Spring 2013

Novel Systematic Phase Noise Reduction Techniques for Phase Interpolator Clock and Data Recovery

Yu Feng
San Jose State University

NOVEL SYSTEMATIC PHASE NOISE REDUCTION TECHNIQUES FOR PHASE INTERPOLATOR CLOCK AND DATA RECOVERY

A Thesis

Presented to

The Faculty of the Department of Electrical Engineering

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Yu M. Feng

May 2013

The Designated Thesis Committee Approves the Thesis Titled

NOVEL SYSTEMATIC PHASE NOISE REDUCTION TECHNIQUES FOR PHASE INTERPOLATOR CLOCK AND DATA RECOVERY

by

Yu M. Feng

APPROVED FOR THE DEPARTMENT OF ELECTRICAL ENGINEERING

SAN JOSÉ STATE UNIVERSITY

May 2013

Dr. Shahab Ardalan	Department of Electrical Engineering
Dr. Sotoudeh Hamedi-Hagh	Department of Electrical Engineering
Prof. Morris Jones	Department of Electrical Engineering

ABSTRACT

NOVEL SYSTEMATIC PHASE NOISE REDUCTION TECHNIQUES FOR PHASE INTERPOLATOR CLOCK AND DATA RECOVERY

by Yu M. Feng

This work focused on high-speed source-synchronous clock and multi-channel data receivers for inter-chip communications. Designs of inter-chip communication are becoming increasingly difficult with the rise in clock rates and the reduction in voltage supplies. Data transmissions at rates of gigabits per second require a fast and accurate clock and data recovery system on the front end of receivers.

Many designs allow for source-synchronous clocking architectures, but this work focused on a dual-loop with a phase-locked loop for frequency tracking and phase integrators for tracking each individual data lane. Limitations with the phase interpolator architecture cause systematic jitter, reducing the data eye.

Various techniques exist that aim to reduce or eliminate this systematic jitter from phase interpolator architectures. A technique based on digital lock detection was developed for this work that eliminates the phase interpolator systematic jitter.

ACKNOLWEDGEMENTS

I would very much like to extend my gratitude towards Dr. Shahab Ardalan for his guidance and feedback throughout this research. You have challenged my mind to expand my thinking into more abstract and system-oriented terms. I find myself gleaning deeper meaning from all my past education with this new mindset. My only regret is not being able to show you more of my ability in design and leaving a taped-out parting gift.

I am grateful for having such great colleagues at Dr. Ardalan's AMS lab. I cannot even count the number of times where bouncing ideas off of one another has led to different angles with which to approach roadblocks in thinking.

This thesis would never have reached much cohesion or any semblance of concise thinking had Angelo Volpe not nearly failed me in freshman composition. A large part of my success in this field is from submitting short, compact reports. I continue to spread his style of persuasive and argumentative writing with each keystroke and each paper I proofread.

Finally, I reserve my most profound appreciation for my family. Without the daily distractions provided by my dear sisters, brothers, and cousins, this work would have been completed years ago. However, accomplishing that without you guys behind me would have been meaningless. My education in this field would also have never amounted to much without the unwavering push from my parents. This work is dedicated to all the warmth and support I have received from all of you.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1. Introduction

Advancements in technology tend to aim at accomplishing more tasks as quickly as possible. In 2012, Cisco estimated an average 885 petabytes per month of global mobile data traffic consisting of laptops, tablets, smartphones, and machine-to-machine devices; an average monthly usage of 11.2 exabytes is projected for 2017 [1]. The number of online domains increased from 19.8 million in 1997 to 908 million in 2012 [2]. As data transferring grows in both quantity and speed, hardware is also advancing.

As computer processors with upwards of 12 cores approached 3.4 GHz clock speeds in recent years, the amount of on-chip data sent to peripheral components demanded for faster chip-to-chip communication [3]. Many important technologies, such as smartphones, computers, and high definition televisions, rely on high speed random access memory (RAM). The JEDEC standard for the still in-development DDR4 is specified for 2.133-4.266 GT/s per lane [4]. As data transfers become increasingly bottlenecked by communication links, effort is spent on increasing efficiency and minimizing error.

The high-speed link for source-synchronous clocks with multiple data channels, shown in Figure 1.1 [5], is used in technologies such as RAM, graphics card RAM (GDDR), and backplane servers [6]. On the receiving side, a clock and data recovery (CDR) circuit resynchronizes the data with the accompanying clock.

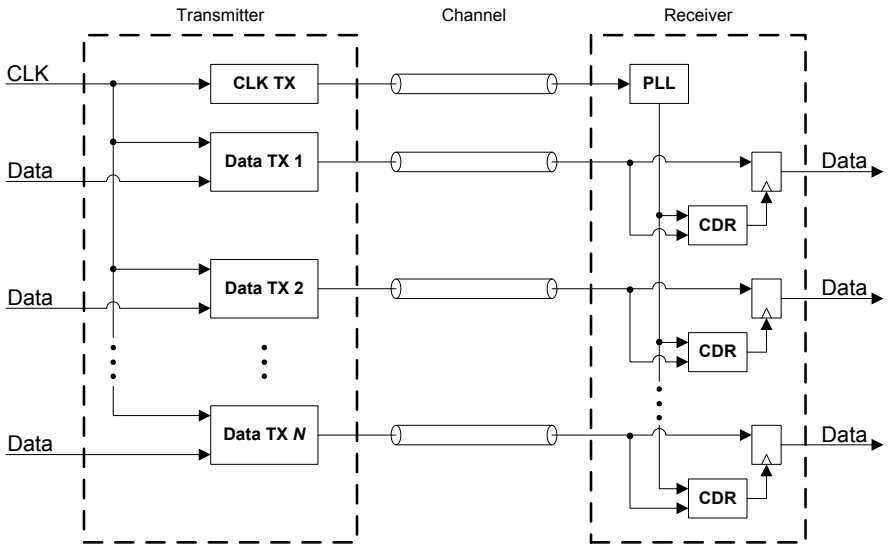

Figure 1.1: Example of a source-synchronously clocked high speed link

1.1. Clock and Data Recovery

In order to minimize error, the receiver must be able to align the clock to the optimal sampling point on the data stream. Ideally, the clock should align with the largest opening in the data eye diagram, shown in Figure 1.2 [7]. An eye diagram, generated by overlaying all data transitions within a single frame, allows designers to see effects of jitter and timing mismatch. This realignment is the function of a CDR.

There are many CDR architectures including phase-locked loops (PLL), delay-locked loops (DLL), or phase interpolators (PI) [5]. Each architecture has its strengths and limitations, but due to the nature of multi-lane data transfer with source-sychronized clocking, this work focused on the PI architecture [8].

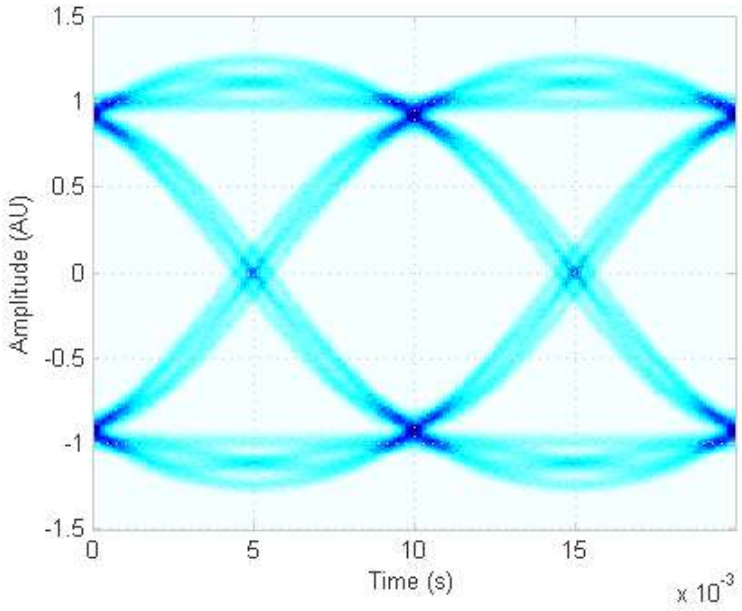

Figure 1.2: Example of an eye diagram

1.1.1. Phase Interpolator

Figure 1.3 is a block diagram of a standard phase interpolator-based CDR [8]. It consists of a local PLL to lock frequency and a separate PI for each lane of data.

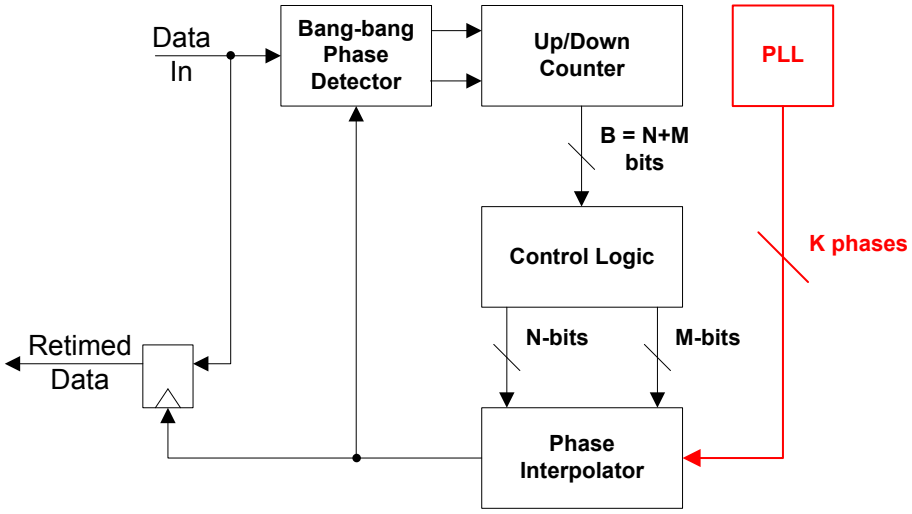

Figure 1.3: Standard phase interpolator CDR

This is a dual-loop architecture, separating the tasks of frequency tracking and high-speed phase tracking. The primary control loop is digital, allowing it accuracy, speed, and portability across processes. The bang-bang phase detector (!!PD) detects whether the recovered clock is ahead or behind the data stream in a binary fashion [9], giving the PI a very quick phase tracking.

1.1.2. Systematic Jitter

The nature of the phase detector gives rise to systematic jitter once the CDR has locked. Without any filtering on the counter-based logic, the PI can make a tracking decision each clock cycle. Once the system is in lock, the !!PD continuously detects early or late, causing the recovered clock to toggle between a few states. This jitter effectively lessens the data eye opening, demonstrated in Figure 1.4.

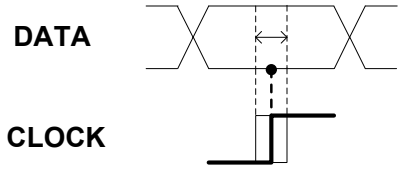

Figure 1.4: Effect of systematic jitter on data eye

1.2. Motivation

While it is not difficult to increase the clock speed of high speed links, random and deterministic jitter dominate the eye opening. In order to increase the speed of high speed links, techniques must be developed to increase efficiency and accuracy. Using the fast-tracking PI architecture as a base, some techniques are explored to reduce systematic jitter.

The first utilizes a ΔΣ modulator in an attempt to shape and filter the noise power and the second technique compromises jitter tolerance in order to eliminate the in-lock jitter.

1.3. Organization

Chapter 2 reviews background information on various CDR building blocks for the PLL-PI dual loop architecture. Each building block of the CDR is examined for points of improvement. The concept and stability of ΔΣ modulation, applications of ΔΣ, and limitations are also explored. Chapter 3 introduces several existing techniques for reducing systematic jitter in CDR. Finally, Chapter 4 presents a novel lock detection PI architecture that eliminates systematic jitter.

Chapter 2. Background

Clock and data recovery circuits have many available topologies, but the heart of all CDR operations lie within the phase-tracking mechanism. Simple single-loop architectures feature a PLL for frequency and phase tracking while dual-loop architectures track frequency and phase separately [5]. This chapter details the functionality of a PLL and explores the dual-loop PLL-PI CDR architecture.

2.1. Phase-Locked Loops

A PLL is a negative feedback system that continuously adjusts a voltage-controlled oscillator (VCO) frequency in order to match an input reference clock [10]. Figure 2.1 shows a generalized charge pump PLL. The basic building blocks consist of a phase-frequency detector (PFD), a charge pump and low-pass loop filter (LF), a VCO, and a feedback divider. For simplicity of analysis, the divider N is chosen to be unity.

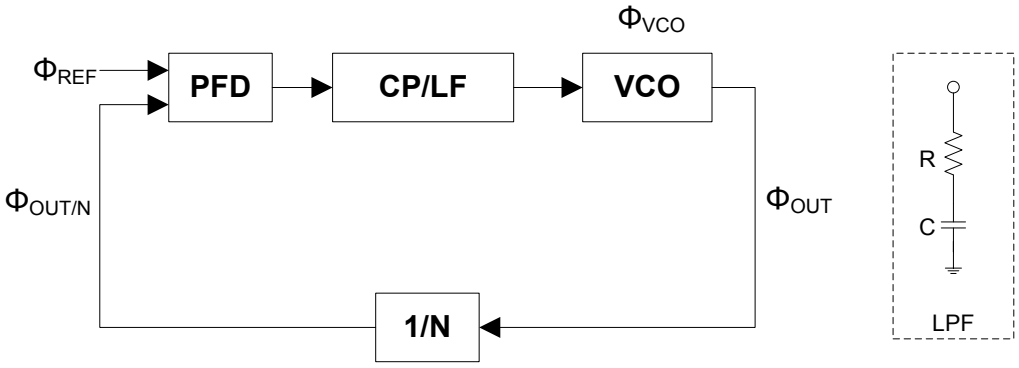

Figure 2.1: Generalized charge pump PLL

The phase detector takes in a reference clock and compares that to the feedback clock from the VCO to generate an "early" (*DN* signal) or "late" (*UP* signal) output. The charge pump and loop filter takes the *UP* and *DN* signals and raises or lowers the control voltage, respectively. Finally, this control voltage adjusts the VCO output clock frequency. Once the system is in a closed loop, the VCO continuously adjusts itself to remain locked to the reference.

2.1.1. Phase-Frequency Detector

The PFD compares two input clocks and generates an output signifying which signal is faster or slower [10]. The output of the PFD is filtered into a control signal to provide a negative feedback for the VCO. A very simplistic detector can be implemented with an XOR gate. The output is high in proportion to the duration that the signals are mismatched, demonstrated in Figure 2.2.

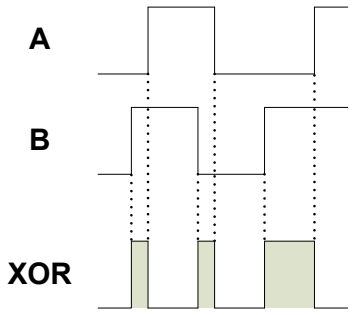

Figure 2.2: Input-output behavior of XOR detector

However, the XOR implementation does not distinguish between rising or falling clock edges. This type of detector can only produce a pulse proportional to the phase

difference in two clock edges. For a PLL, the PFD has to be able to distinguish "early" from "late." These detectors are split into two main types: linear and binary.

2.1.1.1. Linear Detector

The linear PFD, shown in Figure 2.3, outputs a signal "early" or "late" in linear proportion to how much the signals' phases differ [10]. The two D-flipflops (DFF) have their D-inputs tied high and the signals to be matched are connected to their clock. When either Φ_{REF} or Φ_{FB} has a rising edge, the output of the respective DFF goes high. After the second signal has a rising edge, the second DFF also goes high, sending a reset after a time t_{del}. The timing diagram of the signals is given in Figure 2.4.

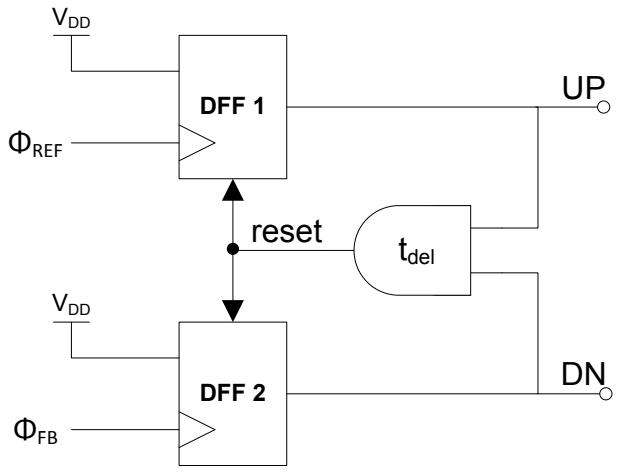

Figure 2.3: Basic linear PFD

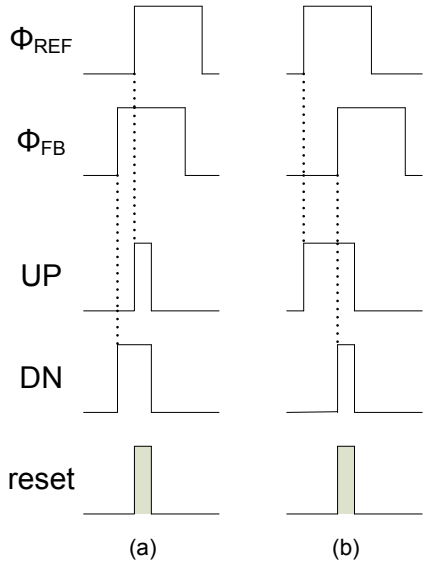

Figure 2.4: Timing diagram of a linear PFD

The duration of the *UP/DN* outputs are thus linearly proportional to the magnitude of the $\Delta\Phi$ between the two inputs. Figure 2.5 is an example of the ideal characteristic plot of the input $\Delta\Phi$ versus output voltage of a typical linear detector.

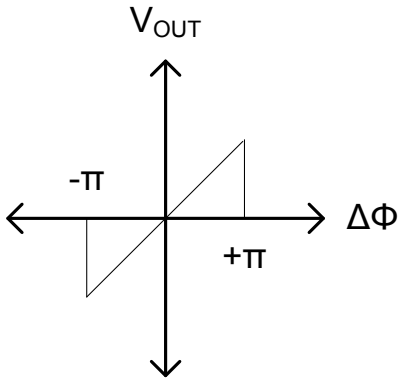

Figure 2.5: Input-output characteristics of a linear detector

2.1.1.2. Binary Detector

While the linear detector requires analog components (charge pump and analog loop filter) to create a final control voltage, the binary detector simply outputs a digital code for *UP* or *DN* given a positive or negative $\Delta\Phi$. True to the name, this detector has two outputs irrespective of its $\Delta\Phi$ magnitude. The simplicity and function of the binary, or bang-bang phase detector (!!PD), is best utilized in all-digital PLL/DLLs or CDRs.

For an all-digital system, the loop filter can be constructed to have a proportional and integral dual path to behave similarly to a linear PFD [11]. In a CDR, the data stream is random, thus requiring a phase detector that locks and retimes to a non-periodic reference. The simplest way to implement this is the Alexander phase detector, shown in Figure 2.6.

