CIRCUIT ARCHITECTURES FOR HIGH SPEED CMOS CLOCK AND DATA RECOVERY CIRCUITS

BY

SABAREESHKUMAR RAVIKUMAR

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Professor José Schutt-Ainé

ABSTRACT

As semiconductor process technologies continue to scale and the demand for ubiquitous computing devices continues to grow with paradigms such as the internet of things (IOT), the availability of low-cost, low-power, high-speed and robust communication interfaces between these devices will be a major challenge that needs to be addressed. Even in traditional desktop computing devices, the off-chip bandwidth does not scale as fast as the on-chip bandwidth and has therefore been an important bottleneck to the growth in processing speed. Thus, intelligent techniques will have to be developed that allow the traditional lossy channels to be deployed at higher data rates, while minimizing cost and power, without paying much of a performance penalty.

Over the last decade and a half, a great amount of research has been done to design monolithic transmitter and receiver integrated circuits (ICs) in silicon complementary metal-oxide semiconductor (CMOS) technology as opposed to traditional discrete SiGe, InP technologies owing to the low cost and ease of integration of CMOS technology. A key component of the receiver is the clock and data recovery (CDR) circuit, which extracts the clock from the incoming data stream and samples the data. The performance of the CDR is a major impediment to increasing data rates in a serial communication system. Several CDR architectures have been proposed to ensure that the performance is comparable to traditional discrete SiGe, InP devices.

In this thesis, three different CDR circuit architectures are designed in a 180 nm CMOS process with a target data rate of 2 Gbps and compared in terms of performance, power and area. In order to provide a fair comparison, the corresponding channel and transmitter blocks are also designed and the entire serial communication link is simulated. The fundamentals of CDR circuit design are introduced and a complete guide to analysis and design of CDR circuits for high speed serial links is presented. The results of the

comparison help to evaluate power, performance and area trade-offs during the design phase and to choose the right architecture for a given application.

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

ACKNOWLEDGMENTS

As I approach the twilight of my life as a graduate student, I can look back and reflect on the numerous people who have made this journey not only possible but also very memorable. Although I am the sole author of this thesis, this work represents the culmination of efforts, ideas, encouragement and motivation from several people.

Firstly, I would like to extend my gratitude to my adviser Professor José Schutt-Ainé for being my mentor and a constant source of inspiration throughout my graduate life. He has taken utmost pains to ensure that I had every resource available, including the complete freedom to pursue my ideas for my research. He has displayed seemingly endless patience during every one of my countless visits to his office and always pushed me to achieve nothing short of the best. Dr. Schutt-Ainé has been a father-figure to me during my entire graduate life and is someone I will look up to as a role model as I begin my professional life.

Secondly, I would like to convey my heart-felt thank you to Professors Elyse Rosenbaum and Martin Wong. Professor Rosenbaum handled two important courses for me which not only spurred my interest in the area of circuits but also proved vital for my research and career. She always found the time to answer my endless stream of pesky questions with great patience and vigor. Professor Martin Wong taught the Introduction to VLSI course in my first semester as a graduate student, which imbued in me a passion to pursue a career in the field. He was also extremely benevolent in writing me excellent letters of recommendation that proved invaluable in my securing a graduate fellowship.

Thirdly, I would like to thank the Bahl family for instituting and supporting the Joan and Lalit Bahl ECE graduate fellowship. The fellowship provided me with not only ample financial support, but also the complete freedom to pursue my interests, which is invaluable in graduate school. I will

forever be indebted to the family and ECE Illinois for this wonderful gesture.

Further, I would like to extend my thanks to my fellow graduate students in Professor Schutt-Ainé's research group who have stood by me, provided constructive criticism of my work and pushed me towards achieving nothing short of excellence. Specifically, I would like to thank Xu Chen for his help with the channel designs, Yubo Liu and Da Wei for maintaining the group servers and also Rishi Ratan for introducing me to this wonderful research group. I would be remiss if I did not acknowledge two wonderful undergraduate mentees, Siyue Li and Yi Ren, for their excellent work on the serializer and deserializer blocks.

