FPGA-BASED DIGITAL PHASE-LOCKED LOOP ANALYSIS AND IMPLEMENTATION

BY

DAN HU

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Advisers:

Professor Steven J. Franke
Visiting Lecturer Christopher D. Schmitz
Abstract

The thesis presents a digital PLL project that will be used as an ECE 463 lab module and serve as a platform for future communication research projects. Field Programmable Gate Array (FPGA) technology is used for all digital signal processing tasks. A Direct Digital Synthesizer (DDS) is used to synthesize analog output, the frequency of which is controlled digitally by the FPGA. This system is implemented in a way that makes it educational and suitable for a lab module. Unlike purely digital PLL, this project involves several analog circuits soldered on PCBs, which will help the students visualize the signal flow in the PLL and get some exposure to mixed-signal systems.


## Contents

1 Introduction .................................................. 1

2 Theory ......................................................... 3
   2.1 PLL theory .............................................. 3
   2.2 Phase detector .......................................... 7
   2.3 DDS theory .............................................. 11
   2.4 Figures .................................................. 14

3 Implementation and Analysis ............................... 18
   3.1 FPGA .................................................... 18
   3.2 Other hardware .......................................... 24
   3.3 Modeling of actual system .............................. 24
   3.4 Figures .................................................. 27

4 Experiment and Characterization ....................... 34
   4.1 Constant parameters in experiment setup ............ 34
   4.2 Response to step stimulus ............................ 35
   4.3 Response to sinusoidal stimulus ...................... 37
   4.4 Processing lag .......................................... 39
   4.5 Lock range .............................................. 40
   4.6 Figures .................................................. 42

5 Conclusion .................................................. 47

References ..................................................... 48
1 Introduction

Phase-locked loop (PLL) is a linear feedback control system that can generate an output signal which has the same frequency and, perhaps, phase as the input reference signal. It consists of three major components: a phase detector (PD) that computes the phase error of the output with respect to the input reference signal, a loop filter that converts the phase error to a control voltage for the Voltage-Controlled Oscillator (VCO), and a VCO that generates a sinusoidal output. PLL has been widely used in many radio communication applications. For example, it can be used in a coherent receiver to recover the carrier frequency that is modulated by the transmitted data. In this thesis, we address issues that are related to FPGA implementation of digital PLL and present experimental characterization results of our PLL project.

Chapter 2 focuses on the theoretical concepts that are needed to understand this PLL project. Section 2.1 summarizes the basic principles and important characteristics of a generic digital PLL. Section 2.2 presents two algorithms for phase detector implementation that are used in this project and examines their limitations. Section 2.3 covers the theory of operation of DDS, which is used as the digital VCO in our project.

Chapter 3 covers the implementation details of the project, with emphases on FPGA programming and actual system modeling. Other hardware components in this project, such as linear regulators and a ceramic filter, are also
introduced in this chapter.

Finally, we characterize various aspects of the real system with several experiments. Chapter 4 explains the experiment setup and presents the measurement results that are used to verify the validity of the analytical model developed in Chapter 3.
2 Theory

Farhang-Boroujeny discusses the theory of continuous-time PLL and discrete-time PLL, i.e. digital PLL, extensively in [1]. This chapter summarizes the theory of digital PLL and addresses issues that are important for FPGA implementation. Section 2.1 derives basic principles and presents mathematical models that will be used to characterize the system in later chapters. Section 2.2 introduces various algorithms for practical hardware implementation of phase detectors. Section 2.3 covers the theory of operation of generic DDS and details related to AD9954, the commercial DDS used in this project.

2.1 PLL theory

Figure 1 (figures are grouped at the end of each chapter) shows the generic block diagram for digital PLL. We denote input signal as \( x[n] = \cos(2\pi f_c n T_s + \theta_c[n]) \), output signal as \( y[n] = \cos(2\pi f_o n T_s + \theta_o[n]) \), and phase detector output as \( \epsilon[n] = k_D(\theta_c[n] - \theta_o[n]) \), where \( k_D \) is the phase detector gain. The control signal \( c[n] \) relates to the VCO phase output by

\[
\theta_o[n + 1] = \theta_o[n] + c[n] k_O T_s, \tag{1}
\]

where \( T_s \) is duration between two consecutive data samples. The transfer function of VCO can be written as
\[ H_{VCO}(z) = \frac{\Theta_o(z)}{C(z)} = \frac{k_O T_s}{z - 1}, \] (2)

where \( k_O \) is the gain of VCO. At steady state when frequency is locked, \( f_c = f_o \). Any small deviation in frequency can be included in the phase component. The linear model of PLL is given in Figure 2. The phase transfer function of PLL, which relates PLL input and output phase, is given as

\[
H(z) = \frac{\Theta_o(z)}{\Theta_c(z)} = \frac{L(z)H_{VCO}(z)}{1 + L(z)H_{VCO}(z)} = \frac{L(z)k_O k_D T_s/(z - 1)}{1 + L(z)k_O k_D T_s/(z - 1)} = \frac{L(z)k_O k_D T_s}{L(z)k_O k_D T_s + z - 1}. \] (3)

For first-order PLL, the loop filter is essentially a constant gain, \( k_L \):

\[ L(z) = k_L. \] (4)