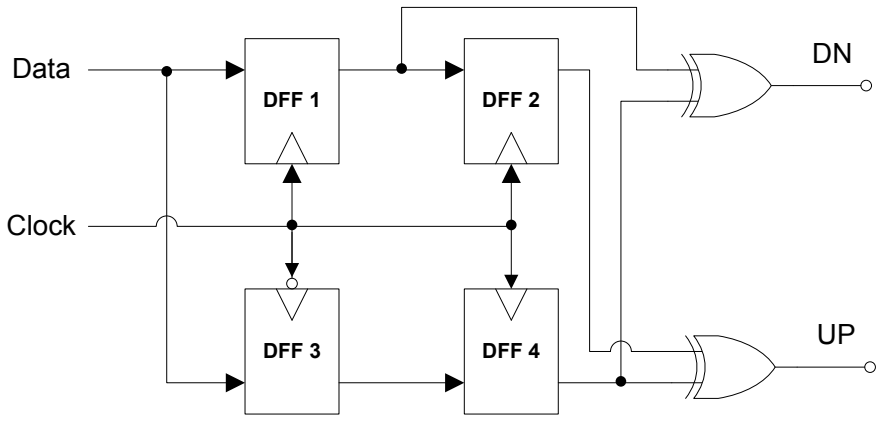

Figure 2.6: Alexander phase detector

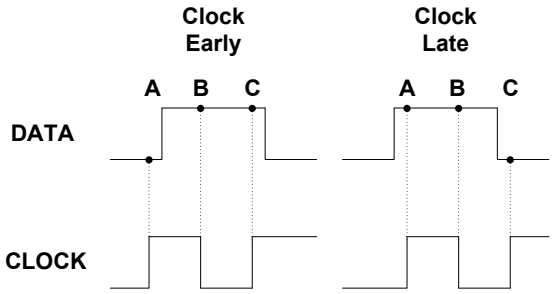

Figure 2.7: Sample points for Alexander PD

Detailed in [9], the data are sampled on three consecutive rising and falling clock edges, shown in Figure 2.7. The sampled values, *A, B,* and *C*, generate the *UP/DN* decisions given in Figure 2.8. By design, this detector can output a "no-change" state, *NC*. However, this state does not distinguish between no data transition, meta-stable samples, or in-lock status.

A	*B*	*C*	*Output*
0	0	0	NC
0	0	1	UP
0	1	0	NC
0	1	1	DN
1	0	0	DN
1	0	1	NC
1	1	0	UP
1	1	1	NC

Figure 2.8: Alexander PD truth table

2.1.2. Charge Pump and Loop Filter

The outputs of the linear PFD are two pulses signifying either *UP* or *DN* proportionally wide as the magnitude of the input $\Delta\Phi$. A mechanism following this block must convert these signals from units of time into voltage. As given in Eq. (2.1), a

capacitor can convert a time-based current into a voltage. This simplifies the block following the PFD into a means of generating a current for a given duration.

$$I_C = C\frac{\partial V}{\partial t} \Rightarrow V_C = \int \frac{I_C}{C}\partial t \tag{2.1}$$

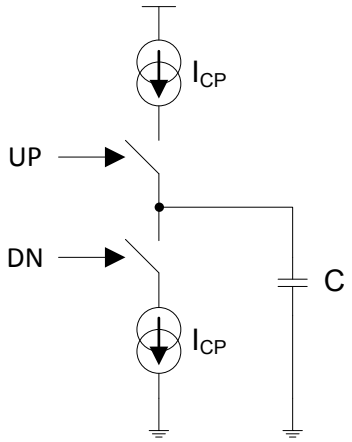

Figure 2.9: Typical charge pump block

A typical charge pump is shown in Figure 2.9. The *UP/DN* signals from the PFD now convert a time-based digital signal into time-based currents of opposite magnitudes. The current sources are matched so an *UP* pulse of equal duration to a *DN* pulse has zero net effect on the voltage stored by the capacitor. While this method proportionally changes the voltage on the capacitor, it does not allow for enough degrees of design freedom in the overall PLL.

In order to separate the definitions of the design parameters ω_n and ζ, a resistor R has to be placed in series with the capacitor C. Since the integration causes large rippling of the voltage on the capacitor, a smoothing capacitor, C_2, is placed in parallel. The final loop filter is shown in Figure 2.10 and the overall transfer function is:

$$Z_{LPF} = \frac{sRC_1 + 1}{s^2 RC_1 C_2 + s(C_1 + C_2)} \tag{2.2}$$

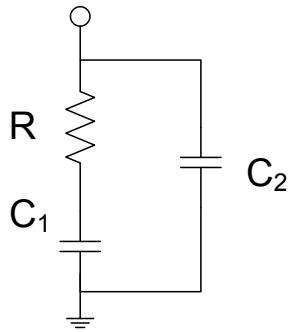

Figure 2.10: Loop filter with smoothing capacitor C_2

If C_1 is five to ten times larger than C_2, then the loop filter still approximately behaves as a first order low-pass [10].

2.1.3. Voltage-Controlled Oscillator

If the blocks thus far can be considered the brain of the PLL, the VCO functions as the body. As the name implies, the VCO is an oscillator outputting a clock whose frequency is scalable by a control voltage. An oscillator can simply be implemented as an amplifier purposely driven into instability. The Barkhausen criteria, Eq. (2.3), is the minimum requirement for oscillation. This is given for a negative feedback system, implying a total of 360° (or 0°) phase shift.

$$\begin{aligned} |H(j\omega_0)| &\geq 1 \\ \angle H(j\omega_0) &= 180° \end{aligned} \tag{2.3}$$

While an LC oscillator generates lower phase noise, it also requires an inductor and capacitor to set a resonance frequency. Not only does this method take up a lot of silicon area, the design is generally limited to a single LC oscillator per die due to substrate noise. The tunable range of an LC oscillator is limited by the varactor technology. Finally, a phase interpolator requires multiple clock phases so it is most advantageous to use a ring oscillator.

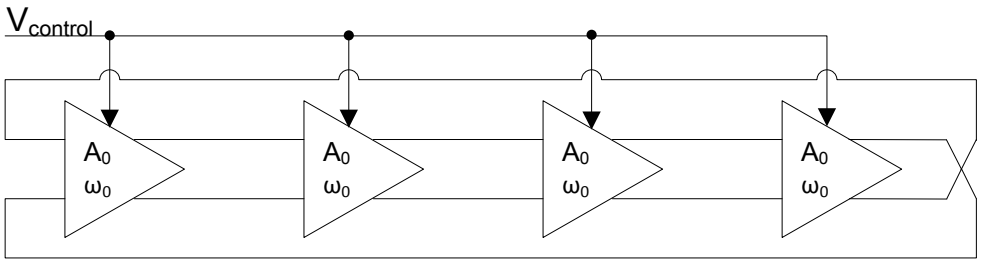

Figure 2.11: Typical 4-stage ring oscillator

Figure 2.11 shows a typical 4-stage ring oscillator. Each stage in the ring has a gain of A_0 and has a frequency response of ω_0. The total transfer function through the chain is given in Eq. (2.4) [10]. Since there are four stages, the phase shift provided by each stage is 180°/4 = 45°. This forces the oscillator to function at a frequency $\omega=\omega_0$ and requires each stage to have a gain expressed in Eq. (2.5). This design provides 8 clock phases separated by 45°.

$$H(s) = -\frac{A_0^4}{\left(1+\frac{s}{\omega_0}\right)^4} \tag{2.4}$$

$$\arctan\left(\frac{\omega}{\omega_0}\right) = 45° \Rightarrow \omega = \omega_0$$

$$\left|H(s)\right| = \frac{A_0^4}{\sqrt{1+\frac{\omega}{\omega_0}^4}} = 1 \Rightarrow A_0 = \sqrt{2} \tag{2.5}$$

2.1.4. PLL Closed-Loop Behavior

Assembling all blocks into a closed-loop linear form, the individual transfer functions are given in Figure 2.12. The two nodes in the PLL model where phase noise can be introduced are Φ_{REF} going into the PFD and Φ_{VCO} of the VCO. In the closed-loop system, the PLL tracks Φ_{REF} along with any phase noise present. Likewise, the PLL attempts to track the VCO-generated phase noise, Φ_{VCO}, fed back to the PFD. These two points provide two separate transfer functions for a closed-loop PLL behavior: $H_{REF}(s)$ and $H_{VCO}(s)$.

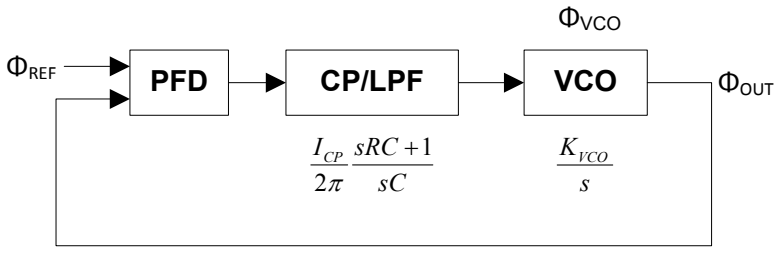

Figure 2.12: Charge pump PLL linearized model

The input transfer function, $H_{REF}(s)$, is given by [10]:

$$H_{REF}(s) = \frac{\Phi_{OUT}}{\Phi_{REF}} = \frac{\frac{I_{CP}K_{VCO}}{2\pi C}(sRC+1)}{s^2 + \frac{K_{VCO}I_{CP}R}{2\pi}s + \frac{K_{VCO}I_{CP}}{2\pi C}} \tag{2.6}$$

Comparing the $H_{REF}(s)$ with a second-order transfer function from basic control theory, Eq. (2.7), yields the parameters in Eq. (2.8).

$$H_{REF}(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.7}$$

$$\omega_n = \sqrt{\frac{I_{CP}K_{VCO}}{2\pi C}}, \zeta = \frac{RC}{2}\sqrt{\frac{I_{CP}K_{VCO}}{2\pi C}} \tag{2.8}$$

Using the same parameters, the VCO transfer function, $H_{VCO}(s)$, is given by:

$$H_{VCO}(s) = \frac{s^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{2.9}$$

These transfer functions indicate that phase noise on Φ_{REF} has a low-pass response through the PLL while the noise on Φ_{VCO} has a high-pass response.

The parameter ω_n represents the PLL's natural frequency and ζ is the damping factor. These two loop parameters determine the PLL stability and give a starting point to calculate device sizes. For closed-loop stability, the PLL natural frequency should be approximately 10-20 times lower than the input reference [12]. For maximal loop response time with minimized transient ringing, ζ should be set as 0.707 for a critically damped system. For fine-tuning the system for stability, the loop filter capacitors, C_1 and C_2, can influence the phase margin:

$$PM = \tan^{-1}\left(\frac{b-1}{2\sqrt{b}}\right), b = \frac{C_1}{C_2} + 1 \tag{2.10}$$

Finally, the settling time of the PLL can be adjusted with the approximation:

$$T_{settle} \approx \frac{4.6}{\zeta \omega_n} \tag{2.11}$$

With these basic guidelines as a starting point, all remaining PLL parameters can be calculated based on the design of the PFD, charge pump, loop filter, and VCO.

2.1.5. PLL Simulink Model Design and Results

The Simulink model for the PLL is shown in Figure 2.13. The reference clock was generated by the VCO model given in Figure 2.14 with an optional noise signal on the V_{CTRL} for performance testing. The VCO model is a modulus counter adding in steps of Δ_Φ expressed as:

$$N_\Delta = \frac{\tau_{VCO}}{T_S}$$
$$\Delta_\Phi = \frac{2\pi}{N_\Delta} = 2\pi \cdot f_{VCO} T_S \tag{2.12}$$

where N_Δ is the number of steps to reach one VCO clock period, τ_{vco}, and T_S is the minimum simulation time step. This generates a sawtooth waveform representing phase ramping from 0 to 2π which is then passed as the argument to a sinusoid. The K_{VCO} and V_{CTRL} inputs allow modification of τ_{vco}.

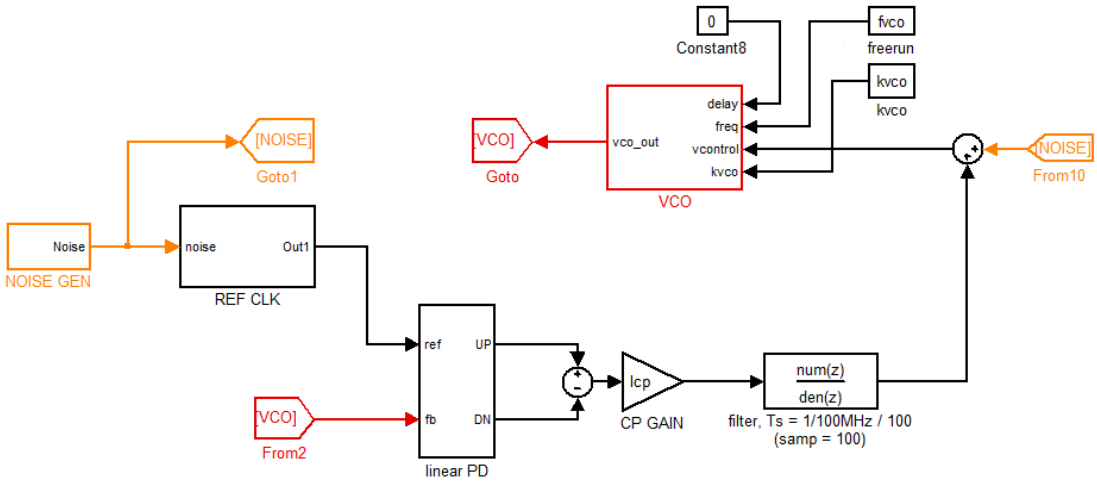

Figure 2.13: Simulink model of a PLL

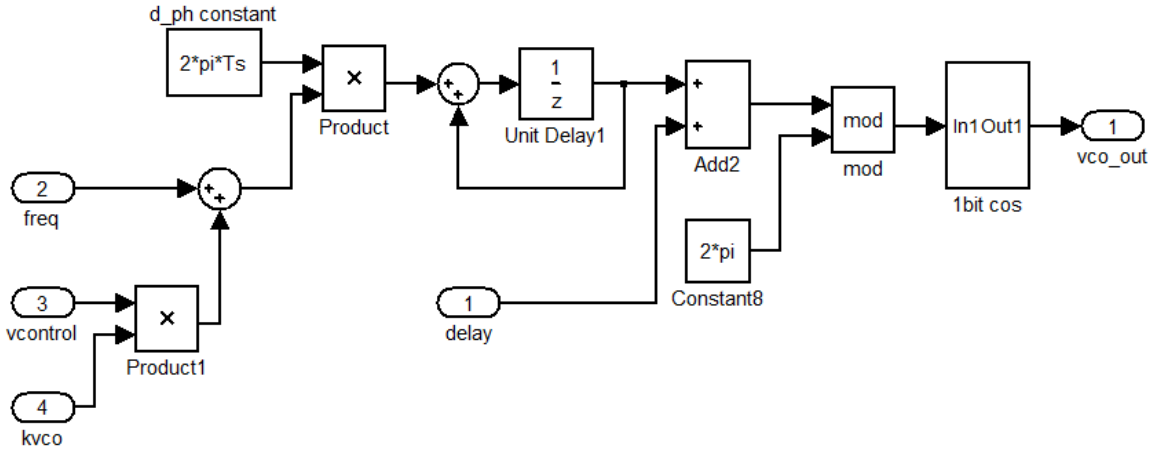

Figure 2.14: Simulink model for the VCO and REF CLK

The linear PFD was modeled as previously described in this chapter with a simulated behavior given in Figure 2.15. The PFD output signal *UP* is assigned a positive value and *DN* is assigned a negative. Sweeping $\Delta\Phi_{in}$ from $-\pi$ to π generated a linear behavior with a zero at $\Delta\Phi_{in} = 0$.

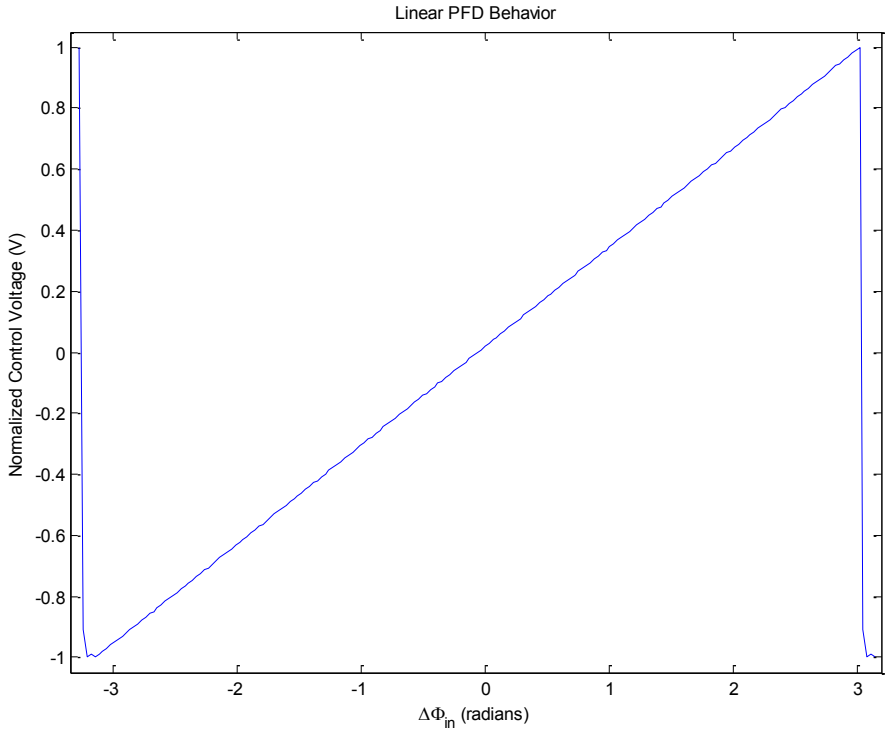

Figure 2.15: Simulink linear PFD behavior

Design began by arbitrarily choosing a reference frequency of 100 MHz, implying an initial ω_n of 5 MHz. Having a ring oscillator design with K_{VCO} = 400 MHz/V gives a tuning range of ±50 MHz using only 0.3 V. An initial loop filter capacitor was estimated to be 50-100 times larger than potential parasitic capacitances on the node. Since the loop filter node is connected to the VCO control gates and the drains of the charge pump, an initial estimation of 50 pF was chosen. For a critically damped loop response, damping factor ζ needs to be 0.707. Now Eq. (2.8) becomes a system of two variables, I_{CP} and R. Solving this system of equations yielded 30 µA for I_{CP} and 2.2 kΩ for R.

The V_{CTRL} smoothing capacitor was empirically chosen by running a simulated sweep starting at $C_1/10$ and reducing C_2 until ripples appeared on V_{CTRL}. A final sizing of

C2 was chosen to be $C_1/25$ to offer ripple reduction while still offering a phase margin of 67.8° given by Eq. (2.10).

With the calculated parameters, the PLL was simulated in a closed loop and tracks a static reference clock. After a time, the reference clock was given a step in phase and Figure 2.16 shows the V_{CTRL} keeping the lock. Similarly, Figure 2.17 exhibits the V_{CTRL} tracking the reference clock after a frequency step. Based on Eq. (2.11), the calculated settling time was approximately 0.523 µs. This estimation is fairly accurate as the step responses all settle within 0.45 µs.