In addition to the fine students mentioned above, I would like to acknowledge the role of Sundaravadivel Rajarajan, who in the capacities of a fellow graduate student, teaching assistant and friend, has helped me through several issues and answered arcane questions pertinent to my research. I am also eternally grateful to my friends Ryan Steinbach, Yashwant Divakar, Lakshmi Rao, Aparna Uvaraj, Preethi Raghunathan, Arvind Pattabhiraman, Robin Jeptah, Rajavasanth Rajasegar and Aniruddh Rangarajan for supporting me throughout my graduate life. I am positive that I missed some names in the above list and I ask you as a good friend to forgive my ungratefulness!

Lastly, I would like to thank my family: my parents, grandfather, cousins and others, who have instilled in me a thirst for learning, believed in my abilities and motivated me to pursue graduate studies. Even when the chips were down, they have stood by my side and encouraged me at every step of the journey.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation

The information age is here, as is confirmed by the sheer magnitude of data on the internet. As of 2013, it was estimated that the total volume of data on the internet was 4.4 zettabytes (1 ZB = 10^{21} bytes = 1 billion terabytes) and this is expected to multiply tenfold by 2020 thanks to the increasing number of computing devices and new landscapes such as the internet of things (IoT) [1]. It is also estimated that about 640 TB of data is transmitted every minute on the internet [2]. A driving force behind this data explosion is the scaling and advancement of semiconductor fabrication technology (SFT) which has dramatically reduced the cost of computing and storage devices. Figure 1.1 [3] highlights the trend, by showing the cost per gigabyte of hard disk memory over the years.

Another key enabler of the information age is the steady increase in broadband access speeds due to advancements in fiber-optic communication systems. Figure 1.2 shows growth in average internet connectivity speeds of end users in the United States [4]; clearly, we are at the cusp of the gigabit internet era with some providers such as Google fiber providing fiber-optic access lines to end-users.

While the internet backbone gets faster and massive data centers pile on gargantuan amounts of data, the interface between computing devices and the internet, such as network switches and processor-memory interface across backplane channels, has been growing at a much slower pace, leading to a bottleneck in computation speed. Figure 1.3 shows the trends in data rate scaling of high speed input-output (IO) signaling links as forecasted by the International Solid State Circuits Conference (ISSCC) 2011 [5]. The important takeaway from the graph is that the data rate in inter-IC communication

