Summer 2014

Phase Locked Loop (PLL) based Clock and Data Recovery Circuits (CDR) using Calibrated Delay Flip Flop

Sagar Waghela
San Jose State University

PHASE LOCKED LOOP (PLL) - BASED CLOCK AND DATA RECOVERY CIRCUIT (CDR) USING CALIBRATED DELAY FLIP FLOP (DFF)

A Thesis

Presented to

The Faculty of the Department of Electrical Engineering

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Sagar Waghela

August 2014

The Designated Thesis Committee Approves the Thesis Titled

PHASE LOCKED LOOP (PLL) - BASED CLOCK AND DATA RECOVERY CIRCUIT (CDR) USING CALIBRATED DELAY FLIP FLOP (DFF)

by

Sagar Waghela

APPROVED FOR THE DEPARTMENT OF ELECTRICAL ENGINEERING

SAN JOSÉ STATE UNIVERSITY

August 2014

Dr. ShahabArdalan	Department of Electrical Engineering
Dr. SotoudehHamedi-Hagh	Department of Electrical Engineering
Dr. M. J. Zoroofchi	Department of Electrical Engineering

ABSTRACT

PHASE LOCKED LOOP (PLL) - BASED CLOCK AND DATA RECOVERY CIRCUIT (CDR) USING CALIBRATED DELAY FLIP FLOP (DFF)

by

Sagar Waghela

A Delay Flip Flop (DFF) is used in the phase detector circuit of the clock and data recovery circuit. A DFF consists of the three important timing parameters: setup time, hold time, and clock-to-output delay. These timing parameters play a vital role in designing a system at the transistor level. This thesis paper explains the impact of metastablity on the clock and data recovery (CDR) system and the importance of calibrating the DFF using a metastable circuit to improve a system's lock time and peak-to-peak jitter performance. The DFF was modeled in MATLAB Simulink software and calibrated by adjusting timing parameters. The CDR system was simulated in Simulink for three different cases: 1) equal setup and hold times, 2) setup time greater than the hold time, and 3) hold time greater than the setup time. The Simulink results were then compared with the Cadence simulation results, and it was observed that the calibration of DFF using a metastable circuit improved the CDR system's lock time and jitter tolerance performance. The overall power dissipation of the designed CDR system was 2.4 mW from a 1 volt supply voltage.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Dr. Shahab Ardalan for guiding me throughout this research with his invaluable knowledge. He was always supportive of my work since I began studying analog mixed signal courses at San Jose State University. I would also like to thank him for providing access to analog mixed signal (AMS) lab with the most recent simulation software tools that are used in the industry. Without his guidance, this research would have been incomplete.

I also would like to thank my committee members Dr. Sotoudeh Hamedi-Hagh and Dr. M. J. Zoroofchi for serving as my committee members. I thank them for their immense support and their helpful ideas throughout my education and research at San Jose State University.

A special thanks to my family. Words cannot express how grateful I am to my parents and siblings for always been supportive to my work, studies, and research. I am quite grateful to have all of you in my life. Without their encouragement, I would not have achieved this level of success thus far.

TABLES OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

s	Second
ns	Nanosecond
ps	Picosecond
V	Volt
mV	Millivolt
PP	Peak-to-peak
mW	Milliwatt
UI	Unit interval
Gbps	Gigabit per second
μA	Microampere
KHz	Kilohertz
MHz	Megahertz
GHz	Gigahertz
DSM	Delta-sigma modulator
KΩ	Kiloohm
nF	Nanofarad
pF	Picofarad
CMOS	Complementary metal oxide semiconductor
NMOS	N-type metal oxide semiconductor
PMOS	P-type metal oxide semiconductor

DC	Direct current
deg	Degree
SDFF	Semi dynamic delay flip flop
XOR	Exclusive-OR gate
PD	Phase detector
CP	Charge pump
VCO	Voltage controlled oscillator
LC	Inductor-capacitor oscillator

Chapter 1. **Introduction**

In wire-linked communication systems, when data flows over a single wire without any accompanying clock, the receiver of the system is required to process this data synchronously. Therefore, the CDR circuits are used in the receiver of the system to recover the clock or timing information from these data. Data bandwidth for wire-linked communication systems is also increasing at a high rate. In 2007, according to the International Technology Roadmap for Semiconductors (ITRS), the non-return to zero (NRZ) data rate for high-performance differential pair point-to-point nets on the package would reach 100 Gbps by the year 2019 as shown in Figure 1.1 [1]. In such high-speed wire-linked communication systems, these data are corrupted both by internal and external noise during its passage from transmitter to receiver, resulting in jitter and skew in the data received at the receiver. Here, the clock and data recovery circuit is necessary to extract the data transmitted by the transmitter from the corrupted received signal and also to recover the accompany clock timing information at the receiver side of the communication systems.

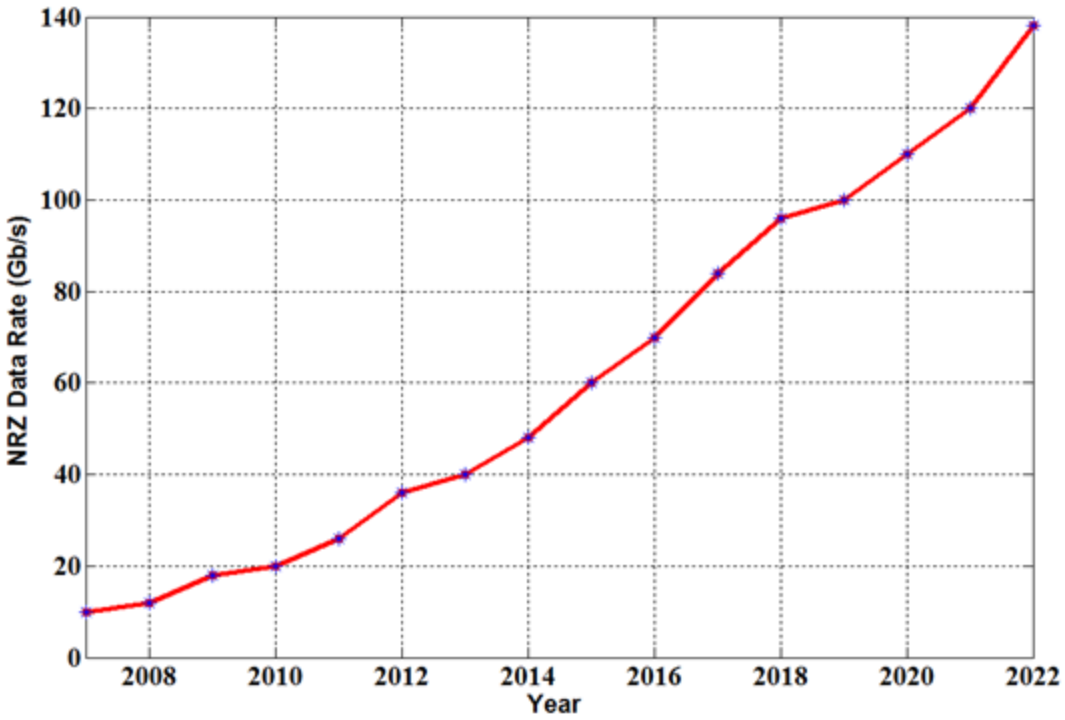

Figure 1.1: NRZ data rate for high performance differential pair point-to-point nets on a package, based on the ITRS 2007 roadmap prediction [1].

A block diagram of a high-speed wire-linked source asynchronous communication system is shown in Figure 1.2. In a source asynchronous system, the transmitter and receiver use different clock sources of the same frequency. As seen from Figure 1.2, the received data are first equalized in the receiver input buffer and then fed to the CDR circuit for retiming before proceeding into the deserializer module. Hence, there exists a frequency offset between the transmitted data and the local clock on the receiver side due to natural device mismatches, which creates the challenges for CDR circuit designers.

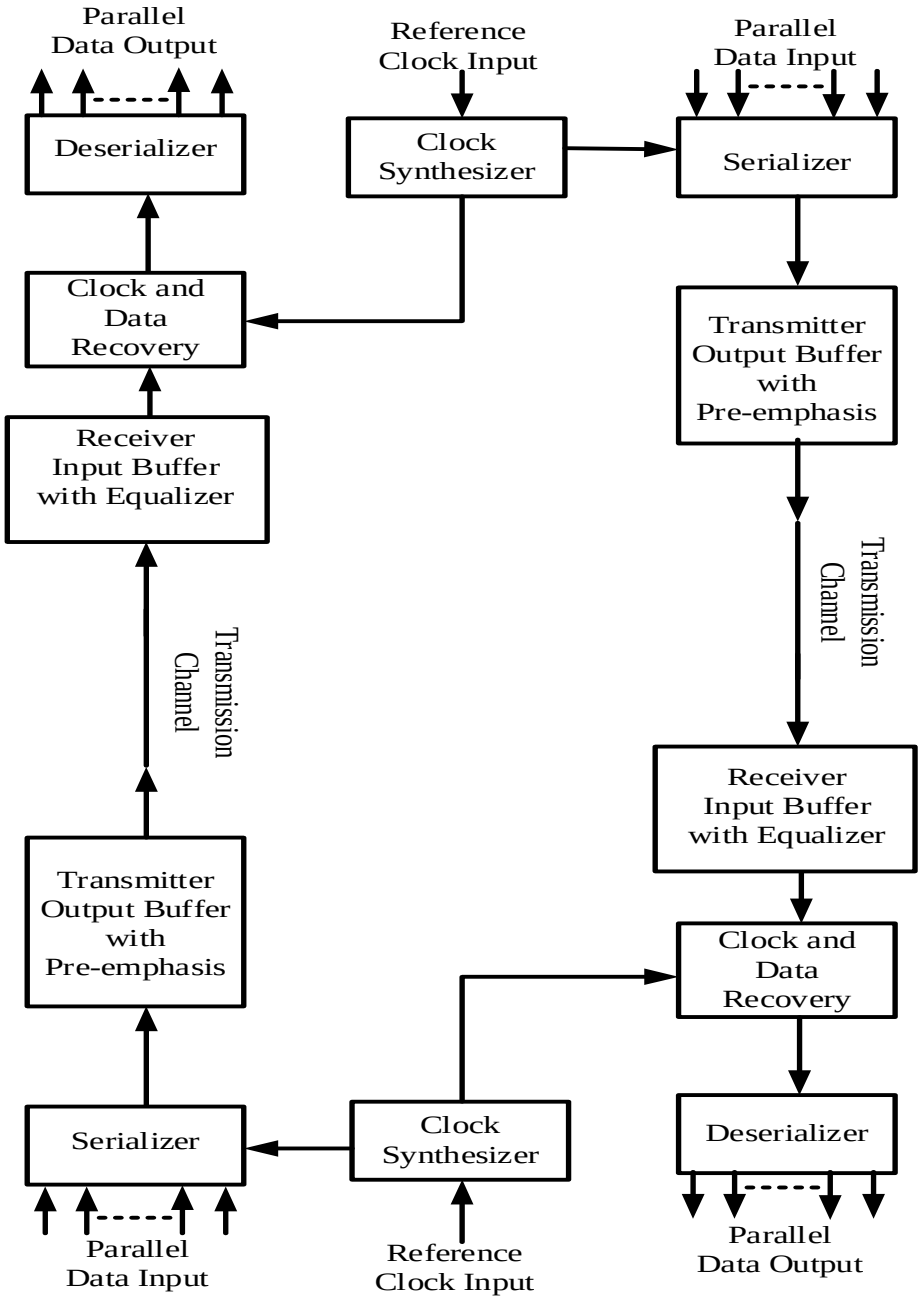

Figure 1.2: Block diagram of a generic high-speed wire-linked source asynchronous communication system.

1.1 CDR

The basic building blocks of a CDR circuit include a clock recovery and data retiming blocks as shown in Figure 1.3. The function of the clock recovery circuit is to detect the transitions in the received data and generate a periodic recovered clock. This recovered clock must satisfy the following conditions:

- The recovered clock's frequency must be equal to the input data rate.
- The recovered clock should have reasonable timing with respect to the input data (i.e., the rising edge of the recovered clock should sample at the center of the data bit, to provide maximum margin for jitter and other time uncertainties).
- The recovered clock should exhibit a minimum jitter because the jitter of the clock contributes to the retimed data jitter.

The data retiming circuit uses a Delay Flip Flop (DFF), which is triggered by the recovered clock to retime the received data. The DFF samples the corrupted received data and regenerates the data with less jitter and skew [2].

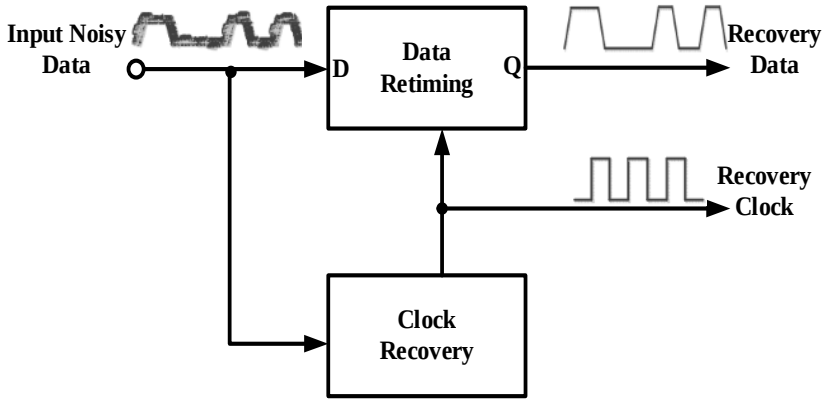

Figure 1.3: Basic block diagram of the clock and data recovery system.

1.2 Motivation and Agenda

This thesis presents the effect of metastability on a CDR system and the importance of calibration of a DFF using a metastable circuit to improve CDR system's lock time and jitter tolerance performance. The metastability effect refers to a violation of setup and hold time requirements of a DFF. A DFF samples input data on an active-edge of a clock and this sometimes occurs at a data transition point, providing an incorrect result at the DFF output. Thus, there is a need to calibrate (i.e, delay or advance) the active edge of the clock using a metastable circuit to satisfy the setup and hold time requirements for the DFF. Once calibrated, the input data gets sampled at the center of the data bit interval or bit transmission time, providing a correct result at the DFF output.

Chapter 2 provides a general background on a CDR system and explains each block of the system in detail. Chapter 3 explains the design of the CDR system using Simulink software. The CDR system with and without the metastable DFF was simulated for four different cases. Chapter 4 explains the CDR design at transistor level using 45 nm technology and modeling using Verilog A language in Cadence Virtuoso 6.1.5. This chapter also explains the design of a DFF, an Alexander phase detector, and a metastable circuit at the transistor level. The designed CDR system was simulated for three different cases and the results were compared with that obtained by using Simulink.

Chapter 2. **Background**

In source asynchronous communication systems, data are transmitted by the transmitter without an accompanying clock and the receiver has to process these data at their end synchronously, recovering the clock from the data. A phase lock loop (PLL) is used at the receiver to recover the clock from the data.

2.1 PLL

A PLL is a negative feedback system where a clock generated by the voltage control oscillator is phase and frequency locked to an input data. The basic topology of PLL based CDR is shown in Figure 2.1.

Figure 2.1 shows basic elements of the PLL based CDR.

- Pre-Amplifier and Limiter
- Phase Detector (PD)
- Charge-Pump (CP) and Low Pass Filter (LPF)
- Voltage Controlled Oscillator (VCO)

The function of the Pre-Amplifier and Limiter is to generate a full voltage swing from the input data, required by the phase detector. The functions of the PD, CP and LPF, and VCO are discussed in following sections.

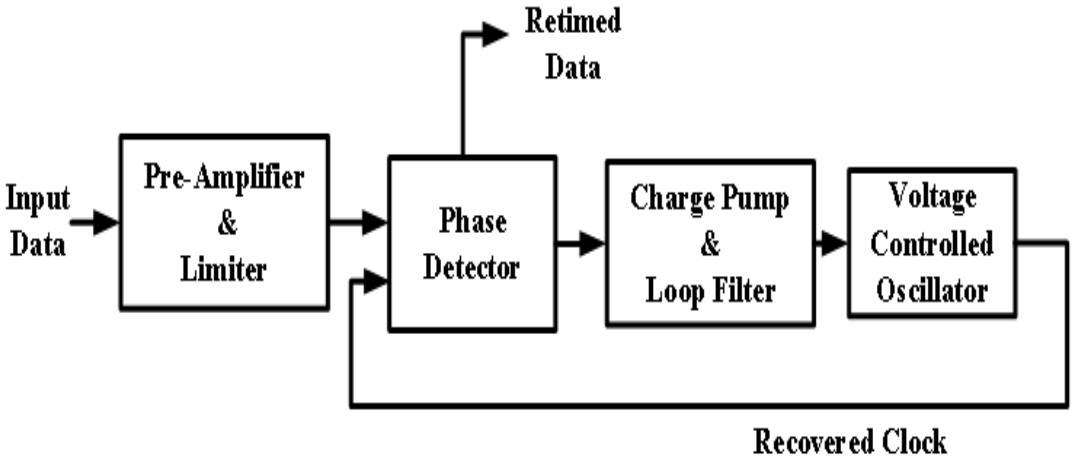

Figure 2.1: PLL based CDR system.

2.1.1 Phase Detector (PD)

The function of the phase detector is to measure the phase difference between two incoming signals. Examples are clock and data signals, data and data signals, and clock and pseudo random bit data (PRBS) signals. Various topologies and designs for phase detectors already exist, such as the Alexander Phase Detector, the Hogge Phase Detector, the Quad-rate Phase Detector, the Octant-rate Phase Detector, etc.

Phase detectors are broadly classified into two classes: linear and binary phase detectors. Linear phase detectors (PD) are used in low to medium speed CDR applications, operating at few hundreds of megahertz (MHz) speed. Binary phase detectors are used in high-speed CDR applications operating at hundreds of gigahertz (GHz) speed.

In the case of linear phase detectors, the output of the phase detector is linearly proportional to the phase difference between two input signals as shown in the characteristics of the linear phase detector in Figure 2.2.

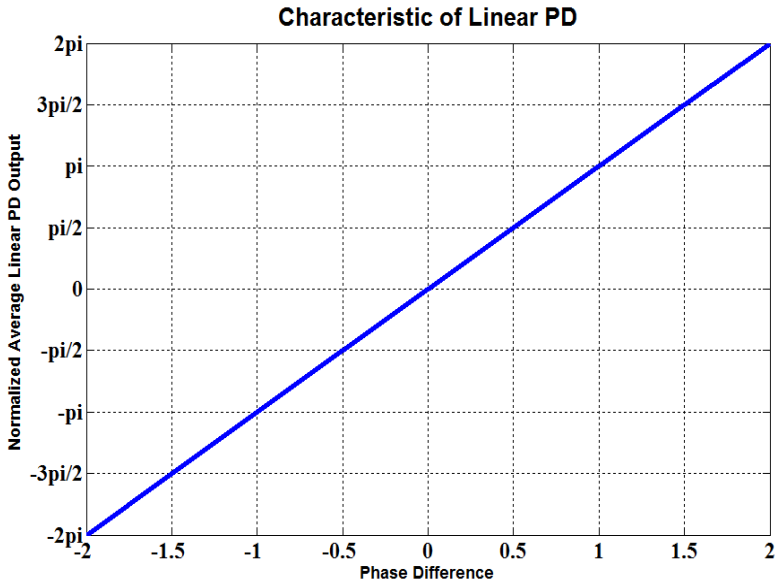

Figure 2.2: Characteristic of a linear PD.

