DESIGN OF PLL-BASED CLOCK AND DATA RECOVERY CIRCUITS FOR HIGH-SPEED SERDES LINKS

BY

ISHITA BISHT

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Electrical and Computer Engineering
in the College of Engineering of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Professor José Schutt-Ainé

ABSTRACT

Over the years, the thirst for high speeds in data transmission has become unquenchable. Todays demands necessitate operation of transmission links at speeds in the gigahertz range. At such accelerated rates, maintaining signal integrity becomes a challenging task. It is imperative for creditable data links to offer high data accuracy at high speeds.

One of the technologies implemented today to achieve high-speed data transmission with signal integrity is the SerDes (Serializer- Deserializer). The SerDes implementation offers the advantage of low manufacturing cost and less crosstalk. In this design the incoming parallel data is mapped onto a single data stream using a serializer before transmission. The serial data is transmitted over the channel along with an embedded clock. The data is received at the other end of the channel and deserialized to generate parallel data output.

The data clock is generated by using a phase locked loop (PLL) as a frequency synthesizer. It steps up the clock frequency of a crystal clock to that of the data rate. The data integrity that the SerDes offers is predominantly due to the clock and data recovery circuit (CDR) employed within the design. The CDR takes the incoming data and generates a clock using the data specs which can then be used by the deserializer to sample the data accurately.

This thesis looks into the basic principles of operation of phase locked loops, Clock and Data recovery circuits and their building blocks for a 1.6 Gbps SerDes link. It summarizes the challenges in design and also presents a Cadence approach to the circuit design in 180 nm CMOS technology.

To the fond memories of my grandparents Mr and Mrs B.S Butola

ACKNOWLEDGMENTS

Apart from my efforts, the success of my project depends significantly on the encouragement and guidelines of several others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this thesis.

Foremost, I would like to express my profound gratitude and deep regards towards my thesis adviser Professor José Schutt-Ainé for his exemplary guidance, monitoring and constant encouragement through the course of my thesis. His advice has been extremely instrumental in shaping my academic path.

I would like to thank Professor Pavan Kumar Hanumolu for his insightful inputs on the project that helped me in accomplishing the thesis and also Professor Chandra for his precious time and extremely helpful advice that encouraged me to believe in myself through my undergraduate senior year. My heartfelt thanks to my graduate mentor, Rishi Ratan for introducing me to the research team and for significantly helping out throughout the course of my research. A big note of thanks to Romesh Nandwana , Guanghua Shu, Da Wei and Yubo Liu for providing excellent inputs when I went to them with questions on the project . Without all of them this thesis would not have materialized. I am grateful to them for their constant support and encouragement as well.

I thank my research teammates Xinjing, Lei Jin, Jerry Yang, Maryam Hajimiri, Brady Salz, Ankit Jain, Haodong , Pradyut Paul, Yi Ren, and Manfei Wu for having shared this experience with me. I also thank Rushabh Mehta for the stimulating discussions and sleepless nights we spent working together on the project. In addition, I would like to express my deepest appreciation to Sagar Dhawan for taking out time from his schedule and helping out when things seemed overwhelming.

I would like to express immense gratitude towards my parents for their unequivocal support and belief in my potential. I express my sincere thanks to Mr Ravikant and Mr Saxena for their intensive career advice. Lastly, I would like to thank Nishan Noronha for his words of encouragement that kept me motivated to strive towards my goals.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation

There has been a tremendous amount of progress over the years in the field of high speed data communication. The complexity of the system-on chip (SoCs) has increased requiring several interconnections on chip. Communicating data at high speeds becomes a challenging task. To meet with the high processing speeds and enhance system performance it becomes essential to have high-speed inter-connects. Broadly the two main architectures that exist are serial and parallel links. In serial communication, the data is sent over a single channel sequentially while in parallel links the data is sent over multiple-channels simultaneously. The multi-bit parallel links have long been in use, but when it comes to high speed long interconnects these fail to meet performance standards. The diminishing cost of integrated circuits (ICs) in combination with the rising demand for high speeds has contributed in favor of serial data links. Serial links offer several advantages over parallel links [1] for such applications such as

1. **Low cost:** Due to lesser number of pins and lower board space usage. The extra space can be reused for other components.

2. **Less crosstalk:** Fewer conductors in proximity, thus serial links minimize cross-talk

3. **Clock skew issues:** Parallel links face clock skew issues between parallel channels. This becomes redundant for serial links as they usually use asynchronous and unclocked data transmission.

Originally serial links were used for long channel transmissions. Today they are widely used in computer networks such as WAN, LAN and Ethernet as well as computer to peripheral devices such as universal serial bus (USB). Multimedia interface like HDMI, computer to storage interface like Serial ATA (SATA), backplane links like PCI Express and high speed telecommunication architectures like SONET and SDH are among other examples of high-speed serial data links.

1.2 Purpose

The objective of this thesis is to present an introduction to the theoretical concepts behind the design of a single loop Clock and Data Recovery Circuit for a high-speed SerDes link. This thesis entails the fundamental concepts underlying the functioning of CDRs and presents a Cadence Virtuoso tutorial for its design in 180 nm CMOS technology.

CHAPTER 2

SERDES LINKS

In modern times to take advantage of both topologies, often applications involve both parallel and serial communications. A SerDes transceiver is one such application. The main function of the SerDes system is to transmit data at high speeds over a channel and receive the correct data at the receiver end. It facilitates the transmission of data between parallel I/O ports over a serial link by sequentializing data. Thus, the number of channels and pins are reduced. The SerDes becomes especially beneficial for high data rates when the issues with parallel buses are more heightened. For the implementation, CMOS technology proves to be the ideal choice due to its low cost and low power consumption. The SerDes is primarily a mixed-signal circuit with several analog as well as digital blocks to perform different functions (See Figure 2.1).

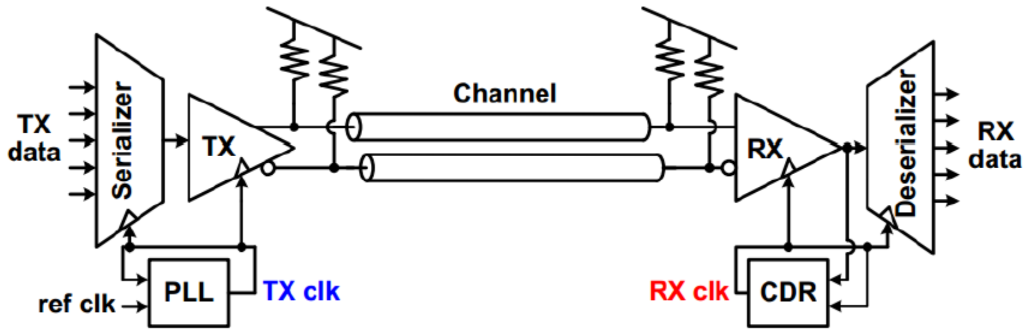


Figure 2.1: The SerDes Link

2.1 Basic Operation

Parallel data is fed into the parallel input ports of the Serializer. These data bits are clocked into the parallel bit registers of the Serializer based on the reference clock edge. Usually the reference clock is the output of a crystal oscillator or the output of a phase locked loop. When the bits are loaded, they are then encoded using standard protocols. The coded data is then serialized and moved on to media such as a copper cable or backplane trace. An equalizer is used to ensure signal integrity of the transmitted data. At the other end the data is received by the clock and data recovery block which provides a recovered clock to retime the data with. The received data is then deserialized and decoded back to its original form.

2.2 Functional Blocks

2.2.1 The Serializer

The function of the serializer is to accept the parallel data bits at a particular rate, serialize them and then drive the data into the communication channel.

2.2.2 The Phase-Locked-Loop (PLL)

The Phase-locked-loop forms an essential part of the SerDes circuitry as it generates the high frequency reference clock used to drive the serial transmitter. The reference clock inputs are often required to meet tight electrical and jitter requirements as the quality of the clock can have a significant impact on the links performance. The PLL is primarily used as a frequency synthesizer and is responsible for stepping up the clock frequency by using a divider in the feedback loop.

2.2.3 Equalizer

Data travelling at high frequencies through a channel suffers primarily due to resistive loss, dielectric absorption and skin effect. This causes the closure of the eye in the eye diagram because of the attenuation of high frequency content. For this purpose, we use an equalizer on the receiver side. This helps in opening the eye by suppression of low frequency content and increment of high frequency content. The equalizer can also be made to adapt to the transient properties of the communication channel. Such equalizers are called adaptive equalizers.

2.2.4 The Deserializer

The deserializer receives the retimed data from the CDR block within the receiver slice and deserializes it back to parallel form using the CDR generated clock.

2.2.5 Clock and Data Recovery (CDR)

Sometimes the data is sent over the channel without an accompanying clock signal. The CDR block performs the role of generating a clock from the received data. It receives the data at an approximately known frequency and then uses phase alignment to align the generated clock to the transitions in the data. This clock and retimed data is then used by the deserializer to recover the data. The aligned clock now allows the data to be sampled at the middle of the eye and hence supports the accurate recovery of data. The middle of the eye is also the most optimum point to take a data sample as it has the least jitter usually.

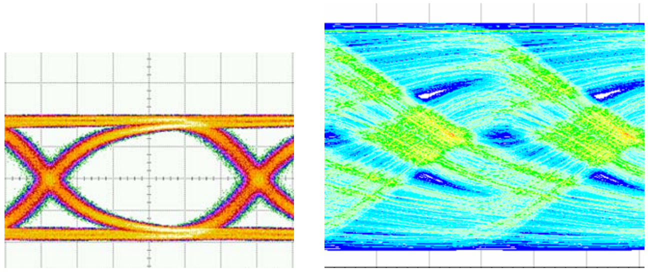
CHAPTER 3

CDR THEORY AND BACKGROUND

Binary data is usually transmitted in the NRZ (non-return to zero) format. In this format each bit has a duration Tb , which is also known as the bit period. A bit is equally likely to be a 1 or a 0. Unlike the RZ (return to zero) data , here the signal does not drop to zero between consecutive data bits. This also gives the NRZ format the advantage of lesser transitions. The regeneration of the data at the receiver end must be error-free. For this purpose it is essential to sample the data at optimum instants. Generally, the timing information is derived from the incoming data itself. The recovered clock removes both jitter as well as data distortion. The clock generated must:

1. Operate at a frequency equal to the data rate
2. Must have appropriate timing with respect to data to allow optimum sampling of the data
3. Must exhibit less jitter

PLLs are widely used in the CDR application to enable recovery and synchronous processing of data. This thesis presents the design of such a PLL-based CDR circuit. Finding the optimum sampling point is one of the most challenging tasks in designing a CDR. Continuous Identical data (CID) sequence also becomes an issue which can be resolved by using a balanced coding scheme such as 8b/10b encoding to guarantee minimum transition. At todays gigahertz transmission frequencies, several variables can affect the integrity of the signals [2]. One way to evaluate the performance and get a better understanding of the effects a noisy channel can have on the recovered data is to observe its eye diagram. An eye diagram is a good indicator of the signal quality in high speed data transmissions. It is generated by overlaying sweeps of different segments of a data stream driven by a clock. Overlaying multiple bits produces an eye diagram which looks like an open eye. Ideally the eye diagrams should look like rectangular boxes. However, in reality due to imperfections in transmission the boxes transform into the shape of an eye. The width of the eye decreases with increase in noise.



