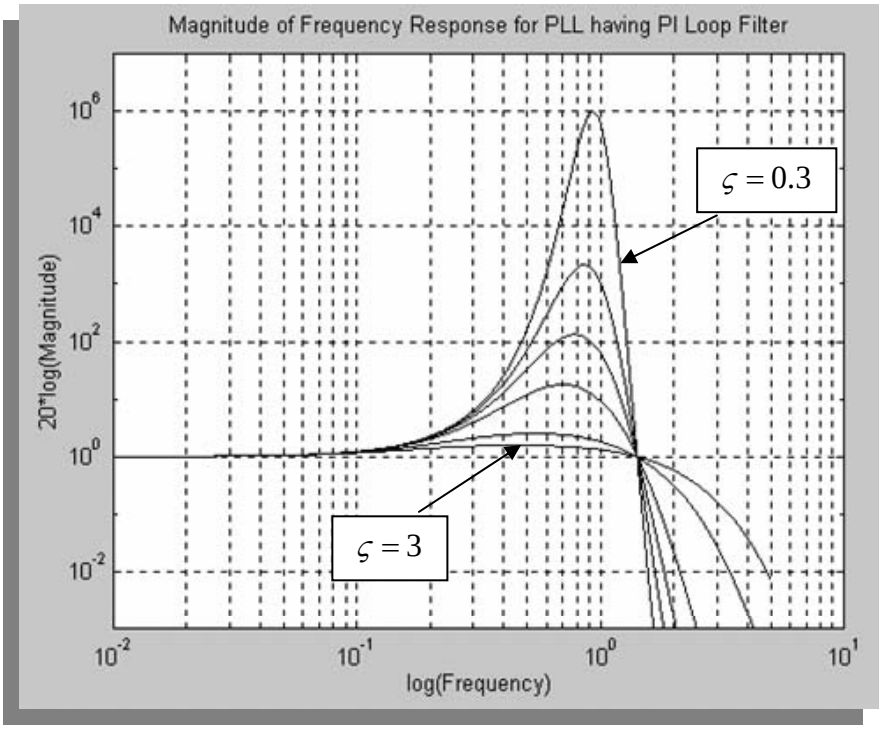

Figure 3-17. Frequency Response of PLL For Different Values of Damping Ratio

Remember that we are dealing with carrier synchronization, a part of synchronous communications, which requires that the receiving system have phase and frequency information of the carrier of the input signal, because without this information, the system will not be able to demodulate the received signal correctly. Mathematically, we can define the steady-state error of the system as

$$e_{ss} = \lim_{t \to \infty} \theta_e(t) \qquad (3.28),$$

where e_{ss} is the steady-state error of the system, t is a time variable, and $\theta_e(t)$ is the inverse Laplace transform of $\theta_e(s)$, (Dorf et al 1995 [10]). Although equation (3.28) can be correctly employed to calculate the steady-state error of the system, it requires the calculation of the inverse Laplace transform of the error transfer function $\theta_e(s)$.

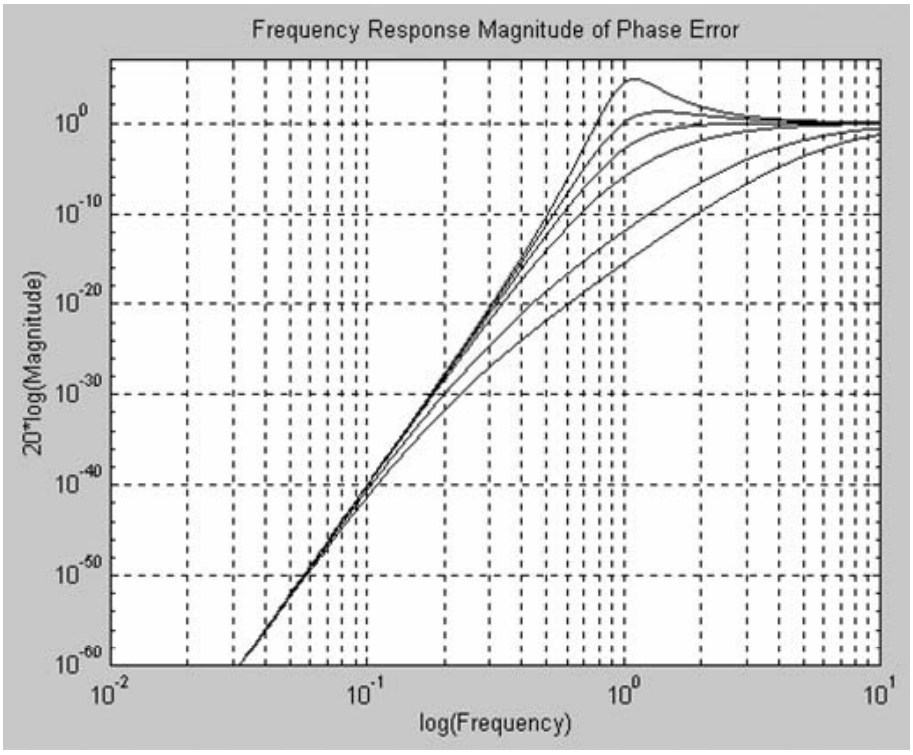

Figure 3-18. Frequency Response of the Phase-Error Transfer Function for Different Values of Damping Ratio

Given that most PLL requirements, if not all, are given in the frequency domain, and since we already have its error transfer function (for which it can be difficult to obtain its inverse transform), it would be desirable to determine the steady-state error right from the frequency domain. This can be done utilizing the final value theorem

$$e_{ss} = \lim_{t\to\infty} \theta_e(t) = \lim_{s\to 0} s \cdot \theta_e(s) \tag{3.29}.$$

Using equation (3.29) we are able to determine an equation that can be applied for PLLs using any type of loop filter. Substituting equation (3.27) into (3.29) yields equation (3.30).

$$e_{ss} = \lim_{s\to 0} s \cdot \theta_e(s) = \lim_{s\to 0} \frac{s^2}{s + K_d K_o F(s)} \cdot \theta_i(s) \tag{3.30}.$$

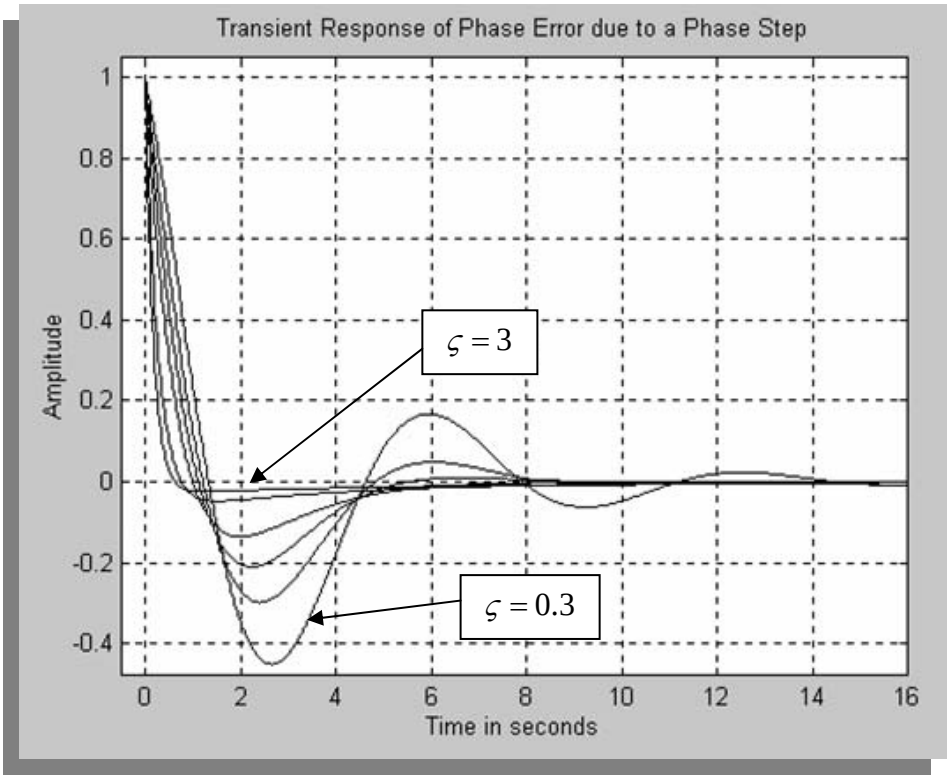

Figure 3-19. Transient Response of the Phase Error Due to a Phase Step

Before substituting each transfer function of the loop filters provided in section 3.3.1, the steady-state error analysis will be provided using a general representation, valid for any type of loop filter, (Best 1999 [1]), $F(s) = \frac{P(s)}{Q(s)s^N}$, where $P(s)$ and $Q(s)$ can be any polynomial in s, and N is the number of poles at $s = 0$. Substituting this expression for $F(s)$ into equation (3.30) we obtain

$$e_{ss} = \lim_{s \to 0} \frac{s^2 s^N Q(s)}{s \cdot s^N Q(s) + K_d K_o P(s)} \cdot \theta_i(s) \qquad (3.31).$$

The steady-state error can now be calculated as follows.

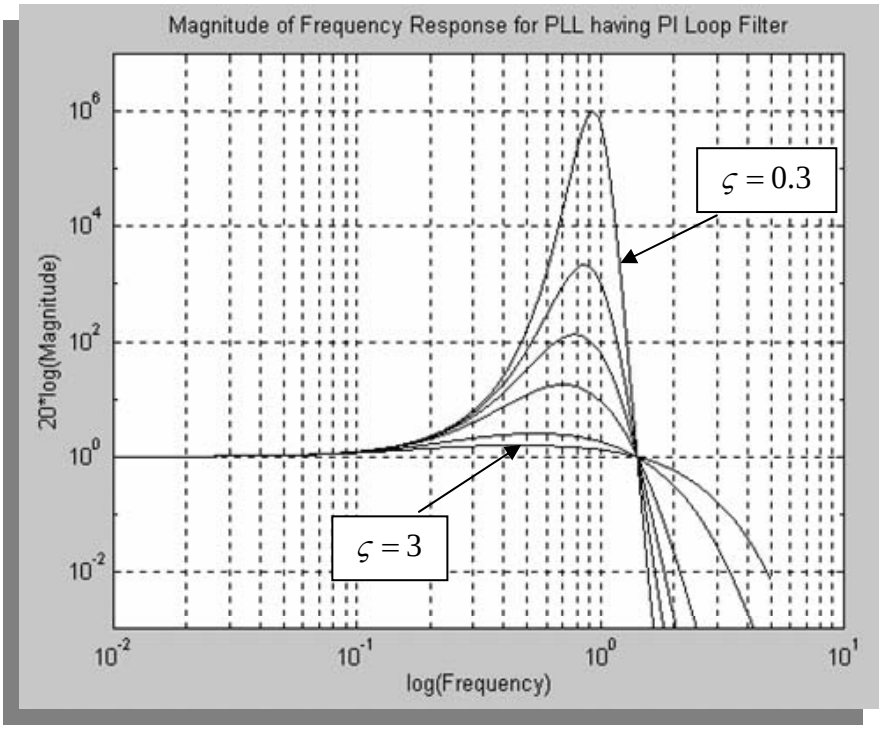

Figure 3-20. Transient Response of the Phase Error Due to a Phase Ramp

First consider a phase step applied to the input signal as shown in figure 3-10. The phase step is defined as $\theta_i(t) = \Delta\Phi \cdot u(t)$, where $\theta_i(t)$ is the phase of the input signal

in time, $\Delta\Phi$ is the magnitude of the phase step, and $u(t)$ is the unit step function. Do not confuse $u(t)$ with the signals $u_i(t)$ and $u_o(t)$, which are the input and output signals of the PLL. Remember that $\theta_i(t)$ is part of the expression of $u_i(t)$ presented in equation (3-1). The Laplace transform of $\theta_i(t)$ is $\theta_i(s) = \frac{\Delta\Phi}{s}$. Substituting this equation into equation (3.31) yields $e_{ss} = \lim_{s\to 0} \frac{s^2 s^N Q(s)}{s \cdot s^N Q(s) + K_d K_o P(s)} \cdot \frac{\Delta\Phi}{s}$, which results in a steady-state error of zero, after solving the expression ($e_{ss} = 0$).

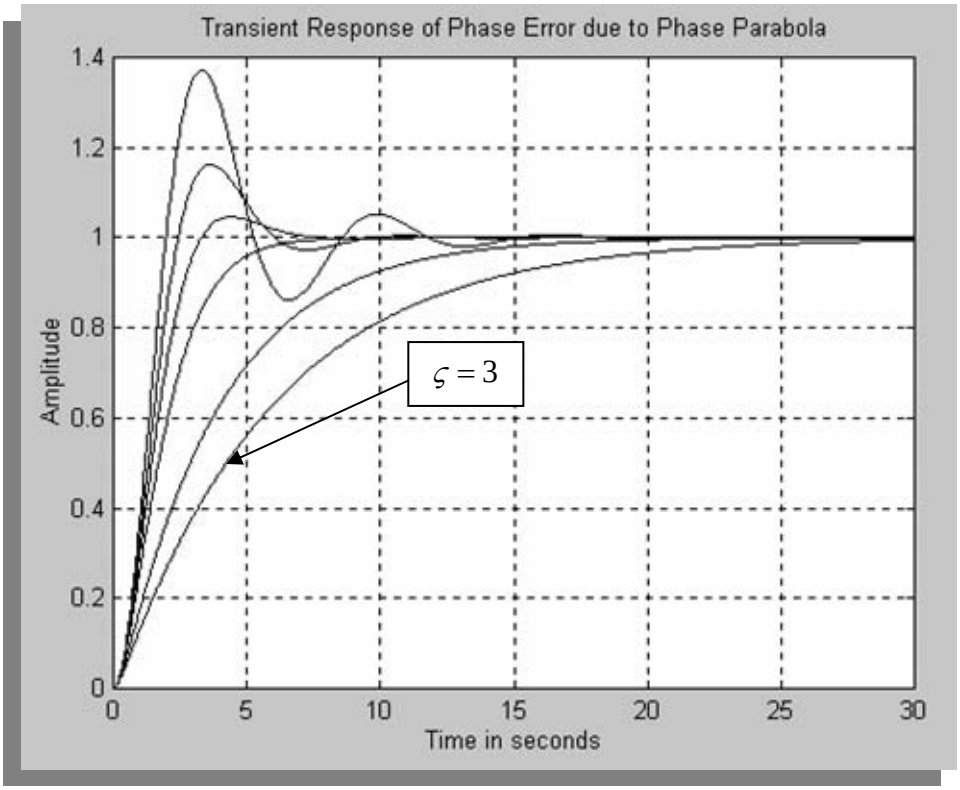

Figure 3-21. Transient Response of the Phase Error Due to a Phase Parabola

This result is independent of $\Delta\Phi$, and is true for any value of N, including zero, hence, valid for any type of loop filter. This means that no matter the amplitude of the phase

step and the type of loop filter employed, PLLs can track down and phase-lock to the signal until the steady-state error is settled to zero.

Consider now a frequency step applied to the input signal as shown in figure 3-12. For this case, the angular frequency of the signal becomes $\omega_i(t) = \omega_{io} + \Delta\omega \cdot u(t)$, where ω_{io} is the center frequency of the input signal, and $\Delta\omega$ is the magnitude of the frequency step. In order to use equation (3.31) to analyze the frequency step, we have to express it in the form of a phase variation.

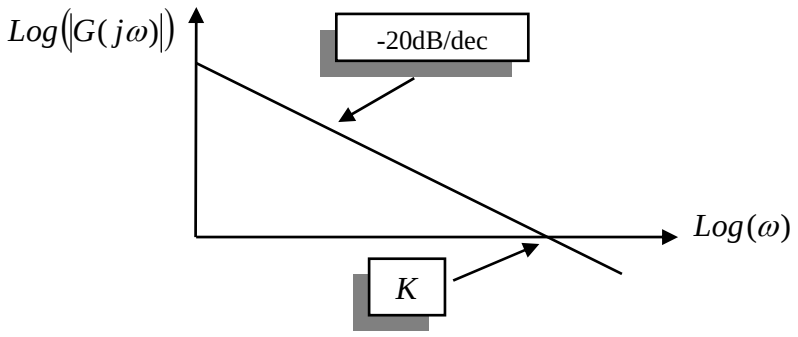

Figure 3-22. Bode Plot of the Open Loop Transfer Function of the First-Order PLL

By definition, $\theta_i(t)$ is the integral over the frequency variation $\Delta\omega$, thus, $\theta_i(t) = \Delta\omega\, t$, which turns out to be a phase ramp. Figure 3-13 shows the equivalent phase ramp employed. Applying the Laplace transform we obtain $\theta_i(s) = \frac{\Delta\omega}{s^2}$. Now, substituting this result into equation (3.31) yields $e_{ss} = \lim_{s\to 0} \frac{s^2 s^N Q(s)}{s \cdot s^N Q(s) + K_d K_o P(s)} \cdot \frac{\Delta\omega}{s^2}$. Simplifying this expression we obtain $e_{ss} = \lim_{s\to 0} \frac{s^N Q(s)}{s \cdot s^N Q(s) + K_d K_o P(s)} \cdot \Delta\omega$. This expression can only

become zero if N is greater than or equal to one. This means that loop filters of first order or higher, with an integrator at $s = 0$ are required to settle the steady-state error to zero. Loop filters of first order or higher result in PLLs of second order or higher. If a first order PLL is used, (the loop filter is substituted by an amplifier of gain K_{LF}), and the steady-state error would be $e_{ss} = \dfrac{\Delta\omega Q(0)}{K_d K_o P(0)}$.

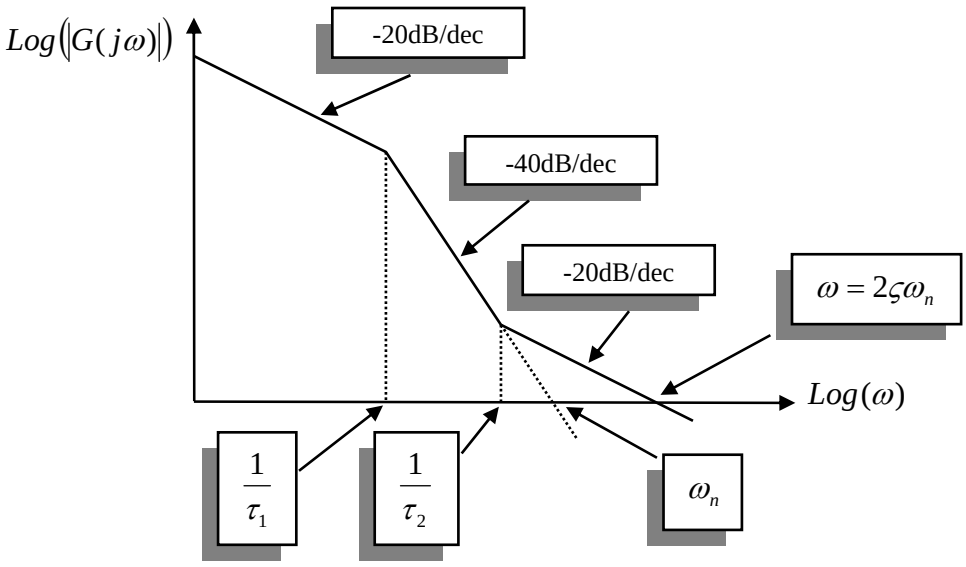

Figure 3-23. Bode Plot of Open Loop Transfer Function of the Second-Order PLL

Since $Q(s) = 1$ and $P(s) = K_{LF}$, for this case, $F(s) = K_{LF}$. Hence, e_{ss} can be written as

$$e_{ss} = \frac{\Delta\omega}{K_d K_o K_{LF}} = \frac{\Delta\omega}{K} \qquad (3.32),$$

where $K = K_d K_o K_{LF}$; this term is known as the gain of the PLL. In control theory, this constant is referred to as the velocity error constant, K_v, (Dorf et al 1995 [10]).

The final analysis considers a frequency ramp applied to the input signal, $u_i(t)$. Refer to figures 3-14 and 3-15 for graphical examples. For a frequency ramp, the angular frequency of the signal is expressed as $\omega_i(t) = \omega_{io} + \Delta\dot{\omega}t$, where $\Delta\dot{\omega}$ is the rate of change of the frequency, and t represents time. Using the process of the previous analysis it turns out that $\theta_i(t) = \Delta\dot{\omega}\frac{t^2}{2}$, hence $\theta_i(s) = \frac{\Delta\dot{\omega}}{s^3}$.

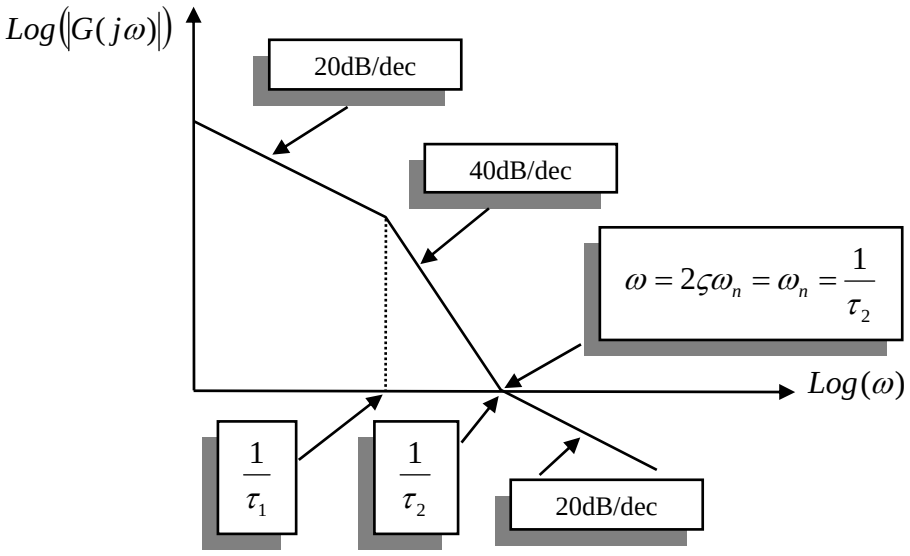

Figure 3-24. Bode Plot for Open Loop Transfer Function for Second-Order PLL when $\varsigma = 0.5$

Figure 3-16 shows a graph of the phase parabola derived. Finally, substituting this result into equation (3.31) yields $e_{ss} = \lim_{s\to 0} \frac{s^N Q(s)}{s^2 \cdot s^N Q(s) + sK_d K_o P(s)} \cdot \Delta\dot{\omega}$. This expression can only be zero if N is larger than or equal to two. Therefore, the steady-state error will be zero if the PLL is third-order or higher. Revising previous analysis it turns out that a first-order PLL will never follow a parabolic phase change ($e_{ss} \to \infty$), whereas a second-

order PLL can track it down with a steady-state error of $e_{ss} = \frac{\Delta \dot{\omega} Q(0)}{K_d K_o P(0)}$. Having this general expression for the steady-state error of a second-order PLL we can substitute the transfer function of the loop filters presented in section 3.3.1 to determine their respective steady-state value. It arises that the steady-state error is infinite for both the active and passive loop filters. This is true because none of these filters have a pole at $s = 0$.

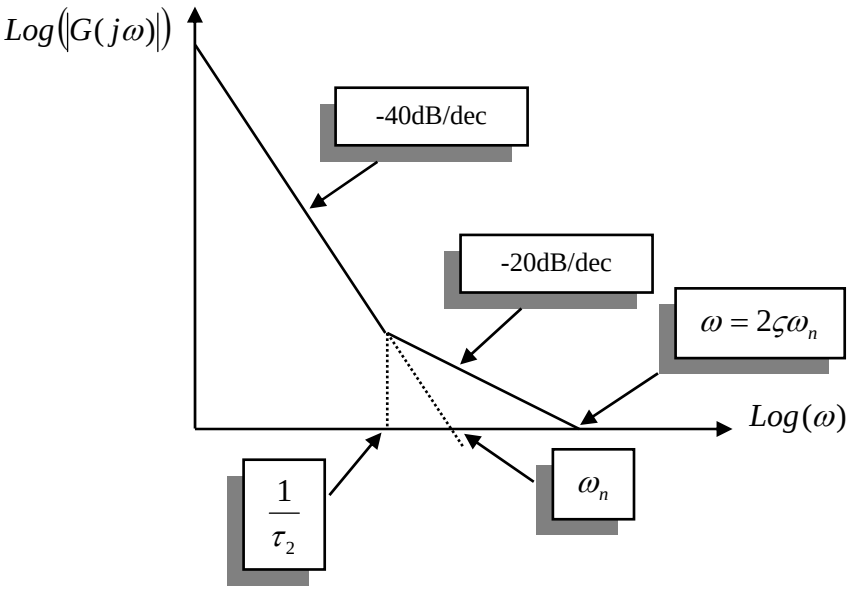

Figure 3-25. Bode Plot of Open Loop Transfer Function of PLL Having PI Loop Filter

Thus, for these two types of loop filters, the second-order PLL behaves as first-order, unless $\tau_1 >> 1$, and $\tau_1 >> \tau_2$, where the transfer function of the passive and active filters can be rewritten as $F(s) \approx \frac{\tau_2 s + 1}{\tau_1 s}$, and $F(s) \approx K_{LF} \frac{\tau_2 s + 1}{\tau_1 s}$, respectively. Consequently, these two loop filters will behave as the PI filter, which has a pole at $s = 0$. For the PI filter, the steady-state error is written as

$$e_{ss} = \frac{\Delta\dot{\omega}}{\frac{K_d K_o}{\tau_1}} = \frac{\Delta\dot{\omega}}{\frac{K}{\tau_1}} \qquad (3.33),$$

where $K = K_d K_o$. When the active loop filter is used, and assuming $\tau_1 >> 1$, $K = K_d K_o K_{LF}$.

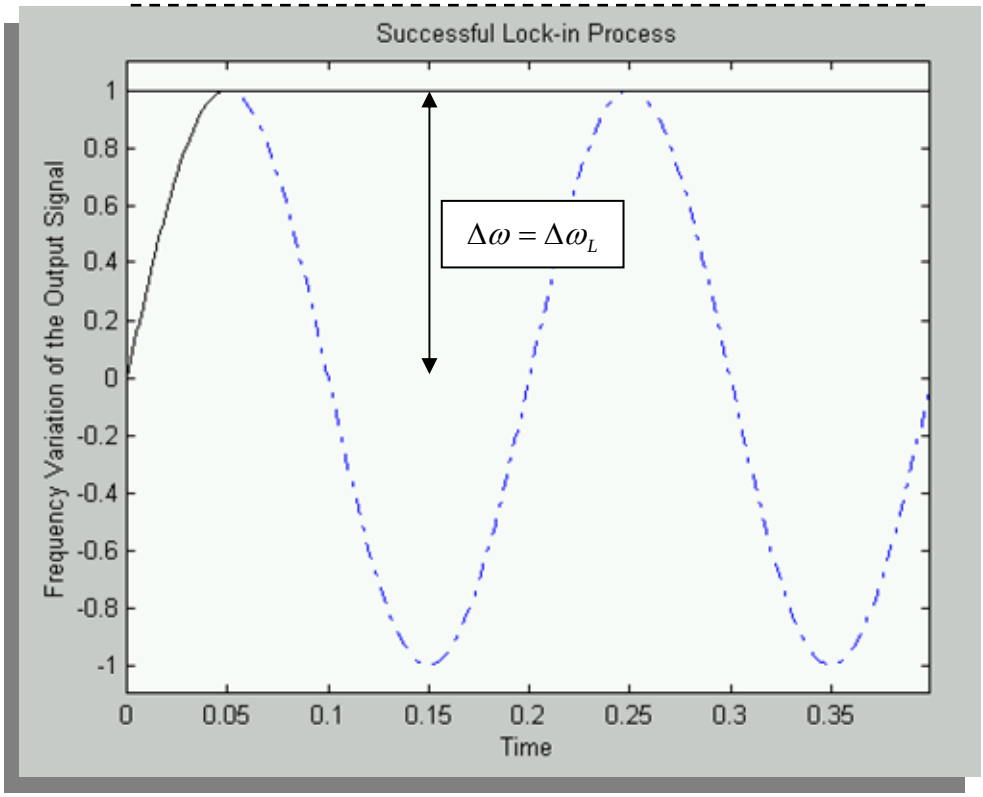

Figure 3-26. Lock-In Process: $\Delta\omega = \Delta\omega_L$

Equation (3.33) has not been simplified to $e_{ss} = \frac{\Delta \dot{\omega} \tau_1}{K}$ for reasons that will be obvious later in the next section. Because the steady-state error of the PLL could only be zero for the PI filter, or a filter behaving like the PI filter, the rest of the analysis performed will assume a PI filter is used as loop filter, which results in a second-order PLL.

3.3.3 Closed-Loop Transfer Function of the PLL

In the previous section we considered an important aspect of PLL analysis, the steady-state error resulting from the response of the system to an excitation signal having phase step, frequency step, and frequency ramp variations.

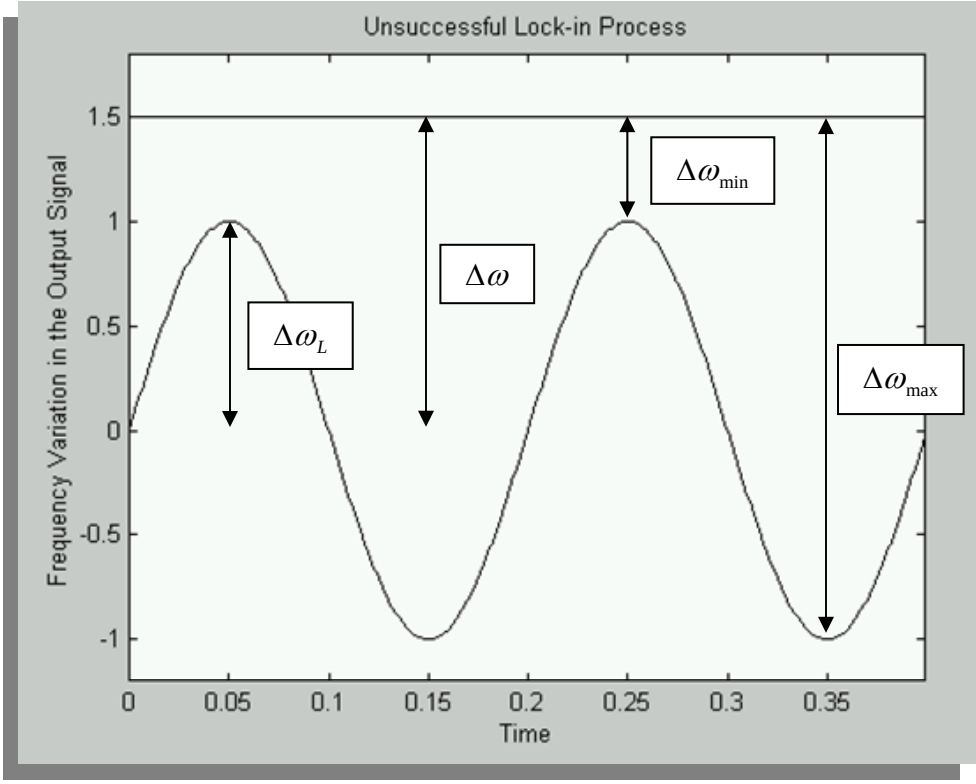

Figure 3-27. Unsuccessful Lock-In Process: $\Delta\omega > \Delta\omega_L$

We concluded that only a third-order PLL is able to track all this three type of signals with a steady-state error equal to zero. However, it is well known that the design and analysis of third-order PLLs is difficult to perform and, at the end, the design is accomplished using second-order loop approximations, (Best 1999 [1], Gardner 1979 [2], Egan 1998 [3], Wolaver 1991 [4], and Lindsey et al 1991 [8]). A second-order loop, tracks phase and frequency steps with zero steady-state error, but follows the frequency ramp with an error equivalent to equation (3.33), $e_{ss} = \frac{\Delta\dot{\omega}}{\frac{K}{\tau_1}}$. Given the importance of the PI loop filter and the significance of the analysis presented in next section, it would not be

appropriate to continue the discussion of PLLs without expressing the specific equations governing the behavior of the second-order PLL with PI loop filter.