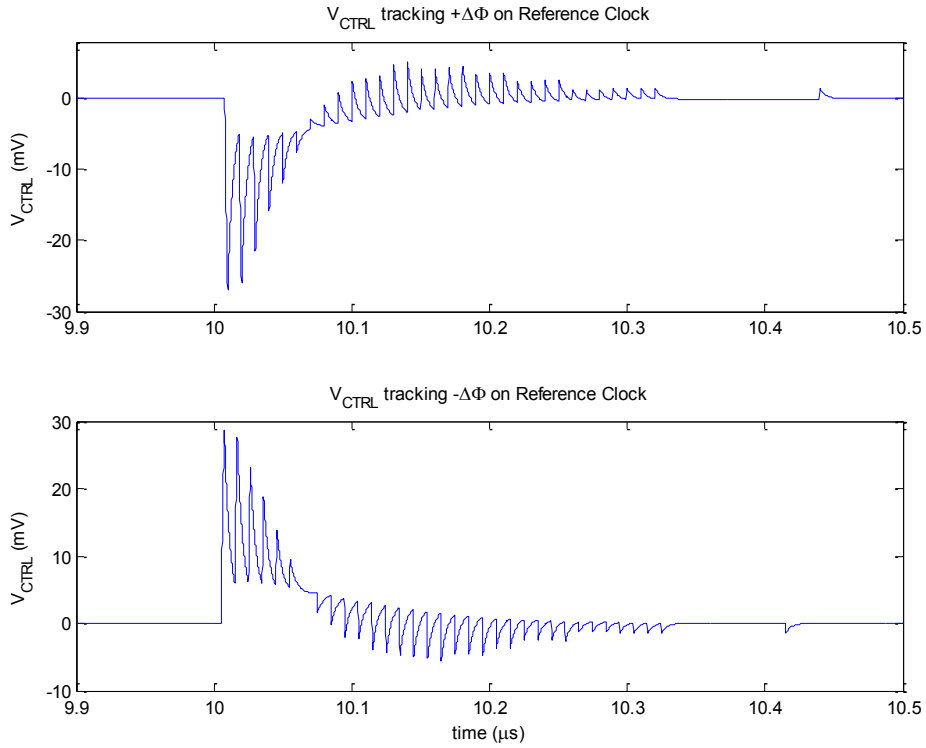

Figure 2.16: V_{CTRL} plots for tracking a phase step

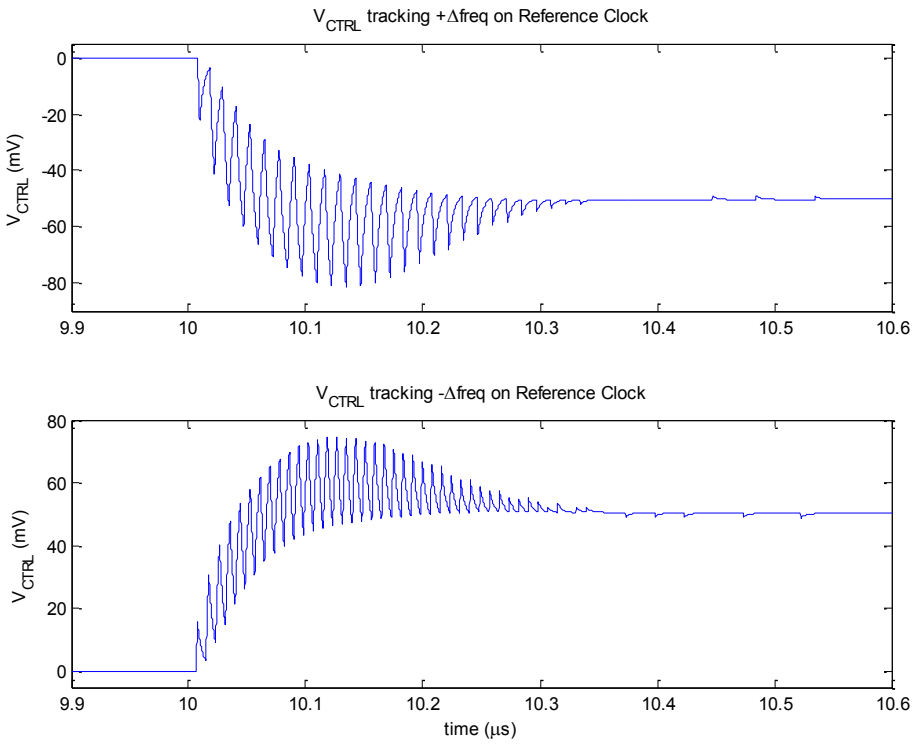

Figure 2.17: V_{CTRL} plots for tracking a frequency step

The next two plots demonstrate the PLL system transfer functions $H_{REF}(s)$ and $H_{VCO}(s)$. A noise source was added to the reference clock at a frequency ω_n /2 and ω_n x100. Another simulation is run with the same noise frequencies on the *VCO*. Figure 2.18 shows that low frequency noise on the reference causes large changes in the V_{CTRL}. Figure 2.19 shows that high frequency noise causes unchecked V_{CTRL} changes while low frequency noise causes the *VCO* to force itself to regain lock and settle.

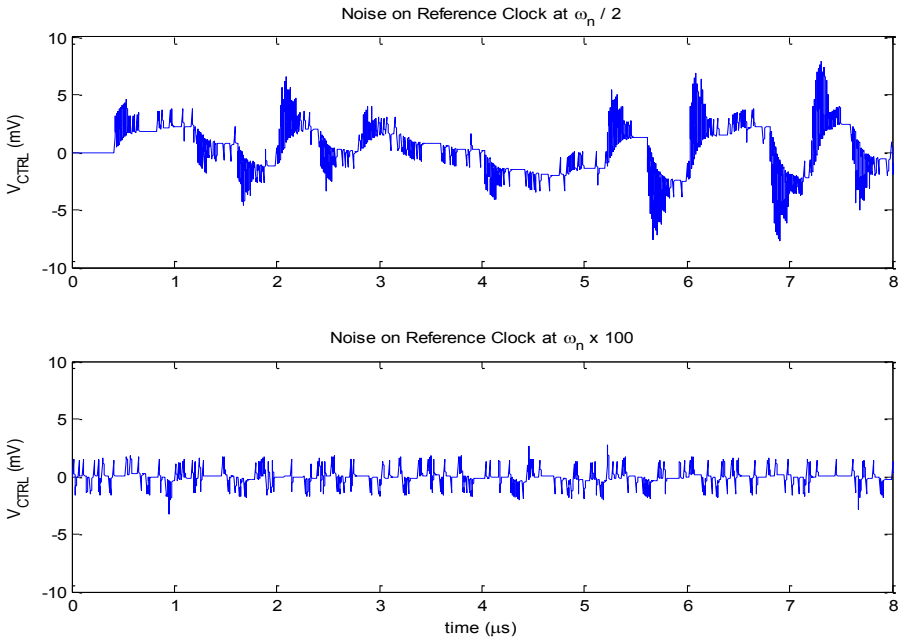

Figure 2.18: PLL low-pass response to reference noise

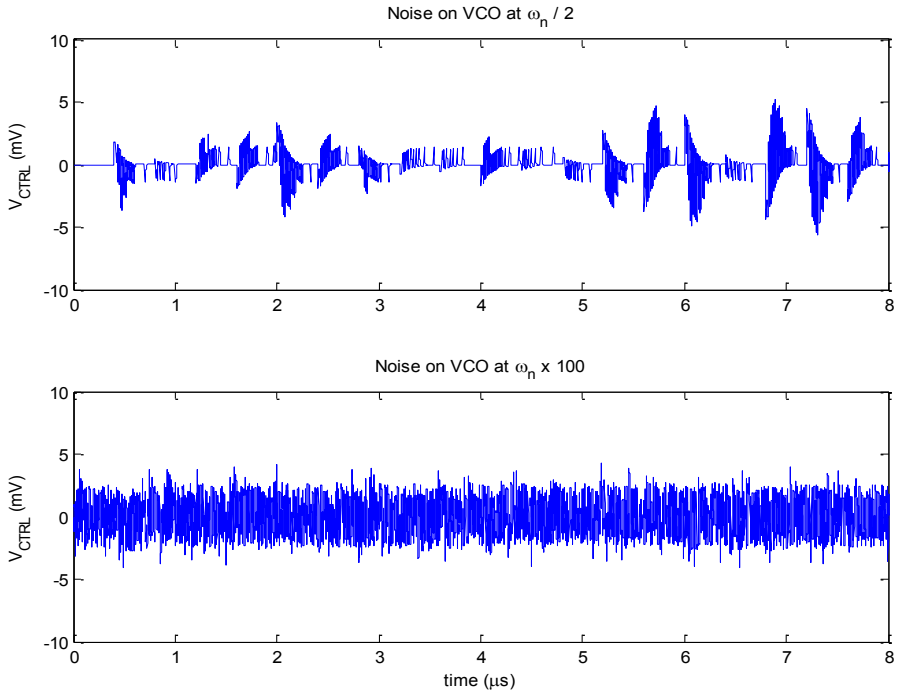

Figure 2.19: PLL high-pass response to VCO noise

2.2. Phase Interpolator CDR

A phase interpolator CDR, shown in Figure 2.20, is a dual-loop system with a multi-phase PLL for frequency tracking and a PI for phase tracking [8]. In a multi-lane CDR, the PLL provides each data receiver with equally spaced clocks locked onto the incoming data frequency. Each receiver then has its own PI that locks onto the phase shifts of the data stream.

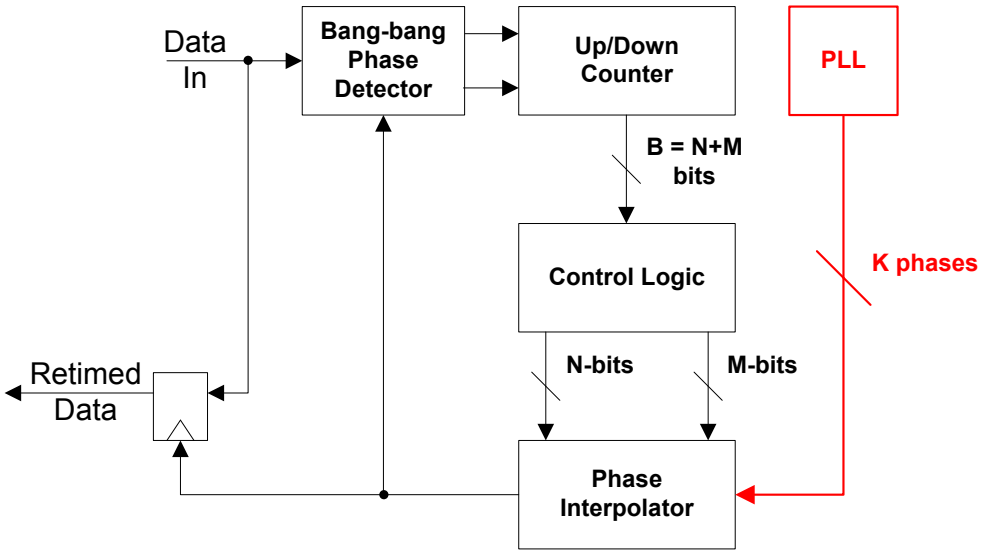

Figure 2.20: Standard phase interpolator block diagram

Some advantages of using a phase interpolator CDR are locking speed and digital control for scalability. A basic PI CDR has a bang-bang phase detector, an up/down counter with digital control logic, and an analog phase mixer to mix *K* phases of a clock generated by a PLL. The control logic and analog phase mixer are explored before examining simulated results of a standard PI CDR.

2.2.1. PI Control Loop

The data stream input is compared with the feedback phase-mixed clock and drives an up-down counter with the bang-bang phase detector *UP*/*DN*/*NC* outputs. In a simple control logic, the *M* MSB bits of the counter is a mux selection to choose two adjacent phases from the PLL, Φ_S and Φ_{S+1}, while the *N* LSB bits translate into the α weighting for the analog phase mixer to blend the two clock phases.

Since the phase angle circle in Figure 2.21 performs a modulus-2π operation, there is no discontinuity when the counter wraps around from 2^B to 0 and vice versa. Either logic or a finite-state machine (FSM) can provide glitch-free switching of Φ_S / Φ_{S+1} into the next region as α increases or decreases.

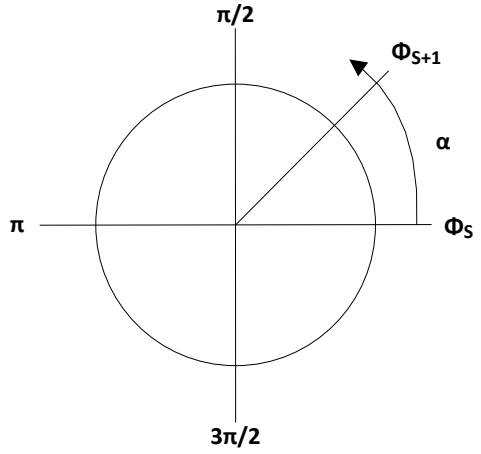

Figure 2.21: PI mux selection and α control

One key component attributing to the speed of a PI is the Alexander PD. The bang-bang behavior, while quick on decision-making, causes systematic jitter once the CDR has achieved lock. Assuming the control logic does not employ filtering and the phase mixer

has sufficient response time, the !!PD makes a decision every clock cycle. Since the detector does not produce a "no-change" output when the system is in lock, it generates a continuous stream of up/down decisions around the locked counter value. In the absence of circuit noise and input jitter, the !!PD toggles between three counter states alternating between X and $X\pm1$. Given a B-bit control word, this causes a systematic jitter of $UI / 2^B$. In the case of low counter resolutions, this causes significant systematic jitter. Reducing this jitter by increasing the resolution takes appreciable effort due to the considerations required for the analog phase mixer.

2.2.2. Analog Phase Mixer

The phase mixer is a current-steering digital-to-phase converter similar in structure to a Gilbert cell [13], shown in Figure 2.22 [8]. The I_C units each consists of N current legs that are combined in with the α weighting. The two adjacent clock phases, Φ_S and Φ_{S+1}, and α are mixed according to:

$$\Phi_{OUT} = \alpha \cdot \Phi_{S+1} + \Phi_S (1-\alpha) \tag{2.13}$$

If the PLL generates $K = 2^M$ clock phases, the control logic devotes M bits to mux selection with the remaining bits serving as α. With a control word resolution of B bits, the phase mixer will have $N = 2^B$-2^M unit current legs. The obvious solution to reducing the overall jitter is to increase the resolution of the control word, but power, linearity, and layout limitations from the phase mixer reduces its practicality [14], [8].

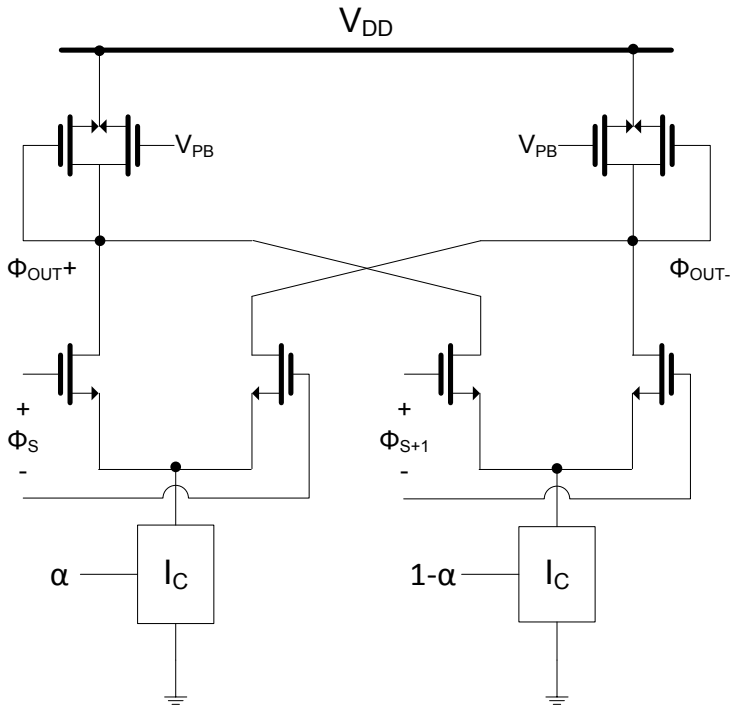

Figure 2.22: PI analog phase mixer

The α weighting translates into a digital code to enable varying amounts of these N unit current legs. Looking from the perspective of a total current I_α, this behavior exactly follows a current-steering DAC [15]. There are two main encodings that α can use to enable the N unit currents: binary and thermometer.

In a binary encoding, if α were represented by n bits, where $N = 2n$, each bit of α corresponds to a binary weighted current. In other words, I_α is selected with n switches enabling current legs that are weighted 2^0, 2^1, 2^2,… 2^{n-1}. Figure 2.23 shows an example layout of unit current elements for a 5-bit α arranged sequentially. The numbers in the grid represent the legs of unit current. This kind of arrangement incurs severe linearity and mismatching between codes that are physically spaced apart. For example, increasing the code for α by a single LSB from 01111 results in 10000. This causes half of the total

N legs to turn off while simultaneously enabling the other half. Besides the sudden spikes of current, there are routing capacitance differences between unit devices within the same code.

1	2	2	4	4	4	4	8
8	8	8	8	8	8	8	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	32
32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	32
32	32	32	32	32	32	32	

Figure 2.23: Example of poor layout matching of unit current devices

A more linear alternative to this is thermometer or unary encoding. In this method, each of the *N* unit current legs has a separate switch. The advantage of thermometer encoding is the elimination of simultaneously enabling multiple current legs: each change in LSB only enables or disables a single current leg. This method increases routing complexity for linearity.

In [15], a DAC was designed with a segmented encoding, where a few LSB were binary encoded and the remainder was thermometer encoded. The segmented encoded DAC was able to significantly reduce the number of switches while still maintaining a similar linearity to thermometer. There are many other techniques for reducing mismatch and improving linearity in DACs such as dynamic element matching [16] and mismatch shaping [17].

2.2.3. PI CDR Simulink Model

The model for a PI CDR is shown in Figure 2.24. It was simulated with only a PRBS-10 data stream to show functionality of the model. The 8-phase clock generator uses eight copies of the *VCO* model from the PLL, each separated by a phase of π/4. The 8:2 mux simply reroutes the eight phases into pairs of neighboring phases based on the *MUXSEL* input. The !!PD was modeled after the Alexander phase detector and has a simulated response shown in Figure 2.25.

The model for the up-down counter, shown in Figure 2.26, contains a decimal-to-binary encoder in order to split the top 3 MSB into a mux select signal and the bottom 5 LSB are used for α.