(a) (b)

Figure 3.1: (a) Open and (b) Closed Eye Diagrams

For optimum data sampling we want to sample the data at the center of an open eye. If the eye is closed it becomes difficult to recover data and equalization techniques must be used to compensate for the loss through the communication channel [2].

3.1 Phase Locked Loop

A PLL in the most basic of terms is a feedback system. It constitutes of a Phase Detector, charge pump loop filter and a voltage controlled oscillator. Figure 3.2 shows a basic phase locked loop.

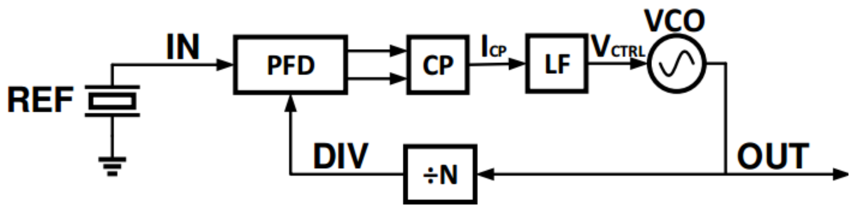


Figure 3.2: A Basic Phase Locked Loop

The PLL is said to achieve lock when the phase difference between the two input signals becomes constant and the corresponding frequencies become equal. The phase frequency detector (PFD) generates an error pulse for a phase difference between the two signals. This error signal is often amplified and converted into an analog signal by a charge pump. The analog output of the pump is passed onto the loop filter which suppresses the high frequencies, enabling the DC component called control voltage. This control voltage is the controlling input of the VCO which determines the oscillation frequency of the VCO. The VCO changes its frequency to accumulate enough phase for the PLL to achieve lock. The VCO output is fed back into the PD for comparison.

3.2 Basic Components of CDR

A PLL-based CDR has the same components as a PLL. The main difference arises in the Phase detection scheme. The PD used in the CDR differs from the PFD employed in the PLL. It should detect the data transitions as well as the phase difference between the data and the recovered clock. Another difference is that one of the inputs into a PLL is a reference clock signal while the input to a CDR is non-periodic data.

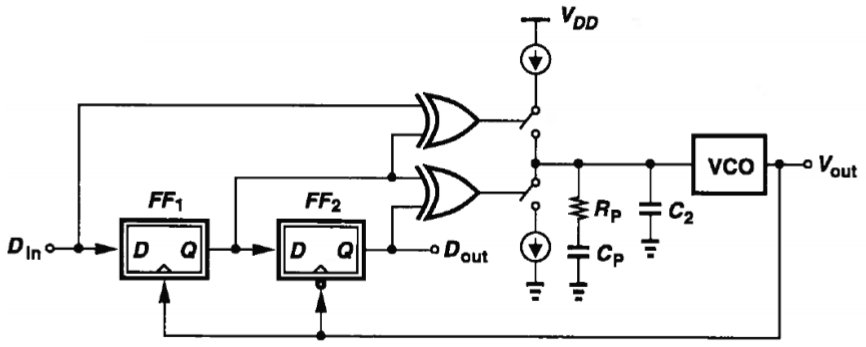


Figure 3.3: A Hogge PD based CDR Circuit [3]

3.2.1 Phase Detector

CDR Phase detectors compare the phase between the input data and the recovered clock used to sample the data and use the information to align the sampling clocks phase to the data. Thus, they should provide two essential functions:

1. Data transition detection

2. Phase difference detection

They can be either linear or non-linear. A linear PD gives both sign and magnitude information regarding the phase error (Ex: Hogge PD). A non-linear PD provides only sign information (Ex: Bang-Bang or Alexander PD).

1. **The Hogge Phase Detector**

   The Hogge PD is a linear phase detector. The PD makes use of the VCO output to sample the data. It consists of two D Flip flops and a XOR gates put together as shown in Figure 3.4 [4].

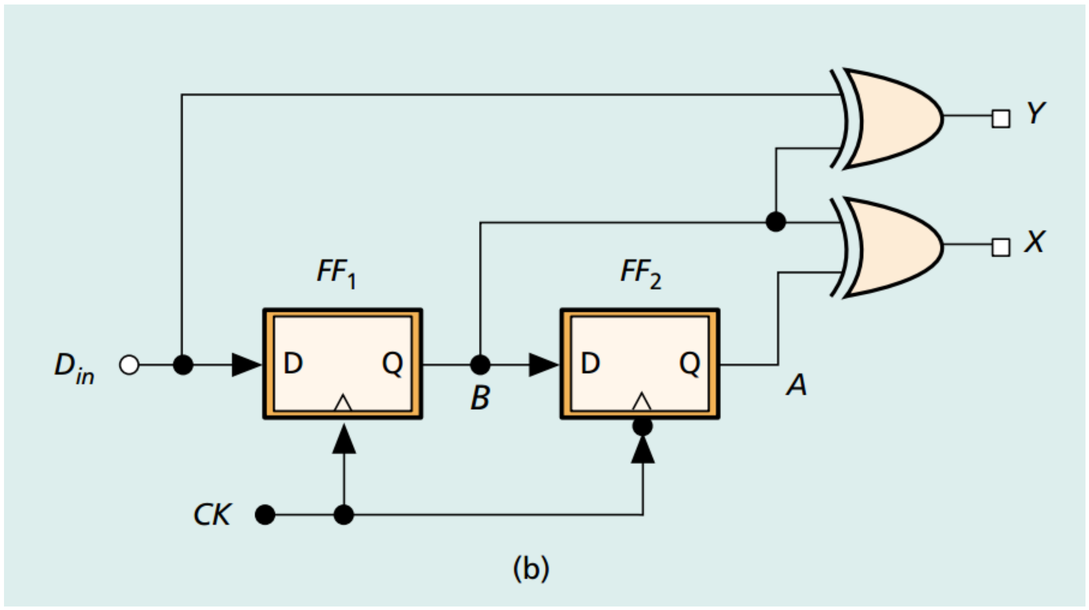
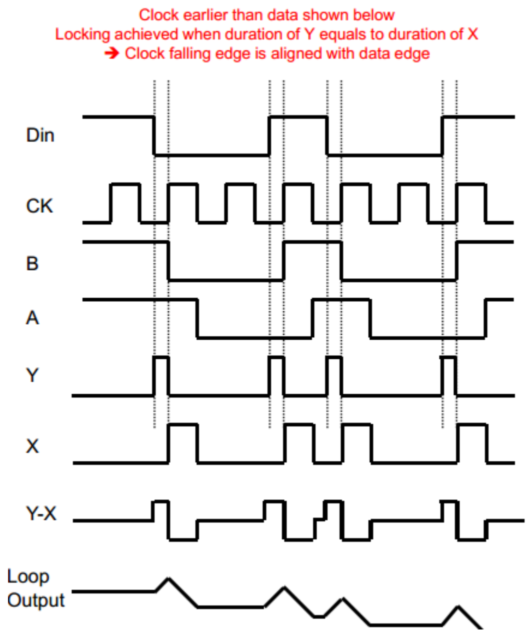


Figure 3.4: A Hogge Phase Detector [4]

The combination works as an edge detector. The sample B is a delayed version of the input and it changes only on the positive clock edge. Thus the output $Y = D_{in} + B$, it contains the pulses whose widths indicate the phase difference between the incoming data and the clock. Note that the circuit produces a pulse for each data transition providing edge detection and the width of each output varies linearly with the phase difference. The retimed data (B) is delayed by half a clock cycle and XORed with itself. This generates pulses of width of half a clock cycle for each transition When the CDR is locked the duration of the XOR output pulses X and Y is the same.



Clock earlier than data shown below
Locking achieved when duration of Y equals to duration of X
➔ Clock falling edge is aligned with data edge
Din
CK
B
A
Y
X
Y-X
Loop Output

Figure 3.5: Output Waveform of Hogge PD

Din = Incoming Data, Clk = Clock

B and A = Intermediate outputs of PD

X and Y = XOR outputs of the PD

Adapted from [5]

To summarize, this PD samples the data using the VCO clock for which a DFF and a XOR gate perform the explicit function of edge detection. Then, we produce a reference pulse, using the other DFF and XOR to eliminate the ambiguity for different data transitions. Due to the Clk to Q delays of the flip flops, the Din and Clk must sustain a skew to equalize the widths of the output pulses. This skew effect becomes significant at high speeds as the skew ΔT becomes a significant fraction of the clock period. This might lead to a phase offset after the loop is locked which degrades the clock phase margin and jitter tolerance. To overcome this, we can widen the reference pulses by $\frac{\Delta T}{2}$ or narrow the proportional output pulses by the same amount. One of the disadvantages of the Hogge PD is that there is a skew of $\frac{T_{Clk}}{2}$ between the two output pulses in the locked condition. This causes a lot of disturbance in the VCO. As this is a linear PD it sends out small average signals to the charge pump resulting in little activity at the charge pump.

2. **The Bang-Bang or Alexander Phase Detector**

   This is the most commonly used CDR Phase Detector. This PD marks two data samples and one edge sample. The figure shows the design of the Alexander PD. Using the three samples the PD can determine whether a data transition is present, and if the clock (leads) is early or late (lags).