Substituting Equation (4) in Equation (3), phase transfer function be-
comes

\[ H(z) = \frac{k_L k_O k_D T_s}{k_L k_O k_D T_s + z - 1} \]

\[ = \frac{K_L T_s}{K_L T_s + z - 1}, \quad (5) \]

where the cumulative gain, \( K_L \), is defined as

\[ K_L = k_L k_O k_D. \quad (6) \]

Z-transform of phase error is given as

\[ E(z) = \Theta_c(z) - H(z) \Theta_c(z) \]

\[ = \frac{(z - 1) \Theta_c(z)}{K_L T_s + z - 1}. \quad (7) \]

According to the final-value theorem for the z-transform, steady-state phase error is evaluated as

\[ \epsilon_{ss} = \lim_{z \to 1} (z - 1) E(z) \]

\[ = \lim_{z \to 1} \frac{(z - 1)^2 \Theta_c(z)}{K_L T_s + z - 1}. \quad (8) \]

When there is a step change in input phase, the PLL input can be written as \( \theta_c[n] = u[n] \) in discrete-time domain, and \( \Theta_c(z) = \frac{z}{z - 1} \) in z-domain. The steady-state phase error converges to zero, as shown in Equation (9).

\[ \epsilon_{ss} = \lim_{z \to 1} \frac{(z - 1) z}{K_L T_s + z - 1} = 0. \quad (9) \]
When there is a step change in input frequency, the PLL input can be written as $\theta_c[n] = 2\pi \Delta f_c T_s n u[n]$ in discrete-time domain, and $\Theta_c(z) = \frac{2\pi \Delta f_c T_s z}{(z-1)^2}$ in z-domain. The steady-state phase error converges to a constant, as shown in Equation (10).

$$
\epsilon_{ss} = \lim_{z \to 1} \frac{2\pi \Delta f_c T_s z}{K_L T_s + z - 1} = \frac{2\pi \Delta f_c}{K_L}.
$$

For an ideal phase detector that has a linear range from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$, the maximum frequency deviation allowed at the input is

$$
|\Delta f_c| < \frac{K_L}{4}.
$$

A second-order loop PLL is obtained when $L(z)$ is a first-order loop filter. We examine the case where the loop filter is a proportional-integral (PI) filter, which can be expressed as

$$
L(z) = k_L \frac{1 + \alpha z^{-1}}{1 - z^{-1}},
$$

where $k_L$ and $\alpha$ are filter parameters. Substituting Equation (12) in Equation (3), phase transfer function becomes

$$
H(z) = \frac{k_L k_O k_D T_s (z + \alpha)}{z^2 + (k_L k_O k_D T_s - 2)z + k_L k_O k_D T_s \alpha + 1} = \frac{K_L T_s (z + \alpha)}{z^2 + (K_L T_s - 2)z + K_L T_s \alpha + 1}
$$

Z-transform of phase error is given as
\[ E(z) = \Theta_c(z) - H(z)\Theta_c(z) = \frac{(z - 1)^2 \Theta_c(z)}{z^2 + (K_LT_s - 2)z + K_LT_s\alpha + 1} \tag{14} \]

Steady-state phase error is evaluated as

\[ \epsilon_{ss} = \lim_{z \to 1} (z - 1)E(z) = \lim_{z \to 1} \frac{(z - 1)^3 \Theta_c(z)}{z^2 + (K_LT_s - 2)z + K_LT_s\alpha + 1} \tag{15} \]

Then we follow the same logic to analyze the steady-state phase errors in the presence of a step change in input phase and input frequency, respectively. Results are presented in Equations (16) and (17).

\[ \epsilon_{ss} = \lim_{z \to 1} \frac{(z - 1)^2z}{\frac{1}{z^2} + (K_LT_s - 2)z + K_LT_s\alpha + 1} = 0. \tag{16} \]

\[ \epsilon_{ss} = \lim_{z \to 1} \frac{(z - 1)2\pi f_c T_s z}{\frac{1}{z^2} + (K_LT_s - 2)z + K_LT_s\alpha + 1} = 0. \tag{17} \]

This second-order PLL has a lock range that goes to infinity since the phase error always converges to zero.

### 2.2 Phase detector

Given the in-phase and quadrature (IQ) components of the input signal,
the phase of the signal can be computed as

$$\epsilon[n] = \arctan\left(\frac{Q[n]}{I[n]}\right),$$

(18)

Figure 3 shows the block diagram of an ideal phase detector and Figure 4 shows the characteristics of an ideal phase detector that has a linear range of \((-\frac{\pi}{2}, \frac{\pi}{2})\). However, an \(\arctan\) function is difficult to implement on FPGA due to the limited resources and computation limits of the hardware. In this section, we present two practical implementations of phase detectors on FPGA and evaluate their performance.

