CLOCK SYNTHESIZER DESIGN WITH ANALOG AND DIGITAL PHASE LOCKED LOOP

BY

DA WEI

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, 2014

Urbana, Illinois

Adviser:

Professor José Schutt-Ainé

ABSTRACT

As process technology has aggresively scaled, the demand for fast, robust computing has grown tremendously. With the rise of large scale data centers to handhold mobile devices, the desire for faster, low-power integrated inter-IC communication protocols is at an all-time high. This trend, together with the widespread presence of broadband Internet, multi-core CPU and system-on-chip designs have pushed the demand for data rate in wireless and wireline links into multi-Gbps ranges.

In order to fulfill the increasing demand for high data rate links, the performance of the I/O channel needs to scale proportionally. But, due to the lag in interconnect scaling and PCB material improvements, the channel bandwidth has not scaled with data rates. The channel is therefore the biggest bottleneck in high-speed I/O communication links. In order to fulfill the demand of high-speed multi-Gigabit data transmission capacity, the system needs to incorporate robust, low-power fast-locking on-chip clocking circuits. The fundamental building block of every clocking circuit used in high-speed links is the Phase-Locked Loop (PLL).

This thesis provides an in-depth analysis of basic analog PLL theory, architecture, transistor level design. The all digital counterpart of the analog PLL will also be presented for its ultra low power and small footprint. The design and complete simulation result of a basic clock synthesizer circuit will be used as a design example for both the analog and digital PLL circuit design so that the readers will have in-depth understanding of the details of designing an on-chip PLL circuit. An overview of high speed communication scheme of serial link will also be discussed to illustrate the importance of the PLL. It concludes by laying the motivation for future applications of PLLs and the need for further research in the area of low-power, high-fidelity fast-locking PLLs.

To my family and friends, for their love and support.

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Professor José Schutt-Ainé for the continuous support of my graduate study and research. His support is always there whenever I need it. His guidance helped me to overcome numerous difficulties encountered during the research. I could not have imagined having a better advisor and mentor for my graduate study.

Besides my advisor, I would like to thank Professor Pavan Kumar Hanumolu for his patience, motivation, enthusiasm, and immense knowledge. He answered me every single questions patiently and thoughtfully despite of his busy schedule and it is because his help makes this thesis possible.

I would also like to express my sincere gratitude to my wonderful colleagues, mentor and friends in graduate school: Rishi Ratan, Yubo Liu, Woo-Seok Choi, Xu Chen, Drew Newell, Xinying Wang, Jerry Yang, Maryam Hajimiri, Tom Comberiate, Romesh Nandwana, Mrunmay Talegaonkar, Ahmed Elkholy, Saurabh Saxena, Tejasvi Anand, Guanghua Shu for the stimulating discussions the sleepless nights we were working together and for all the fun we have had together.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation

Over the last few decades, with the developments in Semiconductor Fabrication Technology (SFT) together with the advances in Integrated Circuit technology (IC) scaling have ignited the exploding growth in computing. This aggressive scaling triggered continuously growing demand for faster data rates, which in turn pushes the clock rate in data links into multi-GHz and the data rate into Mulit-Gbps region. For consumer level computing devices the dramatic increases in processing power, together with transistor scaling and shifts in computer architectures from single core to many-core system and the more and more popular SoC design require the data and information can be accessed with ultra low latency and power consumption. This makes the adaption of serial communication links like PCI Express and DDR4 memory interface in personal computers and mobile devices unavoidable. In enterprise level, the population of broadband Internet connection triggers the thriving of cloud service and the large data centers behind it. In the applications in the data center, high speed network connections like 40GbE/100GbE or infiniband is critical part for rack to rack communication or to build a high volume storage area network. This in turn triggers the increasing needs for 40G/100G network switch. Circuits like DDR4 memory interfaces and backbone of 100G Ethernet switch push the need for robust, high-speed and low-power inter-IC communication schemes to a new height. In order to fulfill this requirement the performance Input/Output (I/O) links should be scaled properly. With this aggressively increasing bandwidth demand, the solution is obvious: increase the width of the data bus. But quickly we ran into trouble. With higher and higher frequency the clock skew, cross talk and latency becomes unbearable. In addition, the limited I/O pin count

of chip package and PCB wiring constraints and the leakage power consumed by the channel drivers, high speed serial link technology becomes more and more handy.

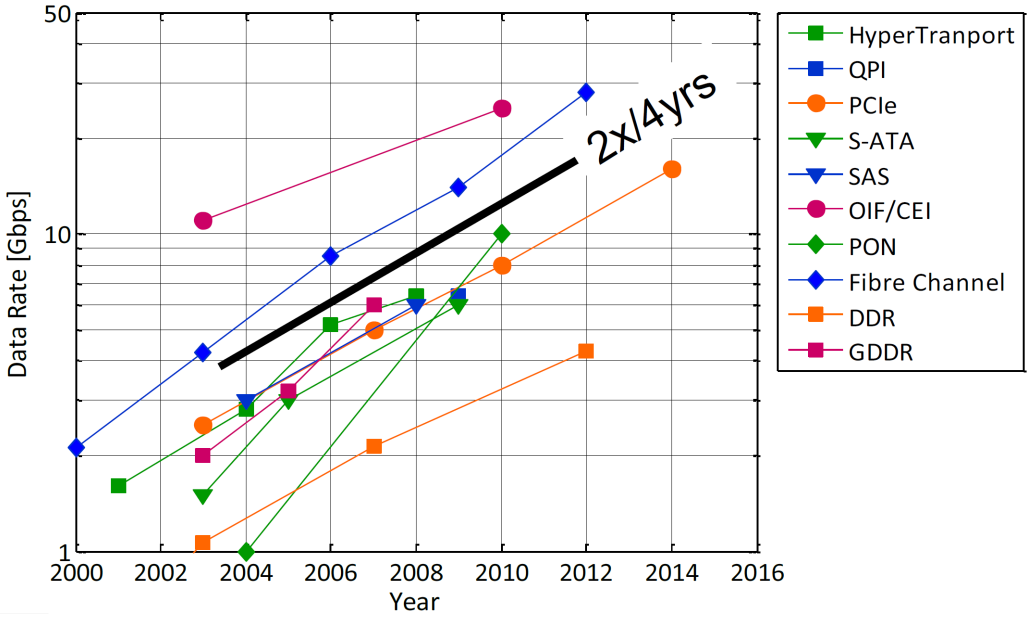

Figure 1.1: Per Pin Data Rate Trend

Figure 1.1 shows the current trend in per pin data rate scaling of high speed I/O signal link as estimated by the International Solid State Circuits Conference's (ISSCC) 2011 bi-annual IC road-mapping report [1]. According to it, the data rate per pin will be scaled by a factor of 2 every 4 years. However, since technology and material that used to fabricate the backplane has not improved proportionally with SFT, the bandwidth of off-chip I/O links is severally limited and becomes the major bottleneck in overall system performance. The outdated channel fabrication technology and the limitation of packaging and I/O pins together exposes great challenges for circuit designers who have to overcome the significantly increased transmission line loss, crosstalk and signal distortion resulting from the low bandwidth and low quality links at high clock frequencies. This makes the continuous improvement on phase-locked loops (PLL) design and the application of it for on-chip clock synthesis and recovery unavoidable.

Although the demand for circuit designers increases from year to year and numerous innovative researches have be done in the field of integrated high-frequency clocking circuits for last decades, a comprehensive tutorial

style documentation of using Electronic Design Automation (EDA) tools to simulate a PLL based clock synthesizer circuit is still missing [2]. Most of the prominent works only focus on the system level and theoretical level description, few of them get into the details of the building blocks. The lack of documentation of complete design methodology makes new students in this field extremely hard to learn from the literary works.

Thus the motivation of this thesis is by using a complete design example of an analog PLL and its digital counterpart to introduce the design methodology and simulation techniques for a mixed signal design engineers who needs the basic skills to conduct a transistor level mixed signal circuit design project on his or her own.

1.2 Outline

1. Chapter 2 provides an overview of a general high speed serial communication link and the significance of PLL in the link.

2. Chapter 3 provides an overview of the theoretical analysis of a type two second order analog PLL and an introduction of its building blocks.

3. Chapter 4 describes the detailed modeling and transistor design and validation method of a basic analog PLL

4. Chapter 5 introduces the advantages of all digital PLL and the theoretical analysis of how to convert an analog PLL into its digital counterpart.

5. Chapter 6 introduces detailed modeling and simulation of the all digital PLL.

6. Chapter 7 concludes the thesis with a brief discussion of the future development of the all digital PLL based clocking circuit designed earlier and propose possible design to achieve ultra low power operation.

CHAPTER 2

BASIC SERIAL LINK COMMUNICATION OVERVIEW

A generalized high speed link model is shown in figure 2.1 [3]. The PLL at transmitter side uses a low frequency reference clock that is usually generated by crystal and mutliplies it in form of a synthesizer to achieve the target frequency needed for signal transmission through the serializer and transmitter driver. The serializer accepts the incoming parallel data and converts it into a serial data stream and send them to the transmitter (TX). The transmitter then generates a series of pulses according to the modulation used and the data is sent through the channel. On the receiver side, once the data bit stream has arrived, the clock data recovery block estimates the clock transitions embedded inside the data stream by using a PLL to track the data transitions. The estimated clock is then fed back to the receiver to sample the data. Once the data and clock are both recovered, they are fed into the deserializer to convert the it back to its original parallel form.

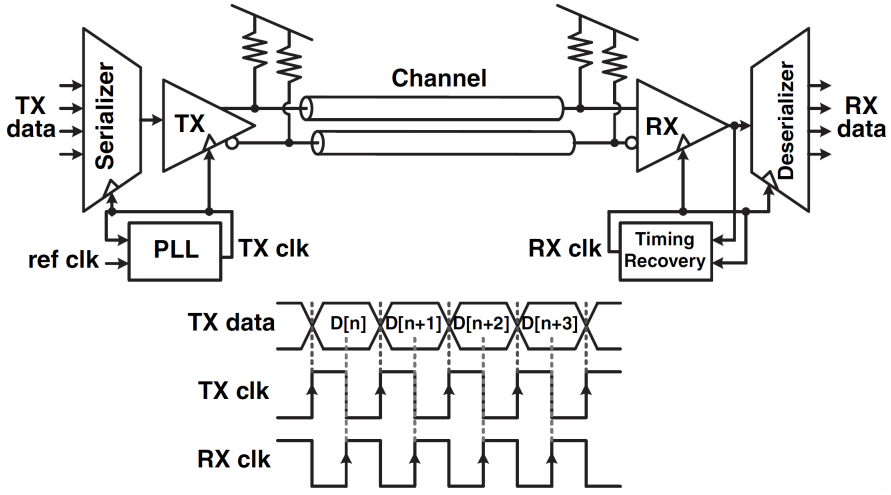

Figure 2.1: Basic High-speed electrical Link System

As discussed earlier, even though the data rate requirement scales with SFT, the inter-IC channel bandwidth is still severally limited by nonlinear

effects. A typical backplane channel interface and its associate non ideal effects are shown in figure 2.2 [3].

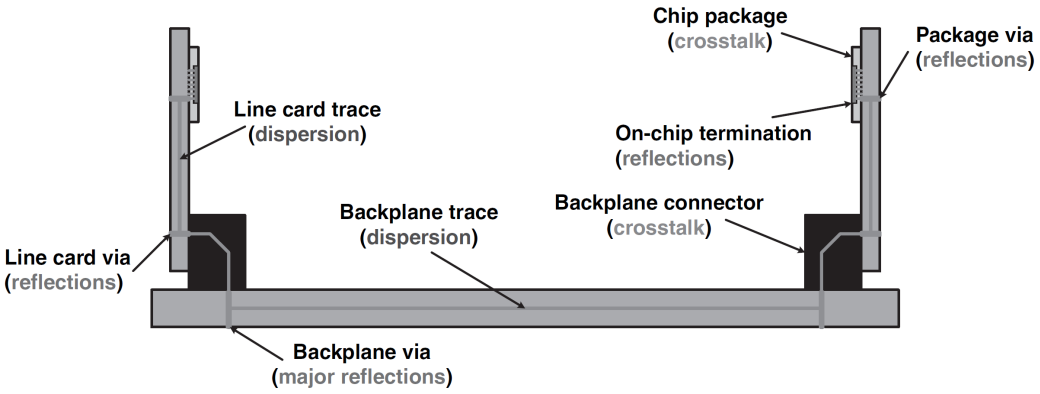

Figure 2.2: Typical Backplane Channel Interface

The channel here is an electrical path that connects the TX and RX together and primarily consists of the printed circuit board (PCB) traces, vias, connectors. The characteristics of the channel are typically modeled by S-parameters that are obtained either through measurement from a vector network analyzer (VNA) or via a computational electromagnetic modeling software like Ansys HFSS. Once the channel is fully characterized, implementation of a high signal fidelity communication system that is fast, robust and can compensate the various kinds of losses due to impedance mismatch between connectors, substrate loss and cross talk effects becomes the main design objective/problem. Additionally, apart from these requirements nowadays power efficiency and desire for small footprint on wafer have also become prominent design objectives especially in consumer applications.

Figure 2.3 and 2.4 show the eye diagram outputs from a backplane channel interface at 1Gbps and 10Gbps data-rates respectively. Notice that the eye is fully open at 1Gbps but at 10Gbps the signal is almost indistinguishable from the noise at the receiver side due to the tremendous loss and distortion incurred along the channel.

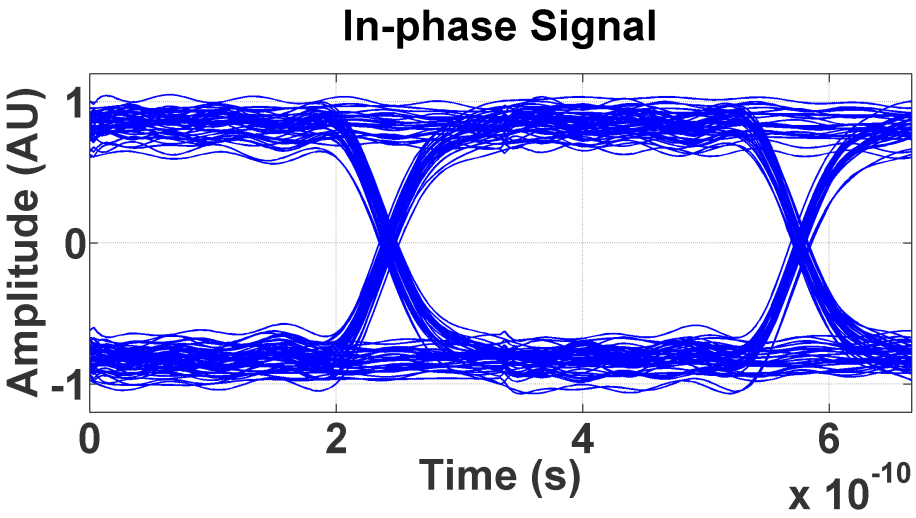

Figure 2.3: 1Gbps Backplane Link Eye Diagram

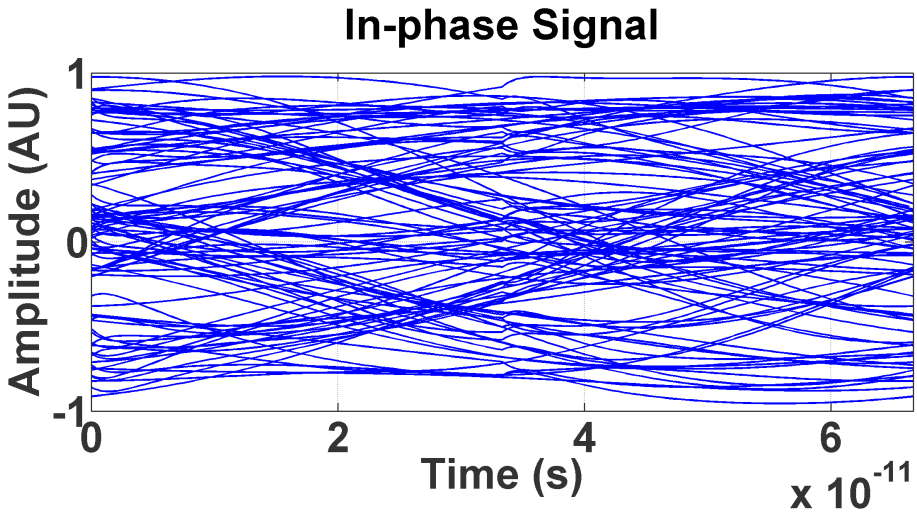

Figure 2.4: 1Gbps Backplane Link Eye Diagram

The significant difference between 1Gbps and 10Gbps signal actually can be easily foreseen from the frequency response of the channel. From Fig.2.5 [4] we can discover that at 10GHz the signal will be severely attenuated. The weaker the signal is, the more vulnerable it is against thermal noise and cross talk, thus higher the jitter. The jitter is the variation of the time when a signal transition occurs. Fig. 2.6 [5] and Fig. 2.7 [6] have covered the basic conception of the jitter and common jitter sources. As we can see, it the high jitter and small unit interval (UI) window at 10 GHz result in this completely closed eye.