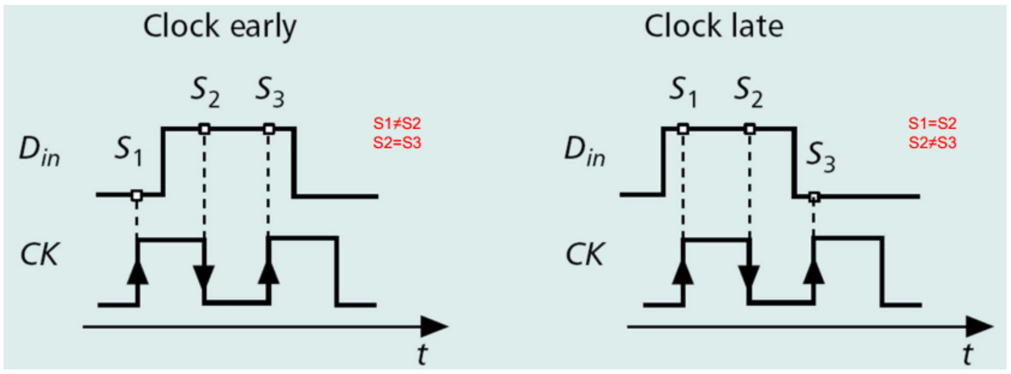


Figure 3.6: Early and Late Clock Definitions. Adapted from [4]

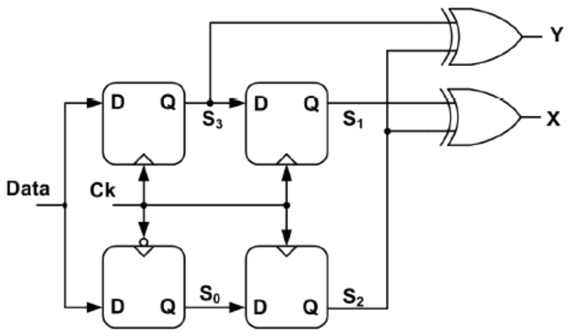


Figure 3.7: Alexander Phase Detector

If there are no transitions then all samples are the same and there is no action taken. If the clock is early, the first sample S_1 is not equal to the other two samples. On the other hand, if the clock is late, the first two samples S_1 and S_2 are the same but S_3 is different. Thus, the XOR outputs of the combinations, $S_1 \oplus S_2$ is high and $S_2 \oplus S_3$ is low. This states that the clock is late. If $S_1 \oplus S_2$ is low and $S_2 \oplus S_3$ is high, clock is early. If $S_1 \oplus S_2$ is equal to $S_2 \oplus S_3$ then no data transitions exist.

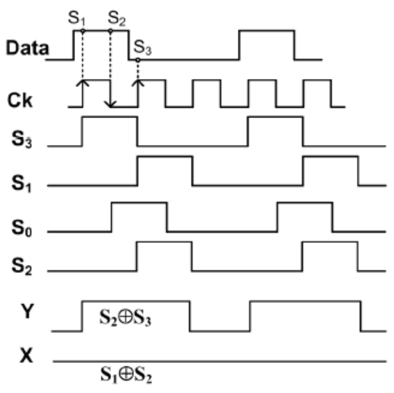


Figure 3.8: Output Waveform of Alexander Phase Detector

The first positive edge of the clock samples high data. At the second positive edge, a delayed version of the first sample is generated at the output of FF_2. Also, the low level on the input data is sampled. The S_1 and S_2 values can be compared and they stay constant for about a clock period. This PD exhibits a very high gain when phase diff is close to being zero. Thus, a Bang Bang PD based CDR locks such that S_2 coincides with the zero crossings of the data. Two of the major advantages of this PD architecture are that it retimes the data by itself, producing valid signals at the outputs and that it takes no action when there are no transitions. The BBPD also offers low jitter specs.

S_1	S_2	S_3	$Y = S_1 \oplus S_2$	$X = S_2 \oplus S_3$	Detection (Action)
0	0	0	0	0	no decision (no action)
0	0	1	0	1	early (slow down)
0	1	0	1	1	no decision (no action)
0	1	1	1	0	late (speed up)
1	0	0	1	0	late (speed up)
1	0	1	1	1	no decision (no action)
1	1	0	0	1	early (slow down)
1	1	1	0	0	no decision (no action)

Figure 3.9: Truth Table for the Alexander Early Late PD

3.2.2 Charge Pumps

A charge pump consists of a current source or sink that charges or discharges the loop filter according to its inputs. The main idea is to have continuous flow of current which changes direction only when needed. This way peaking during transitions is reduced. A unity gain amplifier is added to prevent charge sharing.Charge-Pump PLLs offer many advantages over the classical voltage phase-detector PLL including an infinite pull-in range and zero steady-state phase error.

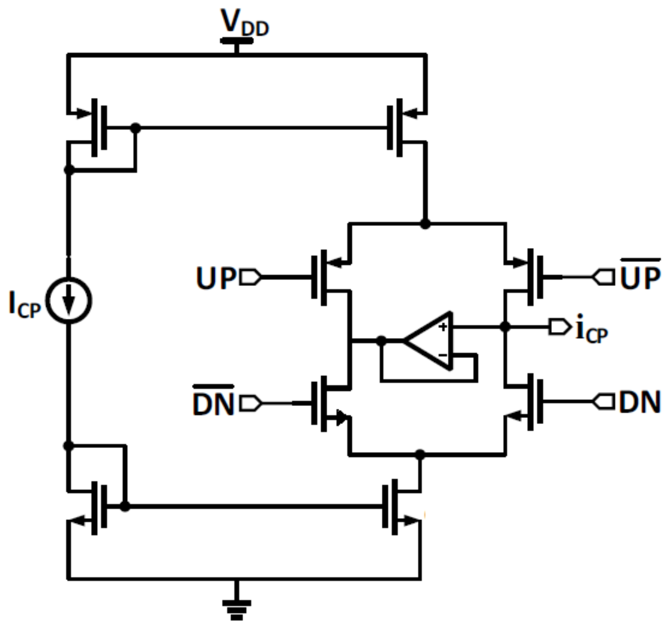


Figure 3.10: Truth Table for the Alexander Early Late PD

The output of the PD forms the inputs to the charge pump. The charge pump converts the digital inputs to current signals which turns on or turns off the current source and sinks in the charge pump. The bootstrapped design shown above has the advantage of differential current steering, it can operate at low swing input signals as well which makes it useful for PLL with a high speed reference clock [6].

3.2.3 Loop Filter

The function of the loop filter is to reject the high frequency components of the PD output. The loop filter is primarily a network of passive components consisting of RC networks. The output of the loop filter is the control voltage of the VCO.

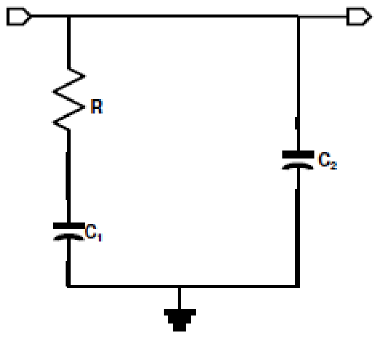


Figure 3.11: Loop Filter

3.2.4 Voltage Controlled Oscillator

A voltage controlled oscillator generates an output signal in response to its control voltage. The frequency of the output voltage is often proportional to the control voltage. The control voltage is the output of the Loop Filter. If the control voltage rises the VCO produces an output with a higher frequency and vice versa. In the locked state of the CDR the control voltage becomes constant, thus the oscillation frequency of the VCO clock is also fixed.

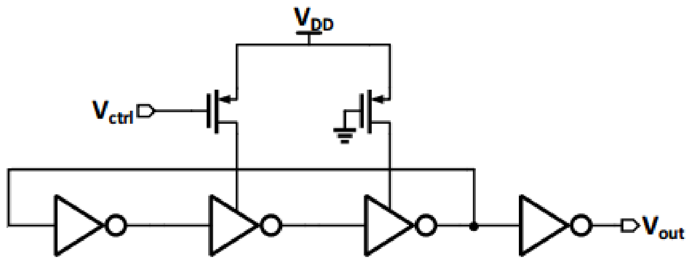


Figure 3.12: Single-Ended Voltage Controlled Oscillator

CHAPTER 4

LINEAR ANALYSIS OF CDR LOOP

For simplicity we can assume a linear model of the PLL[7]. Figure 4.1 shows a simple linear representation of the loop.

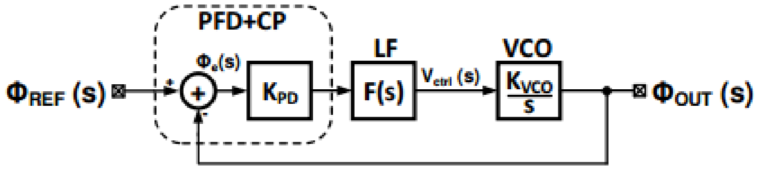


Figure 4.1: Linear Model of CDR

Using the linear model, we can estimate its transfer function as follows,

$$H(s) = \frac{K_{PD} \cdot K_{VCO}}{\frac{s^2}{\omega_{LPF}} + s + K_{PD} \cdot K_{VCO}} \tag{4.1}$$

On comparing this with the familiar transfer function equation from control theory,

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4.2}$$

We obtain the following results,

$$\omega_n = \sqrt{\omega_{LPF} \cdot K} \tag{4.3}$$

$$\zeta = \frac{1}{2} \cdot \sqrt{\frac{\omega_{LPF}}{K}} \tag{4.4}$$

Where the loop gain is,

$$K = K_{PD} \cdot K_{VCO} \tag{4.5}$$

ω_n and ζ are the damping factor and the natural frequency of the system. ω_n is also the geometric mean of the -3db bandwidth of the LPF and the loop gain. It also is an indicator of the gain-bandwidth product of the loop.

The damping factor usually has a value that is greater than 0.5. There is an upper bound on ω_{LPF} due to noise suppression, also limiting the K.

From the figure, the open loop transfer function can be written as,

$$LG(s) = K_{PD} \cdot F(s) \cdot \frac{K_{VCO}}{s} \tag{4.6}$$

$$= K_{PD} \cdot K_{VCO} \cdot \frac{s + \frac{1}{RC_1}}{C_2 s^2 (s + \frac{C_1 + C_2}{RC_1 C_2})} \tag{4.7}$$

From the above ,

$$\omega_z = \frac{1}{RC_1};\ \omega_{p1} = \omega_{p2} = 0; \omega_{p3} = \frac{C_1 + C_2}{RC_1 C_2} \tag{4.8}$$

From the condition for maximum phase margin we get,

$$\omega_{ugb} = \omega_z \sqrt{\frac{C_1}{C_2} + 1} \tag{4.9}$$

Also,

$$\phi_{M_{max}} = \tan^{-1}\left(\sqrt{\frac{C_1}{C_2} + 1}\right) - \tan^{-1}\left(\frac{1}{\sqrt{\frac{C_1}{C_2} + 1}}\right) \tag{4.10}$$

where ω_{ugb} is the open loop unity gain bandwidth and $\omega_z < \omega_{ugb}$.