2.2.1 Modified Costas phase detector

As discussed in [2], \(\arctan\left(\frac{Q[n]}{I[n]}\right)\) can be replaced by its mathematically equivalent \(\sin^{-1}\left(\frac{Q[n]}{\sqrt{Q[n]^2+I[n]^2}}\right)\). With small input phase, \(\sin^{-1}\left(\frac{Q[n]}{\sqrt{Q[n]^2+I[n]^2}}\right)\) can be approximated by \(\frac{Q[n]}{\sqrt{Q[n]^2+I[n]^2}}\). The expression \(\sqrt{Q[n]^2+I[n]^2}\) is the square root of the input signal power and can be removed using Automatic Gain Control (AGC). The Costas phase detector is expressed as \(\epsilon[n] = Q[n]\). The problem of this phase detector is that it has negative slope in the left-half of the I-Q plane. To correct this, the sign of \(I[n]\) is added to flip the phase detector output about the \(I\) axis in the left-half of the I-Q plane, and then the equation can be given as

$$\epsilon[n] = \text{sgn}(I[n])Q[n].$$

(19)
The gain of phase detector, $k_D$, is defined as the phase detector output divided by the input phase error. In this case, it is given as

$$k_D = \frac{\sqrt{P_{in}} \sin(\varepsilon[n])}{\varepsilon[n]},$$  \hspace{1cm} (20)

where $P_{in}$ is the input signal power and $\varepsilon[n]$ is the input phase error. The biggest problem for the Modified Costas phase detector is that $k_D$ depends on the input phase error. This type of phase detector is not strictly linear. As shown in Figure 5, the gain can be assumed roughly linear only when the input phase error is small. This leads to some error in system characterization. However, since the Modified Costas phase detector is a very computationally efficient algorithm, it is widely used in FPGA applications.

2.2.2 CORDIC phase detector

The CORDIC algorithm is an iterative method to compute a wide range of functions, including trigonometric, hyperbolic, and logarithmic, using only shift, add, and sign functions. It is employed in our project to implement the arctan phase detector. Andraka described the theory of the CORDIC algorithm being used in vectorizing mode in [3]. Below is the brief summary of the theory and the evaluation of its performance using MATLAB simulation.

In short, the goal is to find the angle of a vector ($x,y$) given its coordinates in a Cartesian plane. The CORDIC algorithm rotates the vector to align it with the x axis, minimizing $y$. This is done with many iterative steps. At each
step, the sign of y indicates which direction to rotate next and z accumulates the phase change. If it is initialized to zero, the final value will be the angle of the original vector. Equations for implementation are given below:

\[\begin{align*}
x_{i+1} &= x_i - 2^{-i} y_i d_i \\
y_{i+1} &= y_i + 2^{-i} x_i d_i \\
z_{i+1} &= z_i - \arctan(2^{-i}) d_i
\end{align*}\]  

where \( d_i = -\text{sgn}(y_i) \), \( z_1 = 0 \), \( x_1 = x \), and \( y_1 = y \). As a set of known values, \( \arctan(2^{-i}) \) can thus be implemented with a look-up table.

The biggest limitation of the CORDIC algorithm is that there exist upper and lower bounds for the output phase. In other words, the CORDIC algorithm output reaches a plateau after the input phase exceeds a certain value. The bound is computed mathematically in Equation (24).

\[|z_\infty| = \sum_{i=1}^{\infty} \arctan(2^{-i}) = 0.9579\]  

Figure 6 shows the characteristic of the CORDIC phase detector implemented with 6 iterations. With infinite iterations, the CORDIC phase detector should be perfectly linear within the region specified by its upper and lower bounds. The jaggedness that appears in Figure 6 is due to quantization error and can be reduced by increasing the number of iteration cycles. As shown in Figure 6, its phase detector gain \( k_D \) is constant as long as the mag-
The magnitude of input phase error does not exceed 0.9579 radians. To compute the lock range for first-order PLL implemented with CORDIC phase detector, Equation (11) should be modified to account for the clipping effect.

\[-0.9579 < \epsilon_{ss} = \frac{2\pi \Delta f_c}{K_L} < 0.9579\]  

(25)

Then the lock range is given by

\[|\Delta f_c| < 0.1525K_L\]  

(26)

2.3 DDS theory

An Analog Devices AD9954 DDS, configured to operate in single-tone mode, is used as the digital VCO in our project. This section summarizes the theory of operation of AD9954 and analyzes its performance in PLL. Discussion will focus on the aspects of AD9954 that pertain to the project.

As stated in [4], DDS is a technique for using digital data processing blocks as a means to generate a frequency- and phase-tunable output signal referenced to a fixed-frequency clock source. Figure 7 shows a generic block diagram of DDS. To begin with, it is instructive to visualize the sine generator as a vector rotating around a phase wheel, as shown in Figure 8. A revolution around the phase wheel corresponds to a cycle in a generated sine wave. The content of the phase accumulator, referred to as phase register in Figure 7, corresponds to the points on the phase wheel. At each system clock
cycle, the value of FTW is added to the value previously held in the phase accumulator. Larger FTW means skipping more points on the phase wheel per clock cycle, thus completing a cycle in the output sine wave faster. The output of phase accumulator is then translated to an amplitude value via the phase-to-amplitude look-up table. Finally, this amplitude value is converted to analog signal through a D/A converter. The number of points on the phase wheel is determined by the size of the phase accumulator, which, together with system clock $f_s$, determines the frequency tuning resolution of the DDS. For AD9954, a 32-bit register is used for both phase accumulator and FTW. The frequency tuning resolution is obtained as

$$
\delta f = \frac{f_s}{2^{32}}.
$$

(27)

The output frequency, $f_o$, of the DDS is a function of $f_s$, the value of FTW, and the size of the phase accumulator register. Their relationship is given below.