The slope of the line is called phase detector gain and is calculated by following equation.

$$K_{pd} = \frac{V_{pd}}{\Delta\phi} \tag{2.1}$$

In the above equation, phase detector gain is defined as K_{pd}, the average phase detector output is defined as V_{pd}, and the difference between two input signals is defined

as $\Delta\phi$. As the phase difference between two input signals increases, the average phase detector output also increases. Hence, the phase detector gain remains constant [2]. One DFF with an XOR gate is enough to satisfy the requirement of linear phase detector, but as the average value of the phase detector output is a function of the data transition density of the input, this design fails to uniquely represent the phase difference for various data patterns as shown in Figure 2.3, thus, this design is data pattern dependent [2].

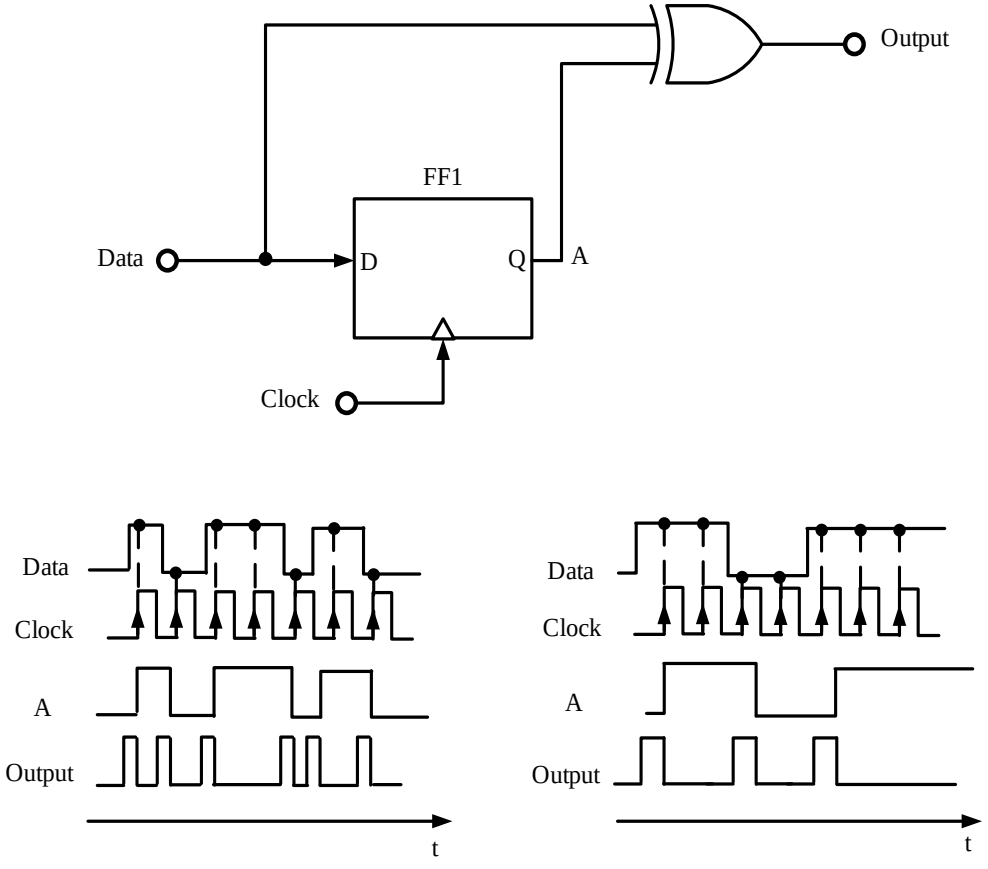

Figure 2.3: Basic linear phase detector and output waveforms.

One example of a linear phase detector is the Hogge Phase Detector. The circuit implementation and output waveform is shown in Figure 2.4. The Hogge Phase Detector consists of two DFFs and two Exclusive OR (XOR) gates. The function of a DFF is to produce a delayed replica of the input signal at its output. The first DFF, named FF1, produces a delayed replica of the input data at the rising edge of the clock and is then XORed with input data. The output of the XOR gate, named X, gives the phase difference between two input signals. To avoid the problem of data pattern dependency, the proportional pulses obtained at node X are accompanied by reference pulses at node Y, which are generated by using an additional DFF (FF2) and XOR gate. The reference pulses appear on the data edge and have constant pulse width, thus avoiding the pattern dependency, as shown in Figure 2.4.

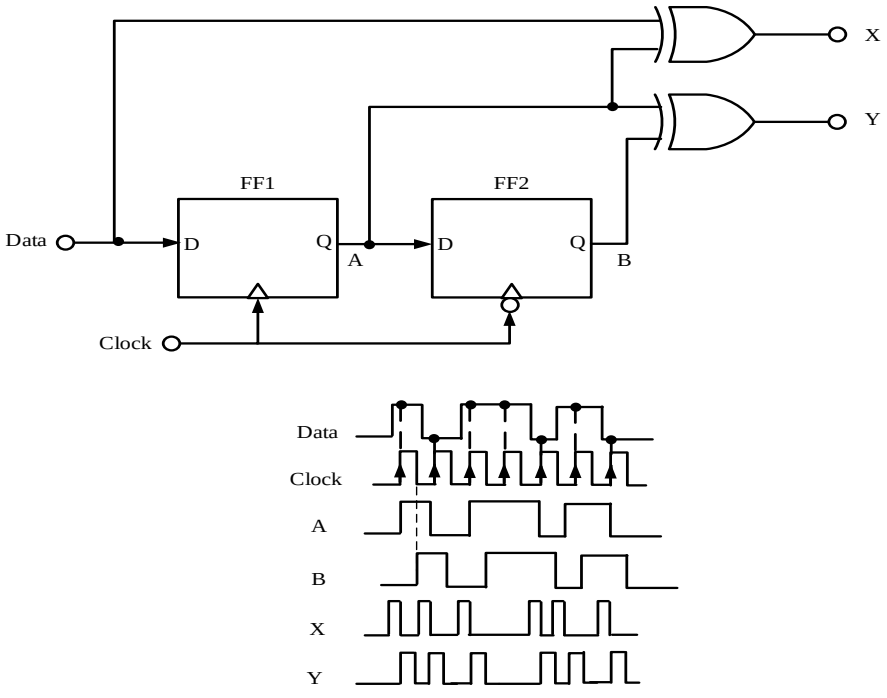

Figure 2.4: Hogge Phase Detector and its output waveform.

In binary phase detectors, the output is either logic one or zero. One example of a binary phase detector is the Alexander Phase Detector. The Alexander Phase Detector accepts two input signals (e.g. clock and data) and determines whether the clock is earlier or later than the data. If the clock is earlier than the data, the early node goes to logic one and the late node goes to logic zero. Otherwise, when the clock is later than the data, the late node goes to logic one and early node goes to logic zero. A more detailed explanation of the Alexander Phase Detector is presented in Section 3.1.1.

2.1.2 Charge Pump (CP) and Low Pass Filter (LPF)

The function of the charge pump is to convert the output voltage of the phase detector to current. This current is then fed to a low pass filter, where the capacitor is either charged or discharged depending on the phase detector output. The circuit diagram of the charge pump with a Type-I LPF (capacitor) is shown in Figure 2.5 and a Type-II LPF is shown in Figure 2.6.

In this research, the Alexander Phase Detector is used, where the output is either early or late. The early and late nodes are connected to respective switches of the charge pump circuit, as shown in Figure 2.5. When the early node is high, closing the early switch, the capacitor starts charging and continues to charge until the early node goes low, opening the early switch. Similarly, when the late node goes high, the capacitor starts to discharge and will continue to discharge until the late node goes low.

Designing a charge pump is not an easy task, because to achieve zero net voltage on the capacitor, the charging current should be equal to the discharging current. Even if the

charging and discharging currents are designed to be close to equal, there will still be leakage current through the charge pump circuit, resulting in an offset voltage on the capacitor. One way to minimize this offset voltage is to calibrate the charge pump circuit by using a feedback loop circuitry.

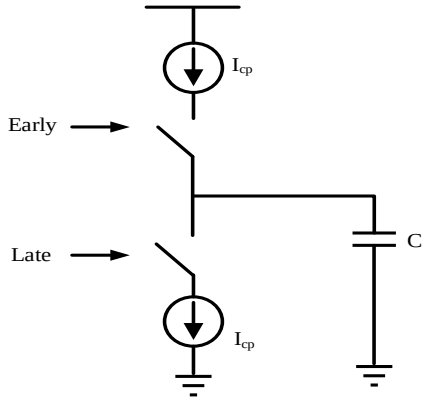

Figure 2.5: Charge pump with a Type-I LPF.

$$I_c = C\frac{dV}{dt} \tag{2.2}$$

$$V_c = \int \frac{I_c}{C}\, dt \tag{2.3}$$

The function of the low pass filter (LPF) is to convert the charge pump current into control voltage. The Type-I LPF is replaced by Type-II LPF due to trade-offs between the settling time, ripple on the control voltage, and the phase error and stability. To minimize the ripples on the control voltage, the capacitor from Figure 2.5 is replaced by

the resistor (R) in series with the capacitor (C_1), both in parallel with the capacitor (C_2), as shown in Figure 2.6. If the capacitor (C_2) is five to ten times less then capacitor (C_1), then the Type-II LPF will still approximately behave as a Type-I LPF [3].

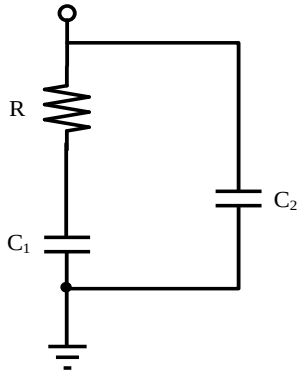

Figure 2.6: Circuit diagram of the Type-II Low Pass Filter.

2.1.3 Voltage Controlled Oscillator (VCO)

The function of the voltage control oscillator is to generate the clock signal at its output, the frequency of which can be changed by varying the input control voltage [4]. Oscillators have wide applications in communication system ranging from clock generation in microprocessors to carrier synthesis in cellular telephones [2].

The circuit to oscillate at ω_0, it must satisfy two conditions as shown by equation (2.4) and they are known as Barkhausen criteria.

$$|H(j\omega_o)| \geq 1 \text{ and } \angle|H(j\omega_o)| = 180^o \qquad (2.4)$$

Two types of CMOS oscillators used widely in today's technology are ring oscillators and inductor - capacitor (LC) oscillators. A ring oscillator consists of an odd number of gain stages in a loop as shown in Figure 2.8 and the bode plot of three stage ring oscillator is shown in Figure 2.7.

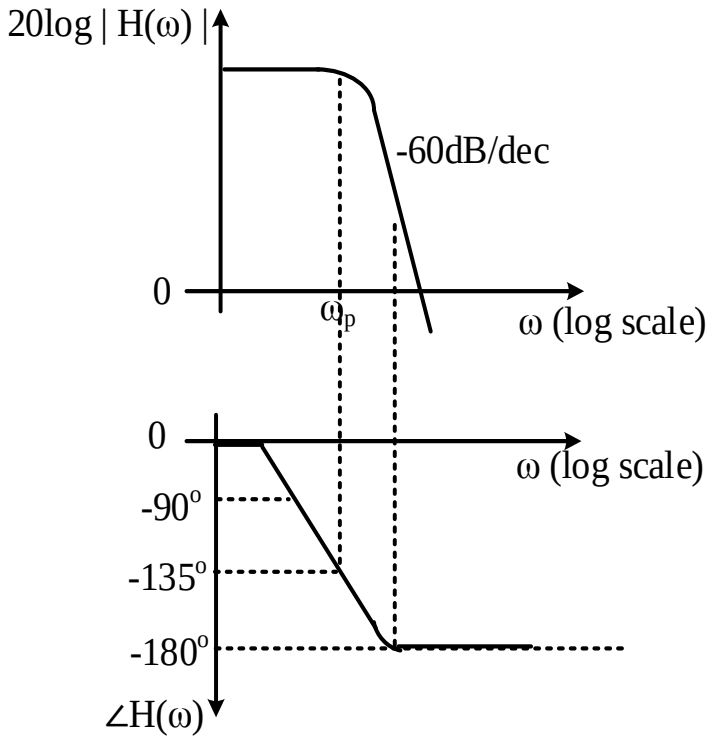

Figure 2.7: Bode plot of three stage ring oscillator.

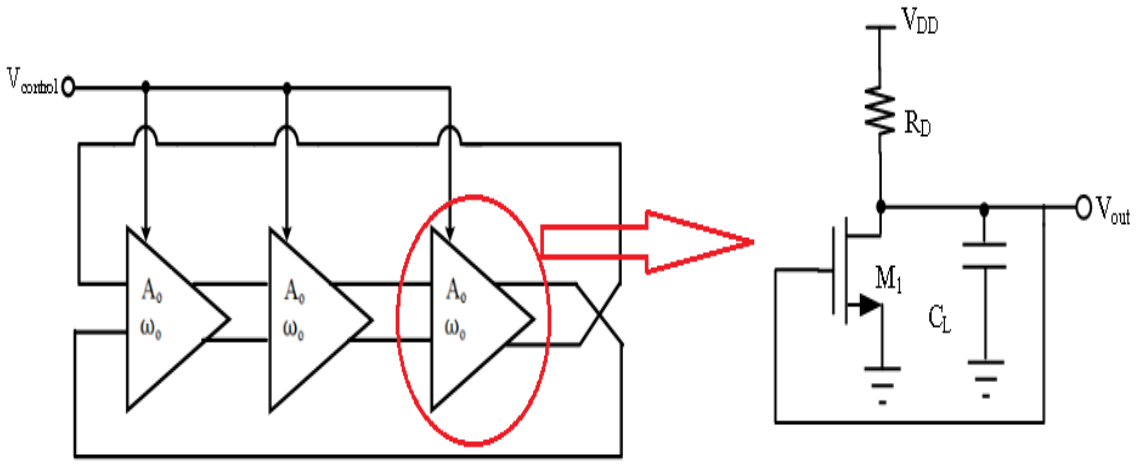

Figure 2.8: Three-stage ring voltage controlled oscillator.

The LC oscillator consists of the cross-coupled common source stages loaded by the inductor (L) placed in parallel with the capacitor (C) as shown in Figure 2.9.

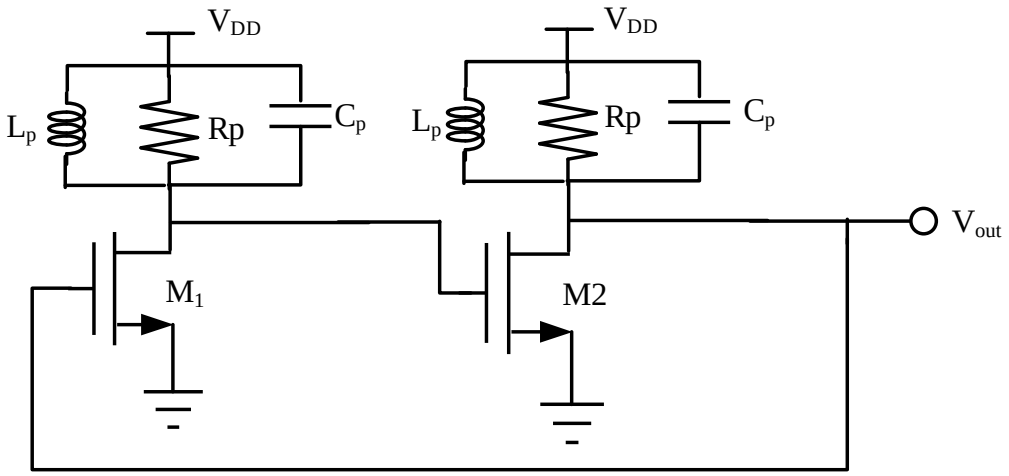

Figure 2.9: Cross coupled LC oscillator.

The cross-coupled LC voltage controlled oscillator is given by Figure 2.10 [5].

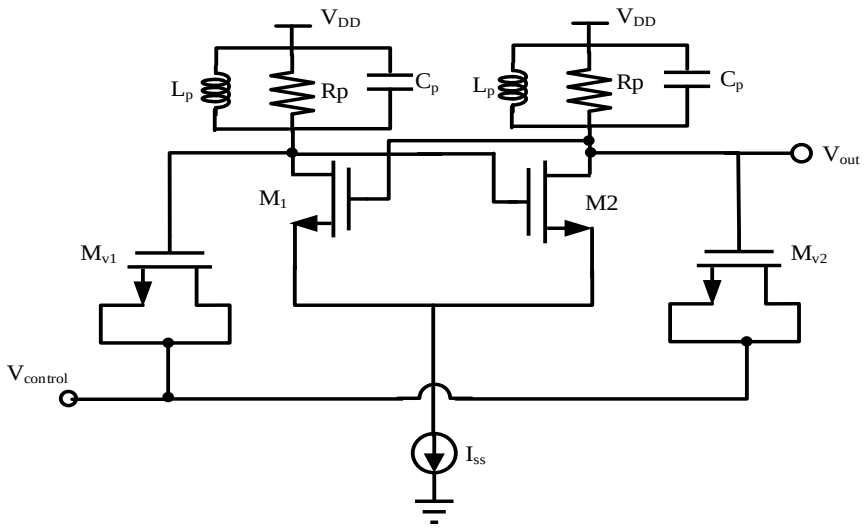

Figure 2.10: Cross coupled LC voltage controlled oscillator [5].

There are many oscillator specifications that one must be aware of. The ring oscillators have a high frequency tuning range and small area consumption whereas the LC oscillator has a limited frequency tuning range and large area consumption comparatively. LC oscillators produce less jitter on the recovered clock in CDR as compared to the ring oscillators. Jitter, another oscillator specification, is described in more detail in the next section. Ring oscillators are preferred over LC oscillators in implementing VCO due to their attractive features of frequency tuning range and area consumption. But, the ring VCO provides low quality factor [6]. Thus, the choice of selecting either of the oscillators depends on the application.

2.2 Jitter in CDR Circuits

Jitter is defined as the amount of variation in the waveform from their ideal position at zero crossing on the time axis. The optical communication (OC) standards for the CDR circuits in terms of jitter are more stringent and difficult to achieve. The jitter is expressed in terms of the bit period or unit interval (UI) by OC standards. For example, a jitter of 0.0001 UI refers to 0.1% of the bit period. Jitter in the CDR circuit is characterized in three terms:

- Jitter transfer
- Jitter generation
- Jitter tolerance

The jitter transfer function in CDR circuits is output jitter when the input jitter is changed at different rates. If the input jitter varies slowly, then the output of the CDR circuits will track the input to ensure phase locking; however, if the input jitter varies at a fast rate, then the output will track the input to a lesser extent, (i.e., the CDR circuit must filter the input jitter). Thus, the jitter transfer function has the same characteristic as that of a low pass filter as shown in Figure 2.11. The OC standards have two specifications for jitter transfer:

1. The bandwidth of CDR circuits should be approximately 120 kHz in OC-192, (i.e., CDR must suppress the jitter components above 120 kHz).
2. The jitter peaking shown in Figure 2.11 must be less than 0.1dB.

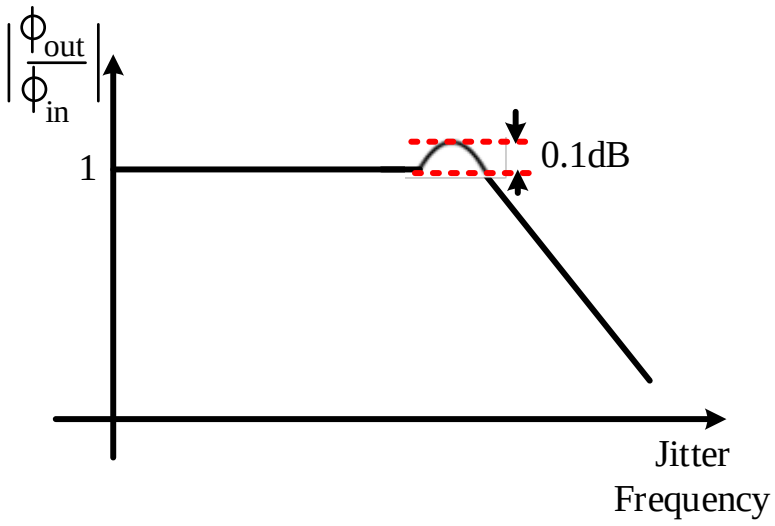

Figure 2.11: Jitter transfer function.