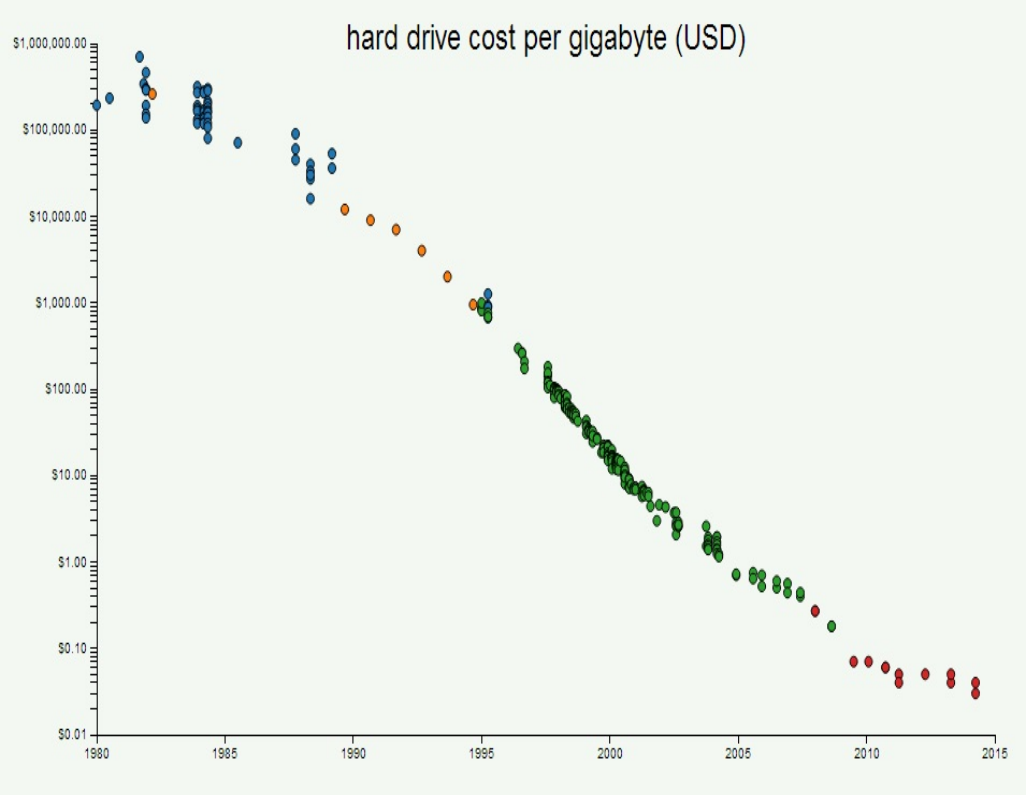

Figure 1.1: Cost per gigabyte of hard disk memory trend

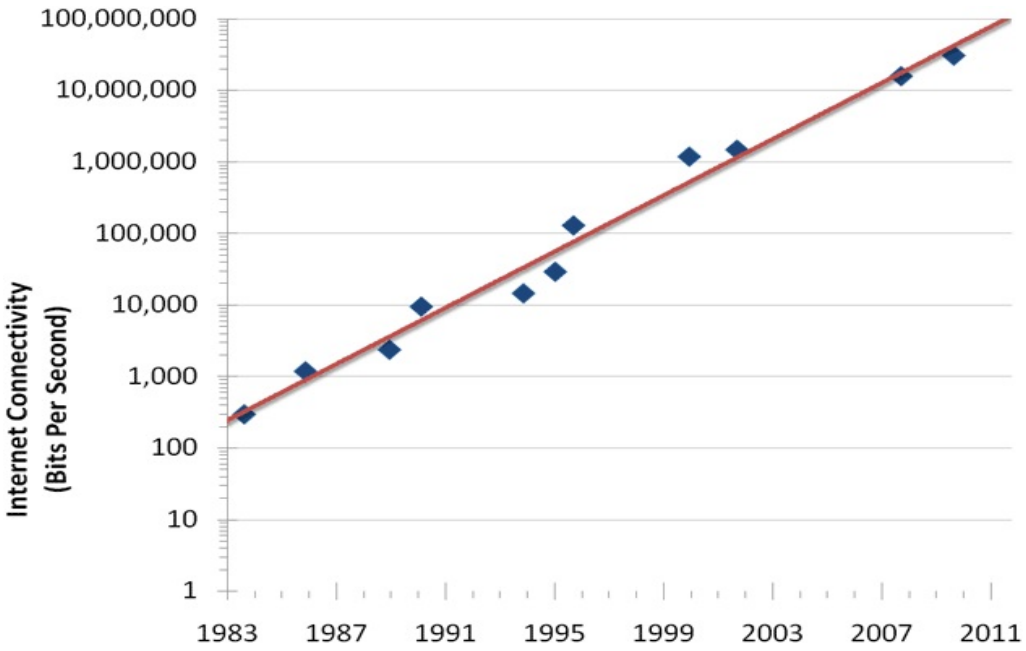

Figure 1.2: Average end-user internet speeds in the United States

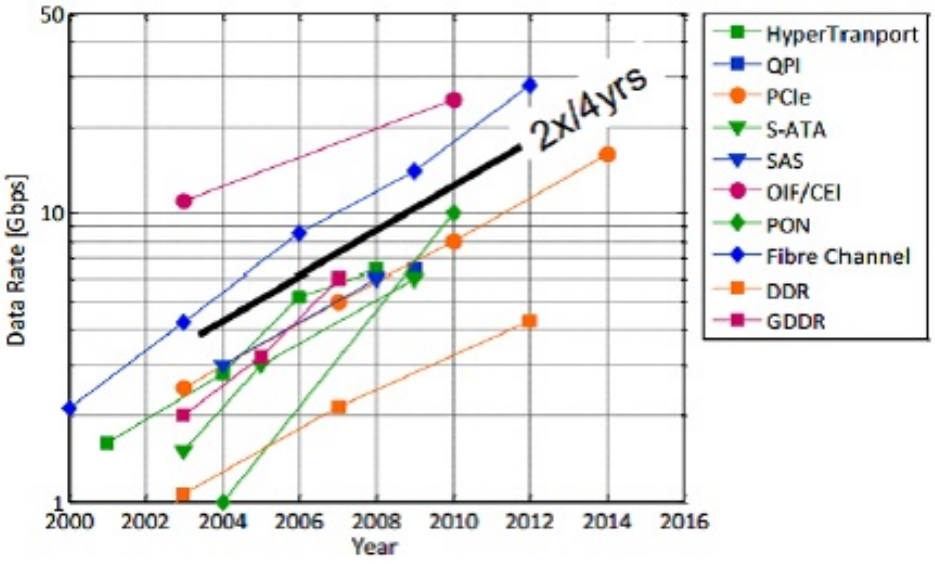

Figure 1.3: IO signaling data-rate trends

links is growing by a factor of 2X every 4 years, which is not only slower than the growth in internet speeds but also slower than Moore's law [6], the trend for semiconductor devices which predicts a 2X increase in the number of on-chip transistors every 24 months.

Note that while the data rates in inter-IC communication links have been increasing, the channel bandwidth remains the same. The increase in data rate mandates an increase in the frequency of operation for the transmitting and receiving circuits to the multi-gigahertz range and this leads to several complications such as increased transmission line losses, cross-talk and, as a result, intersymbol interference (ISI). This makes the circuits extremely sensitive to timing uncertainties, and so the process of designing robust, low-jitter CDR circuits is extremely arduous.

Another motivation is to evaluate a number of novel circuit architectures that have been proposed for CMOS CDR circuits in terms of their performance, power and area metrics in order to identify the way forward in designing high speed serial interfaces. These architectures include static CMOS, complementary logic and current mode logic (CML) as well as channel configurations such as single-ended and differential designs.

1.2 Thesis outline

The goal of this thesis is to provide a comparison between three different circuit architectures for CDR circuits. In addition, it aims to provide a reference manual for designing current mode logic circuits for clock and data recovery circuits. The thesis is organized as follows:

- Chapter 1 provides an introduction to the research problem, describing the need for high-speed serial links and providing the motivation for this thesis.
- Chapter 2 provides an overview of high-speed links with an emphasis on describing each of the building blocks, the figures of merit to characterize these links and the motivation behind the industry-wide shift from parallel to serial-link design for low power, cost-effective robust I/O link design.