For $0 < FTW < 2^{31}$,

$$
f_o = FTW \frac{f_s}{2^{32}}.
$$

(28)

For $2^{31} < FTW < 2^{32} - 1$,

$$
f_o = f_s(1 - \frac{FTW}{2^{32}}).
$$

(29)

FTW value can be updated in real time by programming the desired value
into the FTW register via DDS serial I/O port. The AD9954 serial I/O is compatible with most standard serial communication protocols, including SPI, which is used in this project. This part is elaborated in Section 3.2.

In order to fit DDS into our PLL model, we need to characterize it in discrete-time domain. The relation between two consecutive output phase samples can be expressed as

$$\theta_o[n + 1] = \theta[n] + 2\pi \Delta f[n] T_s,$$

where $\Delta f[n]$ is DDS offset from free-running frequency at time $n$. Comparing with Equation (1), $\Delta f[n]$ relates to $c[n]$ by $c[n]k_O = 2\pi \Delta f[n]$. In our project, DDS is clocked with a 20 MHz external source and the internal system clock is scaled to 400 MHz by a clock multiplier. Therefore, $k_O$ can be computed as

$$k_O = 2\pi \frac{400M}{2^{32}} = 0.5849.$$ 

(31)

An intuitive way to interpret this result is that control signal $c[n] = 1$ corresponds to a 0.5849 rad/s increase in DDS output angular frequency.

AD9954 features precise frequency control with a step size of 0.0931 Hz for 400 MHz system clock. It is robust against component aging and temperature drift. The amount of output frequency jitter largely depends on the quality of the reference clock. Any deviation in reference clock frequency will result in magnified DDS output frequency deviation.
2.4 Figures

Figure 1: Generic discrete-time PLL

Figure 2: Linear model of discrete-time PLL

Figure 3: Ideal phase detector
Figure 4: Ideal phase detector characteristics

Figure 5: Modified Costas phase detector characteristics
Figure 6: CORDIC phase detector characteristics

Figure 7: Generic block diagram of DDS, cited from [4]
Figure 8: Phase wheel, cited from [4]
3 Implementation and Analysis

3.1 FPGA

3.1.1 NI 5640r hardware

NI 5640r is a very powerful IF transceiver that can be programmed with LabVIEW graphical programming language. In this chapter, we will cover the hardware components that are related to our project. More information on NI PCI-5640r can be found in [5].

As shown in Figure 9, NI PCI-5640r has two analog inputs (AI) with built-in digital downconverter (DDC), two analog outputs with built-in digital upconverter (DUC), and seven digital I/O (DIO) lines. In our project, one AI channel is used to receive the incoming signal at 10.7 MHz and five DIO lines are used to communicate with the DDS. Incoming signal is sampled at 100 MHz at the front end of ADC and then converted to baseband IQ data by DDC. IQ data can be processed directly on FPGA using inline processing or streamed to host using Direct Memory Access (DMA) via PCI interface.

NI PCI-5640r has six available base clocks that can be used to control the timing of the processes running on FPGA. Figure 10 shows the names of each clock and Figure 11 shows how each is derived from the source. The device reference clock that drives all base clocks except Configuration_Clk can be chosen from an internal VCXO of 200 MHz or an external clock.
Configuration Clk is the on-board clock used by PCI-DMA operation. It is fixed at 20 MHz and is independent of other clocks. By default, the majority of the functions on the FPGA that are not timed explicitly are compiled to this clock so that their execution time stays constant when other base clocks are customized to run at various frequencies.

ADC_0_Port_A_Clk specifies the baseband IQ data rate for AI channel 0. ADC has a fixed sampling rate of 100 MHz, which is decimated by DDC to a much lower baseband IQ rate, ranging from 1 MHz to 25 MHz, as shown in Figure 12. ADC_0_Port_A_Clk can be either compiled for a fixed frequency or made tunable from the host at run time. When configured as tunable, its value is obtained from

\[
ADC_0_{\text{Port}_A}\_Clk = \frac{ENC_{\text{ADC}_0} \times M}{N \times Decimation_{\text{ADC}_0}}, \tag{32}
\]

where ENC_{\text{ADC}_0} equals the device reference clock divided by \( N \); M, the clock multiplier for AI channel 0 ADC, equals 1 or an integer between 4 and 20; N is the predivide factor that equals 1, 2, 4, or 8; Decimation_{\text{ADC}_0} is the decimation rate of the DDC of AI channel 0. All these values can be configured in host at run time. ADC_1_{\text{Port}_A}\_Clk is the IQ clock for AI channel 1 and can be set up the same way as ADC_0_{\text{Port}_A}\_Clk.

DAC_0_{\text{IQ}_C}\_Clk and DAC_1_{\text{IQ}_C}\_Clk are IQ clocks for AO channels. They specify the data rate before DUC. Since AO channels are not used in this project, we will not elaborate on these two clocks.
RTSI_Ref.Clk, like ADC_0.Port_A.Clk, can be either compiled as a fixed frequency or as tunable from host. Its value is equal to the device reference clock divided by $N_4$, which can be 1, 2, 4, 8, or 16. This clock is usually customized to be used to control the timing of FPGA programs because it is configurable and will not affect other hardware functionality.