Multiple data regenerators are placed along a signal's path to minimize the non-ideal effects of an optical fiber cable in communication networks. Therefore, each data regenerator should have small jitter bandwidth, in order to minimize the accumulated jitter through the chain of data regenerators. Furthermore, as the total jitter transfer function of the data regenerators connected in series is given by the product of the individual data regenerator's jitter transfer function, it is important to have small jitter peaking per data regenerator [7].

Jitter generation is defined as the jitter produced by CDR circuit elements when the input data signal itself does not have jitter. The major sources of jitter generation are:

- Phase noise error in the VCO caused by the noise generated by VCO.

- The ripples on the control voltage due to leakage current from the charge pump circuit.
- The coupling of the input data transition to the VCO through a retiming circuit and phase detector.
- Noise generated by the power supply and substrate.

The jitter in the output of the VCO caused by the noise generated is shown in Figure 2.12.

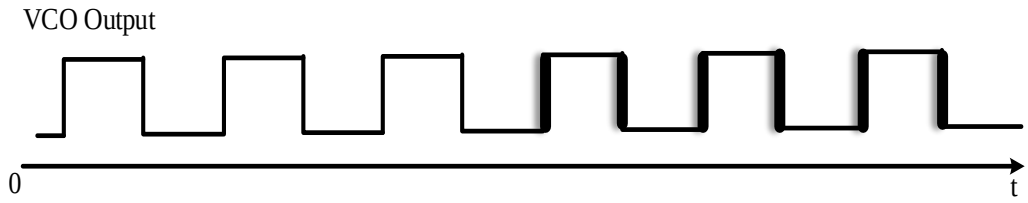

Figure 2.12: Noise in VCO output.

Jitter tolerance is defined as the amount of jitter that the CDR circuit must tolerate on the input data without increasing the bit error rate (BER). If the jitter on the input data varies slowly, the recovered clock will track the transition in the data and always sample the data in the middle of the bit period as shown in Figure 2.13. This will guarantee a low BER. Whereas, if the jitter on the input data varies fast, the recovered clock will not be able to track the transition in the data and will fail to sample the data in the middle of the bit period as shown in Figure 2.14. This will result in a greater BER.

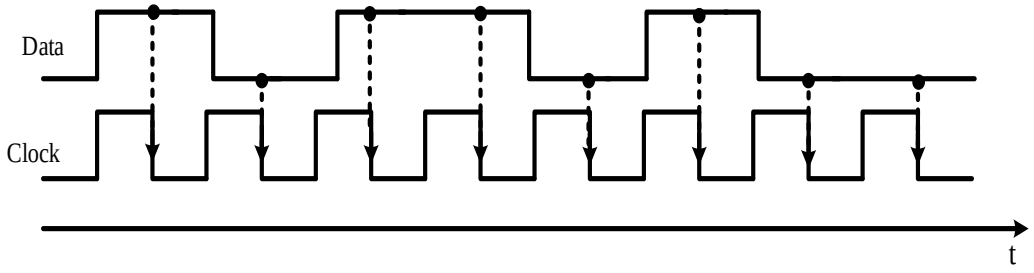

Figure 2.13: Slow jitter on data retiming.

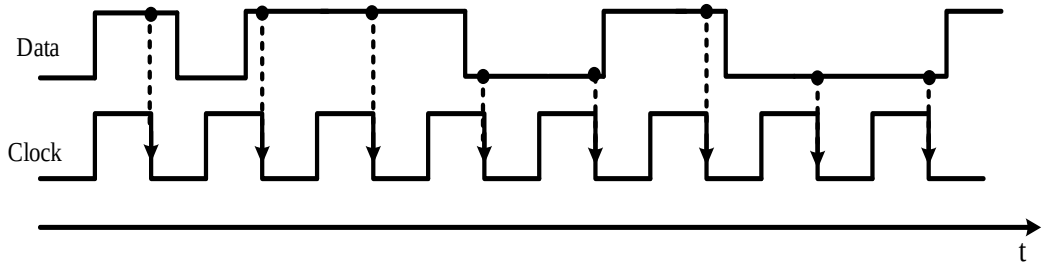

Figure 2.14: Fast jitter on data retiming.

The specification of the jitter tolerance is described by a mask, which is the function of the input jitter frequency, shown in Figure 2.15. Jitter tolerances for the various OC/SONET standards are shown in Table 2.1 [8]. For example, the CDR circuit must tolerate a peak-to-peak jitter of 1.5UI, if the jitter on the input signal varies at the rate below 6 kHz.

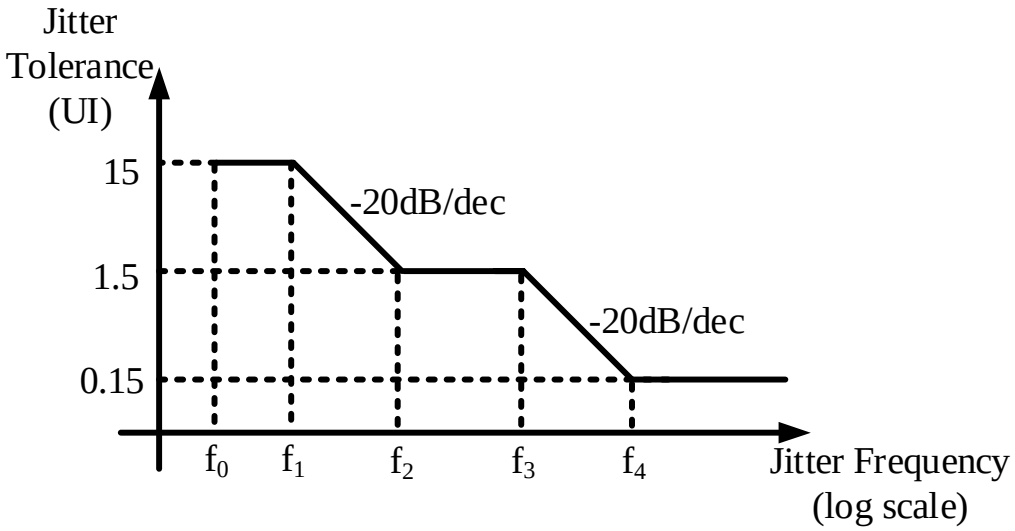

Figure 2.15: Jitter tolerance mask.

Table 2.1: SONET specifications [8].

Data Rate	**f_0(Hz)**	**f_1(Hz)**	**f_2(Hz)**	**f_3(kHz)**	**f_4(kHz)**
OC-3 155Mb	10	30	300	6.5	65
OC-12 622Mb	10	30	300	25	250
OC-48 2.48Gb	10	600	6000	100	1000
OC-192 10Gb	10	2400	24000	400	4000

2.3 Metastability Concept

The concept of metastability is best understood after understanding the basics of delay flip flop (DFF) and its timing parameters. This research has used the Alexander phase detector in the designed CDR circuit, which consists of four DFFs and two XOR

gates. The phase detector is considered as the heart of the CDR circuit. The function of the DFF is to produce the delayed replica of the input signal. The DFF can be implemented using only static circuits, only dynamic circuits, or a combination of static and dynamic circuits.

The static implementation stores the data in the form of a charge on parasitic capacitors associated with the MOSFET for an extended period of time, thus, causing an increase in the leakage current. In contrast to the dynamic implementation that stores the data in a similar manner but for a short period of time in the range of milliseconds, resulting in the reduction of leakage current [10]. Due to this advantage, dynamic implementation is preferred over a static one. Dynamic circuits result in significantly higher performance and lower power dissipation making them useful in designing high performance systems [11].

The type of DFF used in this research is the semi-dynamic DFF (SDFF), shown in Figure 2.16, which was first introduced by Klass [9]. The SDFF consists of both static and dynamic circuits. The main reasons for choosing this type of DFF over the others are short latency, small clock load, small area, and a single-phase clock scheme [9]. An additional feature of the SDFF is that it can incorporate various logic functions with small penalties in delay and transistor count.

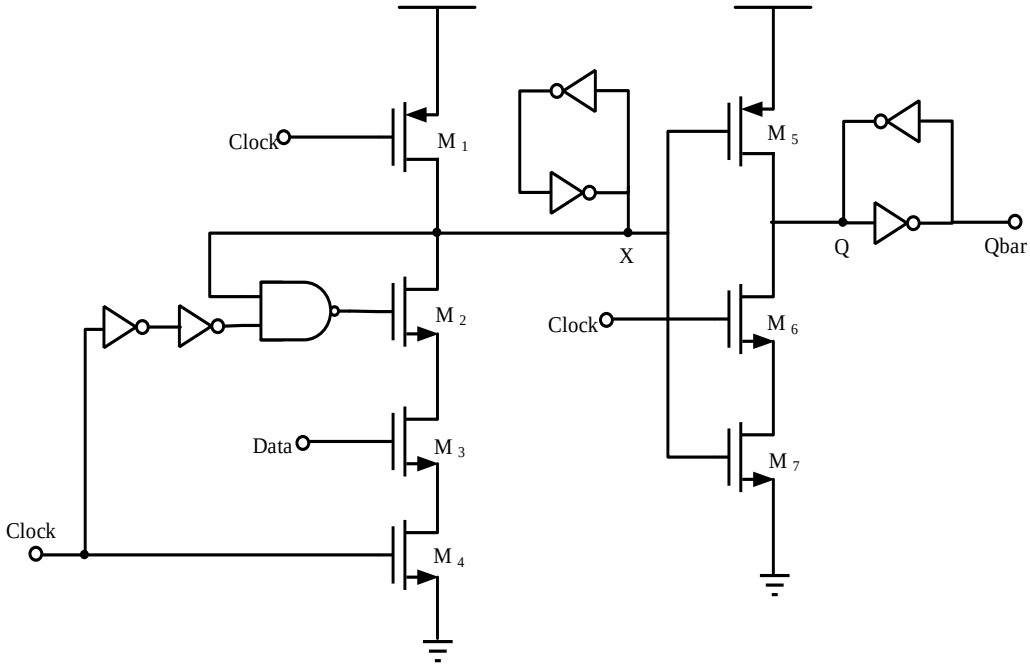

Figure 2.16: Schematic of Semi-dynamic DFF (SDFF).

2.3.1 Features of SDFF

- The SDFF has short latency because the SDFF is refreshed periodically by the clock signal, thus, keeping the SDFF in an idle state for very short period of time.
- The SDFF implemented in this research has a small clock load of five transistors by using a clock CMOS (C^2MOS) approach with the penalty of increasing overall transistor count. The C^2MOS approach provides the advantage of implementing logic functions. The SDFF can also be implemented by using transmission gates to reduce the overall transistor

count, but the drawback with such implementation is a high clock load of about six transistors [10].

- The SDFF using transmission gates or C^2MOS approach can be implemented by using either a true single-phase clocking scheme (TSPC) or a two-phase clocking scheme. The drawback with a two-phase clocking scheme is the overlapping between clock and delayed clock, thus, providing a direct path between the input and output signal, destroying the state of the circuit [10]. To avoid this, a TSPC scheme is used, which uses a single clock to drive the entire circuit.

Thus by taking in account the above features, the SDFF is implemented using a C^2MOS approach with a TSPC scheme.

2.3.2 Timing Parameters

There are three timing parameters associated with DFF as shown in Figure 2.17 and are explained below:

- **Setup Time (T_s):** Defined as the minimum time interval between the rising edge of the clock and that of the input data signal, such that the input is reliably sampled.

- **Hold Time (T_h):** Defined as the minimum time interval between the falling edge of the input data signal and the rising edge of the clock such that the sampled data remains stable throughout that clock period.

- **Clock to output delay ($T_{c\text{-}q}$):** Defined as the time interval between the rising edge of the clock and the rising edge of the output signal when the input data is reliably sampled.

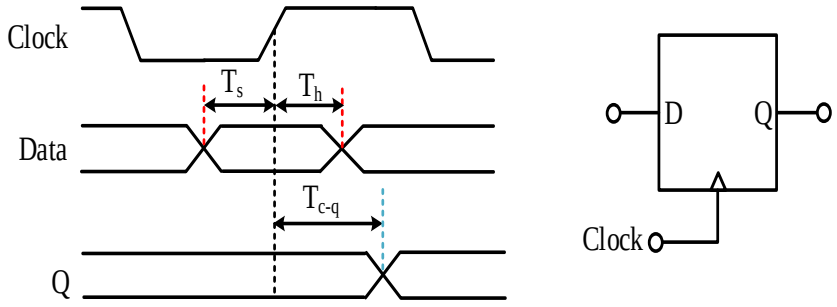

Figure 2.17: Timing parameters of the DFF.

If a violation of the setup or hold time takes place, then the DFF output is not a guaranteed sample of the input data, possibly leading to the wrong logic level. This happens due to the input data not having enough time to toggle between high and low signal logic levels. Thus, the data goes into an idle state due to the setup and hold time violations and will hold the value obtained from previous successful sampling event.

Figure 2.18 shows the sampling of the input data by the rising edge of the clock in two scenarios. In the first scenario, DFF1 samples the input data by the rising edge of the clock during the data transition. In this case, the setup and hold time requirements are violated; thus the result at the DFF1's output will be an incorrect logic level. In the second scenario, DFF2 samples the input data by the rising edge of the clock at the

middle of the data bit interval. In this case, there is no violation of timing parameters, thus the result at the DFF2's output will be a correct logic level. For the system to perform efficiently, the active edge of the clock should sample the input data at the middle of the data bit interval to allow maximum margin for setup and hold time(s). This idea is referred to as a metastable concept in this research and results in increasing the system's jitter and skew tolerance.

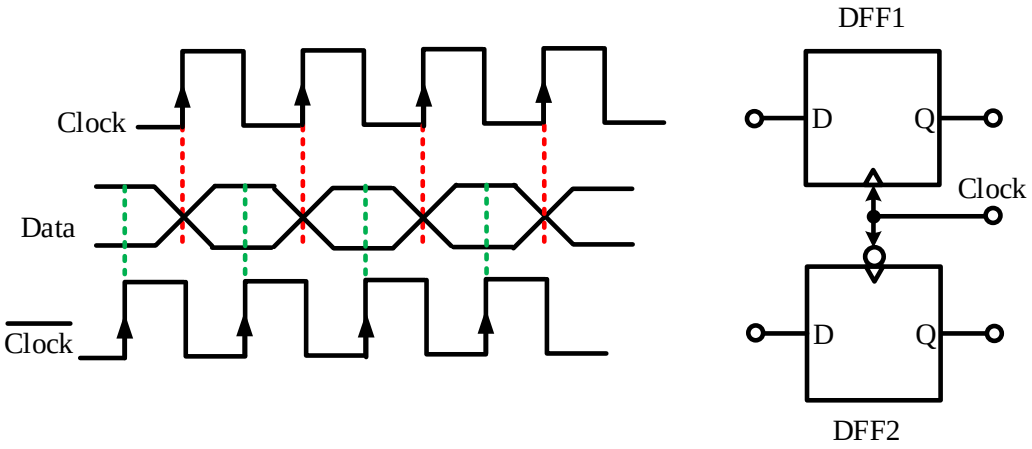

Figure 2.18: Sampling points for DFF1 and DFF2.

In this research, the metastable circuit was designed and the clock delay cell from the metastable circuit was placed in the CDR system, such that the clock always samples the data at the middle of the input data bit interval. The metastable circuit is explained in detail in section 4.5. This circuit first converts the input data into glitch data, whose width or metastable window is equal to the sum of setup (T_s) and hold (T_h) times, shown in Figure 2.19.

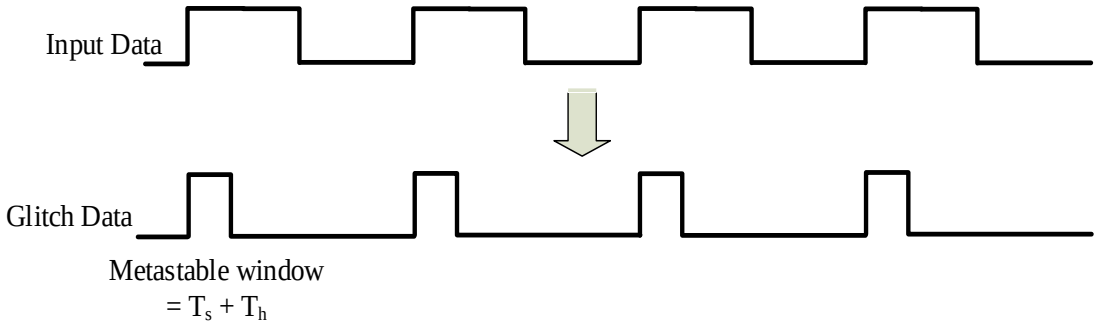

Figure 2.19: Glitch data generation from the input data.

The metastable window's rising edge (left leg) and falling edge (right leg) each can be varied or aligned using three digital bits, with a total of six digital bits to vary the width of metastable window as shown in Figure 2.20.

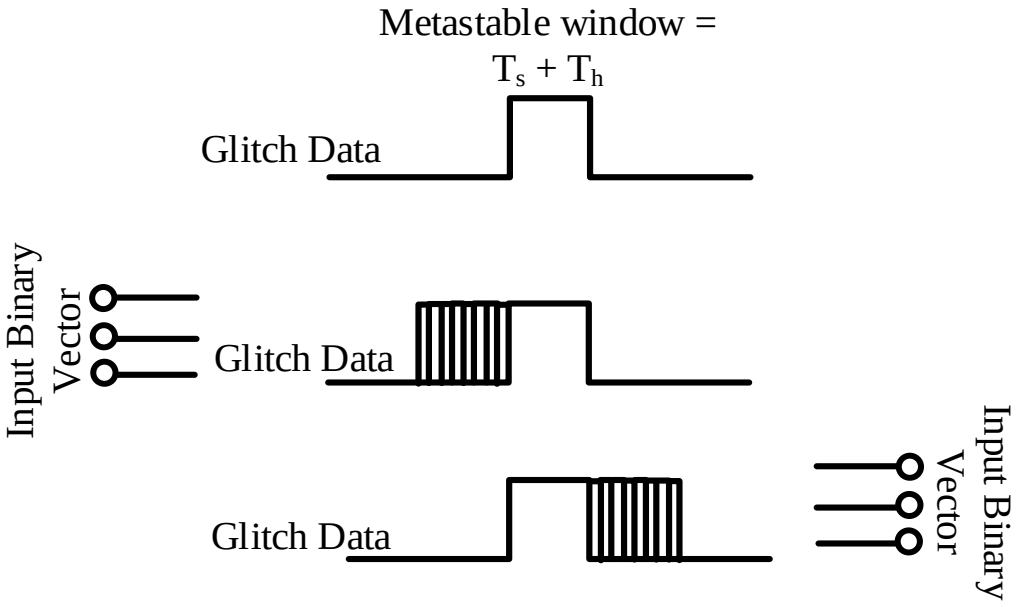

Figure 2.20: Variation in metastable window of glitch data using six digital bits.

The rising edge of the clock is aligned at the center of the metastable window by using a five-bit variable clock delay cell as shown in Figure 2.21. Thus, the clock delay cell from the metastable circuit is placed in the designed CDR system.