From equation (3.22), the transfer function of a PLL is expressed as $H(s)=\frac{K_d K_o F(s)}{s+K_d K_o F(s)}$. The transfer function of the PI loop filter as given by equation (3.26) is $F(s)=\frac{\tau_2 s+1}{\tau_1 s}$. Placing this transfer function into equation (3.22) yields $H(s)=\frac{\frac{K_d K_o \tau_2}{\tau_1}s+\frac{K_d K_o}{\tau_1}}{s^2+\frac{K_d K_o \tau_2}{\tau_1}s+\frac{K_d K_o}{\tau_1}}$. In control theory it is customary to express a second-order system like this one in terms of its damping ratio ς, and natural frequency ω_n. This way we can apply to the PLL the vast amount of theory that has been developed to analyze this type of system. Making use of this concept we can write the previous equation as

$$H(s)=\frac{2\varsigma\omega_n s+\omega_n^2}{s^2+2\varsigma\omega_n s+\omega_n^2} \qquad (3.34),$$

where $2\varsigma\omega_n=\frac{K_d K_o \tau_2}{\tau_1}$, and $\omega_n^2=\frac{K_d K_o}{\tau_1}$. Equation (3.34) makes evident that the PLL is a second-order low-pass filter having DC gain equal to one, $H(0)=1$. The expression for the transfer function of the phase error is obtained by substituting $F(s)$ of the PI loop filter into equation (3.27), thus $\theta_e(s)=\frac{s^2}{s^2+\frac{K_d K_o \tau_2 s}{\tau_1}+\frac{K_d K_o}{\tau_1}}=\frac{s^2}{s^2+2\varsigma\omega_n s+\omega_n^2}$. Using

equation (3.30) we can determine an expression for the steady-state error, $e_{ss} = \lim_{s\to 0} \frac{s^2}{s^2 + 2\varsigma\omega_n + \omega_n^2} \cdot \theta_i(s)$, which is equal to

$$e_{ss} = \frac{\Delta\dot{\omega}}{\omega_n^2}. \qquad (3.35).$$

Comparing equation (3.35) to (3.33), where $e_{ss} = \frac{\Delta\dot{\omega}}{\frac{K_d K_o}{\tau_1}} = \frac{\Delta\dot{\omega}}{\frac{K}{\tau_1}}$, it becomes obvious the decision of not simplifying e_{ss} to $\frac{\Delta\dot{\omega}\,\tau_1}{K}$. Equation (3.35) relates the steady-state error to the natural frequency of the second-order PLL. This is a significant relation because parameters such as the 3-dB bandwidth, ω_{3-dB}, and the noise bandwidth, B_L, are directly related to the natural frequency. The 3-dB bandwidth for a second-order PLL having PI loop Filter is expressed as, (Best 1999 [1]):

$$\omega_{3-dB} = \omega_n\left[1 + 2\varsigma^2 + \sqrt{(1+2\varsigma^2)^2 + 1}\right]^{\frac{1}{2}} \qquad (3.36).$$

Equation (3.36) not only relates the bandwidth of the system to the natural frequency, but to the damping ratio as well. Assume for a moment that we select $\varsigma = 0.7$. Inserting this value into equation (3.36) results in a 3-dB bandwidth of about twice the natural frequency, $\omega_{3-dB} \approx 2.06\omega_n$. Thus, by establishing the values of the damping ratio and the natural frequency of the system we establish its 3-dB bandwidth. The relation between the natural frequency and the noise bandwidth will be presented, along with the noise theory, in other section. Now that we finally have the closed-loop transfer function of the

second-order PLL as well as its phase error transfer function, let us determine their respective frequency response.

The frequency response of the PLL is shown in figure 3-17 for different values of damping ratio. The damping ratio varies as 0.3, 0.5, 0.7, 1, 2, and 3. The frequency axis is given by $\frac{\omega}{\omega_n}$ (it has been normalized to the natural frequency of the system, ω_n), and both axis are in logarithmic scale. Figure 3-17 presents the PLL as a low-pass filter capable of tracking phase and frequency variations in the input signal that range from zero to about its natural frequency. Translating this result to communications, it means that the PLL is able to track a signal having phase or frequency modulation as long as the frequency or the frequencies of the signal lie within zero to about ω_n. (Figure 3-17 also shows that the value of the normalized frequency $\frac{\omega}{\omega_n}$ where all the graphs intersect has an approximate value of $\sqrt{2}$. This value was determined graphically using Matlab.)

The frequency response of the Phase-error transfer function is shown in figure 3-18. As with the frequency response of the PLL, the frequency axis has been normalized by ω_n, and the response is plotted for different values of damping ratio (0.3, 0.5, 0.7, 1, 2, and 3). The figure shows that the phase error response behaves like a high-pass filter. Consequently, the phase error during the tracking process of input signals having phase and frequency modulation smaller than ω_n remains considerably small. However, for larger phase and frequency variations the phase error can be as large as the variation itself

(phase and frequency variation presented by the input signal), meaning that the PLL is no longer able to maintain phase tracking of the input signal.

The effect of the damping ratio in the dynamic response of the PLL is indicated by the figures as well. When $\varsigma < 1$, the response of the system becomes oscillatory, and is referred to as an underdamped response. On the other hand, for $\varsigma = 1$, the system is critically damped, and oscillations are hardly seen in the response. For $\varsigma > 1$, the system is said to be overdamped and, as in the previous case, oscillations can be assumed to be zero. Think of the damping ratio as a measure of responsiveness of the system. When $\varsigma < 1$, the system responds fast to variations in the input signal. The smaller the damping ratio, the faster the response of the system, but the larger the oscillations that occur in the response. If oscillations in the response are big enough, it might be possible that the PLL would never track and lock to the signal. If, on the contrary, $\varsigma \geq 1$, the response of the system becomes sluggish and it might be possible that the PLL would never track and lock to the signal as well. The effect of the damping ratio is also observed in Figures 3-17 and 3-18. These figures show that the smaller the damping ratio, the larger the spike in the frequency response of both the phase error and the PLL.

Before going any further, let us talk revise figures 3-19 to 3-21. These are the transient response of the Phase error due to a phase step, phase ramp, and phase parabola, respectively. These figures have been plotted using the same damping ratio values used for the previous figures. It is evident from these figures that the smaller the damping ratio the larger the overshoot and/or the undershoot in the transient response of the

system's phase error. In addition, figures 3-19 and 3-20 show that for a damping ratio that is either too small or too large, it takes longer for the phase error response to settle to its final value, zero phase error. For the case of a phase parabola, as shown in figure 3-21, it takes long to the phase error response to settle to its final value, $\frac{\Delta\dot{\omega}}{\omega_n^2}$, if the damping ratio has the characteristics just mentioned. It is important to mention that the magnitude of the phase parabola applied to the system was normalized to ω_n^2, making it equal to one. For this reason, the transient of the phase error response settles to one. (All these figures were obtained using Matlab.)

3.3.4 Open Loop Analysis: Bode Plot

Bode plots are very useful to study PLLs because several loop parameters appear as distinctive points on the plots. For this type of analysis we need the open-loop transfer function of the system, which will be denoted $G(s)$. Since we previously presented various types of PLL, the Bode plot analysis will be developed to include at least first- and second-order PLLs. The procedure to obtain a Bode plot will not be covered here, however, the interested reader can refer to a book of basic circuit theory for information about the subject.

Let us start the analysis with the first-order PLL. From figure 3.2, the open loop transfer function of a first-order PLL is determined as $G(s) = \frac{K_d K_o K_{LF}}{s} = \frac{K}{s}$. It can be seen that the only frequency-selective term of this loop arises from the integrator of the

VCO, which results in a Bode plot consisting of a single straight line with slope –20dB/dec (minus twenty decibels per decade), as shown in figure 3-22. Before continuing any further, let us define the term gain crossover. The gain crossover is the frequency at which $|G(j\omega)| = 1$ or 0dB. Applying this definition to the first-order loop, it arises that its gain crossover lies at $\omega = K$. For a first-order loop, the loop gain K is the only parameter available for the designer to determine the loop characteristics. Remember that for this type of PLL, parameter K determined the magnitude of the steady state error for frequency step (phase ramp) at the input signal $\left(e_{ss} = \frac{\Delta\omega}{K} \right)$. Consequently, in order to reduce the steady state error, K has to be quite large, which results in a wider bandwidth (for this loop the 3-dB bandwidth is equal to K), which is crucial for the dynamic operation of the loop in the presence of noise. It is because of this characteristic that first-order PLLs are not commonly used.

For Second-order phase-locked loops the analysis will be mostly developed assuming a passive lag-lead filter, however, the analysis presented applies for PI and active loop filters as well. Stability will not be covered for second-order analog phase-locked loops, because they are always stable. The open loop transfer function of the passive loop filter is $G(s) = \frac{K_d K_o K_{LF}(\tau_2 s + 1)}{s(1 + \tau_1 s)}$. Figure 3.23 shows the Bode plot of this function. It consists of three straight lines with slope $-20dB/dec$, $-40dB/dec$, and $-20dB/dec$, respectively. The leftmost line of the plot results from the integrator of the VCO. This line connects to a second line at the point $\omega = \frac{1}{\tau_1}$, the location where lies the

pole of the loop filter. This point is known as a lag break. The second breaking point occurs at the place where lies the zero of the loop filter, $\omega = \frac{1}{\tau_2}$. This point is known as a lead break. The Bode plot of figure 3-23 can be used to determine ω_n, the natural frequency of the second-order PLL. Take a look at the straight line with slope –40dB/dec. If we extend this line until it intersects the frequency axis, where $|G(j\omega)| = 1$ or 0dB, ω_n will be the frequency value denoted by this intersection. Another important parameter we can obtain from the Bode plot is $2\varsigma\omega_n$. To obtain this parameter just consider the place where the third straight line (with slope –20dB/dec) intersects the frequency axis, $2\varsigma\omega_n$ will be represented by this frequency value. (If this line does not touch the frequency axis, then we have to extend the line until the extension touches the axis.) Consider again the location of ω_n and $2\varsigma\omega_n$ in the plot. Comparing these two parameters it turns out that $2\varsigma\omega_n = \omega_n$ if $\varsigma = 0.5$. This is equivalent to placing the lead breaking point where $|G(j\omega)| = 1$ or 0dB. When this happens, ω_n equals the gain crossover. A graphical representation of this result is given in figure 3-24. For a second-order loop, the gain crossover occurs at $\omega = 2\varsigma\omega_n$. If the damping ratio is selected below 0.5, the response of the PLL becomes highly oscillatory (Best 1999 [1]). This will result in a Bode plot where the line with slope –40dB/dec passes below the frequency axis, as seen in figure 3-24.

The analysis just presented for PLLs having an passive loop filter applies for PI and active filters as well. In fact, the Bode plot for the active loop filter will be exactly

like the one presented for the passive filter. This is not true for a PLL having PI loop filter. The theory is the same, but the Bode plot is different. The Bode plot of the open loop transfer function of this PLL is shown in figure 3-25. This Bode plot is composed of two straight lines having slope of $-40dB/dec$ and $-20dB/dec$, respectively. Since the PI loop filter has an integrator at $s=0$, the Bode plot starts with a slope of $-40dB/dec$. Remember that it is this pole that gives the PI filter the characteristics needed to make the PLL track a phase step and phase ramp with steady-state error of zero and a frequency ramp with steady-state equal to $\frac{\Delta\dot{\omega}}{\omega_n^2}$. Although not shown here, a high-gain third-order PLL would have a slope of $-60dB/dec$, resulting from its three poles located at $s=0$, characteristic that makes it possible for this loop to follow a phase parabola (frequency ramp) with zero steady-state error.

3.4 Parameters for Dynamic Performance of the PLL: Hold-In, Lock-In, Pull-In, and Pull-Out ranges

During operation, the PLL is affected by the conditions acting upon the incoming signal. Some of these conditions were discussed previously as phase and frequency variations. Three specific cases were discussed, phase step, phase ramp (frequency step), and phase parabola (frequency ramp). It was shown that the operation of the loop greatly depended upon the type of loop filter selected. In addition, it was mentioned that for some parameters, like the selection of the damping ratio, it was possible for the PLL not to track and lock to a signal. This happened because the response was either too slow (had no oscillations at all) or too fast (had large oscillations). It is well known that for

signals having the characteristics presented previously, the loop may never lock to the signal. Consequently, it is important to study the dynamic characteristics of the PLL that govern its dynamic response to these signals. These characteristics or parameters are known as lock-in, pull-in, pull-out, and hold-in ranges.

3.4.1 Lock-In Range

When a PLL is tracking the input signal, the phase and frequency parameters of its output signal are equal to those of the input (the phase and frequency difference between both signals is close to zero) hence, we say the PLL has locked to the signal. Before locking to the signal, the PLL enters a dynamic process where the parameters of the incoming signal are estimated. These parameters are used to generate an output signal intended to resemble the phase and frequency of the input. This output signal is compared to the input. If the estimates are not correct, an error signal is generated. This error signal is used to correct the estimated parameters. The process continues until the error signal is zero or settles to a specific value, at which point, lock-in has occurred. It is desired that the PLL be able to acquire and track the input signal as fast as possible. There is a frequency deviation $\Delta\omega$ for which the PLL is able to track the input signal in just a single beat note in the error signal. Remember from equation (3-4), $u_f(t) = \frac{A_i A_o}{2}\cos[(\omega_i - \omega_o)t + \theta_i - \varphi_o]$, that the filtered error signal has the form of a cosine wave when the loop is not locked, which causes the signal to oscillate. If the frequency deviation is inside the lock-in range then $u_f(t)$ will settle to zero after a cycle or less. This is what is called a single beat note.

It is desired that the PLL operates in the lock-in range. To calculate the lock-in range, first assume that the loop is not locked and the input signal has frequency deviation $\Delta\omega_L$, where $\Delta\omega_L$ is the lock-in range. Defining the center frequency as ω_{0c}, the frequency of the input signal can be expressed as $\omega_i = \omega_{0c} + \Delta\omega_L$. From equation (3.5) the frequency of the output signal is $\omega_o = \omega_{0c} + K_o u_f(t)$. Subtracting these two equations we obtain $\Delta\omega = \Delta\omega_L - K_o u_f(t)$, where $\Delta\omega$ the frequency difference between the signals. In order to make $\Delta\omega$ equal to zero, $\Delta\omega_L = K_o u_f(t)$. Using equation (3.4), and neglecting $(\theta_i - \varphi_o)$ for the moment, the output of the loop filter would be $u_f(t) = K_d \left|F(\Delta\omega_L)\right| \cos\left[(\Delta\omega_L)t\right]$, where $K_d = \frac{A_i A_o}{2}$. From this result, the peak frequency deviation that can be obtained is $K_d K_o \left|F(\Delta\omega_L)\right|$. Plugging this result into $\Delta\omega_L = K_o u_f(t)$ we obtain

$$\Delta\omega_L = K_d K_o \left|F(\Delta\omega_L)\right| \qquad (3.37).$$

Substituting $\left|F(\Delta\omega_L)\right|$ for each loop filters given in section 3.3.1 we obtain a non-linear expression for $\Delta\omega_L$. Nevertheless, since it is customary that $\Delta\omega_L$ is larger than the loop filter parameters $\frac{1}{\tau_1}$ and $\frac{1}{\tau_2}$, and $\tau_1 >> \tau_2$ we can approximate $\left|F(\Delta\omega_L)\right|$ for each filter as follows

For the passive filter: $\left|F(\Delta\omega_L)\right| \approx \frac{\tau_2}{\tau_1 + \tau_2} \approx \frac{\tau_2}{\tau_1}$ (3.38),

For the active filter: $\left|F(\Delta\omega_L)\right| \approx K_{LF} \frac{\tau_2}{\tau_1}$ (3.39),

And for the PI Filter: $\left|F(\Delta\omega_L)\right| \approx \frac{\tau_2}{\tau_1}$ (3.40).

Using these approximations the lock-in range is given as

For the passive filter: $\Delta\omega_L \approx \frac{K_d K_o \tau_2}{\tau_1} = 2\varsigma\omega_n$ (3.38),

For the active filter: $\Delta\omega_L \approx K_{LF} \frac{K_d K_o \tau_2}{\tau_1} = 2\varsigma\omega_n K_{LF}$ (3.39),

And for the PI Filter: $\Delta\omega_L \approx \frac{K_d K_o \tau_2}{\tau_1} = 2\varsigma\omega_n$ (3.40).

Assume for the moment that we are using the PI loop filter and we want to calculate $\Delta\omega_L$ without using the approximations stated above. In this case, $\Delta\omega_L$ is expressed as

$$\Delta\omega_L^4 - (2\varsigma\omega_n)^2 \Delta\omega_L^2 - \omega_n^4 = 0 \quad (3.41).$$

Having this non-linear expression, all we have to do is to solve for the roots of $\Delta\omega_L$, which is easily done using Matlab or any graphics calculator such as the HP48GX. When solving equation (3.41) Matlab will give four possible results for $\Delta\omega_L$, as it should be. Select as $\Delta\omega_L$ the largest real and positive result given by Matlab.

Now that we have equations to determine the lock-in range of second-order PLLs, let us examine its meaning graphically. Figure 3-26 shows a lock-in process that occurs when the frequency difference $\Delta\omega$ equals the lock-in range of the PLL, $\Delta\omega_L$. The sine wave presented in the figure represents $K_o u_f(t)$, which in this case equals $\Delta\omega_L$. The figure shows how the frequency of the VCO output signal is increased until it equals that

of the input. Figure 3-27, on the other hand, show the case where $\Delta\omega > \Delta\omega_L$. In this case lock-in does not occur with just a beat note. It is going to take longer for the PLL to lock to the signal. For this case, acquisition will occur as a pull-in process.

3.4.2 The Pull-In Range

The pull-in process occurs when the frequency difference $\Delta\omega = \omega_i - \omega_o$ is larger than $\Delta\omega_L$. Let us take a look at figure 3-27 again. Remember that $K_o u_f(t)$, the VCO input signal is oscillatory when the PLL is not locked. It can be seen from the figure that the frequency of VCO output signal ω_o increases during the positive cycle of $K_o u_f(t)$, whereas it decreases during the negative cycle. When ω_o is modulated in the positive direction, $\Delta\omega$ reaches a minimum value called $\Delta\omega_{\min}$. When the modulation occurs in the negative direction, ω_o reaches a maximum value $\Delta\omega_{\max}$. During this process ω_o is modulated non-harmonically, making the half cycle in which ω_o is modulated in the positive direction to last longer than that in which it is modulated in the negative direction. As a consequence, the average frequency of the VCO is set to a value higher than its central frequency, reducing the difference between input and output $\Delta\omega$. Since this process is regenerative if occurs as explained, $\Delta\omega$ is reduced until it reaches the lock-in range $\Delta\omega_L$, where the PLL locks to the signal in a single beat note.

Since the process involved in deriving the formulas is very complicated, they will be presented as given by, (Best 1999 [1]):

For passive filter: $\Delta\omega_p \approx \frac{4}{\pi}\sqrt{2\varsigma\omega_n K_d K_o - \omega_n^2}$ (3.42),

If $\tau_1 >> \tau_2$ $\Delta\omega_p \approx \frac{4\sqrt{2}}{\pi}\sqrt{\varsigma\omega_n K_d K_o}$ (3.43),

For active filter: $\Delta\omega_p \approx \frac{4}{\pi}\sqrt{2\varsigma\omega_n K_d K_o - \omega_n^2}$,

If $\tau_1 >> \tau_2$ $\Delta\omega_p \approx \frac{4\sqrt{2}}{\pi}\sqrt{\varsigma\omega_n K_d K_o}$,

And for PI Filter: $\Delta\omega_p \to \infty$ (3.44),

where $\Delta\omega_p$ is the pull-in range.

3.4.3 The Pull-Out Range

The pull-out range is considered to be the dynamic range for stable operation of a PLL, (Best 1999 [1]). This range is defined as the frequency step that causes the loop to lose lock momentarily. When the PLL loses track of the signal because of a large frequency step, it returns to the lock operation through a pull-in process. Equations for the pull-out range have not been derived and calculated as in previous sections (this is the approach followed by references we used to explain this subject in this work). Instead, the equation provided is the result of computer simulations, (Best 1999 [1] and Gardner 1979 [2]). The pull-out range $\Delta\omega_{PO}$ is approximately

$$\Delta\omega_{PO} \approx 1.8\omega_n(\varsigma + 1) \quad (3.45).$$

3.4.4 The Hold-In Range

Let us assume that the PLL is tracking the input signal and that the center frequency of the signal equals that of the PLL, ω_{0c}. We start increasing the frequency of the input signal slowly so that the rate of change $\dot{\Delta\omega}$ of the frequency is negligible. Consequently, the phase error θ_e is proportional to $\Delta\omega$. We continue increasing the frequency until we reach a limit where the loop loses track of the signal. In the vicinity of that limit lies a frequency value known as hold-in, which marks the maximum frequency deviation of the input signal that the loop can tolerate, because after that point the loop loses track of the signal forever. This critical frequency value is given the name Hold-in range, and causes the phase error θ_e to attain a value equal to $\frac{\pi}{2}$. Although the hold-in range was referred to as part of the PLL's dynamic characteristics, it is really a range where the stability of the loop is conditionally stable. For this reason a PLL that operates in the hold-in range will be able to track a signal if its parameters (phase and frequency) remain static in time, since the slightest variation in phase or frequency of the input signal will make the loop to unlock forever. This static condition is impossible to maintain in real life due to noise and fluctuations in the signal that can result from environmental interaction and intrinsic operation of the oscillator generating the signal. Therefore, a PLL is never operated in this range. Despite this fact, it is important to know the hold-in range of the loop as a precaution, so that the frequency deviation of the input signal is kept below this range. However, remember that a frequency deviation close to hold-in is only tolerated by the PLL if it was already tracking the signal before the deviation occurred. Thus, if the signal has a frequency deviation close to hold-in, the

loop will never track the signal, unless the PLL have a PI loop filter (this statement will be justified later).

The hold-in range is obtained by calculating the frequency offset in the input signal that causes a phase error θ_e equal to $\frac{\pi}{2}$, (Best 1999 [1]). Thus, we start with a signal whose frequency is given by $\omega_i = \omega_{0c} + \Delta\omega_H$, where $\Delta\omega_H$ is the hold-in range. The equivalent phase signal would be $\theta_i(t) = \Delta\omega_H t$. Using the Laplace transform we obtain $\theta_i(s) = \frac{\Delta\omega_H}{s^2}$. Placing this result into equation (3.30) we obtain $e_{ss} = \lim_{t\to\infty} \theta_e(t) = \lim_{s\to 0} s \cdot \theta_e(s) = \frac{\Delta\omega_H}{K_d K_o F(0)}$. Since the PLL is not operating in the linear region, the linear model for $\theta_e(t)$ cannot be used, thus, $\theta_e(t)$ is substituted by $\sin[\theta_e(t)]$ in the expression. Now, evaluating the resulting expression for the condition when $\theta_e(t) = \frac{\pi}{2}$ (hold-in condition) yields $e_{ss} = \lim_{t\to\infty} \sin[\theta_e(t)] = \sin\left(\frac{\pi}{2}\right) = 1 = \frac{\Delta\omega_H}{K_d K_o F(0)}$.

Simplifying the equation we obtain

$$\Delta\omega_H = K_d K_o F(0) \qquad (3.46).$$

Now all we have to do is to substitute the appropriate loop filter transfer function into the expression to obtain a specific equation for each PLL case:

For the passive filter: $\Delta\omega_H = K_d K_o$ (3.47),

For the Active Filter: $\Delta\omega_H = K_d K_o K_{LF}$ (3.48),

And for the PI Filter: $\Delta\omega_H \to \infty$ (3.49).

Equation (3.49) implies that every time a PI filter is used, the PLL will eventually track and lock to the input signal. The only limitation that this PLL can have is the frequency range covered by the VCO.

Let us finish the discussion of the dynamic characteristics of the PLL by expressing the relation between them

$$\Delta\omega_L < \Delta\omega_{PO} < \Delta\omega_p < \Delta\omega_H \qquad (3.50).$$

4 The Analog Costas Loop

4.1 An Overview

In 1956, John P. Costas published the article Synchronous Communications, (Costas 1956 [22]). This article was addressing one of the growing concerns at the time regarding communication systems. Most commercial and military communication systems were employing amplitude modulation (AM) techniques to transmit information.

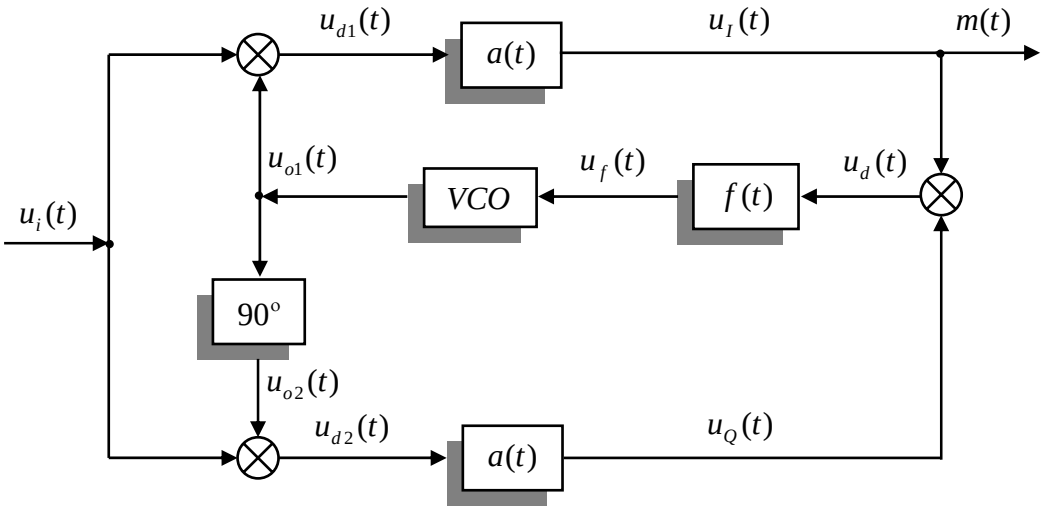

Figure 4-1. The Costas Loop Demodulator

This type of modulation, specifically Double-sideband transmitted carrier AM (DSBWC), was preferred over others proposed due to the simplicity of the systems used to implement it. However, continued growth on communication demands could not be easily met by conventional AM, imposing the need to use other modulation techniques, despite the additional system complexity. New modulation techniques were proposed and it seemed obvious that Single-Sideband AM could be used as the logical replacement for conventional

AM. Hence, Single-Sideband AM was given a great deal of publicity and support. In theory, single-sideband (SSB) allows the transmission of information more efficiently than Double-sideband Transmitted carrier. SSB uses less power and the information transmitted occupies half the bandwidth required for conventional AM; this characteristic is considered to be its primary advantage. A major disadvantage, however, is the difficulty in building a transmitter or an effective receiver, not to mention its susceptibility to jamming. This is not to say that a SSB transmitter could not be designed for simple operation, but this simplicity was obtained at the expense of additional complexity in manufacture and maintenance.

This scenario, his involvement with the technology and the increasing need for efficient and simpler modulation techniques seems to have motivated John P. Costas to question the actual benefits of SSB and propose an alternative that would efficiently generate and detect Double-sideband (DSB) suppressed carrier AM signals. DSB suppressed carrier AM uses the generated power at the transmitter more efficiently, compared to DSB Transmitted Carrier AM, because no extra energy is required to include the carrier in the transmitted signal. It was necessary to include the carrier with the transmitted DSB AM signal because DSB requires synchronous detection and demodulation. This implies that the receiver has to use an exact copy of the carrier in the received signal to achieve a successful demodulation process. This copy of the signal carrier has to be synchronized in phase and frequency. The proposed reception mechanism, known today as the Costas Loop, relies on the feedback principle related to the Phase-Locked Loop, thereby allowing for the synchronous detection and demodulation of Double-sideband suppressed carrier AM. Figure 4-1 shows the basic diagram of the Costas Loop.

Although it may not seem that obvious at first, the Costas Loop seen on the figure is essentially a system with two Phase-Locked Loops that receive the same input signal, but operate in phase quadrature to each other, (Costas 1956 [22], and Best 1999 [1]). However, as the loop moves into and operates at the lock state, it can be shown that it would actually function as a single PLL. This result will enable the development of a linearized model that can be used to design an analog Costas Loop following the same approach employed with analog PLLs.

Unlike conventional PLLs, which require the presence of a carrier in the incoming signal to lock to, the Costas Loop was designed to receive, extract carrier information and demodulate a specific type of suppressed carrier signal: DSB suppressed carrier AM. To accomplish the demodulation process, the signal received is passed through each PLL, which operating in phase quadrature to each other, remove the effects of the modulation and estimate the phase and frequency of its carrier. The estimated carrier information is then fed to a VCO to generate a waveform that is a replica of the carrier. This replica is then used to demodulate the input signal and to correct the phase and frequency estimates made by the system.

To identify each of the PLLs composing the Costas Loop and determine their phase quadrature operation, we ought to take a look again at figure 4-1. Notice the symmetry of the diagram. The upper and lower sides of the diagram, known as the *Arms* or *Channels*, are composed of a multiplier and a filter. To maintain the symmetry of the system, both filters,

also referred to as *Arm Filters*, have to be equal in design; otherwise the operation of the system would be compromised. On the center of the diagram lie the VCO, a 90°-phase shifter, the loop filter, and a third multiplier. The loop filter used in the Costas Loop takes its name from PLL theory, as this filter performs the same operation as that of the loop filter present in an analog PLL. The PLLs composing the Costas Loop are formed when the upper- and lower-side arms are connected to the center components through the multipliers.

The phase quadrature operation that takes place between the PLLs is obtained by using the 90°-phase shifter; which shifts the phase of the VCO output signal by 90 degrees and feeds it to the lower-side PLL. Because of this shifted signal, this PLL is said to be in *phase-quadrature*, hence its side arm is called *Q-Arm* or *Q-Channel*. Conversely, the signal that goes to the upper-side PLL, is not shifted, and so this PLL is said to be *in-phase* with the carrier; therefore the side arm of this loop is called *I-Arm* or *I-Channel*. As an example, assume the carrier of the input signal is a sine wave. As the Costas Loop is able to estimate its phase and frequency, the VCO output will look much like the carrier, a sine wave. This generated sine wave replica is fed directly to the upper-side arm of the loop, whereas the lower-side arm receives a 90-degree phase shifted version (cosine wave); this shifted signal is said to be in phase quadrature. On the other hand, if the carrier were a cosine, the VCO output signal would be a cosine wave, whereas the shifter output signal would be a sine wave. Both examples show how no matter the type of sinusoid used as the carrier, the upper-side arm would always be in phase with the carrier, whereas the lower-side arm would always be in phase quadrature.

How is it that having two PLLs operating in phase quadrature allows the demodulation of a DSB suppressed carrier AM signal? Suppose that the Costas Loop is locked to the carrier of the input signal and this carrier is a sine wave. From communications theory we know that in order to demodulate a DSB suppressed carrier AM signal, we have to multiply the signal by its carrier and low-pass filter the results. This task is accomplished by the upper-side PLL, namely its multiplier and arm filter. Therefore the output of the I-Arm filter would contain the demodulated signal. Unfortunately, a PLL cannot track and lock to a suppressed carrier signal by itself, hence the lower-side PLL had to be added, as well as a third multiplier. When the Costas Loop is in lock, the output of the Q-Arm will be approximately zero, and so the output of the third multiplier. The resulting signal in the Q-channel bases its properties on the quadrature null effect obtained from multiplying two sinusoids that are exactly 90 degrees apart from each other. Prior to the arm filter, there is a second component at twice the carrier frequency, but the arm filter eliminates this component, hence it is not seen at the output. Should a small phase drift occur on either the input carrier or the generated replica, the output of the I-channel would remain essentially the same, whereas the signal level at the Q-channel would no longer be zero. Instead, it would acquire a voltage level proportional to the magnitude of the phase error detected, and its polarity would equal the I-channel polarity for a positive direction of the phase drift and opposite polarity for a negative direction of the phase drift. Therefore, by multiplying the signals on the I- and Q- channels, we obtain the desired DC control signal that is used to adjust and correct the parameters of the locally generated replica.

4.2 Time Domain Analysis

The previous section concentrated on the origins of the Costas Loop and presented a concise description and explanation of its operation and importance. The description provided, however, lacked the mathematical analysis required to further understand its operation and achieve a realizable design. This section provides the mathematical analysis, based on the time domain. The theory provided in this and all other sections of this chapter was obtained from (Costas 1956 [22], and Best 1999 [1]).

Figure 4-1 shows a block diagram of the analog Costas Loop with components and signals expressed in the time domain. These components are three multipliers, two arm filters $a(t)$, a loop filter $f(t)$, a VCO and a 90-degree phase shifter. The signals seen on the figure will be defined as we progress with the discussion.

At the input to the system we have $u_i(t)$, defined on equation 4.1 as a DSB suppressed carrier AM signal with amplitude A_i, modulating signal $m(t)$ and a sine wave as the carrier. This sine wave will have frequency ω_i and phase θ_i. Unless otherwise specified, amplitude terms will have units of *Volts*, frequency terms will have units of $\frac{radians}{\sec ond}$ and phase terms will have units of *radians*. The modulating signal $m(t)$, represents digital data encoded in Non-Return-to-Zero format (*NRZ*).

$$u_i(t) = A_i m(t)\sin(\omega_i t + \theta_i) \quad (4.1),$$

$$u_{o1}(t) = A_o \sin(\omega_i t + \theta_o) \quad (4.2),$$

$$u_{o2}(t) = A_o \cos(\omega_i t + \theta_o) \quad (4.3).$$

Although equation 4.1 corresponds to a DSB suppressed carrier AM signal, by defining $m(t)$ as a NRZ signal, we make equation 4.1 be the definition of the Binary Phase Shift Keying (*BPSK*) signal that will be the input to the system. As the Costas Loop has a quadrature-phase type operation, there are two output signals to be defined: $u_{o1}(t)$ and $u_{o2}(t)$. $u_{o1}(t)$ is the signal that is in-phase with the carrier, whereas $u_{o2}(t)$ will be in phase quadrature to the carrier. In order to mathematically define each of these signals, first we have to know the carrier of the input signal. Since the carrier of the input signal is a sine wave, as seen on equation 4.1, we can define $u_{o1}(t)$ as a sine wave and $u_{o2}(t)$ as a cosine wave; when the system is locked these are the signals that will be seen at the output of the VCO and 90-degree phase shifter, respectively. Both waveforms will have amplitude A_o, frequency ω_i and phase θ_o.