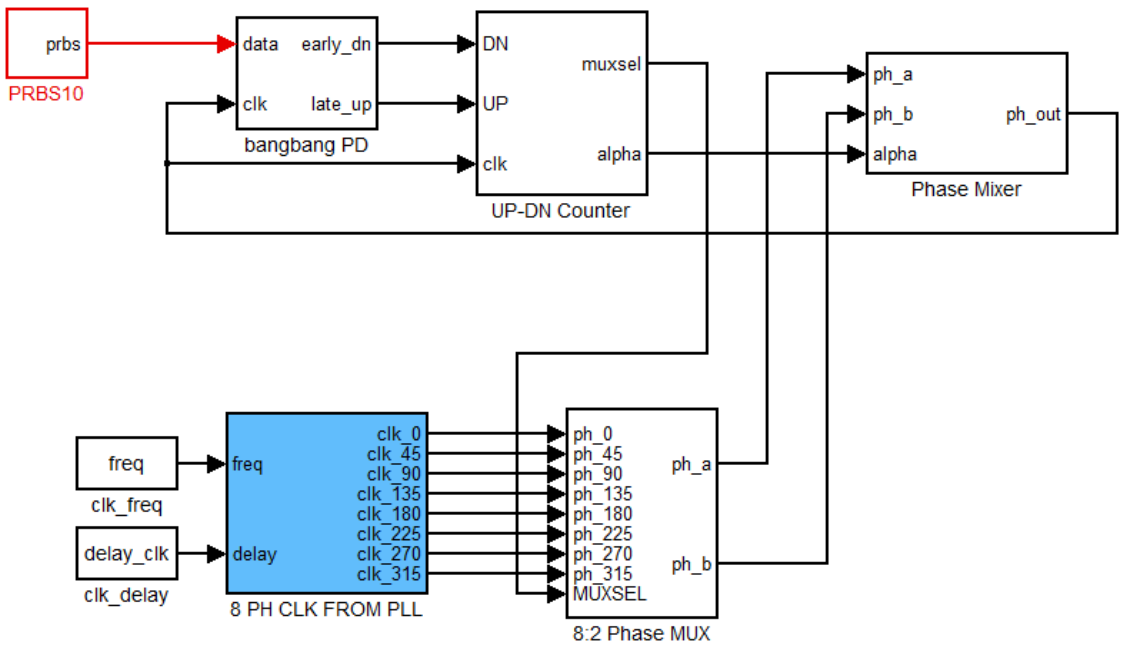

Figure 2.24: Simulink model for a PI CDR

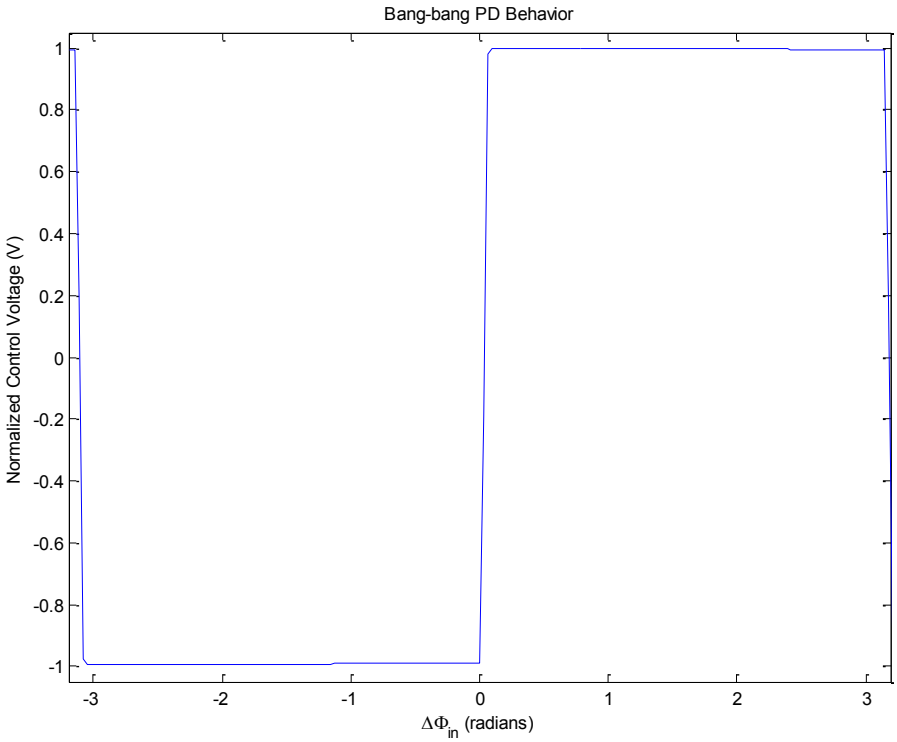

Figure 2.25: Simulink bang-bang PD behavior

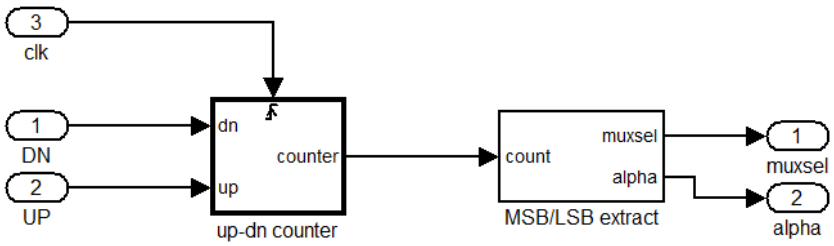

Figure 2.26: Simulink model for the up-down counter

While the analog phase mixer is quite challenging to design on the transistor level, the modeling of it is very straight-forward. Two neighboring clock phases mixed with a weighting factor α following Eq. (2.13) can easily be summed up with very minor glitching using sinusoidal clocks. Figure 2.27 shows the model for the entire phase mixer. The weighting DAC is simply a division block to generate a percentage for α and 1-α.

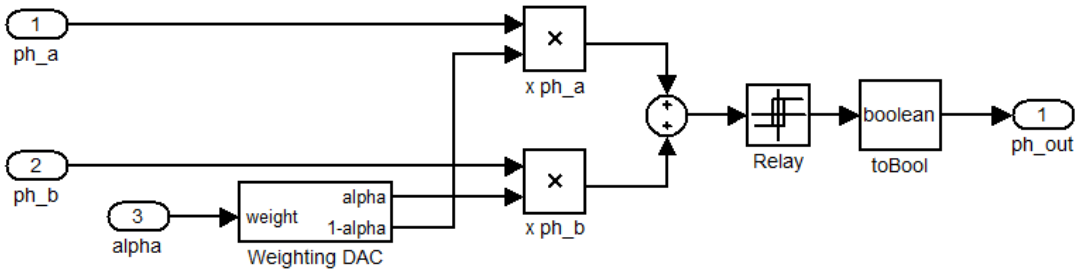

Figure 2.27: Simulink model for the analog phase mixer

To show functionality of the PI CDR, the worst case data tracking scenarios were used. In Figure 2.28, the CDR is locking onto a data stream that is late with a phase shift of π radians and Figure 2.29 shows the CDR locking onto an early phase shift of π radians. The phase shift of ±π was chosen since it represents the maximum a !!PD can handle before wrapping around to the opposite output. This gives the CDR the longest phase acquisition.

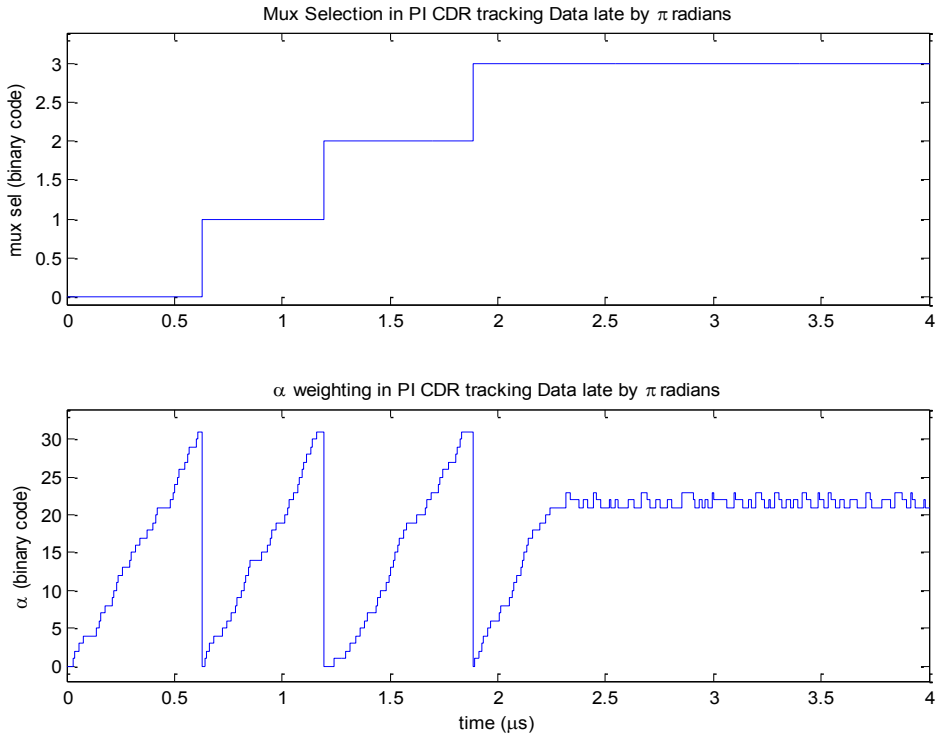

Figure 2.28: PI CDR tracking a data stream late by a phase of π

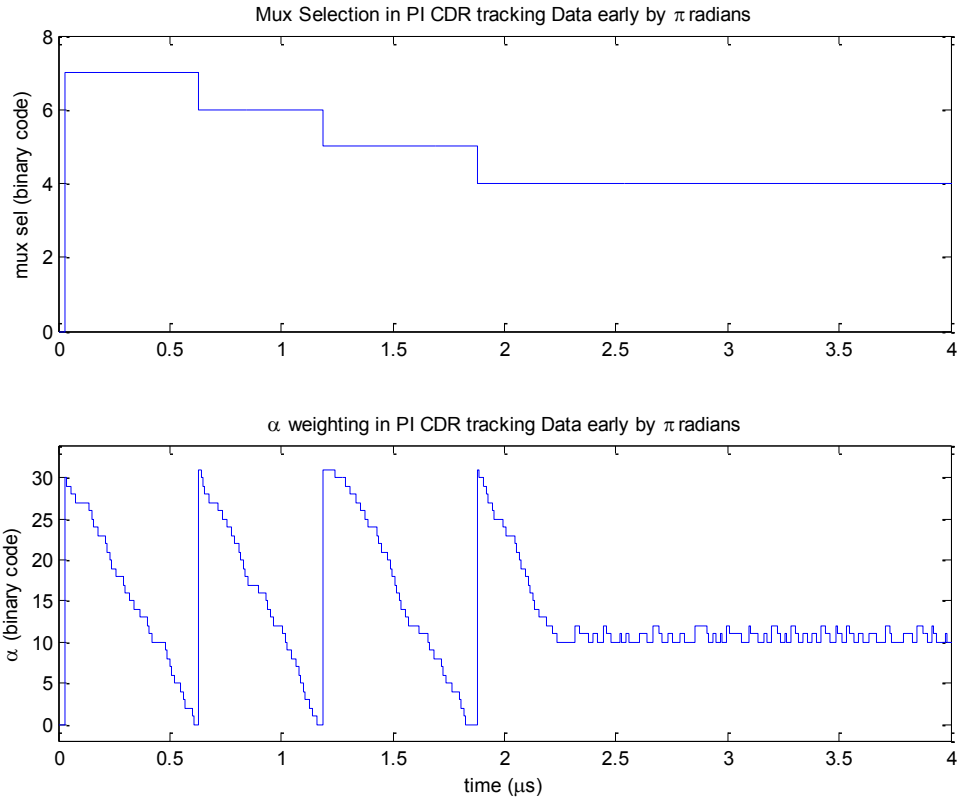

Figure 2.29: PI CDR tracking a data stream early by a phase of π

2.3. ΔΣ Modulator

ΔΣ modulators were originally a means to reduce quantization noise for analog to digital converters (ADC) and digital to analog converters (DAC). This technique required sampling a signal at frequencies much higher than the Nyquist rate, shaping the noise with a high-pass response, then employing a post-conversion filter to cut off the out of band noise power.

2.3.1. Data Converter Overview

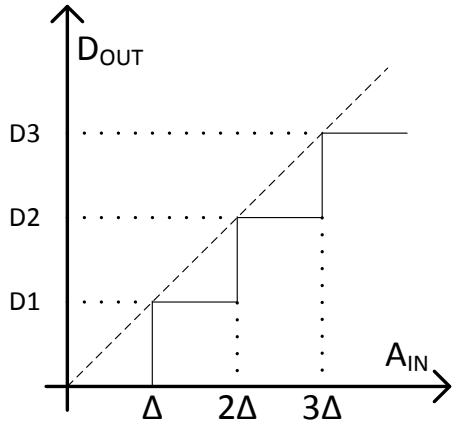

Figure 2.30: Example ADC behavior

An example of an ADC behavior is shown in Figure 2.30, where A_{IN} represents each analog voltage and D_{OUT} are the outputted digital codes. A B-bit ADC converts an analog signal into a digital equivalent with 2^B quantized levels [18]. A uniform error occurs at each level of quantization for each of the 2^B codes. Given an ADC input with a max voltage range of V_{MAX}, each digitized code has an analog voltage step of Δ or V_{LSB}, expressed as:

$$\Delta = \frac{V_{MAX}}{2^B} \tag{2.14}$$

Provided a sawtooth input covering the entire ADC range, integrating the quantization error across one period yields a quantization noise power, ε_{qrms}, given by:

$$\varepsilon_{qrms} = \overline{\varepsilon_q^2} = \frac{\Delta^2}{12} \tag{2.15}$$

Calculating the theoretical maximum signal to noise ratio (SNR) for a sinusoid input with peak-to-peak amplitude of V_{MAX} is simplified down to Eq. (2.16), given as signal to quantization noise ratio (SQNR).

$$SQNR = 6.02B + 1.76 \tag{2.16}$$

With a standard Nyquist-rate converter the sampling bandwidth extends out to twice the maximum signal bandwidth. An oversampled converter can utilize higher sampling frequencies in exchange for lowering the in-band quantization noise.

2.3.2. Noise Reduction of Oversampling and ΔΣ Modulation

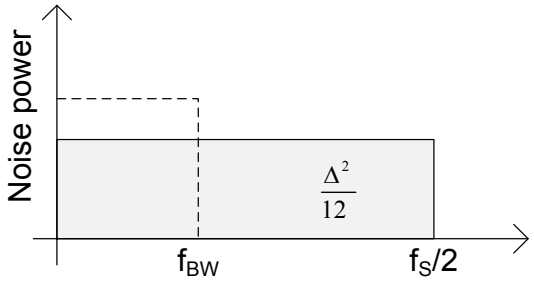

Figure 2.31: Quantization noise spectrum versus signal bandwidth

Figure 2.31 shows an example plot of a Nyquist converter. The quantization noise is spread uniformly across the entire available signal bandwidth $f_S/2$, where f_S is the sampling frequency. With an input signal bandwidth of f_{BW} and an ideal low-pass filter, the in-band quantization noise becomes:

$$\varepsilon_{inband} = \overline{\varepsilon_q^2}\,\frac{f_{BW}}{f_S/2} = \frac{\Delta^2}{12}\,\frac{f_{BW}}{f_S/2} \tag{2.17}$$

Oversampling is done by sampling the signal beyond the Nyquist rate with an oversample ratio (OSR) defined as:

$$OSR = \frac{f_S / 2}{f_{BW}} \tag{2.18}$$

Oversampling converters allows the quantization noise power to be spread across the entirety of the increased sampling bandwidth while the signal still remains within f_{BW}. The new in-band noise has been reduced by a factor of *OSR*. Combining Equations (2.17) and (2.18), a 6 dB reduction of quantization noise power can be observed for every doubling of *OSR*.

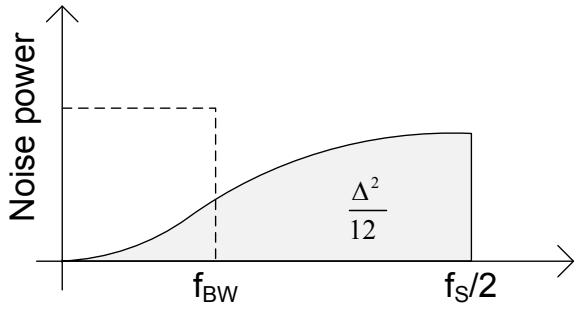

Figure 2.32: ΔΣ quantization noise spectrum versus signal bandwidth

In a ΔΣ converter, the quantization noise is both oversampled and shaped by a high-pass filter, Figure 2.32, vastly reducing the in-band power. Detailed in [19], the in-band quantization noise for a second order ΔΣ ADC is:

$$\varepsilon_{\Delta\Sigma,inband} = \overline{\varepsilon^2{}_{\Delta\Sigma}} = \frac{\pi^4 \cdot \overline{\varepsilon^2{}_q}}{5(OSR)^5} \tag{2.19}$$

At an *OSR* of 8, a second order ΔΣ ADC has a 34dB reduction of the in-band quantization noise, whereas it would take a blindly oversampled converter an *OSR* of 2^6 to achieve a similar reduction. There are two means to reduce the in-band noise: increasing *OSR* or increasing the ΔΣ modulator order. However, these two parameters have their limitations. Increasing the *OSR* places a frequency constraint on the input signal, and the modulator runs into stability problems at orders $L > 2$. The general form of Eq. (2.19) is given as:

$$\overline{\varepsilon^2_{\Delta\Sigma}} = \frac{\pi^{2L} \cdot \overline{\varepsilon^2_q}}{(2L+1)(OSR)^{2L+1}} \tag{2.20}$$

Plotting out Eq. (2.20) with varying modulator orders generates the graph in Figure 2.33. Given an *OSR*, this graph provides an estimation of the required ΔΣ modulation order to reach a desired level of in-band noise reduction.

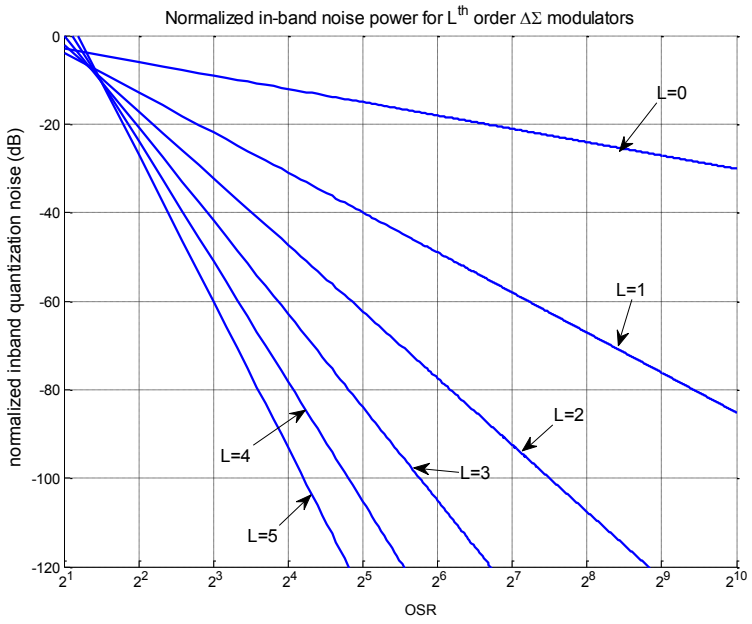

Figure 2.33: ΔΣ in-band quantization noise versus *OSR* for different orders

2.3.3. ΔΣ Modulator Loop Filter

The high-pass response of the ΔΣ modulator affects only the quantization noise. The signal ideally passes through unperturbed. Figure 2.34 is the Z-domain linear model for a second order ΔΣ modulator. The input signal *U* is passed through two filters, $H_1(z)$ and $H_2(z)$, before being quantized. The quantizer is represented as an additive quantization noise signal, *Q*. The quantized output, *V*, is then subtracted from the input path. The modulator continuously sums up the difference between the filtered input and the quantized output (the Σ of the Δ's).