### 3.1.2 Main processing loops

We start this section with the introduction of some terminologies that are widely used to describe LabVIEW FPGA programs. According to [6], Direct Memory Access (DMA) is defined as a method by which one can transfer data to computer memory from a device, also called target, or from computer memory to a device, while the processor remains free for other tasks. FIFO refers to block memory that implements a first-in-first-out data exchange policy. Data to be transferred within FPGA can be stored in target-scoped FIFO; data to be transferred between host and FPGA can be stored in host-to-target DMA FIFO or target-to-host DMA FIFO depending the transportation direction.

All digital signal processing is implemented on FPGA within three parallel processing loops. The first loop is called the acquisition loop, as shown in Figure 13, where IQ data are collected from analog input ports and written into target-scoped FIFO. Phase detector is implemented in this loop. The Boolean variable “CORDIC?” controls whether Modified Costas phase detector or CORDIC phase detector should be used in PLL. For every incoming
IQ pair, one phase error value is computed and stored in a local variable. Then the phase error value is passed to the DIO loop to compute the control signal $c[n]$. But most of the phase error values will be discarded due to the relatively slow execution rate of the DIO loop. The acquisition loop is a single-cycle timed loop, which means its execution time can be precisely controlled by the clock wired to the input node. In other words, the user can slow down the processing rate by skipping samples, regardless of IQ clock rate. In this case, the loop is clocked with ADC_0_Port_A_CLK at 1 MHz, which means every IQ sample acquired is passed through for processing.

Figure 14 shows the data transfer loop, which simply moves data from target-scoped FIFO to target-to-host DMA FIFO. Simple manipulation of data, such as scaling, can be done in this loop sample-by-sample before they are sent to the host. Host will access target-to-host DMA FIFO and retrieve a large chunk of samples every time. This loop must run faster than the acquisition loop to make sure that the target-scope FIFO does not overflow. In our project, this loop is configured to run at 4 MHz.

The third loop is called DIO loop which encloses loop filter $L(z)$ and the DDS-FPGA interface. As mentioned before, IF-RIO has 7 digital output lines, 5 of which are used for SPI communication with DDS. See Table 1 for the pin descriptions.

According to DDS specifications, each communication cycle consists of two phases. Phase 1 is the instruction cycle, which is the writing of an instruction byte, coincident with the first eight SCLK rising edges; phase 2 is
the actual data transfer between AD9954 and the system controller. The number of bytes transferred during phase 2 of the communication cycle is a function of the register being accessed. Most of the time, we only use the FPGA-DDS interface to update FTW. In this case, the data packet consists of 8 bits of instruction and 32 bits of FTW value. The following discussion will be based on 40-bit data packet transfer. At the beginning of each communication cycle, CS is asserted and the first bit in the data packet is loaded on the data line, entering the WHILE loop. The WHILE loop consists of a three-step sequence. The first step is intended to allow the data bit to stabilize on the data line. In step 2, SCLK is set high and DDS starts to read the data bit. SCLK is set low in step 3. The WHILE loop is repeated for 40 times. SCLK rate and duty cycle can be adjusted by changing the loop timer value in step 1 and wait timer value in step 2. At the completion of a communication cycle, I/O update is asserted to transfer internal buffer contents into control registers on DDS. Then, CS, SPI DOUT, and I/O update are set to idle state, waiting for the next communication cycle. The timing diagram can be found in [7]. Figure 15 shows the implementation of the SPI

<table>
<thead>
<tr>
<th>Pin number</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>data line (SPI DOUT)</td>
</tr>
<tr>
<td>3</td>
<td>reset (RESET)</td>
</tr>
<tr>
<td>4</td>
<td>chip select (CS)</td>
</tr>
<tr>
<td>5</td>
<td>serial clock (SPI_SCLK)</td>
</tr>
<tr>
<td>6</td>
<td>I/O update</td>
</tr>
<tr>
<td>1,7</td>
<td>not used</td>
</tr>
</tbody>
</table>
interface on LabVIEW FPGA. Figure 16 shows an oscilloscope acquisition of FPGA digital output.

Precise time control of this loop is desirable but not possible because single-cycle timed loop does not allow WHILE loop nested inside. Fortunately, this does not cause a problem because FPGA programs, once compiled, will maintain deterministic execution as long as functions to be performed require constant clock cycles. The WHILE loop placed without timing defined will run on Configuration_Clk of 20 MHz. This argument is supported by experimental measurement of SPL_SCLK. As shown in Figure 17, the duration between two communication cycles, i.e. 40 SPL_SCLK cycles followed by an idle period, is repeatedly measured to be 35.1 µs. From the perspective of system characterization, FTW is updated once per 35.1 µs, meaning that the sampling rate of the PLL is \( \frac{1}{35.1 \mu s} = 28.5 \text{ KHz} \), i.e \( T_s = 35.1 \mu s \).