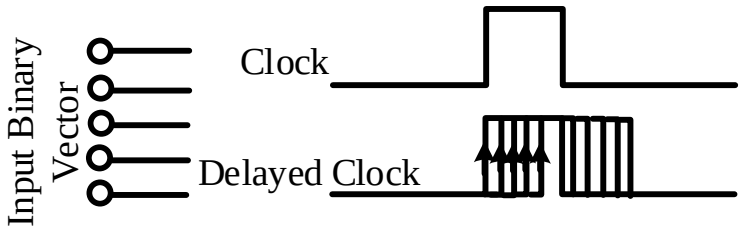

Figure 2.21: Variation of the clock using five digital bits.

Chapter 3. **CDR MATLAB and Simulink Models**

The CDR system is modeled using MATLAB and Simulink software as shown in Figure 3.1. The modeled CDR system has the input data rate of 3 Gbps and the voltage controlled oscillator is operating at 3 GHz. Thus, the full-rate phase detector's architecture is used in modeling the Alexander phase detector.

The Simulink CDR system consists of the following models:

- PRBS-7 data generator
- Alexander phase detector
- Alexander phase detector with metastable DFF
- Charge pump
- Low-pass filter
- Voltage controlled oscillator

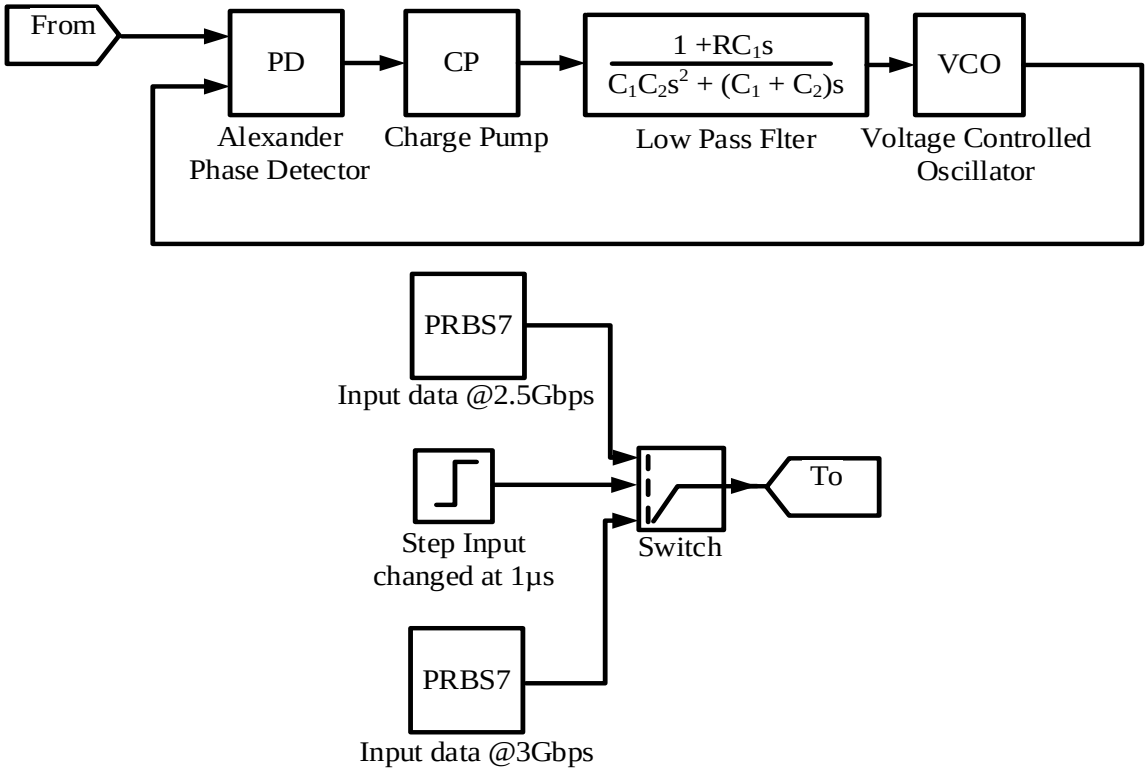

Figure 3.1: Implementation of the CDR system using Simulink.

3.1 PRBS-7 Data Generator

Pseudo random binary sequences (PRBSs) are widely used in communication systems. A PRBS is a random bit sequence that repeats itself. PRBSs are used in testing hardware circuits that are used in communication systems. PRBSs are generated by shifting bits through the number (n) of cascaded shift registers. Some of the shift register's output are added with a modulo-2 function and fed to the input of the first shift register [12]. The PRBS-7 consists of seven shift registers as shown in Figure 3.2 and produces a data sequence of length 127 bits.

There are various types of PRBS data generators like PRBS-7, PRBS-9, PRBS-15, PRBS- 31, etc., and are used depending on application. Table 3.1 presents the properties of various types of PRBS data generators.

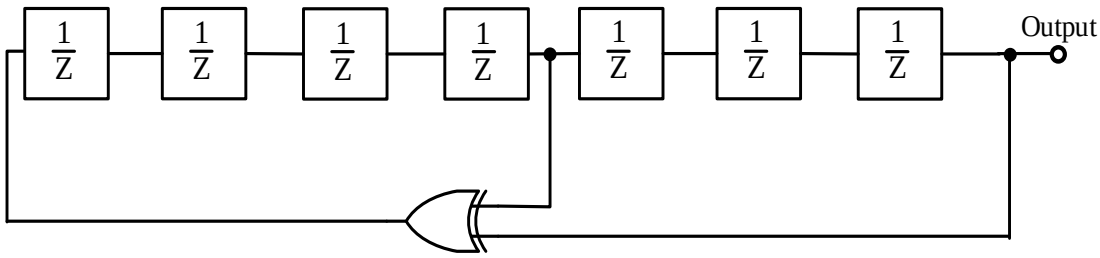

Figure 3.2: Simulink model of PRBS data.

Table 3.1: Properties of PRBS data generators.

Shift Register Length (n)	**Characteristic Polynomial**	**PRBS Length 2^n-1**	**No. of 1's 2^{n-1}**	**No. of 0's 2^{n-1}-1**
7	$X^7 + X^6 + 1$	127	64	63
9	$X^9 + X^5 + 1$	511	256	255
11	$X^{11} + X^9 + 1$	2,047	1024	1023
15	$X^{15} + X^{14} + 1$	32,767	16,384	16,383
23	$X^{23} + X^{18} + 1$	8,388,607	4,194,304	4,194,303
29	$X^{29} + X^{27} + 1$	536,870,911	268,435,456	268,435,455
31	$X^{31} + X^{28} + 1$	2,147,483,647	1,073,741,824	1,073,741,823

3.1.1 Alexander phase detector (PD)

The Alexander PD is a binary phase detector and provides the inherent data retiming for the CDR system [2]. The Alexander PD is used in the high-speed CDR circuits that operate at GHz speed. The Alexander PD consists of four DFFs and two XOR gates as shown in Figure 3.3 and its characteristic is shown in Figure 3.4. The Alexander PD uses three data samples S_1-S_3 that are sampled by the three consecutive clock edges. The Alexander PD performs two functions: 1) Determines, whether there is any transition in the input data, and 2) Whether the clock is earlier or later than the input data.

When there is no transition in the input data, all the three samples will have equal values and no action is taken by the Alexander PD. If the falling edge of the clock leads (is "early") then the first two samples S_1 and S_2 will have equal values and the last sample S_3 will have a value, unequal to that of first two samples. Conversely, if the falling edge of the clock lags (is "late") then the last two samples S_2 and S_3 will have equal values and the first sample S_1 will have a value, unequal to that of last two samples. The decisions of the Alexander PD depend on the values of the three samples (S_1, S_2, and S_3) and are presented in Table 3.2.

In Figure 3.3, the first flip flop (FF1) samples the input data at S_1 and S_3 on the rising edge of the clock and the second flip flop (FF2) delays the output of the first flip flop (FF1) by one clock cycle. The third flip flop (FF3) samples the input data at S_3 on the falling edge of the clock and the fourth flip flop (FF4) delays the output of the third flip flop (FF3) by half a clock cycle.

As seen from the waveform of Figure 3.3, for the early case, the FF1 samples the high data level (logic one) at the first rising edge of the clock. At the second rising edge of the clock, the FF2 performs two functions: 1) Produces the replica of the first sample (S_1) delayed by one clock cycle, at the output of the FF2, and 2) Samples the low data level (logic zero).

The FF3 samples the high data level (logic one) at the first falling edge of the clock. At the next rising edge of the clock, the FF4 produces the replica of the second sample (S_2) delayed by half a clock cycle, at its output. The clock phases of all the four DFFs should be such that, the three samples S_1, S_2, and S_3 reaches a valid logic level for comparison at $t = T_1$ and remains constant for one clock period. Once the three samples S_1, S_2, and S_3 reaches valid logic level and remain constant for one clock period, the XOR gate produces a valid logic level at the output. The same process is vice versed for the late case and shown in Figure 3.3.

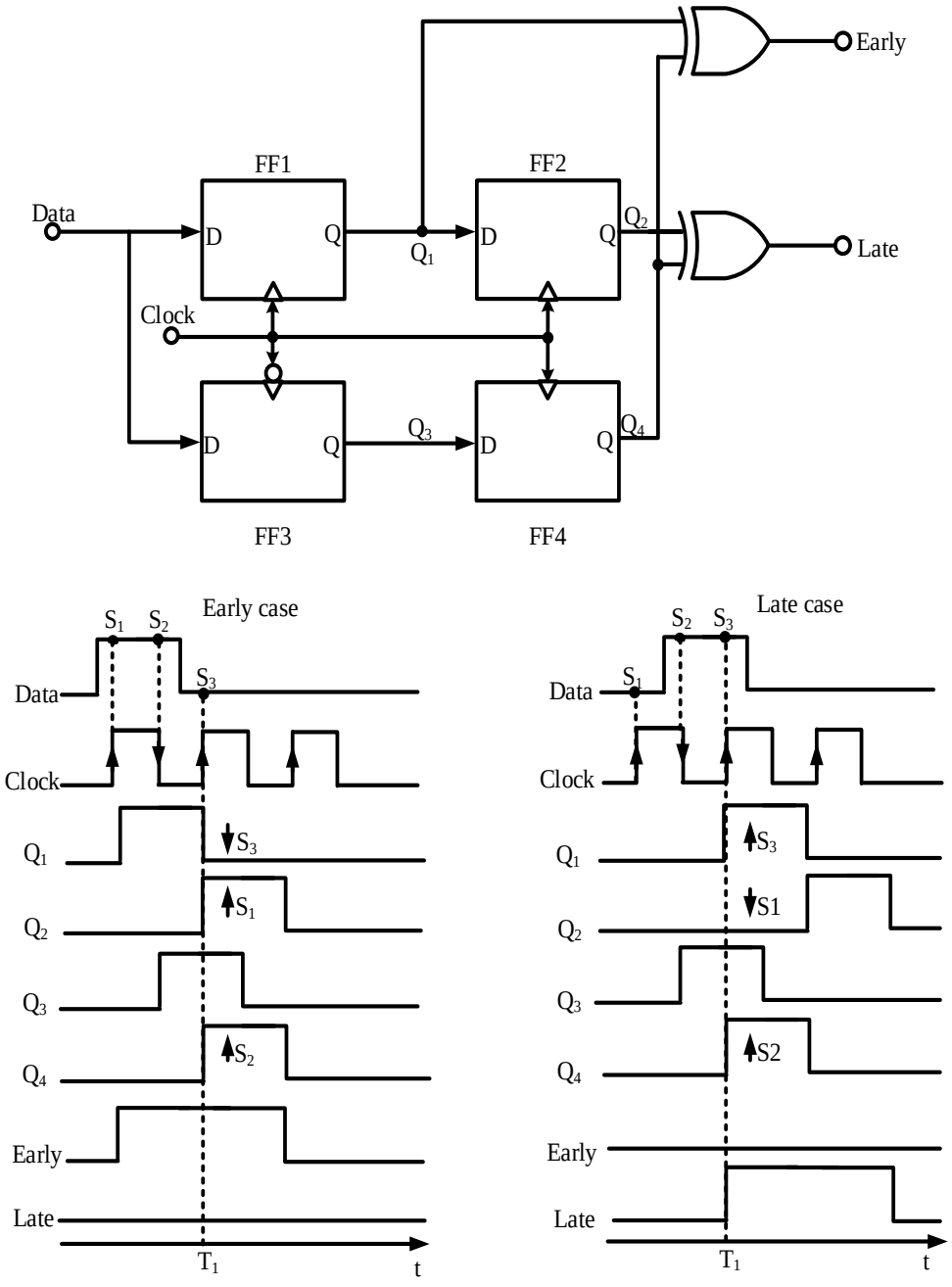

Figure 3.3: Alexander phase detector

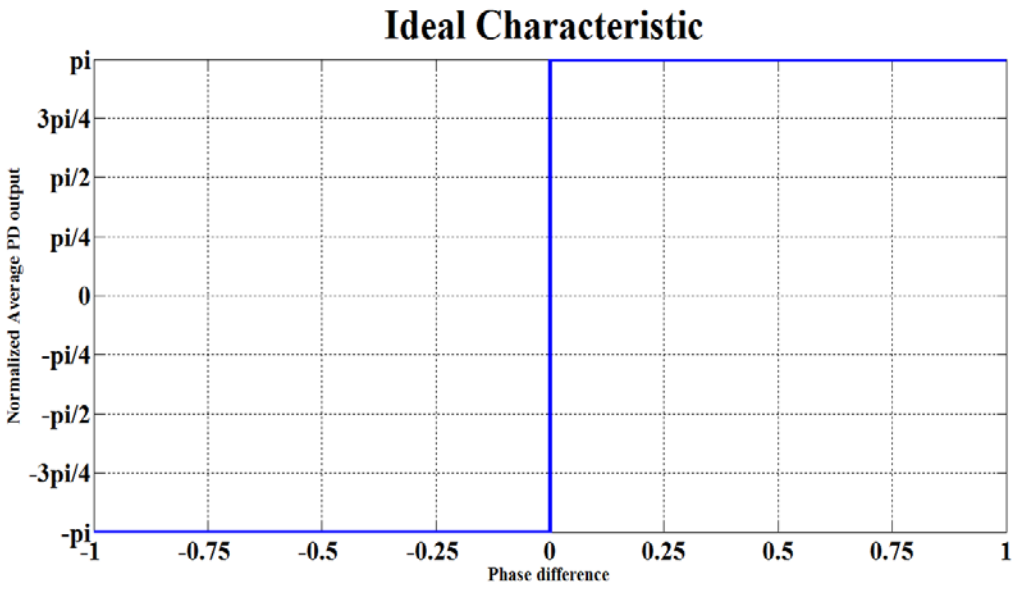

Figure 3.4: Ideal characteristic of Alexander PD.

Table 3.2: Decisions of the Alexander PD.

S_1	S_2	S_3	**Decision**
0	0	0	Cannot determine whether the clock is earlier or later than the data.
0	0	1	Clock is earlier than the data.
0	1	0	CDR is in the lock mode.
0	1	1	Clock is later than the data.
1	0	0	Clock is later than the data.
1	0	1	CDR is in lock mode.
1	1	0	Clock is earlier than the data.
1	1	1	Cannot determine whether the clock is earlier or later than the data.

3.1.2 Charge pump (CP)

The function of the charge pump is to convert the phase difference between the two input signals into the electrical parameter such as voltage, which controls the oscillating frequency of the VCO [13]. The charge pump circuit is modeled in the Simulink by using a gain block and an adder block as shown in Figure 3.5. A gain block holds the value of the charge pump current I_{cp} that charges or discharges the capacitor, when early or late pulses of the Alexander PD are at logic level one respectively.

The I_{cp} value is set to 800 µA and is divided by 2π to cancel the radians unit of the K_{vco} (gain) value of VCO. When the early signal is at logic level one, the capacitor is charged to 127.32 µV (800 µA /2π) and when the late signal is at logic level one, the capacitor is discharged to -127.32µV, but in reality the capacitor discharges to zero volts. The charging and discharging currents of charge pump circuit are easily cancelled in the Simulink model, but in reality leakage current flows through the circuit, thereby creating an offset voltage on the capacitor.

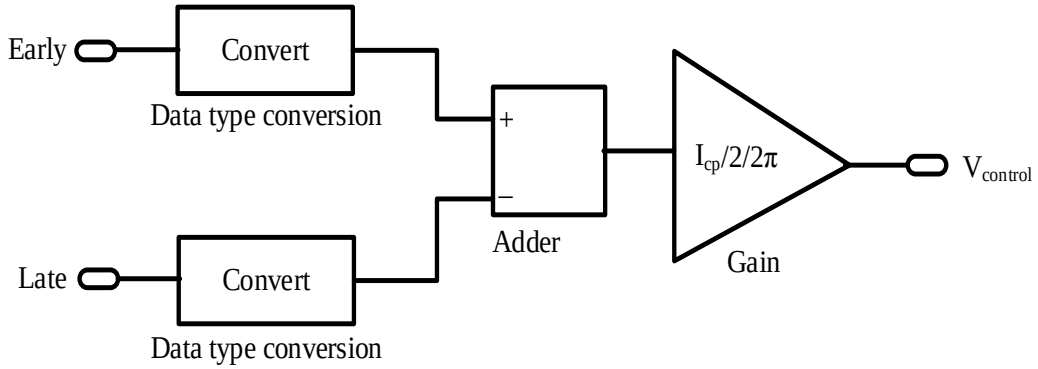

Figure 3.5: Simulink model for the charge pump.

3.1.3 Low pass filter (LPF)

The low pass filter is modeled in the Simulink as per the Figure 2.6 (The resistor (R) is in series with capacitor (C_1) and both are in parallel with capacitor (C_2)) by using a transfer function block. The transfer function of LPF circuit is derived as follows and is given by equation (3.4).

$$H(s) = \left(R + \frac{1}{C_1 s} \right) || \frac{1}{C_2 s} \tag{3.1}$$

$$H(s) = \frac{\left(\frac{1 + R C_1 s}{C_1 s}\right).\left(\frac{1}{C_2 s}\right)}{\left(\frac{1 + R C_1 s}{C_1 s}\right) + \left(\frac{1}{C_2 s}\right)} \tag{3.2}$$

$$H(s) = \frac{\frac{1 + R\,C_1 s}{s^2\,C_1\,C_2}}{\frac{C_1 s + R\,C_1\,C_2 s^2 + C_2 s}{s^2 C_1\,C_2}} \tag{3.3}$$

$$H(s) = \frac{1 + R\,C_1 s}{R\,C_1 C_2 s^2 + (C_1 + C_2)s} \tag{3.4}$$

3.1.4 Voltage controlled oscillator (VCO)

The VCO is modeled in the Simulink by using an adder, a constant, and a math function blocks as shown in Figure 3.6. The VCO model consists of the two variables named K_{vco} and f_o. The K_{vco} is the gain of the VCO in rads/Volts and f_o is the oscillating frequency of the VCO in Hz. The frequency of the clock generated by the VCO varies linearly with the input terminal $V_{control}$. When the input terminal $V_{control}$ is zero, the VCO produces the waveform that oscillates at frequency f_o and on increase in the input terminal $V_{control}$, the oscillating frequency of the waveform generated by VCO increases linearly.