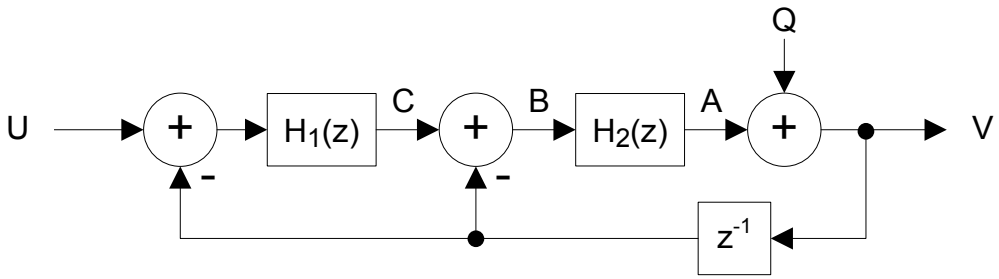

Figure 2.34: Standard linearized model for second order ΔΣ modulation

The transfer function of the modulator is calculated by introducing a few intermediate nodes, *A*, *B*, and *C*, and defining them as:

$$V = A + Q \ , \ A = H_2 B \ , \ B = C - z^{-1}V \ , \ C = H_1(U - z^{-1}V) \qquad (2.21)$$

Eliminating all the intermediate terms yields the output in terms of only the input and quantization noise:

$$V = Q - z^{-1}H_2V + H_1H_2U - z^{-1}H_1H_2V \qquad (2.22)$$

$$V = \frac{1}{1 + z^{-1}H_2 + z^{-1}H_1H_2} Q + \frac{H_1H_2}{1 + z^{-1}H_2 + z^{-1}H_1H_2} U \tag{2.23}$$

The result in Eq. (2.23) indicates that the output is composed of the input and quantization noise, each with its own transfer function. The desired transfer function for the quantization noise is a second order high-pass response while leaving the input alone. Defining the noise transfer function (NTF) and signal transfer function (STF) with their respective Z-domain responses yields:

$$NTF = \frac{1}{1 + z^{-1}H_2 + z^{-1}H_1H_2} = (1 - z^{-1})^2$$
$$STF = \frac{H_1H_2}{1 + z^{-1}H_2 + z^{-1}H_1H_2} = 1 \tag{2.24}$$

Solving Eq. (2.24) provides a standardized and easily realizable form for each of the cascaded filters:

$$H_1 = H_2 = \frac{1}{1 - z^{-1}} \tag{2.25}$$

2.3.4. ΔΣ Simulink Model

The architecture used for the model is from [20], shown in Figure 2.35. This has a feedback and feed-forward in the modulator structure. *U* and M have the same number of bits. Following empiric simulated results done in [20], *U* was chosen to from between ±*M* /2 for well-behaved results. When *U* is chosen outside of these limits, it increases the

probability that the accumulators in the loop filter saturates, relieving the ΔΣ output of the desired high-pass noise shaping.

In this simulation, the frequencies were in the order of 10^0-10^4 Hz. Since the ΔΣ is modeled in terms of transfer functions, there does not exist any silicon limitations on speed. Any arbitrary frequency may be used for an input signal as long as the *OSR* and sampling frequencies are in the same proportion.

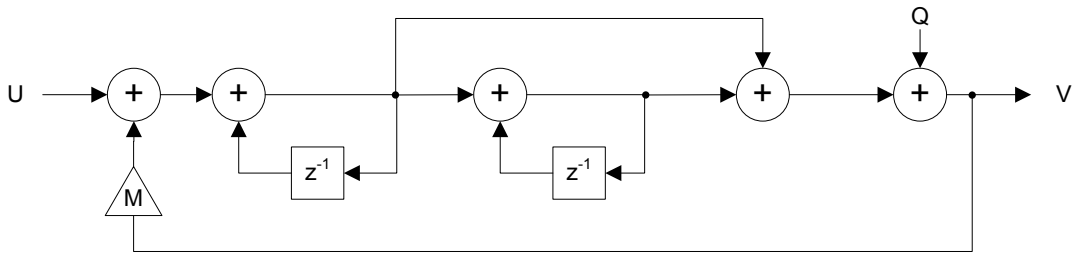

Figure 2.35: Simulink model for a 1-bit second order ΔΣ modulator

The sinusoidal digital input to the ΔΣ modulator and the reconstructed output of the ΔΣ DAC are shown in Figure 2.36. The order of the low-pass reconstruction filter was initially chosen to be one greater than the ΔΣ modulator order. That filter did not adequately cut away noise nor did the filter properly reconstruct the ΔΣ 1-bit output. A simulation tested fourth order (two orders greater than the order the ΔΣ) proved to be sufficient to recover the sinusoidal signal.

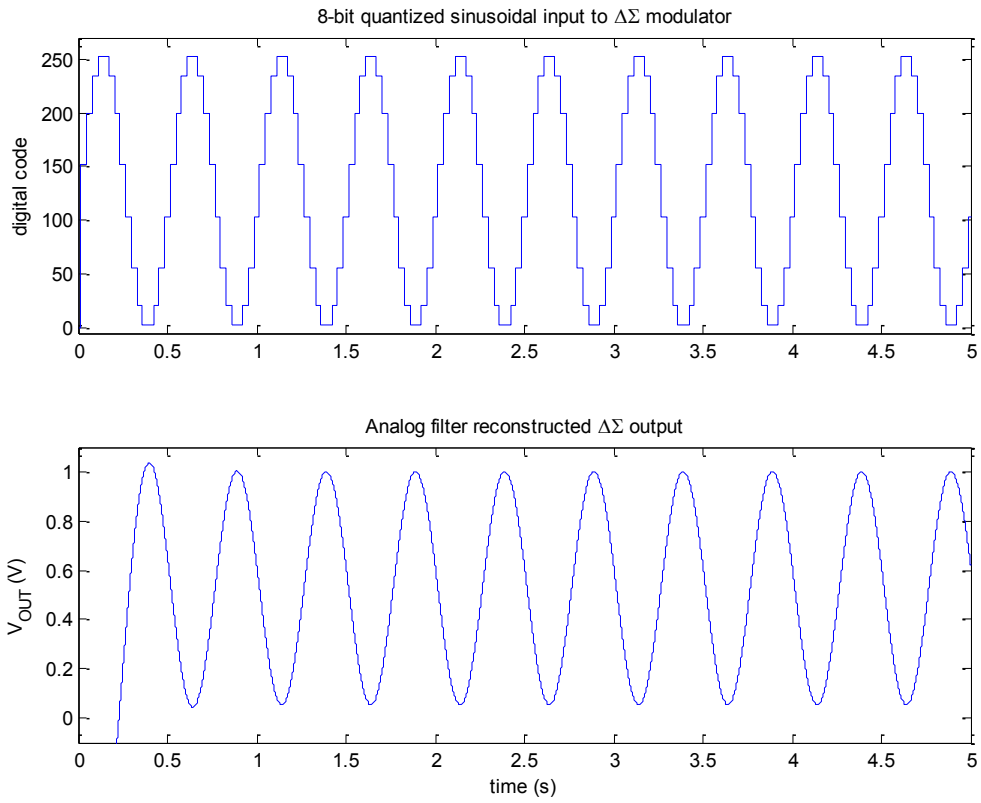

Figure 2.36: 1-bit quantized second order ΔΣ modulator transient results

The input signal frequency was chosen to fall within an exact FFT bin as to avoid spectral smearing [19]. Due to the lack of a digital interpolation filter on the input to the ΔΣ modulator, the power spectrum, Figure 2.37, shows significant frequency components within the pass-band. These frequency spikes occur on multiples of the input sinusoid frequency. Likewise, the analog to digital quantization of an idealized sinusoidal introduced many frequency components due to the sampling with respect to the converter clock. The first three spikes presented on the spectrum plot are the signal (first spike) and the harmonics of this low frequency input.

The function of an interpolation filter is to increase the sampling rate by zero-padding the oversampled results [19]. For example, passing an input with frequency f_{in} through an 8x interpolation filter will produce an output at 16x f_{in}. The interpolation filter output has the first sample of the input followed by 7 zeros, a second sample of the input followed by 7 zeros, and so on. The input signal is thus given an artificial high-frequency sampling component, pushing these sampling-related harmonics beyond the pass-band. Using an approximate calculation, assuming an interpolation filter was sufficient to remove the input harmonics within the pass-band, an estimated "*SNR* (est)" was calculated. The Matlab code for this estimation simply ignores the harmonics within the pass-band when calculating *SNR*.

The pairs of harmonic spikes that occur further past the signal bandwidth are caused by the sampling clock mixing with the signal frequency. The input signal is 2 Hz and the digital quantization was clocked at 32 Hz. At every multiple of 32 Hz, a pair of spikes 2 Hz apart is the result of these frequencies mixing. Finally, the power spectrum of the higher frequencies of the ΔΣ modulator exhibits the theorized +40 dB/dec second order high-pass response.

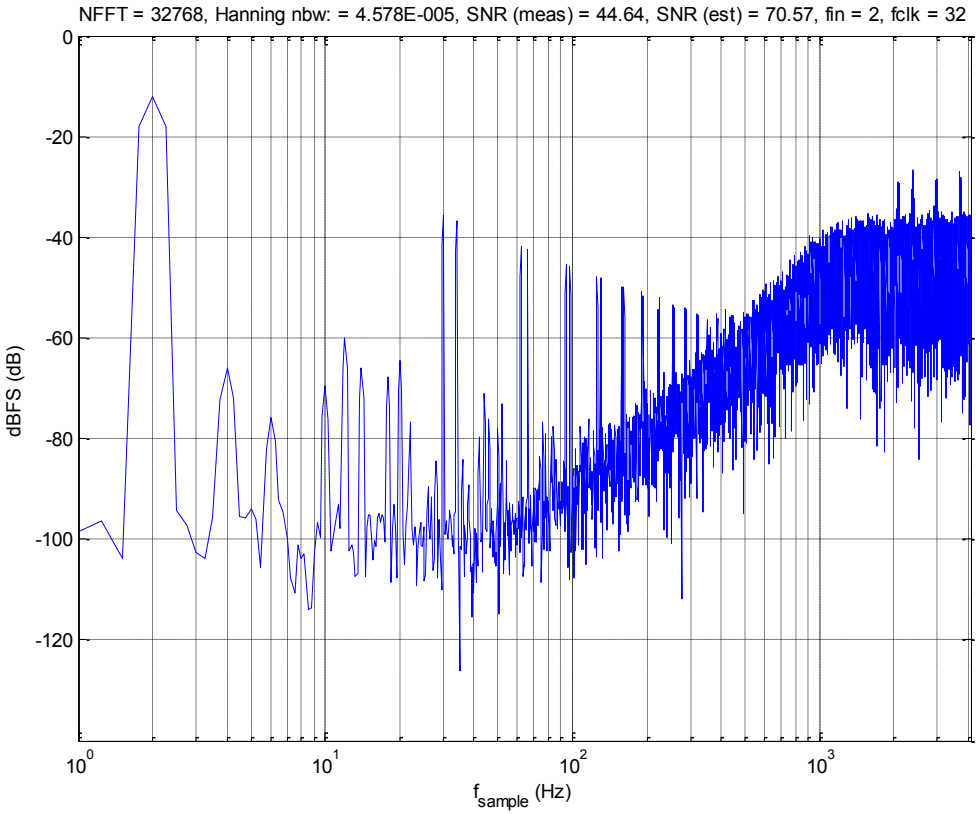

Figure 2.37: 1-bit quantized second order ΔΣ modulator output power spectrum

Chapter 3. Techniques for CDR Jitter Reduction

The last chapter introduced the building blocks for CDRs and discussed some limitations in their design. Many techniques are currently available for alleviating these jitter limitations in CDRs. This chapter covers an analog modification to the phase mixer in a standard PI and two novel CDR architectures.

3.1. Nested Phase Interpolation

The nested phase interpolator [21] CDR follows the same architecture as a standard PI except it utilizes a nested interpolator. While the standard PI breaks up its control word into a mux selection and α, the nested PI further separates α into a α_{course} and α_{fine}. The goal is to use cascaded stages of lower resolution interpolation orders, shown in Figure 3.1, to save on area and power.

The mux selection provides the initial two adjacent clock phases, Φ_M and Φ_{M+1}, to the coarse interpolators. These interpolators are driven by α_{coarse} and $\alpha_{coarse+1}$ in order to generate two adjacent coarsely interpolated phases. Finally, the α_{fine} drives the fine phase interpolator to arrive at the final Φ_{OUT}. A nested design of equal resolution to a standard PI can offer much lower power and area consumption.

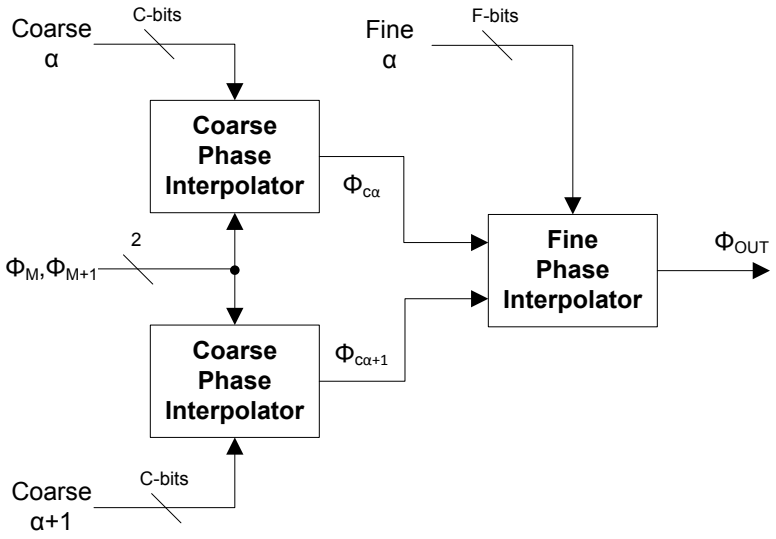

Figure 3.1: Nested PI block diagram

Discussed previously, a typical phase mixer with an *M*-bit *α* control word will have *2M* current legs. A nested interpolator design has two *C*-bit coarse interpolators driving an *F*-bit fine interpolator for the final output clock, where $F = M - C$.

The number of current legs in a traditional phase mixer (left side of inequality) can be related to the nested design by:

$$\frac{2^M}{2} \geq 2 \cdot 2^C + 2^{M-C} \tag{3.1}$$

Rearranging the inequality and splitting up the left hand side into two parts then taking the $\log_2$ of both sides yield:

$$2^{M-2} + 2^{M-2} \geq 2^{C+1} + 2^{M-C} \tag{3.2}$$

$$M - 2 \geq C + 1 \ , \ M - 2 \geq M - C \tag{3.3}$$

Solving for M and C results in the lower limits of $M \geq 5$ and $C \geq 3$. When a nested PI is designed with these bounding conditions, the total number of current legs used by will be less than half of a standard PI.

3.2. Averaging Phase Interpolation

Covered in the previous chapter, a standard PI CDR has systematic jitter limited by the resolution of the control word. As there is a practical and power limitation on increasing the control word, [22] utilized a PLL inside the CDR phase tracking loop to act as an analog phase mixer. This design allows for a coarse CDR interpolator as the PLL itself provides the fine interpolation. Figure 3.2 shows the block diagram of this architecture.

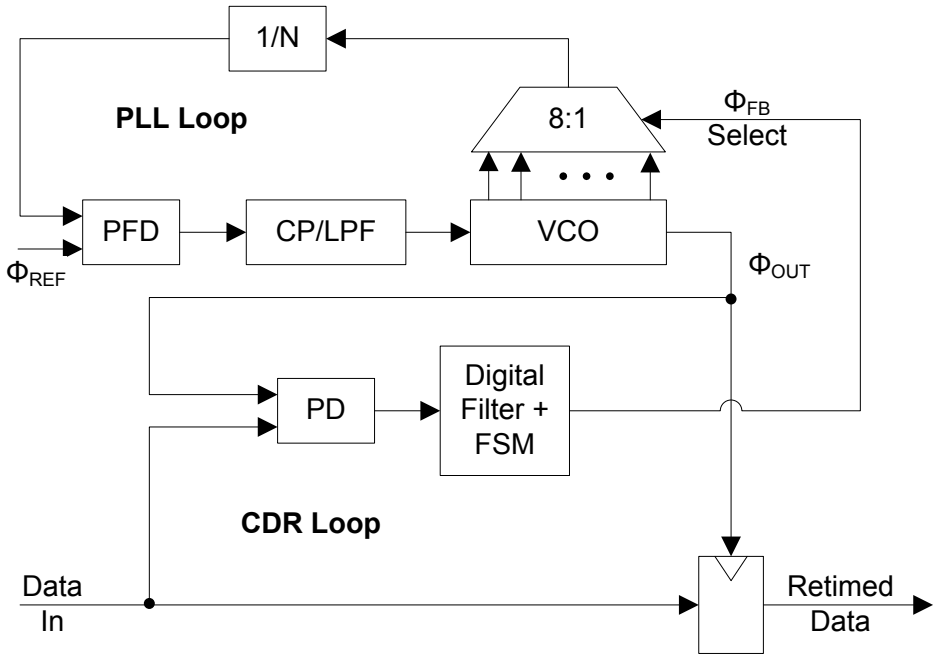

Figure 3.2: Averaging phase interpolator block diagram

The interpolation in this design comes from the PLL loop filter. The PLL initially locks onto the reference frequency and generates K evenly spaced clocks from a multi-phase VCO. Using a single Φ_{OUT} from the VCO, the CDR loop generates a control word that selects one of the VCO phases as the PLL feedback. As the feedback clock changes, the PLL loop forces the VCO to slowly shift towards the new clock phase. As the feedback phase selection is changing between Φ_M and Φ_{M+1}, the PLL loop filter causes the control voltage to settle on a phase in between the two selections.

The averaging phase interpolation uses a digital FSM-based implementation for the coarse interpolator to eliminate the analog requirements of a traditional PI. The chosen phases Φ_M and Φ_{M+1} are selected by the FSM with an interpolation weighting α, producing a repeated pattern of Φ_M and Φ_{M+1} to use as the PLL feedback. In [22], the pattern was given over four clock cycles, allowing this implementation a coarse interpolation in steps of 25% between Φ_M and Φ_{M+1}. With this design, the clocking jitter was limited by the VCO noise.