Three types of loop filters are implemented in this loop. Two of them have been described in Section 2.1. The third one is a three-stage Cascaded integrator-comb (CIC) filter which is introduced to exemplify a computationally efficient implementation of narrowband lowpass filter. It should be noted that all quantitative analysis of second-order PLL in the following sections are based on PI filter. Since FPGA has limited computation ability, all math related functions must be implemented with fixed-point arithmetic. The size of each fixed-point variable must be carefully selected to strike a balance between high throughput and precision. Figures 18, 19, and 20 show the implementation of each loop filter, respectively.
3.2 Other hardware

A PCB-based crystal oscillator with 20 MHz output frequency is used as the reference clock for AD9954 DDS. As described in [8], the oscillator circuit consists of a Common-collector BJT in Colpitts configuration and another BJT as emitter follower to match the output impedance to 50 Ω. A ceramic filter centered at 10.7 MHz with a bandwidth of 200 KHz is used as IF filter. It is soldered on the PCB with two L-networks that match its input and output impedance to 50 Ω. Mini-Circuits ZP-3LH is the mixer that shifts the input signal from 70 MHz to 10.7 MHz. An Agilent E3648A DC power supply provides a +12 V DC voltage for BJTs in the crystal oscillator. This voltage is stepped down to 3.3 V and 1.8 V by two low drop-out (LDO) linear regulators to provide DC power for AD9954 DDS.

3.3 Modeling of actual system

In this section, we will analyze the actual system and characterize it using the theoretical model developed in Chapter 2. Figure 21 contains the block diagram of the actual system.

Incoming analog signal is

\[ x(t) = \cos(2\pi f_c t + \theta_c(t)). \]  

(33)

It is mixed with the DDS output \( \cos(2\pi f_o t + \theta_o(t)) \) and shifted down to
intermediate frequency (IF) and then passed through a bandpass filter (BPF) that cuts off high frequency components. It becomes

\[ x'(t) = a_{km}k_f \cos(2\pi(f_c - f_o)t + \theta_c(t) - \theta_o(t)), \tag{34} \]

where \( k_m \) is the attenuation in voltage of the mixer and \( k_f \) is the attenuation in voltage at the passband of the filter.

The signal then is digitized at IF-RIO analog input and converted to baseband IQ data.

\[
I[n] = a_{km}k_f k_{ADC} \cos(2\pi(f_c - f_o - f_{NCO})nT_s + \ldots + \theta_c[n] - \theta_o[n] - \theta_{NCO}[n]), \tag{35}
\]

\[
Q[n] = a_{km}k_f k_{ADC} \sin(2\pi(f_c - f_o - f_{NCO})nT_s + \ldots + \theta_c[n] - \theta_o[n] - \theta_{NCO}[n]), \tag{36}
\]

where \( f_{NCO} \) is the frequency of the Numerically-Controlled Oscillator (NCO) that is embedded in the NI PCI-5640r AI channel. This value is configurable from host in real time. In this case, it is set nominally to 10.7 MHz so that the input signal is shifted from IF to baseband. \( \theta_{NCO} \) is the phase of the NCO. We define \( k = a_{km}k_f k_{ADC} \) and actual phase error is \( \varepsilon[n] = \theta_c[n] - \theta_o[n] - \theta_{NCO}[n] \).

At steady state, when \( f_c = f_o + f_{NCO} \), Equations (35) and (36) can be simplified as
\[ I[n] = k\cos(\varepsilon[n]) = k\cos(\varepsilon[n]), \]  
\[ Q[n] = k\sin(\varepsilon[n]) = k\sin(\varepsilon[n]). \]

The phase detector outputs an estimate of the phase error, \( \varepsilon[n] \), the accuracy depending on the range of input phase error and the type of phase detector. For the Modified Costas phase detector, its output is given as

\[ \varepsilon[n] = sgn(I[n])Q[n] = k_D \varepsilon[n], \]  
where

\[ k_D = \frac{k\sin(\varepsilon[n])}{\varepsilon[n]}, \text{ for } -\pi \frac{n}{2} + n\pi < |\varepsilon[n]| < \pi \frac{n}{2} + n\pi, \ n \in \mathbb{Z}. \]  

For the CORDIC phase detector, its output is given as

\[ \varepsilon[n] = \begin{cases} 
  k_D \varepsilon[n] & \text{if } |\varepsilon[n]| < 0.09579 \\
  \pm 0.09579k_D & \text{else}
\end{cases}, \]  
where

\[ k_D = \frac{2^{15} - 1}{\frac{\pi}{2}} = 20860. \]
3.4 Figures

Figure 9: Block diagram of NI 5640r, cited from [5]
Figure 10: FPGA base clocks, modified from [6]

Figure 11: IF-RIO clock resources, modified from [6]
Figure 12: Analog I/O channel block diagram, cited from [6]

Figure 13: Acquisition loop
Figure 14: Data transfer loop

Figure 15: SPI interface
Figure 16: Real-time display of FPGA digital output. Channel 1 is SCLK; Channel 2 is the data line.

Figure 17: Measurement of $T_s$
Figure 18: Constant gain

Figure 19: CIC filter
Figure 20: PI filter

Figure 21: Block diagram of the actual system
4 Experiment and Characterization

4.1 Constant parameters in experiment setup

A 0 dBm 70 MHz sine wave is used as the input signal. Then $a = \frac{1}{2} V_{pp} = 0.3168$. Mixer conversion loss is measured to be 5.5 dB at 10.7 MHz, indicating an input output voltage ratio of $k_m = 0.53$. IF filter insertion loss is measured to be 2.8 dB at 10.7 MHz, indicating an input output voltage ratio of $k_f = 0.725$ at passband. According to [9], full-scale input range of the NI PCI-5640r is 8.5 dBm at 10 MHz and baseband IQ data are represented as signed 16-bit numbers. Therefore, $k_{ADC} = \frac{2^{15} - 1}{0.8414} = 38943$. Combining all aforementioned gains gives $k = a k_m k_f k_{ADC} = 4741$, which will be used to compute the gain of Modified Costas phase detector. According to Equations (39) and (40), the output of the Modified Costas phase detector can be represented as $\epsilon[n] = sgn(I[n])Q[n] = \pm a k_m k_f k_{ADC} \sin(\epsilon[n]) = \pm 4741 \sin(\epsilon[n])$.