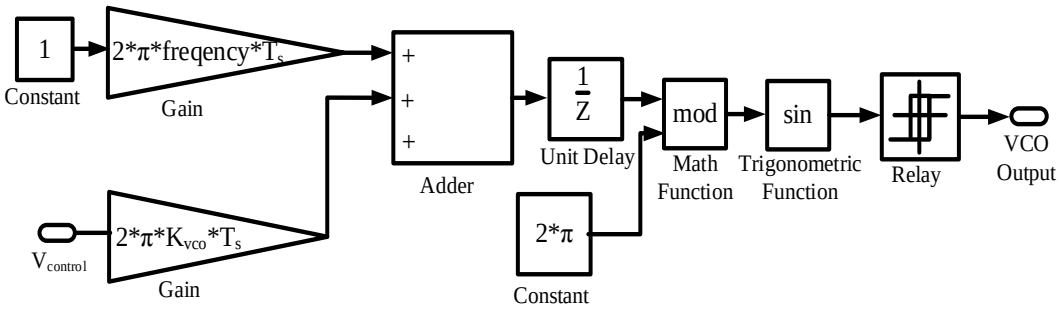

Figure 3.6: Simulink model for the VCO.

3.2 Phase lock loop (PLL) dynamics

The PLL is known as a second order system because it consists of the two dominant poles. The first pole is contributed by a combination of the charge pump and a low pass filter and the second pole is contributed by the VCO. The PLL in terms of the control systems is shown in Figure 3.7 [14].

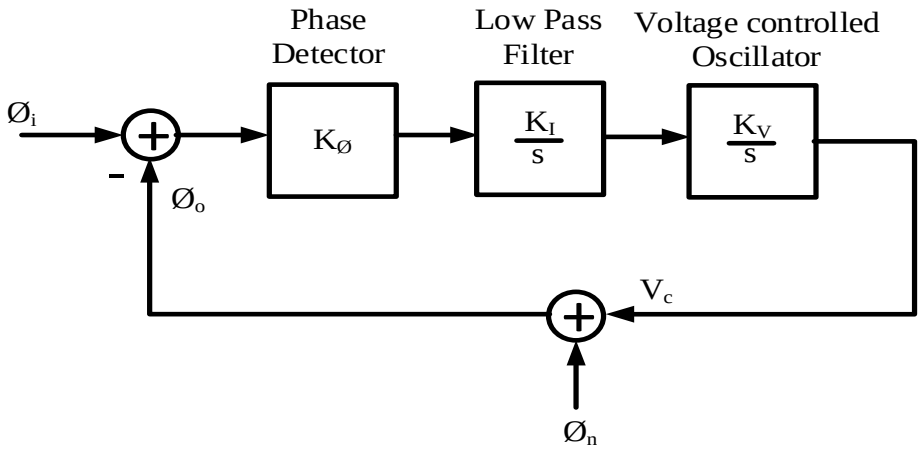

Figure 3.7: Dynamic of the PLL [14].

$$H_{LPF}(s) = K_P + \frac{K_I}{s} \tag{3.5}$$

$$K_I = \frac{1}{C}\ , \quad K_P = R \tag{3.6}$$

$$K_\phi = \frac{I_{CP}}{2\pi} \tag{3.7}$$

The open and closed loop transfer functions of the PLL are given by equations (3.8) and (3.9) respectively.

$$G(s) = \frac{K_{VCO}\, K_{\phi}\, (K_I + K_P s)}{s^2} \tag{3.8}$$

$$H(s) = \frac{K\,(1+\frac{s}{z})}{s^2 + K\frac{s}{z} + K} \tag{3.9}$$

$$K = K_{VCO}\, K_{\phi}\, K_I \text{ and } z = \frac{K_I}{K_P} \tag{3.10}$$

The open and closed loop functions of the PLL in terms of the cut-off frequency of the PLL loop ω_n and the damping factor ξ are given by equations (3.11) and (3.12).

$$G(s) = \frac{2\,\xi\,\omega_n s + \omega_n{}^2}{s^2} \tag{3.11}$$

$$H(s) = \frac{2\,\xi\,\omega_n s + \omega_n{}^2}{s^2 + 2\,\xi\,\omega_n s + \omega_n{}^2} \tag{3.12}$$

The cut-off frequency of the PLL loop ω_n and the damping factor ξ can be formulated as follows:

$$\omega_n = \sqrt{K_V K_\phi K_I} = \sqrt{\frac{I_{CP} K_V}{2\pi C}} \quad (3.13)$$

$$\xi = \frac{\omega_n}{2} \frac{K_P}{K_I} = \frac{R}{2} \sqrt{\frac{I_{CP} K_V C}{2\pi}} \quad (3.14)$$

From the dynamics and formulations mentioned above, the loop bandwidth and the phase margin can be plotted in order to determine the stability and the allowable bandwidth of the PLL.

3.3 CDR Simulink model simulation results

The variables of the designed CDR model are initialized as shown in the Table 3.3. To test the stability of the designed CDR model, a frequency step in the input data is performed, i.e., the input data frequency is changed from 3 Gbps to 2.5 Gbps at 1 µs and the appropriate change in the value of the control voltage ($V_{control}$) of the VCO is observed.

The equation (3.15) is used to manually calculate the value of the $V_{control}$ of the VCO during the frequency step.

$$V_{control} = \frac{f_{vco} - f_O}{K_{vco}} \quad (3.15)$$

Where the f_o is the oscillating frequency of the VCO, the f_{vco} is the frequency of the clock generated by the VCO, and the K_{vco} is the gain of the VCO.

Table 3.3: Initialization of variables of the designed CDR system.

Variables	**Value**
Input Data 1	3 Gbps
Input Data 2	2.5 Gbps
I_{cp}	800 µA
R	1 kΩ
C1	1 pF
C2	C1/10
K_{VCO}	500 MHz/V

The designed CDR model is simulated for four different cases as follows:

- **Case 1:** The designed CDR model is simulated using an ideal DFF from the Simulink library in the Alexander PD model for 4 µs. The $V_{control}$ of the VCO during the frequency step in the input data from 3 Gbps to 2.5 Gbps is plotted in the Figure 3.8. The lock time is defined as the time taken by the $V_{control}$ of the VCO to settle to a constant value, when the frequency step in the input data is performed. The lock time of the designed CDR model was 0.110 µs. The eyediagram of the recovered clock shows the peak-to-peak jitter present in the recovered clock and for the designed CDR model is shown in Figure

3.9. The peak-to-peak jitter observed in the recovered clock of the designed CDR system was 0.03 UI for the 3 Gbps input data and 0.033 UI when the input data frequency was changed to 2.5 Gbps at 1 µs.

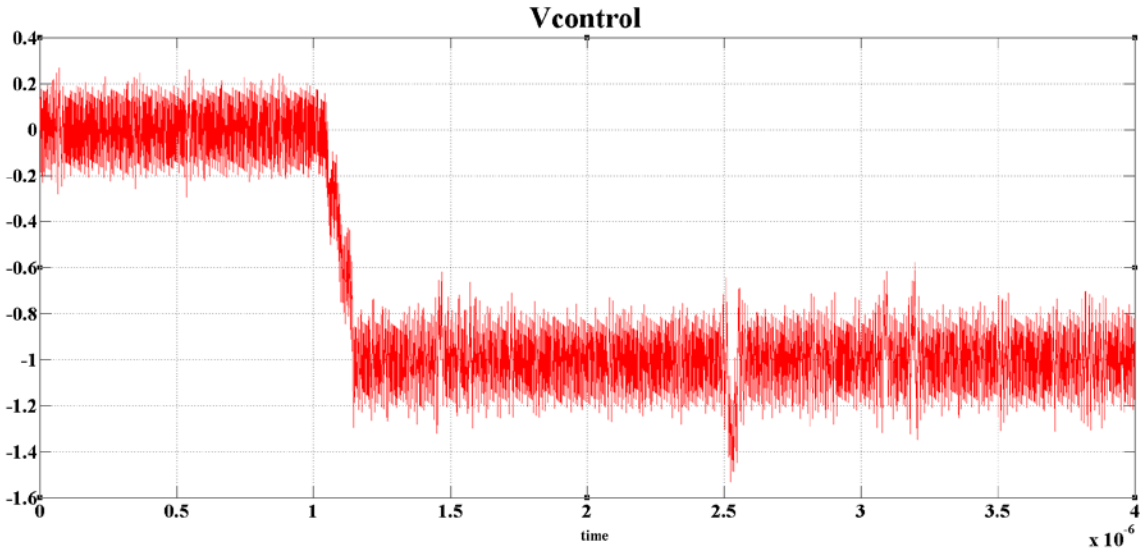

Figure 3.8: Control voltage of the VCO for Case 1.

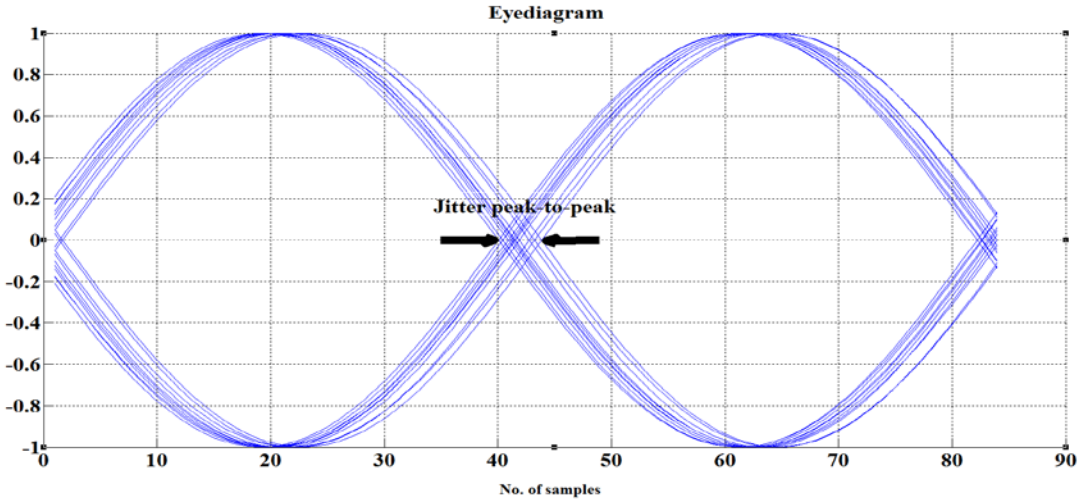

Figure 3.9: Eyediagram of the designed CDR system.

- **Case 2:** The designed CDR system is simulated using the metastable DFF, modeled in Simulink, for 4 µs. The SDFF designed in the transistor level using 45 nm technology in the Cadence Virtuoso has the setup time ($T_{s\ (actual)}$) equal to -11.57 ps, the hold time ($T_{h\ (actual)}$) equal to 54.16 ps, and the clock-to-output delay ($T_{c\text{-}q}$) equal to 84.64 ps (explained in detail in section 4.1 and shown in Table 4.1). The pulse width of the metastable window of the input data is set to 42.59 ps ($T_{s\ (actual)} + T_{h\ (actual)}$) by initializing the timing parameters of the modeled metastable DFF as follows:

$$T_s = T_h = \frac{T_{s\ (actual)} + T_{h\ (actual)}}{2} = 21.29\ \text{ps} \qquad (3.16)$$

The designed CDR model is simulated and the $V_{control}$ of the VCO is plotted in Figure 3.10. The lock time of the CDR model was 1.7 µs. The peak-to-peak jitter observed in the recovered clock was 0.04 UI for the 3 Gbps input data and 0.037 UI when the input data frequency was changed to 2.5 Gbps at 1 µs.

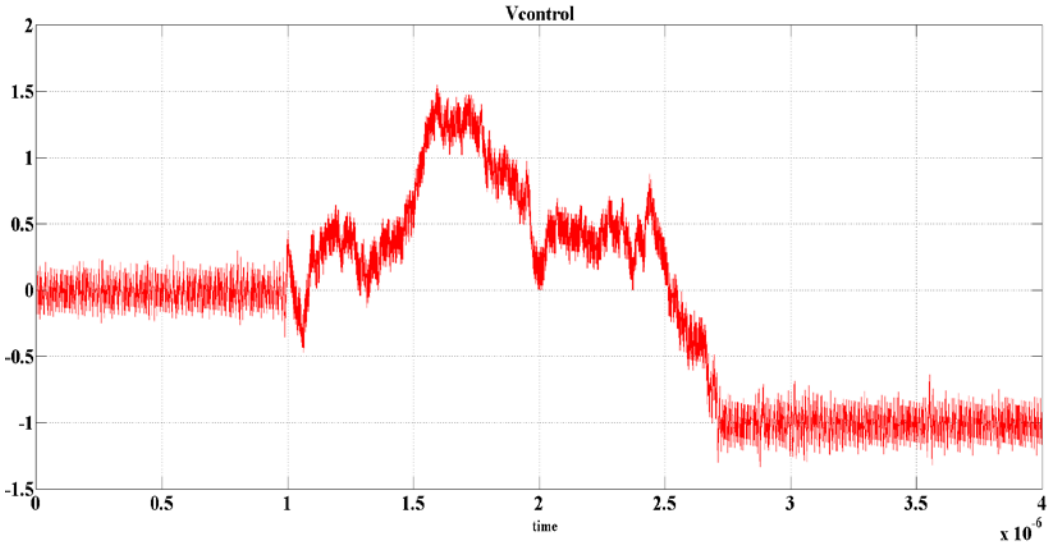

Figure 3.10: Control voltage of the VCO for Case 2.

- **Case 3:** The designed CDR system is simulated using the metastable DFF, modeled in Simulink, for 4 µs. The setup time (T_s) is initialized to a greater value than the hold time (T_h) in the modeled metastable DFF as follows:

$$T_s = 34.07 \text{ ps and } T_h = 8.51 \text{ ps} \tag{3.17}$$

The designed CDR model is simulated and the $V_{control}$ of the VCO is plotted in Figure 3.11. This figure shows that the designed CDR model does not lock when the input frequency is changed from 3 Gbps to 2.5 Gbps at 1 µs. The peak-to-peak jitter observed in the recovered clock was 0.0612 UI for the 3 Gbps input data.

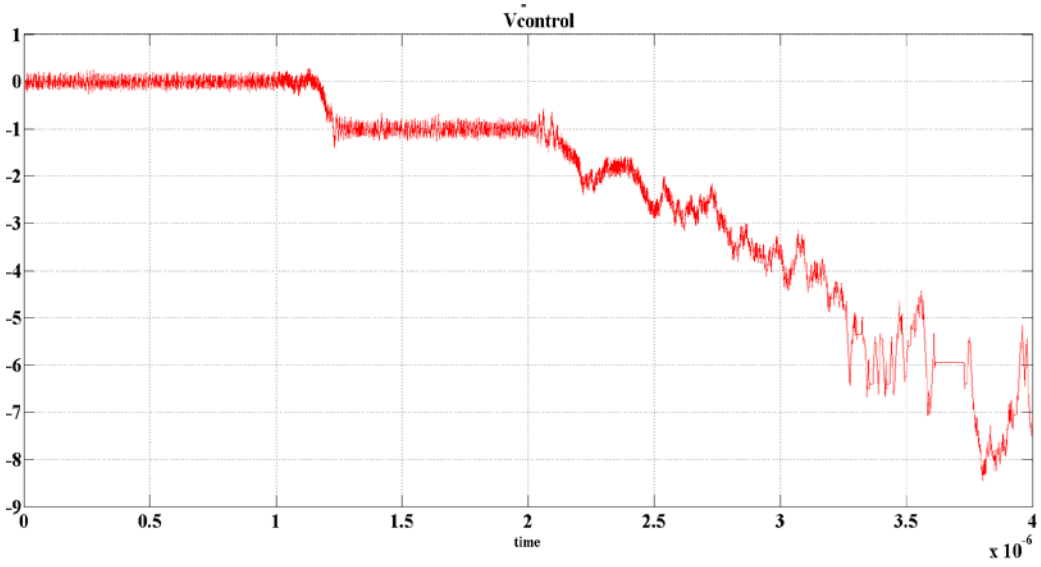

Figure 3.11: Control voltage of the VCO for Case 3.

- **Case 4:** The designed CDR system is simulated using the metastable DFF, modeled in Simulink, for 4 µs. The hold time (T_h) is initialized to a greater value than the setup time (T_s) in the modeled metastable DFF as follows:

$$T_s = 8.51 \text{ ps and } T_h = 34.07 \text{ ps} \tag{3.18}$$

The designed CDR model is simulated and the $V_{control}$ of the VCO is plotted in Figure 3.12. This figure shows that the designed CDR model does not lock when the input frequency is changed from 3 Gbps to 2.5 Gbps at 1 µs. The

peak-to-peak jitter observed in the recovered clock was 0.05 UI for the 3 Gbps input data.

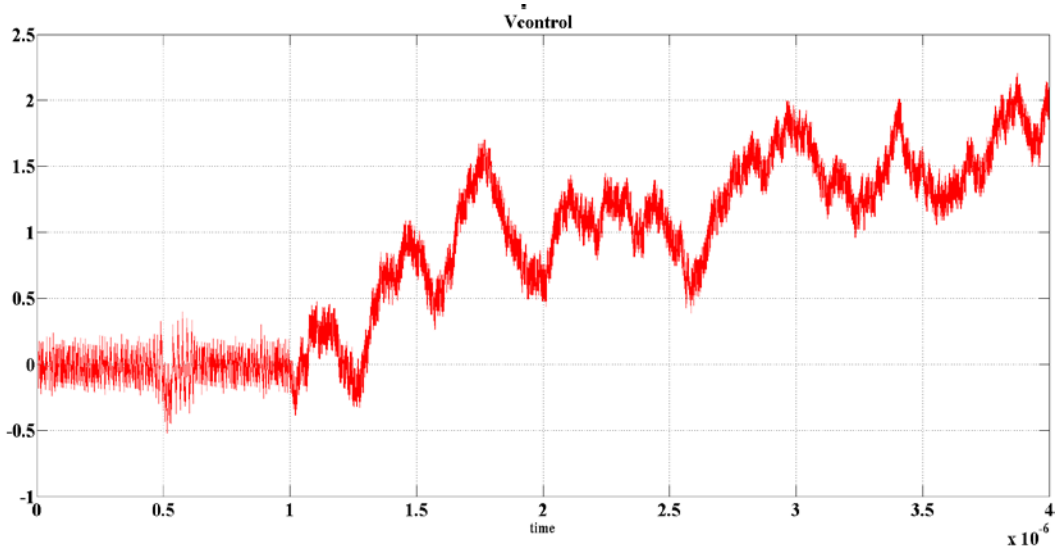

Figure 3.12: Control voltage for case 4.

The results of the four different cases are tabulated in Table 3.4:

Table 3.4: Simulation results of the designed CDR system.

Case No.	Type of DFF used in CDR system	Lock Time (µs)	Peak-to-Peak Jitter 3 Gbps (UI)	Peak-to-Peak Jitter 2.5 Gbps (UI)
1	Ideal DFF	0.11	0.03	0.033
2	Metastable DFF	1.7	0.04	0.037
3	Metastable DFF	-	0.06	-
4	Metastable DFF	-	0.05	-

In summary, as seen from the Table 3.4, Case 1 is the best case as the designed CDR system has the minimum lock time and peak-to-peak jitter as compared to Case 2, 3, and 4. In Case 1, the ideal DFF from Simulink library was used in the Alexander phase detector. The timing parameters of the ideal DFF from Simulink library are the setup time (T_s), the hold time (T_h), and the clock-to-output delay ($T_{c\text{-}q}$) are equal to zero. But, when the DFF is designed at transistor level, these timing parameters are no longer zero. Thus, Case 1 is not the possible practically. The timing parameters of the DFF are taken into consideration in the Case 2, 3, and 4. The Table 3.4 shows that the Case 2 is the best case among Case 2, 3, and 4 in terms of minimum lock time and minimum peak-to-peak jitter in the recovered clock of the designed CDR system.