3.2.1. Simulink Modeled Results

The model for this architecture, shown in Figure 3.3, uses an 8-bit control word from the CDR loop to control the FSM-based phase averaging. The control word has the 3 MSB represent the mux selection, the next 3 bits represent the coarse interpolator α, and the final 2 LSB are the FSM phase averaging pattern control.

The same PLL from Chapter 2 was modified with an 8-phase VCO, but the loop parameters are the same as before. The coarse phase interpolator inside the 8-phase VCO

is the same model as the analog phase mixer used in the PI CDR of Chapter 2. The FSM responsible for the phase averaging control pattern is modeled in Figure 3.4 using a 3-cycle pattern to generate phase averaging weights of 0, 33%, 66%, and 100%. The modeled FSM covers both the boundary cases to reduce the effect of glitching during a switch from α to $\alpha+1$.

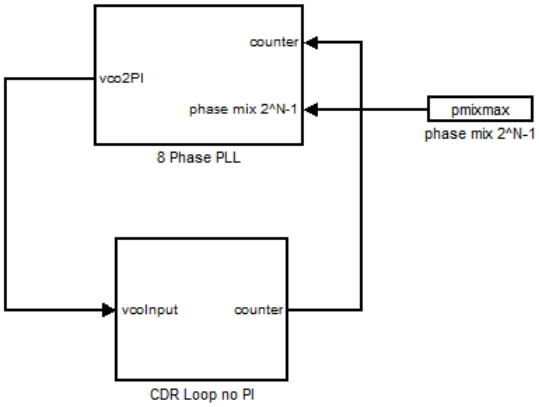

Figure 3.3: Simulink model of averaging phase interpolator

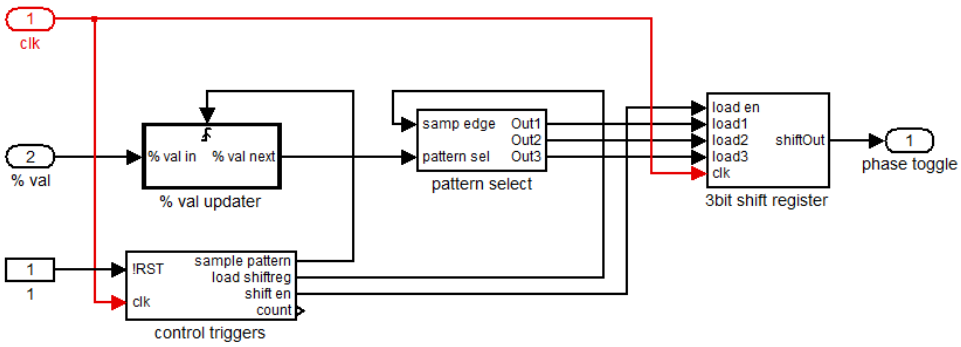

Figure 3.4: Simulink model of the phase averaging FSM control

The CDR loop in the model is shown in Figure 3.5 below. All building blocks are the same as the ones used in the PI CDR model in Chapter 2. The output of the up-down counter is low-pass filtered and requantized to an 8-bit control word with a frequency of

the PLL ω_n x5. Having the control word slightly above ω_n allows the PLL to treat the toggling feedback clock similarly to noise on the reference clock. This is the mechanism by which the averaging phase interpolator creates the fine interpolation.

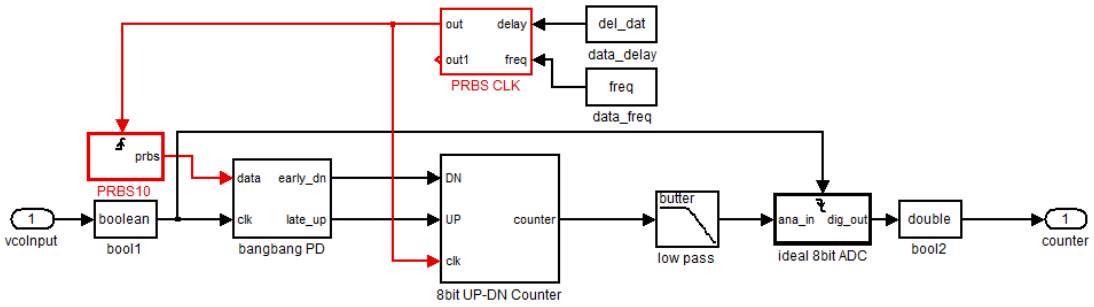

Figure 3.5: Simulink model of the CDR loop

Figure 3.6 is show the CDR digital control word and the PLL V_{CTRL} for the averaging phase interpolator locking onto a PRBS-10 data stream. Once the system locks onto the data, the V_{CTRL} is constantly shifting between Φ_M and Φ_{M+1}, granting the fine phase interpolated Φ_{OUT}.

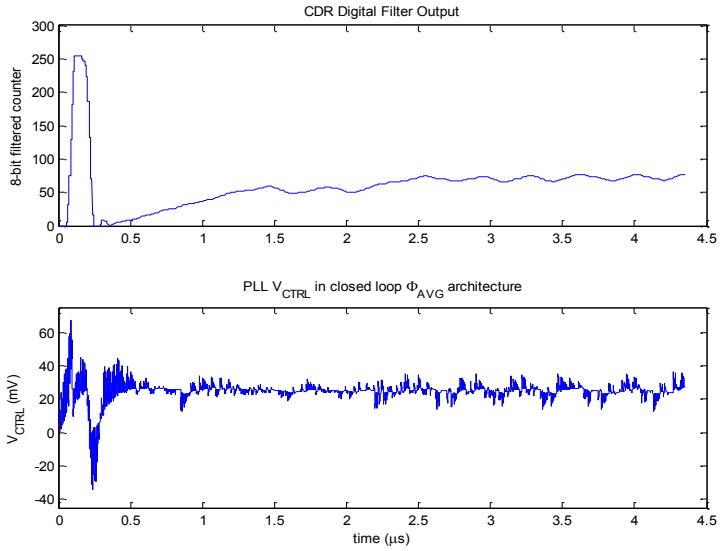

Figure 3.6: CDR digital control word and PLL V_{CTRL} during a track and lock

There are a couple drawbacks to this design. Since the CDR is essentially PLL-driven, the response time of the data tracking is bound by ω_n. This can be adjusted by changing the ω_n of the PLL, but that involves increasing the reference frequency in order to maintain PLL loop stability. The second and more critical drawback is the frequency component caused by the repeated patterns from the FSM. For a coarse interpolation of 33%, the FSM outputs Φ_M for two clock cycles followed by one cycle of Φ_{M+1}. This pattern makes an impact on the frequency response of the system. Employing a ΔΣ modulator to randomize the FSM pattern can alleviate this limitation.

3.3. ΔΣ Averaging Phase Interpolation

Building upon the previous design, inserting a ΔΣ modulator into the feedback clock selection can eliminate the need for an analog phase interpolator while reducing low-frequency repetition [23]. The ΔΣ architecture, shown in Figure 3.7, follows the previous design with a few caveats.

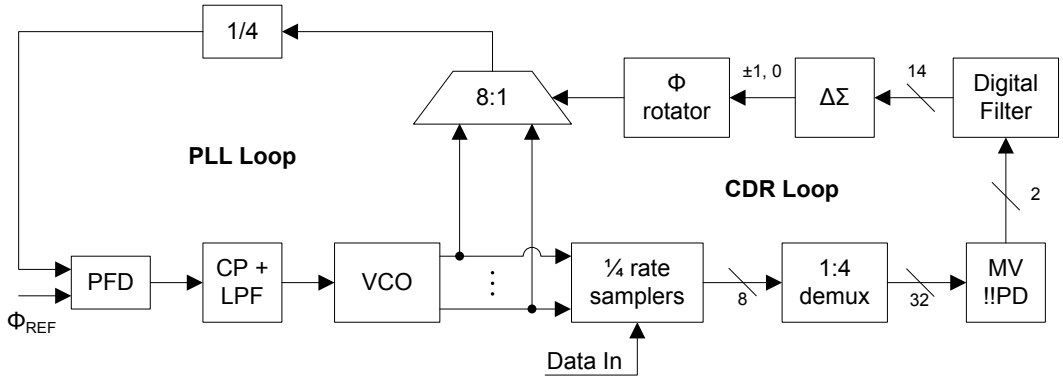

Figure 3.7: ΔΣ averaging phase interpolator block diagram

This ΔΣ architecture uses a quarter-rate !!PD in the CDR loop to demultiplex the incoming data stream. After going through two more stages of half-rate demultiplexing, the decision-making logic only requires a clock of $f_{DAT}/16$. The demuxed phase detector outputs are then majority voted to produce a single !!PD decision which is given to a proportional and integral dual-path digital filter. This filter output is the control word for the PLL feedback selection. Passing the control word through a tri-level quantized second order error-feedback ΔΣ modulator with 8x OSR produces a stream of -1, 0, and +1. This controls a phase rotator implemented as a circular shift register which selects the PLL feedback phase. By means of the ΔΣ modulator, the interpolation control is no longer a repetitious FSM sequence.

3.3.1. Second Order Error-Feedback ΔΣ Modulator

Figure 3.8 shows a second order ΔΣ modulator employing the error-feedback architecture [19]. The previous example used a generic linear realization with two cascaded stages to form the second order modulator. The error-feedback architecture achieves a second order transfer function through a single filter. This architecture can be easily implemented in a DAC, but precision limitations on analog switched capacitor integrators render this architecture impractical for ADCs.

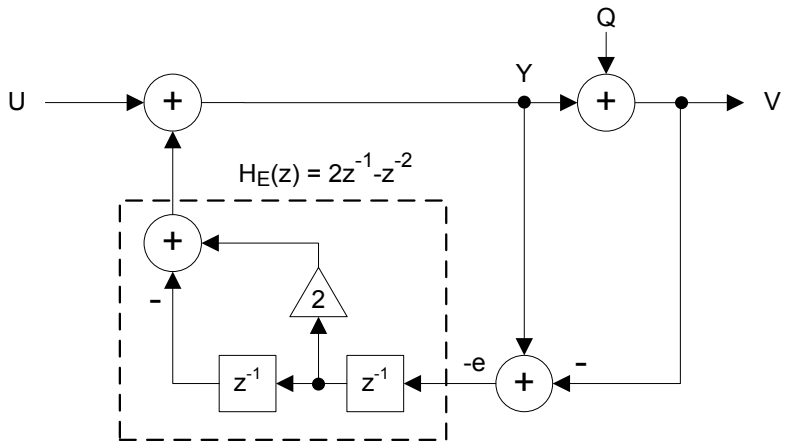

Figure 3.8: Second order error feedback architecture

The error-feedback architecture utilizes the error from quantization, *e*, instead of the quantized output itself. The error is passed through a second order filter and added back into the input signal. Similar to the previous analysis of the generic cascaded modulator, the output V is expressed in terms of the signal and error:

$$-e = Y - V \Rightarrow V = Y + e \ , \ Y = U + H_E e \tag{3.4}$$

$$V = U - H_E e + e = U + (1 - H_E)e \tag{3.5}$$

Then Eq. (3.5) is equated with the desired NTF and STF and the filter transfer function is derived as:

$$NTF = 1 - H_E = (1 - z^{-1})^2 \ , \ STF = 1 \tag{3.6}$$

$$H_E = 2z^{-1} - z^{-2} \tag{3.7}$$

3.3.2. Simulink Modeled Error-Feedback ΔΣ Results

The error-feedback ΔΣ modulator used in [23] had a 3-level output, allowing the ΔΣ modulator output to select between Φ_{S-1}, Φ_S, and Φ_{S+1} to give a range of ±50% from Φ_S. In this model the same ΔΣ modulator output was sent to a low-pass filter to show the DC average of the output. The results in Figure 3.9 were in agreement with Eq. (3.6), when the DC input is the minimum allowed for well-behaved results, the DC output settled to -0.5, corresponding to a Φ_{OUT} half way between Φ_S and Φ_{S-1}. Likewise were the cases for a middle DC input and minimum DC input, shown in Figure 3.10 and Figure 3.11, corresponding to DC outputs of 0 and +0.5, respectively.

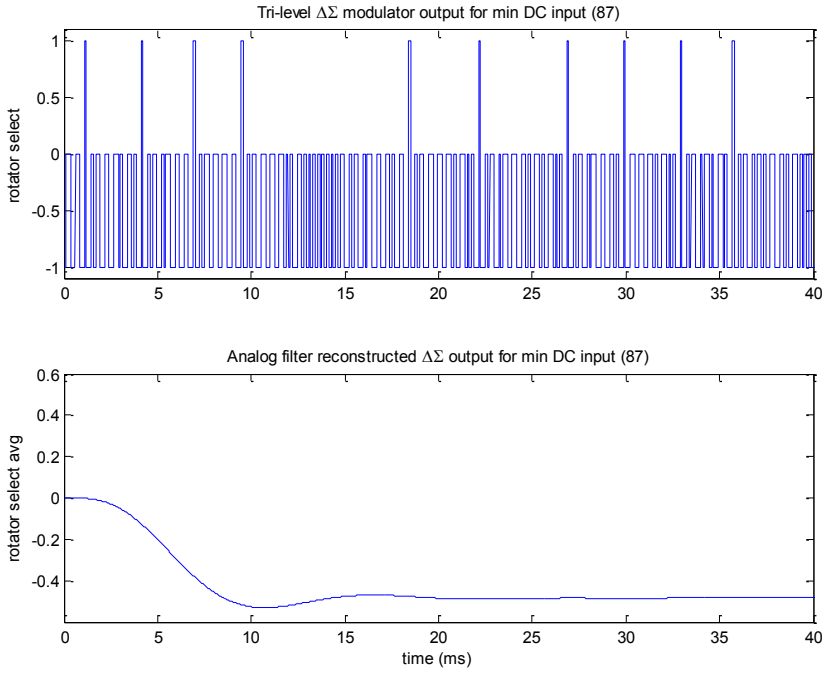

Figure 3.9: Tri-level ΔΣ modulator transient response at DC input = 87 (min)

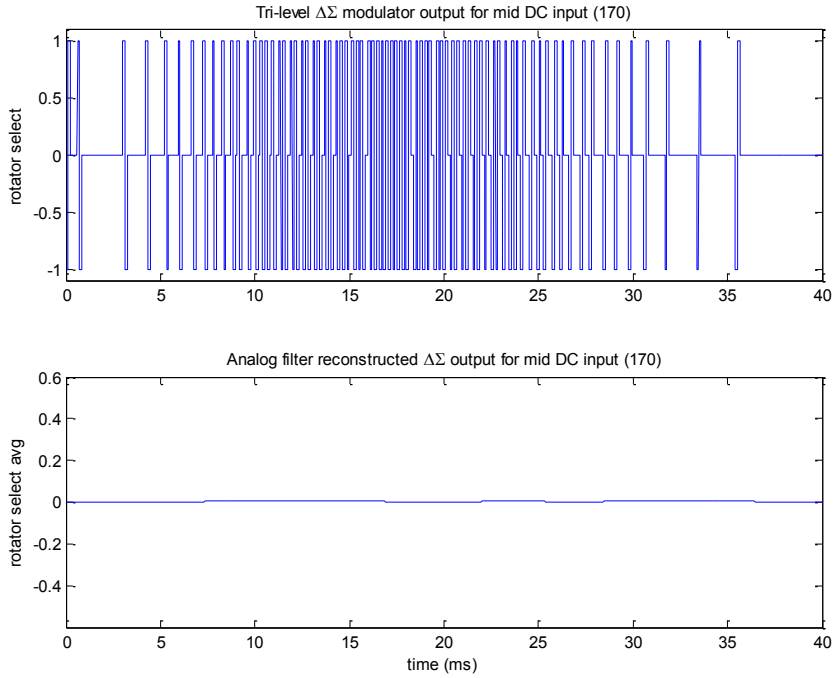

Figure 3.10: Tri-level ΔΣ modulator transient response at DC input = 170 (mid)

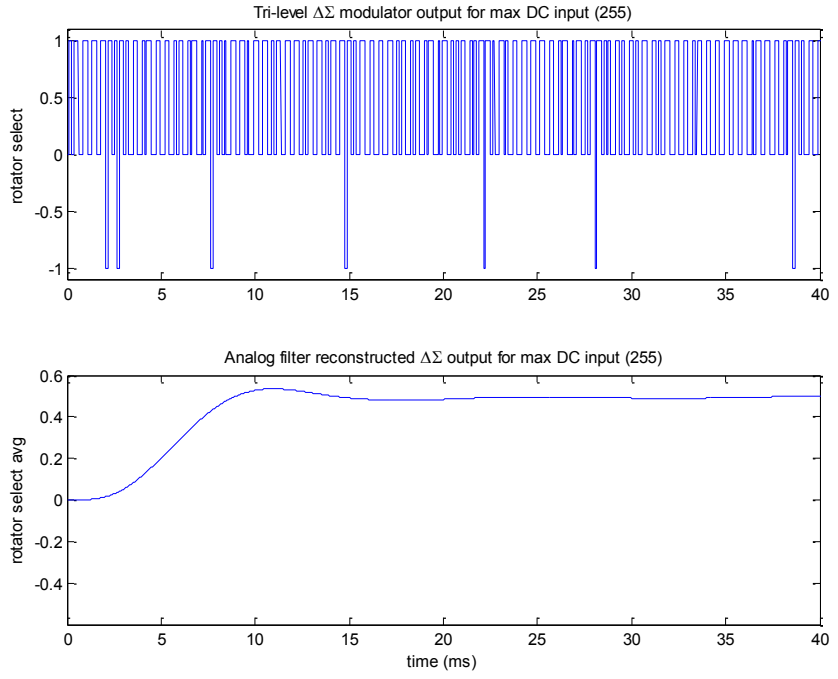

Figure 3.11: Tri-level ΔΣ modulator transient response at DC input = 255 (max)

3.3.3. Simulink Modeled Top-level Results

Since the phase detector is a quarter-rate design, its bang-bang nature repeats once every ±π/4, shown in Figure 3.12, instead of ±π as in the standard !!PD.

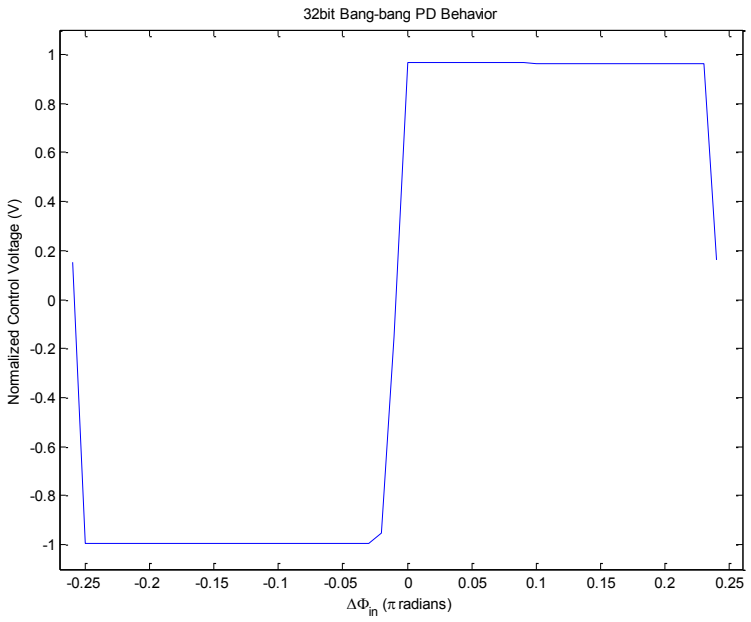

Figure 3.12: 32-bit majority voted phase detector behavior

The top-level signals in Figure 3.13 show the majority-voted PD outputs and the ΔΣ Φ-rotator output. This feedback selection to the PLL caused the locked V_{CTRL} to constantly jump around a settled value.