In Section 2.3 and Section 3.1, we have obtained $k_O = 0.5849$ and $T_s = 35.1 \mu s$, respectively. Plugging in constants from different stages to Equations (5) and (13), the phase transfer functions of the actual system are given as follows:

for first-order PLL with the Modified Costas phase detector,

$$H(z) = \frac{k_L k_O k_D T_s}{k_L k_O k_D T_s + z - 1} = \frac{0.0968 k_L \sin(\epsilon[n])}{0.0968 k_L \sin(\epsilon[n]) + z - 1}; \quad (43)$$

for first-order PLL with the CORDIC phase detector,
for second-order PLL with Modified Costas phase detector,

$$H(z) = \frac{k_L k_O k_D T_s}{k_L k_O k_D T_s + z - 1} = \frac{0.4246 k_L \sin(\varepsilon[n])}{0.4246 k_L \sin(\varepsilon[n]) + z - 1}; \quad (44)$$

for second-order PLL with CORDIC phase detector,

$$H(z) = \frac{k_L k_O k_D T_s (z + \alpha)}{z^2 + (k_L k_O k_D T_s - 2)z + k_L k_O k_D T_s \alpha + 1},$$

$$= \frac{0.0968 k_L \sin(\varepsilon[n]) (z + \alpha)}{z^2 + (0.0968 k_L \sin(\varepsilon[n]) - 2)z + 0.0968 k_L \sin(\varepsilon[n]) \alpha + 1}; \quad (45)$$

4.2 Response to step stimulus

It is instructive to study the PLL response when it is subject to a step change input in phase. As examined in Section 2.1, if the step deviation is within lock range, PLL should be able to track the change after a transient period, the length of which is determined by loop filter parameters.

The most intuitive way to measure the step response is to introduce a known step change in the input signal and measure the phase response at the output. However, this method requires measuring the input signal and the PLL response in the same time domain. A two-channel VSA would be able to carry out this task. Due to the lack of the above mentioned equipment, we use an alternative but equivalent way to characterize it.
Figure 22 shows the block diagram of the experiment setup. A zero-offset square wave stimulus with tunable amplitude is added to the control signal $c[n]$. In a linear feedback system, this is equivalent to introducing a repeated step change in the input frequency. The square wave frequency is chosen to be as low as 14 Hz, so that PLL is guaranteed to reach steady state before the next step change takes place. The control signal $c[n]$ is measured as the PLL response phase.

Figures 23 and 24 show a few measured curves with different choices of loop filter type and parameters. Different $k_L$ are chosen for PLL implemented with the Modified Costas PD and for PLL implemented with the CORDIC phase detector to offset the effect of their different phase detector gains. As a result, both PLLs have the same cumulative gain, $K_L$, which is defined as $K_L = k_L k_O k_D$. Measurement results are graphed together with their corresponding MATLAB simulations to verify how well the models we developed in the Section 3.3 describe the actual system. The following lines of code, modified from [10], are used to generate the MATLAB simulation in Figure 24 for the second-order PLL.

```matlab
data_size = length(measured_data);
Ts = 35.1e-6;
z = tf('z',Ts);
alpha = -0.994036
K_D = 4741;
K_O = 0.5849
```
K_L = 0.720648;
K = K_D*K_O*K_L*T_s;
H = K*z^{(-1)}*(1+alpha*z^{(-1)})/(1+(K-2)*z^{(-1)}+(K*alpha +1)*z^{(-2)});
step(H, data_size*T_s);

Both figures indicate that our analytical models characterize the real system very well. As shown in Figure 25, PLL implemented with CORDIC phase detector shows sawtooth-like response curve and oscillates between two levels at steady state. This phenomenon is expected due to the quantization effect inherent to the CORDIC algorithm and can be improved by increasing iteration times. PLL with the CORDIC phase detector matches simulation better because the Modified Costas phase detector is not strictly linear, especially when the input phase error is large. This explains why the dashed line has a relatively large discrepancy in the overshoot region, but it is still able to match the simulation closely when it just takes off at the beginning and reaches steady state in the end.