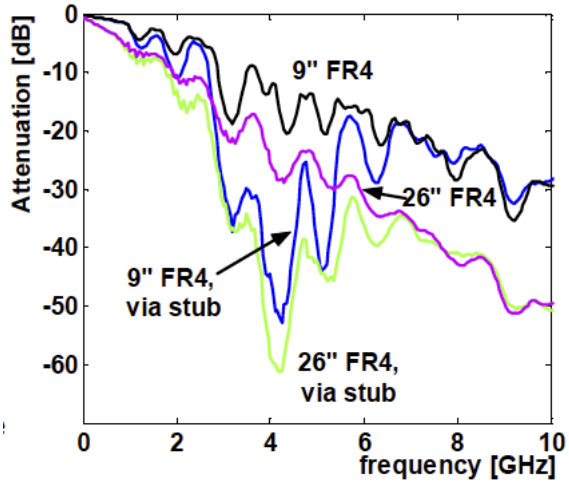

Figure 2.5: Typical Serial Link Channel Responses

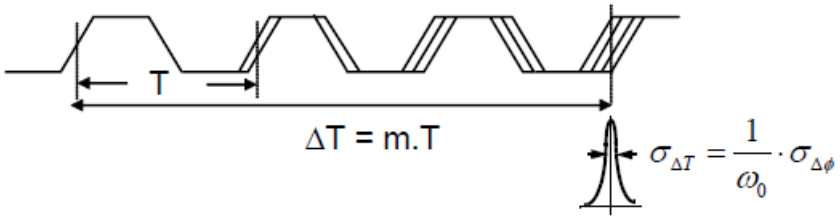

Figure 2.6: Time Domain Jitter

In addition to the channel loss, the power consumption at high data rate becomes more and more severe. Figure 2.8 shows the power break down of a 4.8 Gbps serial link designed for a fully buffered DIMM system [7].

For the design, more than 80% of power is consumed by the clocking system and 50% of this power is claimed by PLL (both TX and RX PLL). According to the data rate per pin trend discussed in previous section, the PLL will claim more and more portion of the total power consumption at higher data rates. This again indicates that the goal of the mixed signal designer is to constantly explore new methods to improve the PLL and associate clocking system design so that the transmitter can have high accurate clock with more pure spectrum that minimizes the jitter and the receiver can recover the data despite of the high channel loss and at meanwhile maintains high power efficiency.

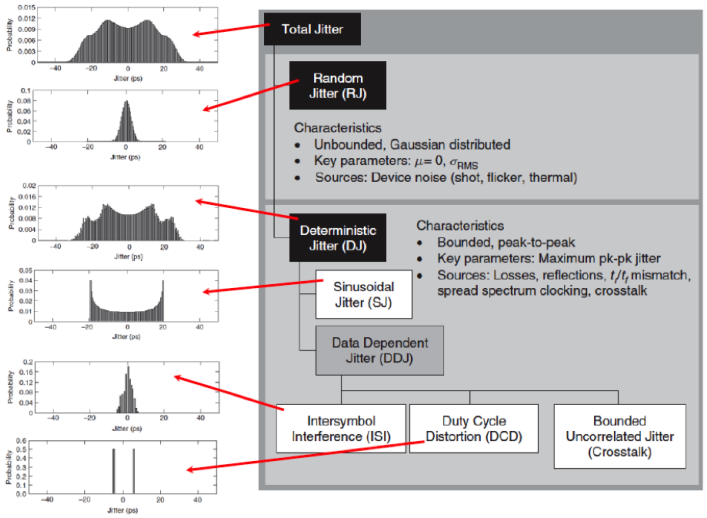

Figure 2.7: Typical Jitter Sources

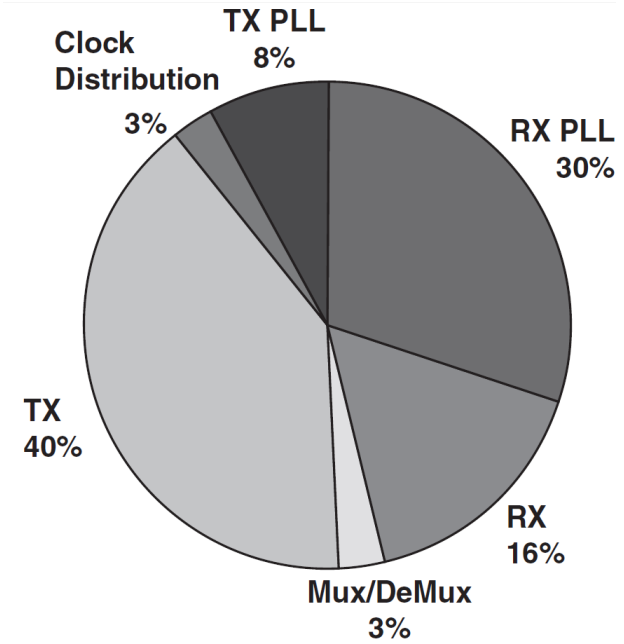

Figure 2.8: Power Break Down of a 4.8Gbps Serial Link

CHAPTER 3

THEORETICAL ANALYSIS OF PLL BUILDING BLOCKS

3.1 PLL Applications

As the the most fundamental and ubiquitous circuits, PLLs can be found in any communication (wireless or wireline) and high speed digital systems. It is a device that mainly used for synchronization purpose. The main function includes:

1. Synthesizing high speed clock from a low frequency reference clock, a.k.a a clock synthesizer.

2. Synchronizing the local generated clock with the clock that embedded in the incoming data stream, a.k.a a clock recovery circuit.

3. Compensating the clock skew in the clock distribution network, a.k.a a zero delay buffer.

4. Extract the carrier signal when carrier suppressed modulation is used, a.k.a. a carrier extractor.

Over last decade, pure CMOS implementation of PLL gained lots of attentions due to the low cost and increasing demand for high speed and accurate clock generator in high performance digital systems. This section will focus on the clock synthesizer PLL analysis and design.

3.2 Basic PLL Building Blocks

A typical PLL is a negative feedback loop (Figure 3.1) that compares the phase of generated clock and the phase of the reference clock and adjust the output frequency accordingly. Unlike typical negative feedback circuits that

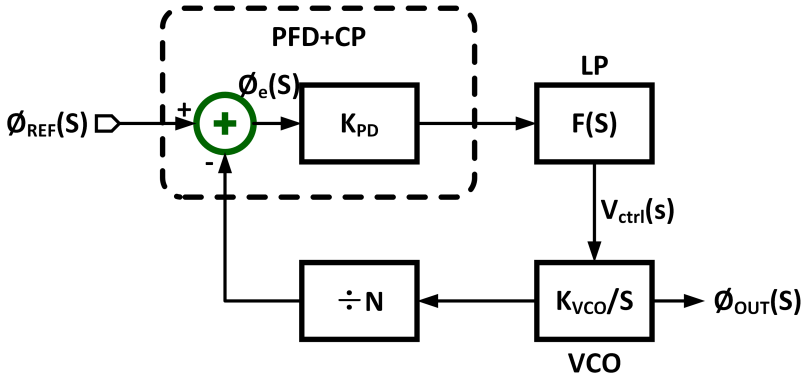

Figure 3.1: Basic PLL Feedback Loop

only deal with voltage or current, PLL circuits focus more on the phase. It is always the phase difference between the reference clock and generated clock that triggers the negative feedback mechanism that stabilizes the loop. On the other hand, the PLL still needs use voltage and/or current signal to change the control voltage of the voltage controlled oscillator or VCO to generate the desired clock signal, so in the world of PLL, instead of dealing with only the voltage and current signal, we also care about the phase signal. As we will see in the later, the feedback information will first in form of phase, then it will be converted to voltage, then to current and at last back to phase. In this point of view, the phase frequency detect can also be seen as a phase to voltage converter and the voltage controlled oscillator actually is a voltage controlled phase generator. The control voltage of the VCO controls the speed of generating phase, or frequency. We will discuss this further in the following section.

3.2.1 Phase Frequency Detector

The phase frequency detector (PFD) is a circuit that linearly translate the phase difference into voltage signals. The ideal average input/output relationship should be:

$$V_e = K_{PD} \times \phi_e \tag{3.1}$$

K_{PFD} is defined as the PFD gain.

Unfortunately, this is clear impractical since according to this formula, output voltage will become arbitrary large if the phase difference is large. In

order to make it possible to build, we should add another condition, which is:

$$\phi_e| < 2\pi \tag{3.2}$$

Besides the linear range, the PFD unit should also be able to detect the frequency difference in order to adjust the control voltage of VCO. This requires the output voltage of PFD is always above (or below) a certain value when the frequency of generated clock is faster than the frequency of reference clock (positive phase error) or below (or above) a certain value when the frequency of generated clock is faster than the frequency of reference clock (negative phase error). The final result of input phase error and output voltage relationship is shown in Fig. 3.2:

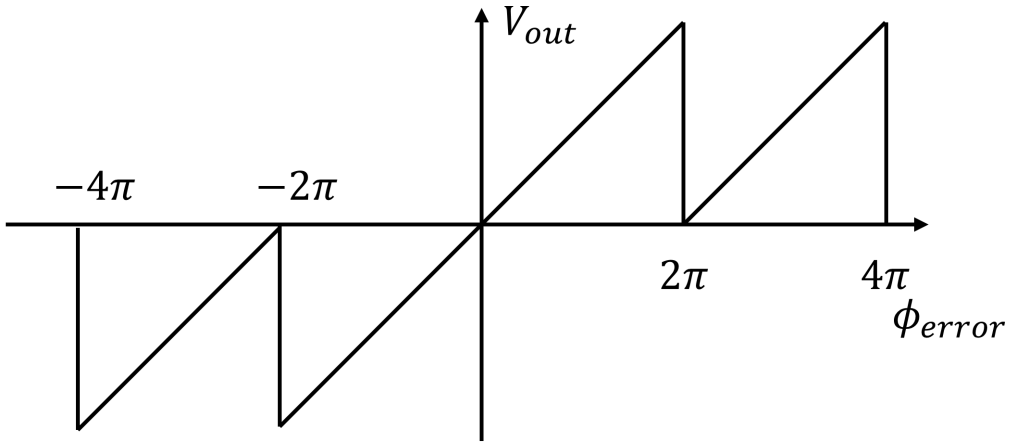

Figure 3.2: Basic PLL Feedback Loop

$$V_e = K_{PD} \times \phi_e \tag{3.3}$$

$$where\ |\phi_e| < 2\pi$$

K_{PFD} is defined as the PFD gain.

3.2.2 Charge Pump

Charge pump is the device that translates the digital voltage signals generated from PFD into a current signal. Even though this transformation doesn't add any new information, it is still an essential part of the loop. Since the voltage controlled oscillator needs a stable voltage to maintain its

oscillating frequency, a charge storage capacitor is needed. The capacitor should be as large as possible since we don't want it voltage decrease too fast before the PFD can charge it again. It requires a significant amount of charge to be dumped into the capacitor in a short time period. But as we will see in the next chapter, PFD is a pure digital circuit which can only generate voltage pulses. Generating large amount of current is not its function. Thus a charge pump is needed here to perform this task. Together with the PFD the s-domain transform becomes the following:

$$K_{PFD} = \frac{i_{Charge\ Pump}}{2\pi} \tag{3.4}$$

3.2.3 Low Pass Filter

Again, as we will see in the next chapter, a PFD is essentially a D flip-flop based simpler, which generates high frequency noise during the switching. A low pass filter is needed here to:

1. Suppress the high frequency noise generated by the PFD.
2. Act as the charge storage device that can maintain a stable voltage by itself.

In addition, a zero should be introduced by its S domain transform function for stability reason, a RC based low pass filter is chose here. Considering the resistor will introduce a voltage ripple as large as $I_{cp}R$ an additional ripple suppression capacitor $C_2$is introduced here. The final design is shown by figure 3.3.

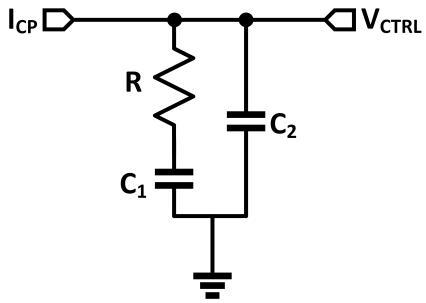

Figure 3.3: Loop Filter

Its s domain transform is following:

$$LF(s) = \frac{V_{ctrl}(s)}{i_{Charge\ Pump}} = \frac{s + \frac{1}{RC_1}}{C_2 s(s + \frac{C_1+C_2}{RC_1C_2})} \tag{3.5}$$

3.2.4 Voltage Controlled Oscillator

VCO is the device that generate the target clock. Ideally, its output frequency should be linearly related to the input control voltage. VCOs Laplace transform function is derived as following:

$$\omega_{out}(t) = K_{VCO} v_{ctrl}(t) \tag{3.6}$$

$$\mathcal{L}[\omega_{out}(t)] = \omega_{out}(s) = K_{VCO} v_{ctrl}(s) \tag{3.7}$$

$$\phi_{out}(t) = \int_0^t \omega_{out}(\tau) d\tau = \int_0^t K_{VCO} v_{ctrl}(\tau) d\tau \tag{3.8}$$

$$\mathcal{L}[\phi_{out}(t)] = \phi_{out}(s) = \frac{\omega_{out}(s)}{s} = \frac{K_{VCO} v_{ctrl}(s)}{s} \tag{3.9}$$

Thus, the Laplace transform function for VCO is:

$$H_{VCO}(s) = \frac{\phi_{out}(s)}{v_{ctrl}(s)} = \frac{K_{VCO}}{s} \tag{3.10}$$

The K_{VCO} is defined as the VCO gain.

3.2.5 Divide by N Divider

For a clock multiplication clock synthesizer, a divide by N block is also needed. Since the PFD is comparing the frequency and phase of the output of the divider and the reference clock, the VCO has to run N times faster than the reference clock in order to make the divider output synchronizes with the reference clock. Here we assume N = 1 for simplicity.

3.2.6 PLL Loop Dynamics

Before getting the loop analysis I want to first clarify some important concept in PLL circuit as they are critical parts of the loop dynamic analysis and stability analysis.

1. The order of a PLL is determined by the number of poles in the loop.

2. The type of a PLL is determined by the number of integrators.

From the Eq. 3.10 we can conclude that he VCO always has a pre-existing pole thus every PLL is at least of order 1 and type 1. As the loop-filter poles increases and the PLL order and type increases as well and higher the type. Higher order ensures faster locking process but severely degrade the stability. In reality most of the PLL designs are third order.