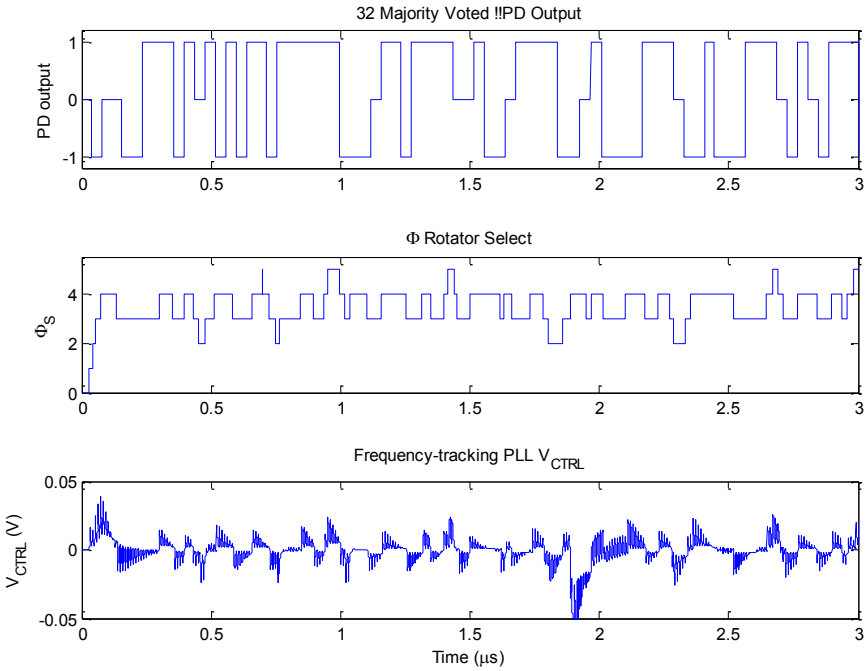

Figure 3.13: Top-level locking results

While this architecture removes the low frequency component associated with the FSM-based interpolation, the actual CDR tracking loop is even slower. Due to the quarter-rate input stage and multiple demux stages, the interpolation control word can only make a filtered decision once every 32 data bits. This is then further limited by the PLL ω_n. For a CDR emphasizing tracking response time, neither the averaging phase interpolator nor the ΔΣ variation will suffice.

Chapter 4. Lock Detection

None of the previously discussed techniques address the limitation of the Alexander phase detector. No matter how well the feedback clock is held constant, so long as the !!PD does not stop detection during a locked state, the PI control logic will always have jitter. The only method to solve this underlying issue is to detect when the CDR is in lock and disable the phase detector. Before the design of a lock detector can be implemented, the strengths and weaknesses of similar techniques are first explored.

4.1. Techniques Related to CDR Lock Detection

The concept of lock detection has already been used in many existing systems [24], [25]. Two key components for a lock detection scheme are the lock detector itself and an in-lock mode. The lock detection is merely a monitor within the decision-making loop that can detect when the system can be considered "in lock." Vital components within the control loop can then use this in-lock flag to toggle between tracking mode and an in-lock mode.

For PLL/DLLs and CDRs, the phase detector is the most critical component in the control loop, and lock detection can be implemented by monitoring the control voltage from the detector output. Some techniques that accomplish this are control voltage thresholds and tri-state phase detectors.

4.1.1. Control Voltage Thresholds

In [24], lock detection was achieved in a PI CDR using a linear phase detector and an analog filter to locate an in-lock control voltage. Similarly, analog VCO control voltage with thresholds can be used to determine lock in type-II PLLs. However, a digital control is preferred for maintaining scalability and accuracy.

Digital lock detection was employed for a clock multiplying PLL with a delay-locked loop (DLL) in [25]. It generates two pulses of length T (an in-lock threshold) for the reference and feedback clocks to increment a duration counter. Once this duration counter reaches its limit, the system recognizes a frequency lock and is switched into DLL mode until it leaves lock. However, this technique does not address jitter from the DLL mode itself.

4.1.2. Tri-state Phase Detectors

Tri-state !!PDs were developed in charge-pump PLL type CDRs to reduce the charge pump's effect on the control voltage [26], [27]. During a no-change PD output, the charge-pump is allowed to neither charge nor discharge the capacitor. This forces the control voltage of the oscillator to remain constant. Since these are binary phase detectors, they follow Alexander's equations and are limited to outputting a no-change only during a period of non-transitioning data.

A dual-path charge-pump PLL type CDR employs a "three-state" phase and frequency detector in [28]. The phase detector is a half-rate multiplexed design. Once in lock, two of the phase detectors sample the data and two sample the transitions. Control logic halts the frequency tracking during the locked state, effectively holding the oscillator control voltage constant. This design utilizes the loop filter to smooth out the bang-bang nature of the phase detector while control logic disables the frequency loop.

These binary detectors all have a mechanism to reduce the effect of the bang-bang effect, but they are still limited by Alexander's equations. In order to eliminate the systematic PI jitter, a !!PD needs to be able to distinguish between a no-change and in-lock state.

4.2. Proposed Lock Detection

The proposed lock detection combined the concepts of a tri-state phase detector with a digital control word threshold. Since the heart of the PI control is a counter, two digital thresholds need to be set as boundaries. Once the counter remains within these thresholds for a number of clock cycles, the system recognizes lock and stop the !!PD from affecting the feedback. The design behind the proposed lock detection technique and the modeled results are detailed in the following sections.

4.2.1. Lock Detection PI Design

The logic flowchart of the lock detection PI is shown in Figure 4.1. Two configurable parameters define the "in-lock" status: a configurable threshold, *T*, and

locking cycle count, L. In order to flag a lock, the live counter must remain within $X \pm T$ for L clock cycles. Each time the counter passes the threshold, X updates to the new counter value. Once the CDR is locked, the analog phase mixer receives the constant X control word instead of the live counter toggling between a few in-lock values.

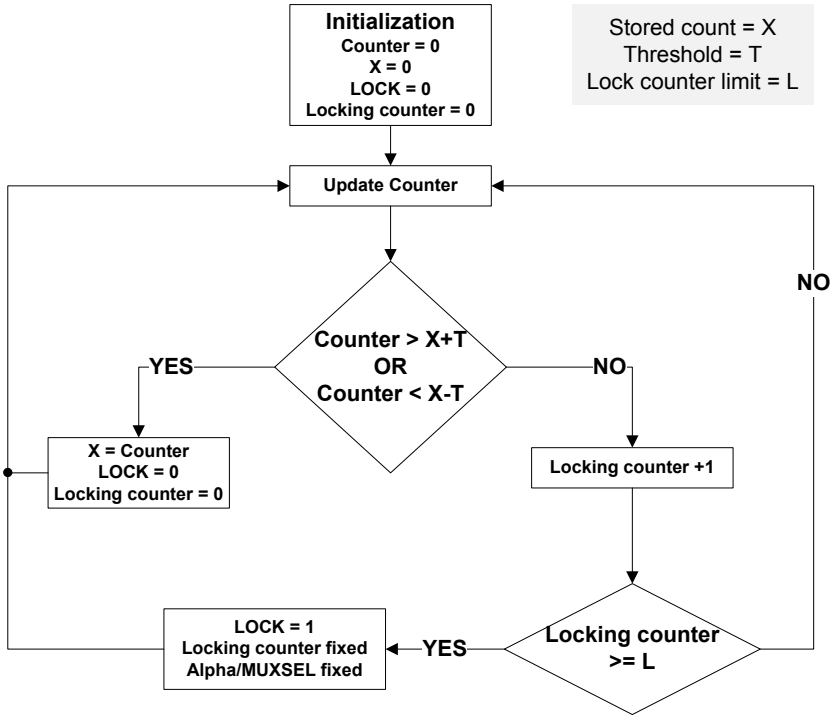

Figure 4.1: Lock detect PI logic flowchart

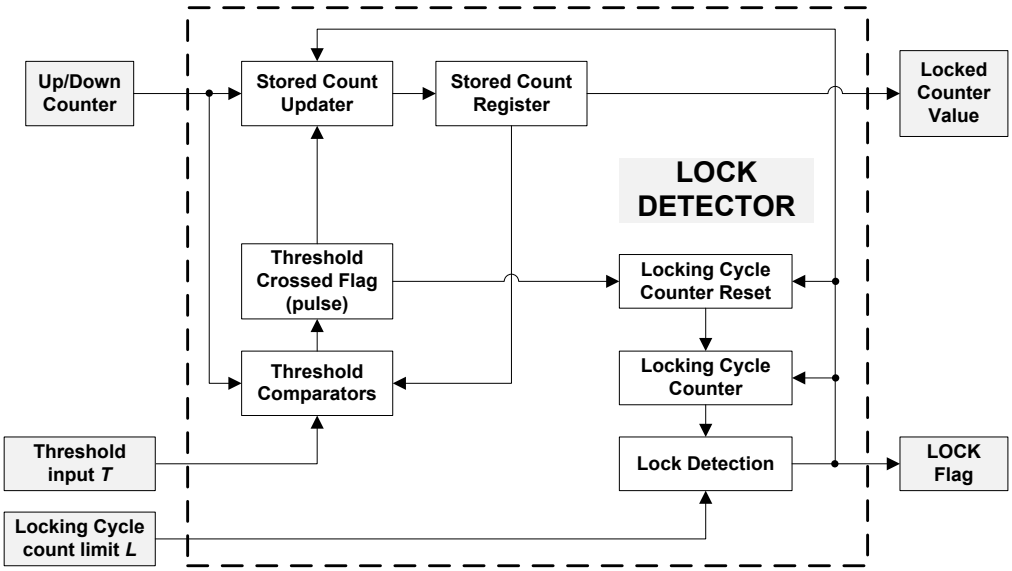

Figure 4.2: Lock detector block diagram

The lock detector logic shown in Figure 4.2 takes in the live counter value along with the two configuration parameters and outputs a static counter value with a lock status flag. During every clock cycle, the input value from the up-down counter is compared to the thresholds. If the counter ever crosses the thresholds, the locking cycle counter resets and value of the counter is stored in the register. This stored value is then used to recalculate all the new thresholds. This process repeats itself until the up-down counter has stayed within the thresholds for L clock cycles. At this point, the lock flag is set and the counter value stored in the register is sent to the constant-phase phase mixer to generate the in-lock Φ_{OUT}.

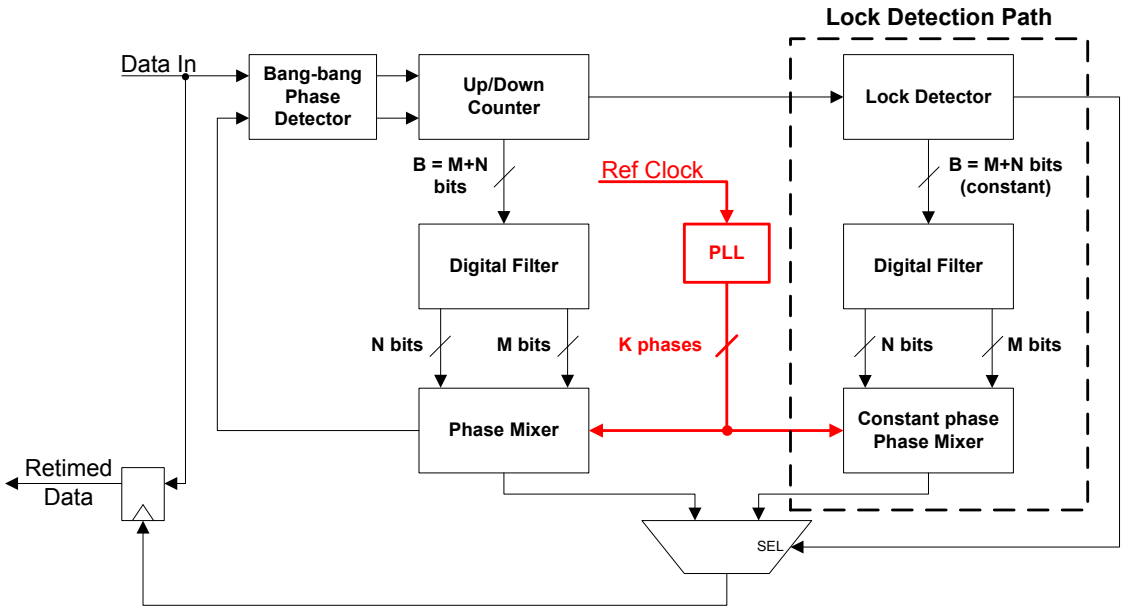

Figure 4.3: Proposed lock detection PI block diagram

Figure 4.3 is the top-level block diagram for the lock detection PI. It employs two phase mixers in the loop: one tracks the live counter while the second mixer is dedicated to the stored value, *X*. While in the tracking stage, both the live and the constant-phase mixers are tracking. However, once in lock, the live mixer exhibits typical PI systematic jitter. The constant-phase mixer is controlled by the constant control word, thus eliminating the systematic jitter. The phase mixers used in the proposed design use the nested interpolator architecture. Since it employs a 5-bit α, this consumes the same total analog mixer current and area as a standard PI.

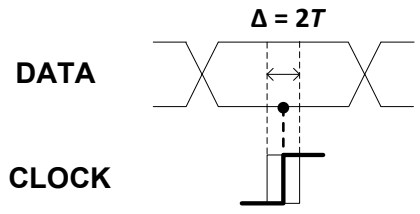

Figure 4.4: Jitter tolerance reduction

The proposed lock detection technique incurs a jitter tolerance reduction during lock, shown in Figure 4.4, due to the stored count, X, having a threshold tolerance of T. This gives a $\Delta = 2T$ region around the ideal sampling point for the locked state. This proposed design compromised jitter tolerance in order to eliminate the systematic PI jitter. An alternative design can increase either the coarse or fine interpolation control words to reduce the width of Δ at the cost of additional power and area.

4.2.2. Lock Detection PI Results

All the blocks in the lock detection PI model, shown in Figure 4.5, were the same as described in the standard PI CDR from Chapter 2. The lock detect modifications were made in the lock detection up-down counter and the dual phase mixers. The lock detection up-down counter, shown in Figure 4.6, has two of the MSB/LSB extraction blocks along with the lock detection logic.

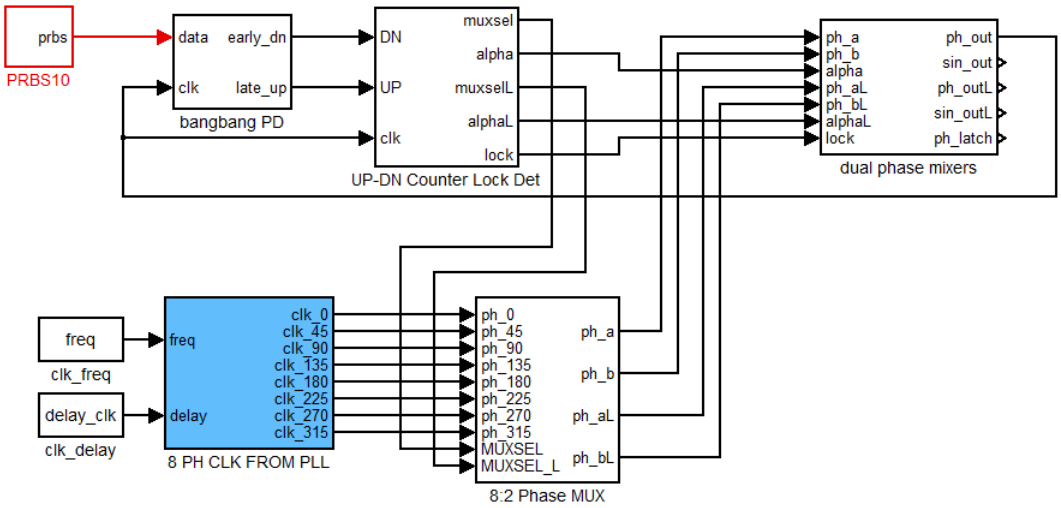

Figure 4.5: Simulink model for the proposed lock detection PI CDR

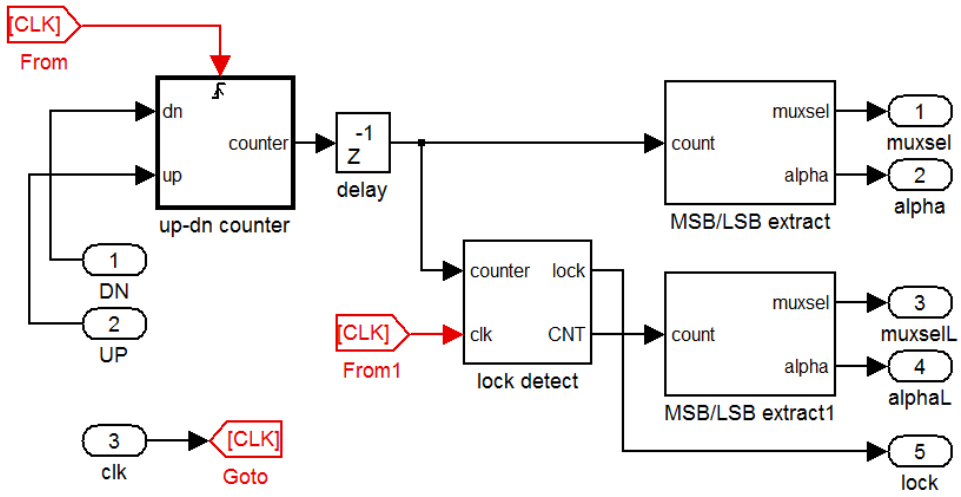

Figure 4.6: Simulink model for the up-down counter

The two MSB/LSB extractors provide both the free-running and in-lock versions of the mux select and α. The lock detection logic, shown in Figure 4.7, maintains the threshold calculation, stored counter value, and locking cycle counter described previously. The output lock flag then passes to the dual phase mixers to select between the live interpolated result and the in-lock interpolated result.