4.1 Design Procedure

The design procedure of the loop filter is as follows: For our CDR design we estimate the K_{PD} and accordingly determine the bandwidth of the CDR to be designed. The bandwidth taken into consideration is the shown above. We also determine the phase margin value to design with. The R value of the resistor of the loop filter is determined to offer low noise at the V_{cntrl} input. We can calculate constant K_c as follows,

$$K_c = \frac{C_1}{C_2} = 2(\tan^2(\phi_M) + \tan(\phi_M \sqrt{\tan^2(\phi_M) + 1})) \tag{4.11}$$

Next,

$$\omega_z = \frac{\omega_{ubg}}{\sqrt{\frac{C_1}{C_2} + 1}} \tag{4.12}$$

$$C_1 = \frac{1}{\omega_z R} and C_2 = \frac{C_1}{K_c}; \tag{4.13}$$

The charge pump current is,

$$I_{CP} = \frac{2\pi C_2}{K_{VCO}} \cdot \omega_{ugb}^2 \cdot \sqrt{\frac{\omega_{p3}^2 + \omega_{ugb}^2}{\omega_z^2 + \omega_{ugb}^2}} \tag{4.14}$$

give us our $C_1, C_2 and I_{CP}$ values as well. To confirm the lock condition, The closed loop transfer function is,

$$H_{PLL}(s) = \frac{LG(s)}{1 + LG(s)} \tag{4.15}$$

The steady state error would be defined as

$$\frac{\Phi_{error}(s)}{\Phi_{in}(s)} = H_e(s) = 1 - H_{PLL}(s) = \frac{1}{1 + LG(s)} \tag{4.16}$$

Using the final value theorem,

$$\begin{aligned}
\Phi_{ss\_error}^{F_{step}} &= \lim_{s \to 0} s \cdot H_e(s) \cdot \Phi_{in}(s) && \text{(4.17a)} \\
&= \lim_{s \to 0} s \cdot \frac{1}{1 + LG(s)} \cdot \frac{\Delta\omega}{s^2} && \text{(4.17b)} \\
&= \lim_{s \to 0} \frac{[RC_1C_2s^2 + (C_1 + C_2)s]\Delta\omega}{RC_1C_2s^3 + (C_1 + C_2)s^2 + K_{VCO}K_{PD}s + 1} && \text{(4.17c)} \\
&= \frac{0}{1} && \text{(4.17d)} \\
&= 0 && \text{(4.17e)}
\end{aligned}$$

The zero phase error indicates that the PLL has achieved lock condition.

CHAPTER 5

TRANSISTOR-LEVEL SIMULATIONS

The transistor-level schematics are created on the Cadence Virtuoso platform which is an Electronic Design Automation (EDA) tool for fully customized IC designs. It incorporates schematics, behavioral modelling, circuit simulations and custom layout. The simulator used for our research was Spectre. It is closely integrated with Cadence Virtuoso and is capable of providing in depth transistor level analysis in multiple domains. Spectre is SPICE-class circuit simulator. It provides fast, accurate SPICE-level simulations for analog, RF and mixed signal circuits.

5.1 Transient Simulations

The transient response is the simulation of circuit parameters over a time period. It is used to understand the time-domain behavior of the node voltages and currents.

5.2 CDR Design and Simulation in Cadence Virtuoso using Spectre Simulator

The following tutorial presents a step-by-step procedure to implement a Hogge PD based CDR circuit at transistor level and to simulate it using Spectre. We will create transistor level blocks of the CDR components and then later integrate them into a single schematic at the block level. For the Hogge PD we need:

1. Positive Edge triggered D Flip Flop (DFF)
2. Negative Edge triggered D Flip Flop (DFF)

3. Exclusive-OR Gate

We create each of the above as individual cells, create symbols for them and using these symbols integrate them to form the PD on a new schematic.

5.2.1 Creating a Schematic for positive DFF

The schematic for the positive DFF should be modelled after the following TSPC DFF implementation.

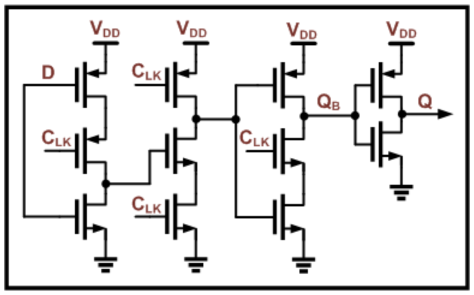
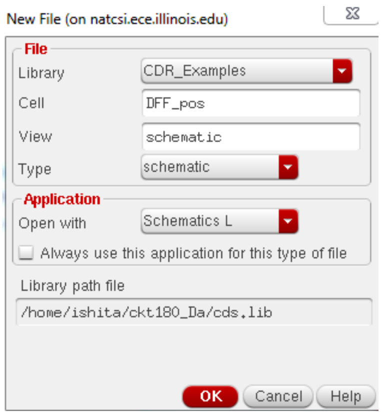


Figure 5.1: NAND PFD Implementation

1. Start by creating a new library under the Library Manager window. Name it CDR. Then, highlight the new library and clock on File New Cell View.



New File (on natcsi.ece.illinois.edu)

File

Library: CDR_Examples

Cell: DFF_pos

View: schematic

Type: schematic

Application

Open with: Schematics L

☐ Always use this application for this type of file

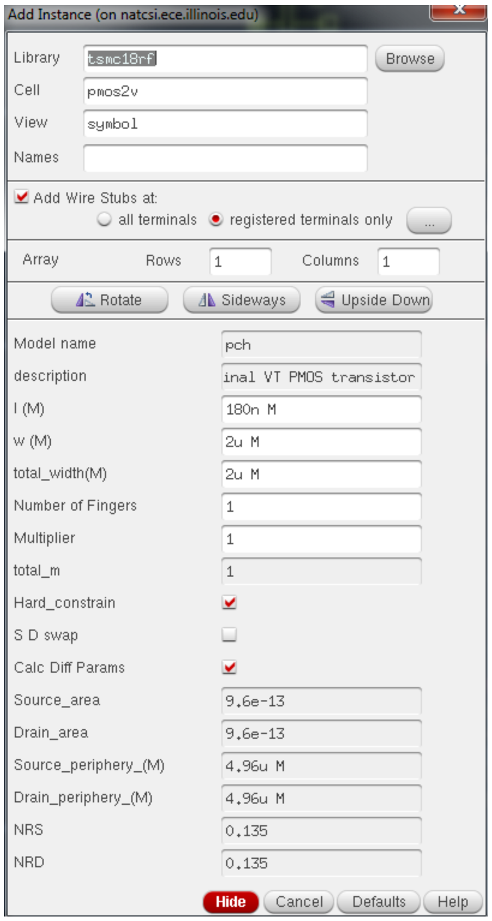
Library path file: /home/ishita/ckt180_Da/cds.lib

Figure 5.2: Creating a New Schematic

Under cell, enter the name of the cell. Click on OK. This creates a schematic under the name within the library created. The schematic will show up on the library manager as well. On double-clicking the schematic window will open.

2. This schematic window is where the transistor level DFF is created. The circuit components are available under different libraries. To create the DFF we need PMOS and NMOS instances. Use the *create instance* icon on the tool bar or use the shortcut key *'I'* to access the instances catalog. Select the 'tsmc18rf' library. Enter the instance name or search through the list of components.

3. Select *pmos2v* as the PMOS transistor of the DFF. For the NMOS transistor use *nmos2v*.


Add Instance (on natcsi.ece.illinois.edu)

Library	tsmc18rf
Cell	pmos2v
View	symbol
Names	

[✓] Add Wire Stubs at: all terminals [✓ registered terminals only]

Array Rows 1 Columns 1

Model name	pch
description	inal VT PMOS transistor
l (M)	180n M
w (M)	2u M
total_width(M)	2u M
Number of Fingers	1
Multiplier	1
total_m	1
Hard_constrain	✓
S D swap	☐
Calc Diff Params	✓
Source_area	9.6e-13
Drain_area	9.6e-13
Source_periphery_(M)	4.96u M
Drain_periphery_(M)	4.96u M
NRS	0.135
NRD	0.135

Figure 5.3: Adding an instance

Properties such as width, multipliers etc can be added when adding the component. These properties can be edited by selecting the component in the schematic and pressing the key Q to open *Edit Object Properties* table.

4. To connect the components use (narrow wire) or the shortcut key.

5. After constructing the TSPC based positive edge triggered DFF, the schematic should look like the one shown in Figure 5.4.

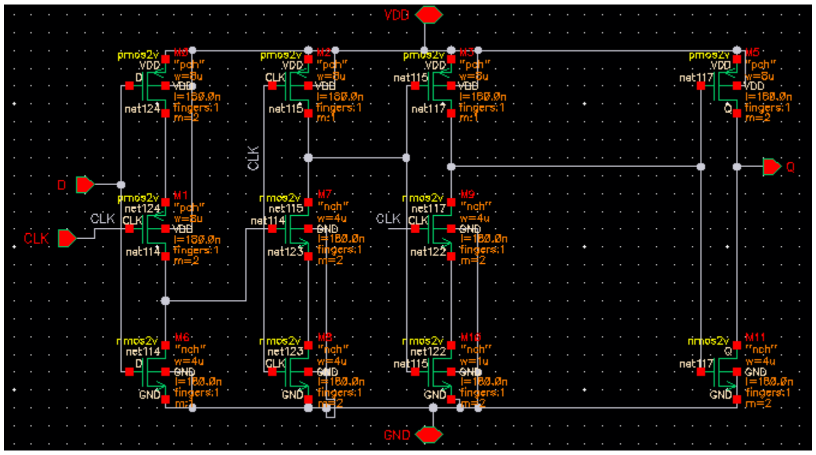
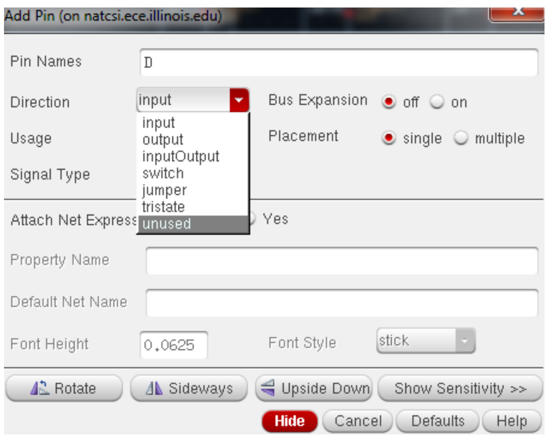


Figure 5.4: Positive Edge Triggered TSPC DFF in Cadence Virtuoso

6. To add the pins for the input and outputs use the icon from the toolbar or use the shortcut key p. Use *inputOutput* direction for VDD and GND pins as shown in Figure 5.5.