Chapter 4. **CDR modeling using Cadence**

This chapter presents the paper's work in transistor level design using 45 nm technology and modeling of the CDR system using the Verilog-A language in the Cadence Virtuoso 6.1.5. The designed CDR system is operated at the input data rate of 3 Gbps as shown in Figure 4.1. The two PRBS blocks used at the input of the designed CDR system consists of the two input data in the PRBS pattern. The block named "PRBS7-1" consists of the input data, having frequency of 3 Gbps and the other block named "PRBS7-2" consists of the input data, having frequency of 2.9 Gbps.

The phase detector block is the Alexander PD consisting of the four DFFs and two XOR gates designed at the transistor level using 45 nm technology. The transistor level designing of the Alexander PD is explained in detail in section 4.1, 4.2, and 4.3. The charge pump (CP) block is modeled using the Verilog-A language and consist of one input variable named I_{cp} (charge pump current). The low pass filter (LPF) block consists of the resistor (R) connected in series with the capacitor (C_1) and both are placed in parallel to the capacitor (C_2). The VCO is also modeled using the Verilog-A language and has two input variables named f_o (oscillating frequency of the VCO) and K_{vco} (gain of the VCO). The slicer block is used to convert the sinusoidal output of the VCO into the square wave and is modeled using the Verilog-A language.

Finally, the data delay circuit is designed at the transistor level using 45 nm technology and is explained in detail in section 4.4. The Verilog-A codes and the

transistor sizes of the circuits of the designed CDR system are provided in the sections A.1 and A.2 of the Appendix respectively.

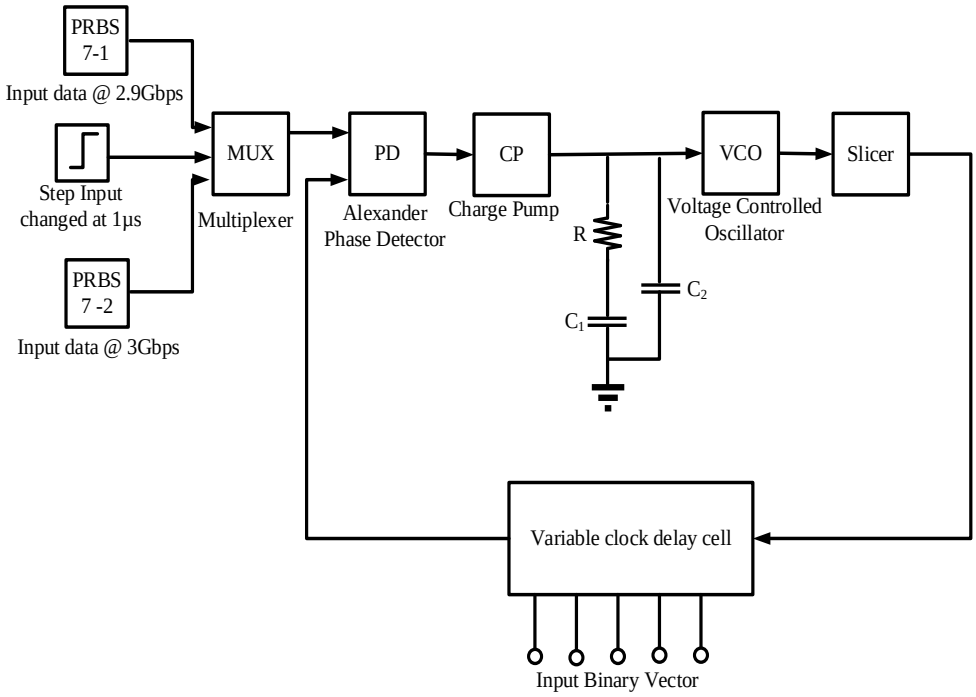

Figure 4.1: Schematic of the designed CDR system.

4.1 SDFF

The Semi-dynamic D flip flop (SDFF) is used in the designed CDR system due to the benefits discussed in section 2.3.1. The SDFF circuit is divided into four parts named A, B, C, and D as shown in Figure 4.2 and explained as follows:

- **Part A** is a dynamic inverter-I consisting of one PMOS transistor (M_1) in the pull up network and three NMOS transistors (M_2, M_3, and M_4) connected in series in the pull down network. The function of the dynamic inverter-I is to sample the inverted input data (D) on the node X when the clock is in the evaluation phase.

- **Part B** is a dynamic inverter-II consisting of one PMOS transistor (M_5) in the pull up network and two NMOS transistors (M_6 and M_7) connected in series in the pull down network. The function of the dynamic inverter-II is to sample the inverted value present at node X on the output (Q) when the clock is either in the precharge or the evaluation phase.

- **Part C** is a static keeper circuit consisting of two inverters connected in back to back configuration. The function of the static keeper is to hold the voltage of the node X and the output (Q) to the appropriate logic level. Due to the presence of the two static keepers at node X and output (Q) and the two dynamic inverters, the DFF is called as the Semi dynamic D flip flop (SDFF).

- **Part D** is a NAND gate driven by clock signal, which is delayed by the series combination of the two inverters. A NAND gate along with the series combination of the two inverters forms the glitch generator circuit. The glitch

generator circuit generates the narrow clock pulse around the rising edge of the clock signal. The period of which corresponds to the total sum of a NAND gate and the two inverter delays. The purpose of the glitch generator circuit is to lower the possibility of the race-through problem and sensitivity to the noise.

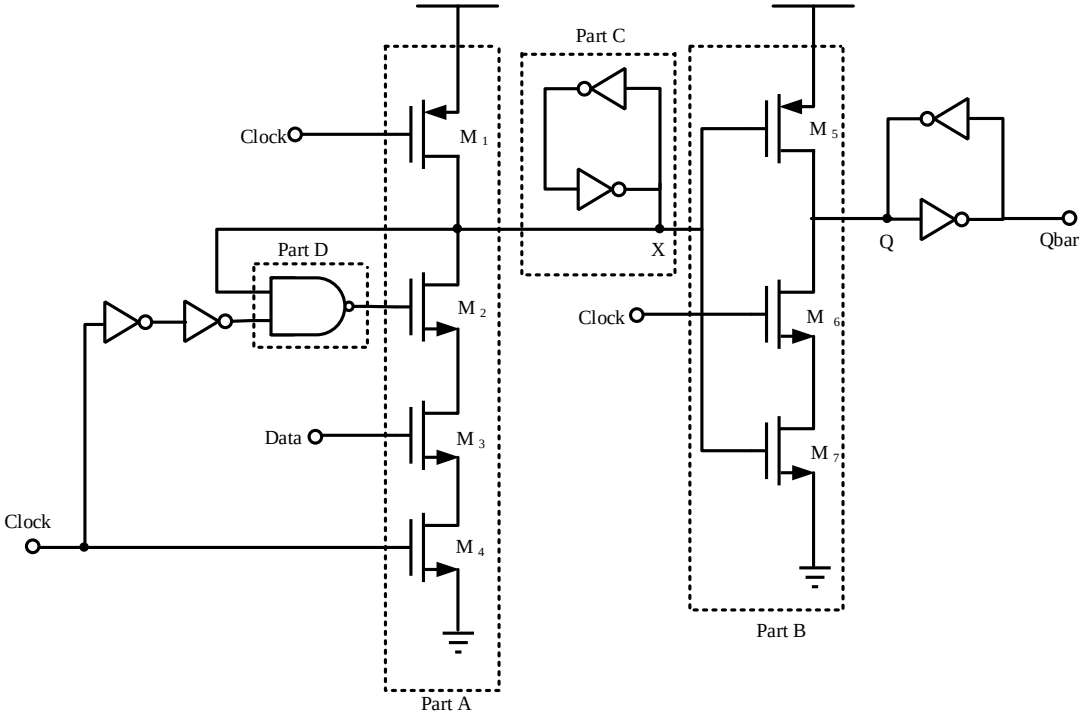

Figure 4.2: Cadence schematic of the SDFF circuit.

The working of the SDFF is divided into two phases: the Precharge phase and the Evaluation phase, as follows:

- In the **Precharge phase,** the clock is at the logic zero; turning on the PMOS transistor (M_1) and turning off the NMOS transistors (M_4 and M_6). The node X is charged to VDD (logic one) through the PMOS transistor (M_1). As node X is one of the inputs of the NAND gate, whose other input is zero from the clock, the NAND gate's output is set to the logic one and turns on the NMOS transistor (M_2). In precharge phase, the output (Q) is cut off from the first stage and is held to either a previous or the random value, causing the dynamic inverter-II to be at a high impedance stage.

  Since the drain terminal of the NMOS transistor (M_2) is connected to the node X, the voltage at node X is pulled below the VDD. As the node X drives the dynamic inverter-II, the load capacitor present at the output (Q) will not get charged to the VDD and will result in low noise margin. To avoid the problem of the low noise margin, the static keepers are used at node X and output (Q) to achieve the rail to rail full supply voltage swing.

- In the **Evaluation phase**, the clock makes the transition from logic zero to logic one and turns on the NMOS transistors (M_4 and M_6) and turns off the PMOS transistor (M_1). As soon as the clock reaches the switching threshold of the dynamic inverter-II, the NMOS transistor (M_6) turns on and the output

(Q) will get discharged all the way to GND (logic zero). In evaluation phase, the circuit behaves as a transparent circuit and the output of the NAND gate remains at logic one for a short interval of time (corresponding to the total sum of a NAND gate and the two inverters delays).

- **Transparency of the SDFF in Evaluation phase:**

  1) Latching a logic zero: When the input data is at a logic zero, the NMOS transistor (M_3) is turned off and as the clock is high, the output (Q) is held to the previous stage, which is a logic zero.

  2) Latching a logic one: When the input data is at logic one, the NMOS transistor (M_3) is turned on. Due to the existence of the direct path between the node X and the GND, the node X is discharged all the way to the GND, thus, turning on the PMOS transistor (M_5) and charging output(Q) to VDD.

The output waveform of the designed SDFF circuit is shown in Figure 4.3 and the transistor sizes of the designed SDFF circuit are mentioned in section A.2.1 of the Appendix. The timing parameters of designed SDFF circuit are presented in Table 4.1.

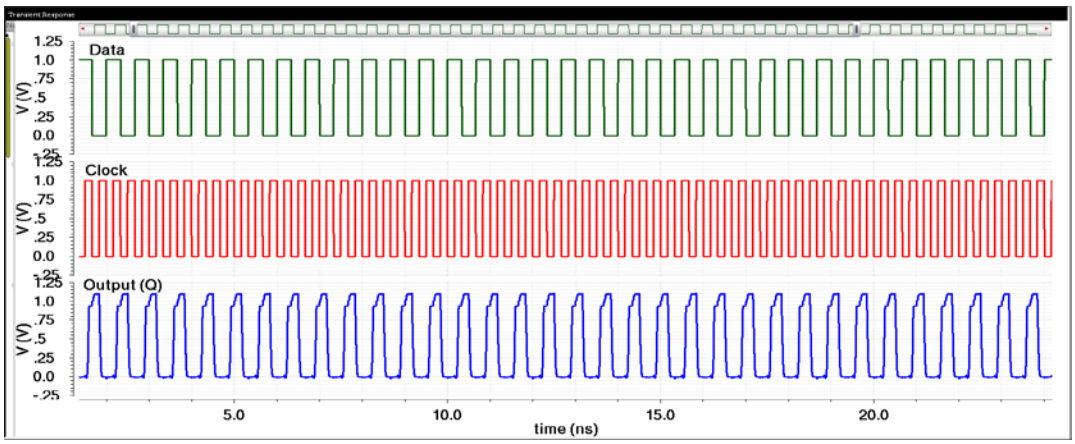

Figure 4.3: Output waveform of the designed SDFF.

Table 4.1: Timing parameters of the designed SDFF.

Timing Parameters	**Time (ps)**
Setup time (T_s)	-11.57
Hold time (T_h)	54.16
Clock to output delay ($T_{c\text{-}q}$)	85.64

4.2 Exclusive OR (XOR) gate

The XOR gate accepts two input signals and gives the output as logic one, when both the inputs have unequal value otherwise it gives output as logic zero. The XOR circuit is designed at transistor level using 45 nm technology as shown in Figure 4.4.

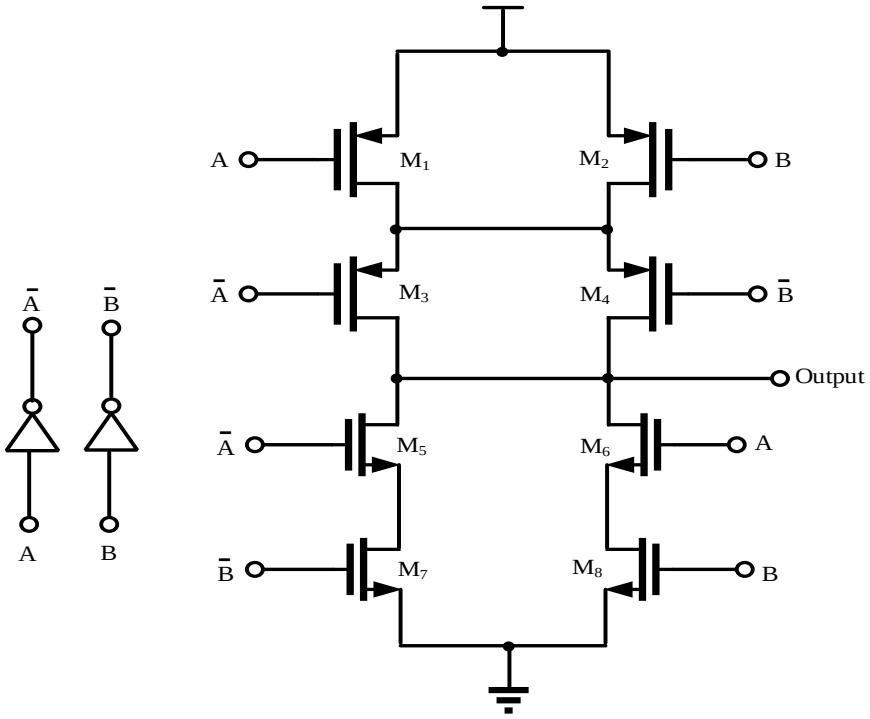

Figure 4.4: Cadence schematic of the XOR gate.

The output waveform of the XOR circuit is shown in Figure 4.5 and the transistor sizes are mentioned in section A.2.2 of the Appendix.

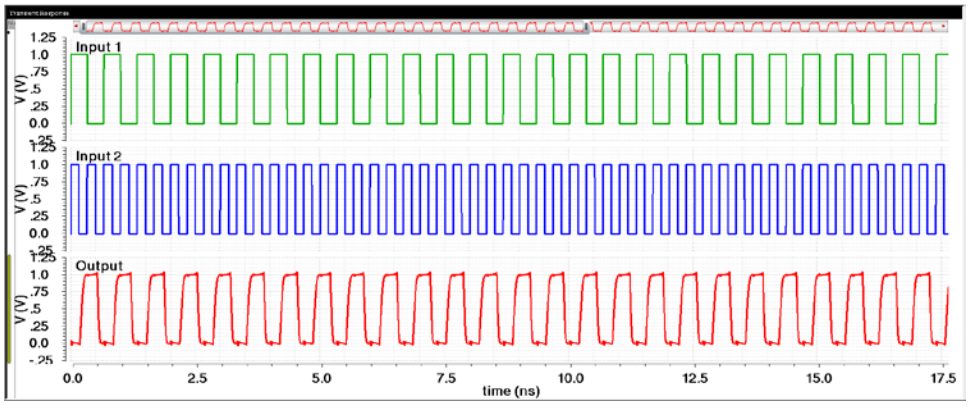

Figure 4.5: Output waveform of the designed gate.

4.3 Alexander PD

The Alexander PD consists of the above designed four DFFs and two XOR gates as shown in Figure 3.3. The main function of an Alexander PD is to determine whether the clock is earlier or later than the input data signal as explained in section 3.1.3.

4.4 Inverter

The function of the inverter is to invert the incoming data signal. The inverters are also used as buffers in various analog and digital circuits. The inverter consists of one NMOS (M_1) and one PMOS (M_2) transistor as sown in Figure 4.6 and the output waveform is shown in Figure 4.7.

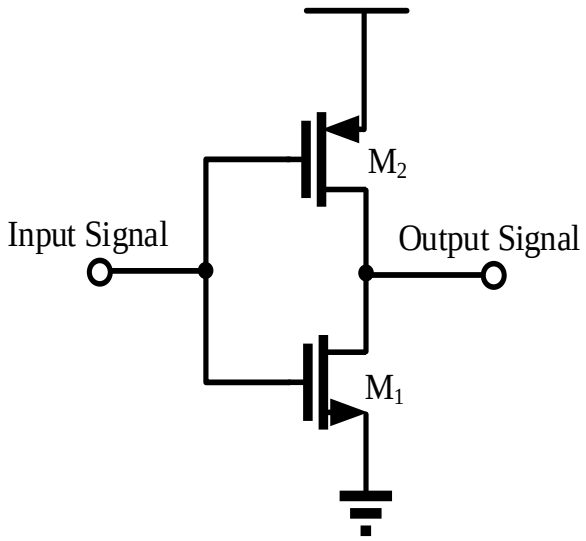

Figure 4.6: Cadence schematic of the Inverter.

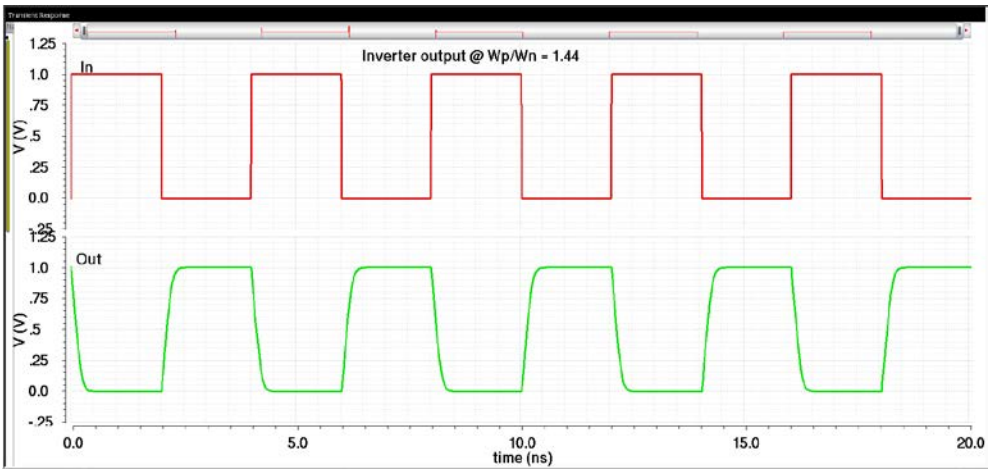

Figure 4.7: Output waveform of the designed inverter.

When the input data switches from the low logic level to the high logic level, the time interval between the input data and the output of the inverter is called low-high propagation delay and is given by t_{plh}. On the other side, When the input data switches from the high logic level to the low logic level, the time interval between the input data and the output of the inverter is called high-low propagation delay and is given by t_{phl}. To have equal propagation delays ($t_{plh} = t_{phl}$) and the switching threshold of 0.5 V, the width of the PMOS transistor (M_2) is sized to 1.44 times the width of the NMOS transistor (M_1) as shown in Figure 4.8. The transistor sizes of the inverter are mentioned in section A.2.3 of the Appendix.