From the previous section we can now define the open loop transfer function as following:

$$LG(s) = K_{PD} \cdot F(s) \cdot \frac{K_{VCO}}{s} \tag{3.11a}$$

$$= K_{PD} \cdot K_{VCO} \cdot \frac{s + \frac{1}{RC_1}}{C_2 s^2 (s + \frac{C_1 + C_2}{RC_1C_2})} \tag{3.11b}$$

From the open loop gain it is clear that

$$\omega_z = \frac{1}{RC_1};\ \omega_{p1} = \omega_{p2} = 0; \omega_{p3} = \frac{C_1 + C_2}{RC_1C_2} \tag{3.12}$$

The phase margin will be:

$$\phi_M = \arctan(\frac{\omega_{ugb}}{\omega_z}) - \arctan(\frac{\omega_{ugb}}{\omega_p 3}) \tag{3.13}$$

where ω_{ugb} is the open loop unity gain bandwidth and $\omega_z < \omega_{ugb}$. In order to achieve maximum phase margin, the value of C_1 and C_2 has to be chosen carefully. To calculate the expression of ϕ_{M_max} we should take the first order derivative of Eq. 3.13 with respect to ω_{ugb} and equate the result to zero, then we can have:

$$\omega_{ugb} = \omega_z \sqrt{\frac{C_1}{C_2} + 1} \tag{3.14}$$

Then,

$$\phi_{M\_max} = \arctan(\sqrt{\frac{C_1}{C_2}+1}) - \arctan(\frac{1}{\sqrt{\frac{C_1}{C_2}+1}}) \tag{3.15}$$

The design procedure of the loop filter is following: First choose desired bandwidth ω_{ugb}, phase margin ϕ_M and resistor R according to specification. Then calculate the K_c from Eq. 3.15:

$$K_c = \frac{C_1}{C_2} = 2(\tan^2(\phi_M) + \tan(\phi_M)\sqrt{\tan^2(\phi_M)+1}) \tag{3.16}$$

Then from Eq. 3.14 we have:

$$\omega_z = \frac{\omega_{ubg}}{\sqrt{\frac{C_1}{C_2}+1}} \tag{3.17}$$

Then,

$$C_1 = \frac{1}{\omega_z R}; C_2 = \frac{C_1}{K_c}; \tag{3.18}$$

Finally, from Eq. 3.4, 3.11b and 3.12 we can can determine the value for I_CP:

$$I_{CP} = \frac{2\pi C_2}{K_{VCO}} \cdot \omega_{ugb}^2 \cdot \sqrt{\frac{\omega_{p3}^2 + \omega_{ugb}^2}{\omega_z^2 + \omega_{ugb}^2}} \tag{3.19}$$

The comparison between the open loop response with optimal phase margin and the response with sub-optimal phase margin can be visualized by Fig.

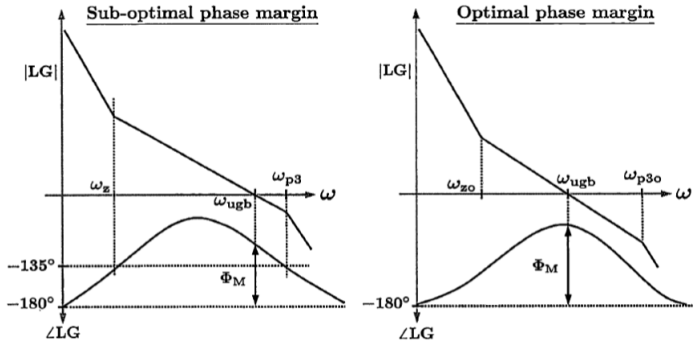

Figure 3.4: Open Loop Phase Margin

Next, we need to analytically confirm that the PLL will indeed lock when there is a frequency step applies at the input. Without loss of generality let's assume there is input frequency step $\omega_{in} = \frac{\Delta\omega}{s}$, then $\Phi_{in}(s) = \frac{\Delta\omega}{s^2}$. First,

obtain the closed loop transfer function:

$$H_{PLL}(s) = \frac{LG(s)}{1 + LG(s)} \tag{3.20}$$

Then define steady state error transfer function:

$$\frac{\Phi_{error}(s)}{\Phi_{in}(s)} = H_e(s) = 1 - H_{PLL}(s) = \frac{1}{1 + LG(s)} \tag{3.21}$$

Now apply final value theorem, the steady state error will be:

$$\begin{aligned}
\Phi_{ss_error}^{F_{step}} &= \lim_{s \to 0} s \cdot H_e(s) \cdot \Phi_{in}(s) && \text{(3.22a)} \\
&= \lim_{s \to 0} s \cdot \frac{1}{1 + LG(s)} \cdot \frac{\Delta\omega}{s^2} && \text{(3.22b)} \\
&= \lim_{s \to 0} \frac{[RC_1C_2s^2 + (C_1 + C_2)s]\Delta\omega}{RC_1C_2s^3 + (C_1 + C_2)s^2 + K_{VCO}K_{PD}s + 1} && \text{(3.22c)} \\
&= \frac{0}{1} && \text{(3.22d)} \\
&= 0 && \text{(3.22e)}
\end{aligned}$$

Eq. 3.22e indicates that the type two third order PLL we have designed can eliminate any steady state phase error and relock when a frequency step is applied at the input. The calculation of the loop filter components also ensures positive phase margin thus the loop is always stable.

3.3 PLL Noise Analysis

We all know that noise is unavoidable in every circuit or system, they are extremely important for PLL circuit since most of the time PLL will be used as the clock generator. Its noise level basically set the minimum noise of the whole system.

Virtually every block in the PLL loop will generate noise, the instead of the loop shown in Fig. 3.1, the model now becomes the following:

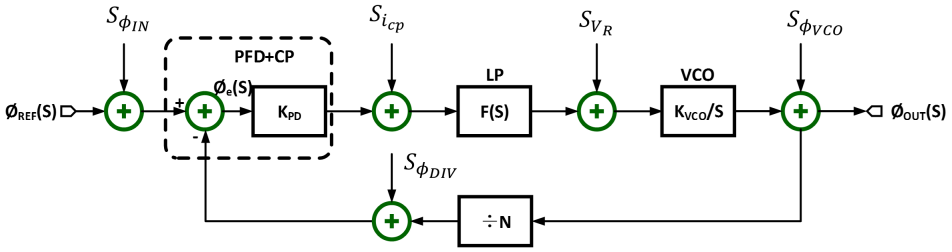

Figure 3.5: Basic PLL Feedback Loop with Noise

Since we only consider the random noise, the correlation between each noise source is zero. Then the output referred noise level and noise transform function (NTF) can easily calculated:

$$S_{\Phi_{OUT}}^{\Phi_{IN}} = S_{\Phi_{IN}}|NTF_{IN}(s)|^2 \tag{3.23a}$$
$$S_{\Phi_{OUT}}^{i_{CP}} = S_{i_{CP}}|NTF_{CP}(s)|^2 \tag{3.23b}$$
$$S_{\Phi_{OUT}}^{v_R} = S_{v_R}|NTF_R(s)|^2 \tag{3.23c}$$
$$S_{\Phi_{OUT}}^{\Phi_{VCO}} = S_{\Phi_{VCO}}|NTF_{VCO}(s)|^2 \tag{3.23d}$$

Where,

$$NTF_{IN}(s) = \frac{\Phi_{OUT}(s)}{\Phi_{IN}(s)} = \frac{N \cdot LG(s)}{1 + LG(s)} \tag{3.24a}$$
$$NTF_{DIV}(s) = NTF_{IN}(s) \tag{3.24b}$$
$$NTF_{CP}(s) = \frac{\Phi_{OUT}(s)}{i_{CP}(s)} = \frac{2\pi}{I_{CP}} \cdot NTF_{IN}(s) \tag{3.24c}$$
$$NTF_R(s) = \frac{\Phi_{OUT}(s)}{v_R(s)} = \frac{\frac{K_{VCO}}{s}}{1 + LG(s)} \tag{3.24d}$$

Fig. 3.6 shows a typical characteristic of each NTF function of a PLL.

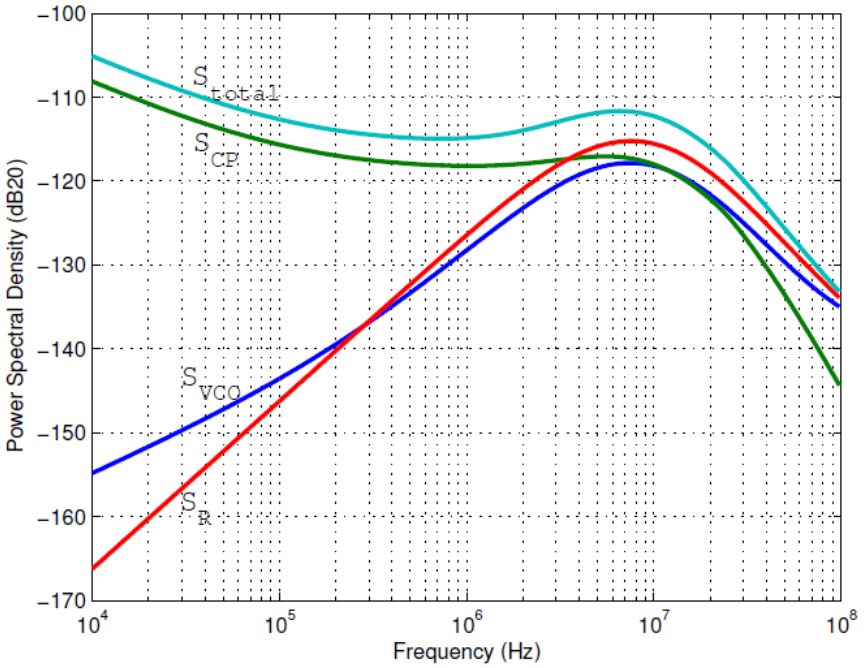

Figure 3.6: Typical PLL Output Referred Noise Simulation

Now we are ready to model and visualize its behavior with simulations.

CHAPTER 4

ANALOG PLLs DESIGN EXAMPLE

4.1 Behavior Modeling of Analog Circuit

From previous chapter we did a thoughtful analysis of each building block of a clock synthesizer PLL. The next step is clearly building one and simulate it to verify the design. Before starting to dig into the detailed implementation the reader should be noticed that PLL is a mixed signal device, in which the digital part, analog part and the interface between digital and analog circuit all play important roles. For the design of the digital part is straight forward: it will work as long as it can operate at target frequency. But for the analog part it is much more complex: the bias point could be wrong, the common mode voltage could be off or the current mismatch is larger than the tolerance, etc. With every aspect tangles together it will impossible for the designer to find out if the there is bugs in the building blocks or the loop parameters are off. What makes it even worse is that the simulation time would be too long that deliver the final design in time will become impossible. The right way to avoid this is to use behavior modeling and its lightning fast simulator for the analog part first and determine the right loop parameters and then design the analog part and interface according to it. As long as the analog components meet the specification the whole circuit will work. In this chapter we will adopt this design methodology and design the digital part of the PLL (Phase Frequency Detector and Divided by Eight frequency Divider) in transistor but design the analog parts (Charge Pump and Voltage Controlled Oscillator) in VerilogA first. Only after closing the loop and getting the loop dynamic verified, the transistor level design of the CP and VCO will be considered.

The Clock Synthesizer that will be designed should take a 200MHz reference clock and output a 1.6GHz clock with minimum jitter. The TSMC

180nm RF will be used as the physical design kit. The reason for choosing 1.6GHz as the target frequency because it is high enough to make the design practice non-trivial yet it is low enough that advanced circuit design techniques like dynamic logic in digital part and low swing signaling in analog part are not necessary.

4.2 Phase Frequency Detector – Transistor Level

4.2.1 Logic Design

The concept of the function of Phase Frequency Detector is explained in previous chapter. The basic design idea is simple: for each reference clock period, a decision about adjusting the output frequency of the Voltage Controlled Oscillator should be made. The comparison of the reference clock frequency and the VCO output clock frequency start when a failing transition is seen. If the transition of the reference clock is seen first, the VCO should run faster and vice versa. The comparison ends at both transitions are seen and the PFD should be reset. If both the transitions happen at the same time, it should hold the control voltage unchanged as now the PLL is locked. It is essentially a state machine with three states:

1. VCO Go Faster
2. VCO Go Slower
3. VCO hold

The state transition diagram is shown in Fig. 4.1.

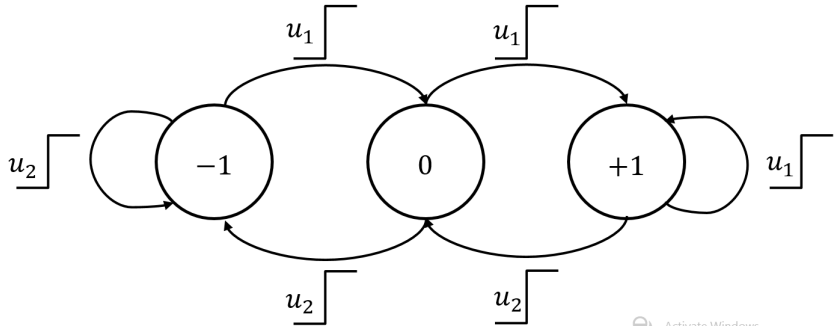

Figure 4.1: PFD State Machine

4.2.2 Transistor Implementation

The circuit implementation of this state machine is shown in Fig. 4.2. It contains two D flip-flops with a reset path. The TSPC D flip-flop should be adopted for the speed reason. In addition, since the input of these flip-flops are all VDD, significant simplification can be made for further speed up and power saving, as shown in Fig. 4.3 [8].

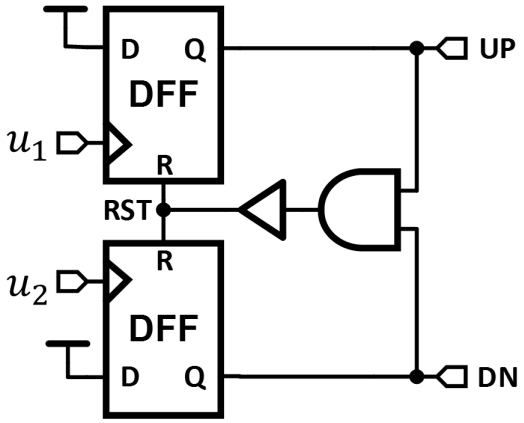

Figure 4.2: PFD Schematic

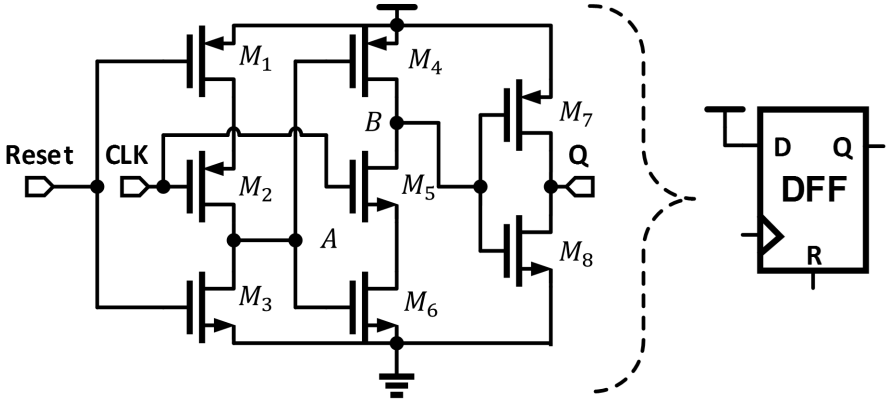

Figure 4.3: D FlipFlop Used in PFD

When operates normally, reset remains LOW. This will turn on PMOS transistor M_1, whenever the clock becomes HIGH, PMOS transistor M_2 will be on, node A will be charged to HIGH, which in turn turn off PMOS transistor M_4 and turn on NMOS transistor M_6. Since CLK is already HIGH, NMOS transistor M_5 is turned on, then node B is discharged to LOW. Since M_7 and M_8 essentially is an inverter, the output will remain high (since we

assume the input is always HIGH). When the Reset becomes high, M_1 will be off and M_3 will be on. No matter what value CLK is, node A is discharged to LOW, which turn off M_6 and turn on M_4. Node B will be charged to HIGH regardless the value of CLK, then the output will be reset to LOW.

Note that the delay of the reset path is added deliberately to prevent dead zone problem. Since the switch of the charge pump needs certain amount of time to turn on or off, very small phase difference of REF and VCO clock cannot be captured. For example, if there is 1 fs time difference between the arrival time of both the edges, one of the UP and DN signal is supposed to be activated for 1 fs. In reality, 1 fs is way too fast for any change of the circuit, then this 1 fs phase error remains untouched. In order to solve this problem, a delay time of ΔT is introduced in the reset path, now one of the signal will be HIGH for ΔT and other will be HIGH for $(\Delta T + 1)fs$, the net change will be 1 fs. The waveform of this operation is shown in Fig. 4.4 and Fig. 4.5. Since turning on both the UP and DN signal will create a direct current path from VDD to GND and burning significant amount of power, the ΔT should be just enough to operate the charge pump switch and no more.