Add Pin (on natcsi.ece.illinois.edu)

Pin Names: D

Direction: input

Bus Expansion: [✓ off] on

Placement: [✓ single] multiple

Usage

Signal Type

Attach Net Express Yes

Property Name

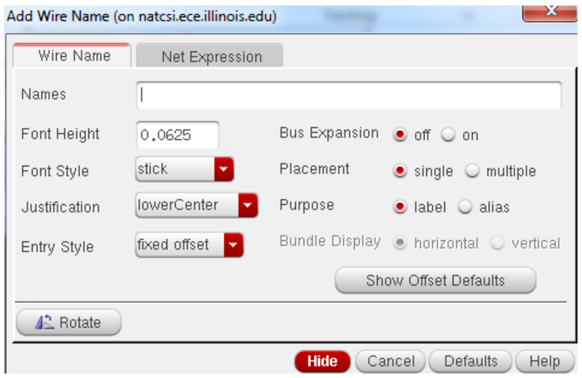
Default Net Name

Font Height: 0.0625

Font Style: stick

Figure 5.5: Adding a Pin

7. Use labels to give connecting wires a name. Use or the shortcut key l to open the label window.


Add Wire Name (on natcsi.ece.illinois.edu)

Wire Name | Net Expression

Names:

Font Height: 0.0625

Font Style: stick

Justification: lowerCenter

Entry Style: fixed offset

Bus Expansion: [✓ off] on

Placement: [✓ single] multiple

Purpose: [✓ label] alias

Bundle Display: [✓ horizontal] vertical

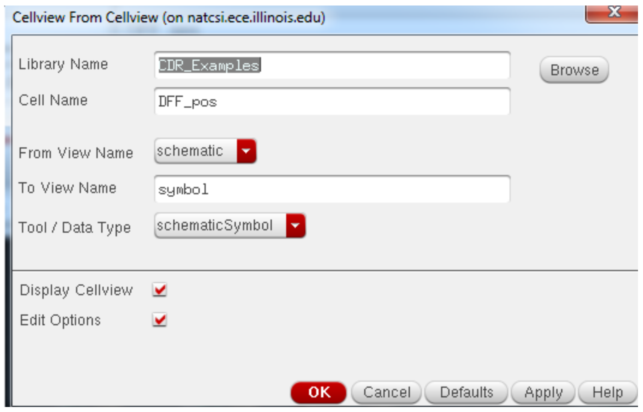
Figure 5.6: Naming a wire conneciton

8. After completing the schematic *Check and Save* it ✓ 💾. This is very important to do before simulation.

5.2.2 Creating a Symbol

Now the DFF created above can be converted into a symbol. We use symbols as circuits become extremely complex if transistor levels are used for each functional block. Thus, using symbols makes it easier to integrate the circuit for a higher level simulation.

1. To create a symbol, first open the schematic. From the schematic window go to Create *Cellview* From *Cellview*. This opens a window to generate a symbol for the schematic.


Cellview From Cellview (on natcsi.ece.illinois.edu)

Library Name: CDR_Examples

Cell Name: DFF_pos

From View Name: schematic

To View Name: symbol

Tool / Data Type: schematicSymbol

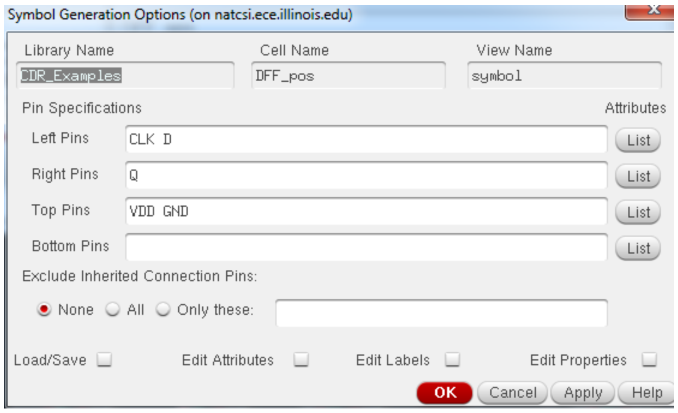
Display Cellview: ✓

Edit Options: ✓

Figure 5.7: Creating a Symbol from Cellview

Click OK above and another window opens with the symbol options.

In that window indicate the pins needed on the symbol and the side on which to draw them.


Symbol Generation Options (on natcsi.ece.illinois.edu)

Library Name: CDR_Examples
Cell Name: DFF_pos
View Name: symbol

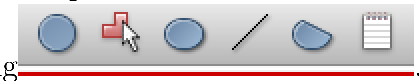
Pin Specifications

Pin Specifications		Attributes
Left Pins	CLK D	List
Right Pins	Q	List
Top Pins	VDD GND	List
Bottom Pins		List

Exclude Inherited Connection Pins: [✓ None] All Only these:

Load/Save ☐ Edit Attributes ☐ Edit Labels ☐ Edit Properties ☐

Figure 5.8: Generating a Symbol

2. This will create a symbol for the circuit which opens up in a new window. The shape of the symbol can be modified using . The shape has no effect on the functioning of the block.

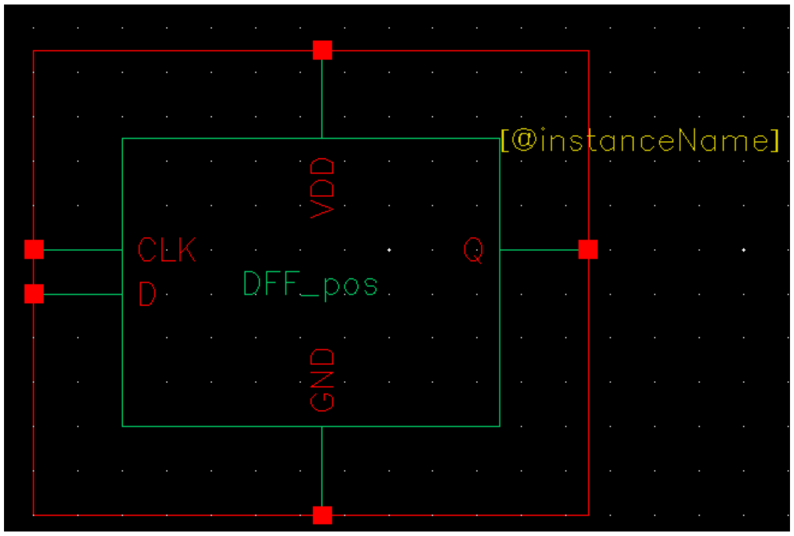


Figure 5.9: Symbol for positive edge DFF

5.2.3 Schematics for Negative TSPC DFF

1. Create a schematic and then symbol for the negative edge DFF and XOR gates in a similar fashion as shown above. The designs are shown in Figures 5.10-5.11.

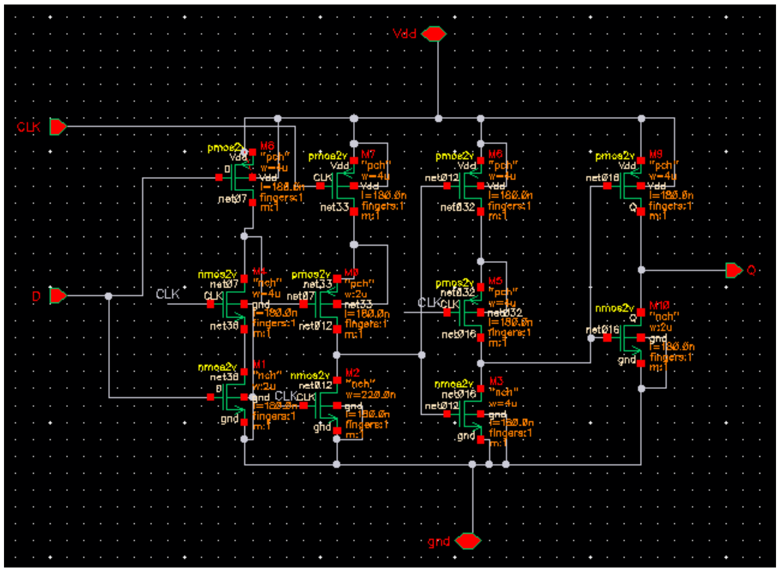


Figure 5.10: Negative Edge triggered TSPC DFF in Cadence Virtuoso

Remember to add an inverter to the above output to have Q as the final output and not Qbar.

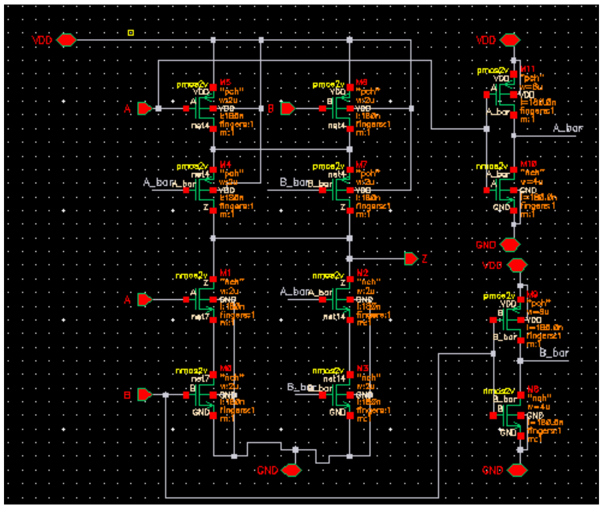


Figure 5.11: An exclusive OR (XOR) Gate

5.2.4 Integrating the blocks for Hogge PD component

Now that the component blocks of the Hogge PD are built, we can put them together to form the Hogge PD in a new schematic.

1. Under the same library name *CDR* create a new cell and name it HPD.

2. In the new schematic window that opens up add instances of the DFF and XOR by looking for them under the *CDR* library in the component catalog.

3. As mentioned in the previous section, connect the sub parts and form the symbol for the PD. Now, when creating the schematic for the CDR directly make use of the *HPD* component.

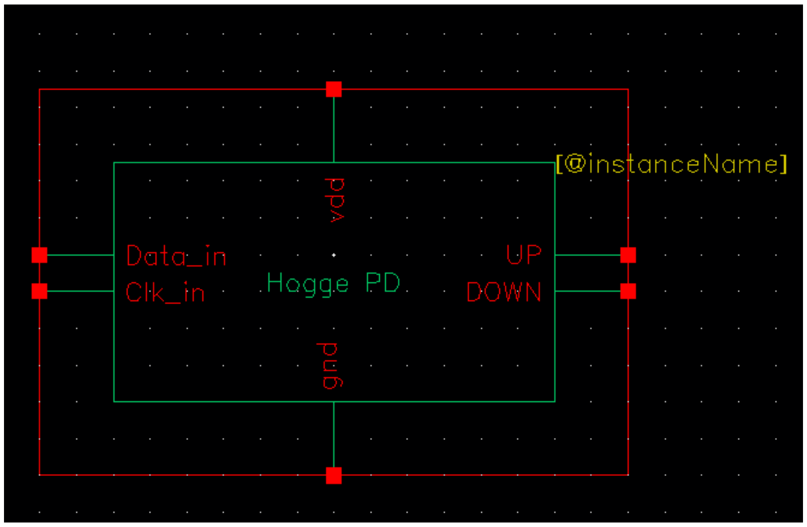


Figure 5.12: Symbol for Hogge PD

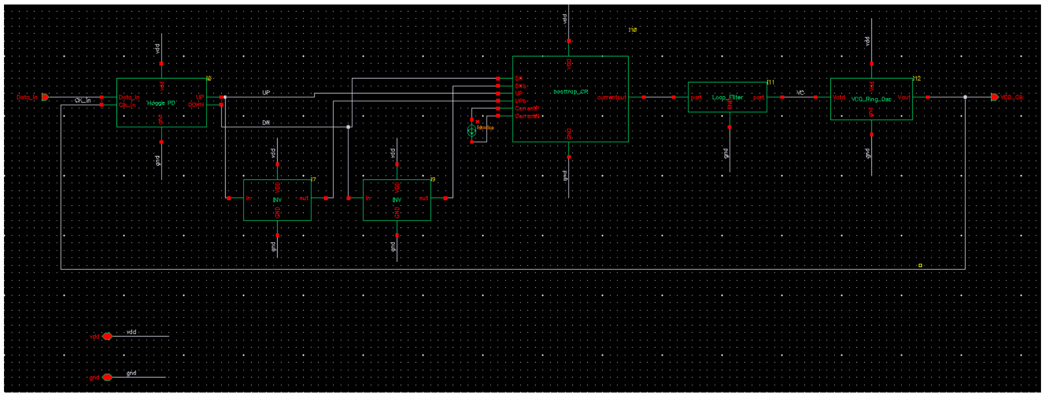


Figure 5.13: Complete CDR Loop

5.2.5 Creating a Testbench

A test bench allows the addition of the external circuit to test the component. Add the voltage power supply and ground connections as well as any external input sources.