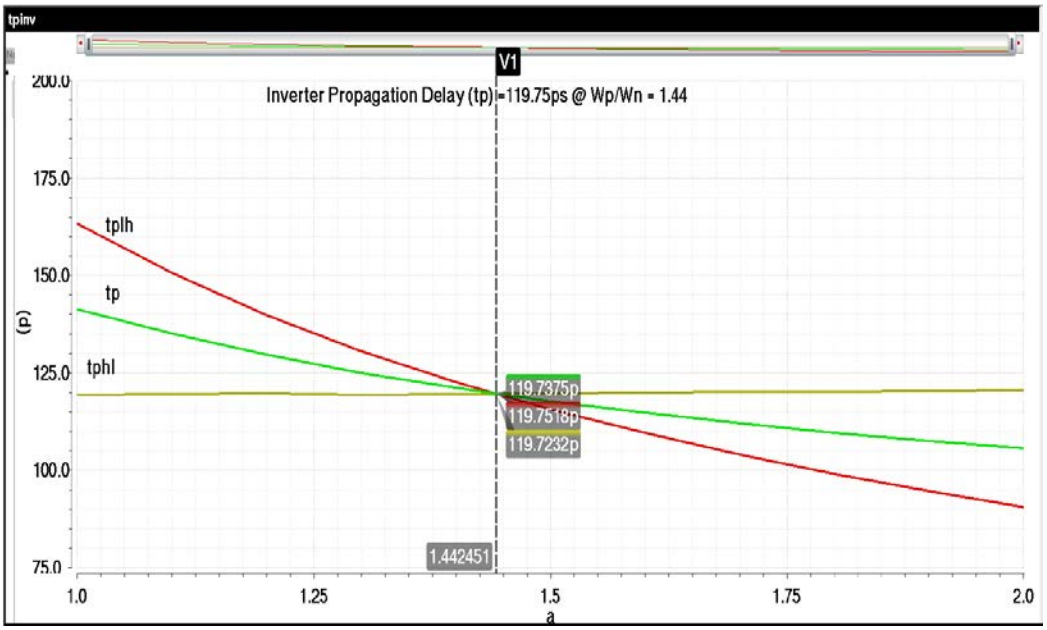

Figure 4.8: Propagation delay waveform of the designed inverter.

4.5 Metastable circuit

The metastable concept is best explained in the section 2.3. This section presents the design of the metastable circuit at the transistor level using 45nm technology. The metastable circuit consists of a glitch generator circuit and a variable clock delay cell. The glitch generator circuit consists of a variable data delay cell, the chain of inverters, and a NAND gate as shown in Figure 4.9.

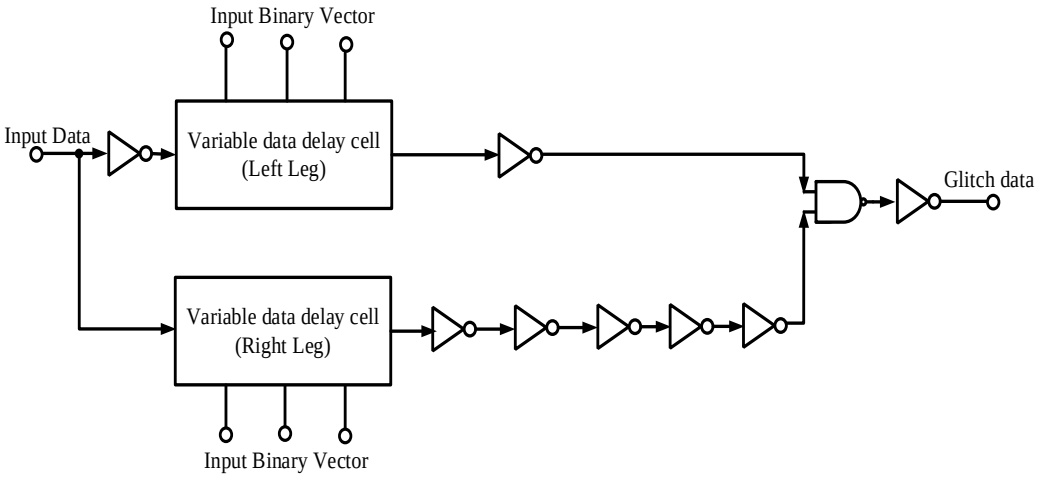

Figure 4.9: Cadence schematic of the glitch generator circuit.

The variable delay cells are widely used in the integrated circuits to delay the active edge of the clock or of any random signals [15]. The variable data delay cell is designed using the current starved circuit as shown in Figure 4.10. The figure shows the controlling transistors (M_{n0}, M_{n1}, $M_{n2,}$ & M_{n3}, and M_{p0}, M_{p1}, M_{p2}, & M_{p3}) are turned on at the source node of the transistors (M_1) and (M_2) by applying the binary vector to its input terminal [15]. To achieve a binary incremental delay of 2 ps in the input data, the controlling transistors are sized in the binary fashion. The pulse width of the data pulse (metastable window) generated by the glitch generator circuit is varied by six digital bits (the left and right leg each are varied using three digital bits) as shown in Figures 4.11 and 4.12. The total pulse width obtained due to each binary vector is tabulated in the Tables 4.2 and 4.3. The transistor sizes of the designed data delay cell are mentioned in section A.2.4 of the Appendix.

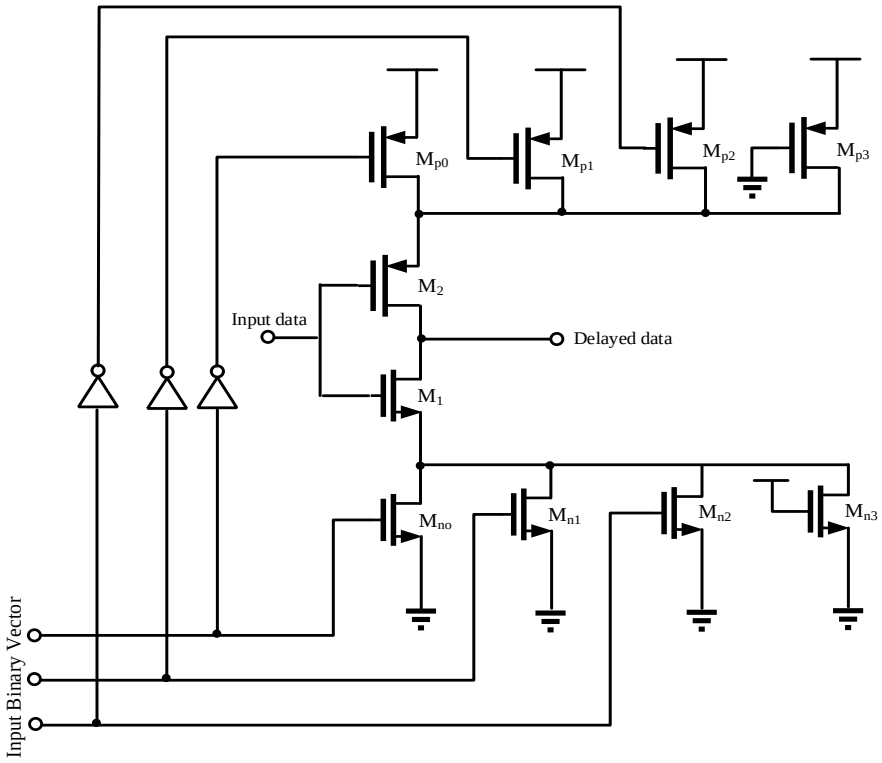

Figure 4.10: Cadence schematic of the data delay cell [15].

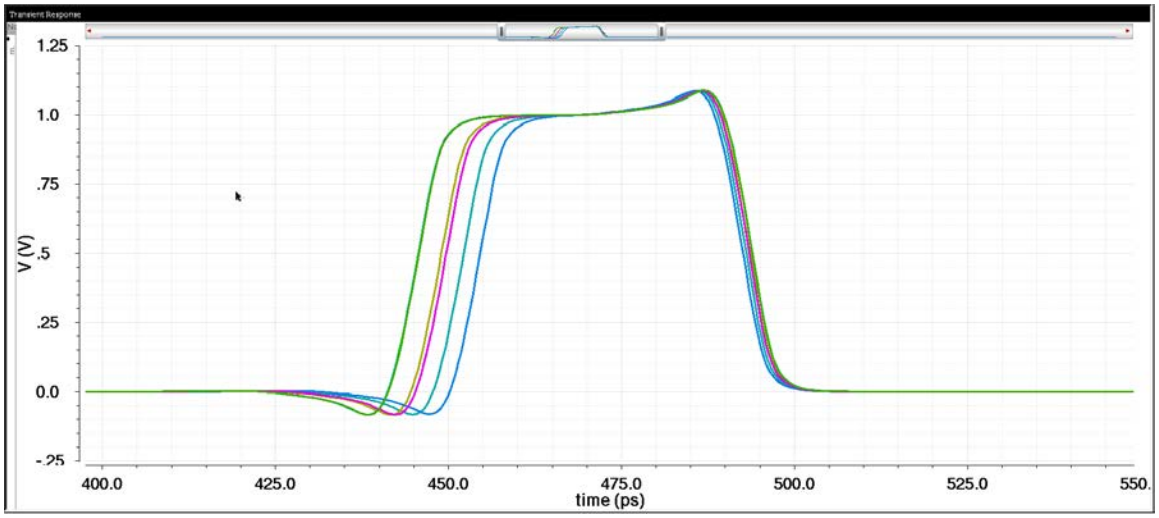

Figure 4.11: Variation in the left leg of the metastable window.

Table 4.2: Binary vectors to vary the left leg of the metastable window.

A_1	A_2	A_3	Delay time (ps)
0	0	0	38.27
0	0	1	40.37
0	1	0	42.54
0	1	1	44.64
1	0	0	46.25
1	0	1	48.35
1	1	0	44.55
1	1	1	52.62

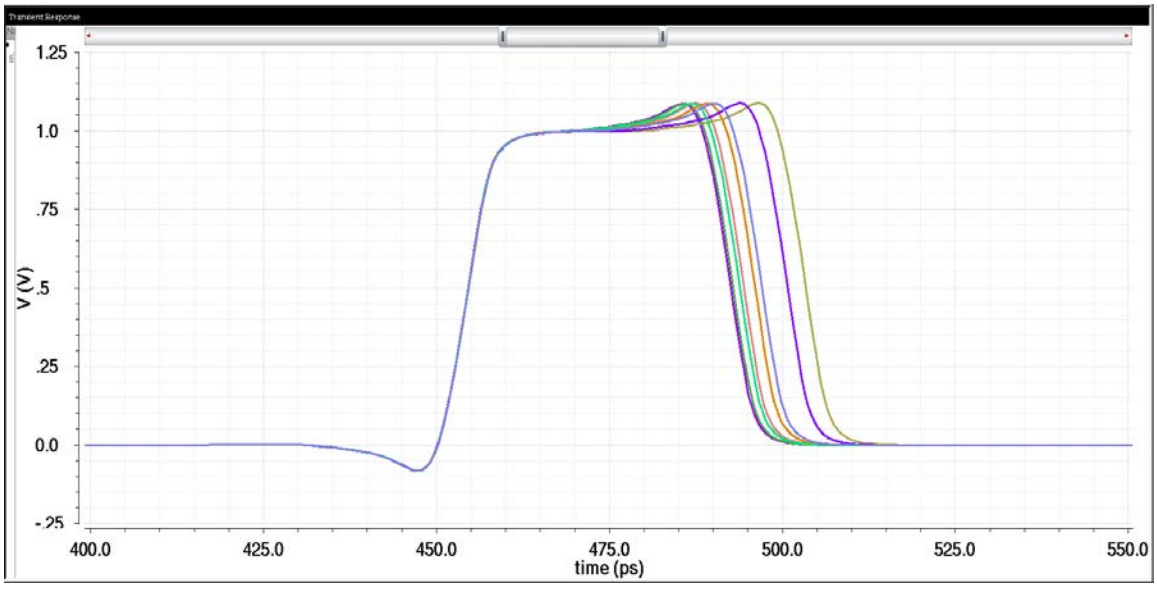

Figure 4.12: Variation in the right leg of the metastable window.

Table 4.3: Binary vectors to vary the right leg of the metastable window.

B_1	B_2	B_3	Delay time (ps)
1	1	1	38.27
1	1	0	40.37
1	0	1	42.54
1	0	0	44.64
0	1	1	46.25
0	1	0	48.35
0	0	1	44.55
0	0	0	52.62

The clock delay cell is designed by using the two inverters connected in back to back configuration and each inverter contains the PMOS transistor, connected in parallel configuration as shown in Figure 4.13. The clock delay cell, delays the clock in the precision of 2 ps by using the five digital bits as shown in Figure 4.14. The total delay in the clock caused by each binary vector is tabulated in Table 4.4. The transistor sizes of the clock delay cell are mentioned in section A.2.5 of the Appendix.

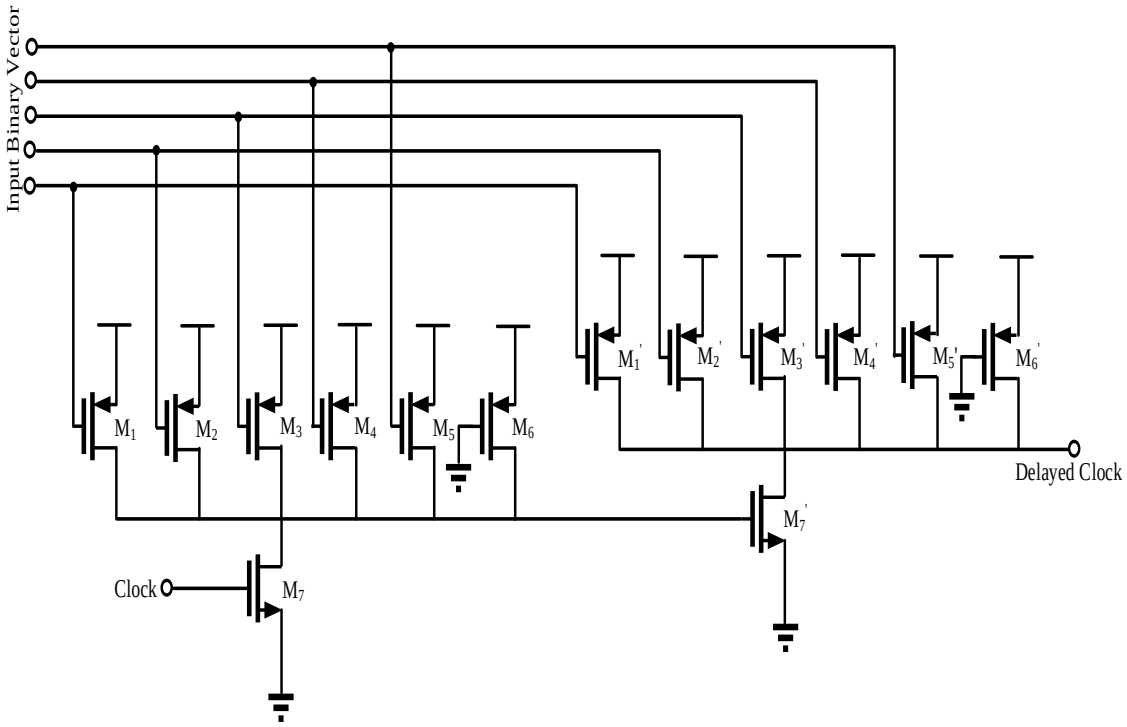

Figure 4.13: Cadence schematic of the clock delay cell.

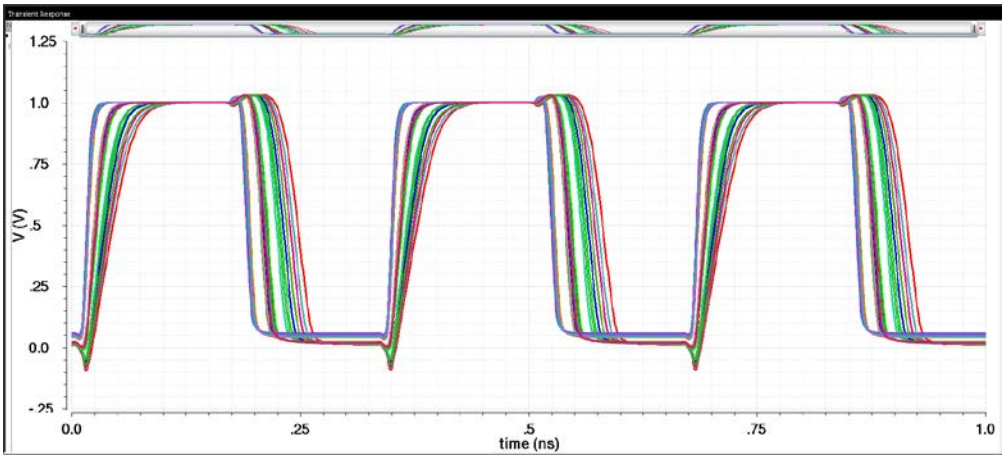

Figure 4.14: Variable delay in the clock.

Table 4.4: Binary vectors to vary the clock.

C_1	C_2	C_3	C_4	C_5	**Delay time (ps)**
1	1	1	1	1	42.15
1	1	1	1	0	17.88
1	1	1	0	1	25.91
1	1	1	0	0	3.17
1	1	0	1	1	33.85
1	1	0	1	0	11.41
1	1	0	0	1	17.61
1	1	0	0	0	17.88
1	0	1	1	1	37.63
1	0	1	1	0	14.89
1	0	1	0	1	21.39
1	0	1	0	0	17.88
1	0	0	1	1	29.33
1	0	0	1	0	6.59
1	0	0	0	1	13.09
1	0	0	0	0	17.88
0	1	1	1	1	39.54
0	1	1	1	0	18.8
0	1	1	0	1	23.3

0	1	1	0	0	17.88
0	1	0	1	1	31.24
0	1	0	1	0	8.5
0	1	0	0	1	15
0	1	0	0	0	17.88
0	0	1	1	1	35.02
0	0	1	1	0	12.28
0	0	1	0	1	18.78
0	0	1	0	0	17.88
0	0	0	1	1	26.72
0	0	0	1	0	17.88
0	0	0	0	1	17.88
0	0	0	0	0	54.41

4.6 CDR simulation results

The designed CDR system without the clock delay cell was simulated for 3 µs with the frequency step in the input signal from 3 Gbps to 2.9 Gbps at 1 µs. The variables present in the designed CDR system are initialized as shown in Table 4.5. During the change in the frequency of the input data from 3 Gbps to 2.9 Gbps, the control voltage ($V_{control}$) of the VCO makes the transition from 500 mV to 300 mV respectively (by equations 4.1, 4.2, and4.3). The transition of the $V_{control}$ of the VCO is shown in Figure 4.15 and the eyediagram is shown in Figure 4.16. The lock time observed was 0.32 µs

and the peak-to-peak jitter observed in the recovery clock of the designed CDR system was 0.022 UI.

$$\text{Vcontrol} = \frac{f_{VCO} - f_O}{K_{VCO}} \quad (4.1)$$

$$\text{Vcontrol} = \frac{3\ \text{GHz} - 2.75\ \text{GHz}}{500\ \text{MHz/V}} = 500\ \text{mV} \quad (4.2)$$

$$\text{Vcontrol} = \frac{2.9\ \text{GHz} - 2.75\ \text{GHz}}{500\ \text{MHz/V}} = 300\ \text{mV} \quad (4.3)$$

Table 4.5: Initialization of the variables of the designed CDR system

Variables	**Value**
Input Data 1	3 Gbps
Input Data 2	2.9 Gbps
I_{cp}	800 µA
R	1 kΩ
C1	20 pF
C2	C1/40
K_{VCO}	500 MHz/V

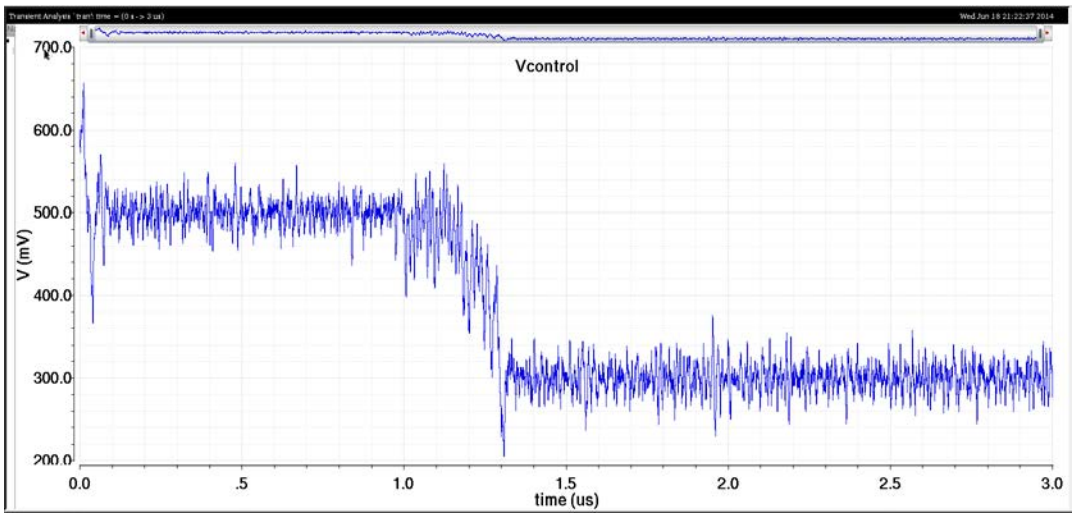

Figure 4.15: Simulation result of the designed CDR system.