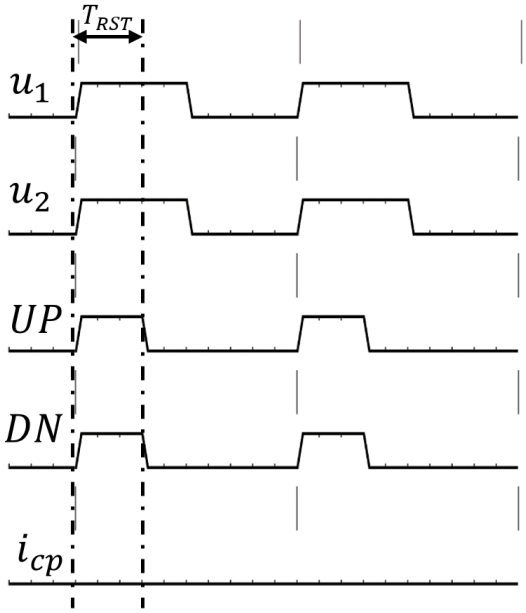

Figure 4.4: PFD with Delayed reset Path Output in Locked State

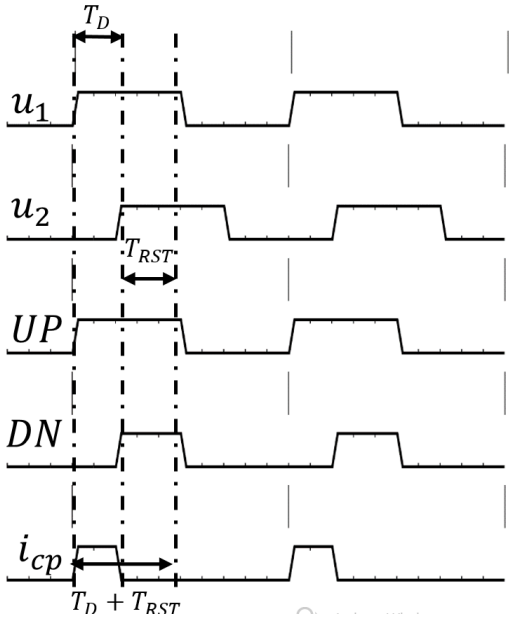

Figure 4.5: PFD with Delayed reset Path Output in Unlocked State

4.2.3 Simulation and Validation

The simulation setup should contain three scenarios:

1. CLK_{VCO} is faster.
2. CLK_{VCO} is slower.
3. CLK_{VCO} and CLK_{VCO} are matched.

The scenario 1 is shown in Fig. 4.6. As we can see, the UP signal is assigned at beginning of the period and get reset when the failing edge of VCO comes.

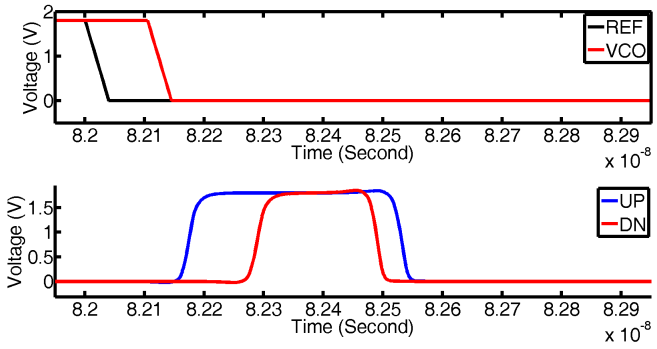

Figure 4.6: PFD Simulation When VCO is Fast

The scenario 2 is shown in Fig. 4.7. As we can see, the DN signal is assigned at beginning of the period and get reset when the failing edge of REF comes.

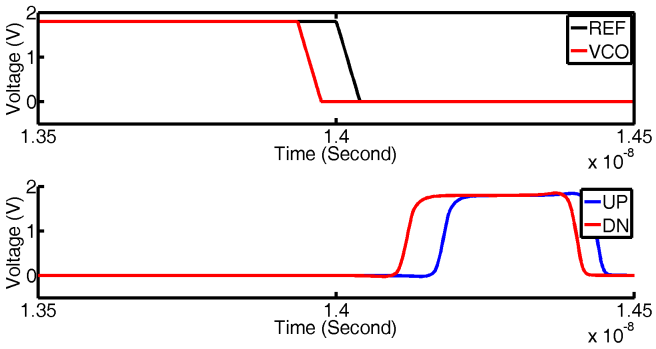

Figure 4.7: PFD Simulation When VCO is Slow

The scenario 3 is shown in Fig. 4.8. As we can see, both the UP and DN signal get assigned and reset.

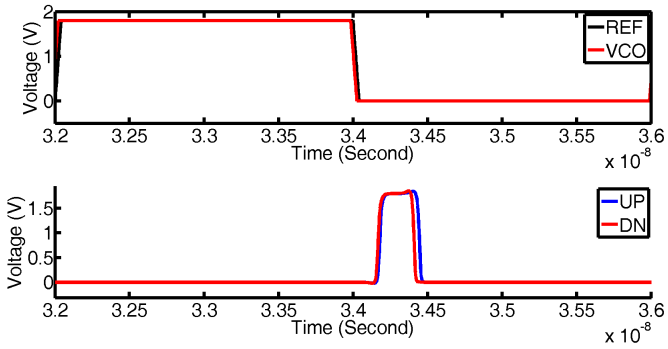

Figure 4.8: PFD Simulation When Locked

These simulation verifies the functionality of the PFD.

4.3 Charge Pump

4.3.1 Logic Design

The conceptual design is shown in Fig. 4.9. The UP and DN signals should come from PFD. When UP signal is set, the upper current source is turned on, charging the charge storage capacitor, increase the voltage voltage. When DN

signal is set, the lower current source is turned on, discharging the charge storage capacitor, decrease the voltage voltage. When both the current sources are off, the charge storage capacitor holds the control voltage. When both the current sources are on, the same among of charge will dump in and sink out of the charge storage capacitor, causing zero net charge change, the charge storage capacitor holds the control voltage.

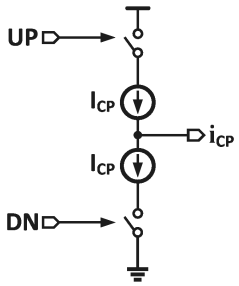

Figure 4.9: Conceptual Charge Pump Design

4.3.2 Behavioral Model

Since we are only trying to verify the loop dynamic, the charge pump will be implemented in VerilogA. The code is following:

Listing 4.1: Charge Pump Behavioral Model

```
‘include "constants.vams"
‘include "disciplines.vams"
module CP(DN, UP, Iout);
input DN, UP;
output Iout;
electrical DN, UP, Iout;
parameter real Icp = 1u;
analog begin
                if (V(DN) < 0.3 && V(UP) > 1.5)
                        I(Iout) <+ Icp;
                else if (V(DN) > 1.5 && V(UP) < 0.3)
                        I(Iout) <+ -Icp;
                else
                        I(Iout) <+ 0;
end
endmodule
```

4.4 Loop Filter

The loop filter will be designed with following parameters:

1. $\omega_{ugb} = 5MHz$ for better stability and less PFD switching noise, assuming it will be the major noise source of the PLL.
2. The filter resistor will be 5K Ω for lower thermal noise.
3. The phase margin will be 75° for faster converge.

The detailed calculation is following: From Equ. 3.16 we have

$$K_c = \frac{C_1}{C_2} = 2(\tan^2(75^\circ) + \tan(75^\circ)\sqrt{\tan^2(75^\circ) + 1}) = 56.6955 \tag{4.1}$$

Then from Equ. 3.17 we have

$$\omega_z = \frac{5 * 2\pi * 10^6}{\sqrt{56.6955 + 1}} = 4.1360 \cdot 10^6 rad/s \tag{4.2}$$

Now, from Equ. 3.18 we have

$$C_1 = \frac{1}{2.068 \cdot 10^6 \cdot 5000} = 48.356pF; \tag{4.3}$$

$$C_2 = \frac{48.356p}{56.6955} = 852.91fF; \tag{4.4}$$

Then from Equ. 3.12 we have

$$\omega_{p3} = \frac{48.356p + 852.91f}{5000 \cdot 48.356p \cdot 852.91f} = 2.3863 \cdot 10^9 rad/s \tag{4.5}$$

Finally from Equ. 3.19 we have

$$I_{CP} = \frac{2\pi \cdot 852.91 \cdot 10^{-15}}{700 \cdot 10^6} \cdot (5 * 2\pi * 10^6)^2 \cdot \tag{4.6a}$$

$$\sqrt{\frac{(2.3863 \cdot 10^9)^2 + (5 * 2\pi * 10^6)^2}{(4.1360 \cdot 10^6)^2 + (5 * 2\pi * 10^6)^2}} \tag{4.6b}$$

$$= 57.392\mu A \tag{4.6c}$$

The final design is shown in Fig. 4.10. The phase margin and impulse response of the loop filter rae verified in MATLAB, the bode plot is shown

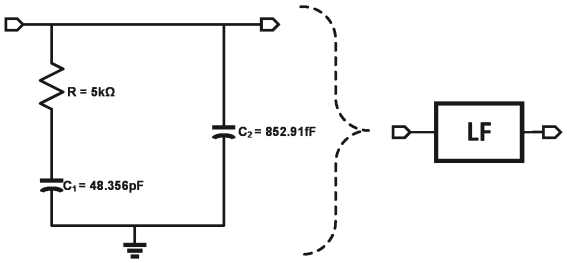

Figure 4.10: Final Filter Design

in Fig. 4.11. This ensures the phase margin is large enough. The impulse response is shown in Fig. 4.12 as we can see the response converges fast enough yet no noticeable overshot is presented.

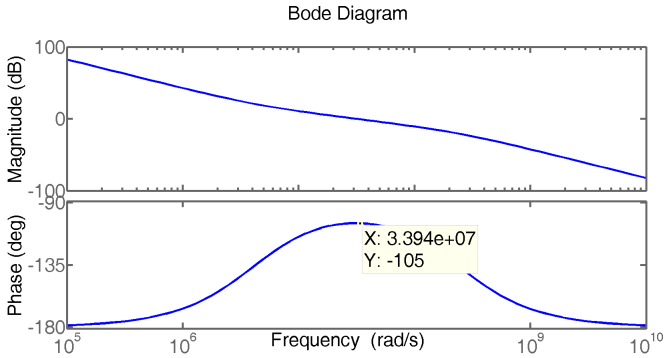

Figure 4.11: Bode Plot of PLL Open Loop

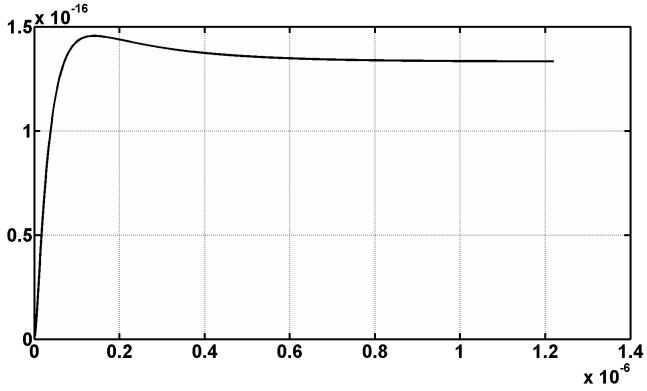

Figure 4.12: Impulse Response of PLL Open Loop

4.5 Voltage Controlled Oscillator

4.5.1 Logic Design

The voltage controlled oscillator should change its output frequency linearly according the controlled voltage. Here I choose the following parameters:

1. Tuning range from 1.0GHz to 2.2GHz.
2. $k_{VCO} = 700MHz/V$
3. Nominal working condition should be 1.6GHz @ 0.9V.

4.5.2 Behavioral Model

The VerilogA code of the VCO goes here. Notice that the random phase noise is added for more accurate simulation.

Listing 4.2: Voltage Controlled Oscillator Behavioral Model

```
‘define PI 3.14159265358979323846264338327950288419716939
‘include "constants.vams"
‘include "disciplines.vams"

module VCO (out, in);
input in;
electrical in;
output out;
electrical out;
parameter real vmin=0;
parameter real vmid=0.9;
parameter real vmax=1.8;
parameter real fmin=970M;
parameter real fmid=1600M;
parameter real fmax=2230M;
parameter real Kvco=700M;
parameter real vl=0;
        // high output voltage
parameter real vh=1.8;
        // output transition time
parameter real tt=0.01/fmax from (0:inf);
```

```
// time tolerance
parameter real ttol=1u/fmax from (0:1/fmax);
real freq, phase;
integer n;
        // VCO FM noise @ fos (single-sideband)
parameter real noise_acc_dbc = -90;
parameter real fos = 1M;
        // VCO PM noise (single-sideband)
parameter real noise_white_dbc = -125;
        // phase noise enable(1), disable(0)
parameter pn_en = 1;
real fm_jitt_std;
real pm_jitt_std;
real fm_del;
real pm_del_2;
real pm_del;
real jitter;
integer file;
integer seed1;
integer seed2;
analog begin
        @(initial_step) begin
                file = $fopen("jitter.txt","w");
                seed1 = -311;
                seed2 = -561;
        end
        freq = (vmid - V(in))*Kvco+fmid;
        if (freq > fmax) freq = fmax;
        if (freq < fmin) freq = fmin;

        $bound_step(0.6/freq);
                //phase is the integral of the freq modulo 2p
        phase = 2*`M_PI*idtmod(freq, 0.0, 1.0, -0.5)+jitter/2;

                // identify the point where switching occurs
    @(cross(phase + `M_PI/2, +1, ttol) or
        cross(phase - `M_PI/2, +1, ttol)) begin
                // Calculate the phase noise
        n = (phase >= -`M_PI/2) && (phase < `M_PI/2);
        fm_jitt_std = fos*sqrt(pow(10,(noise_acc_dbc/10))
                                /(pow(freq,3)));
        pm_jitt_std = sqrt(pow(10,(noise_white_dbc/10))
                                /freq)/2/`PI;
```

```
        pm_del_2 = pm_del;
        pm_del = pm_jitt_std/2*$rdist_normal(seed1,0,1)
                                *freq*2*`PI;
        fm_del = fm_jitt_std/2*$rdist_normal(seed2,0,1)
                                *freq*2*`PI;
        jitter = pm_del - pm_del_2 + fm_del;
    end

    // generate the output
    V(out) <+ transition(n ? vh : vl, 0, tt);
        @(cross(V(out)-vmax/2, +1))
                $fwrite(file,"%f\n",$realtime*1E9);
end
endmodule
```

4.6 Frequency Divider

4.6.1 Logic Design

For the divide-by-eight frequency, a three stage cascaded D Flip Flop is used for its simplicity. The schematic is shown in Fig. 4.13.

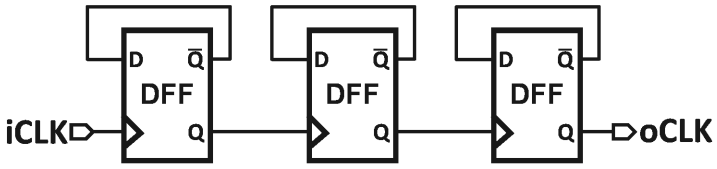

Figure 4.13: Frequency Divider Design

4.6.2 Transistor Implementation

For the D Filp Flop the transitional TSPC D Flip Flop is used. The schematic is shown in Fig. 4.14.