- Chapter 3 provides a strong mathematical framework to analyze the CDR and to arrive at the optimal loop parameters.
- Chapter 4 describes the behavioral modeling of the CDR circuit using Verilog-AMS to model the various building blocks.
- Chapter 5 describes the design of the single-ended CDR architecture.
- Chapter 6 describes the design of the complementary CDR architecture.
- Chapter 7 describes the design of the current mode logic CDR architecture.
- Chapter 8 presents the results obtained from the simulation of the three CDR architectures and also provides a comparison between the architectures.
- Chapter 9 concludes the thesis with a discussion of why current mode logic is a superior circuit architecture for high speed applications and also suggests some potential future work in this area.

CHAPTER 2

HIGH SPEED SERIAL LINKS OVERVIEW

In this chapter, an overview of the various components involved in high speed serial links is provided. We begin with a discussion of why serial links are preferred to parallel links and this is followed by a detailed description of a typical serial link. Next, we describe the various blocks of the SERializer DESerializer (SERDES) system. Finally, some performance metrics for high speed serial links are discussed.

2.1 Serial vs. parallel links

Traditional IO buses have been based on parallel links such as the IDE (integrated drive electronics), PCI (peripheral component interconnect) and AGP (accelerated graphics port) interfaces. These interfaces required one physical conductor for each bit of the transmitted data word, resulting in wide data buses that were usually limited in speed to less than 100 Mb/s. High performance interconnects were limited to high-end work stations such as the Cray supercomputer [7]. In the past decade and a half, increasing microprocessor clock frequencies, the move to multicore processors and paradigms like graphics processing units (GPUs) and system on chips (SoCs) have mandated much faster access to data than ever before. One way to increase the bandwidth of a parallel link is to increase the number of conductors. However, this approach is prohibitive and at some point serial links become more attractive [8]. This has led to the development of several new interface standards based on serial links such as PCI-express (PCI-e), Serial ATA (SATA) and RapidIO, all of which can be commonly found in any desktop computer today.