Our final test bench for testing the CDR would be as shown in Figure 5.14:

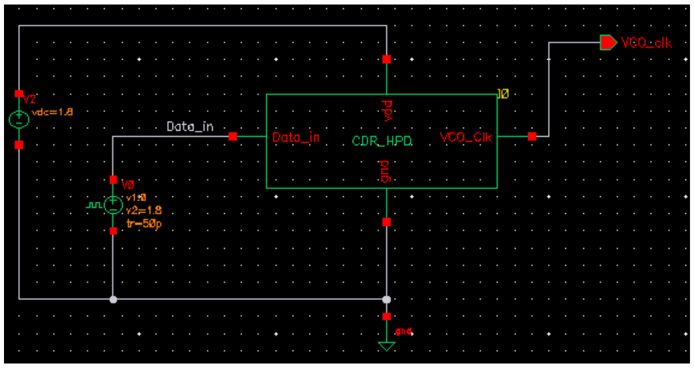
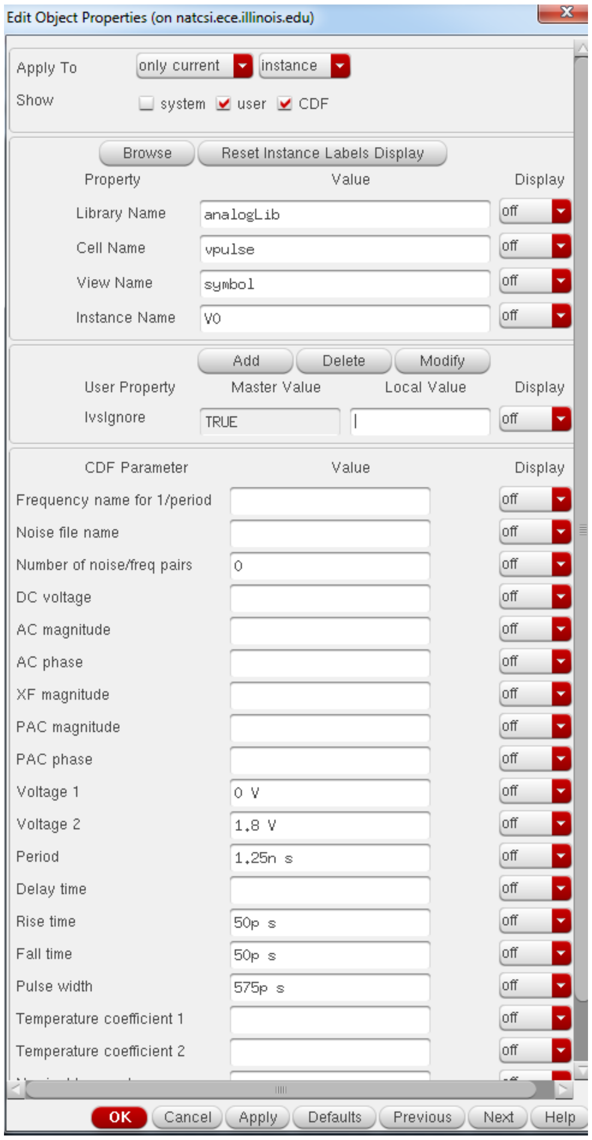


Figure 5.14: Test bench for CDR

The power suppy is component *vdc* found in analogLib and ground is *gnd* found in analogLib. The specs for the input data source are:


Edit Object Properties (on natcsi.ece.illinois.edu)

Apply To	only current	instance
Show	☐ system ✓ user ✓ CDF	

Property	Value	Display
Library Name	analogLib	off
Cell Name	vpulse	off
View Name	symbol	off
Instance Name	V0	off

User Property	Master Value	Local Value	Display
lvsIgnore	TRUE		off

CDF Parameter	Value	Display
Frequency name for 1/period		off
Noise file name		off
Number of noise/freq pairs	0	off
DC voltage		off
AC magnitude		off
AC phase		off
XF magnitude		off
PAC magnitude		off
PAC phase		off
Voltage 1	0 V	off
Voltage 2	1.8 V	off
Period	1.25n s	off
Delay time		off
Rise time	50p s	off
Fall time	50p s	off
Pulse width	575p s	off
Temperature coefficient 1		off
Temperature coefficient 2		off

Figure 5.15: Input Source Specifications

The data rate is 1.6Gbps. This means 1 bit period lasts for 625 ps making the period of the data cycle to be twice which is 1.25 ns.

5.2.6 Using Spectre for simulation

1. After the test bench is ready and clicking Check and Save, open ADE L from the Launch option. This opens up the simulator window.

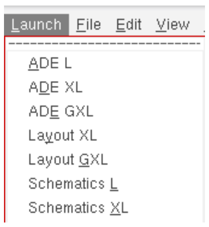


Figure 5.16: Launching the Analog Design Environment

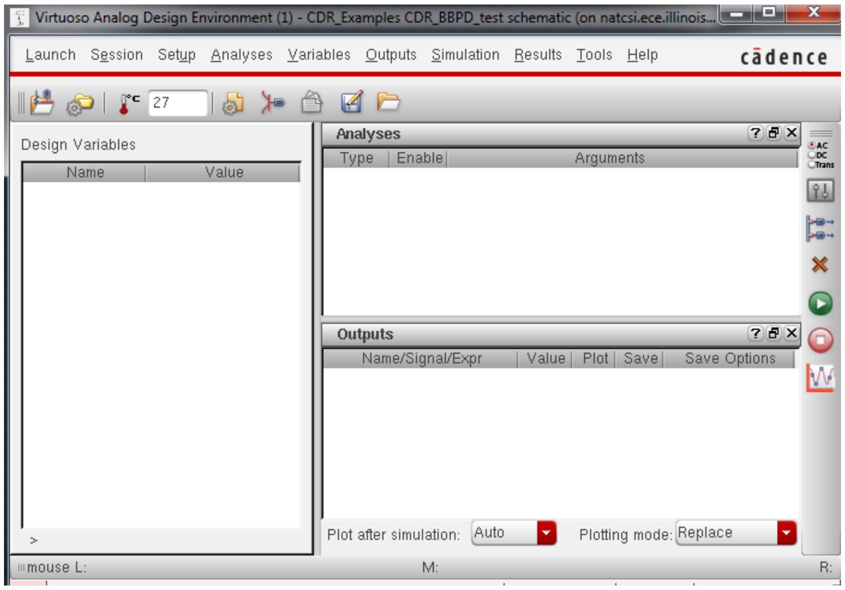
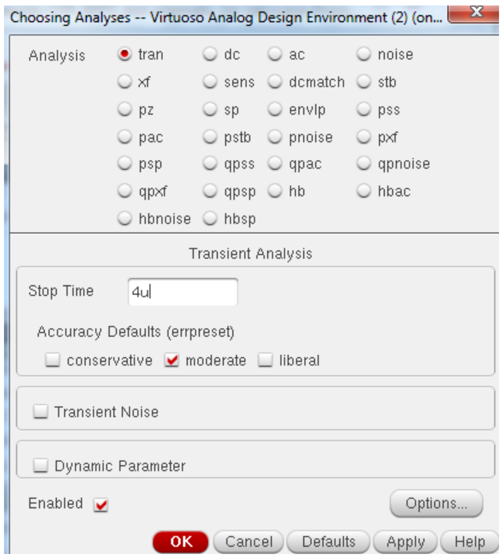

Virtuoso Analog Design Environment (1) - CDR_Examples CDR_BBPD_test schematic (on natcsi.ece.illinois...
Launch Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence
27
Design Variables
Name
Value
Analyses
Type
Enable
Arguments
Outputs
Name/Signal/Expr
Value
Plot
Save
Save Options
Plot after simulation:
Auto
Plotting mode:
Replace
mouse L:
M:
R:

Figure 5.17: Simulation state

2. Before starting the simulation, ensure that spectre is set as the simulator and that the model files are set or else the simulator will fail to recognize the components. Click Setup Simulator/Directory Host, check the simulator being used. It should be set to *Spectre*. For model files, click Setup Model Libraries. The window that opens gives an option to add the file path for the model library.

3. Once step 2 is completed, begin setting up the simulation. For assessing the time-domain behavior of the CDR use the transient analysis. In the ADE window, click on *Analyses* and choose *tran*. The other parameters are set as below. The Stop Time is the simulation time.


Choosing Analyses -- Virtuoso Analog Design Environment (2) (on...

Analysis: [✓ tran] dc ac noise xf sens dcmatch stb pz sp envlp pss pac pstb pnoise pxf psp qpss qpac qpnoise qpxf qpsp hb hbac hbnoise hbsp

Transient Analysis

Stop Time: 4u

Accuracy Defaults (errpreset): conservative [✓ moderate] liberal

Transient Noise

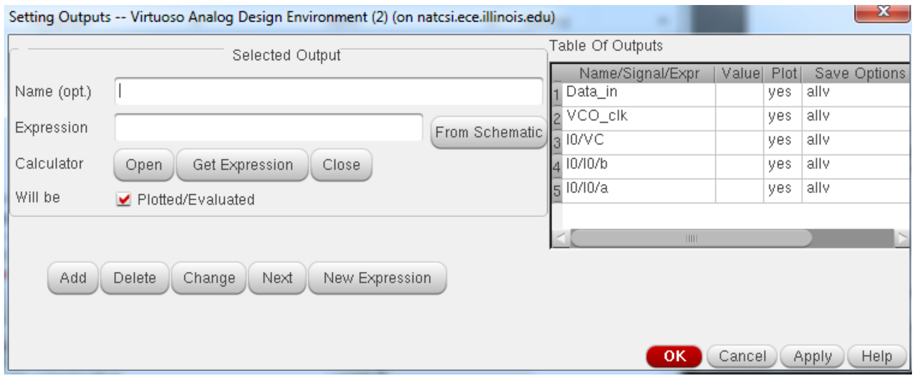
Dynamic Parameter

Enabled ✓

Figure 5.18: Choosing Analyses for simulation

4. Enter the variable by clicking on Variables Copy from Cellview . For this the data values should have variables in the schematic.