As mentioned on section 4.1, the Costas Loop can be broken into two PLLs, one operating in phase and the other in phase *quadrature*. This configuration gives the Costas Loop the ability to detect and demodulate DSB suppressed carrier AM signals and any other signal that can be expressed in this format, such as BPSK. Without a carrier signal added to the input, a PLL fails to detect and demodulate a signal such as $u_i(t)$, because after the detection and filtering process the signal entering the VCO would be like $u_f(t) = \frac{A_i A_o m(t)}{2} \sin(\theta_i - \theta_o)$. In addition to having a wide bandwidth, this signal is subject to variations in sign resulting from $m(t)$. As we defined $m(t)$ to be a NRZ signal, its amplitude varies from $+V$ to $-V$; for simplicity purpose, we will assume that the magnitude varies from 1 to –1. Due to this change in polarity the VCO sees a phase error of

$\theta_e = \theta_i - \theta_o + \pi$ instead of $\theta_e = \theta_i - \theta_o$, hence, the loop locks in anti-phase with the carrier. Therefore, in order to solve this problem, the effects of the modulation have to be eliminated. Eliminating the effects of the modulation is what the Costas Loop does to properly detect and demodulate the received signal.

Now that we have the input and output signals properly defined, we can proceed with the analysis of the Costas Loop. Referring again to figure 4-1 we see that $u_i(t)$ is multiplied to $u_{o1}(t)$ and $u_{o2}(t)$ at the upper- and lower-side multipliers, respectively. Using trigonometric identities, the signals obtained from these multiplications are:

$$u_{d1}(t) = \frac{A_i A_o}{2} m(t)\cos(\theta_i - \theta_o) - \frac{A_i A_o}{2} m(t)\sin(2\omega_i t + \theta_i + \theta_o) \qquad (4.4),$$

$$u_{d2}(t) = \frac{A_i A_o}{2} m(t)\sin(\theta_i - \theta_o) + \frac{A_i A_o}{2} m(t)\sin(2\omega_i t + \theta_i + \theta_o) \qquad (4.5).$$

At these two multipliers phase detection already takes place as the first term on each of their outputs has a component with phase difference $\theta_i - \theta_o$. It is these two components that have the phase error information the VCO will use to generate a replica of the carrier. Then again, the VCO will not be able to use this phase error information unless the modulation component attached to it is removed. At this point, this modulation component is at base-band and has a bandwidth in Hertz equal to the data rate, R; the base-band bandwidth of a BPSK signal is equal to its data rate. Equations 4.4 and 4.5 also have a high frequency term, which is at twice the carrier frequency, $2\omega_i$. These two terms are undesirable and need to be removed. Removal of these components is accomplished through the arm filters. Although not considered for this project, it is worth mentioning that the arm filters can be designed to be matched filters to provide improved noise immunity to the system [8]. Nevertheless, for

simplicity a low-pass filter can be chosen, which can be designed to approximate a matched filter by making its bandwidth equals to R, the data rate of $m(t)$.

Signals $u_{d1}(t)$ and $u_{d2}(t)$ are passed through the arm filters to remove their high frequency components. For the purpose of this analysis we will assume the arm filters are low-pass and their output signals are $u_I(t)$ and $u_Q(t)$, respectively. These two signals are represented by equations 4.6 and 4.7. Having the high frequency terms successfully removed by the arm filters, signals $u_I(t)$ and $u_Q(t)$ are multiplied to obtain $u_d(t)$, as given by equation 4.8.

$$u_I(t) = \frac{A_i A_o m(t)}{2}\cos(\theta_i - \theta_o) \qquad (4.6),$$

$$u_Q(t) = \frac{A_i A_o m(t)}{2}\sin(\theta_i - \theta_o) \qquad (4.7).$$

It is at this multiplier that the modulation component $m(t)$ is removed and the last stage of the phase detection process takes place. However, unlike conventional PLLs, the Costas Loop will track a doubled-phase error $2(\theta_i - \theta_o)$, as shown on equation 4.8.

$$u_d(t) = \frac{\left[A_i A_o m(t)\right]^2}{8}\sin(\theta_i - \theta_o - \theta_i + \theta_o) + \frac{\left[A_i A_o m(t)\right]^2}{8}\sin\left[2(\theta_i - \theta_o)\right]$$

$$u_d(t) = \frac{\left[A_i A_o m(t)\right]^2}{8}\sin(0) + \frac{\left[A_i A_o m(t)\right]^2}{8}\sin\left[2(\theta_i - \theta_o)\right]$$

$$u_d(t) = \frac{\left(A_i A_o\right)^2}{8}\sin\left[2(\theta_i - \theta_o)\right] \qquad (4.8)$$

Since $m(t)$ was defined as a NRZ signal whose amplitude varies from 1 to –1, squaring this signal makes it possible to remove its modulating effect as $m^2(t)=1$. To finish the process, signal $u_d(t)$ is low-pass filtered by $f(t)$, the Loop Filter, to obtain $u_f(t)$.

$$u_f(t)=\frac{(A_i A_o)^2}{8}\sin[2(\theta_i-\theta_2)] \qquad (4.9),$$

It is obvious that equations 4.8 and 4.9 represent the same signal, that is, assuming the loop filter is wide enough to let $u_f(t)$ pass through unaffected. Hence, we could ask ourselves: what is the purpose of the loop filter in the Costas Loop? First of all, based on PLL theory, we know that it is the loop filter that determines important characteristics of the system, such as steady-state response to variations in the input, as covered in chapter three. Nevertheless, the importance of this filter becomes obvious when we include noise effects in the analysis. As a quick example, let us assume that additive Gaussian noise is present at the input signal and that this signal has been pre-filtered before being fed to the Costas Loop. The filter used at this stage is band-pass with bandwidth $2R$; when a BPSK signal is not at base-band frequency, it occupies a bandwidth of twice its data rate. This signal is fed to the Costas loop where it is multiplied by a locally generated replica of its carrier, and a 90-degree shifted version of it, to obtain signals $u_{d1}(t)$ and $u_{d2}(t)$. These two signals are low-pass filtered by the arm filters with bandwidth equal to R. This is the bandwidth of signals $u_I(t)$ and $u_Q(t)$, which includes $m(t)$ plus the noise. At the third multiplier, $u_d(t)$ the resulting signal has a bandwidth equal to R. This bandwidth is occupied by the phase error information, which is at or close to DC level, and the noise, which occupies the whole bandwidth given by R. At this point the use of the loop filter becomes obvious. This low-pass filter can be designed to

have a bandwidth much narrower than R, which will reduce the noise fed to the VCO even further, hence, improving system performance.

Now that we understand the importance of the loop filter in the Costas loop, we can take the analysis of equation 4.9 a bit further. Suppose the system is locked or close to lock to the carrier signal. In such case, the phase difference between the carrier of the input signal and its locally generated replica will be zero or close to zero, hence $\theta_i \approx \theta_o$. If this assumption holds true, we can use the small angle approximation to simplify equation 4.9. From the small angle approximation, $\sin[2(\theta_i - \theta_o)] \approx 2(\theta_i - \theta_o)$. Defining $\frac{(A_i A_o)^2}{4}$ as K_d, the phase detector gain (this is the gain at the third multiplier), we obtain

$$u_f(t) \approx \frac{K_d}{2}[2(\theta_i - \theta_o)]$$

$$u_f(t) = K_d(\theta_i - \theta_o) \qquad (4.10).$$

Using the small angle approximation, equation (4.10) shows that the Costas Loop would actually track a single-phase error when it is at or close to the locked state. This result is very important, as equation (4.10) is exactly the same outcome obtained on equation (3.8) for the conventional PLL.

Up to this point, we have developed the discussion and presented equations geared towards the analysis of the phase-track capabilities of the system, but have not discussed yet how does the demodulation process takes place. In order to develop this discussion, we have to make use of equations (4.6) and (4.7). The main goal of the Costas Loop is to recover the transmitted data, which in this case, it is encoded as a NRZ waveform, and it is received as a

DSB suppressed carrier AM signal. To do so, it estimates the phase and frequency parameters of the input carrier, modulated by $m(t)$, and generates a replica that is used to correct these estimates. If these parameters are estimated correctly, the loop will move to the lock state, in which case the phase error will be close to zero; that is, $\theta_i \approx \theta_o$. When this condition is met, equations (4.6) and (4.7) become equations (4.11) and (4.12).

$$u_I(t) = \frac{A_i A_o}{2} m(t)\cos(0)$$

$$u_I(t) = \frac{A_i A_o}{2} m(t) \tag{4.11}$$

$$u_Q(t) = \frac{A_i A_o}{2} m(t)\sin(0)$$

$$u_Q(t) = 0 \tag{4.12}$$

These results clearly show that when the *Costas Loop* is locked, $m(t)$, the desired signal, can be obtained directly from the I-Channel. This eliminates the need to implement extra circuitry to attain its recovery. Since $m(t)$ is multiplied by the constant $\frac{A_i A_o}{2}$, if we set $A_i = 1$ and $A_o = 2$, $u_I(t)$ in the I-Channel becomes $m(t)$.

4.3 Linear Model of the Costas Loop

When we first got involved with the analysis and design of a Costas Loop, most publications would provide a high level analysis of the system and did not attempt to develop a simpler linear model, such as that available for PLLs. Why is it that a linear model is

always available or mentioned on most publications related to PLLs, yet we could not obtain such a document for the Costas Loop topic? Even though we do not have an answer for this question, we do know that some of the analysis developed for PLLs can be applied to the analysis and design of Costas Loops. How is this possible? Is it possible because the Costas Loop (along with its added components) is still a PLL, and it can be modeled as two PLLs, one operating in-phase and the other in phase-quadrature? On this section, we will attempt to provide a linear model for this system, and in order to do so, we will employ some of the equations provided on the previous sections and the assumption that this system can be modeled as two PLLs.

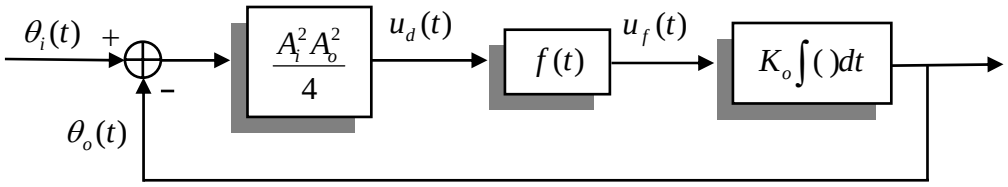

Figure 4-2. Block Diagram of Linear Model of the Costas Loop in the Time Domain

So far, the analysis of the Costas Loop is assuming that the input to the system is a BPSK modulated signal expressed as DSB suppressed carrier AM. What if we consider the case where the input is just an un-modulated carrier of the form $u_i(t) = A_i \sin(\omega_i t + \theta_i)$ and the system is already locked to this signal? Equations (4.4) and (4.5) show the output of the first two multipliers as

$$u_{d1}(t) = \frac{A_i A_o}{2}\cos(\theta_i - \theta_o) - \frac{A_i A_o}{2}\sin(2\omega_i t + \theta_i + \theta_o) \qquad (4.13)$$

$$u_{d2}(t) = \frac{A_i A_o}{2}\sin(\theta_i - \theta_o) + \frac{A_i A_o}{2}\sin(2\omega_i t + \theta_i + \theta_o) \qquad (4.14).$$

The arm filters remove the high frequency terms and based on equations (4.6) and (4.7) their outputs are

$$u_I(t) = \frac{A_i A_o}{2}\cos(\theta_i - \theta_o) \quad (4.15),$$

$$u_Q(t) = \frac{A_i A_o}{2}\sin(\theta_i - \theta_o) \quad (4.16).$$

These two signals are then fed to the third multiplier, where the result is exactly the same as that provided by equation 4.8

$$u_d(t) = \frac{(A_i A_o)^2}{8}\sin[2(\theta_i - \theta_o)] \quad (4.17).$$

$u_d(t)$ is passed through the loop filter to obtain

$$u_f(t) = \frac{(A_i A_o)^2}{8}\sin[2(\theta_i - \theta_2)] \quad (4.18).$$

If we now assume that the loop is locked or close to lock, equation (4.18) can be expressed as

$$u_f(t) = K_d(\theta_i - \theta_o) \quad (4.19).$$

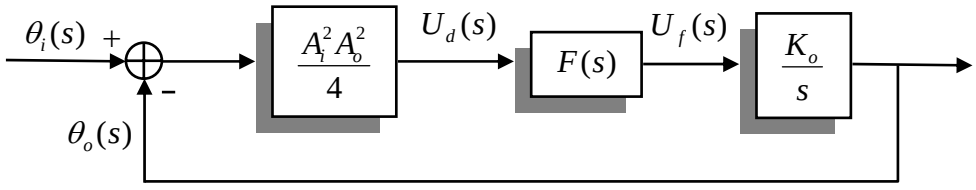

Figure 4-3. Block Diagram of Linear Model of the Costas Loop in the Frequency Domain

Using the locked state assumption, we can see that equation (4.15) would become the constant $\frac{A_i A_o}{2}$. It is $u_Q(t)$ that will keep the system locked to the carrier. On the next step

of the process, $u_Q(t)$ is multiplied to $u_I(t)$, which is the constant $\frac{A_i A_o}{2}$. Since the I-Channel contribution has become a DC level of magnitude $\frac{A_i A_o}{2}$, I propose to substitute the whole I-Channel by a gain term equal to $\frac{A_i A_o}{2}$. By implementing this change we will have to remove three components from the diagram: the multiplier that feeds the signal to the I-Channel, the third or rightmost multiplier, and the 90-degree phase shifter. This is an important assumption as the result leads us to a PLL model with a small variation in the phase detector gain. A block diagram of the resulting model, is presented on figures 2 and 3. Figure 3, however, shows the linearized model expressed in the frequency domain. From chapter three, it is obvious that this model has a closed-loop transfer function of the form given by equation (3.22), that is

$$H(s) = \frac{KF(s)}{s + KF(s)} \qquad (4.20).$$

For this transfer function, the gain term $K = K_d \times K_o$. But $K_d = \frac{(A_i A_o)^2}{4}$, instead of $K_d = \frac{A_i A_o}{2}$, as it occurs for the conventional PLL. The analysis presented on this section would be of importance, as it shows that the Costas Loop essentially functions as a PLL when it is operating in the locked state. It is for this reason that I devoted so much time presenting the PLL theory on chapter three.

5 Discrete-Time Model of the PLL

Chapters three and four show the theory that applies to analog phase-locked loops and analog Costas loops. As part of the theory, time and frequency analysis of both systems was presented, including linear models on the frequency domain that help define the parameters for the desired characteristics, understand the response of the system to various excitation signals and achieve its final design implementation. All the theory included on these two chapters, can be obtained on any of the references provided along with this project, except for the time and frequency models developed for the for the Costas Loop. These models were mainly developed to show the relation between Costas Loops and PLLs, and will be used later on to design and implement a Costas Loop demodulator. Based on the analog phase-locked loop (APLL) theory and its similarity to that of the Costas Loop, specifically its linearized model, we will derive a discrete-time model that can be used to design a discrete-time PLL or Costas Loop and achieve its subsequent digital implementation.

A digital implementation of a PLL system has many advantages over its analog counterpart. For example, (Shayan et al 1989 [21]) establishes that a digital PLL has better dynamic tracking ability because perfect integrators can be realized in the loop filter (only approximations are implemented in the analog domain). Furthermore, digital implementations have proven to solve other problems, such as sensitivity to DC drifts, component saturation, difficulties building higher-order loops, and the need for initial calibration and periodic adjustment. Considering the specific characteristics of the Costas Loop, a digital implementation provides yet an additional advantage: complete signal balance

between I- and Q- channel. It is well known (Costas 1956 [22]) of the Costas Loop depends upon the balance between the I- and Q-Channel. This balance is achieved by designing both arm filters with equal characteristics, namely bandwidth, gain and frequency response.

An analog Costas Loop is composed of three multipliers, two arm filters, a loop filter and a VCO. An analog PLL, on the other hand, does not have arm filters and uses only one multiplier, along with a loop filter and a VCO. In order to have a discrete-time and subsequent digital implementation of either a PLL or Costas Loop, while keeping the same architectural block diagram design as their analog counterparts, each of their components can be discretized and digitized. Discrete-time implementation of these systems can be realized by following the procedure presented in (Best 1999 [1]) for linear digital PLLs.

5.1 Discrete-Time Loop Filter

An analog PLL achieves outstanding performance when the loop filter is of the PI type. The transfer function of this filter is given by equation (3.26) as $F(s)=\frac{\tau_2 s+1}{\tau_1 s}$. A discrete-time version of this filter can be obtained by using the bilinear transformation. The bilinear transformation enables us to calculate the z-transfer function of a system directly from its s-transfer function (Laplace transfer function)

$$s=\frac{2}{T}\cdot\frac{1-z^{-1}}{1+z^{-1}} \qquad (5.1).$$

Substituting equation (5.1) into equation (3.26) we obtain $F(z)$, the z-transfer function of $F(s)$,

$$F(z)=\frac{\tau_2\left(\frac{2}{T}\cdot\frac{1-z^{-1}}{1+z^{-1}}\right)+1}{\tau_1\left(\frac{2}{T}\cdot\frac{1-z^{-1}}{1+z^{-1}}\right)}$$

$$F(z)=\frac{\frac{2\tau_2+T}{2\tau_1}+\frac{T-2\tau_2}{2\tau_1}z^{-1}}{1-z^{-1}}=\frac{b_o+b_1z^{-1}}{1-z^{-1}} \qquad (5.2),$$

where $b_o=\frac{2\tau_2+T}{2\tau_1}$, and $b_1=\frac{T-2\tau_2}{2\tau_1}$. In (Best 1999 [1]), constants b_o and b_1 are defined as $b_o=\frac{T}{2\tau_1}\cdot\left[1+\frac{1}{\tan\left(\frac{T}{2\tau_2}\right)}\right]$, and $b_1=\frac{T}{2\tau_1}\cdot\left[1-\frac{1}{\tan\left(\frac{T}{2\tau_2}\right)}\right]$. If the sampling period T is very small, then the equations presented in (Best 1999 [1]) are equal to the ones just derived here using the bilinear transformation.

From equation (5.2) we can derive a recursive equation that we can use to implement the loop filter either physically or by software. (Oppenheim et al 1989 [7]) describes a well known procedure that can be applied to a rational expression in the z-domain (representing a Linear time-invariant system) to obtain a linear constant-coefficient difference equation of the form $\sum_{k=0}^{N}a_k y[n-k]=\sum_{k=0}^{M}b_k x[n-k]$. For our digital loop filter, $F(z)=\frac{b_o+b_1z^{-1}}{1-z^{-1}}=\frac{U_f(z)}{U_d(z)}$, where $U_d(z)$ is the output signal of the phase detector (input to the loop filter) and $U_f(z)$ is the output of the loop filter expressed in the $z-$domain. Using the procedure presented in (Oppenheim et al 1989 [7]), which is also employed in (Best 1999 [1]), we obtain

$$\left(1-z^{-1}\right)\cdot U_f(z)=\left(b_o+b_1z^{-1}\right)\cdot U_d(z)$$

$$u_f[n] - u_f[n-1] = b_o u_d[n] + b_1 u_d[n-1] \quad (5.3),$$

$$u_f[n] = b_o u_d[n] + b_1 u_d[n-1] + u_f[n-1] \quad (5.4).$$

Equation (5.3) is a linear constant-coefficient difference equation that can be implemented through a recursive calculation process. From equation (5.3) we move to equation (5.4), which indicates that the current value of the loop filter output $u_f[n]$, depends on the past value of its output $u_f[n-1]$, the current value of its input $u_d[n]$ and the past value of its input $u_d[n-1]$, weighted by their respective constants.

5.2 The Numerically Controlled Oscillator

The Voltage-Controlled Oscillator used in the analog PLL generated a sinusoid signal whose phase and frequency was controlled by the output signal of the loop filter. Since PLLs are feedback systems, the input signal is compared to that of the VCO output and an error signal is generated to correct the phase and frequency parameters of the locally generated VCO signal. This process is exactly followed by DPLLs; however, as other components are digitized, the VCO is substituted by its digital version, the Numerically Controlled Oscillator (NCO). This NCO generates a sine wave whose value is known at the sampling instants $t = 0, T, 2T, ..., nT$. (Best 1999 [1]) presents a process to derive a recursive equation to determine the phase of the NCO output signal and attain its subsequent implementation. This process is shown below.

Equation (3.5) provides the instantaneous angular frequency of the VCO output signal, $\omega_o = \omega_{0c} + K_o u_f(t)$. The total phase of this signal can be obtained by integrating

equation (3.5) with respect to time. Back in Chapter 3, when the analysis of the VCO was presented, we only integrated over the frequency variation $K_o u_f(t)$, which resulted in the parameter $\theta_o(t)$. ω_{0c} was not used in the calculation because it is a constant that represents the center frequency of the VCO and it is established and fixed with external components; it is only the frequency variation term that carries the information to adjust the output frequency of the VCO. To derive the NCO recursive equation, however, we will have to calculate the total phase of the VCO output signal by integrating the instantaneous frequency with respect to time.

$$\vartheta_o(t) = \int \omega_o(t)\, dt = \int \left(\omega_{0c} + K_o u_f(t)\right) dt$$
$$\vartheta_o(t) = \omega_{0c} t + K_o \int u_f(t)\, dt \qquad (5.5).$$

To develop the recursive equation, let us assume that we sample the Loop Filter output signal $u_f(t)$ and we know its value at sampling instant $t = nT$. Let us further assume that this sample stays constant for a whole period of time that we can represent as $(n-1)T \le t \le nT$. Given that $u_f(t)$ is constant in an interval equal to a sampling period, we can calculate $\Delta\vartheta_o(t)$, the change in the total phase of the NCO output signal in that interval.

$$\Delta\vartheta_o = \int_{(n-1)T}^{nT} (\omega_{0c} + K_o u_f[nT])dt$$
$$\Delta\vartheta_o = \omega_{0c} t\Big|_{t=(n-1)T}^{t=nT} + K_o u_f[nT]\Big|_{t=(n-1)T}^{t=nT}$$
$$\Delta\vartheta_o = \omega_{0c} nT - \omega_{0c} nT + \omega_{0c} T + K_o u_f[nT]nT - K_o u_f[nT]nT + K_o u_f[nT]T$$
$$\Delta\vartheta_o = \omega_{0c} T + K_o T u_f[nT]$$
$$\Delta\vartheta_o = \left(\omega_{0c} + K_o u_f[n]\right)T \qquad (5.6).$$

Assuming that we know the phase $\vartheta_o[n-1]$ at sampling instant $t = (n-1)T$, we could use equation (5.6) to extrapolate the total phase $\vartheta_o[n]$ at sampling instant $t = nT$, using this recursive equation

$$\vartheta_o[n] = \vartheta_o[n-1] + \left(\omega_{o_c} + K_o u_f[n]\right)T \qquad (5.7).$$

The above procedure was presented in (Best 1999 [1]) assuming the time period $nT \le t \le (n+1)T$. Although I will not show the derivation required to obtain $\Delta\vartheta_o(t)$ with this new interval, the interested reader can very well substitute these new values to prove that the resulting $\Delta\vartheta_o(t)$ is the same as that shown in equation (5.6). Using this result, we can derive another recursive equation. This time, we assume that we know the phase $\vartheta_o[n]$ at sampling instant $t = nT$ and we can extrapolate the total phase $\vartheta_o[n+1]$ at sampling instant $t = (n+1)T$ using the recursive equation

$$\vartheta_o[n+1] = \vartheta_o[n] + \left(\omega_{o_c} + K_o u_f[n]\right)T \qquad (5.8).$$

Equations (5.7) and (5.8) are recursive equations that implement a digital integrator, also known as an accumulator. Equation (5.7) could be considered the classical implementation of an accumulator. Equation (5.8) was obtained directly from (Best 1999 [1]) and has been added so that we can analyze the implementation using software System View, by Elanix; the NCO available in this software has an accumulator implementation of the form of equation (5.8). For both equations, $K_o T$ is the gain of the NCO, ω_{0c} is a constant that represents the center frequency in $Rads/s$ and $\omega_{0c}T$ is its corresponding phase value; $u_f[n]$ is the output signal of the discrete-time Loop Filter at sampling instant n. (Oppenheim et al 1989 [7]) shows that it is common practice to represent a discrete-time function like $u_f[nT]$ as $u_f[n]$ by dropping the variable T that represents the sampling period. This is usually done to normalize the time axis and express it as a sample axis.

These recursive equations can be easily implemented in software. If we write a program to calculate $\vartheta_o[n]$ based on equation (5.7) or $\vartheta_o[n+1]$ based on equation (5.8), and we execute the program for a long period of time, $\vartheta_o[n]$ and $\vartheta_o[n+1]$ will become very large, causing an overflow. To avoid this situation $\vartheta_o[n]$ and $\vartheta_o[n+1]$ can be limited to a range of 2π radians, say $-\pi \le \vartheta_2 < \pi$. (Best 1999 [1]) suggests that to bound $\vartheta_o[n]$ to the 2π period $-\pi \le \vartheta_2 < \pi$, we should subtract 2π whenever $\vartheta_o[n]$ and $\vartheta_o[n+1]$ exceed π. Having recursive equations (5.7) and (5.8) we can determine their respective transfer function $N(z)$, in the $z-$domain using the procedure presented in (Oppenheim et al 1989 [7]).

$$\vartheta_o[n] - \vartheta_o[n-1] = \left(\omega_o + K_o u_f[n]\right)T$$

$$\vartheta_o(z)\left(1 - z^{-1}\right) = K_o T U_f(z)$$

$$\frac{\vartheta_o(z)}{U_f(z)} = \frac{K_o T}{\left(1 - z^{-1}\right)} = N(z) \qquad (5.9).$$

When the NCO uses recursive equation

$$\vartheta_o[n+1] - \vartheta_o[n] = \left(\omega_o + K_o u_f[n]\right)T$$

$$\vartheta_o(z)\left(z - 1\right) = K_o T U_f(z)$$

$$\frac{\vartheta_o(z)}{U_f(z)} = \frac{K_o T}{\left(z - 1\right)} = \frac{K_o T z^{-1}}{\left(1 - z^{-1}\right)} = N(z) \qquad (5.9).$$

$\omega_o T$ was not used to calculate the transfer function because it is a constant whose only purpose is to establish the center frequency (and its respective phase value) of the NCO. Equations (5.9) and (5.10) are the transfer function of a specific accumulator. Each accumulator type is the digital version of the analog integrator that appears in the VCO.

5.3 Closed-Loop Transfer Function

Now that we have the transfer function of all the components of the digital PLL, we can determine the closed-loop transfer function. From control theory, the general form of the closed-loop transfer function is

$$H(z) = \frac{K_d F(z) N(z)}{1 + K_d F(z) N(z)} \tag{5.11}.$$

Substituting the NCO transfer function given by (5.9) into equation (5.11) we obtain

$$H(z) = \frac{K_d K_o T F(z)}{1 - z^{-1} + K_d K_o T F(z)} \tag{5.12}.$$

Substituting the NCO transfer function given by (5.10) into equation (5.11) we obtain

$$H(z) = \frac{K_d K_o T F(z) z^{-1}}{1 - z^{-1} + K_d K_o T F(z) z^{-1}} \tag{5.13}.$$

Finally, we substitute the transfer function of the equivalent digital PI loop filter, $F(z)$ as given by equation (5.2), into equations (5.12) and (5.13).

When using equation (5.12) we obtain

$$H(z) = \frac{K_d K_o T (b_o + b_1 z^{-1})}{(1 - z^{-1})^2 + K_d K_o T (b_o + b_1 z^{-1})}$$

$$H(z) = \frac{K_d K_o T b_o + K_d K_o T b_1 z^{-1}}{1 - 2z^{-1} + z^{-2} + K_d K_o T b_o + K_d K_o T b_1 z^{-1}}$$

$$H(z) = \frac{K_d K_o T b_o + K_d K_o T b_1 z^{-1}}{(K_d K_o T b_o + 1) + (K_d K_o T b_1 - 2) z^{-1} + z^{-2}}$$

$$H(z) = \frac{\dfrac{K_d K_o T b_o}{(K_d K_o T b_o + 1)} + \dfrac{K_d K_o T b_1}{(K_d K_o T b_o + 1)} z^{-1}}{1 + \dfrac{(K_d K_o T b_1 - 2)}{(K_d K_o T b_o + 1)} z^{-1} + \dfrac{1}{(K_d K_o T b_o + 1)} z^{-2}}$$

$$H(z) = \frac{d_o + d_1 z^{-1}}{1 + c_1 z^{-1} + c_2 z^{-2}} \qquad (5.14),$$

where $c_1 = \frac{K_d K_o T b_1 - 2}{K_d K_o T b_o + 1}$, $c_2 = \frac{1}{K_d K_o T b_o + 1}$, $d_o = \frac{K_d K_o T b_o}{K_d K_o T b_o + 1}$, and $d_1 = \frac{K_d K_o T b_1}{K_d K_o T b_o + 1}$.

When using equation (5.13) we obtain

$$H(z) = \frac{K_d K_o T (b_o + b_1 z^{-1}) z^{-1}}{(1 - z^{-1})^2 + K_d K_o T (b_o + b_1 z^{-1}) z^{-1}}$$

$$H(z) = \frac{K_d K_o T b_o z^{-1} + K_d K_o T b_1 z^{-2}}{1 - 2z^{-1} + z^{-2} + K_d K_o T b_o z^{-1} + K_d K_o T b_1 z^{-2}}$$

$$H(z) = \frac{K_d K_o T b_o z^{-1} + K_d K_o T b_1 z^{-2}}{1 + (K_d K_o T b_o - 2) z^{-1} + (K_d K_o T b_1 + 1) z^{-2}}$$

$$H(z) = \frac{d_o z^{-1} + d_1 z^{-2}}{1 + c_1 z^{-1} + c_2 z^{-2}} \qquad (5.15),$$

where $c_1 = K_d K_o T b_o - 2$, $c_2 = K_d K_o T b_1 + 1$, $d_o = K_d K_o T b_o$, and $d_1 = K_d K_o T b_1$.

Closed-loop transfer functions (5.14) and (5.15) have gain terms K_d, K_o and T. It would be practical to define a new gain variable to represent the gain of the DPLL. This variable would be called the digital gain and will be represented as

$$K_D = K_d K_o T \qquad (5.16).$$

Note that since T has been added as a gain term, the overall digital gain K_D is going to be much smaller than K, the analog gain term, where $K = K_d K_o$; this is assuming that $T << 1$.

5.4 The Error Transfer Function

In chapter 3 section 3.3.2, we dealt with the steady-state error of the analog PLL (APLL). To perform that analysis we made use of equation (3.27), $\theta_e(s) = [1 - H(s)]\theta_i(s)$, the phase error signal. In that equation, the expression $1 - H(s)$ represented the error

transfer function $H_e(s)$. The same concept applied to analog PLLs can be applied to DPLLs. The error transfer function can be expressed as $H_e(z) = \frac{\theta_e(z)}{\theta_i(z)} = 1 - H(z)$, where $\theta_e(z) = \theta_i(z) - \vartheta_o(z)$; $\theta_i(z)$ and $\vartheta_o(z)$ are the phase of the input and output signals, respectively. To derive the Error transfer function, we will make use of equations (5.12) and (5.13). But we will substitute K_d, K_o and T by $K_D = K_d K_o T$. Using equation (5.12) we obtain that $H_e(z)$ is

$$H_e(z) = 1 - H(z) = 1 - \frac{K_D F(z)}{1 - z^{-1} + K_D F(z)}$$

$$H_e(z) = \frac{1 - z^{-1}}{1 - z^{-1} + K_D F(z)}$$

$$H_e(z) = \frac{(1 - z^{-1})^2}{(1 - z^{-1})^2 + K_D (b_o + b_1 z^{-1})} \qquad (5.17).$$

Using equation (5.13) $H_e(z)$ is

$$H_e(z) = 1 - H(z) = 1 - \frac{K_D F(z) z^{-1}}{1 - z^{-1} + K_D F(z) z^{-1}}$$

$$H_e(z) = \frac{1 - z^{-1}}{1 - z^{-1} + K_D F(z) z^{-1}}$$

$$H_e(z) = \frac{(1 - z^{-1})^2}{(1 - z^{-1})^2 + K_D (b_o z^{-1} + b_1 z^{-2})} \qquad (5.18)$$

Equations (5.17) and (5.18) were obtained assuming $F(z)$ is of the form given by equation (5.2).