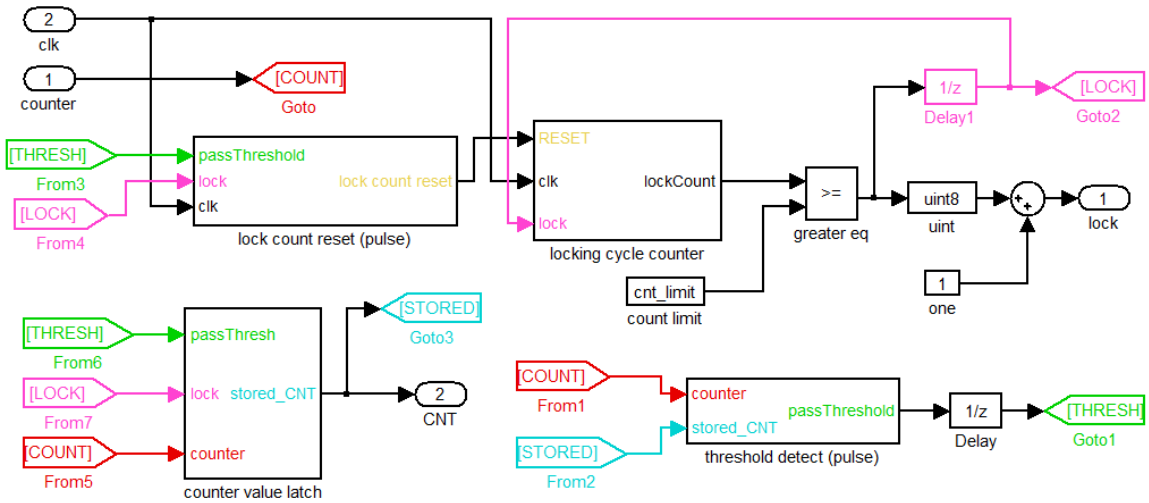

Figure 4.7: Simulink model for the lock detector

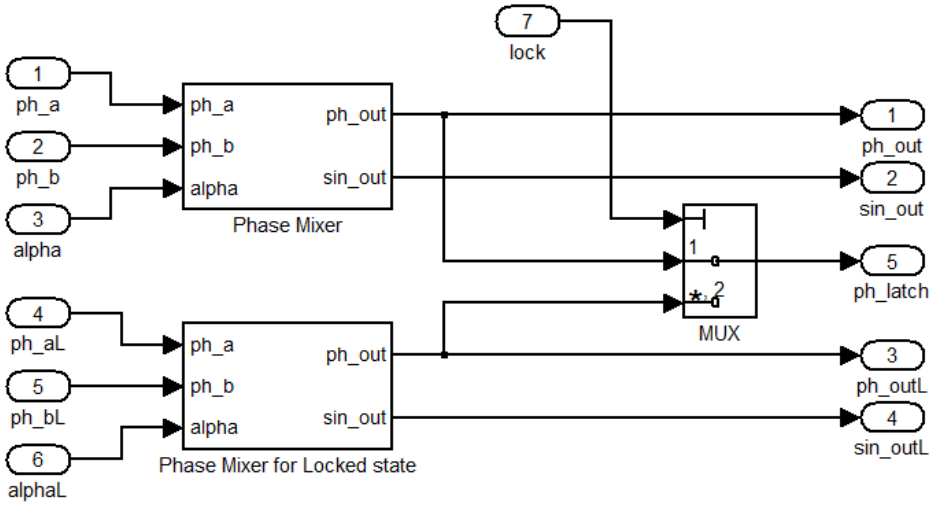

Figure 4.8: Simulink model for the dual phase mixers

The dual phase mixers, shown in Figure 4.8, are just two copies of the PI CDR analog phase mixers from Chapter 2. These run in parallel during the CDR tracking stage with the free-running Φ_{OUT} being used as feedback and the data retiming clock. Once in lock, the free-running Φ_{OUT} continues to be feedback, but the data retiming clock switches over to the in-lock interpolated phase.

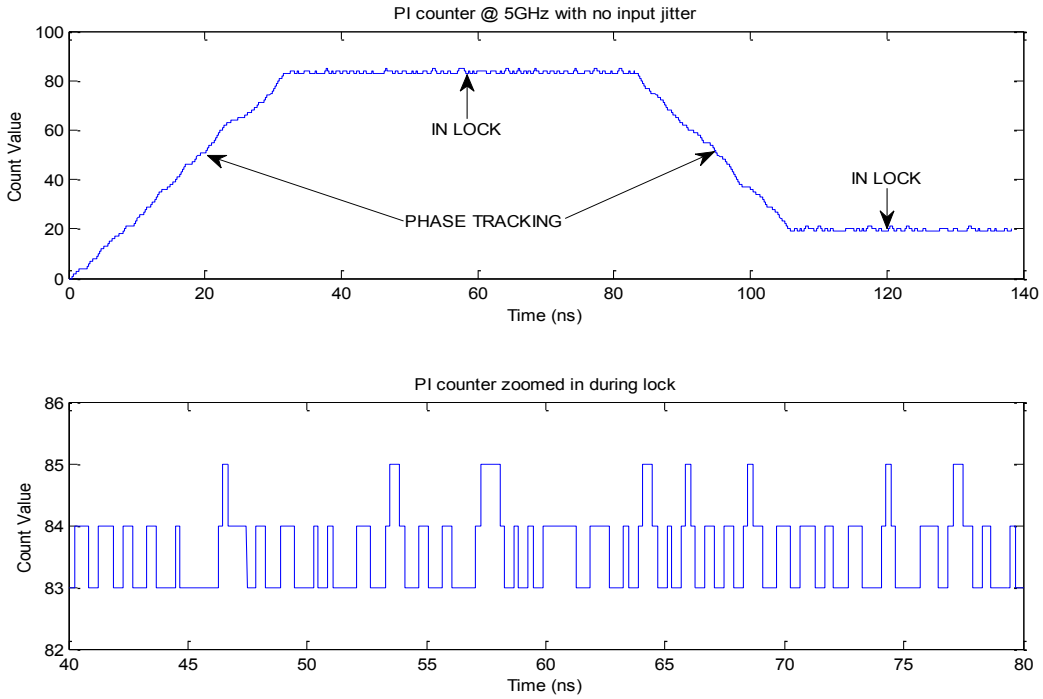

Figure 4.9: PI counter with no input jitter, lock detection disabled

The counter output shown in Figure 4.9 was measured with the lock detection disabled. The counter tracked to the data stream and once it reached lock, it exhibited the bang-bang jitter described in Chapter 2. After a time, the input data was given a $\pi/2$ phase shift and attempts to relock.

Once a lock detection threshold was set, the bang-bang jitter falling within the thresholds was eliminated, shown in Figure 4.10. Zooming in on the counter signal during lock shows the flat control signal of the in-lock interpolator running in parallel with the jittery live control signal. Figure 4.11 shows the eye diagrams exhibiting the elimination of systematic jitter by the lock detection PI compared to a standard PI architecture. Effects of input jitter can be reduced by increasing T, seen in Figure 4.11.e and Figure

4.11.f. The measured jitter in Figure 4.11.a was 0.008UI and jitter in Figure 4.11.d and Figure 4.11.e were both 0.024UI.

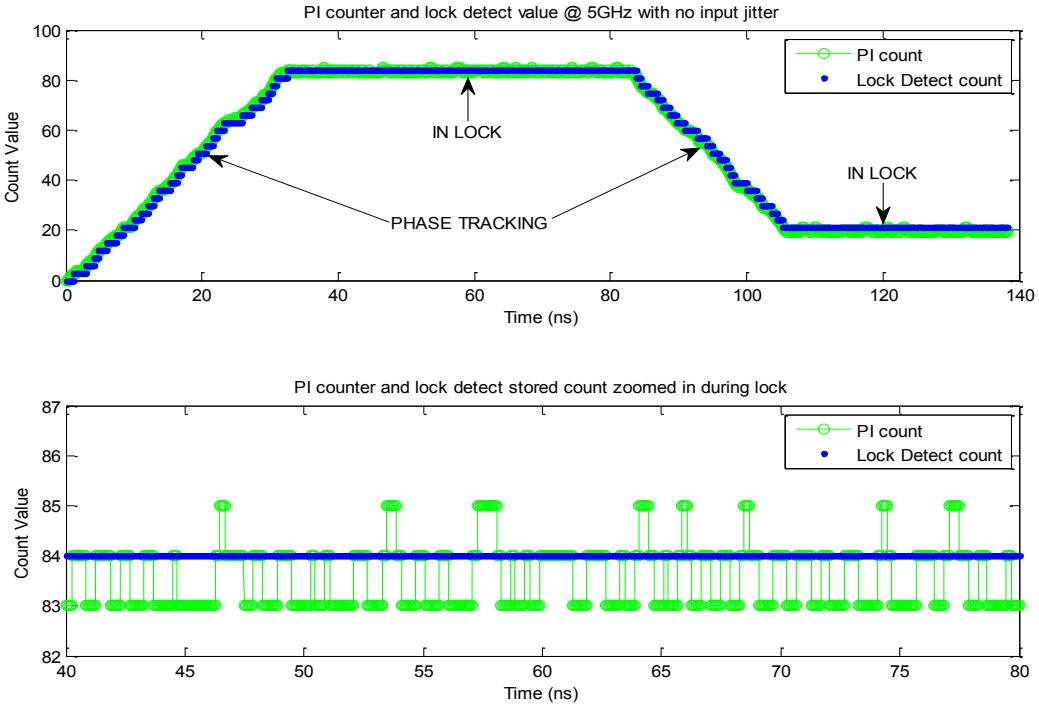

Figure 4.10: PI counter with no input jitter, lock detection T=2, L=50

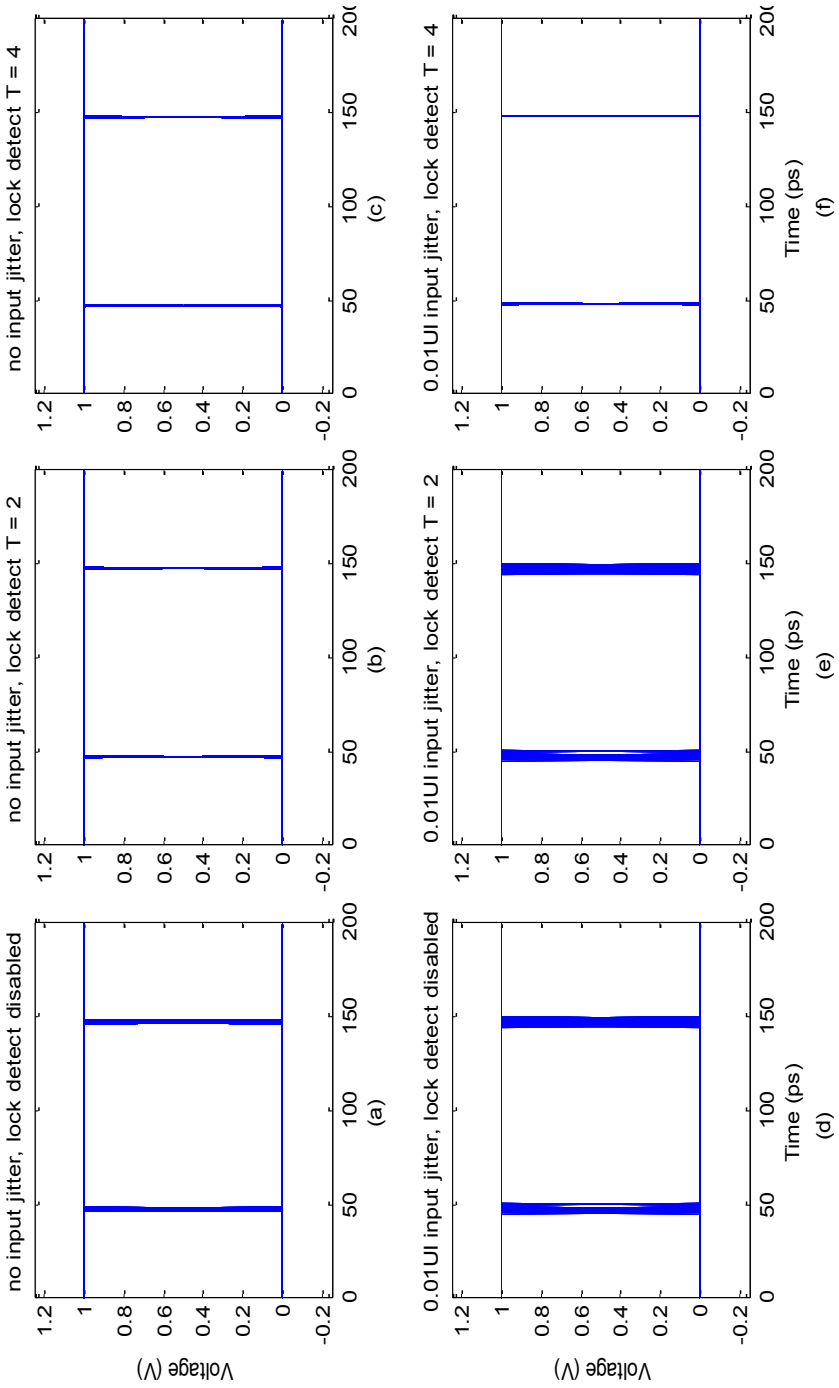

Figure 4.11: Eye diagrams for various lock detect setups

Chapter 5. Conclusions

With the ever-increasing need for higher data rates on inter-chip communications, CDR technology has to be refined in order to maintain quality. As clock speeds increase and voltage levels decrease, more importance is placed on fast and accurate recovery. Dual-loop PLL-PI clock and data recovery systems are attractive for use in source-synchronous clock receivers with multiple data lanes. However, their architectural jitter limitations pose a challenge for designers.

The building blocks for dual-loop PLL-PI architectures were explored for possible modifications in order to reduce systematic jitter. $\Delta\Sigma$ modulation was also examined as a potential means to shape the systematic jitter to higher frequencies. Three existing technologies that aim to reduce phase interpolation jitter were studied in order to understand the limitations in various architectures.

Many existing methods of lock detection and control voltage thresholding were compared and improved upon to reach the proposed design of a lock detection phase interpolator CDR. The proposed design successfully eliminated phase interpolator systematic jitter at the cost of reduced jitter tolerance. The successful modeling of the lock detection PI CDR opens the way to further testing of this concept and efforts can be turned towards transistor-level design.

References

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012-2017," Feb. 2013. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf.

[2] "The Internet Systems Consortium Domain Survey," July 2012. [Online]. Available: http://ftp.isc.org/www/survey/reports/current.

[3] "Intel Public Roadmap for Desktop, Mobile, Data Center," Intel, 2012. [Online]. Available: http://www.intel.com/content/www/us/en/processors/public-roadmap-article.html.

[4] "JEDEC DDR4 SDRAM Standard," Sept. 2012. [Online]. Available: http://www.jedec.org/standards-documents/docs/jesd79-4.

[5] M.T. Hsieh, G.E. Sobelman, "Architectures for multi-gigabit wire-linked clock and data recovery," *IEEE Circuits and Systems Magazine,* vol. 8, no. 4, pp. 45-57, 2008.

[6] H.A. Collins, R.E. Nikel, TriCN Associates LLC, "High-Speed Source Synchronous Interface Design," *Insight,* vol. 4, no. 3, pp. 19-21, 1999.

[7] "Matlab Eye Diagram Help," Mathworks, 2011.

[8] S. Sidiropoulos, M.A. Horowitz, "A semidigital dual delay-locked loop," *IEEE Journal of Solid-State Circuits,* vol. 32, no. 11, pp. 1683-1692, Nov. 1997.

[9] J.D.H. Alexander, "Clock recovery from random binary signals," *Electronic Letters,* vol. 11, no. 22, pp. 541-542, Oct. 1975.

[10] B. Razavi, Design of Analog CMOS Integrated Circuits, Int. Ed., Beijing, P.R. China: Tsinghua University Press, 2001.

[11] R.C. Walker, "Designing Bang-Bang PLLs for Clock and Data Recovery in Serial Data Transmission Systems," in *Phase-Locking in High-Performance Systems From Devices to Architectures*, Hoboken, NJ, IEEE Press, 2003, pp. 34-45.

[12] I. Galton, "Delta-Sigma Fractional-N Phase-Locked Loops," in *Phase-Locking in High-Performance Systems From Devices to Architectures*, Hoboken, NJ, IEEE Press, 2003, pp. 23-33.

[13] B. Gilbert, "A New Wide-Band Amplifier Technique," *IEEE Journal of Solid-State Circuits,* Vols. SC-3, no. 4, pp. 353-365, Dec. 1968.

[14] R. Kreienkamp, U. Langmann, C. Zimmermann, T. Aoyama, H. Siedhoff, "A 10-gb/s CMOS clock and data recovery circuit with an analog phase interpolator," *IEEE Journal of Solid-State Circuits,* vol. 40, no. 3, pp. 736-743, Mar. 2005.

[15] A. Van den Bosch et al, "A 10-bit 1-GSample/s Nyquist Current-Steering CMOS D/A Converter," *IEEE Journal of Solid-State Circuits,* vol. 36, no. 3, pp. 315-324, 2001.

[16] E. Siragusa, I. Galton, "A digitally enhanced 1.8-V 15-bit 40-MSample/s CMOS pipelined ADC," *IEEE Journal of Solid-State Circuits,* vol. 39, no. 12, pp. 2126-2138, Dec. 2004.

[17] T. Shui, R. Schreier, and F. Hudson, "Mismatch Shaping for a Current-Mode Multibit Delta-Sigma DAC," *IEEE Journal of Solid-State Circuits,* vol. 34, no. 3, pp. 331-338, 1999.

[18] B. Razavi, Principles of Data Conversion System Design, New York, NY: Wiley-IEEE Press, 1995.

[19] R. Scheier and G.C. Temes, Understanding Delta-Sigma Data Converters, Hoboken, NJ: IEEE Press, 2005.

[20] T.A.D. Riley, M.A. Copeland, and T.A. Kwasniewski, "Delta-sigma modulation in fractional-N frequency synthesis," *IEEE Journal of Solid-State Circuits,* vol. 28, no. 5, pp. 553-559, May 1993.

[21] Y. Jiang, A. Piovaccari, "A compact phase interpolator for 3.125G Serdes application," *Southwest Symposium on Mixed-Signal Design,* pp. 249- 252, Feb. 2003.

[22] P. Larsson, "A 2–1600-MHz CMOS Clock Recovery PLL with Low-Vdd Capability," *IEEE Journal of Solid-State Circuits,* vol. 34, no. 12, pp. 1951-1960, Dec. 1999.

[23] P.K. Hanumolu, "Design Techniques for Clocking High Performance Signaling Systems," Ph.D. dissertation, Dept. of Elect. and Comp. Eng., Oregon State Univ., Aug. 2006.

[24] C. Kromer, et al., "A 25-Gb/s CDR in 90-nm CMOS for High-Density Interconnects," *IEEE Journal of Solid-State Circuits,* vol. 41, no. 12, pp. 2921-2929, Dec. 2006.

[25] S.L.J. Gierkink, "A 1V 15.6mW 1–2GHz −119dBc/Hz @ 200kHz clock multiplying DLL," *Custom Integrated Circuits Conference, 2008,* pp. 439-442, Sept. 2008.

[26] D. Rennie, M. Sachdev, "A Novel Tri-State Binary Phase Detector," *IEEE International Symposium on Circuits and Systems,* pp. 185-188, May 2007.

[27] B. Lai, R.C. Walker, "A Monolithic 622Mb/s Clock Extraction Data Retiming Circuit," *IEEE International Solid-State Circuits Conference,* pp. 144-145, Feb. 1991.

[28] A. Rezayee, K. Martin, "A 9-16Gb/s clock and data recovery circuit with three-state phase detector and dual-path loop architecture," *Proceedings of the 29th European Solid-State Circuits Conference,* pp. 683-686, Sept. 2003.