5. Set the outputs by either clicking on or by selecting *Outputs Setup*. This will open the following window,


Setting Outputs -- Virtuoso Analog Design Environment (2) (on natcsi.ece.illinois.edu)

Selected Output

Name (opt.):

Expression:

Will be: ✓ Plotted/Evaluated

Table Of Outputs

	Name/Signal/Expr	Value	Plot	Save Options
1	Data_in		yes	allv
2	VCO_clk		yes	allv
3	I0/VC		yes	allv
4	I0/I0/b		yes	allv
5	I0/I0/a		yes	allv

Figure 5.19: Selecting outputs

6. In this window enter the Outputs to be plotted. Use the *From Schematic* option which opens up the circuit schematic and select the outputs. Once the outputs are added, click OK. This will move back to the ADE window where the outputs will show up in the *Outputs* section. Check the plot and save option for outputs.

7. Once ready, click on the green Run arrow. This sets the simulation into motion. If the Output log reads *simulation successful* then wait for the simulation to complete. If there are any errors, the *Output Log* will display the error messages.

5.2.7 Transient Output

The simulator plots the checked outputs automatically in a new *Visualization* window. The output should be as shown below. Zoom into different sections by either using the buttons in the toolbar. One of the shortcut methods is to select an area of the waveform while keeping the right-mouse key pressed.

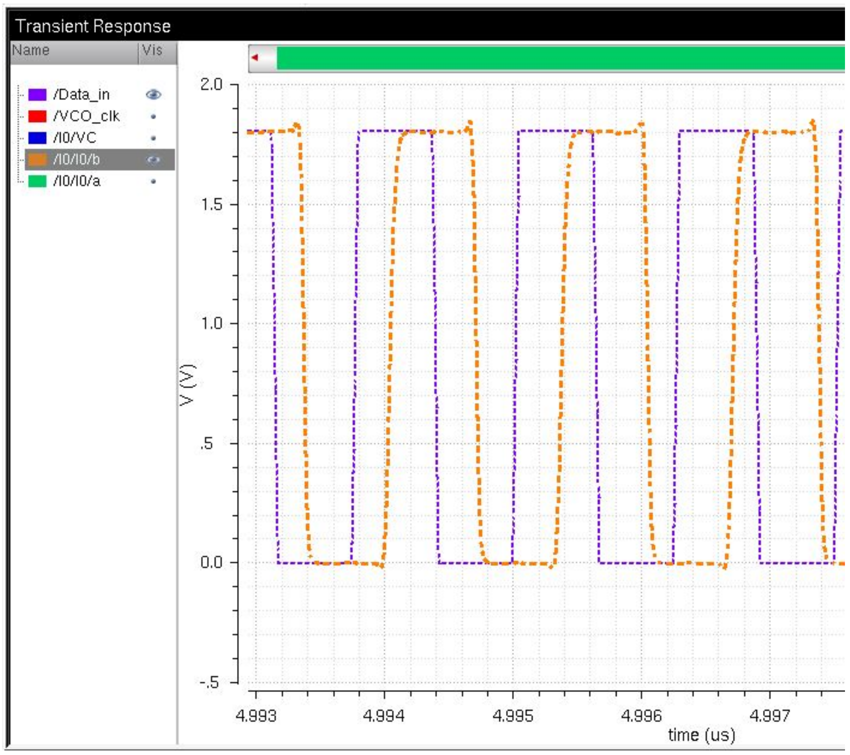


Figure 5.20: Transient Response

5.2.8 Waveform Inference

From the output waveform we can verify that the CDR is recovering the input data as the output follows the input data. This also implies that the

CDR has achieved lock.

Another way to ensure that the CDR has achieved lock is to plot the control voltage of the VCO and the VCO Output Clock. In the locked condition, the VCO should have a constant control voltage and the Output Clock should settle to a constant oscillation frequency.

5.2.9 Plotting VCO control Voltage and Output Frequency

1. For plotting VCO control voltage add the control voltage node as one of the outputs the same way as earlier. Descend from one level to another by selecting the block and pressing E. To ascend press Ctrl + E.

2. To plot the VCO frequency use the Calculator tool. The Calculator tool can be accessed from both the ADE and Visualizations window.

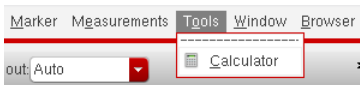
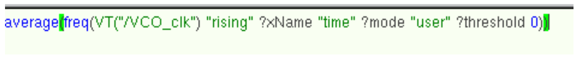


Figure 5.21: Opening the Calculator from the Visualizations window

3. Within the calculator clock on vt. This opens the CDR schematic and select the VCO Output

4. From the *functions* panel select freq and then average. Eventually the calculator expression should read like this.



```
average(freq(VT("/VCO_clk") "rising" ?xName "time" ?mode "user" ?threshold 0))
```

Figure 5.22: Calculator expression for VCO frequency

5. Now, go to the ADE simulation window Outputs Setup and in the *Setting Outputs* window click on *Get Expression*. This will pick up the expression in the calculator and copy it here. Now name the output as freq.

6. The gain of the VCO (K_{VCO}) can be plotted in the calculator the same way. For this, use the deriv function.

7. Go to *Tools* *Parametric Analysis*. Select control voltage (vcntrl) as the variable and sweep from 0 to 1.8 in linear steps of 0.1. Click on the Run button to start the parametric analysis.