5.5 The Steady-State Error of the DPLL

Using the final value theorem, the steady-state error was defined in section 3.3.2 as $e_{ss} = \lim_{s\to 0} s\,\theta_e(s)$. From (Shayan et al 1989 [21]), the steady-state error is defined in the $z-$domain as

$$e_z = \lim_{z\to 1}(1-z^{-1})\theta_e(z) = \lim_{z\to 1}\left(\frac{z-1}{z}\right)\theta_e(z) \qquad (5.19).$$

Before continuing the discussion, it would be a good idea to express equations (5.17) and (5.18) in terms of z rather than z^{-1}

$$H_e(z) = \frac{(z-1)^2}{(z-1)^2 + K_d K_o T(b_o z^2 + b_1 z)}, \text{ from equation (5.17);}$$

$$H_e(z) = \frac{(z-1)^2}{(z-1)^2 + K_d K_o T(b_o z + b_1)}, \text{ from equation (5.18).}$$

Using these results and equation (5.19) we determine that the steady state error for each of the DPLL with PI loop filter is

$$e_z = \lim_{z\to 1}\left(\frac{z-1}{z}\right)\cdot\frac{(z-1)^2}{(z-1)^2 + K_d K_o T(b_o z^2 + b_1 z)}\theta_i(z) \qquad (5.20)$$

$$e_z = \lim_{z\to 1}\left(\frac{z-1}{z}\right)\cdot\frac{(z-1)^2}{(z-1)^2 + K_d K_o T(b_o z + b_1)}\theta_i(z) \qquad (5.21).$$

Since the final result for each of the DPLLs is the same, I will only show the results for equation (5.21). This way I will not show the same results over and over. I picked equation (5.21) to show the results because it represents the DPLL implemented (simulated) using software System View, by Elanix. However, the interested reader can substitute each of the input signals into equation (5.21) to prove this statement. As we did with the APLL, the steady-state error of the DPLL will be determined for input signals with variations such as

phase step, frequency step (phase ramp), and frequency ramp (phase parabola). From (Lindsey et al 1981 [18]), the phase step, phase ramp and phase parabola are expressed in the $z-$domain as

$$Phase\ Step = \theta_0 \frac{z}{z-1}$$

$$Phase\ Ramp = \Omega_0 T \frac{z}{(z-1)^2}$$

$$Phase\ Parabola = \frac{\Omega_1 T}{2} \frac{z(z+1)}{(z-1)^3}$$

Utilizing the notation presented in chapter 3 for the magnitude of each signal, the above equations can be rewritten as

$$Phase\ Step = \Delta\Phi \frac{z}{z-1} \qquad (5.22),$$

$$Phase\ Ramp = \Delta\omega T \frac{z}{(z-1)^2} \qquad (5.23),$$

$$Phase\ Parabola = \frac{\Delta\dot{\omega} T^2}{2} \frac{z(z+1)}{(z-1)^3} \qquad (5.24),$$

where, $\Delta\Phi$, $\Delta\omega$, and $\Delta\dot{\omega}$ represent the magnitude of the phase step, phase ramp, and phase parabola, respectively.

Having the excitation signals expressed in the $z-$domain, we are able to proceed with the steady-state analysis. When the input signal has a phase variation of the form of equation (5.22), the resulting steady state error is

Phase Step:

$$e_z = \lim_{z\to 1}\left(\frac{z-1}{z}\right) \cdot \frac{(z-1)^2}{(z-1)^2 + K_d K_o T(b_o z + b_1)} \cdot \Delta\Phi \frac{z}{z-1},$$

$$e_z = \lim_{z\to 1} \frac{\Delta\Phi(z-1)^2}{(z-1)^2 + K_d K_o T(b_o z + b_1)} = 0 .$$

This result agrees with the steady-state error obtained for its analog counterpart. Now the input signal $\theta_i(z)$ is a phase ramp, as given by equation (5.23)

Phase Ramp (Frequency Step):

$$e_z = \lim_{z\to 1}\left(\frac{z-1}{z}\right)\cdot\frac{(z-1)^2}{(z-1)^2 + K_d K_o T(b_o z + b_1)}\cdot \Delta\omega T\,\frac{z}{(z-1)^2},$$

$$e_z = \lim_{z\to 1} \frac{\Delta\omega T(z-1)}{(z-1)^2 + K_d K_o T(b_o z + b_1)} = 0 .$$

Once again, the results obtained agree with those obtained with the APLL. Finally, let us assume that the input signal is a phase parabola, as given by equation (5.24)

Phase Parabola (Frequency Ramp):

$$e_z = \lim_{z\to 1}\left(\frac{z-1}{z}\right)\cdot\frac{(z-1)^2}{(z-1)^2 + K_d K_o T(b_o z + b_1)}\cdot \frac{\Delta\dot{\omega} T^2}{2}\,\frac{z(z+1)}{(z-1)^3},$$

$$e_z = \lim_{z\to 1} \frac{\Delta\dot{\omega} T^2(z+1)}{(z-1)^2 + K_d K_o T(b_o z + b_1)} = \frac{\Delta\dot{\omega} T}{K_d K_o (b_o + b_1)} .$$

Substituting the definition for b_o and b_1 into the above result shows that the response of the DPLL to a phase parabola is identical to its analog counter part.

$$e_z = \frac{\Delta\dot{\omega} T}{K_d K_o (b_o + b_1)} = \frac{\Delta\dot{\omega}}{\frac{K_d K_o}{\tau_1}} = \frac{\Delta\dot{\omega}}{\omega_n^2} .$$

As we could see from the steady-state error analysis, the second-order DPLLs presented in this project have theoretically the same performance as the second-order APLL

using a PI loop filter. As with the APLL, it is the loop filter that determines the response of the system to these and any other types of signals that enter the system. Taking a look at the PI digital loop filter used and the definitions for b_o and b_1, where $b_o = \frac{2\tau_2 + T}{2\tau_1}$, and $b_1 = \frac{T - 2\tau_2}{2\tau_1}$, it is obvious that each of the DPLLs will only have the same performance as the APLL if we express b_o and b_1 with enough precision as to allow a noticeable difference in magnitude between them. This is more important when the system is operating at high frequencies, where the sampling period T is so small that it could be assumed to be negligible. The idea behind this thought is not to make the mistake of assuming that T is negligible, which would make $b_o = -b_1$. When this happens, the digital loop filter will behave like an all pass filter with a gain that equals or approximates $\frac{\tau_2}{\tau_1}$, making the DPLL behave like a first-order loop. This can be easily proved using the definitions for $F(z)$, b_o and b_1. If we assume that the sampling period is negligible or its effect is not effectively accounted for because we did not use enough precision to express b_o and b_1, then $b_o = \frac{\tau_2}{\tau_1}$ and $b_1 = \frac{\tau_2}{\tau_1}$. Applying these results to $F(z)$, which result in $b_o = -b_1$, we obtain

$$F(z) = \frac{b_o + b_1 z^{-1}}{1 - z^{-1}} = \frac{-b_1(1 - z^{-1})}{1 - z^{-1}} = -b_1 = \frac{\tau_2}{\tau_1}.$$

Substituting $F(z)$ into $H(z)$ we obtain $H(z) = \frac{\frac{K_D \tau_2}{\tau_1} z^{-1}}{1 + \frac{K_D \tau_2}{\tau_1} z^{-1}} = \frac{\frac{K_D \tau_2}{\tau_1}}{z + \frac{K_D \tau_2}{\tau_1}}$. The response of this system could be very well compared to a first-order PLL. This could be easily seen if we

calculate the steady-state error for this system as we did previously for the second-order loop. Using the same example shown above for a phase ramp input signal, we obtain that the steady-state error for this system is $e_z = \lim_{z\to 1}\left(\frac{z-1}{z}\right)\cdot\frac{(z-1)}{(z-1)+\frac{K_d K_o T\tau_2}{\tau_1}}\cdot\Delta\omega T\,\frac{z}{(z-1)^2}$

$e_z = \frac{\Delta\omega T}{\frac{K_d K_o T\tau_2}{\tau_1}} = \frac{\Delta\omega}{\frac{K_d K_o \tau_2}{\tau_1}} = \frac{\Delta\omega}{\tau_2\omega_n}$. Therefore, this result shows that, like a first-order APLL, this system will track a phase ramp with a steady-state error different from zero. In fact, the error is proportional to the magnitude of the phase ramp.

6 Fixed Point Arithmetic

Fixed-point binary representation uses binary digits to express and manipulate decimal integer and rational numbers as binary integer numbers. These numbers can be assumed to be signed or unsigned, and can be manipulated using logic and arithmetic operations. Manipulating fixed-point numbers could be easily attained using the hardware-assisted integer operations embedded in any microprocessor. This inherent quality of fixed-point arithmetic makes its implementation less expensive than floating-point; that is, in addition to requiring less processing time.

Adhering to the approach followed by (Yates [33]) and combined with examples and equations provided by (Labrosse 1998 [30]), the following sections will present some of the theory used to express and handle the four common binary representations known as unsigned integers, unsigned fixed-point rationals, signed two's complement integer and signed two's complement fixed-point rationals.

6.1 Unsigned Integer and Fixed-Point Rational

An $N-$bit binary word interpreted as an unsigned integer number can be expressed as $U_{a,0}(x)$; x is an $N-$bit binary number and $a = N$ for unsigned integers. The range of values that an $N-$bit unsigned integer number can take on is determined by $0 \le U_{a,0}(x) \le (2^N - 1)$. Expressing an unsigned integer number as $U_{a,0}(x)$ will be understandable when we present unsigned fixed-point rational numbers, and express

unsigned integer numbers as a special case of fixed-point rationals. The value of a particular $N-$bit binary number x interpreted as $U_{a,0}(x)$ can be obtained using this equation

$$U_{a,0}(x)=\sum_{n=0}^{N-1}2^{n}.x_{n} \qquad (6.1),$$

where x_n represents bit n of x and the dot between 2^n and x_n means multiplication. As an example, consider the eight-bit binary number $x=00010000=0_70_60_51_40_30_20_10_0$. The sub index added to each bit indicates the significance and order each bit has in the binary number. It emphasizes the fact that the theory provided assumes that the least significant bit, which has sub index 0, is located to the rightmost bit position; therefore, the order of significance increases from right to left. Interpreting this number as $U_{a,0}(x)$ results in

$$U_{a,0}(x)=\sum_{n=0}^{7}x_{n}=2^{0}.x_{0}+2^{1}.x_{1}+2^{2}.x_{2}+2^{3}.x_{3}+2^{4}.x_{4}+2^{5}.x_{5}+2^{6}.x_{6}+2^{7}.x_{7}$$

$$U_{a,0}(x)=\sum_{n=0}^{7}x_{n}=2^{0}.0+2^{1}.0+2^{2}.0+2^{3}.0+2^{4}.1+2^{5}.0+2^{6}.0+2^{7}.0=16$$

Using this same approach, we can present the theory related to unsigned fixed-point rationals.

An $N-$bit binary number x interpreted as an unsigned fixed-point rational can be expressed as $U_{a,b}(x)$, where a is the number of integer bits and b is the number of fractional bits used to represent the fixed-point number. a and b follow this relation

$a = N - b$. The value of a particular $N-$bit binary number x interpreted as $U_{a,b}(x)$ can be obtained using this equation

$$U_{a,b}(x) = 2^{-b}\sum_{n=0}^{N-1} 2^n.x_n \qquad (6.2)$$

Note that when $b = 0$ equation (6.2) becomes (6.1). For this reason we assume that unsigned integer numbers are a special case of unsigned fixed-point rationals. The range of values that an unsigned fixed-point rational number can take on are determined by $0 \le U_{a,b}(x) \le (2^a - 2^{-b})$, where $2^a - 2^b = 2^{-b}(2^N - 1)$. As an example, let us use the same 8-bit binary number we used previously. This time, the 8-bit binary number will be interpreted as an unsigned fixed-point rational that has the radix point between sub index locations 3 and 2: $x = 00010.000 = 0_7 0_6 0_5 1_4 0_3.0_2 0_1 0_0$. For this example, $N = 8$, $b = 3$ and $a = 8 - 3 = 5$, hence the range of values $U_{5,3}(x)$ can take on are: $0 \le U_{5,3}(x) \le (2^5 - 2^{-3})$. So the values range from 0 to $32 - 1/8 = 31.875$. Using equation (6.2) we obtain the value for the specific example as $U_{5,3}(x) = 2^{-3}\sum_{n=0}^{7} 2^n.x_n = 2^{-3}.2^4.1 = 2^{-3}.16 = 2.000$.

6.2 Signed two's complement Integers and Fixed-Point Rationals

An $N-$bit binary word, interpreted as a signed two's complement integer number, can be expressed as $A_{a,0}(x)$, where $a = N - 1$. Note that unlike unsigned integers, where the number of bits representing an integer number was equal to N ($a = N$), signed two's complement integers only use $N - 1$ bits to represent the integer

part of the number. This occurs because the most significant bit, the leftmost bit, represents the sign of the number. The value of a particular $N-$bit binary number x interpreted as $A_{a,0}(x)$ can be obtained using this equation

$$A_{a,0}(x) = -2^{N-1}x_{n-1} + \sum_{n=0}^{N-2} 2^n.x_n \tag{6.3}$$

The range of values that a signed two's complement integer number can take on are $-2^{N-1} \le A_{a,0}(x) \le (2^{N-1}-1)$.

When the $N-$bit binary word is interpreted as a signed two's complement fixed-point rational, it is expressed as $A_{a,b}(x)$. From this expression we can see that signed two's complement integers can be considered a special case of signed two's complement fixed-point rationals, as occurred with unsigned integers and unsigned fixed-point rationals. To see the relation, all we have to do is equal b to zero, and $A_{a,b}(x) = A_{a,0}(x)$. To calculate the value of a particular $N-$bit binary number, we use equation

$$A_{a,b}(x) = 2^{-b}\left[-2^{N-1}x_{n-1} + \sum_{n=0}^{N-2} 2^n.x_n\right] \tag{6.4}$$

Equation (6.4) shows that to obtain the expression for signed two's complement fixed-point numbers, we multiply equation (6.3) by the scaling factor 2^{-b}. The range of numbers that can be represented by an $N-$bit binary are obtained with this relation: $-2^{N-1-b} \le A_{a,b}(x) \le (2^{N-1-b} - 2^{-b})$. For this representation, variables a, b, and N are related by $a = N-b-1$. Examples are provided in the sections below, where we combine the representation given by (Labrosse 1998 [30]) and (Yates [33]).

6.3 Logic and Arithmetic Fixed-Point Operations

In the next sections we will examine basic logic and arithmetic operations that can be applied to signed or unsigned fixed-point rational numbers. The operations we will cover are addition, multiplication, word-length reduction, and shifting. Before moving on to next section, we should combine the notations used by (Labrosse 1998 [30]) and (Yates [33]) to express signed and un-signed fixed-point numbers. On his notation, (Labrosse 1998 [30]) uses the concept of the mantissa, which essentially consists of the signed or unsigned integer value of an $N-$bit binary word. The mantissa, expressed as m, can be interpreted as $U_{a,0}(x)$ or $A_{a,0}(x)$. Then, to obtain a fixed-point rational representation, we multiply by the scaling factor, 2^{-b}. So, we expresses fixed-point numbers, whether signed or unsigned, as $fixed-point =< mantissa > S < exponent >$ or, using the changes added, $fixed-point =< m > S < -b >$. Combining this notation with that provided by (Yates [33]), we obtain

$$X_{a,b}(x) =< m > S < -b > \tag{6.5}$$

where $X_{a,b}(x)$ could represent $U_{a,b}(x)$ or $A_{a,b}(x)$; proper substitution of these functions can be done as we know what type of number we are working with. This notation will prove to be useful as we work with logic and arithmetic operations.

6.3.1 Shifting and Word length Reduction

Before introducing the basic arithmetic operations that apply to fixed-point numbers, it is important to cover the shifting and word length reduction operations first.

Although the shift operation could be defined as logical or arithmetic, depending on the implementation, we will adhere to the arithmetic definition used to handle fixed-point numbers. Shift operations manipulate binary words at the bit level by moving all the bits of the binary word to the left or the right. A shift is executed either to the left or to the right by a positive integer number of bits k. An arithmetic shift operation can be employed to divide or multiply a fixed-point number by a power of two, thereby modifying its scaling factor. When a fixed-point number is scaled up by 2^k, we are multiplying the number by 2^k, which is equivalent to a shift left operation (k bits to the left). This operation can be denoted as

$$SL_k[X_{a,b}(x)] = X_{(a-k,b+k)}(x) \qquad (6.6).$$

On the other hand, if we scale down the fixed-point number by 2^k, we are dividing the number by 2^k, which implies a shift right operation (k bits to the right). This operation can be denoted as

$$SR_k[X_{a,b}(x)] = X_{(a+k,b-k)}(x) \qquad (6.7).$$

Word length reduction implies the extraction of either the n most significant bits or the n least significant bits from an $N-$ bit binary word. The extracted bits will form a new binary word with length equal to n; to keep consistency of variables, we can then express n as N. This can be done because after the extraction, we will only work with the new extracted binary word. Following the notation used by (Yates [33]), the extraction of the n most significant bits of $X_{a,b}(x)$ can be denoted as

$$HI_n[X_{a,b}(x)] = X_{(a,b-n)}(x) \qquad (6.8)$$

The extraction of the n least significant bits of $X_{a,b}(x)$ can be denoted as

$$LO_n[X_{a,b}(x)] = X_{(a-n,b)}(x) \tag{6.9}$$

Note: For all these operations to work, both k and n must be much smaller than N.

6.3.2 Addition

Two binary numbers, interpreted as signed or unsigned fixed-point rationals, can only be added if they are both signed or unsigned and have the same word length and scaling. That is, two numbers $Y_{a,b}(x)$ and $W_{c,d}(x)$ can only be added if $a = c$ and $b = d$; this also implies that both numbers have the same number of bits, hence $N_Y = N_W$. Since $a = c$ and $b = d$, the result of the addition would be

$$X_{a+1,b}(x) = Y_{a,b}(x) + W_{a,b}(x) \tag{6.10}$$

Equation (6.10) implies that adding two $N-$bit numbers requires $(N+1)$ bits to express the result.

6.3.3 Multiplication

The rules that apply to multiplication of two fixed-point numbers are not as strict as those that apply to addition. The only requirement that must be followed is that both numbers be signed or unsigned. When multiplying two unsigned fixed-point numbers $Y_{a,b}(x)$ and $W_{c,d}(x)$, each having N_Y and N_W number of bits, the result would be

$$U_{(a+c,b+d)}(x) = Y_{a,b}(x) \times W_{c,d}(x) \tag{6.11}$$

When multiplying two signed fixed-point numbers $Y_{a,b}(x)$ and $W_{c,d}(x)$, the resulting number would be

$$A_{(a+c+1,b+d)}(x) = Y_{a,b}(x) \times W_{c,d}(x) \tag{6.12}$$

Both equations basically state that the number of bits used to represent the resulting number equals the addition of N_Y and N_W, the size in bits of each fixed-point number. In other words, $N_X = N_Y + N_W$, where X represents U or A.

6.4 Examples

To better understand the theory provided in the previous sections of these chapter, let us discuss two examples. All numbers used in these examples will $N-$bit numbers interpreted as signed fixed-point rationals and will be expressed using the format $X_{a,b}(x) =< m > S < -b >$.

***Example 1*:** Addition of 16-bit signed numbers $\mathrm{A1}_{(0,15)}(\mathrm{x}) = 20480\mathrm{S}\text{-}15$ (0.625) and $A2_{(-3,18)}(x) = 31745S - 18$ (0.1211). Express the result as a 16-bit number.

Although these two numbers have the same number of bits, they cannot be added because their exponents do not have the same order of magnitude; $A1$ has an exponent equal to $S-15$ and $A2$ has an exponent equal to $S-18$. To be able to add these numbers, we must convert the number with smaller exponent to the same order of magnitude as the other. To do this we must divide $A2_{(-3,18)}(x)$ by $2^3 s - 3 = 2^3 \times 2^{-3}$. In other words, we divide the mantissa by 8 and add 3 to the exponent. Note that all we did

was to divide $A2_{(-3,18)}(x)$ by 1, as $2^3 \times 2^{-3} = 1$. Another way of obtaining the same result is to shift $A2_{(-3,18)}(x)$ three bits to the right. Using either process, the result would be $A2_{(0,15)}(x) = 3968S - 15$. Please, note that if you divide 31745 by 8, the result is 3968.125, however, we truncate the number and keep only its integer part without rounding it.

Adding $A1$ and $A2$ we obtain $A3_{(1,15)}(x) = 24376S - 15$. We know that $a = N - b - 1$ for a signed number. Using this relation we get that $N = a + b + 1$. So, for this example, $N = 1 + 15 + 1 = 17$. This extra bit that results from the addition is known as the carry bit. This bit can be depreciated if the result can still be represented as a 16-bit number. To verify that the result can be represented as a 16-bit number, we must determine the range of a 16-bit signed fixed-point rational number. The range is obtained using $-2^{N-1-b} \le A_{a,b}(x) \le (2^{N-1-b} - 2^{-b})$. But since we have expressed the number in the format given by equation (6.5), it would be easier to use the range for signed integers and compare the result to the mantissa. In this case we have that $-2^{N-1} \le A_{a,0}(x) \le (2^{N-1} - 1)$, gives a range of $-32768 \le A_{a,0}(x) \le 32767$. Comparing this result to the mantissa of $A3$, we confirm that this number can be represented as a 16-bit number; $32767 \ge 24376$. So, all we have to do is to depreciate the value of the carrier and keep the remaining 16 bits. The final result can be expressed as $A_{(0,15)}(x) = 24376S - 15;\ N = 0 + 15 + 1 = 16$.

***Example 2*:** Multiplication of 16-bit signed numbers $A1_{(0,15)}(x) = 20480S\text{-}15$ (0.625) and $A2_{(-3,18)}(x) = 31745S - 18$ (0.1211). Express the result as a 16-bit number.

To multiply these two numbers, just multiply their mantissas and add their exponents. The result would be $A3_{(-2,35)}(x) = 650137600S - 35$. Since the number of bits representing the result is equal to the sum of the number of bits in each number, $N_{A3} = N_{A1} + N_{A2}$, $A3$ is a 32-bit number. From example 1 we know that the highest positive number that could be represented by a 16-bit number is 32767. Looking at the mantissa, we realize that we still have to do some processing to express $A3$ as a 16-bit number.

Since we want the result to be a 16-bit number, we have to divide $A3$ by a number α. To make sure that the mantissa of the result fits the 16-bit range for signed integer numbers, we must make α equal to $2^{15} \times 2^{-15}$. As you can see, we are still dividing by one, so we are not changing the actual value of the fixed-point number. The result of the division is $A3_{(13,20)}(x) = 19840S - 20$. See that the mantissa of the result does fit the 16-bit range for a signed integer number. Now all we have to do is extract the 16 least significant bits of $A3_{(13,20)}(x)$. The result is

$$LO_{16}[A_{(13,20)}(x)] = A_{(-3,20)}(x) = 19840S - 20.$$

7 Designing the Analog Costas Loop

Designing an analog Costas Loop can be attained by combining its theory and operating principles with those of an APLL. We have previously shown that the Costas Loop can be linearized and modeled as an APLL if we assume that it is operating in a locked state or it is close to lock, and the bandwidth of the arm filters is wide enough, when compared to the bandwidth of the Loop filter. When the Costas Loop was initially introduced by John P. Costas, its main purpose was the demodulation of double-sideband suppressed carrier AM signals. Although technically the Costas Loop will be operating as if it was tracking and demodulating a double-sideband suppressed carrier AM signal, in this project it is actually going to be demodulating a Binary Phase Shift Keying signal (BPSK). This signal will have these characteristics:

Carrier Frequency $f_{Carrier} = 7MHz$,

Data Rate $DR = 300KHz = 300Kbps$,

Amplitude $A_i = 1V$,

Bandwidth $BW = 2DR = 600KHz$.

Starting with this data we can proceed with the design of a Second-Order Analog Costas Loop with PI Loop Filter.

Overall, there are some parameters and equations that characterize an APLL and, consequently the Costas Loop. These parameters could be summarized as damping ratio, natural frequency, bandwidth, noise bandwidth, dynamic operation, phase detector and VCO gains, and loop filter time constants.

While establishing the theory behind analog phase-locked loops and Costas Loops, many authors do not make it easy to understand the relation between the loop gain terms K_o and K_d, their meaning and actual use in the design process. In my opinion, the relation that should be explained better is that between K_d, the phase detector gain, and the magnitude of the input and output signals of the Costas Loop, A_i and A_o. To avoid this problem, these relations were defined in chapters three and four for the analog PLL and Costas Loop, respectively. For the analog Costas Loop, this relation was defined in chapter 4 as $K_d = \frac{A_i^2 A_o^2}{4}$. Since the amplitude of the input signal has already been established to be 1V (throughout the discussion of this material we will not be talking about the unit of volts when referring to amplitudes, instead we will just refer to it as a plain magnitude value; but the units will always be assumed to be volts), we can start by defining the gain of the phase detector in the linearized system and calculating the magnitude of the output signal. Setting the gain of the phase detector to $K_d = 1$, we obtain that the magnitude of the output signal should be $A_o = \frac{2}{A_i}\sqrt{K_d} = 2$. The next gain term to consider would be the gain of the VCO, K_o. The selection of K_o is very important as it controls the range of frequencies the VCO can reach, based on its input signal $u_f(t)$. This gain term, along with the phase detector gain, compose the overall gain of the analog system, $K = K_d K_o$. It is this gain parameter that controls various characteristics of the system, such as lock-in range and steady-state error. For this specific case we will set the gain of the VCO to $K_o = 100K$; then the system gain would be $K = 100K$. Having established the gain of the system, we have to select a

suitable value for its damping ratio ς. From chapter 3 we know that in order for a second-order system to achieve a fast and smooth frequency response, its damping ratio should be we 0.7, so we will fix this value to $\varsigma = 0.7$. Another parameter to establish is the natural frequency of the system.

Although this project does not cover the analysis of noise or optimizations for the analog Costas Loop or PLL, we will be using the noise bandwidth equation to establish the relation between the natural frequency of the system and its noise bandwidth. (Best 1999 [1]), (Gardner 1979 [2]), (Egan 1998 [3]), (Wolaver 1991 [4]), (Stensby 1997 [6]), and (Lindsey et al 1991 [8]) define the noise bandwidth as $B_L = \frac{\omega_n}{2}\left(\zeta + \frac{1}{4\zeta}\right)$. Based on the defined value for the damping ratio, we have that $B_L = 0.5286\,\omega_n$. Using this relation, we can calculate any of the two values, as long as one of them is fixed. Fixing the natural frequency to $\omega_n = 500\,rads/\sec$, we calculate the noise bandwidth for the system to be $B_L = 264.26\,Hz$. Although we chose a specific value for the natural frequency to then calculate the respective noise bandwidth, we could have definitely followed another path in which the noise bandwidth was previously defined based on the application and design constraints. For many applications, the noise bandwidth should be very narrow. In such case, we can establish the desired noise bandwidth first and then calculate the natural frequency. Each of the calculated parameters: system gains, damping ratio, natural frequency and noise bandwidth, are related to the linearized closed loop transfer function of the Costas Loop. Now we have to use those values to calculate the parameters that define the loop filter.

The loop filter used in this project is a PI-type loop filter. This filter was chosen because it provides the best response for the system. The parameters that define this filter are time constants τ_1 and τ_2. Using the closed loop transfer function given by equation 3.34, we can have the following relation: $2\varsigma\omega_n = \frac{K_d K_o \tau_2}{\tau_1}$ and $\omega_n^2 = \frac{K_d K_o}{\tau_1}$, which can be used to calculate τ_1 and τ_2. However, before using these two equations, we have to make sure that the results comply with the relation $\tau_1 = a\tau_2$, where $a > 2$. This relation was obtained by calculating the $3-dB$ frequency of the loop filter: $\omega_{LF\_3-dB} = \frac{1}{\sqrt{0.25\tau_1^2 - \tau_2^2}}$. In order for $\sqrt{0.25\tau_1^2 - \tau_2^2} > 0$, it turns out that $\tau_1 > 2\tau_2$, which results in $a > 2$; otherwise, the 3-dB frequency would have been an imaginary value. (Gardner 1979 [2]) suggests that τ_1 and τ_2 can have any relation, although it advises that $\tau_1 >> \tau_2$. However, on simulations run using system view, the analog Costas Loop does not lock to the input signal if $\tau_1 > 2\tau_2$. Nevertheless, it is possible that it just takes too long for the system to lock and the limited resources on the computer used are not enough to show a lock-in result. From $2\varsigma\omega_n = \frac{K_d K_o \tau_2}{\tau_1}$ and $\omega_n^2 = \frac{K_d K_o}{\tau_1}$ we obtain that $a = \frac{K_d K_o}{2\zeta\omega_n}$. Substituting the respective value for each variable we get $a = \frac{100{,}000}{2(0.7)(500)} = 142.86$.

Combining the above equations for $2\varsigma\omega_n$ and ω_n^2 we derive an equation for τ_2, which is $\tau_2 = \frac{2\zeta}{\omega_n} = \frac{2(0.7)}{500} = 0.0028$. Having this result we can use the relation $\tau_1 = a\tau_2$ to

obtain $\tau_1 = 142.86\ \tau_2 = 0.4$. We could have also used $\tau_1 = \frac{K_d K_o}{\omega_n^2}$, however, it is important to verify that the relation $\tau_1 > 2\tau_2$ always holds. For this system, the linearized closed-loop transfer function would be $H(s) = \frac{700s + 250e3}{s^2 + 700s + 250e3}$.

Analog PLL theory emphasizes that the system gain K, can be varied and usually be given a very large value to ensure proper system response. The system gain can be varied as required by the design. Nevertheless, whatever its value is, it will affect time constant τ_1. We know that with the addition of the loop filter, we are able to control the system gain for overall performance result, whereas keeping a desired system bandwidth. All this is achieved through time constants τ_1 and τ_2 in the loop filter. Since $\tau_2 = \frac{2\zeta}{\omega_n}$, it is a and subsequently τ_1, which alter the loop filter response to compensate for the change in the system gain; thereby, keeping the same overall system bandwidth. Having determined the parameters of the PI-type loop filter, we now proceed to the design of the arm filters.

The arm filters process signals $u_{d1}(t)$ and $u_{d2}(t)$, which are obtained after the input signal $u_i(t)$ is multiplied by the in-phase and quadrature-phase output signals $u_{o1}(t)$ and $u_{o2}(t)$. $u_{d1}(t)$ and $u_{d2}(t)$ are the resulting signal on each arm of the system and each signal has two frequency components: a DC or base-band component and a component at twice the input frequency. It is the base-band component that we will use to synchronize the system to and obtain the data from. In order to extract these base-band components, we have to design the arm filters to be low-pass and to have a bandwidth equal to or greater than the base-band

signal, but smaller than $2f_{Carrier} - DR$. Since the base-band signal has bandwidth equal to DR (the bandwidth we are referring to is the single-side bandwidth), the bandwidth of the arm filters will be equal to DR. Please, note that DR and $f_{Carrier}$ are given in Hertz. We will express these two frequency terms in Hertz for simplicity, since the data rate can be expressed in Hertz, instead of bits per second.

Following the bandwidth requirement, we design the arm filters to have a 3-db bandwidth of $DR = 300KHz$ and a gain of 0-dB in that bandwidth. In addition to this requirement, we will design the arm filters to be of the Bessel type. Since the phase response of a Bessel filter remains constant at zero degrees inside the designed bandwidth, it does not introduce phase variations to the signals that could affect the response of the system. Have to point out, however, that other type of filters can be used as well. The order of the filter would depend on the designer and the application, but it is common to use a first-order filter, which is usually more than enough to get the desired results. This statement about the order of the filter would become important when designing the digital system, reducing complexity and computation time.

8 Simulation Results: Analog Costas Loop Implementation

The results of the design will be provided using simulation software SystemView and Matlab. Using SystemView we will provide simulation results of the operation of the analog Costas Loop, and Matlab will be used to show the Bode plot diagrams for the system. To obtain this bode plot, a simple Matlab program was required. This program is included in appendix D. Apendix A shows a block diagram of the design in SystemView. The Costas Loop will be demodulating a BPSK signal with a data rate of 300KHz and a carrier frequency of 7MHz. The arm filters will be of the Bessel type and will have a bandwidth of 300KHz; we chose a single pole for this filter.

Based on the linear model derived for the analog Costas Loop, the design implementation done in this project is a second order system. This type of system is always stable, hence we will not cover stability analysis. We can start providing the results of the Bode Plot diagram. From equation (3.36), the frequency at which the 3-dB bandwidth of the system occurs is $\omega_{3-dB} = 2.06\omega_n$. Since the natural frequency of the system $\omega_n = 500 rads/s$ and $\varsigma = 0.7$, $\omega_{3-dB} = 1030 rads/s$.

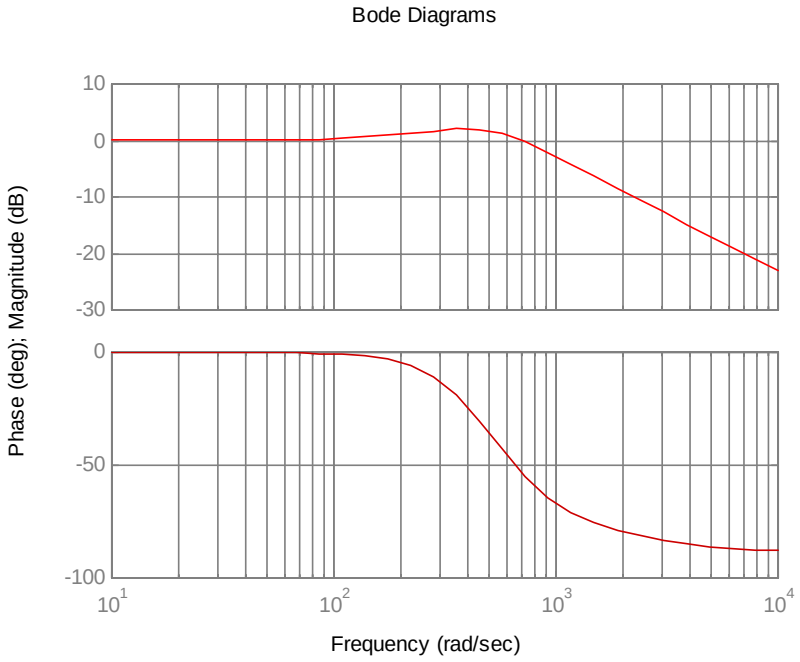

Figure 8-1. Bode Plot Diagrams

To obtain the magnitude and phase Bode Plot diagrams, we provided the parameters determined for the system transfer function to a Matlab program. The Bode diagram shows that the 3-dB frequency of the system is around 1000 rads/s, which properly relates to the results obtained mathematically.