The move to serial links was only natural since, historically, several high speed links such as fiber-optic and co-axial cables have operated serially owing

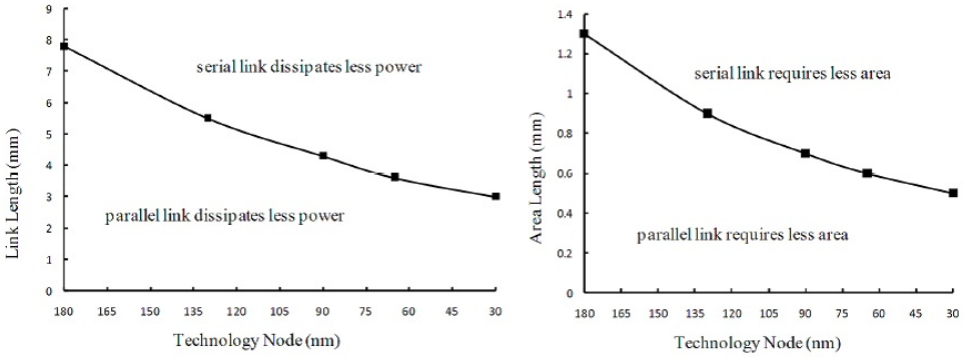

Figure 2.1: Comparison of serial vs. parallel links for different CMOS technology nodes

to cable cost and synchronization difficulties with increasing transmission speeds and distances. In addition to increasing the IO bandwidth, serial links are also efficient in terms of cost, area and power. They also eliminate several problems faced in parallel links such as crosstalk, data skew and clock transmission.

A study conducted by Dobkin [9] compared serial and parallel links in terms of their area and power across several CMOS process nodes with different feature sizes and the results are illustrated in Figure 2.1. The key takeaways from the comparison are:

- For a given CMOS technology, there is a limiting value of link length beyond which serial links are superior to parallel links in terms of power and area.

- The limiting value discussed above also scales down correspondingly as the feature size of the CMOS technology scales down.

Therefore, as semiconductor fabrication technology continues to scale down, serial links become more and more beneficial as compared to parallel links. Another important point to note is that as the feature size scales down, the supply voltages have also scaled down whereas the voltage levels required by the legacy parallel bus have not scaled proportionately [10].

In addition, a serial link greatly reduces the number of printed circuit board (PCB) traces on the motherboard as well as the number of IO pins required by the processor. This has several benefits such as easier package design for the microprocessor IC and better PCB design since a single trace

occupies much less area and hence can be isolated better. Serial links do not require the transmitter clock to be forwarded along with the data, thereby saving an extra trace/pin and also eliminating the effects of clock skew that are found in parallel links. At today's transmission speeds, the tolerance for data skew between the various conductors of a parallel link is extremely low and has reached the practical limit for PCBs using FR4 substrates. Also, the capacitive-inductive coupling between the multiple conductors on a parallel link leads to severe cross-talk effects and causes signal integrity issues. This problem is overcome in serial links by using only one conductor and providing sufficient isolation.

Thus, serial communication has become the solution to higher and more efficient data transmission in order to meet the demands and trends of higher capacity of communication technology [11].

2.2 A simple serial link

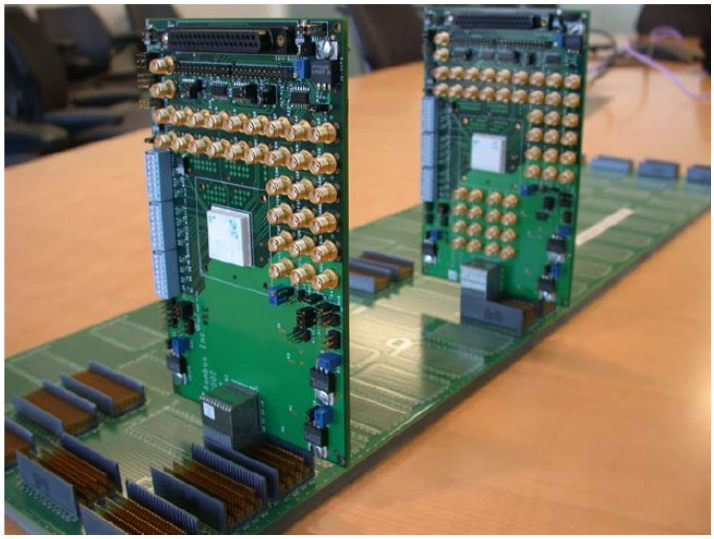

Figure 2.2: A backplane trace between two line cards

Several serial links are used today such as fiber-optic cables, co-axial cables, LAN cables and backplane PCB traces. Since the focus of this thesis is high speed computing interfaces, we look at an example of PCB traces between two line cards on a backplane shown in Figure 2.2 [12]. Such backplanes are becoming extremely common in today's big data servers and large routers.