4.3 Response to sinusoidal stimulus

A sinusoidal stimulus is fed into the system in order to characterize the phase transfer function of the PLL. The same phase transfer function also relates stimulus v[n] to PLL response c[n]. Equation (47) can be used to measure the magnitude of the phase transfer function, |H(jω)|.
\[ |H(j\omega)| = \left| \frac{C(j\omega)}{V(j\omega)} \right|. \] (47)

The experiment setup is similar to the one used in Section 4.2 except that the stimulus is replaced with a sinusoidal wave generator, which, implemented on FPGA, is simply a look-up table consisting of two periods of sinusoid pattern. The table size is 1024. The frequency of the sine stimulus can be represented as \( \frac{nf_s}{512} \), where \( n \) is the iterator step size, which can be controlled from host, and \( f_s \) is the sampling frequency of the PLL. There are two conditions for \( n \): first, it has to be an integer since it corresponds to the look-up table address; second, it has to be relatively small since skipping too many samples leads to severely deformed output waveforms. An obvious drawback of this method is that the total number of realizable frequency levels is limited. The NI FPGA offers an option to linearly interpolate between look-up table entries if the input address is a noninteger. But this is not practical in our system. As mentioned in Section 3.1.2, the sampling frequency of PLL is determined by the execution rate of the WHILE loop that encloses the loop filter and FPGA-DDS interface. The timing of this loop is not precisely controlled by any system clocks. Therefore, its execution rate will be consistent only if every process inside this loop uses constant clock cycle to execute. However, the execution time of an interpolation-enabled look-up table varies depending on address input: it takes one clock cycle for integer addresses and at least two clock cycles for noninteger addresses when
interpolation is needed. Nonuniform sample intervals of PLL will result if
the look-up table is operating in interpolation mode.

Figures 26 and 27 show measured phase transfer function curves for first-order and second-order PLL, respectively, graphed with MATLAB simula-
tions. Data are collected at low frequencies for the reasons mentioned above.

4.4 Processing lag

The experiment described in Section 4.2 can also be used to measure FPGA processing lag. It can be seen in Figures 28 and 29 that PLL starts
to respond to step stimulus after two samples. Thus processing delay can be calculated as

\[ \tau = 2T_s = 2 \times 35.1 \mu s = 70.2 \mu s. \] (48)

Processing delay is inherent to all digital systems. In this case, it is
determined by the amount of operations performed on FPGA and the FPGA
processing speed. To make our system model more accurate, we have to
account for this delay by grouping it with the cumulative gain \( K_L = k_L k_O k_D \).
For the first-order PLL, the phase transfer function becomes

\[
H(z) = \frac{z^{-2}K_LT_s(z + \alpha)}{z^2 + (z^{-2}K_LT_s - 2)z + z^{-2}K_LT_s\alpha + 1} = \frac{K_LT_s z^{-3} + \alpha z^{-2}K_LT_s z^{-4}}{1 - 2z^{-1} + z^{-2} + K_LT_s z^{-3} + K_LT_s\alpha z^{-4}} \] (49)
We redo the step response simulation in MATLAB with the modified phase transfer function. As before, the same parameters are used to facilitate comparison of the two simulations. Simulations of both system functions are graphed together with measurement data in Figure 30. As shown in Figure 31, the simulation of the modified phase transfer function has smaller discrepancy with the measurement data.

4.5 Lock range

In this section, we will present experimental measurement results of the lock range of different types of PLLs and evaluate how well these values agree with the theoretical calculation given in Section 2.1.

For PLL with the Modified Costas PD, \( k_L = 0.6 \). The critical condition for lock-in status is that the magnitude of steady-state phase error equals \( \frac{\pi}{2} \). Substituting Equation (40) in Equation (11), we obtain

\[
|\Delta f_c| = \frac{k \sin(\frac{\pi}{2}) k_D k_O k_L}{\frac{\pi}{2} \cdot 4} = 264.8 \text{ Hz}
\]  

(50)

The lock range was measured to be from \(-259.2 \text{ Hz}\) to \(257 \text{ Hz}\), which agrees with Equation (50) with a 1.8% error.

For PLL with the CORDIC PD, \( k_L = 0.14 \). The largest frequency offset that can be tolerated in the PLL before losing lock is given as

\[
|\Delta f_c| = 0.1525K_L = 0.1525k_D k_O k_L = 260.5 \text{ Hz}.
\]  

(51)
The experimental result indicates a lock range from −257.6 Hz to 256.2 Hz, which agrees with Equation (51) with a 1.5% error.

The statement we made in Section 2.1 that second-order PLL has infinite lock range has been verified by experiment. In a real system, the limiting factors that prevent the lock range going to true infinity are the sampling rate of the system and the size of the register that holds the control signal $c[n]$. 
4.6 Figures

Figure 22: Step response block diagram

Figure 23: The step response of first-order PLL
Figure 24: The step response of second-order PLL

Figure 25: Zoom-in of Figure 24
Figure 26: Magnitude of phase transfer function of first-order PLL

Figure 27: Magnitude of phase transfer function of second-order PLL
Figure 28: Processing delay measurement using step stimulus

Figure 29: Zoom-in of Figure 28
Figure 30: Step response of second order PLL with modified phase transfer function

Figure 31: Zoom-in of Figure 29
5 Conclusion

The thesis presents a working digital PLL with two types of phase detectors and three types of loop filters. This project is based on the NI PCI-5640r and programmed with LabVIEW. The AD9954 DDS, used as a digital VCO, is interfaced with the FPGA of the NI PCI-5640r via serial I/O. The system is characterized numerically with in-lab measurements, the results of which are compared with theoretical models to verify that the PLL operates as expected.

This project will serve as a lab module for ECE 463 in the future. It can be used in a transmitter to generate stable carrier frequencies and in a coherent receiver to recover the carrier of the input signal. Furthermore, the FPGA-DDS interface developed in this project provides a convenient platform for future research efforts that involve the usage of DDS. DDS can be easily controlled by programming or in real-time with LabVIEW and the NI PCI-5640r.
References