The simulation results obtained with SystemView will be provided graphically. These results are divided in four groups, based on the phase and frequency offset assumed for the carrier f_i of the input signal on each simulation. The phase and frequency offset combination for each group is shown on the table below.

Phase Offset (Degrees)	Frequency Offset (Hertz)
0	0
45	0
0	100
0	700

Table 8-1. Phase and Frequency Offset Combinations

Each of the simulation results include graphs for signals $u_Q(t)$, $u_I(t)$ and $Ku_f(t)$. For the case of $u_f(t)$ we decided to show the results multiplied by the system gain, $K = 100e3$. The first two graphs, figures 8-2 and 8-3 show signals $u_Q(t)$ and $u_I(t)$ when f_i, the carrier frequency of the input signal has a phase and frequency offset equal to zero. As we can see from the figures, $u_Q(t)$ is quickly attenuated and settles to a value that is practically zero after 800 microseconds; Costas Loop theory indicates that $u_Q(t)$ must be zero when the system is locked. $u_I(t)$, obtained after the arm filter, contains the demodulated data. This signal settles quickly as well and the magnitude of the recovered NRZ data is definitely bounded by amplitudes 1 and –1; these are the pick to pick amplitude values for the NRZ data originally sent. The third graph is the output of the loop filter $u_f(t)$. It started with an under damped oscillation and quickly settled to zero after 800 microseconds, just as the other signals did.

The second simulation assumes a phase offset of 45° or $\frac{\pi}{4}$ rads, but the frequency offset remains unaltered at zero. We will be using degrees as the unit to refer to the phase offset because this is the unit used by SystemView to establish waveform parameters. The same applies for the frequency offset. SystemView uses Hertz as the unit to establish frequency parameters for the system, waveforms and other components, therefore this is the unit that we will be using in this report. All three figures show that the Costas Loop was able to track and lock quickly to the input signal as it did in the previous simulation. This time however, the output signal of the loop filter had a positive oscillation. The amplitude of this signal settled back to zero as expected, once the system was locked. The time required for this process still took around 800 microseconds.

The third simulation assumes a frequency offset of 100 Hz and zero phase offset. Figures 8-8, 8-9 and 8-19 show again that the Costas Loop was still able to track and lock to the input signal, only this time it took the system longer to settle. Approximately, it took the system 6ms to lock to this signal. It is important to notice that the settling value for $Ku_f(t)$ was in the vicinity of 100, showing the relation between $Ku_f(t)$ and the system response.

The fourth simulation assumes a frequency offset of 700Hz; the phase offset remains at zero degrees. It appears from figures 8-11, 8-12 and 8-13 that the system was not able to lock. At least that is the case in the time window covered by the simulation. However, we notice that the system is moving in the right direction to lock to the input signal. Although no graphical result is added, the system was able to lock to this signal after 16ms.

From PLL theory, the lock-in range of a second-order PLL using PI-type loop filter is approximately $2\zeta\omega_n$, as given by equation (3.40). This results in a lock-in frequency of 700rads/s or 111 Hz for this system. Simulations and comparisons done with a similar PLL using SystemView showed that both systems had very similar responses to phase and frequency changes in their respective input signal.

8.1 Simulation Results with phase and frequency offsets equal to zero

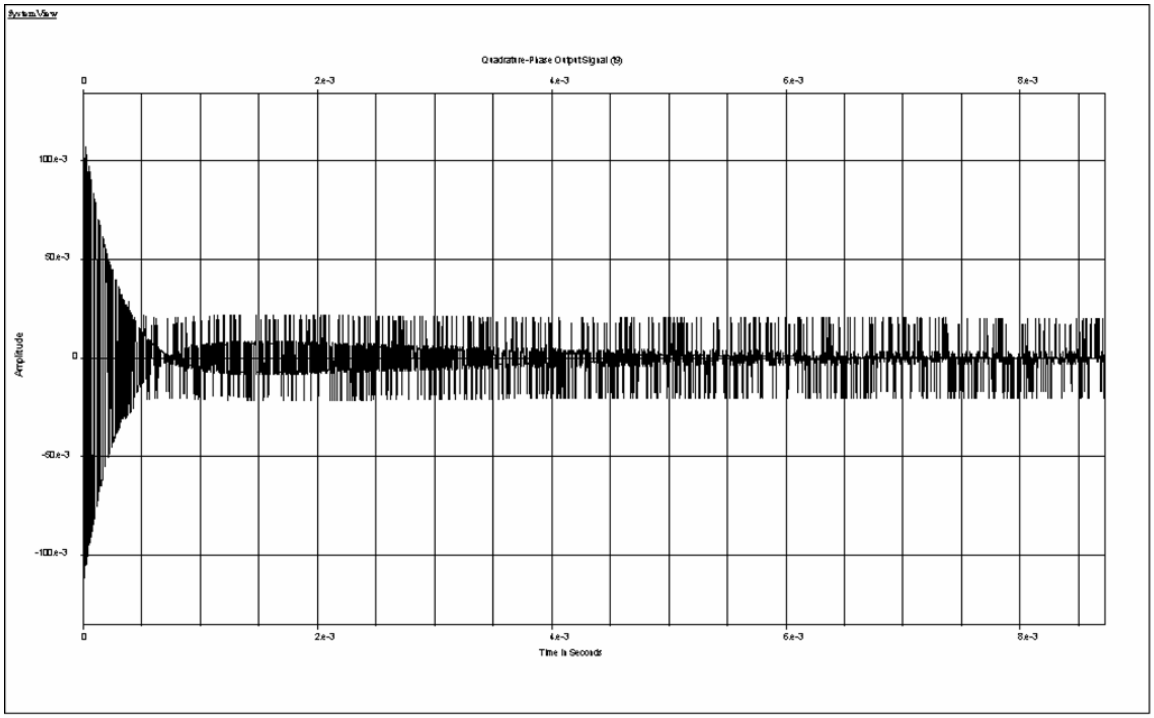

Figure 8-2. Quadrature-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and

the y-axis represents the amplitude at 50mV/Division.

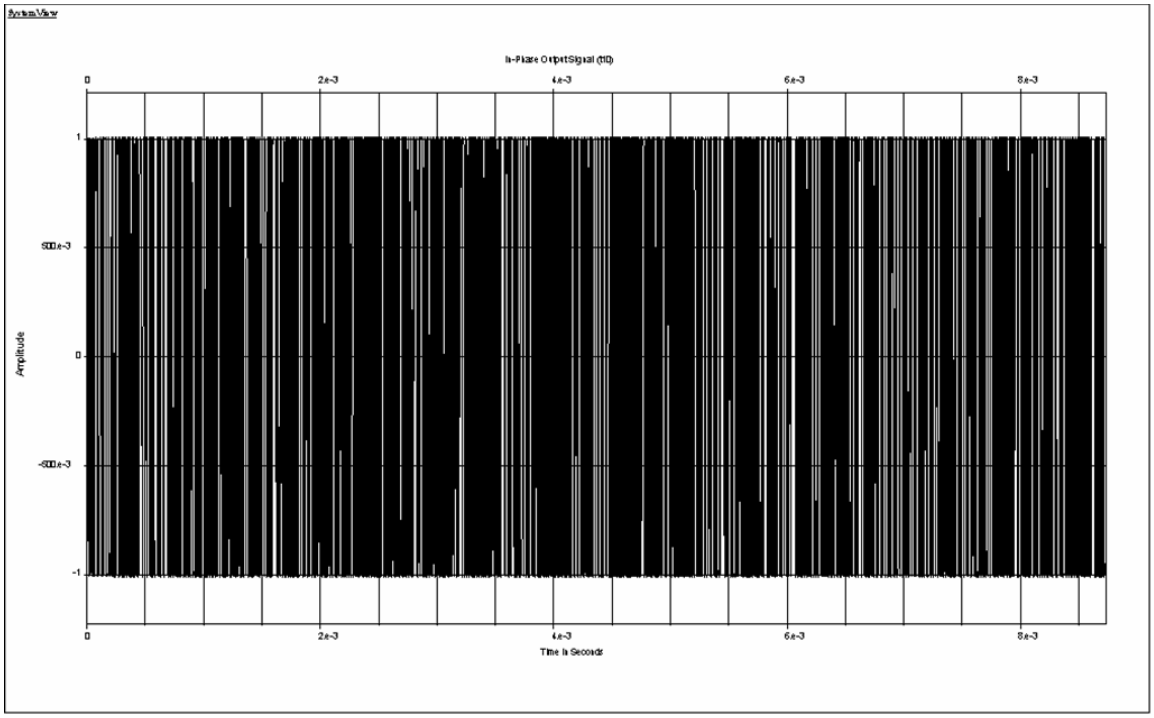

Figure 8-3. In-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 500mV/Division.

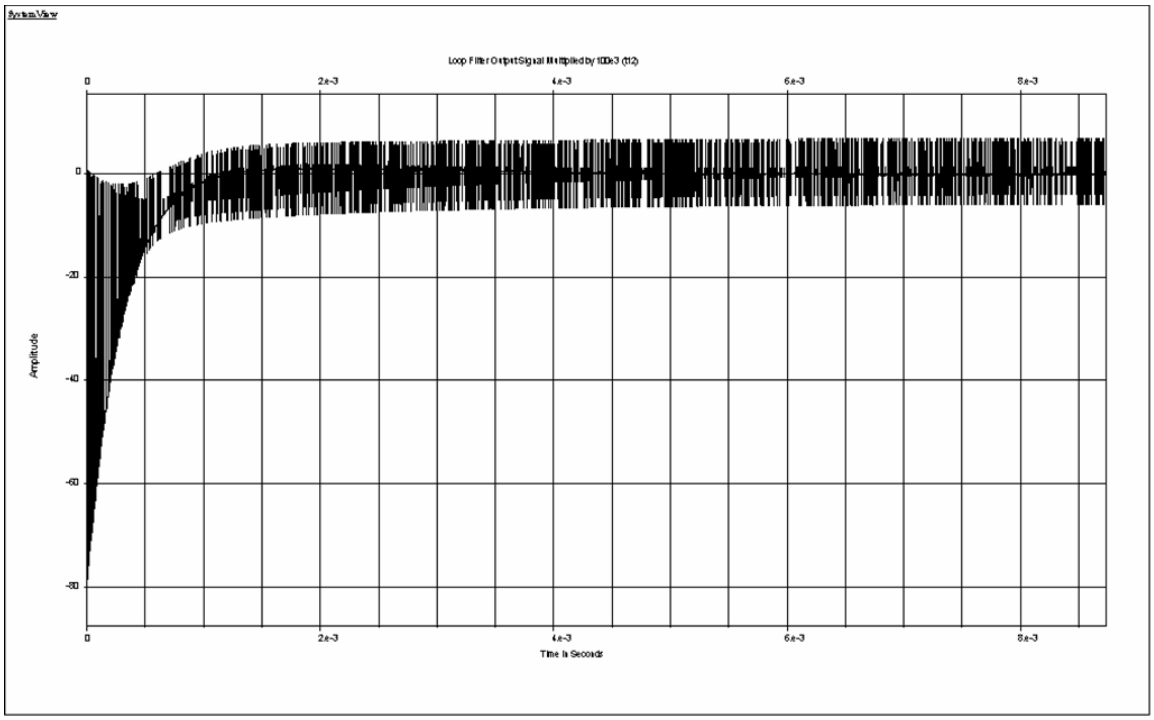

Figure 8-4. Loop Filter Output Signal Multiplied by K=100e3

In this graph, the x-axis represents time at 0.5ms/Division and

the y-axis represents the amplitude at 20Hz/Division.

(By multiplying the output of the loop filter by K we obtain a frequency measure.)

8.2 Simulation Results with phase offset equal to 45 degrees

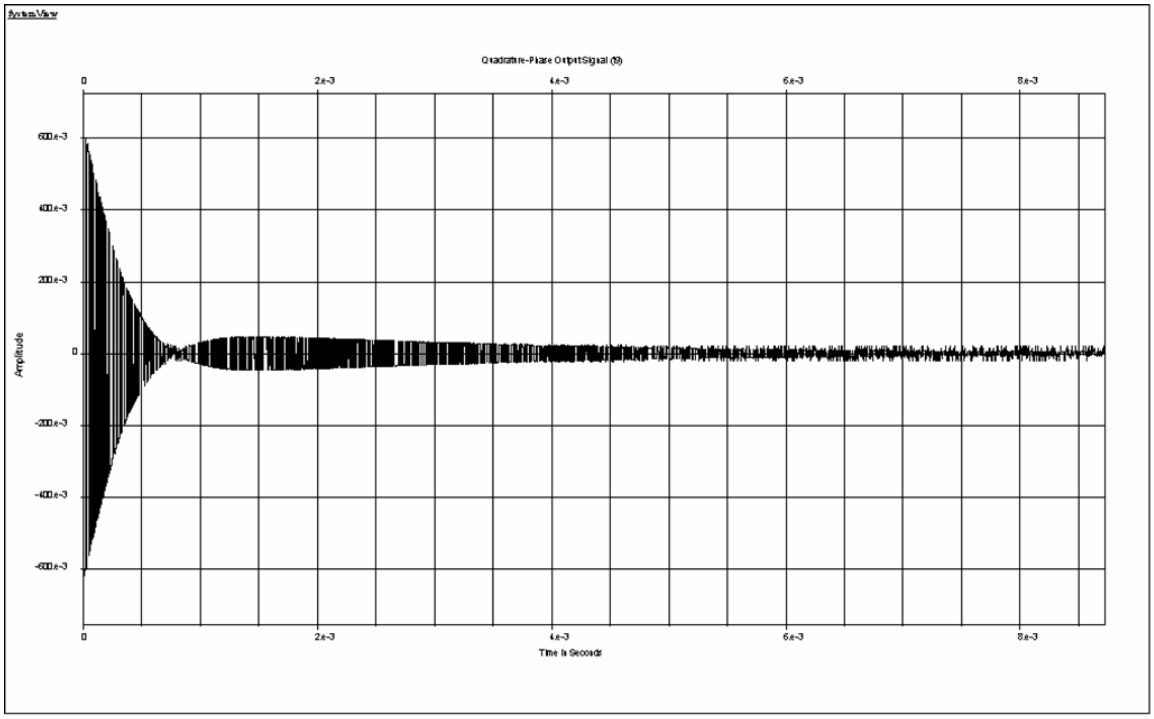

Figure 8-5. Quadrature-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 200mV/Division.

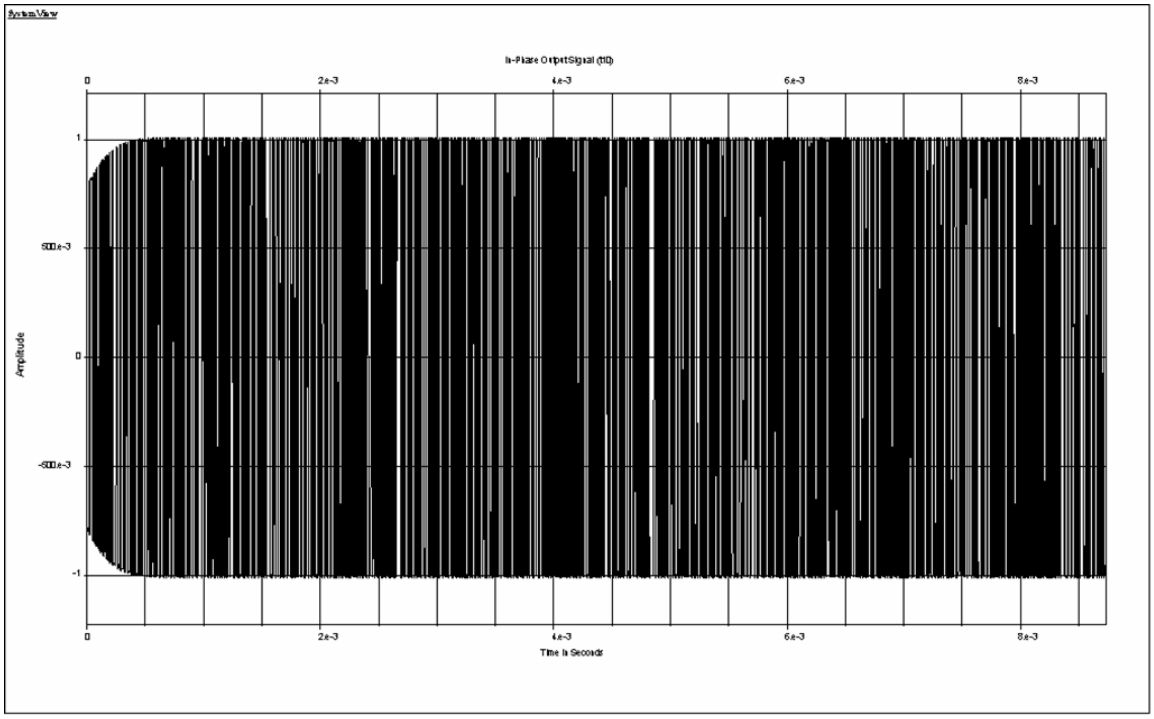

Figure 8-6. In-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 500mV/Division.

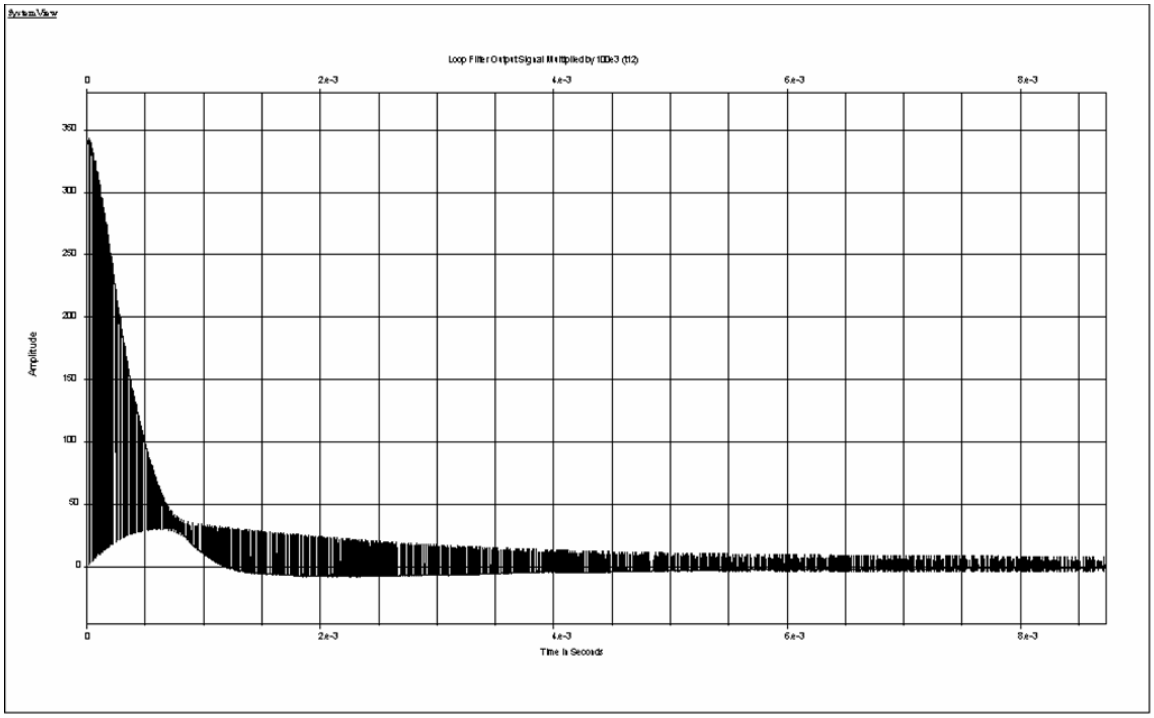

Figure 8-7. Loop Filter Output Signal Multiplied by K=100e3

In this graph, the x-axis represents time at 0.5ms/Division and

the y-axis represents the amplitude at 50Hz/Division.

(By multiplying the output of the loop filter by K we obtain a frequency measure.)

8.3 Simulation Results with frequency offset equal to 100Hz

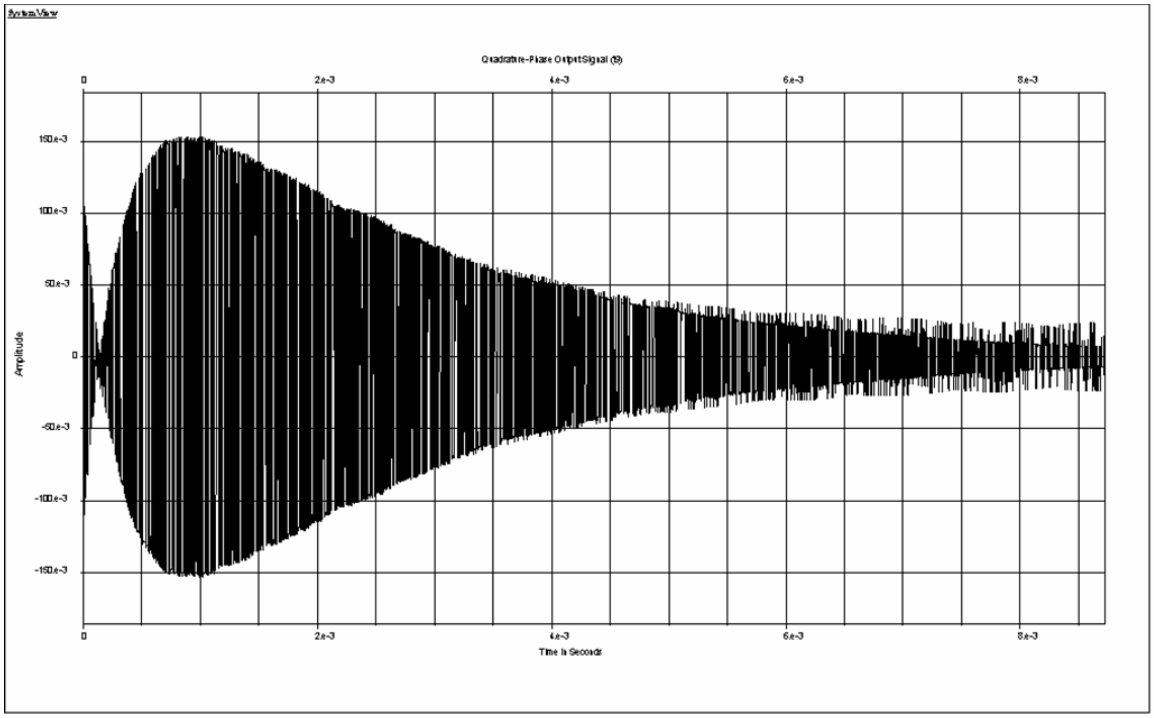

Figure 8-8. Quadrature-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 50mV/Division.

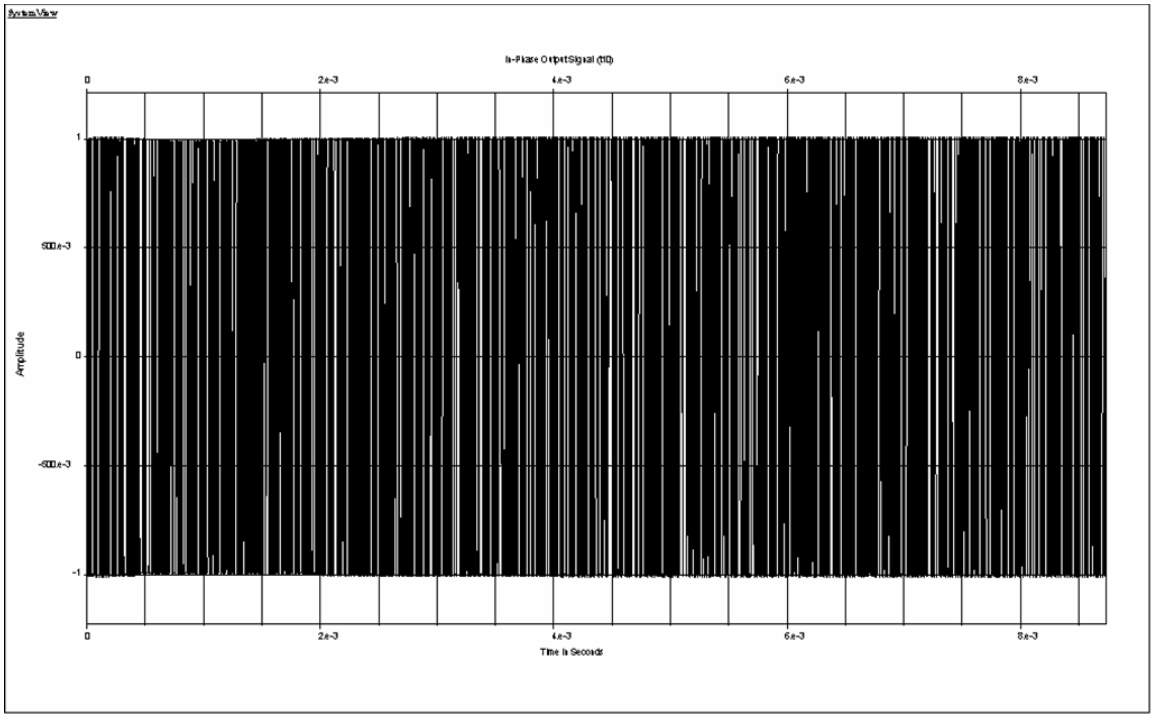

Figure 8-9. In-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 500mV/Division.

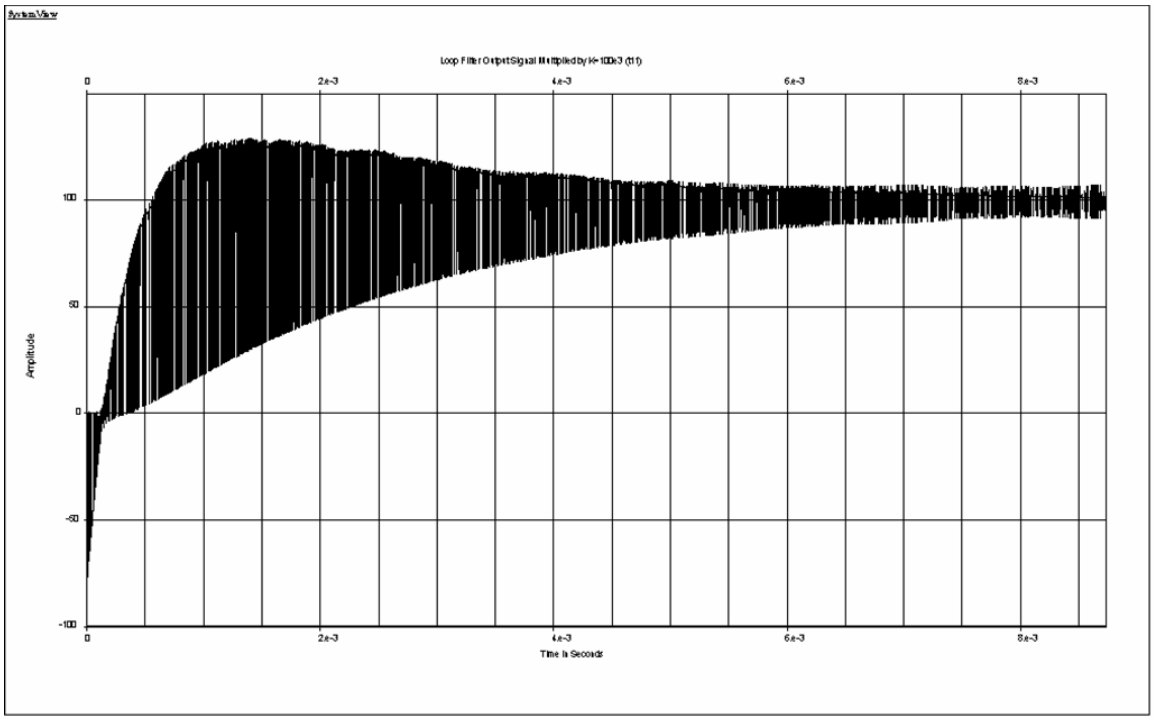

Figure 8-10. Loop Filter Output Signal Multiplied by K=100e3

In this graph, the x-axis represents time at 0.5ms/Division and

the y-axis represents the amplitude at 50Hz/Division.

(By multiplying the output of the loop filter by K we obtain a frequency measure.)

8.4 Simulation Results with frequency offset equal to 700Hz

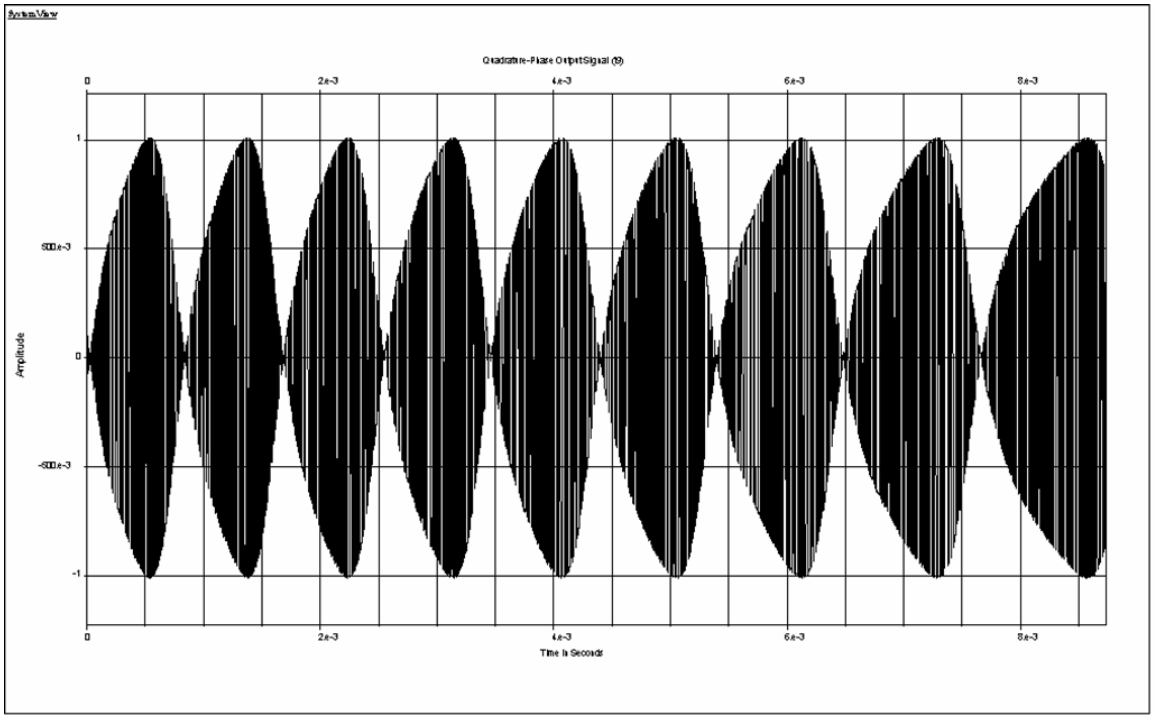

Figure 8-11. Quadrature-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 500mV/Division.

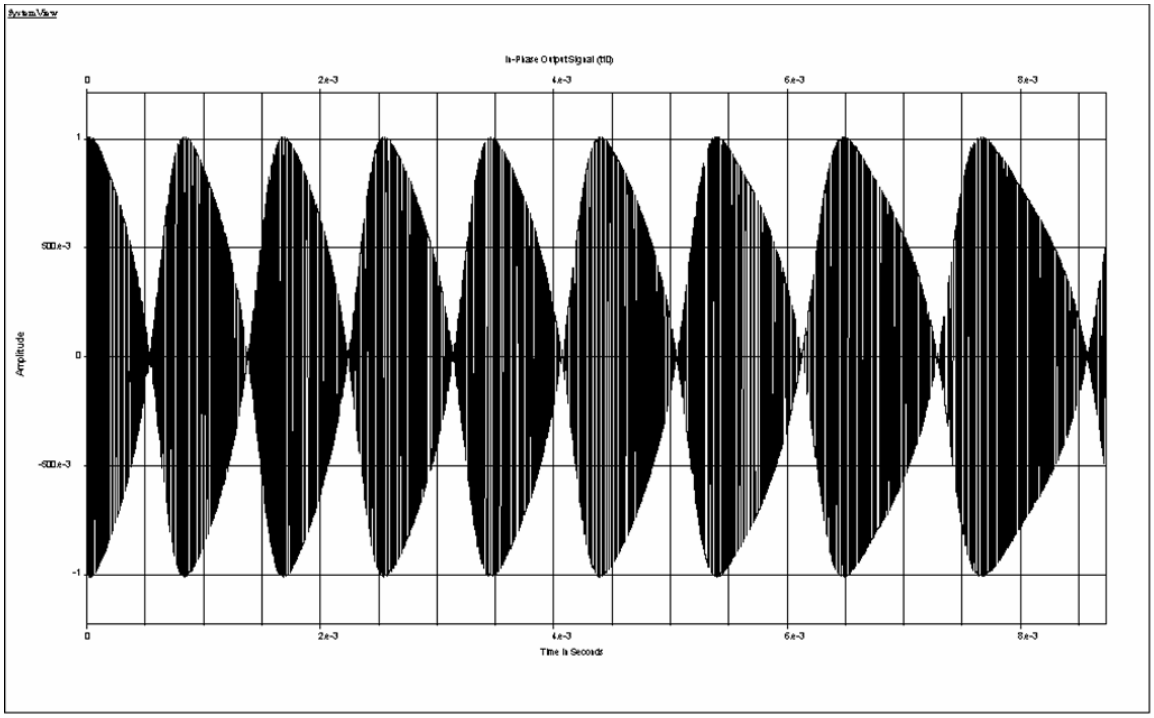

Figure 8-12. In-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 500mV/Division.