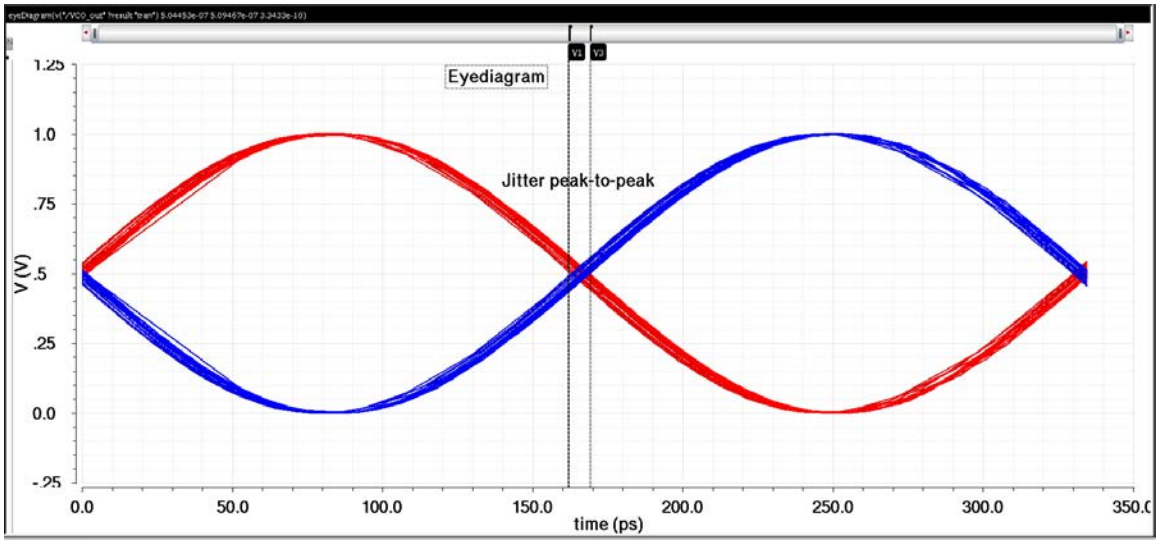

Figure 4.16: Eyediagram of the designed CDR system.

The designed CDR system is simulated with the variable clock delay cell in the feedback loop as shown in Figure 4.1. The designed CDR system was simulation for 2 μs with the input data rate of 3 Gbps.

The width of the data pulse generated by the glitch generator circuit should be equal to 44.52 ps (sum of the setup time (T_s) and the hold time (T_H), mentioned in Table. 4.1). From the Tables 4.2 and 4.3, the data pulse with the width of 44.52 ps is achieved by using the [0 1 0 1 1 1] binary vector. The rising edge of the clock is delayed or aligned with the data pulse by using a five bit variable clock delay cell as explained in the following cases:

- **Case1:** In this case, the designed CDR system was simulated by making the setup time (T_s) equal to the hold time (T_h) of the SDFF. The setup time (T_s) is made equal to the hold time (T_h) by delaying or aligning the rising edge of the clock to the center of the metastable window as shown in Figure 4.17. Thus, the rising edge of the clock is delayed by 22.26 ps (half of the data pulse width of 44.52 ps) by using the [0 0 0 1 1] binary vector as shown in Table 4.4. Note, this binary vector provides the delay of approximately 25.43 ps, thus, there exists an offset of approximately 3 ps due to noise in the circuit. The peak-to-peak jitter observed in the recovery clock of the designed CDR system was 0.016 UI.

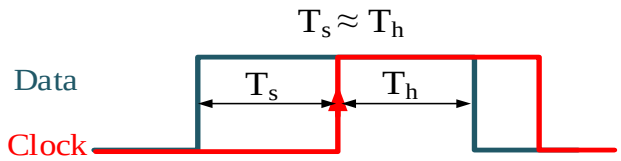

Figure 4.17: Alignment of the clock edge for Case 1.

- **Case2:** In this case, the designed CDR circuit was simulated by making the setup time (T_s) greater than the hold time (T_h) of the SDFF. The setup time (T_s) is made greater than the hold time (T_h) by delaying or aligning the rising edge of the clock as shown in Figure 4.18. Hence, the rising edge of the clock is delayed by 36.53 ps by using the [1 0 0 1 1] binary vector as shown in Table 4.4. The peak-to-peak jitter observed in the recovery clock of the designed CDR system was 0.089 UI.

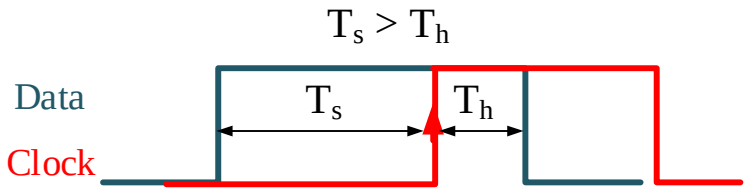

Figure 4.18: Alignment of the clock edge for Case 2.

- **Case3:** In this case, the designed CDR system was simulated by making the hold time (T_h) greater than the setup time (T_s) of the SDFF. The hold time (T_h) is made greater than the setup time (T_s) by aligning or delaying the clock as shown in the Figure 4.19. Hence, the rising edge of the clock is delayed by

14.98 ps by using the [1 1 1 1 1] binary vector as shown in Table 4.4. The peak-to-peak jitter observed in the recovery clock of the designed CDR system was 0.095 UI.

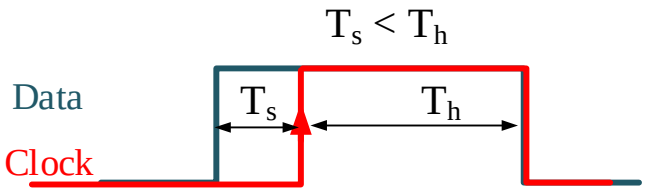

Figure 4.19: Alignment of the clock edge for Case 3.

Chapter 5. **Conclusion**

The CDR system was first modeled using the Simulink software and simulated for three different cases: the equal setup and hold time, the setup time greater than the hold time, and the hold time greater than the setup time. The results in Table 3.4 show that when setup time is equal to hold time, the designed CDR system performs best in terms of minimum lock time and peak-to-peak jitter. The lock time reported in this case was 1.7 µs. The peak-to-peak jitter observed in the recovered clock was 0.04 UI for the 3 Gbps input data and 0.037 UI for the 2.5 Gbps input data.

To validate the observations in Simulink, the CDR system was designed at transistor level using 45 nm technology and modeled using Verilog A language in Cadence Virtuoso 6.1.5. The results obtained from Cadence simulations show that when the setup time is equal to the hold time, the peak-to-peak jitter observed in the recovered clock was 0.016 UI, which is less as compared to that observed in other two cases (when the setup time is greater than hold time and hold time is greater than setup time). Thus, the calibration of the DFF using a metastable circuit improves the lock time and peak-to-peak jitter performance of the designed CDR system.

Future work involves designing a charge pump and a voltage controlled oscillator at transistor level using 45 nm technology as per the Verilog A model in Cadence Virtuoso. Final work will be the fabrication of the designed CDR system onto a chip.

References

[1] ITRS, "International Technology Roadmap for Semiconductors 2007 Edition: Assembly and Packaging,"*International Technology Roadmap for Semiconductors (ITRS),* http://www.itrs.net, 2007.

[2] B. Razavi, *Design of Integrated Circuits for Optical Communication*, 1st Ed., New York: McGraw-Hill, 2003, Ch. 7-9, pp. 213-329.

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

[4] David J. Rennie, "Analysis and Design of Robust Multi-Gb/s Clock and Data Recovery Circuits," Ph.D. dissertation, Dept. Elect. and Comp. Eng., Univ. Waterloo, ON, 2007.

[5] B. Razavi, *Phase-Locking in High-Performance Systems: From Devices to Architectures*, Piscataway, New Jersey: Wiley-IEEE Press, pp. 294-300, 2003.

[6] Liang Dai and R. Harjani, "Design of low-phase-noise CMOS ring oscillator," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 49, no. 5, pp. 328-338, May 2002.

[7] P. Trischitta and E. Varma, *Jitter in Digital Transmission Systems*, Norwood, MA: Artech House, 1989.

[8] Bellcore TA-NMT-000253 "*Synchronous Optical (SONET) Transport Systems: Common Generic Criteria,*" Issue 6. Sept. 1990.

[9] F.Klass, C.Amir, A.Das, K. Aingaran, C.Truong, R.Wang, A.Mehta, R.Healdand G. Yee, "A New Family of Semi Dynamic and Dynamic Flip-Flops with Embedded Logic for High-Performance Processors," *IEEE Journal of Solid Circuits*, vol.34, no.5, pp.712 – 716, 1999.

[10] M.Rabey, A. Chandrakasan and B. Nikolic, *Digital Integrated Circuits*,2nd Ed., New Jersey, Prentice Hall, pp.332-336, 2003.

[11] H.Patrovi, R.Burd, U.Salim, F.Weber, L.DiGregorio, and D. Draper, "Flow-through latch and edge-triggered flip-flop Hybrid elements,"*in ISSCC Dig. Tech. Papers*, pp. 138-139, Feb. 1996.

[12] A. B. Christian Hansen, "Test and Signaling of a 40Gbps Transmitter/Reciever Prototype," M.S. Thesis, Dept. of IMM., Tech. Univ. Denmark, KongensLyngby, Denmark, 2003.

[13] T. C. Weigandt, "Low-Phase-Noise, Low-Timing-Jitter Design Techniques for Delay Cell Based VCOs and Frequency Synthesizers," Ph.D. dissertation, EECS Department, Univ. California, Berkeley, 1998.

[14] Kundert, K.S. Jri Lee, and B Razavi, "Designing Bang-Bang PLLs for Clock and Data Recovery in Serial Data Transmission Systems," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 9, pp. 1571-1580, Sept. 2004.

[15] M. M. Nejad and M. Sachdev,"A Digitally Programmable Delay Element: Design and Analysis,"*Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol.11, no. 5, pp. 871-878, Oct. 2003.

Appendix

A.1 Verilog A Codes

A.1.1 PRBS-7 Data Generator

```
// VerilogA for Thesis, PRbs7, veriloga

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

module PRbs7(clkp, clkn, outx, outb);

inputclkp, clkn;
outputoutx, outb;
voltageclkp, clkn, outx, outb;
parameter integer bit_num = 8 from [2:32];
parameter integer seed = 1 from [1:inf];

integer x, a1, a2, a3, a4, b, mask;

analog begin
@(initial_step) begin
case (1)
          (bit_num ==  2): begin a1=0; a2= 1; a3= 0; a4= 0; end
// 2 [0,1]
            (bit_num ==  3): begin a1=0; a2= 2; a3= 0; a4= 0; end
// 3 [0,2]
            (bit_num ==  4): begin a1=0; a2= 3; a3= 0; a4= 0; end
// 4 [0,3]
            (bit_num ==  5): begin a1=1; a2= 4; a3= 0; a4= 0; end
// 5 [1,4]
          (bit_num ==  6): begin a1=0; a2= 5; a3= 0; a4= 0; end
// 6 [0,5]
            (bit_num ==  7): begin a1=0; a2= 6; a3= 0; a4= 0; end
// 7 [0,6]
            (bit_num ==  8): begin a1=1; a2= 2; a3= 3; a4= 7; end
// 8 [1,2,3,7]
            (bit_num ==  9): begin a1=3; a2= 8; a3= 0; a4= 0; end
// 9 [3,8]
            (bit_num == 10): begin a1=2; a2= 9; a3= 0; a4= 0; end
//10 [2,9]
            (bit_num == 11): begin a1=1; a2=10; a3= 0; a4= 0; end
//11 [1,10]
```

```
          (bit_num == 12): begin a1=0; a2= 3; a3= 5; a4=11; end
//12 [0,3,5,11]
            (bit_num == 13): begin a1=0; a2= 2; a3= 3; a4=12; end
//13 [0,2,3,12]
            (bit_num == 14): begin a1=0; a2= 2; a3= 4; a4=13; end
//14 [0,2,4,13]
            (bit_num == 15): begin a1=0; a2=14; a3= 0; a4= 0; end
//15 [0,14]
          (bit_num == 16): begin a1=1; a2= 2; a3= 4; a4=15; end
//16 [1,2,4,15]
            (bit_num == 17): begin a1=2; a2=16; a3= 0; a4= 0; end
//17 [2,16]
            (bit_num == 18): begin a1=6; a2=17; a3= 0; a4= 0; end
//18 [6,17]
            (bit_num == 19): begin a1=0; a2= 1; a3= 4; a4=18; end
//19 [0,1,4,18]
          (bit_num == 20): begin a1=2; a2=19; a3= 0; a4= 0; end
//20 [2,19]
            (bit_num == 21): begin a1=1; a2=20; a3= 0; a4= 0; end
//21 [1,20]
            (bit_num == 22): begin a1=0; a2=21; a3= 0; a4= 0; end
//22 [0,21]
          (bit_num == 23): begin a1=4; a2=22; a3= 0; a4= 0; end
//23 [4,22]
          (bit_num == 24): begin a1=0; a2= 2; a3= 3; a4=23; end
//24 [0,2,3,23]
          (bit_num == 25): begin a1=7; a2=25; a3= 0; a4= 0; end
//25 [7,25]
            (bit_num == 26): begin a1=0; a2= 1; a3= 5; a4=25; end
//26 [0,1,5,25]
          (bit_num == 27): begin a1=0; a2= 1; a3= 4; a4=26; end
//27 [0,1,4,26]
            (bit_num == 28): begin a1=2; a2=27; a3= 0; a4= 0; end
//28 [2,27]
            (bit_num == 29): begin a1=1; a2=28; a3= 0; a4= 0; end
//29 [1,28]
            (bit_num == 30): begin a1=0; a2= 3; a3= 5; a4=29; end
//30 [0,3,5,29]
          (bit_num == 31): begin a1=2; a2=30; a3= 0; a4= 0; end
//31 [2,30]
            (bit_num == 32): begin a1=1; a2= 5; a3= 6; a4=31; end
//32 [1,5,6,31]
default $strobe("Error. Should never get here.");
endcase
mask = pow(2, bit_num) -1;
x = seed;
x = x & mask; //mask the unavailable bit;

end
```

```
@(cross(V(clkp, clkn), +1, 1p)) begin
b = ((x>>a1)^(x>>a2)^(x>>a3)^(x>>a4))%2;
x = ((x<<1) & (mask-1)) + b;
end

V(outx) <+ x;
V(outb) <+ b;

end

endmodule
```

A.1.2 Multiplexer

```
// VerilogA for Thesis, Mux, veriloga

`include "constants.vams"
`include "disciplines.vams"
module Mux(Va, Vb, S, Vo);
inputVa,Vb,S; electrical Va,Vb,S;
output Vo; electrical Vo;
realoutv;
analog begin
if (V(S) > 0.5)
outv = V(Va);
else
outv = V(Vb);
V(Vo) <+ transition(outv,0,1f,1f);
end
endmodule
```

A.1.3 Charge Pump

```
// VerilogA for Thesis, CP, veriloga

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

module CP (Up,Dn,Icp, Vdd, Vss);

outputIcp; electrical Icp;     // current output
inputUp,Dn; electrical Up,Dn;
inoutVss, Vdd; electrical Vss,Vdd;
electricalrst;
```

```
//electrical subb;


parameter real icpn=1u;       // maximum sinking current
parameter real vth = 0.5;
parameter real icp= 800e-6/2/3.14;
realsubb, iout;

analog begin

subb = V(Up)-V(Dn);
iout = icp*subb;

I(Icp)<+ transition(iout);

end
endmodule
```

A.1.4 Voltage Controlled Oscillator

```
// VerilogA for Thesis, VCO, veriloga

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

module VCO(Vc,Out, Vss, Vdd);

inputVc; electrical Vc;
output Out; electrical Out;
inoutVss, Vdd; electrical Vss,Vdd;
parameter real f0 = 2.75e9;
parameter real Kvco = 500e6;
real f, amp, offset;

analog begin

f = f0 + Kvco*V(Vc);
amp = (V(Vdd)-V(Vss))/2;
offset = V(Vss)+amp;
V(Out) <+ amp*sin(2*`M_PI*idtmod(f,0,1))+offset;

end

endmodule
```

A.1.5 Slicer

```
// VerilogA for Thesis, SLICER, veriloga

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

module SLICER(in, out,out_b);

input in; electrical in;
outputout,out_b; electrical out,out_b;

parameter real vth = 0.5;
realoutv, outvb;

analog begin

if (V(in) > 0.5)
begin
outv = 1;
outvb = 0;
end
else
begin
outv = 0;
outvb = 1;
end

V(out) <+ transition(outv,0,1p,1p);
V(out_b) <+ transition(outvb,0,1p,1p);

end
endmodule
```

A.2 Transistor Sizes

A.2.1 Semi dynamic DFF (SDFF)

Table 0.1: Transistor sizes of the designed SDFF .

Transistor	Width (µm)	Length (nm)
M_1	4	45
M_2	10	45
M_3	10	45
M_4	10	45
M_5	15	45
M_6	2	45
M_7	2	45

A.2.2 Exclusive OR gate (XOR)

Table 0.2: Transistor sizes of the designed XOR gate.

Transistor	Width (µm)	Length (nm)
M_1	12	45
M_2	12	45
M_3	12	45
M_4	12	45
M_5	11	45
M_6	11	45

M_7	11	45
M_8	11	45

A.2.3 Inverter

Table 0.3: Transistor sizes of the designed inverter.

Transistor	**Width (nm)**	**Length (nm)**
M_1	120	45
M_2	180	45

A.2.4 Data delay cell

Table 0.4: Transistor sizes of the designed data delay cell.

Transistor	**Width (µm)**	**Length (nm)**
M_1	2	45
M_2	2.5	45
M_{n0}	0.18	45
M_{n1}	0.6	45
M_{n2}	1.2	45
M_{n3}	0.6	45
M_{p0}	0.25	45
M_{p1}	0.86	45

M_{p2}	1.72	45
M_{p3}	0.86	45

A.2.5 Clock delay cell

Table 0.5: Transistor sizes of the clock delay cell.

Transistor	**Width (µm)**	**Length (nm)**
M_1& M_1’	0.12	400
M_2& M_2’	0.12	180
M_3& M_3’	0.18	45
M_4& M_4’	0.6	45
M_5& M_5’	2	45
M_6& M_6’	0.6	45
M_7& M_7’	6	45