Line cards are used to communicate to the external world through either fiber-optic or LAN cables. High speed SERDES chips receive data from a line card and communicate it to the switch card which directs the input stream to the correct line card depending on its address.

In this system, the chips are mounted on packages which are then soldered to the line card. The line cards connect to the backplane using through-hole connectors. The backplane has a number of traces which connect the line cards and switch cards to each other. A cross-section of the system is shown in Figure 2.3 that shows the complete signaling path [13].

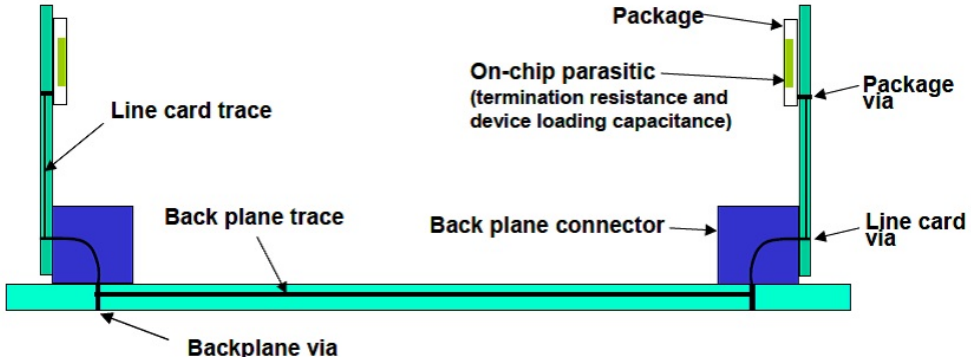

Figure 2.3: A cross-sectional illustrated view of the backplane trace between two line cards

In order to intelligently design the SERDES (SERializer DESerializer) system, it is essential to understand the nature of the channel and the various imperfections that it suffers from. The circuit designer does not have a lot of control over the channel as the channel design is often performed by system level engineers. However, the circuit designer should be able to accurately predict the behavior of the channel and this is often accomplished by channel models. One of the most commonly used channel models is the S parameter model. The S parameters are frequency domain parameters that can be used to completely characterize the channel response in the time domain. The S parameters can be obtained by actual measurement in the laboratory using a vector network analyzer (VNA). The S parameters can also be obtained through numerical simulations of the channel geometry using electromagnetic field solvers such as Q3D or ANSYS HFSS.

Once the S parameters of the channel are known it is possible to estimate the amount of loss, intersymbol interference, crosstalk and jitter in the data

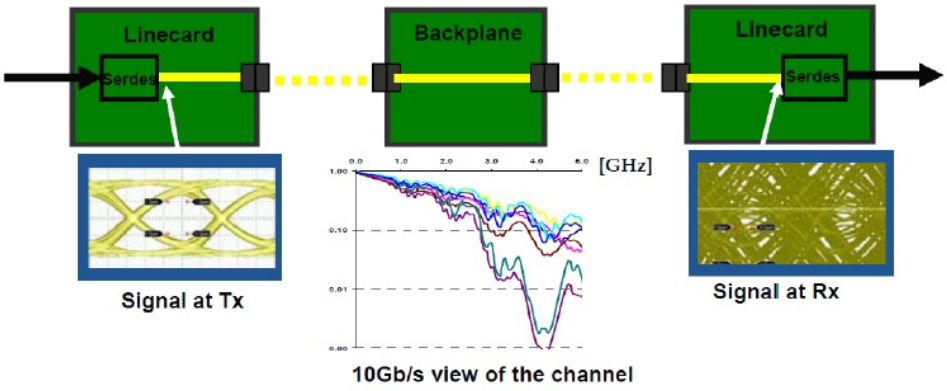

Figure 2.4: Signal distortion due to channel imperfections

stream that arrives at the receiver. These factors affect the timing window in which the receiver must sample the data and also how sensitive the receiver needs to be. For example, Figure 2.4 [14] shows how the clean data from the transmitter is corrupted by the channel when operating at 10 Gb/s. Thus, it is the job of the circuit designer to design a high fidelity receiver that can correctly sample the distorted data stream with minimal power and area, while ensuring that the number of errors is infinitesimally small (about one in every 10^{12} bits).