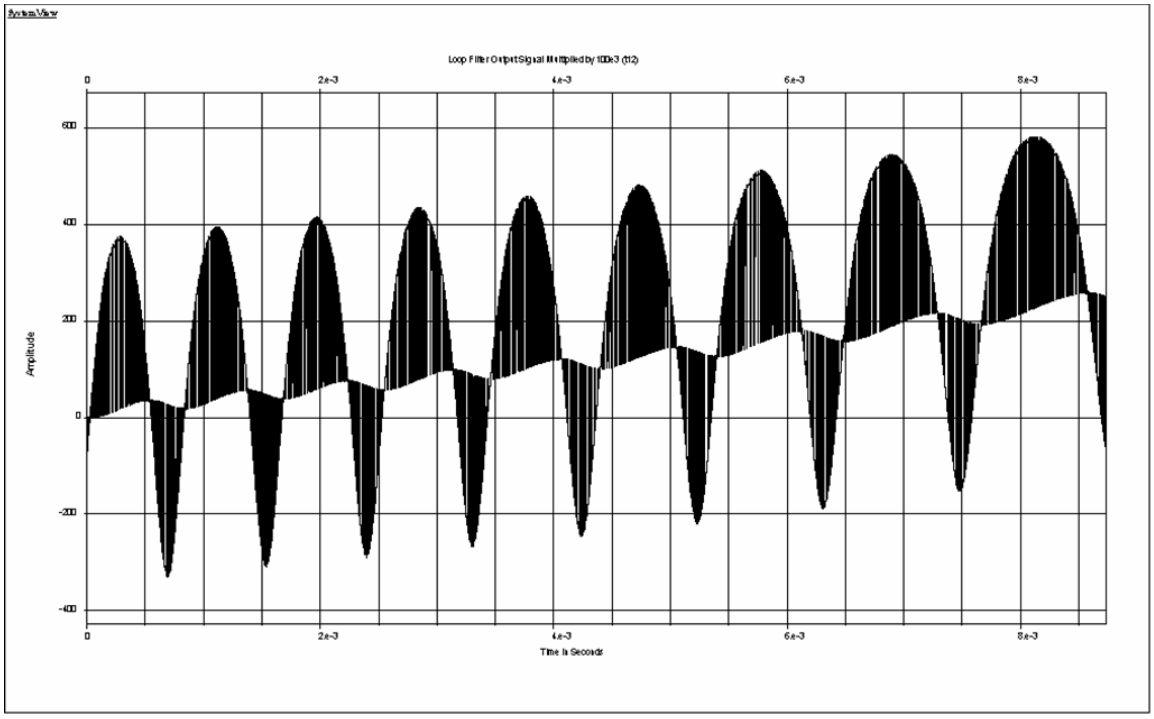

Figure 8-13. Loop Filter Output Signal Multiplied by K=100e3

In this graph, the x-axis represents time at 0.5ms/Division and

the y-axis represents the amplitude at 200Hz/Division.

(By multiplying the output of the loop filter by K we obtain a frequency measure.)

9 Designing the Digital Costas Loop Demodulator

9.1 Discrete-Time design

Design of the digital Costas Loop should be straightforward from the results obtained in chapter 7 for the analog Costas Loop. Chapter 5 shows the theory and equations that relate a second-order APLL with PI-type loop filter to two specific digital counterparts. This theory establishes a relation between the parameters of the analog phase-locked loop and that of the digital phase-locked loop system, allowing us to achieve a totally digital implementation with analog like characteristics and performance. In addition to relating the parameters of both systems, the equations that define the digital phase-locked loop also relate the characteristics and performance of the system to the selected sampling period. Therefore, before going any further, we should establish the criteria to determine the sampling period of the system. From the sampling theorem of discrete-time signal processing theory (Oppenheim et al 1989 [7]), we know that the sampling frequency (also considered the system frequency) f_s must be larger than or equal to $2f_{Max}$, where f_{Max} is the largest frequency in the signal or signals processed by the system. Consequently, in order to determine f_{Max} and subsequently f_s, we must determine the frequency range occupied by the signals in the analog system, the analog Costas Loop. For this analysis we will not use the linear model.

From Chapter 4 we obtain the conglomerate of signals processed by the analog Costas Loop that have a frequency content higher than DC level. For the readers convenience these signals are presented below.

$$u_i(t) = A_i m(t)\sin(\omega_i t + \theta_i)$$
$$u_{o1}(t) = A_o \sin(\omega_i t + \theta_o)$$
$$u_{o2}(t) = A_o \cos(\omega_i t + \theta_o)$$
$$u_{d1} = \frac{A_i A_o}{2} m(t)\cos(\theta_i - \theta_o) - \frac{A_i A_o}{2} m(t)\sin(2\omega_i t + \theta_i + \theta_o)$$
$$u_{d2} = \frac{A_i A_o}{2} m(t)\sin(\theta_i - \theta_o) + \frac{A_i A_o}{2} m(t)\sin(2\omega_i t + \theta_i + \theta_o)$$
$$u_I(t) = \frac{A_i A_o m(t)}{2}\cos(\theta_i - \theta_o)$$
$$u_Q(t) = \frac{A_i A_o m(t)}{2}\sin(\theta_i - \theta_o)$$

We can start by examining signal $u_i(t)$, the BPSK signal that enters the system. Technically, $u_i(t)$ occupies a bandwidth equal to twice the Data Rate, $2DR$, and its carrier frequency is $f_i = \frac{\omega_i}{2\pi}$; the data rate DR and bandwidth occupied by $u_i(t)$ comes from the signal component $m(t)$. However, for this to be true we must assume that $u_i(t)$ was filtered prior to entering the Costas Loop; the filter used is band-pass with single-sided bandwidth equal to $2DR$ and center frequency f_i (f_i is the center frequency in a range of frequencies occupied by $2DR$). This pre-filter is added to eliminate any frequency content outside the desired frequency range that could have been added to the signal during transmission, reception or previous processing of the signal. The highest frequency present in $u_i(t)$ is obtained by adding its carrier frequency to its data rate, $f_i + DR$. Signals $u_{o1}(t)$ and $u_{o2}(t)$ just have a spectral line at frequency f_i. Signals $u_{d1}(t)$ and $u_{d2}(t)$ are composed of the sum of two frequency components, each having a single-sided spectral line at 0 Hz and $2f_i$; at each frequency component, $m(t)$ provides a single-sided bandwidth equal to DR and $2DR$, respectively. Taking this into account, the highest frequency in these signals is $2f_i + DR$.

Signals $u_I(t)$ and $u_Q(t)$ are base-band signals with bandwidth equal to DR, hence the highest frequency present on these two signals is DR. Finally, although not included in the list of signals above, we should take a look at signals $u_d(t)$ and $u_f(t)$. These two signals have basically a DC component; therefore, their frequency value is practically zero. Based on this analysis, it turns out that the highest frequency term processed by the system is $f_{Max} = 2f_i + DR$. Having this result, we use the sampling theorem to determine the system frequency (sampling frequency) f_s.

$$f_s = 2f_{Max} = 2(2f_i + DR) = 4f_i + 2DR \qquad (8.1),$$

where $f_i = f_{Carrier}$. From Chapter 7 we know that $f_{Carrier} = 7MHz$ and $DR = 300KHz$. Hence, the sampling frequency of the digital Costas Loop has to be at least $f_s = 28.6MHz$; we will round this value to $f_s = 30MHz$. For an actual design, the system frequency (sampling frequency) selected must take into account any frequency variations in the input signal that could increase the value of f_{Max}; an example could be the Doppler effect. Expressing this additional variation as f_v, the sampling frequency could then be expressed as

$$f_s = 2f_{Max} = 2(2f_i + DR + f_v) = 4f_i + 2DR + 2f_v \qquad (8.2).$$

Now that we have the sampling frequency of the system we can calculate the sampling period, T_s. From (Oppenheim et al 1989 [7]) $T_s = \frac{1}{f_s}$ thus, substituting f_s into this equation, we obtain that the sampling period of the system is $T_s = \frac{1}{30MHz} = 33.33\,ns$.

Before going any further, there is an important rule of thumb that we must follow to make sure that a second-order digital phase-locked loop operates like a second-order analog

phase-locked loop. (Lindsey et al 1981 [18]), (Hao et al 1991 [19]), (Aguirre at el 1984 [20]), and (Shayan et al 1989 [21]) establish that a DPLL will operate like an APLL if its noise bandwidth is less than or equal to one tenth of its sampling frequency. Using this rule of thumb, we can derive what is well known as the time-bandwidth product relation.

$$B_L \le \frac{f_s}{10}$$
$$B_L \le 0.1 f_s$$
$$f_s = \frac{1}{T_s}$$
$$B_L T_s = 0.1 \qquad (8.3).$$

In (Lindsey et al 1981 [18]) and (Shayan et al 1989 [21]) it was also shown that the noise bandwidth of the DPLL is equal to that of the APLL when the time-bandwidth product relation is followed. Therefore, if we follow this relation we can assume that the noise bandwidth of the digital system is the same as that of its analog counterpart. The time-bandwidth product relation for our system is $B_L T_s = 8.808e-6$. This quantity is definitely much smaller than 0.1, hence our digital system should have a frequency response equal to its analog counterpart. With these results we now proceed to determine the other parameters of the digital system.

From Chapter 5, equation (5.16), the closed-loop transfer function of the DPLL is $H(z) = \frac{d_o z^{-1} + d_1 z^{-2}}{1 + c_1 z^{-1} + c_2 z^{-2}}$, where each of its components is given by $c_1 = K_d K_o T b_o - 2$, $c_2 = K_d K_o T b_1 + 1$, $d_o = K_d K_o T b_o$, $d_1 = K_d K_o T b_1$ and $T = T_s$. It is obvious that all these components depend on the sampling period T_s, the digital loop filter parameters b_o and b_1,

and the gain terms of the analog system K_d and K_o. However, these gain terms, along with the sampling period, are part of the gain of the digital system, defined as the digital gain in chapter 5, $K_D = K_d K_o T$, equation (5.17). Using the values previously calculated for K_d, K_o and T we determine that the gain of the digital system is $K_D = (1)(100e3)(33.33e-9) = 0.003333$. Unlike the analog system, which had a closed loop gain $K = 100e3$, the closed loop gain of the digital system is simply $K_D = 0.003333$. This huge difference in magnitude is the result of T_s, the sampling period.

To calculate the parameters of the digital loop filter b_o and b_1, we use equation (5.2), its transfer function $F(z) = \frac{b_o + b_1 z^{-1}}{1 - z^{-1}}$, where $b_o = \frac{2\tau_2 + T}{2\tau_1}$ and $b_1 = \frac{T - 2\tau_2}{2\tau_1}$. Using the values previously calculated for τ_1, τ_2 and T we obtain that $b_o = 7.0000417e-3$ and $b_1 = -6.9999583e-3$. Please, note that it is important to keep the difference between b_o and b_1. Based, on the values just calculated, it would have been very easy to express them as $b_o = 0.007$ and $b_1 = -0.007$. We could have also assumed that the sampling period was negligible (so small in magnitude) and decided to calculate b_o and b_1 using the relation $\frac{\tau_2}{\tau_1}$ with its respective sign. These two mistakes should be avoided at all cost since, by doing so, we would change the frequency response of the digital loop filter completely: instead of responding as a first-order, low-pass PI-type filter, it would respond as an all pass filter with DC gain equal to $-\frac{\tau_2}{\tau_1}$. Therefore, no matter how small the magnitude of the sampling period is, we must take it always into account when calculating the parameters of the digital

loop filter. This fact will have an impact at the time of deciding the quantization and scaling used to represent these parameters on a digital implementation.

Finally, it is time to calculate the parameters of the closed-loop transfer function, d_o, d_1, c_1, and c_2. Using the values already determined for K_D, b_o and b_1 we obtain that $d_o = 2.3331139\,e-5$, $d_1 = -2.3330861\,e-5$, $c_1 = -1.9999766689$ and $c_2 = 0.9999766691$. The closed-loop transfer function is then expressed as $H(z) = \dfrac{(2.3331139\,e-5)z^{-1} - (2.3330861\,e-5)\,z^{-2}}{1-1.9999766689\,z^{-1}+0.9999766691\,z^{-2}}$.

9.2 Establishing Fixed-Point Representation of the system parameters

To establish the fixed-point representation of the system parameters we use the theory provided in chapter 6. It is important to notice that there are some parameters, such as those in the loop filter transfer function, that we can easily represent as a fixed-point number (signed fixed-point number for this project). There are others however, that we will have to consider based on the magnitude of the signals processed by the system. We have to pay attention to the signal range and desired precision. This seems like a very difficult job, but in order to make it easier, it is a good idea to run simulations on the analog implementation and obtain from there the appropriate signal values and ranges that the system should handle. Doing so we must also consider phase and frequency variations in the input signal and the respective changes in magnitude seen on other signals in the system, $u_{d1}(t)$, $u_{d2}(t)$ and $u_f(t)$ for instance. When the difference in phase between the carrier of the input signal and

the VCO's generated replica is close to zero, the amplitude of $u_{d1}(t)$ and $u_{d2}(t)$ does not reach a magnitude of two. However, when the difference is close to 90°, their amplitudes reach a magnitude of two very easily; this is true assuming that $u_{d1}(t)$, $u_{d2}(t)$ and all the other signals in the system are as defined in chapters four and five. $u_f(t)$ is another signal to consider, this is because its amplitude is in the order of 10^{-3} when the phase error is large and 10^{-6} when the system is locked.

9.2.1 The BPSK signal source

We can start by defining the fixed-point representation of the input signal. This signal has amplitude bounded by 1 and –1, and we will not consider the effects of noise for this design. Choosing a total number of 8 bits to represent this signal, the fixed-point representation would have the format $A(1,6)$. The fixed-point representation of this signal was established using a converter token from SystemView's DSP library. The token was formatted to handle a signed fixed-point number of 8 bits in size and 6 bits allocated for the fraction size.

9.2.2 The Pre-filter

The pre-filter in the system is a 4^{th} order band-pass filter of the Bessel-type. The cutoff frequencies are 6.69e6 and 7.31e6. This filter was designed using SystemView 's linear systems design tool (shown as a token). Based on the results provided by system view, many of the parameters of this filter were in the order of 10^{-5}. These parameters can be

represented with 24 bits using an $A(1,22)$ format. This format also covers for some of the parameters that have a magnitude greater than one, but less than two. Simulations should be run using SystemView with just this filter and the input signal to make sure that no overflow occurs internally in the filter as a result of calculations when processing the input signal. Using this type of simulation, we can establish the format of the output signal of this filter as $A(1,20)$. Finally, at the output of the filter, we added a converter token to return the format of the signal to its original value $A(1,6)$. This is possible because SystemView operates on the magnitudes of the signals.

9.2.3 The I- and Q- Mixers

The I- and Q- multipliers are next to consider. These multipliers will be handling two 8-bit signals having fixed-point format $A(2,6)$ (input) and $A(2,5)$ (NCO output). Multiplying two 8-bit numbers results in a 16-bit number, therefore the register size for this token should be 16 bits long. However, based on the amplitude of this signal and simulation results we can consider using less bits to represent it and use format $A(2,7)$, a 10 bit representation.

9.2.4 The Arm Filters

To design the arm filters, we followed the same process employed for the band-pass pre-filter. This is a fourth-order Bessel-type low-pass filter with bandwidth 310KHz. A first-order filter with bandwidth 310KHz could have been designed, as we would have gotten

results just as good. For this filter we used a 34-bit register to represent its parameters and the format is $A(3,30)$. For the output signal we used format $A(3,24)$; this format was validated using simulations. At the output of each filter, we placed a converter token to change the signal representation format to $A(1,14)$. We tried to do this type of format change right from the filter tokens, but this feature is only available through the converter tokens.

9.2.5 The Third Multiplier

The third multiplier would be the next component to consider. This multiplier processes two signals with samples that are 16-bit long each. So, the register size for this multiplier is 32 bits and the format for the output would be $A(3,28)$. The output of this multiplier was not converted to a different format. Since it has so much information that requires high precision, we preferred to keep its format intact. To determine if the precision could be changed, just run a simulation on the analog system and use graphical analysis to have an idea of the minimum precision required to represent the signal. Then right click on the SystemView graph that shows this signal to obtain statistical information. This is the same analysis that should be done with every part of the system.

9.2.6 The Loop Filter

The parameters of the loop filter are very small in magnitude and they required 34 bits for representation and internal calculations of the filter. The format used to represent these parameters is $A(1,32)$. For the output of this filter we picked format $A(18,27)$, a 46-bit

representation. We chose this format to have enough bits in the representation to execute a 17-bit left shift operation that implements the gain of the NCO. This is the same gain used for the VCO in the analog system. The only difference with this implementation of the NCO gain term is that 2^{17} is 131072, which is much larger than 100e3, the gain of the analog system. For this reason this system will have a bit better of a performance, compared to the analog one.

9.2.7 The NCO

The final component to design is the NCO. The frequency of the carrier in the input signal is 7MHz, therefore the central frequency of the NCO has to be 7MHz. On the analog system, the VCO had a gain equal to 100e3. Hence, it allowed the VCO to oscillate between frequencies covered by $\pm Ku_f(t)$. The same thing applies to the NCO. Let us start by considering the relation $F_o = \frac{F_c}{3}$, where F_o is the output frequency of the NCO and F_c is its clock frequency. In our case, this is the system frequency. This is the relation that must be followed to make sure that the signal produced by the NCO is a clean one. Previously, we calculated that the system frequency would be 30MHz, based on the frequency of the input signal, its data rate, the processing done by the system and the Nyquist sampling rate. Following the above relation it turns out that F_o can be up to 10MHz. This result indicates that based on the system frequency, the NCO can have a central frequency of 10MHz and still be able to cover 100KHz of range above and below this value. Another equation to use in the design is $F_o = \frac{F_c W}{2^P}$, where W represents the input to the NCO and P is the size of

the register in the accumulator (NCO is designed as an accumulator). $\frac{F_c}{2^P}$ represents the precision of the NCO, which we will establish as 0.1, therefore, we can have P equal to 28. Another way to address the calculation of P is by determining the maximum value of W, which can be done through simulations. For our design, the size of use accumulator in the SystemView NCO token is 28 bits. The remaining part would be to determine how many bits from the accumulator to use to represent the phase.

The NCO also has a parameter for the number of bits to use in the amplitude of the output signal. This we had already determined to be 8 bits, so we do not have to consider it at this time. It is important to mention however, that these amplitude bits have nothing to do with those in the NCO accumulator. The amplitude bits are related to an internal ROM memory that has these values stored. Each memory locations in this ROM is addressed by the bits used to represent the phase, which come out of the accumulator register. Nevertheless, at the output of the NCO, the amplitude bits will be seen by system view as signed integer, so they will have to be converted to the appropriate format.

For this design, 12 bits from the accumulator register will be used to represent the phase. SystemView gives the option to use the NCO as either a frequency or phase modulator. We will use the NCO token as a frequency modulator, hence signal $Ku_f[n]$ will be connected to this input. The other input, phase modulation, cannot be left unconnected. The connection to this input should be done through a Custom source token with a constant output value equal to zero. It is important for SystemView, when running simulations, that all signals have the same interpretation format. Therefore, the output of the customer source

must be converted to a signed fixed-point format using a converter. The output of this converter has to be 12-bits long, as it is the phase representation chosen for this design.

Having designed each of the parts that form the digital Costas Loop, it is time to run the simulations and get the results. The results of these simulations are provided in the next chapter. Appendix A shows a block diagram implementation of the digital signal in SystemView. It also shows the design and simulation window used for the digital design; this is the window that is seen for any other type of design. Appendix B shows the parameter definition for each of the tokens used in the digital and analog designs. Appendix C shows various SystemView windows used to establish some of the system settings for simulations. Appendix D has Matlab programs that show the system response for the analog and digital systems as well as the Root locus diagrams for the digital systems. The root locus diagram can be used to determine the stability of the system, considering that the digital system is not completely stable (as it occurs with its second-order analog counterpart) and its stability can be determined as a function of the system gain.

10 Simulation Results: Digital Costas Loop Implementation

The results of the digital design will be provided using simulation software SystemView and Matlab, as we did with the analog system. Since the simulation results for the digital system are the same as those obtained for the analog one, the data provided here may seem redundant. As with the analog design, we will use SystemView to provide the simulation results of the operation of the digital Costas Loop, and Matlab to show the Bode plot diagrams. For a digital system it is customary to plot frequency response diagrams using a normalized frequency axis that goes from –1 to 1, when we wish to show the double-sideband bandwidth. However, to show the similarity between the bode plots for this system and its analog counterpart, we decided to show the frequency axis in rads/s. To obtain the bode plot, a simple Matlab program was required, and is included in appendix D. Appendix D also has a Matlab program that plots the root locus for the digital system to determine the stability based on the system gain K_d. Appendix A shows the block diagram implementation for the analog and digital system. Appendix B shows the parameters for each of the tokens used in the digital and analog implementation. Appendix C shows some system windows used to establish SystemView simulation settings.

To obtain the magnitude and phase Bode Plot diagrams, we provided the parameters determined for the system transfer function to the Matlab program. The Bode diagram shows that the frequency and phase response of the system is equal to the analog Costas Loop (based on the linear model).

The digital Costas Loop also will be demodulating a BPSK signal with the same characteristics used on the analog system: data rate of 300KHz and carrier frequency of 7MHz.

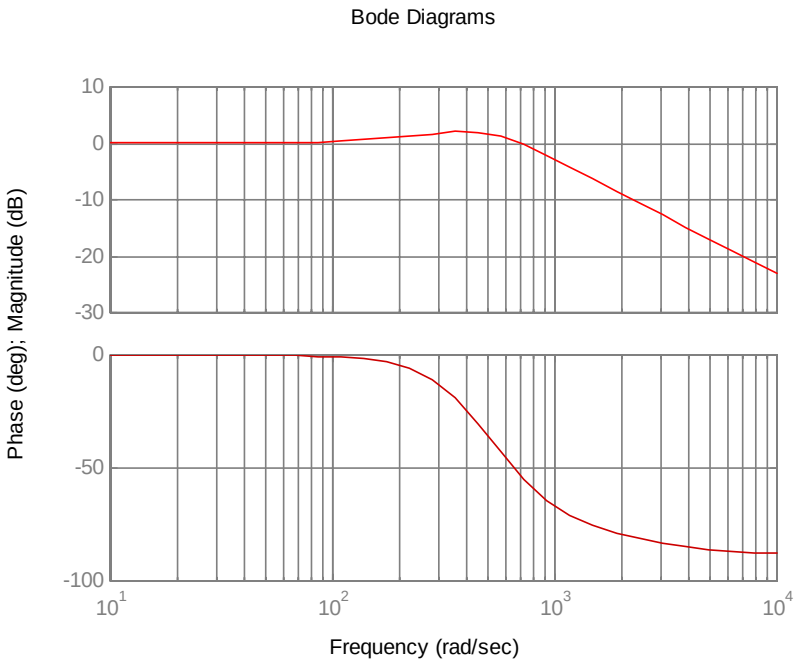

Figure 10-1. Bode Plot Diagrams

The simulation results obtained with SystemView are provided graphically in the next sections. The results are provided based on the phase and frequency offset assumed for the carrier f_i of the input signal on each simulation. The phase and frequency offset combination for each simulation is shown on the table below.

Phase Offset (Degrees)	Frequency Offset (Hertz)
0	0
45	0
0	100
0	700

Table 10-1. Phase and Frequency Offset Combinations

Each of the simulation results include graphs for signals $u_Q[n]$, $u_I[n]$ and $Ku_f[n]$. For the case of $u_f[n]$ we decided to show the results multiplied by the external gain, $K_o = 100e3$; the NCO token does not provide a gain option as the VCO does. The first two graphs, figures 10-2 and 10-3 show signals $u_I[n]$ and $u_Q[n]$ when f_i, the carrier frequency of the input signal has a phase and frequency offset equal to zero. As we can see from the figures, $u_Q[n]$ is quickly attenuated and settles to an average value of zero at around 800 microseconds; Costas Loop theory indicates that $u_Q[n]$ must be zero when the system is locked. $u_I[n]$, obtained after the arm filter, contains the demodulated data. This signal settles quickly as well and the magnitude of the recovered NRZ data is definitely bounded by amplitudes 1 and –1; as before, these are the pick to pick amplitude values for the NRZ data originally sent. The third graph is the output of the loop filter $u_f[n]$. It started with an under damped oscillation and quickly settled to an average value of zero at around 800 microseconds, just as the other signals did.

The second simulation assumes a phase offset of 45° or $\frac{\pi}{4}$ rads, but the frequency offset remains unaltered at zero. We will be using degrees as the unit to refer to the phase offset because this is the unit used by SystemView to establish waveform parameters. The same applies for the frequency offset. SystemView uses Hertz as the unit to establish frequency parameters for the system, waveforms and other components, therefore this is the unit that we will be using in this chapter to refer to frequency. All three figures show that the Costas Loop was able to track and lock quickly to the input signal as it did in the previous simulation. This time however, the output signal of the loop filter started with a positive oscillation. The amplitude of this signal settled back to zero as expected, once the system was locked. The time required for this process still took around 800 microseconds.

The third simulation assumes a frequency offset of 100 Hz and zero phase offset. Figures 10-8, 10-9 and 10-19 show again that the Costas Loop was still able to track and lock to the input signal, only this time it took the system longer to settle. Approximately, it took the system 6ms to lock to this signal. It is important to notice that the settling value for $Ku_f(t)$ was in the vicinity of 100 (amplitude), showing the relation between $Ku_f(t)$ and the system response.

The fourth simulation assumes a frequency offset of 700Hz; the phase offset remains at zero degrees. Unlike its analog counterpart, it appears from figures 10-11, 10-12 and 10-13 that the system is locking to the input signal. The reason for this difference is the fact that this system has an NCO gain of 2^{17}, which is larger than the VCO gain of 100e3 in the analog system.

10.1 Simulation Results with phase and frequency offsets equal to zero

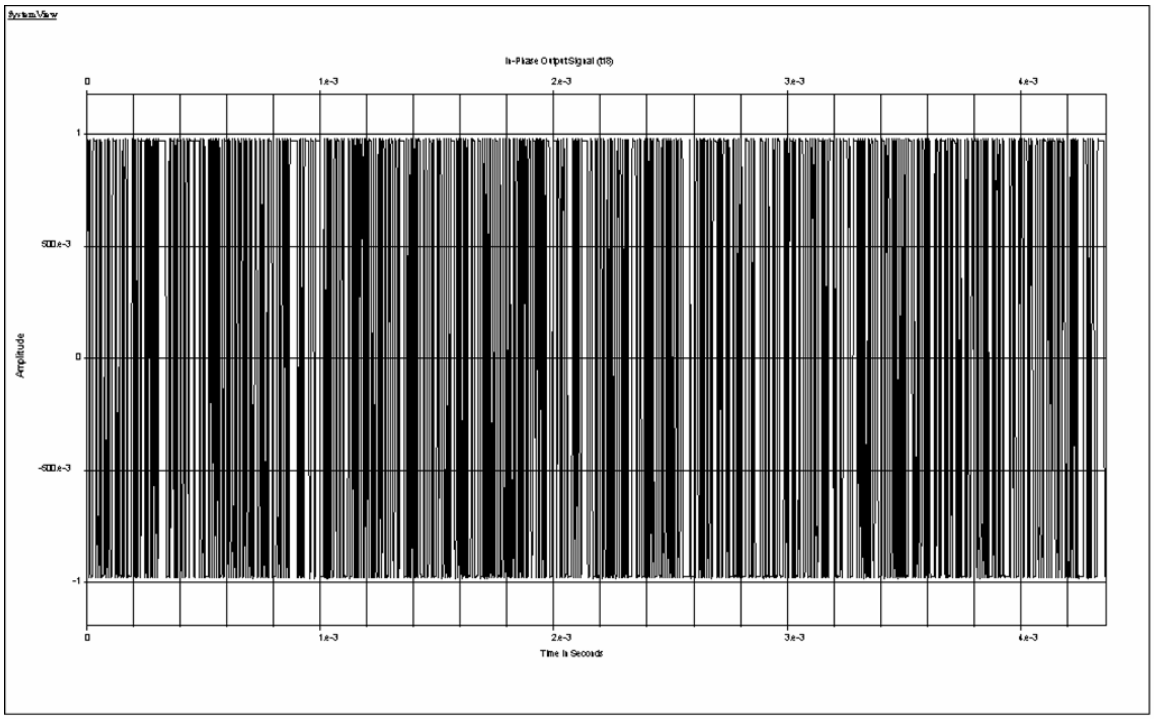

Figure 10-2. In-Phase Output Signal

In this graph, the x-axis represents time at 0.2ms/Division and

the y-axis represents the amplitude at 500mV/Division.

The mV unit was kept to show the similarity between this system and

the analog implementation.

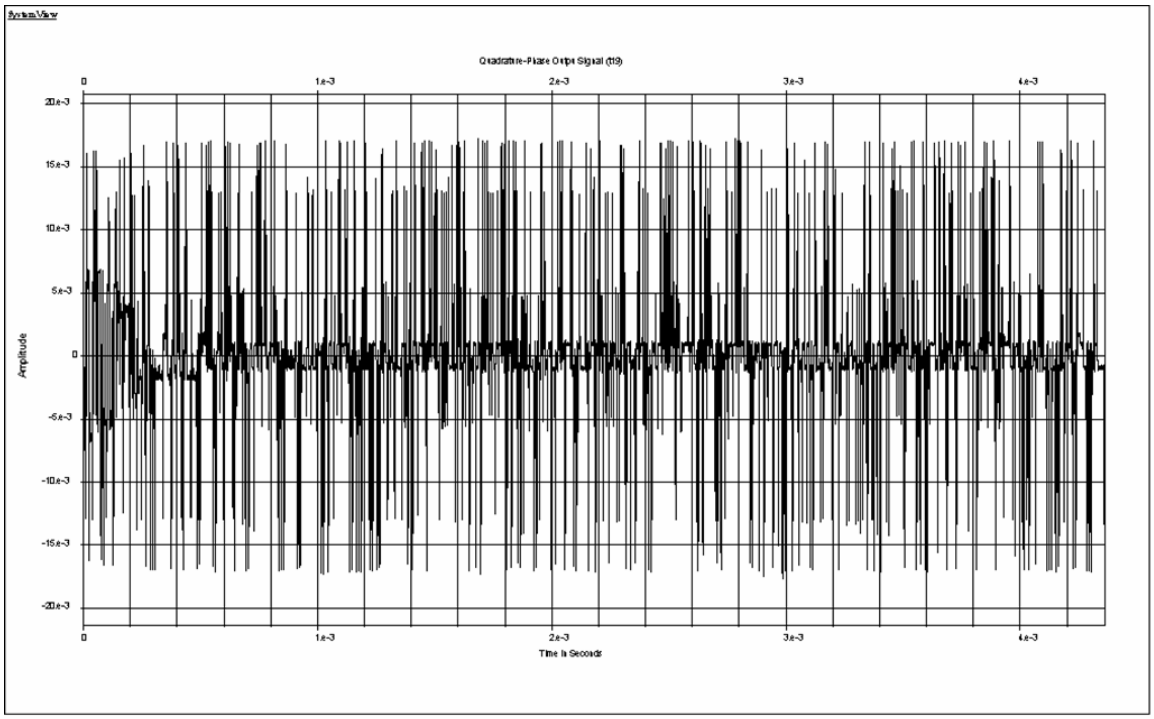

Figure 10-3. Quadrature-Phase Output Signal

In this graph, the x-axis represents time at 0.2ms/Division and the y-axis represents the amplitude at 5mV/Division.

The mV unit was kept to show the similarity between this system and the analog implementation.