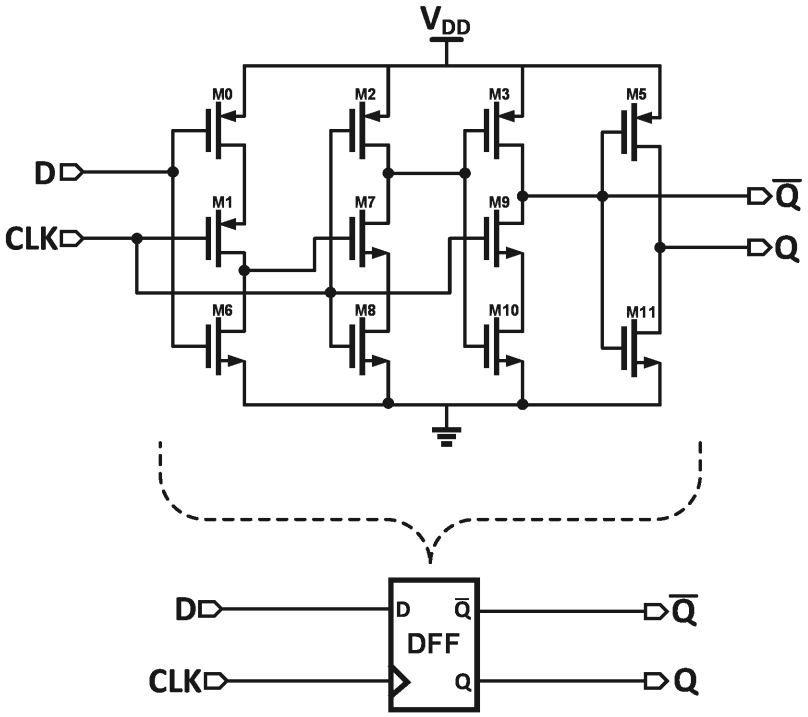

Figure 4.14: D FlipFlop Used in Frequency Divider

4.6.3 Simulation and Validation

In the simulation a 2GHz input clock is fed into the divider, the output waveform is shown in Fig. 4.15. As we can see, the frequency of the output waveform is indeed 250MHz, which indicates the divider is working. Please notice that there is a delay between the input and output waveform, since the PFD will only compare the output of the divider and reference clock, the delay between VCO output and divdier output is unavoidable, thus the output of the PLL will have a constant phase difference compared with the reference clock. But since we are building a clock synthesizer, as long as the frequency is locked, constant phase difference is not important.

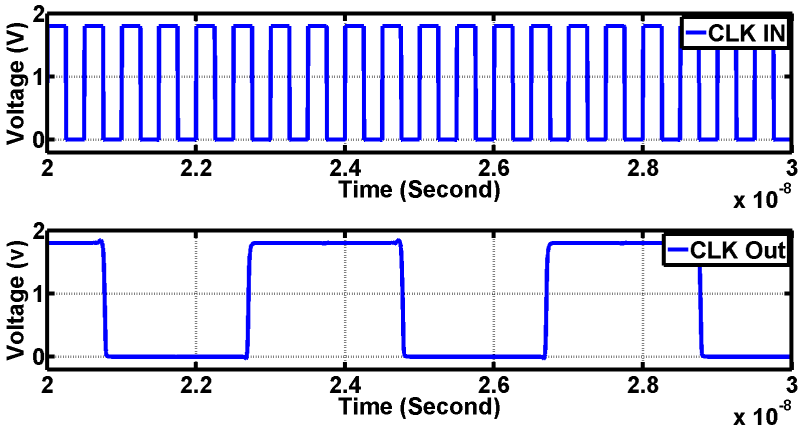

Figure 4.15: Frequency Divider Simulation Result

4.7 Behavioral Level PLL validation

4.7.1 Closed Loop

With all the components we need now the PLL feedback loop is closed. The time domain transient simulation will be ran to validate the functionality and performance of the PLL. In the simulation, -90 dBc/Hz @ 1MHz offset VCO phase noise is added for accuracy.

4.7.2 Control Voltage Waveform

In Fig. 4.16 shows the control voltage waves. As we can see, the voltage settles fast at the beginning of the simulation which indicates the PLL locked. Then at 3 ns point a frequency step is added to the reference clock to force the PLL lose lock, the control voltage drops dramatically and then resettled at a new voltage. This indicates the PLL is indeed functional.

In Fig. 4.17 shows the zoomed in version of the control voltage at steady state. As we can see the ripple is as small 1mV, this indicates the small phase noise of the PLL output.

In Fig. 4.18 shows the zoomed in version of the control voltage at the frequency step. As we can see the cycle slippery was happened, this may due to the extremely small bandwidth we have chosen.

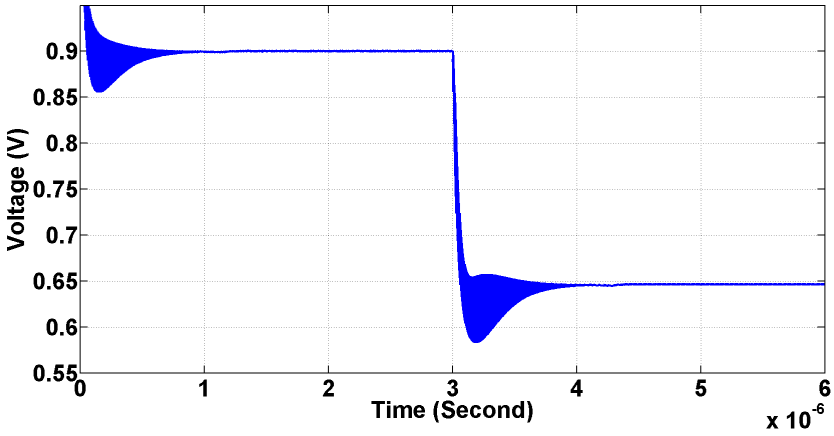

Figure 4.16: Control Voltage Waveform Full Range of Behavioral Model PLL

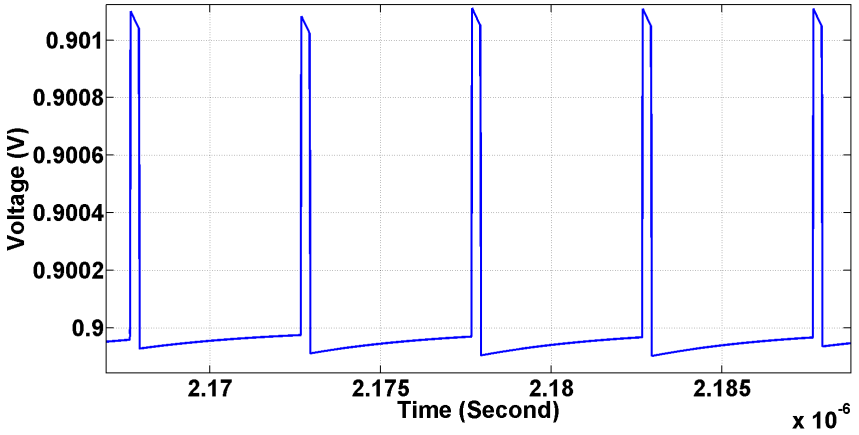

Figure 4.17: Control Voltage Ripple of Behavioral Model PLL

4.7.3 Transient Output

In Fig. 4.19 shows the time domain output of the VCO clock. As we can see, the output is indeed 1.6GHz

From the previous simulation we conclude that the PLL is working. Now we should design the transistor level implementation of the charge pump and the voltage controlled oscillator.

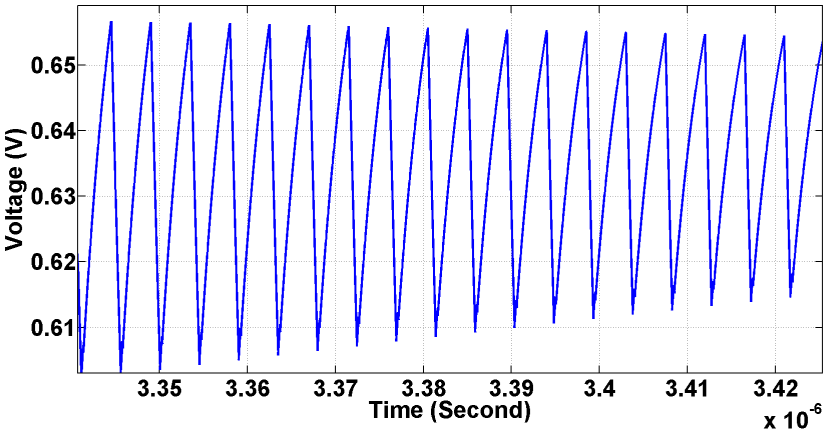

Figure 4.18: Cycle Slippery of Behavioral Model PLL

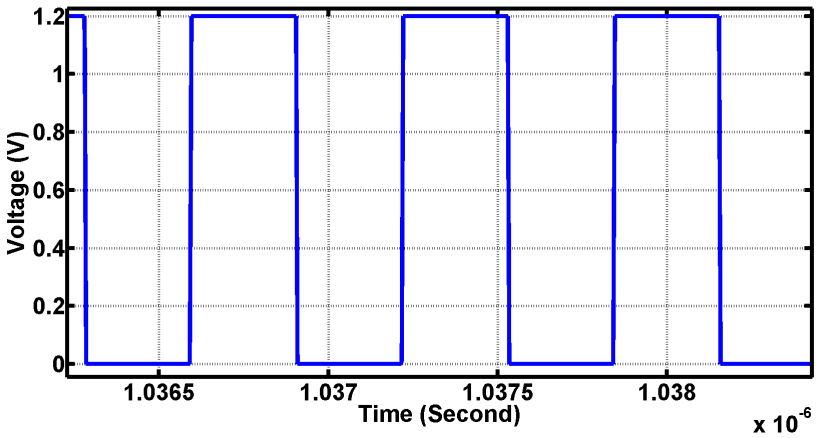

Figure 4.19: Time Domain Output of Behavioral Model PLL

4.8 Charge Pump – Transistor Level

4.8.1 Circuit Implementation

The design concept of a charge pump is explained in previous section. The bootstrapped current-switched charge pump is used here as shown in Fig. 4.20 for the following reasons:

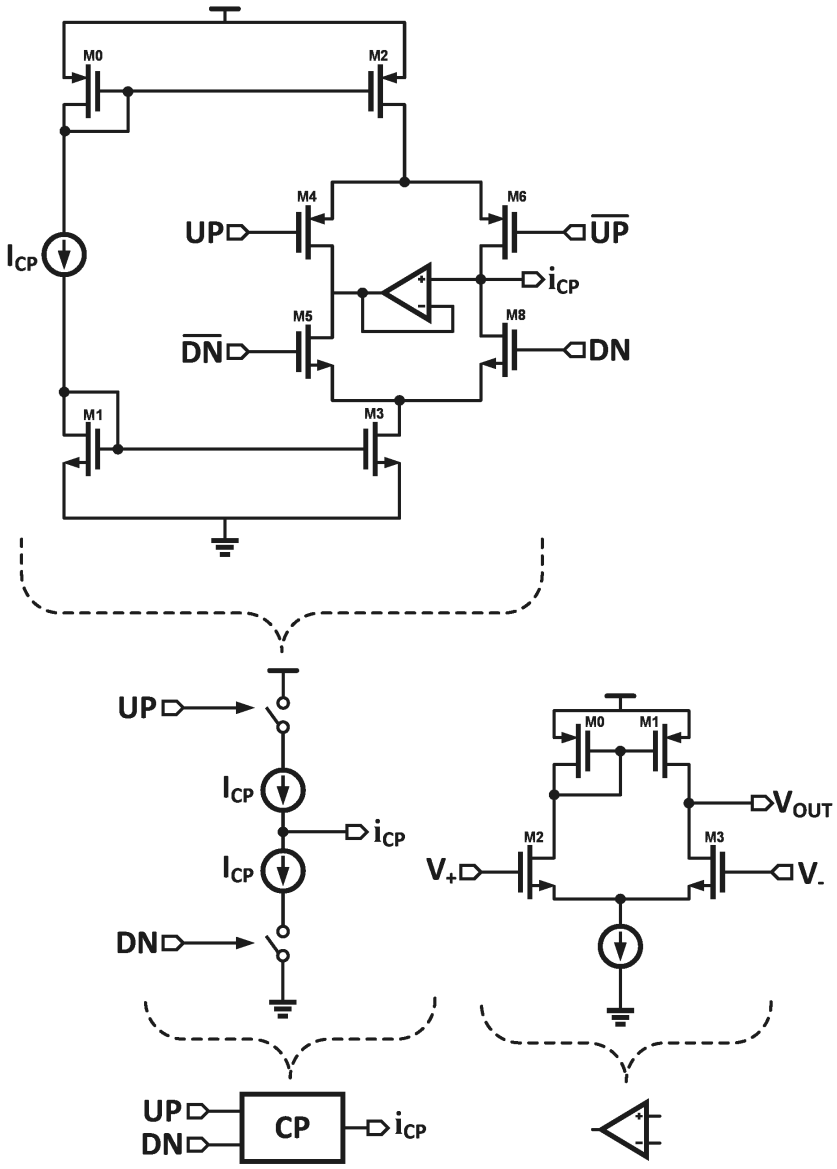

Figure 4.20: Transistor Level Design of Charge Pump

1. Transistor $M0$, $M1$, $M2$ and $M3$ form a current mirror, which ensures that they will always stay in saturation condition to achieve fast current steering.

2. Differential UP and DN signals are used, which greatly reduces the switching time mismatch between UP and DN signals thus reduce the current mismatch.

3. A voltage follower is used to enforce $M4$, $M6$ and $M5$, $M8$ will have exact the same V_{DS} thus further reduce the current mismatch.

4.8.2 Simulation and Validation

The simulation for the charge pump is a little different from previous simulation, since we are dealing with the current output rather than the voltage output and it should be simulated with PFD together. We need to verify that at lock point the charge pump will indeed output enough current. Considering the VDD for TSMC 180nm technology is 1.8V, we should design the PLL that locks at 0.9V. In the simulation we should force output voltage to 0.9V and measure the output current, the setup is shown in Fig. 4.21. As we can see from Fig. 4.22 when VCO is slower, net current is taken output the charge storage capacitor to make the VCO run fast. When it locks, no net current is taken output, as shown in Fig. 4.23. When VCO is faster, net current is dumped into the charge storage capacitor to make VCO run faster as shown in Fig. 4.24. This indicates the transistor level charge pump is working.

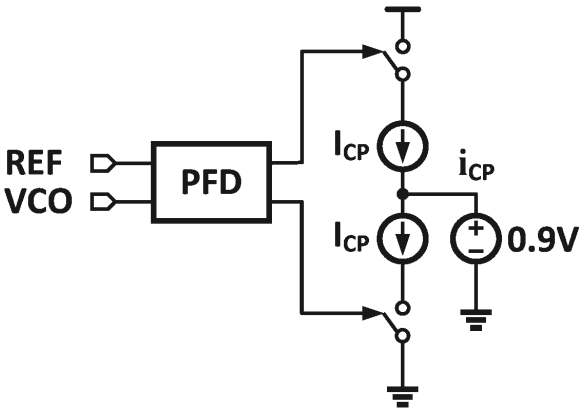

Figure 4.21: Charge Pump Simulation Setup

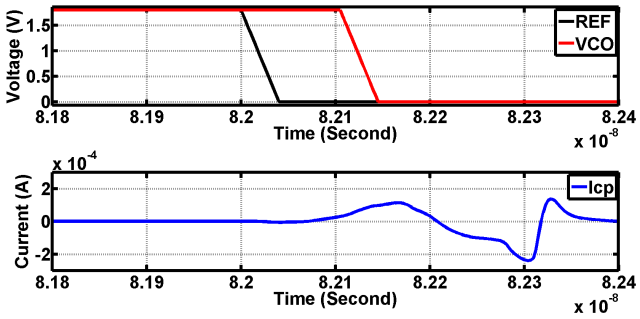

Figure 4.22: Charge Pump Simulation when VCO is Slower

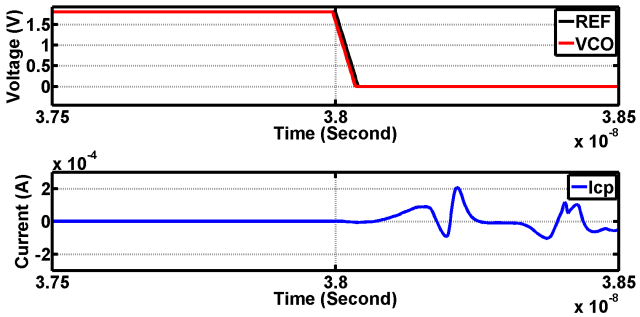

Figure 4.23: Charge Pump Simulation when Locked

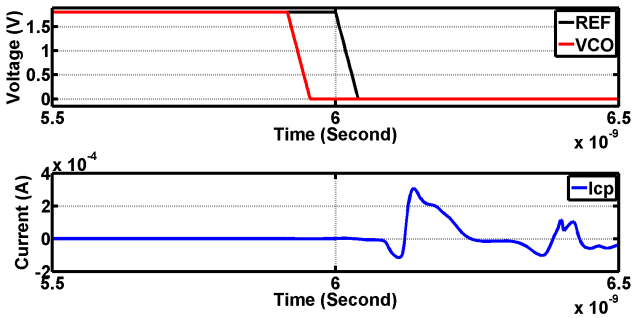

Figure 4.24: Charge Pump Simulation when VCO is Faster

4.9 Voltage Controlled Oscillator – Transistor Level

4.9.1 Circuit Implementation

For the VCO, current starved ring oscillator is used for its robustness, easiness to implement, low power consumption and small footprint. The detailed implementation is shown in Fig. 4.25. For this design, three inverter stages are used to achieve enough loop gain. Assume each stage has delay ΔT, then the period of oscillation is $6\Delta T$. M_9 is used to control the total current that can flow into the inverter stages. Since the stage delay ΔT is related to the maximum current that can flow into that stage, by controlling the current supply we can control the oscillation frequency. M_8 is an always on transistor so that even M_9 is completely off there will be some current flow into the inverter stages to make sure minimum speed oscillation is still active. Since the current is very sensitive to the gate voltage, by adding an always on transistor can also achieve lower K_{VCO} so that the loop will be more stable. A buffer stage is added at the end of the ring to isolate the VCO and its load

capacitor. In this way the oscillation frequency is independent of the VCO loads.