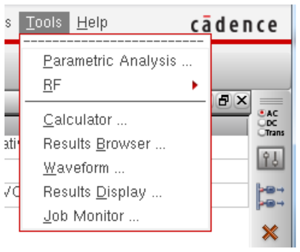


Figure 5.23: Starting Parametric Analysis

5.2.10 Simulation Output Waveforms

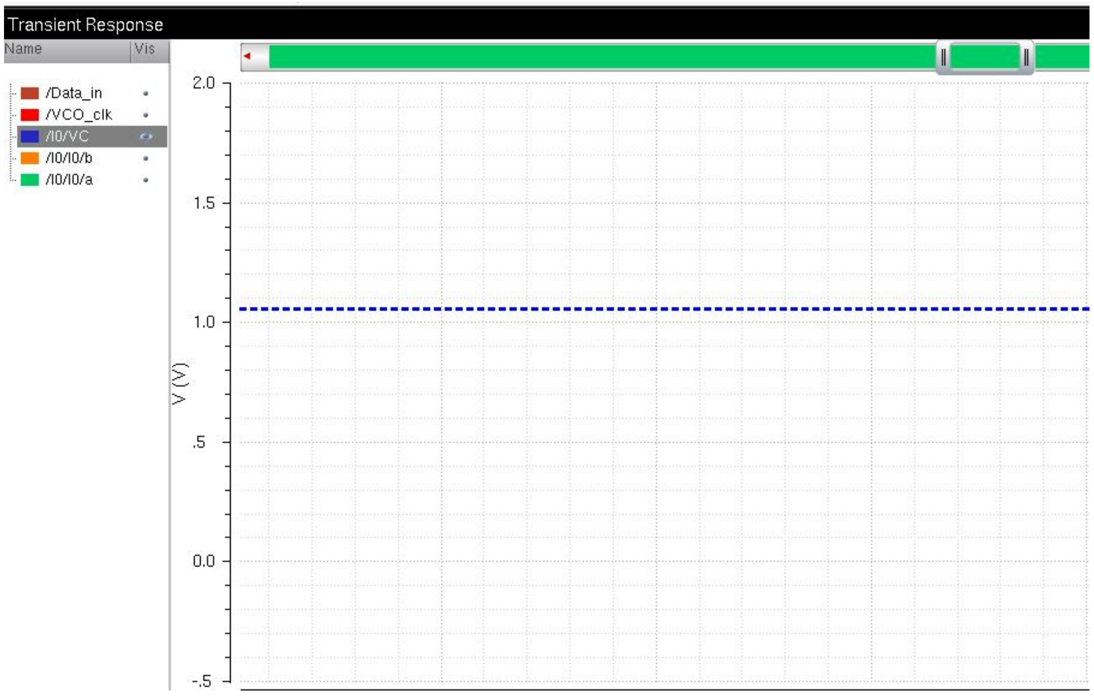


Figure 5.24: Plot of V_{cntrl}

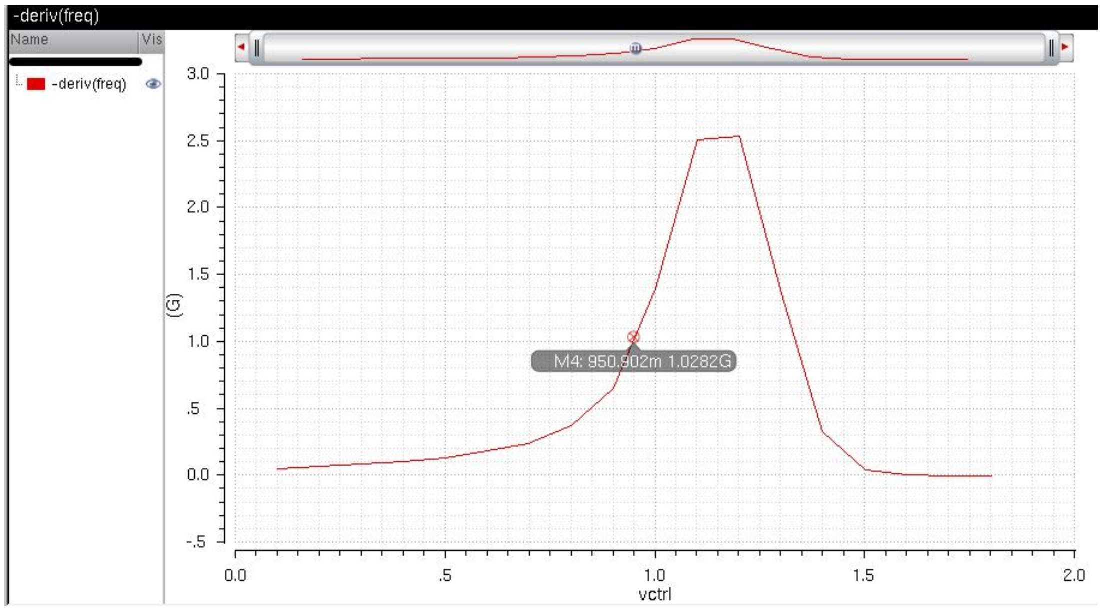


Figure 5.25: K_{VCO} Plot

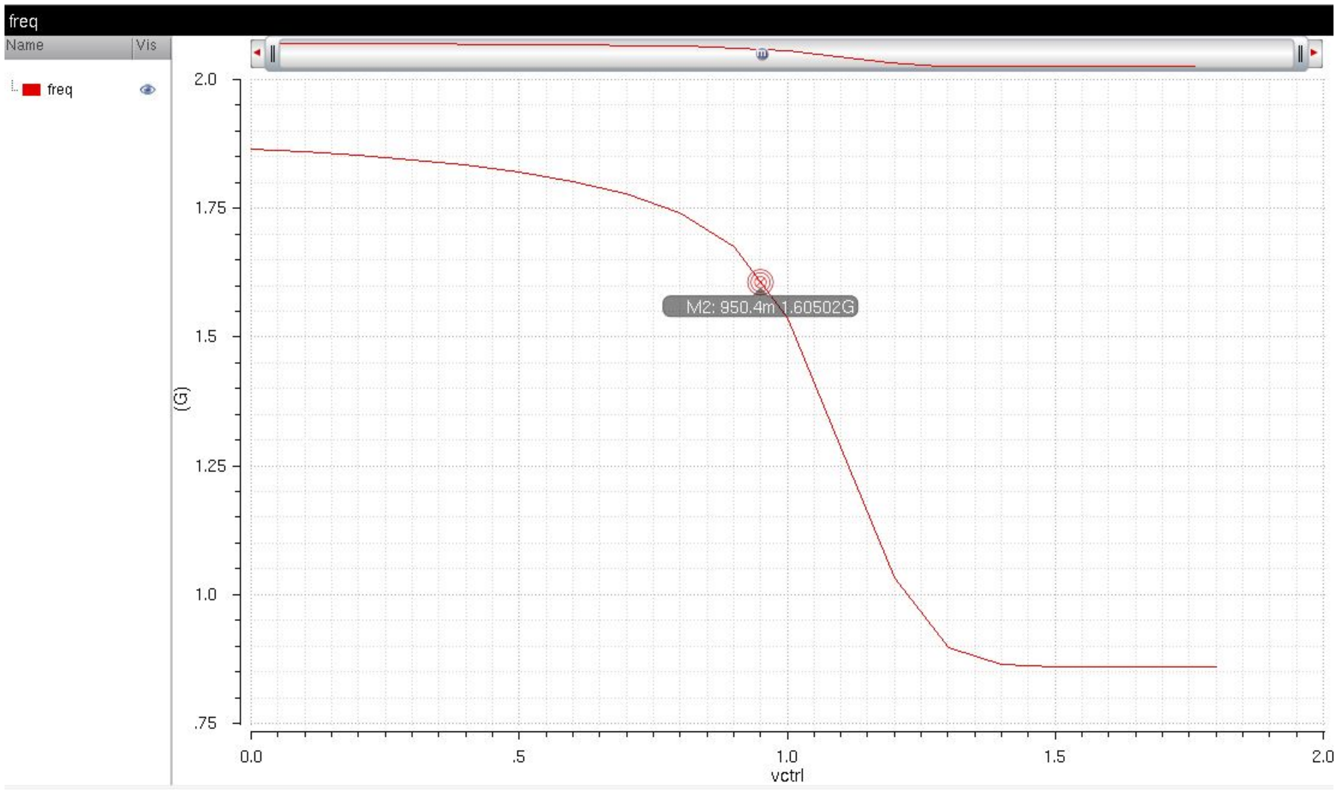


Figure 5.26: Frequency vs V_{cntrl}

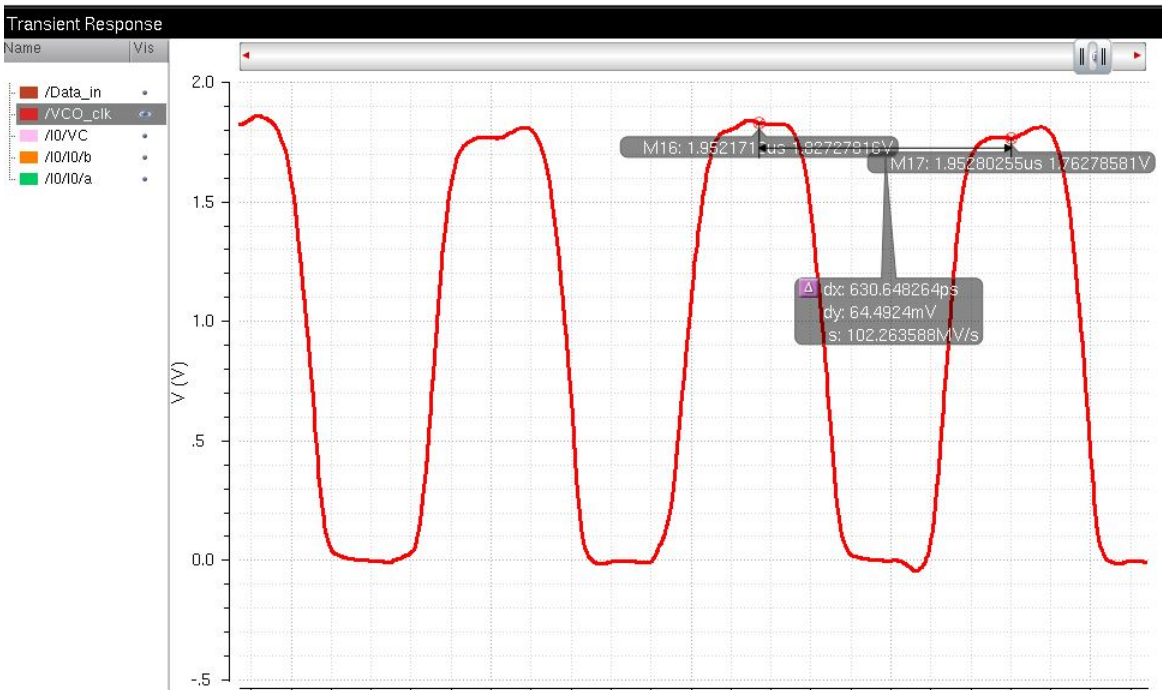


Figure 5.27: Frequency vs V_{cntrl}

CHAPTER 6

CONCLUSION

This thesis presented an introduction to the PLL-based clock and data recovery circuits. A Hogge-PD CDR was implemented at the transistor level. The thesis also presents a Cadence tutorial for the design and simulations. The CDR designed achieved lock with a VCO ouput Frequency of 1.6 GHz.

6.1 Design Improvements

Many researchers have proposed a wide variety of CDR architectures for high-speed applications. For example, some are based on a digital PLL, a Delay Lock Loop (DLL), a Phase Interpolator (PI), oversampling etc. The best choice would depend on the application and spec requirements. PLL-based CDR may include a frequency detection loop which helps prevent false locking. If the loop loses phase lock then the FD loop is activated which compares the frequencies of the input data and the VCO clock and drives the VCO towards lock. It also eliminates the need for an external reference clock. However, the two loops might interfere with each other causing failure to lock. Also, the two loops require different bandwidths. For this purpose, the system can be separated into two separate loops. This is the dual loop CDR architecture. This covers a large area. A digital PLL covers a small area. In this design, the loop filter and the charge pumps are all-digital components.

Table 6.1 presents a good comparison of the advantages and disadvantages of the different CDR architectures [8].

COMPARISON OF CDR ARCHITECTURES			
TYPE	PROS	CONS	APPLICATIONS
PLL	Input jitter rejection Input frequency tracking	Jitter Peaking Large LPF area (analog) Long acquisition time	SONET/SDH/Gigabit ethernet High speed serial link
DLL	No jitter peaking Stable/First Order system	Large LPF area (analog) Limited phase capturing range	Highspeed serial link Chip interconnects
PLL/DLL	No jitter peaking Small loop BW/Fast acquisition	Requires analysis for two loops	SONET/Ethernet/Fiber Multi-Gbps Link/Optical Receiver

Table 6.1: Comparision of CDR architectures

6.2 Future Work

The different components can be improved upon to have less noise, jitter and faster locking time at higher speeds. This would involve using different implementations such as the dual loop CDR, DLL and the digital CDR. The PLL based CDR offers good input jitter rejection but has stability issues. The DLL doesnt have jitter peaking or stability issues but has a limited phase capture range. Thus, a particular architecture should be chosen depending on the requirements.

A dual loop CDR would be a good step to begin with. It is one of the most commonly used architectures today.

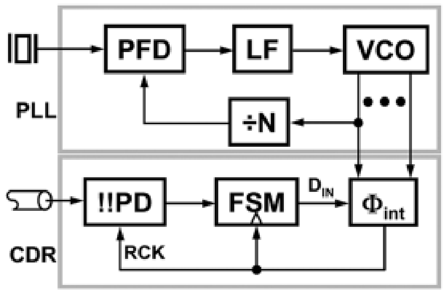


Figure 6.1: A Dual Loop CDR architecture

It primarily consists of two loops, a core PLL and a peripheral CDR loop. The PLL generates multiple phases of clock that are used by the phase interpolator to introduce a controlled phase shift in the recovered clock. The data recovery loop is based on a phase detector (PD), a finite state machine (FSM) and a phase interpolator (PI). The phase error output of the phase detector drives the finite state machine which controls the phase interpolator. The FSM filters the up and down signals. The main role of the PI is to combine the quadrature clocks and create different phases with fine steps. The CDR provides negative feedback which forces the recovered clock phase to align with the center of the data.

REFERENCES

[1] M. Assaad, “Design and modelling of clock and data recovery integrated circuit in 130 nm cmos technology for 10 gb/s serial data communications,” Ph.D. dissertation, Univ. of Glasgow, Glasgow, 2009. [Online]. Available: theses.gla.ac.uk/707/1/2009assaadphd.pdf

[2] S. Li, “Trade-offs in high-speed serial link ics,” *International Journal of High Speed Electronics and Systems*, 2005.

[3] S. Palermo, *CMOS Nanoelectronics Analog and RF VLSI Circuits, Chapter 9*. New York City, N.Y.: McGraw-Hill, 2011.

[4] B. Razavi, “Challenges in the design of high-speed clock and data recovery circuits,” 2002. [Online]. Available: http://www.designers-guide.org/Forum/Attachments/Challenges_in_the_design_high-speed_clock_and_data_recovery_circuits.pdf

[5] H.-S. Circuits and Y. U. Systems Lab, “High-speed serial interface.” [Online]. Available: http://tera.yonsei.ac.kr/class/2013_1_2/lecture/Lect14_CDR-1_ContinuousModeCDR.pdf

[6] O. G. A. E. I. T. A. B. Emre Salman, Hande Akn and Y. Gurbuz, “Design and verification of a pll based clock and data recovery circuit,” 2005. [Online]. Available: http://www.ece.sunysb.edu/~emre/papers/mms.pdf

[7] P. Hanumolu et al., “Analysis of charge-pump phase-locked loops,” *IEEE Transactions on Circuits and Systems-I*, vol. 51, no. 9, 2004.

[8] M. ta Hsieh and G. E. Sobelman, “Architectures for multi-gigabit wire-linked clock and data recovery.”