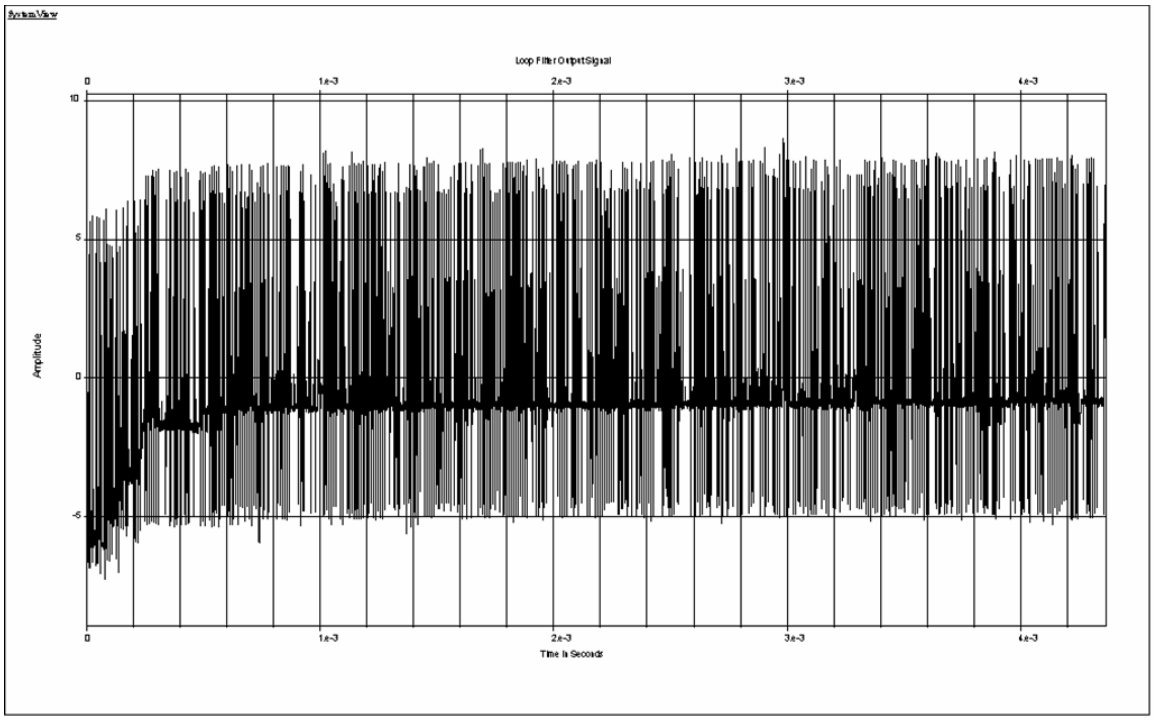

Figure 10-4. Loop Filter Output Signal Multiplied by 2^{17}

In this graph, the x-axis represents time at 0.2ms/Division and

the y-axis represents the amplitude at 5Hz/Division.

(By multiplying the output of the loop filter by K_D we obtain a frequency measure.)

10.2 Simulation Results with phase offset equal to 45 degrees

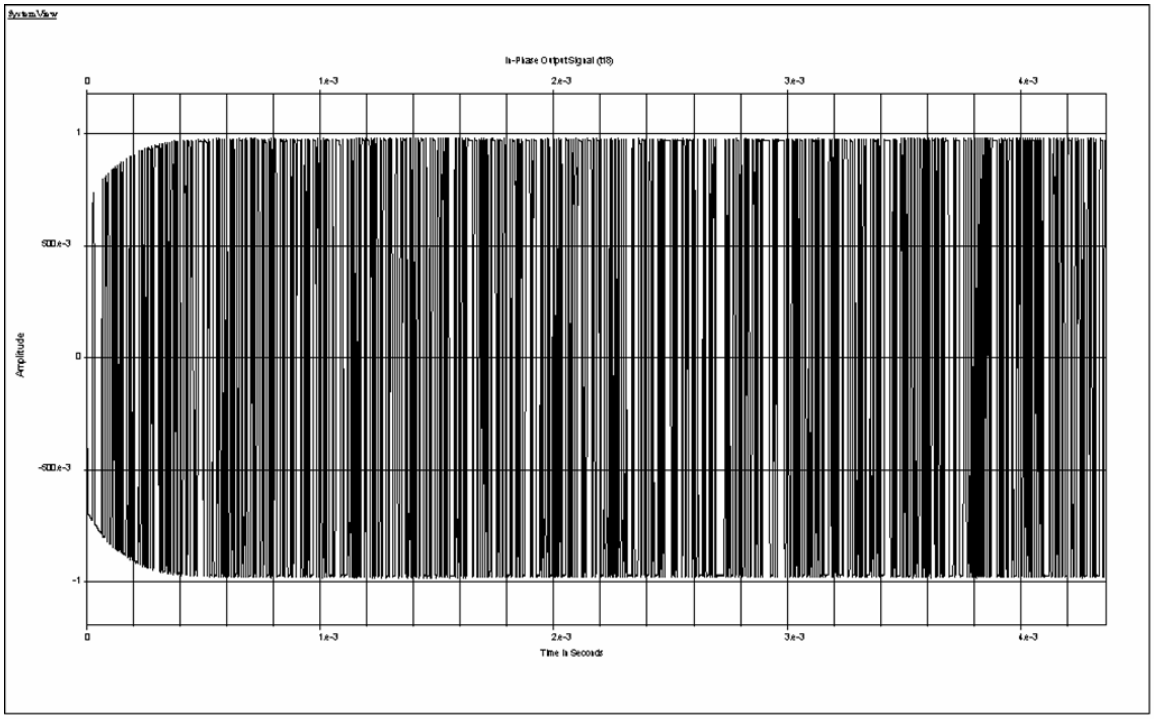

Figure 10-5. In-Phase Output Signal

In this graph, the x-axis represents time at 0.2ms/Division and the y-axis represents the amplitude at 500mV/Division. The mV unit was kept to show the similarity between this system and the analog implementation.

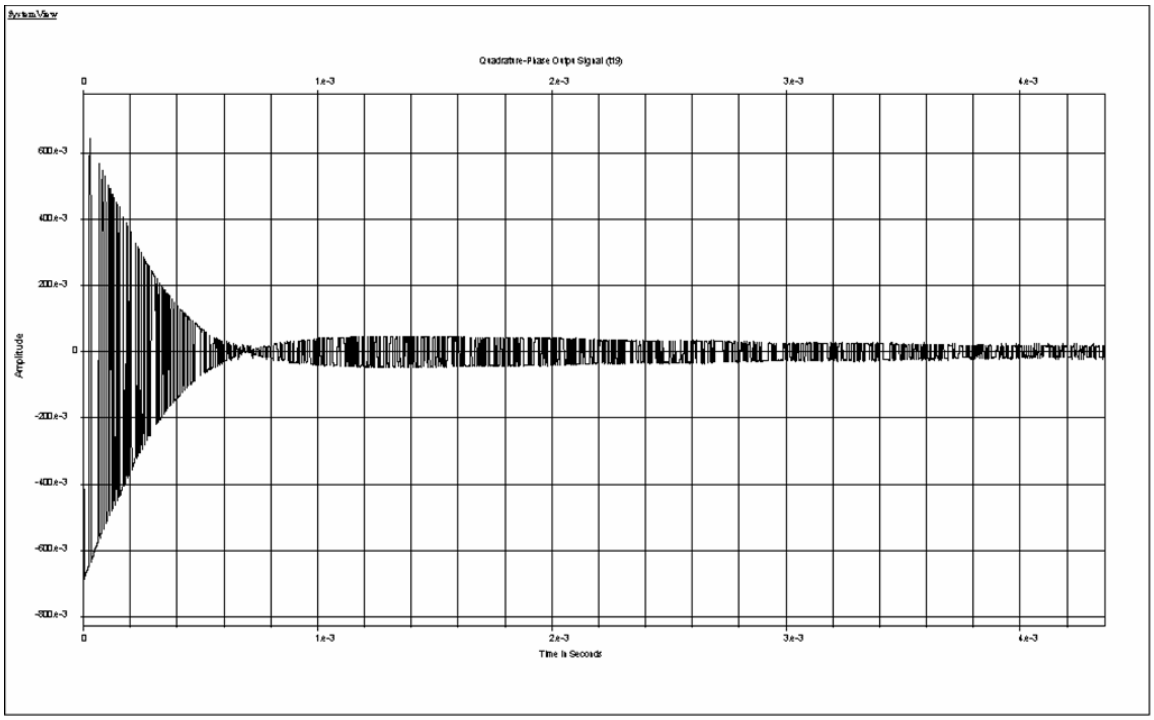

Figure 10-6. Quadrature-Phase Output Signal

In this graph, the x-axis represents time at 0.2ms/Division and the y-axis represents the amplitude at 200mV/Division. The mV unit was kept to show the similarity between this system and the analog implementation.

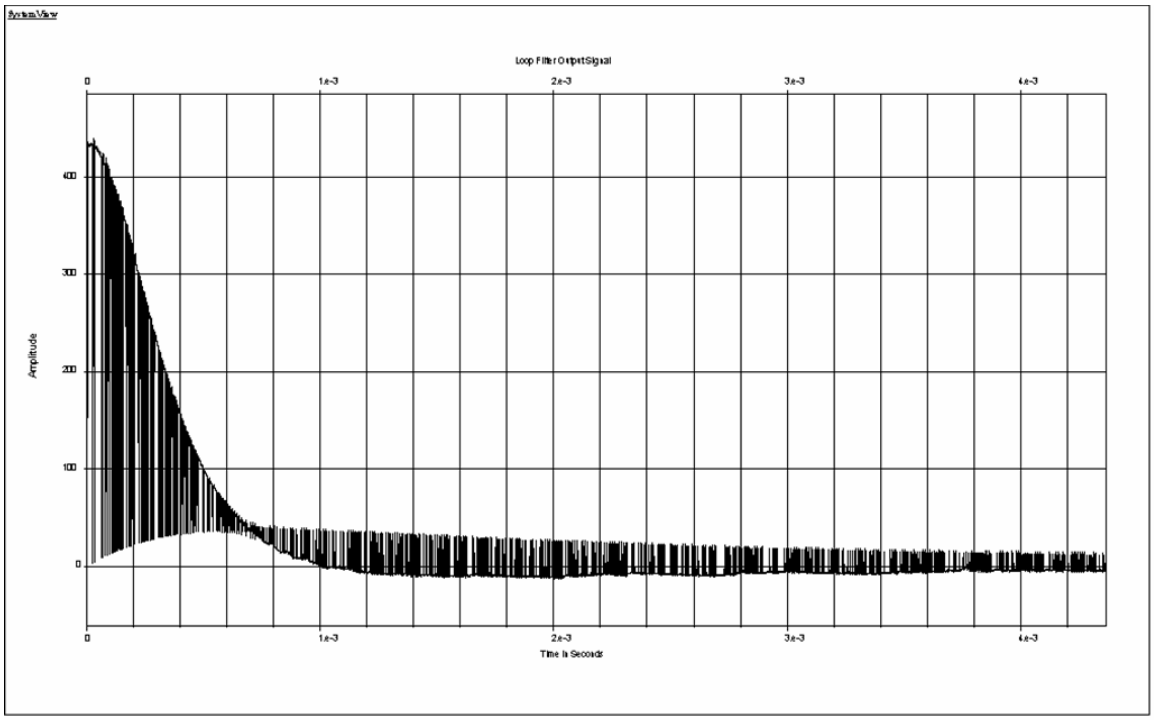

Figure 10-7. Loop Filter Output Signal Multiplied by 2^{17}

In this graph, the x-axis represents time at 0.2ms/Division and

the y-axis represents the amplitude at 5Hz/Division.

(By multiplying the output of the loop filter by K_D we obtain a frequency measure.)

10.3 Simulation Results with frequency offset equal to 100Hz

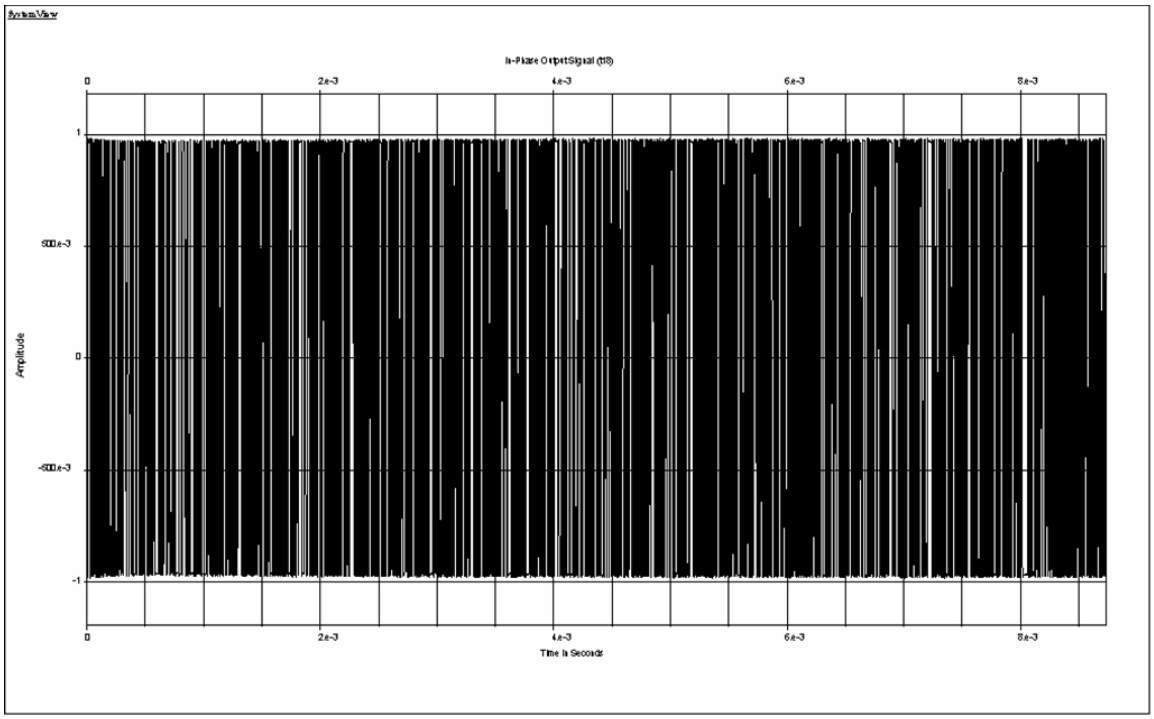

Figure 10-8. In-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 500mV/Division. The mV unit was kept to show the similarity between this system and the analog implementation.

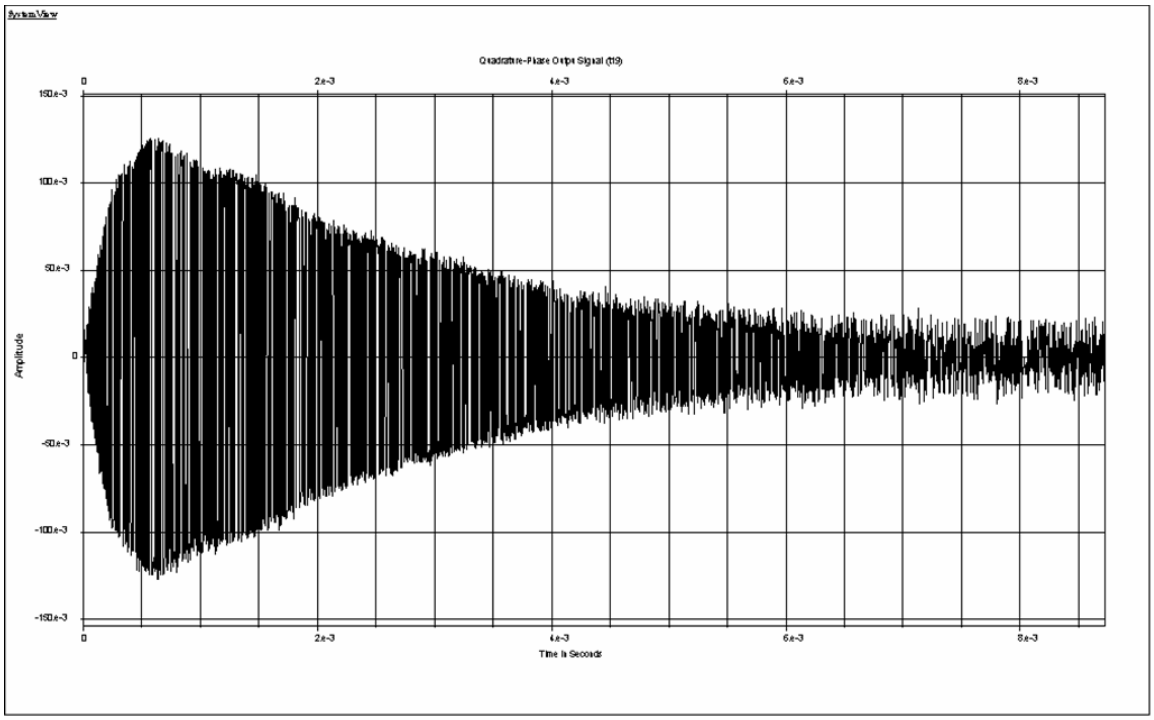

Figure 10-9. Quadrature-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 50mV/Division.

The mV unit was kept to show the similarity between this system and the analog implementation.

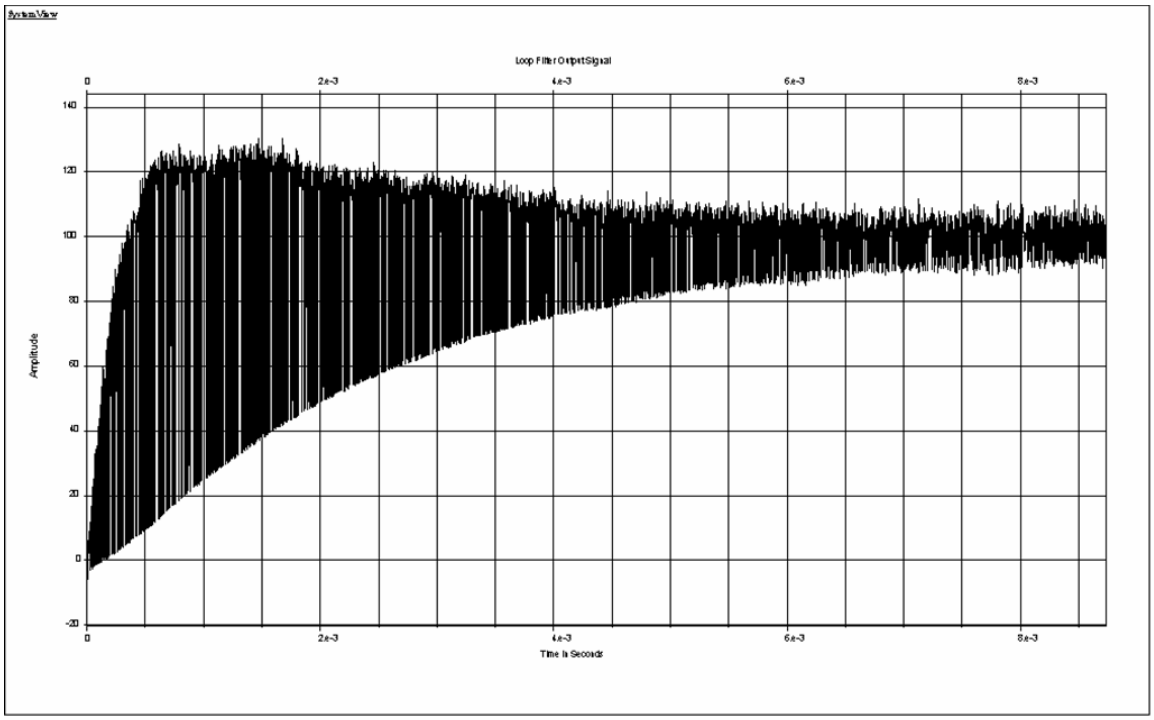

Figure 10-10. Loop Filter Output Signal Multiplied by 2^{17}

In this graph, the x-axis represents time at 0.5ms/Division and

the y-axis represents the amplitude at 20Hz/Division.

(By multiplying the output of the loop filter by K_D we obtain a frequency measure.)

10.4 Simulation Results with frequency offset equal to 700Hz

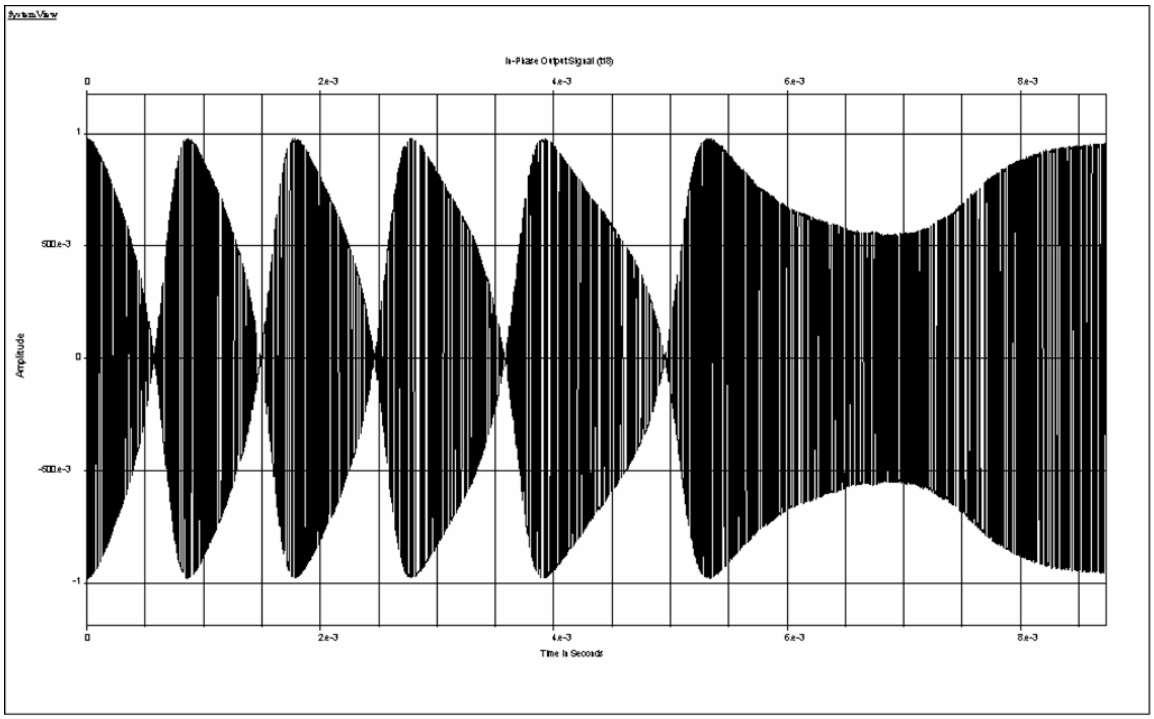

Figure 10-11. In-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 500mV/Division. The mV unit was kept to show the similarity between this system and the analog implementation.

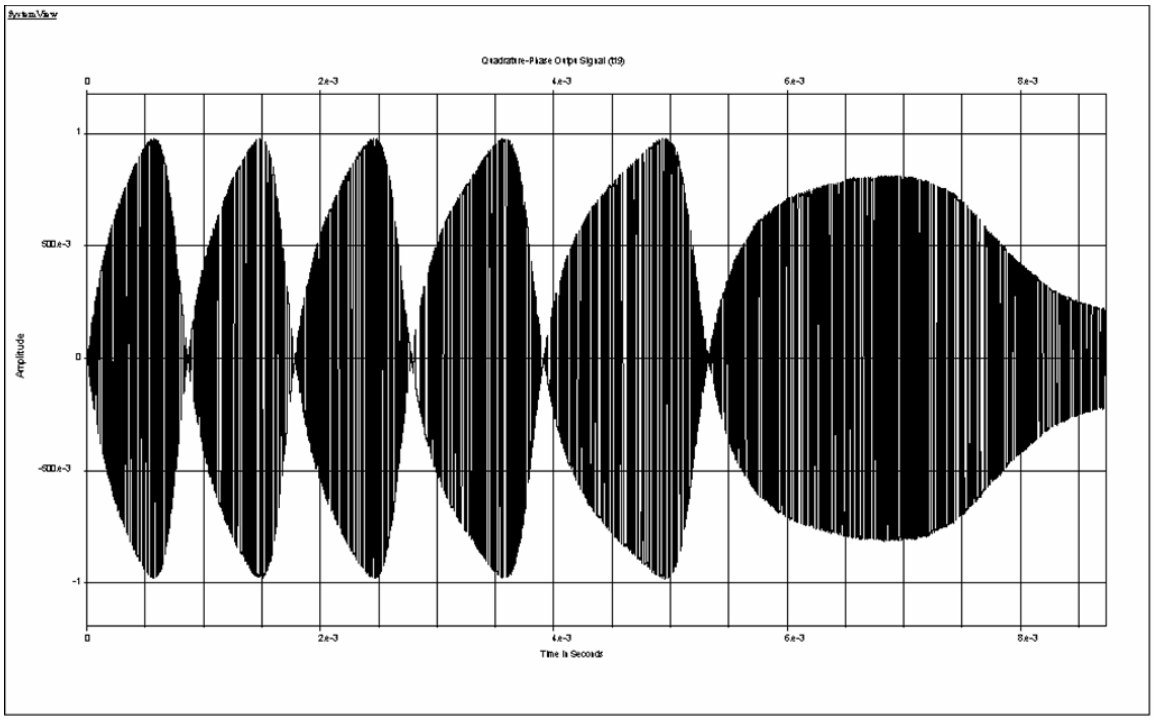

Figure 10-12. Quadrature-Phase Output Signal

In this graph, the x-axis represents time at 0.5ms/Division and the y-axis represents the amplitude at 500mV/Division. The mV unit was kept to show the similarity between this system and the analog implementation.

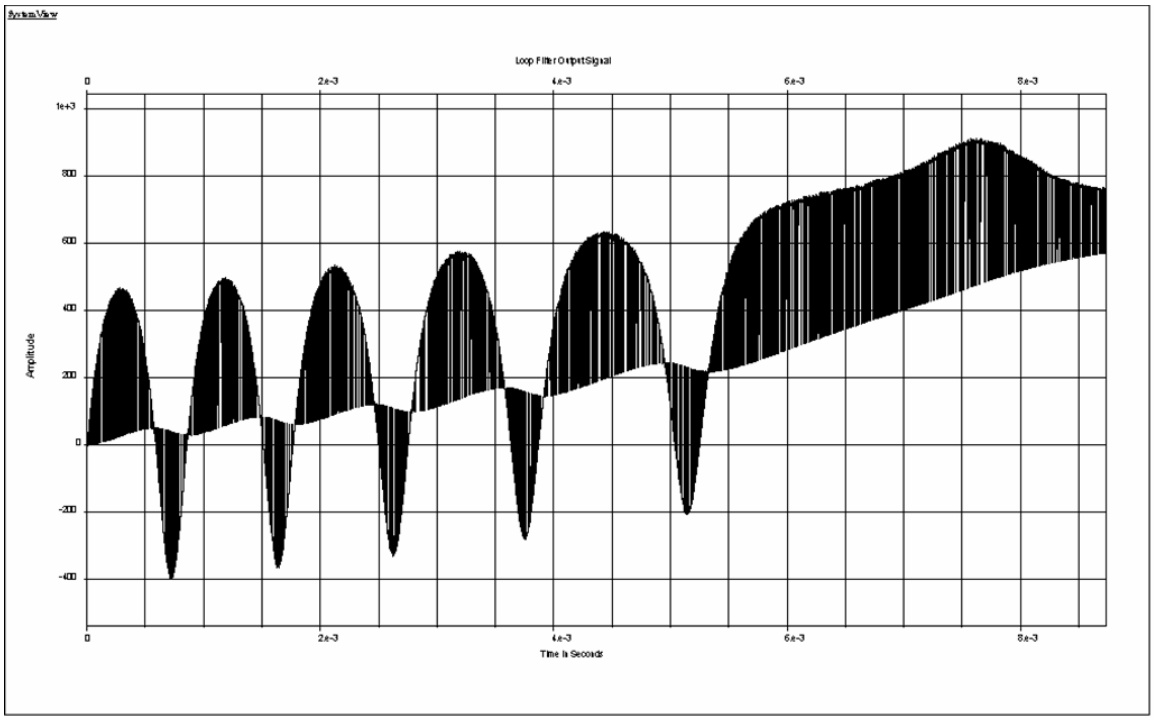

Figure 10-13. Loop Filter Output Signal Multiplied by 2^{17}

In this graph, the x-axis represents time at 0.5ms/Division and

the y-axis represents the amplitude at 20Hz/Division.

(By multiplying the output of the loop filter by K_D we obtain a frequency measure.)

11 Materials and Methodology Employed

Since IC technology became available, there has been a trend to implement many of the systems originally existing in the analog domain to the digital domain. The Costas Loop is a good example. Even when that is the case, there was little information available about this subject; but this was not the case for the phase-locked loop. There was a wealth of information on IEEE publications covering different aspects of the analog Costas Loop analysis, yet almost no data about digital implementations. The vast majority of the data covered analog Costas Loop and Phase-Locked loops, and digital phase-locked loops. That was the case for publications on the web. It was not until the past few years that abundant data became available about the Costas Loop subject, most of them about experiments or courses lectures.

Another missing link in the theory about Costas Loops was the lack of a linear model that could make easier its analysis and design. The lack of this model made things a bit difficult in the beginning, but as the theory became familiar, especially its PLL analysis, it became obvious that the Costas Loop could be modeled as a linear PLL. This was the first step towards the realization of this project. Then, it was a matter of combining Costas Loop theory with the linear PLL theory to obtain a suitable design with characteristics that could be easily determined.

The linear model developed was then used to obtain a discrete-time model to be used for the final digital implementation of the system. So, the process followed is basically to design an analog Costas Loop using the linear model developed and then use the parameters

obtained to design its digital counterpart. To allow an easy implementation, signed fixed-point arithmetic was employed. This number representation format was chosen because it is easily available on DSP and FPGA chips and makes processing a lot faster than say, floating point.

The implementation of the analog and digital systems was done using the 2001 student version of the simulation software SystemView, by Elanix. This software allows limited design capabilities for DSP and communications systems. However, the capabilities supported in this version were enough to accomplish both analog and digital designs. It would be better, however, to use an unrestricted version of the software, which can be easily obtained especially through Elanix University programs (Educational purposes, of course). Information about SystemView can be obtained at www.elanix.com. Other type of analysis, such as bode diagrams, was done using a 1997 student version of Matlab 5.0. This version comes with the necessary toolboxes to support discrete signal processing and controls analysis. These toolboxes are Signal Processing and Controls. More Information about MatLab can be obtained at www.mathworks.com.

SystemView has many tokens (tools) available to simulate many analog and digital communications devices. The token used to simulate digital systems are called DSP tokens. There are many functions available within each group (DSP and analog), but the ones we used are listed below. For example, the analog design used a BPSK source generator, linear filters from the linear systems design tool, multipliers, VCO, and analysis tokens (Sinks) that obtained the results from the simulation and displayed them graphically. For the digital

implementation, we used components that could implement these same functions digitally: Converter, multipliers, digital filters designed through the linear systems design tool, shifter, NCO and the analysis tokens (Sinks). The design process for the digital system using SystemView is very well explained in chapter 9, so we would advise to refer to this chapter for more details.

12 Conclusion and Future Work

The design of the digital Costas Loop presented in this project was based on digitizing each of the components that formed the system. The purpose of this transformation was to obtain a digital system that could have the same characteristics, response and performance as the analog implementation. To make this happen, we first derived a linear model of the analog Costas Loop that resembled the linear model of the analog phase-locked loop. Then we continued with the digital transformation and implementation of the system using SystemView. The results of various simulations were provided for both analog and digital systems. Graphical results showed that both systems tracked and locked to input signals that had the same characteristics. The response of both systems was comparable, and it was obvious as we added phase and frequency offsets to the input signal.

Certainly, this type of implementation is referred to as a software system by (Best 1999 [1]). However, even when a software implementation of this system is possible using high-speed programmable DSP devices, advances in technology can make the implementation happen in hardware directly. Definitely, there are changes that would have to be added to such an implementation and an example is the Digital Costas Loop chip provided in (Best 1999 [1]) by Harris Corporation, now Intersil. This chip is the HSP50210, that has applications on Satellite receivers and modems, digital carrier tracking, BPSK, QPSK, 8-PSK OQPSK, FSK, AM and FM demodulators.

The Costas Loop designed in this project did not include noise effect, false lock, cycle slipping, and other optimization and performance analysis. Therefore, for future work we can consider extending the analysis currently provided to any one of these analysis proposed, including an actual physical implementation of the system.

Bibliography

1. Best, Roland E.: Phase-Locked Loops, Design, Simulations, and Applications, 4th ed., McGraw-Hill, New York, 1999.
2. Gardner, Floyd M.: Phaselock Techniques, 2nd ed., John Wiley and Sons, New York, 1979.
3. Egan, William F.: Phase-Lock Basics, John Wiley and Sons, New York, New York, 1998.
4. Wolaver, D. H.: Phase-Locked Loop Circuit Design, Prentice Hall, New Jersey, 1991.
5. Dixon R. C.: Spread Spectrum System, John Wiley and Sons, New York, 1984.
6. Stensby, J. L.: Phase-Locked Loops, Theory and Applications, CRC Press LLC, Florida, 1997.
7. Oppenheim Alan V., and Schafer Ronald W.: Discrete-Time Signal Processing, Prentice-Hall, New Jersey, 1989
8. Lindsey, William C., and Simon, Marvin K.: Telecommunication Systems Engineering, Prentice-Hall, New Jersey, 1973, Reprinted by Dover Publications, New York, 1991.
9. Roden, Martin S.: Analog and Digital Communication Systems, 3rd ed., Prentice-Hall, New Jersey, 1991.
10. Dorf, Richard C., and Bishop, Robert H.: Modern Control Systems, 7th ed., Addison-Wesley, New York, 1995.
11. Goldberg, Bar-Giora: Digital Techniques in Frequency Synthesis, McGraw-Hill, New York, 1996.