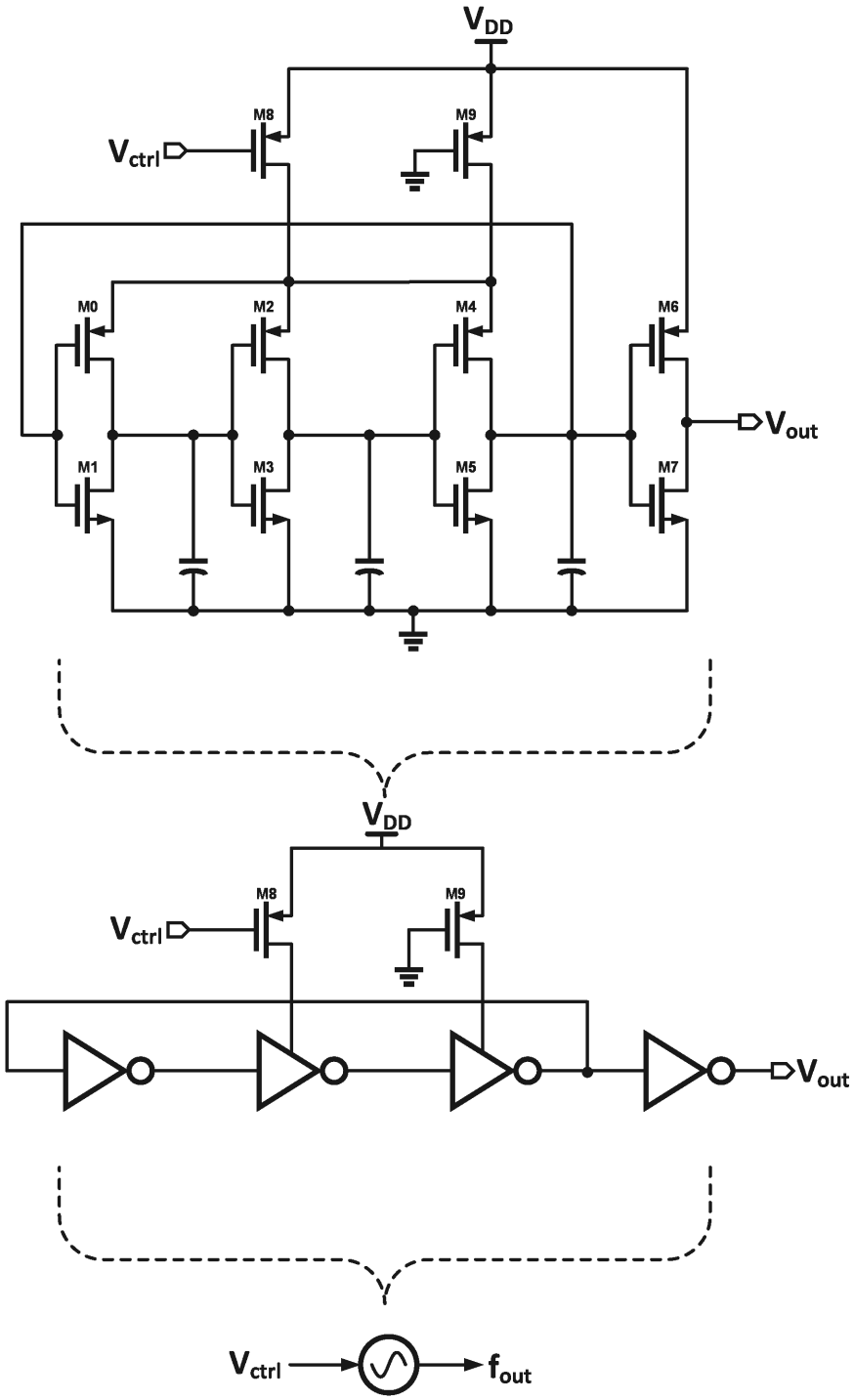

Figure 4.25: Transistor Level Design of VCO

4.9.2 Simulation and Validation

The simulation plan of the VCO is pretty straight forward, a parameter sweep of the control voltage will generate all the information will needed. Fig. 4.26 shows the output frequency V.S. control voltage. Fig. 4.27 shows the K_{VCO} V.S. the control voltage. Since the lock point is about 0.9V, we choose -700MHz/V as the K_{VCO} we have.

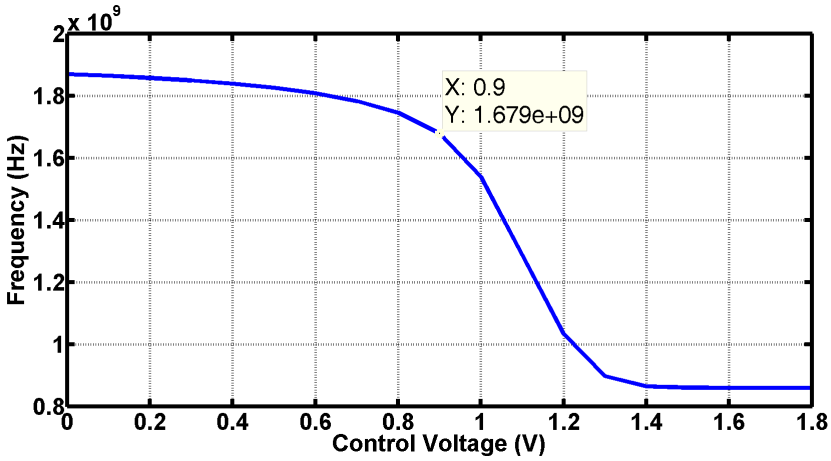

Figure 4.26: VCO Tuning Range

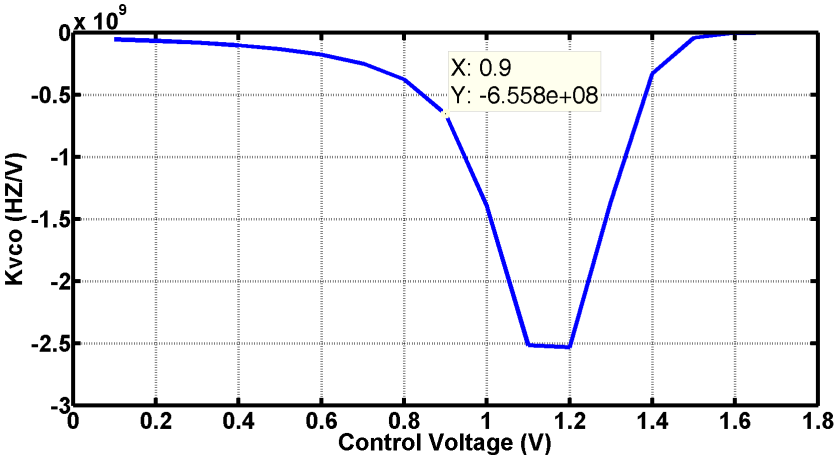

Figure 4.27: Kvco of VCO

4.10 Transistor Level PLL validation

The complete transister level PLL is shown in Fig. 4.28. The simulation setup is almost identical to the behavioral model PLL in the previous section. Both the time domain and frequency domain simulation will be ran to validate the functionality and the performance of the PLL. As shown in the figures, the transistor level PLL has the same characteristics as the behavioral model PLL, with slightly larger control voltage ripple and smoother rising and failing edges at the output. Also the phase noise is measured as -90 dBc/Hz @ 1MHz offset.

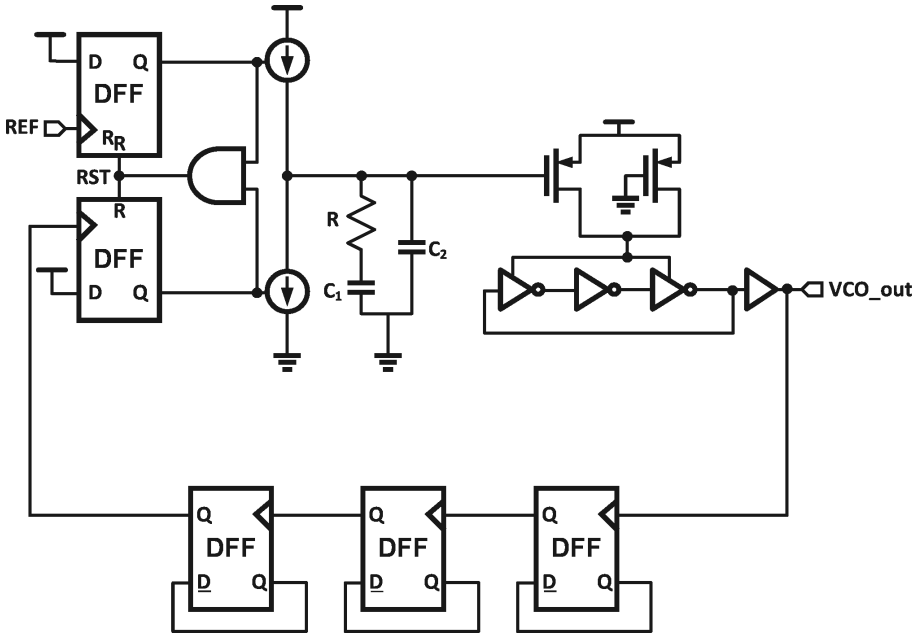

Figure 4.28: Closed Loop PLL

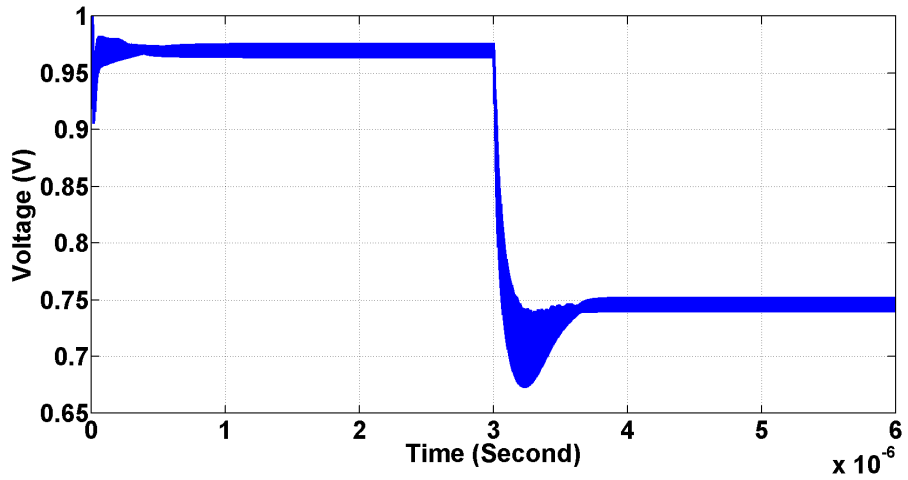

Figure 4.29: Control Voltage Waveform Full Range of Transistor Level PLL

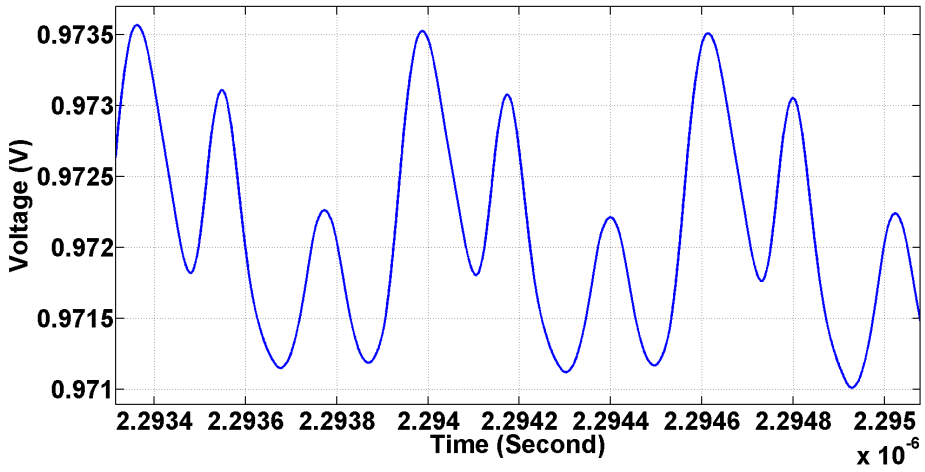

Figure 4.30: Control Voltage Ripple of Transistor Level PLL

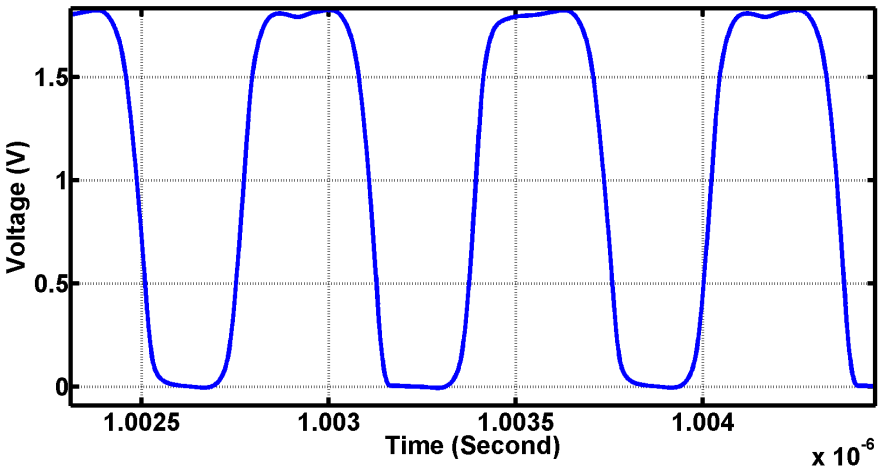

Figure 4.31: Time Domain Output of Transistor Level PLL

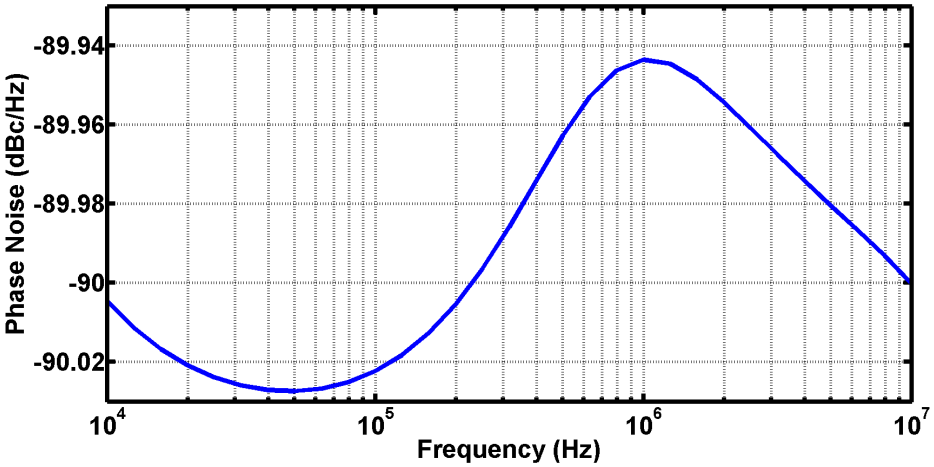

Figure 4.32: Phase Noise of Transistor Level PLL

CHAPTER 5

ALL DIGITAL PLL ANALYSIS

Although the analog PLL that discussed in the previous chapter have good performance in terms of frequency stability, the weakness is also obvious: the mixed signal nature makes it impossible to scale with the semiconductor technology. In fact the higher process-voltage-temperature (PVT) variations and lower gain offered in the state-of-the-art technology actually dramatically reduce the performance of the analog PLL. In addition, the analog PLL requires a charge pump to achieve acceptable lock-in range which constantly draw a significant amount of current to generate appropriate bias point, which in turn burns a lot of energy. A capacitor that used in the analog filter is also un-shrinkable with technology advance. All these reasons make the analog PLL consumes more and more power and silicon area in order to match the performance of the digital cores they are serving for. In order to take advantage of the semiconductor technology advances and to get rid of the power hungry charge pump and area consuming capacitor to reduce power, eliminate the noise-susceptible analog control for the VCO and inherent noise immunity of digital circuit, all digital PLL is the future of the PLL design.

5.1 Time to Digital Converter

The first component that need to be added to the loop is the Time to Digital Converter (TDC). Since the PFD can only produce analog UP and DN signals that related to the phase error, a TDC is needed to translate this time information into digital control bits for the digital filter and digital controlled oscillator. A conventional PFD together with a TDC will form a Phase to Digital Converter (P2D) which translate the phase error to digital values. A typical P2D design is shwon in Fig. 5.1 and a typical TDC design is shown in Fig. 5.2.

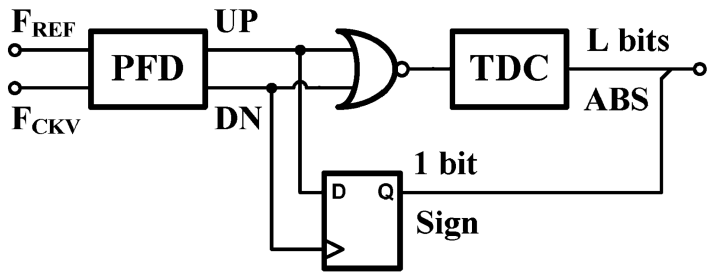

Figure 5.1: Typical P2D Design

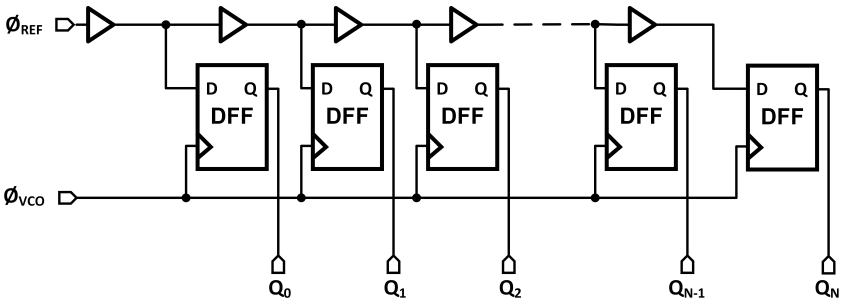

Figure 5.2: Typical P2D Design

In this design the UP and DN signals are overlapped by a XOR gate, which will create a pulse whose width is proportinal to the absolute value of the phase error. A TDC will translate the pulse width to digital word. A sign bit is generated by a sampling D FlipFlop to tell the TDC whether the VCO frequency is slower or faster than the reference clock.

The TDC design introduced a very important specification of the ADPLL: P2D phase resolution, which defined by

$$\Delta\Phi_{P2D} = \frac{2\pi\Delta_{TDC}}{T_{REF}} \tag{5.1}$$

In this equation Δ_{TDC} represents the time resolution of the TDC unit. The $\Delta\Phi_{P2D}$ defines the minimum phase error the ADPLL can handle. If the phase error is smaller than this value, the P2D becomes a BangBang Phase Detector, then the linear analysis becomes invalid. On the other hand, if the phase error is larger than the $\Delta\Phi_{P2D}$ then the linear approximation of the ADPLL is still valid, the TDC block can be modeled as a gain block in S domain. The ADPLL loop becomes the following:

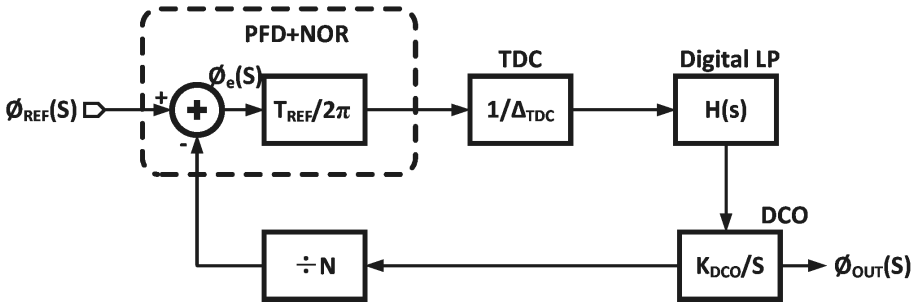

Figure 5.3: S Domain Approximation of ADPLL

5.2 Digital Loop Filter

The most important component that needs to be digitized is the loop filter. By replacing it with digital filter we can get rid of the capacitor and charge pump all together. The easiest way to digitize it is to take what we have in the analog loop filter and use bilinear transform to translate the S domain analysis we have in the chapter 2 to the Z domain and implement it in digital circuit. Of course in order to do it the P2D phase resolution should be small enough so that at the target frequency the linear approximation of the digital loop filter is still valid. Note that the second order low pass filter used in the analog PLL is because we need the second capacitor C_2 to suppress the control voltage ripple, since we don't have control voltage anymore, a typical first order RC low pass filter is sufficient. With a closer examination of its transform function from Eq. 3.5 we can say it contains only one proportional path (R) and an integration path ($\frac{1}{sC}$) (set $C_2 = 0$). This can be transferred into Z domain by using bilinear transformation [9]:

$$s = \frac{2}{T_s} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \tag{5.2}$$

The T_s usually is the period of sampling clock, in our case it is the period of the reference clock. The disadvantage of bilinear transform is frequency wrapping which degrades the frequency response near the Nyquist rate. Since the bandwidth of PLL is usually ten times smaller than the reference clock, frequency wrapping can be negligible here. The actual implementation is also straight forward: the proportional path can be implemented by a multiplication unit and the integral path can be implemented by an accumulator.

The transform can be visualized by Fig. 5.4.

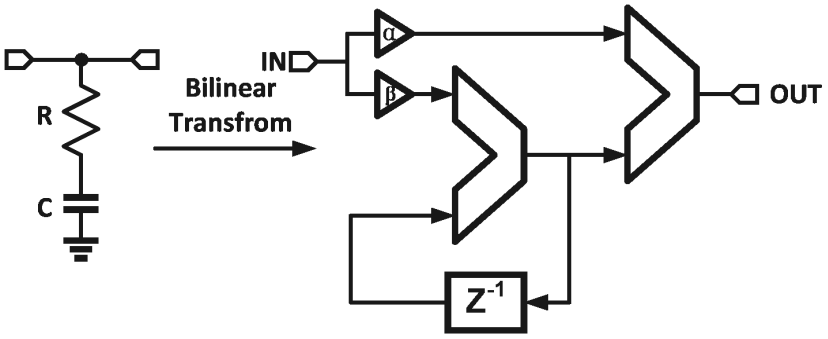

Figure 5.4: Bilinear Transform for Low Pass Filter

5.3 Digital Controlled Oscillator

The design of Digital Controlled Oscillator (DCO) can also be derived from the design of VCO. The key here is to determine how does the digital control word tune the frequency. In case of a ring oscillator based VCO, the digital control word can used to turn on or off bias current sources, in case of LC tank based VCO, the digital control word should be used to turn on or off tank capacitors. The only difference between the DCO and VCO is that in DCO design a large array of small current sources (in ring oscillator case) or a large array of small tank capacitors (in LC tank case) will present in stead of a single tunable current source or a single varactor. The S domain transform of the DCO is the same as the S domain transform of the VCO:

$$DCO(s) = \frac{K_{DCO}}{s} \tag{5.3}$$

5.4 Frequency Divider

Since the Frequency Divider in the analog PLL is already fully digital circuit, no more change needed here.

5.5 ADPLL Loop Dynamic Analysis

5.5.1 Design Parameters

From the previous analysis we can see that the only parameters that need to be determined are the α and β in the digital loop filter. The Fig. 5.4 indicates the Z domain transfer function of the digital loop filter is

$$H(z) = \alpha + \beta \frac{1}{1 - z^{-1}} = \frac{(\alpha + \beta - \alpha z^{-1})}{1 - z^{-1}} \tag{5.4}$$

On the other hand, the analog filter s domain transform and its bilinear transform are given by:

$$Z(s) = \frac{V(s)}{I(s)} = R + \frac{1}{sC} \tag{5.5}$$

$$H(z) = \frac{(\frac{T_s}{2C}) + R + z^{-1}(\frac{T_s}{2C} - R)}{1 - z^{-1}} \tag{5.6}$$

By comparing Eq. 5.6 and Eq. 5.4 we can conclude that

$$\alpha = R - \frac{T_s}{2C} \tag{5.7}$$

$$\beta = \frac{T_s}{C} \tag{5.8}$$

5.5.2 More discuss about α and β

The phase margin is a very important specification of a PLL circuit since it defines the stability of the PLL. The frequency of reference clock, bandwidth of the PLL and α-to-β ratio together determine the phase margin of a ADPLL. The derivation is following: The zero frequency is given by:

$$\omega_z = \frac{1}{RC} \tag{5.9}$$

The phase margin is given by

$$PM = \arctan(\frac{\omega_{ugb}}{\omega_z}) \tag{5.10}$$

Then we can have

$$\omega_z = \frac{\omega_{ugb}}{\tan(PM)} \tag{5.11}$$

From Eq. 5.7 and Eq. 5.8

$$\frac{\alpha}{\beta} = \frac{RC}{T_s} - \frac{1}{2} \tag{5.12a}$$

$$= \frac{1}{T_s \omega_z} - \frac{1}{2} \tag{5.12b}$$

$$= \frac{1}{T_s} \frac{\tan(PM)}{\omega_{ugb}} - \frac{1}{2} \tag{5.12c}$$

$$= \frac{F_{REF}}{F_{ugb}} \frac{\tan(PM)}{2\pi} - \frac{1}{2} \tag{5.12d}$$

CHAPTER 6

ALL DIGITAL PLL DESIGN EXAMPLE

An ADPLL is designed in this chapter as an example about how to design and simulate such a device. The ADPLL showed here contains a transistor level P2D, a synthesis-able Verilog level digital filter, a behavioral model DCO with phase noise enabled and a transistor level frequency divider. The reference clock is 200MHz with the output frequency of the ADPLL is 1.6GHz and the phase margin should be 75° with bandwidth 5MHz.

6.1 Phase to Digital Converter

The PFD part of the P2D is just the conventional PFD showed in Fig. 4.2. The TDC is a little bit different from what we have discussed in the last chapter. Since the goal here is to build a frequency synthesizer, as long as it can lock the frequency, a constant phase difference between reference clock and DCO output won't cause any problem. Therefore a single bit TDC that can tell whether the DCO should go faster or slower is sufficient. By designing in this way a power consuming high accuracy TDC is avoided. The design is shown in Fig. 6.1. As we can see the TDC is a two cascaded RS latches. The first stage is the decision circuit, which outputs 01 or 10 depending the UP and DN signal from the PFD. The second stage is the storage stage, which holds the value of the decision stage until next period starts. This configuration is also called bangband PFD.

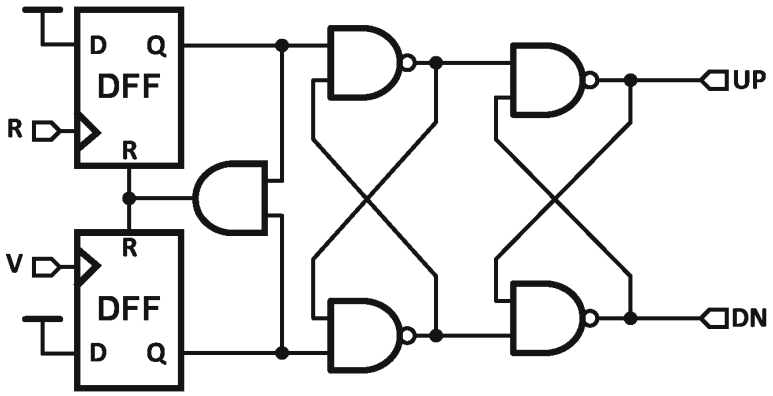

Figure 6.1: Transister Level P2D Design

6.2 Digital Low Pass Filter

From Eq. 5.12d in order to achieve 75° phase margin, the value of $\frac{\alpha}{\beta}$ should be around 58.9, 64 is chosen for easiness of implementation. When comes to the exact value of α, unfortunately there is no straight forward method to calculate it since bangbang PFD is highly nonlinear thus all the S domain analysis is invalid. The only way to determine the value is to make parameter sweep and choose the value that gives the lowest phase noise. After performing parameter sweep $\alpha = 1536$ and $\beta = 24$ are chosen. The complete code is included:

Listing 6.1: Digital Filter Verolog Code

```
//Verilog-AMS HDL for digital filter

`include "constants.vams"
`include "disciplines.vams"

module digf (up, dn, clk, ctrl_code);
        input up, dn;
        input clk;
        electrical clk;
        parameter vdd = 1.8;
        parameter integer phug = 1536;
        integer frug = phug / 64;
        output [15:0] ctrl_code;
        reg [15:0] ctrl_reg;
        assign ctrl_code = ctrl_reg[15:0];
```

```
        reg [15:0]      cnt;
        initial begin
                cnt <=16'b1000_0000_0000_0000;
        end
        always @(cross(V(clk)-0.5*vdd, 1)) begin
                if (up == 1'b1 && dn == 1'b0) begin
                        cnt = cnt - frug;
                        ctrl_reg = cnt - phug;
                end
                else if (up == 1'b0 && dn == 1'b1) begin
                        cnt = cnt + frug;
                        ctrl_reg = cnt + phug;
                end
                else    ctrl_reg = cnt;
        end

endmodule
```

6.3 Digital Controlled Oscillator

The behavioral model of the DCO is very similar to code of the VCO, the only difference is that now the DCO takes digital control word to control its frequency. The complete code is following:

Listing 6.2: Digital Controlled Oscillator Verolog Code

```
//Verilog-AMS HDL for DCO
`define PI 3.14159265358979323846264338327950 2
`include "constants.vams"
`include "disciplines.vams"
`timescale 1ns/1fs
module dco (dco_out, ctrl_code);
        input [15:0] ctrl_code;
                // DCO FM noise @ fos (single-sideband)
        parameter real noise_acc_dbc = -100;
        parameter real fos = 1M;
                // DCO PM noise (single-sideband)
        parameter real noise_white_dbc = -125;
                // DCO output clk
    output dco_out;
        reg dco_out;
```

```
reg dco_out_ideal;
reg dco_out_jitt;
            // DCO output frequency
parameter real freq = 1600M;
            // DCO gain (/LSB)
parameter real Kdco = 5e-6;
    real nom_delay;
real ideal_delay;
real jitt_delay;
    real fm_jitt_std;
    real pm_jitt_std;
    real fm_del;
real pm_del_2;
real pm_del;
integer seed1 = -311;
integer seed2 = -561;
integer file;
    integer i;
    integer count;
initial begin
    dco_out_ideal = 1'b0;
    dco_out_jitt = 1'b0;
            nom_delay = 1/freq*1e9/2;
    ideal_delay = nom_delay;
    jitt_delay = nom_delay;
                    // FM jitter (period jitter std)
            fm_jitt_std     = fos*sqrt(10**(noise_acc_dbc
                    /10)/(freq**3))*1e9;
                    // PM jitter (period jitter std)
            pm_jitt_std = sqrt(10**(noise_white_dbc
                    /10)/freq)/2/`PI*1e9;
    fm_del = 0;
    pm_del = 0;
    pm_del_2 = 0;
end
always @(ctrl_code) begin
            i = ctrl_code - 2**15;
                    // DCO period proportional
                    //  to ctrl_code (freq ~ 1/code)
    ideal_delay = nom_delay * (1 + i * Kdco);
end
            // accumulating jitter modeling
always @(negedge dco_out_jitt) begin
```

```
        pm_del_2 = pm_del;
                        // divided by 2 for half period
        pm_del = pm_jitt_std/2*$dist_normal(seed1,0,1e6)*1e-6;
        fm_del = fm_jitt_std/2*$dist_normal(seed2,0,1e6)*1e-6;
        jitt_delay = ideal_delay + pm_del - pm_del_2 + fm_del;
    end
    always #(ideal_delay)   dco_out_ideal   <= ~dco_out_ideal;
    always #(jitt_delay)    dco_out_jitt    <= ~dco_out_jitt;
    always @(pn_en,dco_out_ideal, dco_out_jitt) begin
        dco_out <= dco_out_jitt;
    end
endmodule
```

6.4 Frequency Divider

The frequency divider is exactly the same as the circuit in analog PLL, shown in Fig. 4.13.