12. Kamperman F.L.A.J.: Design of an all-digital direct-sequence spread-spectrum receiver, PIRM ‘94/WCN, pp 1364-1367.
13. Gupta, Someshwar C: Phase-Locked Loops, Proceedings of the IEEE, Vol 63, No 2, pp 291-306, February 1975.
14. Spiegel, Murray R.: Manual de Fórmulas y Tablas Matemáticas, McGraw-Hill, Mexico, 1993.
15. Houpis, Constantine. H., and Lamont, Gary B.: Digital Control Systems, 2nd ed., McGraw-Hill, New York, 1992.
16. Hsieh, G. C., and Hung, J. C.: Phase-Locked Loop Techniques-A Survey, IEEE Transactions on Industrial Electronics, Vol. 43, No. 6, pp. 609-614, December 1996.
17. Kratzet, Stephen: PLL Hardware Design and Software Simulation Using the 16-bit Version of SystemView by Elanix, Application Note AN102A, Elanix Inc., April 7, 1997.
18. Lindsey, William C., and Chak Ming Chie: A Survey of Digital Phase-Locked Loops, Proceedings of the IEEE, Vol. 69, No. 4, pp. 410-431, April 1981.
19. Hao, Shi, and Puqiang Yan: A High Lock-In Speed Digital Phase-Locked Loop, IEEE Transactions on Communications, Vol. 39, No. 3, pp. 365-368, March 1991.
20. Aguirre, S., and Hurd, W. J.: Design and Performance of Sampled Data Loops for Subcarrier and Carrier Tracking, JPL TDA Progress Report 42-79, pp. 81-95, Jet Propulsion Laboratory, Pasadena, California, April 1984.
21. Shayan, Y. R., and Le-Ngoc, T.: All Digital Phase-Locked Loop: Concepts, Design, and Applications, IEEE Proceedings, Vol. 136, Pt. F, No. 1, pp. 53-56, February 1989.

22. Costas, John P.: Synchronous Communications, Proceedings of the IRE, pp. 1713-1718, December 1956.

23. Simon, Marvin K., and Lindsey, William C.: Optimum Performance of Suppressed Carrier Receivers with Costas Loop Tracking, IEEE Transactions on Communications, Vol. COM-25, No. 2, February 1977.

24. Lindsey, William C., and Simon, Marvin K.: Optimum Design and Performance of Costas Loop Receivers Containing Soft Bandpass Limiters, IEEE Transactions on Communications, Vol. COM-25, No. 8, August 1977.

25. Data Sheet HSP50210, Digital Costas Loop, Intersil Corporation, http://www.intersil.com, January 1999.

26. Baier J., and Ertl R.: Increasing the Frequency Resolution of NCO-Systems Using a Circuit Based on a Digital Adder, IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 43, No. 3, March 1996.

27. Data Sheet HSP45116, Numerically Controlled Oscillator/Modulator, Intersil Corporation, http://www.intersil.com, May 1999.

28. Vankka, Jouko: Methods of Mapping From Phase to Sine Amplitude in Direct Digital Synthesis, IEEE International Frequency Control Symposium, pp. 942-950, 1996.

29. Technical Staff of Osicom Technologies Inc.: Direct-Digital Frequency Synthesis, A Basic Tutorial, November 1999, http://www.osicom.com/notes/ddstutor.html.

30. Labrosse, Jean J.: Fixed-Point Arithmetic for Embedded Systems, C/C++ Users Journal, www.cuj.com. February 1998.

31. Instrumentation Newsletter, Using Stable Timing and PXI Phase-Locking Technology to Take Better Measurements, National Instruments, http://ni.com, Second Quarter 2000.

32. Sklar, Bernard: Digital Communications Fundamentals and Applications 2nd Edition, Prentice Hall PTR, New Jersey, 2001.

33. Yates, Charles R: Fixed-Point Arithmetic An Introduction, personal.bellsouth.net/y/a/yatesc/fp.pdf

A. Appendix: Diagrams of the Analog and Digital Costas Loops

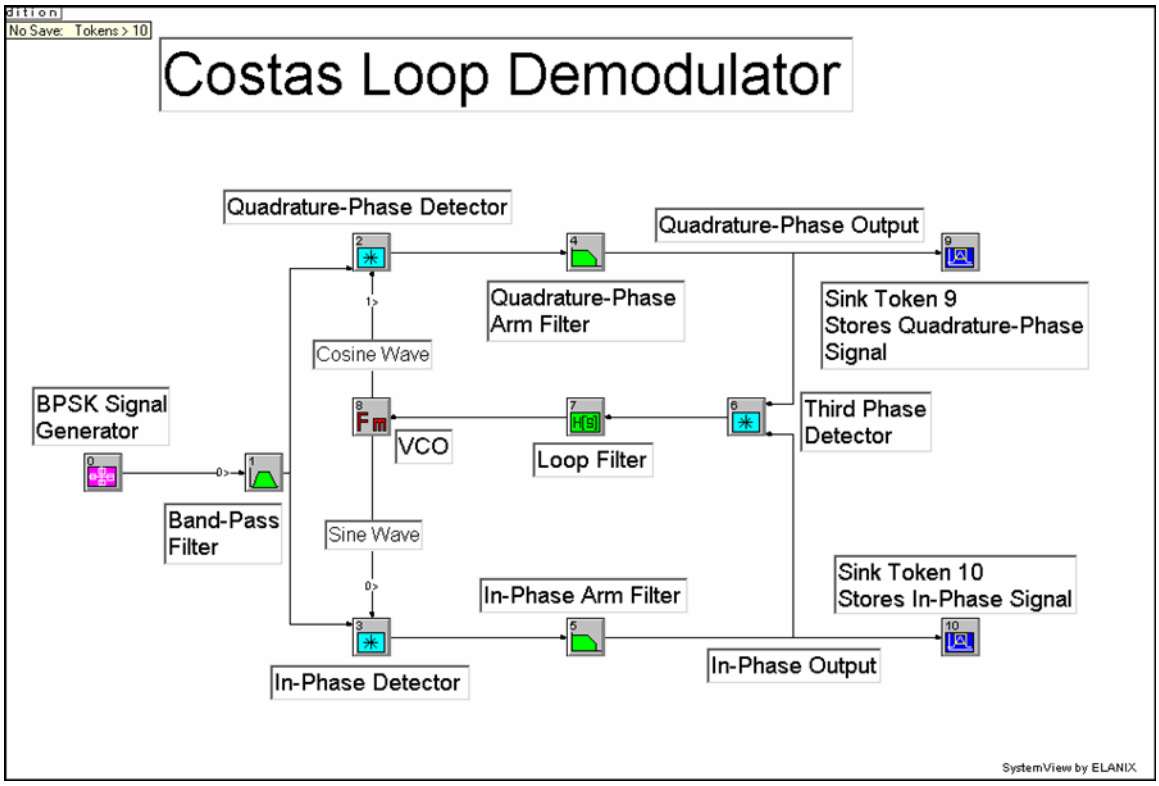

Figure A-1: SystemView Schematic of the Analog Costas Loop

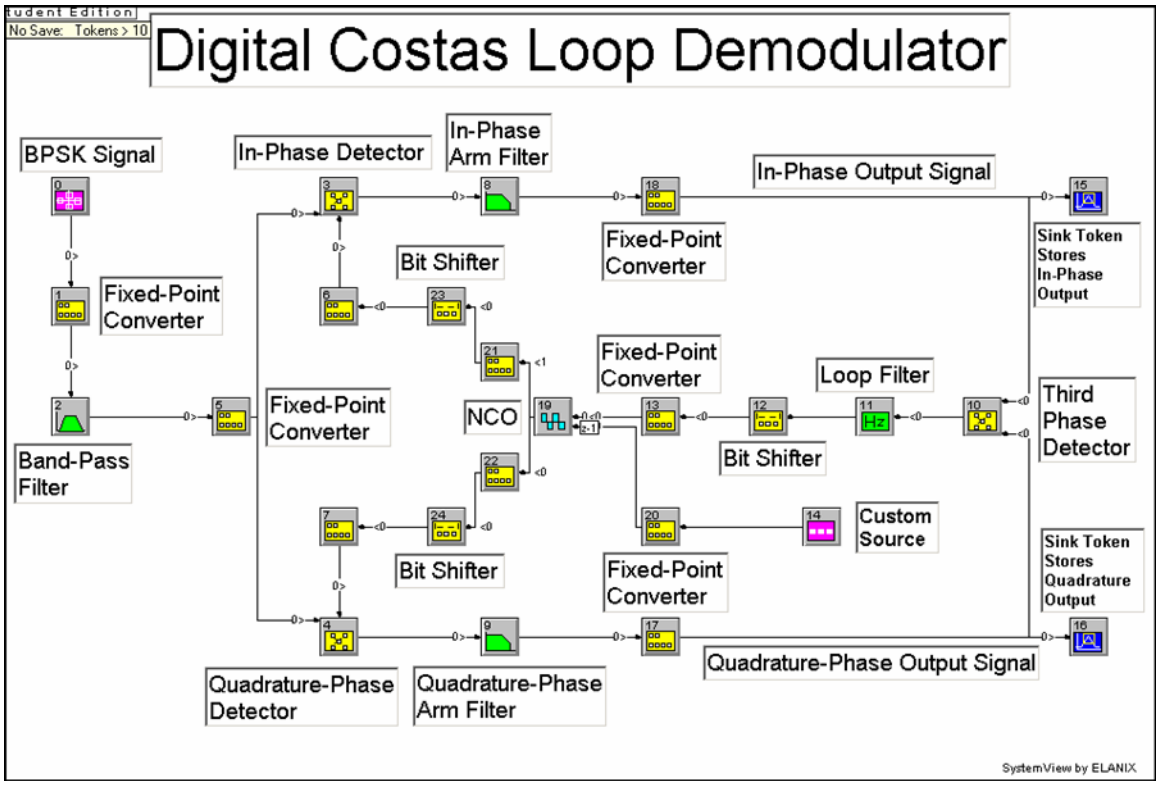

Figure A-2: SystemView Schematic of the Digital Costas Loop

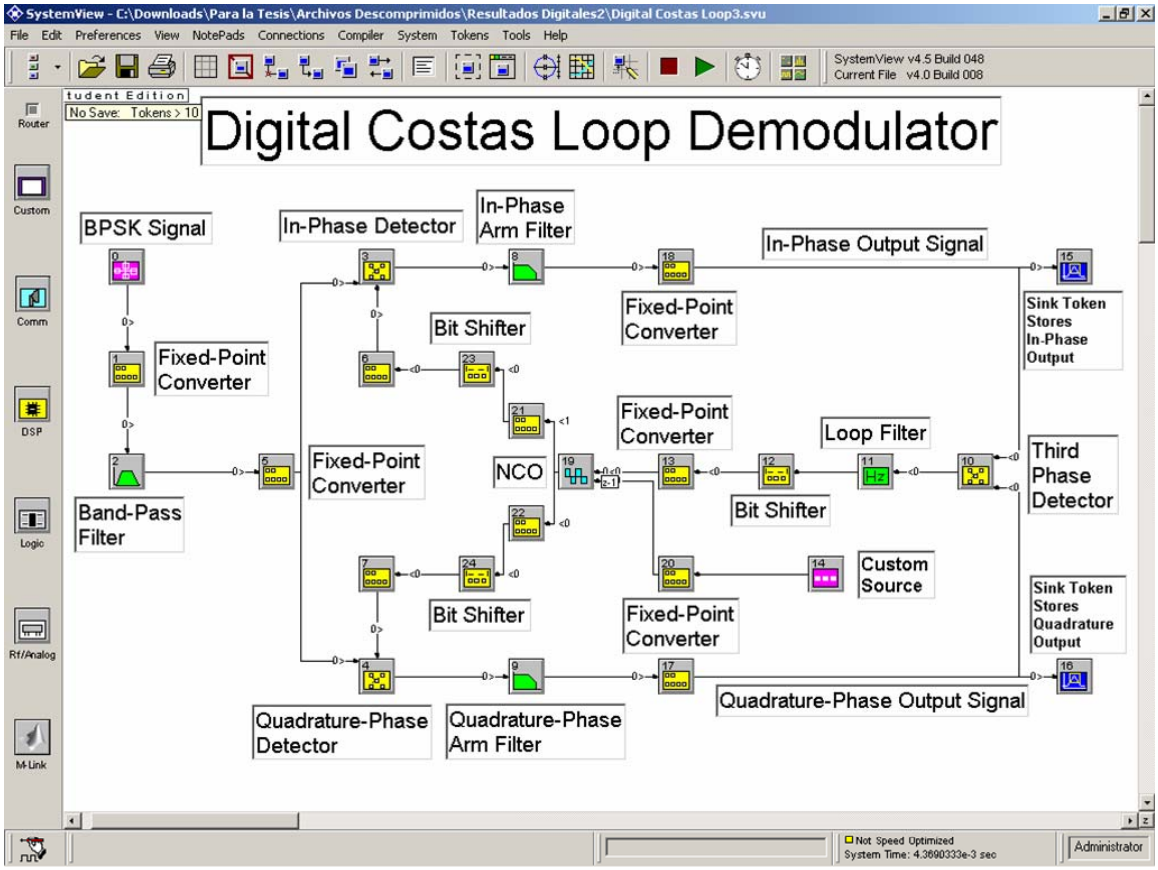

Figure A-3: Design and Simulation Window for SystemView

B. Appendix: Parameters Definition for SystemView

Source: PSK Amp = 1 v Freq = 7.e+6 Hz Phase = 10 deg Rate = 300.e+3 Hz Symbols = 2 Output 0 = Modulated Carrier Output 1 = Baseband Symbols (Token 0) BPSK-Modulated Signal	Operator: Linear Sys DSP Mode Disabled FPGA Aware = True Butterworth Bandpass IIR 3 Poles Low Fc = 6.69e+6 Hz Hi Fc = 7.31e+6 Hz Quant Bits = None Init Cndtn = Transient (Token 1) Pre-Filter	Operator: Linear Sys DSP Mode Disabled FPGA Aware = True Butterworth Lowpass IIR 3 Poles Fc = 310.e+3 Hz Quant Bits = None Init Cndtn = Transient (Tokens 4 & 5) I- & Q-Arm Filters
Operator: Linear Sys DSP Mode Disabled FPGA Aware = True Custom Laplace 1 Sections Quant Bits = None Num1 = (2.8e-3)s+1 Den1 = (400.e-3)s Init Cndtn = Transient (Token 7) Loop Filter	Function: FM Amp = 2 v Freq = 7.e+6 Hz Phase = 0 deg Mod Gain = 100.e+3 Hz/v Output 0 = Quadrature (Sin) Output 1 = In-Phase (Cos) (Token 8) VCO	Multiplier: Non Parametric Inputs from 1 8 Outputs to 4 (Token 2 & 3) I- & Q-Phase Detectors.
Sink: Analysis Input from t4 Output Port 0 (Token 9 & 10) Sink Tokens That Store The Results Of The Simulations.		

Table B-1: Parameters Definition for the Components of the Analog Costas Loop

Source: PSK Amp = 1 v Freq = 7.003e+6 Hz Phase = 20 deg Rate = 300.e+3 Hz Symbols = 2 Output 0 = Modulated Carrier Output 1 = Baseband Symbols (Token 0) BPSK Signal Generator	DSP: Converter FPGA Aware = True Data Type Out = Signed Fixed Pt Register Out = 8 bits Fraction Size = 6 bits Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag (Token 1) Converts Samples On The BPSK Signal To Signed Fixed-Point.	Operator: Linear Sys DSP Mode Enabled FPGA Aware = True Bessel Bandpass IIR 4 Poles Low Fc = 6.69e+6 Hz Hi Fc = 7.31e+6 Hz Quant Bits =24 Init Cndtn = Transient Coeff Data Type = Signed Fixed Pt Coeff Register = 24 bits Coeff Fraction Size = 22 bits Coeff Convert Mode = Numeric Value Data Type Out = Signed Fixed Pt Register Out = 22 bits Fraction Size = 20 bits Output Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag (Token 2) Pre-Filter: Filters The BPSK Signal To Limit Its Bandwidth
DSP: Converter FPGA Aware = True Data Type Out = Signed Fixed Pt Register Out = 8 bits Fraction Size = 6 bits Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag (Token 5) Converts The Output Of The Pre-Filter to A(1,6) Format.	DSP: Multiplier FPGA Aware = False Data Type Out = Signed Fixed Pt Register Out = 10 bits Fraction Size = 7 bits Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag (Tokens 3 & 4) Multipliers That Act As The I & Q Phase Detectors.	Comm: NCO Amp Bits = 8 Acc Bits = 28 Phase Bits = 12 Freq Offset = 7.e+6 Hz Phase Offset = 0 deg Freq In = t16 Output 0 Phase In = t17 Output 0 Output 0 = InPhase Output 1 = Quadrature (Token 19) Generates Local Carrier Replica Of The Input Signal.

Table B-2: Parameters Definition for the Components of the Digital Costas Loop

DSP: Converter FPGA Aware = True Data Type Out = Signed Fixed Pt Register Out = 8 bits Fraction Size = 5 bits Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag (Tokens 6 & 7) Converts The Processed Output Of The NCO to A(2,5) Format.	DSP: Bit Shift Direction = Right Shift By = 6 bits FPGA Aware = False Output 0 = Data t6 Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag Data Type Out = Signed Fixed Pt Register Out = 16 bits Exponent Out = 5 bits Max Rate (Port 0) = 30e+6 Hz (Token 23 & 24) Divides The Output Of The NCO By 64.	Operator: Linear Sys DSP Mode Enabled FPGA Aware = True Bessel Lowpass IIR 4 Poles Fc = 310.e+3 Hz Quant Bits =34 Init Cndtn = Transient Coeff Data Type = Signed Fixed Pt Coeff Register = 34 bits Coeff Fraction Size = 30 bits Coeff Convert Mode = Numeric Value Data Type Out = Signed Fixed Pt Register Out = 28 bits Fraction Size = 24 bits Output Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag (Tokens 8 & 9) I- and Q-Arm Filters.
DSP: Converter FPGA Aware = True Data Type Out = Signed Fixed Pt Register Out = 16 bits Fraction Size = 14 bits Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag (Tokens 17 & 18) Convert The Output Of The ARM Filters to A(1,14) Format.	DSP: Multiplier FPGA Aware = False Data Type Out = Signed Fixed Pt Register Out = 32 bits Fraction Size = 28 bits Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag (Token 10) Third Multiplier Or Phase Detector That Eliminates The Effects Of The Modulation.	Operator: Linear Sys DSP Mode Enabled FPGA Aware = True Custom Digital System 2 Num Coefs and 2 Den Coefs Quant Bits =34 Init Cndtn = 0 Coeff Data Type = Signed Fixed Pt Coeff Register = 34 bits Coeff Fraction Size = 32 bits Coeff Convert Mode = Numeric Value Data Type Out = Signed Fixed Pt Register Out = 46 bits Fraction Size = 27 bits Output Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag (Token 11) Loop Filter

Table B-2: Continuation

DSP: Bit Shift FPGA Aware = False Direction = Left Shift By = 17 bits Output 0 = Data Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag (Token 12) Shifter Used To Implement The Gain of the NCO	DSP: Converter FPGA Aware = True Data Type Out = Signed Fixed Pt Register Out = 28 bits Fraction Size = 9 bits Convert Mode = Numeric Value Output 0 = Data Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag (Token 13) Converters The Output Of the Bit Shifter to A(18,9) Format.	Source: Custom No. of Assigned Outputs = 1 Algebra p(0)=0 Operating Data Type = IEEE Double Output Data Type = IEEE Double Max Rate = 30e+6 Hz (Token 14) Custom Signal To Provide A Value To The Input Phase Parameter Of The NCO.
DSP: Converter Data Type Out = Signed Fixed Pt Register Out = 12 bits Fraction Size = 0 bits Convert Mode = Numeric Value FPGA Aware = True Output 0 = Data t19 Output 1 = Overflow Flag Output 2 = Carry Flag Output 3 = Zero Flag Output 4 = Sign Flag Output 5 = Underflow Flag Max Rate (Port 0) = 30e+6 Hz (Token 20) Converts The Output Of The Custom Signal to A(11,0) Format.	Sink: Analysis Input from t18 Output Port 0 Max Input Rate = 30e+6 Hz (Token 15 & 16) Sink Token That Store The Results Of The Simulations.	

Table B-2: Continuation

C. Appendix: System Specifications for SystemView

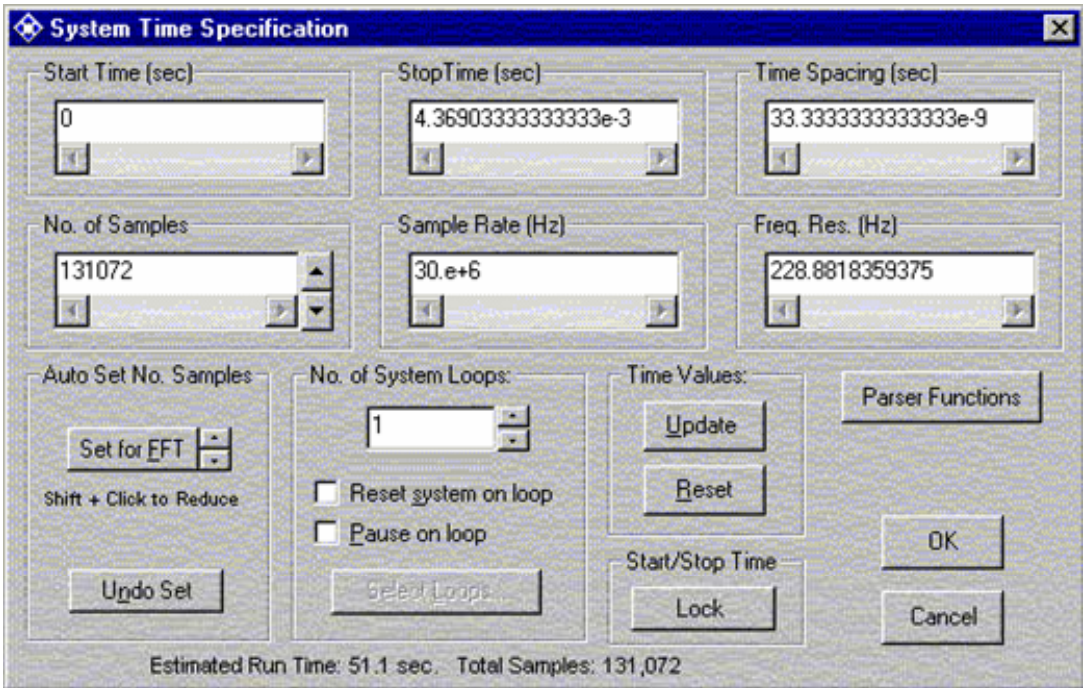

Figure C-1: System Specifications for SystemView

D. Appendix: Matlab Programs

D-1 Plotting the Bode Diagram for the Analog and Digital Systems

```
% This is a Matlab code that will calculate the parameters
% of both analog and digital Phase-Locked Loops in order
% to present a plot of the frequency response for each case.
% To obtain the plot of the frequency response, we need to use
% the transfer function of each of the Systems (APLL and DPLL).
% The transfer function of the APLL is defined as H(s) = N(s)/D(s).
% Here I am expressing it as HS = NS/DS; where NS is the numerator
% and DS the denominator of the APLL transfer function.
% The transfer function of the DPLL is defined as H(z) = N(z)/D(z).
% Here I am expressing it as HZ = NZ/DZ; where NZ is the numerator
% and DZ the denominator of the DPLL transfer function.

% Defining Acronyms:

% PLL     = Phase-Locked Loop
% APLL    = Analog Phase-Locked Loop
% DPLL    = Digital Phase-Locked Loop
% ALF     = Analog Loop Filter
% DLF     = Digital Loop Filter
% NCO(z)  = Transfer Function of the NCO or Numerically Controlled
%           Oscillator


% Defigning Parameters:

% t2   = When expressed as 1/t2, Indicates The Zero Location Of The
%        Analog Loop Filter
% t1   = Parameter Of The Denominator Of The Analog Loop Filter;
%        1/t1 can be thought of as a gain term of the Analog Loop
%        Filter
% K    = Open-Loop Gain of the APLL; K = Kd*Ko;
% Kd   = Gain Of The Analog Phase Detector
% Ko   = Gain Of The VCO; Ko*Ts is the gain of the NCO
% T    = Sampling Period of the DPLL; T = 1/Fs;
% Fs   = Sampling Rate of the DPLL
% b0   = Indicates The Zero Location Of The Digital Loop Filter
% b1   = Indicates The Pole Location Of The Digital Loop Filter
% wn   = Natural Frequency Of The APLL
% a    = Constant Relating t1 and t2; t1 = a*t2; a has to be greater
%        than or equal to 2.
% r    = Damping Ratio of the APLL

% Parameters of the Transfer Function of the APLL:

% NS   = 2*r*wn*s + wn^2          =   Numerator of the APLL Transfer
%                                     Function
% DS   = s^2 + 2*r*wn*s + wn^2    =   Denominator of the APLL Transfer
%                                     Function
% HS   = NS/DS                    =   Transfer Function of the APLL
```

```
% Parameters of the Transfer Function of the DPLL with PI Loop Filter
% NCO(z) = 1/(1-z^-1):

% NZ   = do + d1*z^-1             =   Numerator of the DPLL Transfer
%                                     Function
% DZ   = 1 + c1*z^-1 + c2*z^-2    =   Denominator of the DPLL APLL
%                                     Transfer Function
% HZ   = NZ/DZ                    =   Transfer Function of the DPLL
% al   = 1 + Kd*Ko*T*bo
% do   = Kd*Ko*T*bo/al
% d1   = Kd*Ko*T*b1/al
% c1   = (Kd*Ko*T*b1 - 2)/al
% c2   = 1/al

% Parameters of the Transfer Function of the DPLL with PI Loop Filter
% NCO(z) = z^-1/(1-z^-1):

% NZ2  = do2 + d12*z^-1
% DZ2  = 1 + c12*z^-1 + c22*z^-2
% HZ2  = NZ2/DZ2;
% do2  = K*T*bo;
% d12  = K*T*b1;
% c12  = (K*T*bo-2);
% c22  = K*T*b1+1;


% Preparing Matlab Session

close all;     % Closes all figures open
clc;           % Clears the Command Line Screen
clear all;     % Clears All Defined Varialbes
format long;   % Sets all Matlab Computations to scaled fixed-point
               % format with 15 digits

% Defining Parameters of the APLL

Ko       = 100e3;
Kd       = 1;
K        = Ko*Kd;
wn       = 500;
r        = 0.7;
a        = Ko/(2*r*wn);
t2       = (2*r)/wn;
t1       = a*t2;

% Calculating the Transfer Function of the APLL
num      = [2*r*wn wn^2];
den      = [1 num];
HS       = tf(num,den);

% Defining Sampling Frequency (Fs) and Time Period (T)
Fs       = 30e6;
T        = 1/Fs;


% Calculating Parameters of the Digital Loop Filter
```

```
bo       = (T+2*t2)/(2*t1);
b1       = (T-2*t2)/(2*t1);

% Calculating Parameters of the DPLL with NCO(z) = z^-1/(1-z^-1)
al       = (K*T*bo+1);
do       = K*T*bo/al;
d1       = K*T*b1/al;
c1       = (K*T*b1-2)/al;
c2       = 1/al;

% Calculating the Transfer Function of the DPLL with
% NCO(z) = z^-1/(1-z^-1)

NZ       = [do d1 0];
DZ       = [1 c1 c2];
HZ       = tf(NZ,DZ,T);

% Calculating Parameters of the DPLL with NCO(z) = 1/(1-z^-1)

do2      = K*T*bo;
d12      = K*T*b1;
c12      = (K*T*bo-2);
c22      = K*T*b1+1;

% Calculating the Transfer Function of DPLL with NCO(z)=1/(1-z^-1)

NZ2      = [do2 d12];
DZ2      = [1 c12 c22];
HZ2      = tf(NZ2,DZ2,T);

% Generating Bode Plot For The Transfer Function Of The APLL

figure(1)       % Opening a Window For a Figure (Bode Plot)
bode(HS);
title('Bode Plots For APLL With PI Loop Filter')
grid on
zoom on

% Generating Bode Plot For The Transfer Function Of The DPLL
figure(2)       % Opening a Window For a Figure (Bode Plot)
bode(HZ)
title('Bode Plots For DPLL With PI Loop Filter 1/(1-z^-1)')
grid on
zoom on

% Generating Bode Plot For The Transfer Function Of The DPLL
figure(3)       % Opening a Window For a Figure (Bode Plot)
bode(HZ2)
title('Bode Plots For DPLL With PI Loop Filter  z^-1/(1-z^-1)')
grid on
zoom on
```

D-2 Plotting the Root Locus for the Digital System

```
% This is a Matlab code that will calculate the parameters
% for two second-order digital Phase-Locked Loops based on
% those of an analog system in order to present the plot
% of the Root Locus for each system.  The difference
% between each digital system is based on the transfer
% function of the NCO.  The two transfer functions to
% be used are 1/(1-z^-1) and z^-1/(1-z^-1).
%
% To obtain the plot of the Root Locus, we need to use
% the Characteristic Equation of the System .
% The characteristic equation is defined as 1 + K*(p/q)=0.
% p and q are respectively the numerator and denominator
% of the open loop transfer function and k is the system
% gain.  The upper case K will be used to represent the
% gain of the analog system, whereas the lower case k
% represents the gain of the digital system.

% Defining Acronyms:

% PLL  = Phase-Locked Loop
% APLL = Analog Phase-Locked Loop
% DPLL = Digital Phase-Locked Loop
% ALF  = Analog Loop Filter
% DLF  = Digital Loop Filter


% Defigning Parameters of the APLL:

% t2 = 1/t2 Indicates The Zero Location Of The Analog Loop Filter
% t1 = Parameter Of The Denominator Of The Analog Loop Filter;
%      1/t1 can be thought of as a gain term of the Analog Loop Filter
% K  = Open-Loop Gain of the PLL; K = Kd*Ko;
% Kd = Gain Of The Phase Detector
% Ko = Gain Of The VCO and NCO
% wn = Natural Frequency Of The APLL
% a  = Constant Relating t1 and t2; t1 = a*t2.
% r  = Damping Ratio
% ps = Numerator of the Characteristic Equation of the APLL
% qs = Denominator of the Characteristic Equation of the APLL

% Defining Parameters of the DPLL:

% T  = Sampling Period of the DPLL; T = 1/Fs;
% Fs = Sampling Rate of the DPLL
% b0 = Indicates The Zero Location Of The Digital Loop Filter
% b1 = Indicates The Pole Location Of The Digital Loop Filter


% Defining Parameters of the characteristic equation when the
% NCO in the DPLL has transfer function 1/(1-z^-1).

% pz = Numerator of the Characteristic Equation of the DPLL
% qz = Denominator of the Characteristic Equation of the DPLL
```

```
% Defining Parameters of the characteristic equation when the
% NCO in the DPLL has transfer function z^-1/(1-z^-1).

% pz2= Numerator of the Characteristic Equation of the DPLL
% qz2= Denominator of the Characteristic Equation of the DPLL


% Preparing Matlab For The Process:

close all     % Closing All Figures
clc           % Clearing Command Window
clear all     % Clearing Memory
format long;  % Defining precision

% Defigning Gain Terms for the analog system:

Ko = 100e3;
Kd = 1;
k  = Ko*Kd;

% Calculating Parameters Of The APLL:

r  = 1/sqrt(2);
wn = 500;
a  = Ko/(2*r*wn);
t2 = (2*r)/wn;
t1 = a*t2;

% Calculating Parameters Of The DPLL:

Fs = 30e6;
T  = 1/Fs;
bo = (2*t2+T)/(2*t1);
b1 = (T-2*t2)/(2*t1);
K  = k*T;  % <--- Gain term for the DPLL

% Expressing Numerator and Denominator Of Characteristic
% Equation for The DPLL having NCO z^-1/(1-z^-1):
% For this specific system, if t1=0.4, t2=0.0028 and T=1/30e6,
% the location of the open loop zero is z=0.9999880953.

pz = [bo b1];
qz = [1 -2 1];


% Expressing Numerator and Denominator Of Characteristic
% Equation for The DPLL having NCO z^-1/(1-z^-1):
% For this specific system, if t1=0.4, t2=0.0028 and T=1/30e6,
% the location of the open loop zero is z=0.9999880953 and z = 0.

pz2 = [bo b1 0];
qz2 = qz;
```

```
% Parameters to define unit circle.
x  = [-1:0.1:1];
y  = [sqrt(ones(1,length(x))-x.^2)];


% Ploting ROOT LOCUS Of DPLL having open loop transfer function pz/qz:

figure(1)
rlocus(pz,qz);            % Plotting Root Locus
[gain,poles] = rlocfind(pz,qz,0.9);  % Finding gain and poles of HZ
                                     % when a pole location is 0.9
hold on
plot(x,y,'--',x,-y,'--')  % Plotting unit circle
zoom on

% Plotting ROOT LOCUS DPLL having open loop transfer function pz2/qz2:

figure(2)
rlocus(pz2,qz2);          % Plotting Root Locus
[gain2,poles2] = rlocfind(pz2,qz2,0.9); % Finding gain and poles of HZ
                                        % when a pole location is 0.9
hold on
plot(x,y,'--',x,-y,'--')  % Plotting unit circle
zoom on
```