6.5 Simulation and Results

6.5.1 Control Code

The waveform of the control code is shown in Fig. 6.2. The values are converted to decimal. As we can see, the result is the same as the analog case. The control code fast settled to a fixed value at the beginning of the simulation, indicates the ADPLL acquired lock. At 3 us point a frequency step is applied to the reference clock to force PLL lose lock. And then the control code quickly re-settled to a new constant, indicating the ADPLL regain lock. Fig. 6.3 shows a zoomed-in version of the control code waveform. As we can see the code oscillating around the locking point, this is due to the nature of the bangbang PFD and is the major source of the phase noise.

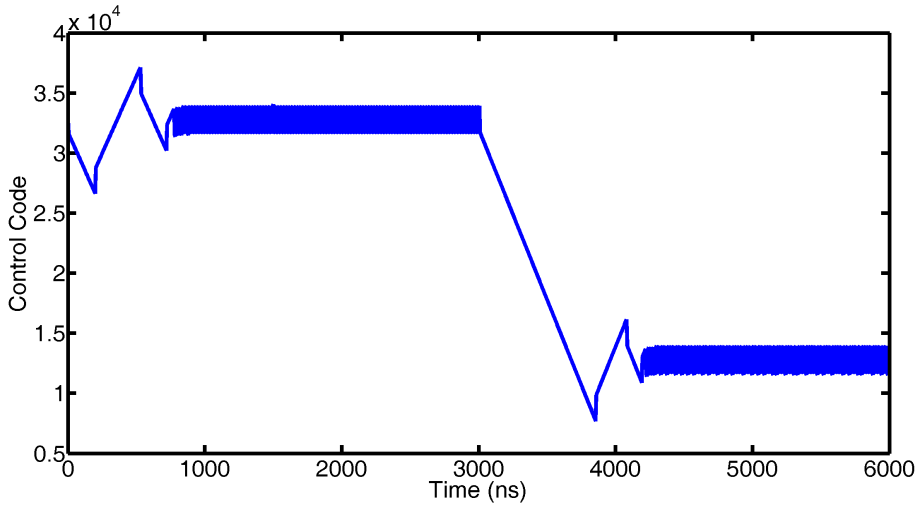

Figure 6.2: Control Code of ADPLL

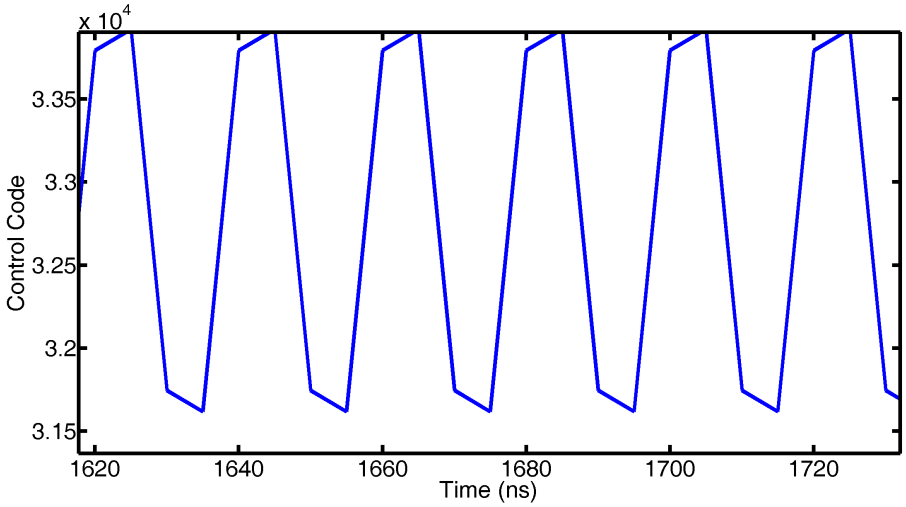

Figure 6.3: Control Code of ADPLL – Zoomed In

6.5.2 Time Domain Output

Fig. 6.4 shows the time domain output of the DCO (after divider) and the reference clock. As we can see the two signals are exactly the same, which proves that ADPLL is indeed locked.

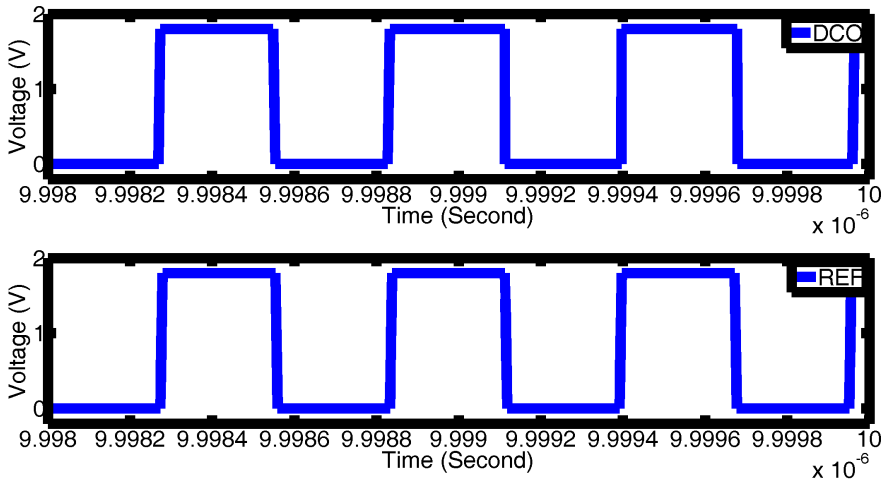

Figure 6.4: Control Code of ADPLL

6.5.3 Jitter and Phase Noise

The phase noise is shown in Fig. 6.5. The ADPLL achieved astonishing -130dBc/Hz @ 1MHz offset phase noise performance. The jitter analysis indicates the edge to edge jitter standard deviation is 2.5ps. The jitter histogram is included in Fig. 6.6.

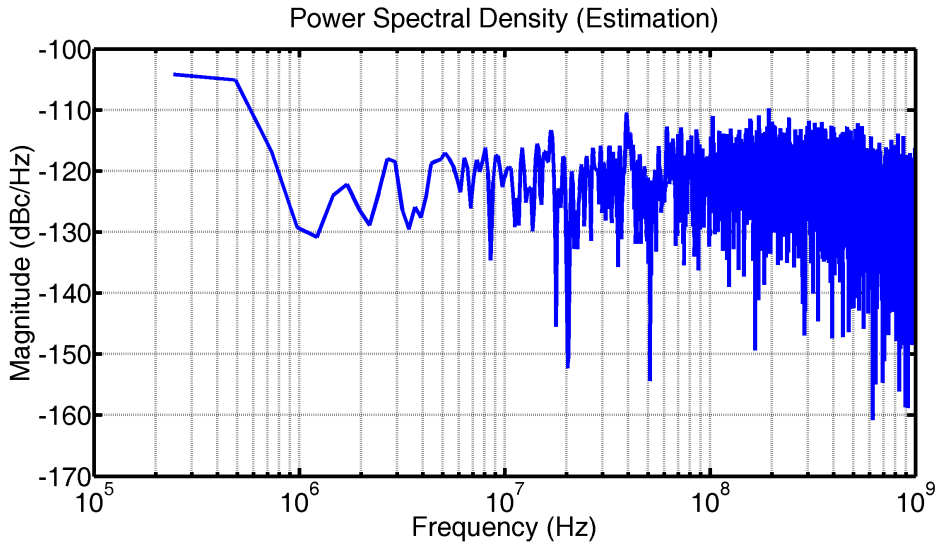

Figure 6.5: Control Code of ADPLL

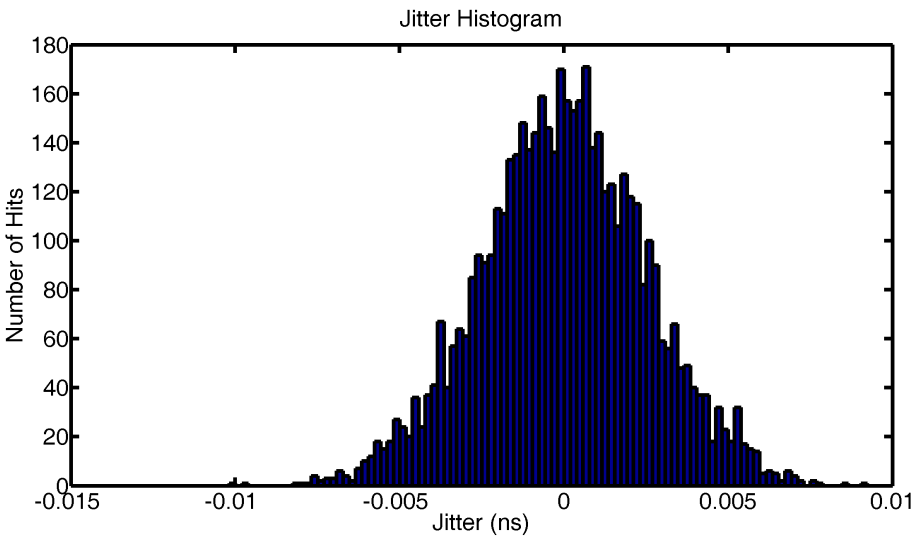

Figure 6.6: Jitter Histogram of ADPLL

The ADPLL performance is summarized in the following table.

Table 6.1: ALL Digital PLL Performance

Tuning range	1340MHz - 1860MHz
Jitter(std)	2.5 ps
Phase noise	-130dBc/Hz @ 1MHz
Multiply ratio	8X

CHAPTER 7

SUMMARY AND FUTURE WORK

7.1 Conclusion

So far this thesis has discussed about the application of the PLL and the importance of adopting PLL in the clocking circuit in chapter 1. An analog PLL analysis and in depth design example is presented in chapter 2 and 3. In order to achieve higher performance, lower power consumption, a all digital PLL analysis and design example is presented in chapter 4 and 5. Although output phase noise of -130 dBc/Hz @ 1MHz offset with 2.5ps peak to peak jitter is achieved by the ADPLL, there are still many circuit optimization can be done. In the next section a few ideas are proposed. Hopefully this thesis will serve as the start point for the new students in this field and inspire them to discover innovative research ideas.

7.2 Future Work

7.2.1 Lower PFD Noise

In the simulation of the ADPLL we discovered that the PFD quantization noise dominates the ADPLL noise. In order to suppress the PFD quantization noise the bandwidth has be to low, which in turn passes more VCO noise to the output. One way to avoid this is to combine the analog proportional path with the digital integral path since the proportional gain is much larger than the integral gain. By adopting analog proportional path we can avoid the major part of the quantization noise without using charge storage capacitor or charge pump.

7.2.2 Transistor Level ADPLL

The next step for the ADPLL design is clearly to synthesize the digital filter and build the DCO with capacitor array so that the whole system can be simulated and validate at transistor level.

7.2.3 Further Power Consumption Reduction

As we know at the near threshold voltage operation the device has much higher efficiency than in the normal operation condition. Of course the oscillation frequency will be much lower but this can be compensated by introducing multi-phase clock. Of course at near threshold voltage region the PVT problem becomes much more severe. The phase space calibration is unavoidable to compensate the process and device mismatch. Figure 7.1 shows a the basic architecture of a eight phases DCO with phase calibration. In

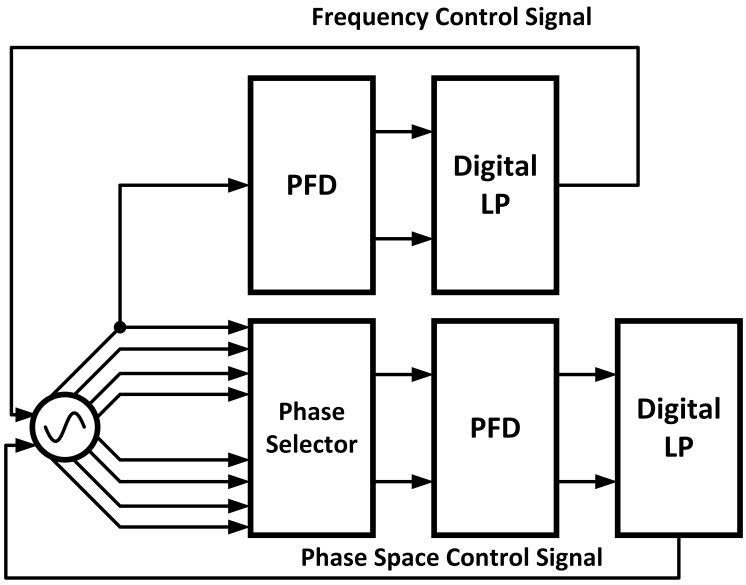

Figure 7.1: Near Threshold PLL with Phase Space Calibration

this design, there are two loops in the PLL circuit, one is for frequency and phase sync just like a conventional PLL; the other loop is specifically built for phase space calibration. When the PLL is turned on, the phase space calibration loop takes over the DCO and calibrates the phase space between ad joint phases. After several iterations the phase space between each phase will settle down to a fix value, after the calibration, the main loop takes over

to perform the conventional PLL function, the calibration loop will be turned off to save power.

REFERENCES

[1] K. Smith, A. Wang, and L. Fujino, "Through the looking glass?part 2 of 2: Trend tracking for isscc 2013 [isscc trends]," *Solid-State Circuits Magazine, IEEE*, vol. 5, no. 2, pp. 33–43, June 2013.

[2] J. C. Chen, "Mutli-gigabit serdes: The cornerstone of high speed serial interconnects." [Online]. Available: http://www.design-reuse.com/articles/10541/multi-gigabit-serdes-the-cornerstone-of-high-speed-serial-interconnects.html

[3] K. Iniewski, *CMOS Nanoelectronics: Analog and RF VLSI Circuits*. McGraw-Hill, 2011.

[4] V. Stojanović, "Channel-limited high-speed links: Modeling, analysis and design," Ph.D. dissertation, Stanford University, Palo Alto, 2004. [Online]. Available: http://chipgen.stanford.edu/papers/vs_thesis.pdf

[5] M. Mansuri, "Low-power low-jitter on-chip clock generation," Ph.D. dissertation, Univ. of California, Los-Angeles, 2003. [Online]. Available: http://www.ece.tamu.edu/~spalermo/ecen689/pll_thesis_mansuri_ucla_2003.pdf

[6] D. Friedman, "International solid-state circuits conference trends 2013," 2013. [Online]. Available: http://isscc.org/doc/2013/2013_Trends.pdf

[7] D. Pfaff, S. Kanesapillai, V. Yavorskyy, C. Carvalho, R. Yousefi, M. Khan, T. Monson, M. Ayoub, and C. Reitlingshoefer, "A 1.8w 115gb/s serial link for fully buffered dimm with 2.1ns pass-through latency in 90nm cmos," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, Feb 2008, pp. 462–628.

[8] G. W. R. Tsung-Yen Tsai, Sadok Aouini, "High speed on-chip signal generation for debug and diagnosis," *Journal of Electronic Testing*, vol. 28, no. 5, pp. 625–640, 10 2012.

[9] P. Hanumolu, G.-Y. Wei, U.-K. Moon, and K. Mayaram, "Digitally-enhanced phase-locking circuits," in *Custom Integrated Circuits Conference, 2007. CICC '07. IEEE*, Sept 2007, pp. 361–368.