2.3 SERDES building blocks

The SERDES system refers to the complete assembly of transmitter, channel and receiver that constitute the high speed serial link. A typical block diagram of the SERDES system is shown in Figure 2.5. We will now discuss each of the blocks in detail.

2.3.1 Serializer

It is important to note that most computer data is in the form of words of some length of bits: usually a power of 2 such as 16, 32, 64, etc. Thus, the input to the SERDES system is a set of bitlines which are parallel in nature, i.e., the bitlines are synchronous with each other. Every clock cycle, a new word arrives on these parallel lines and the information on all of these

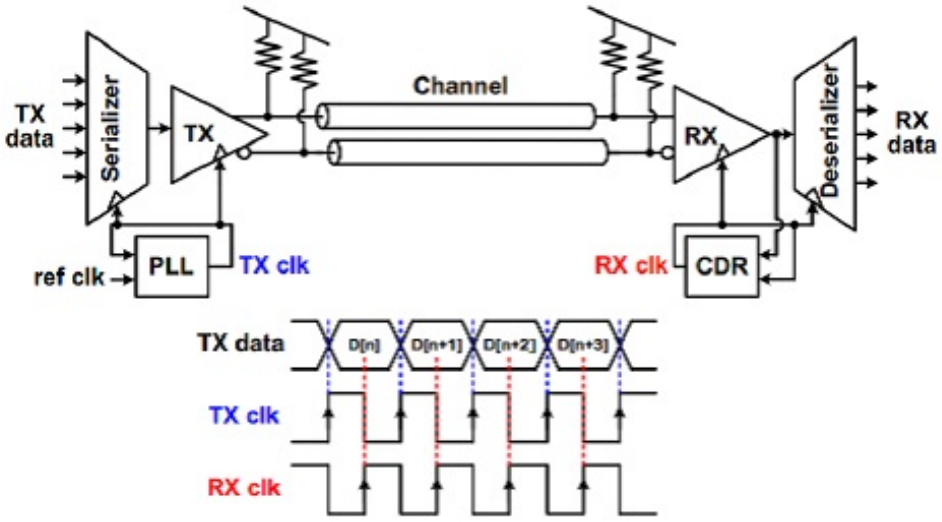

Figure 2.5: A typical SERDES system

lines must be transmitted before the next clock cycle, when a new data word would arrive. This functionality is carried out by the serializer blocks. Thus, a serializer converts a parallel stream of data into a serial stream, suitable for transmission over a high speed serial link. Depending on the number of bits serialized, the serializer is termed as $2^N : 1$ serializer, where 2^N represents the data word length, i.e., the number of input lines.

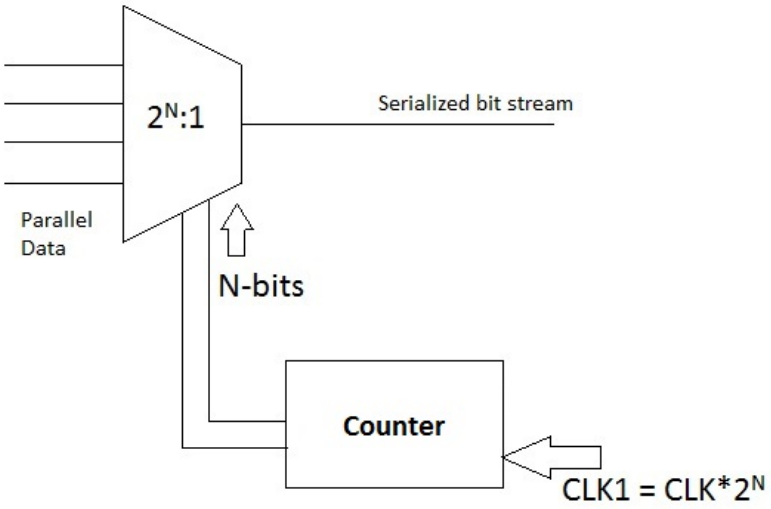

Figure 2.6: A simple multiplexer-based serializer

One possible implementation of the serializer is a $2^N : 1$ multiplexer, with

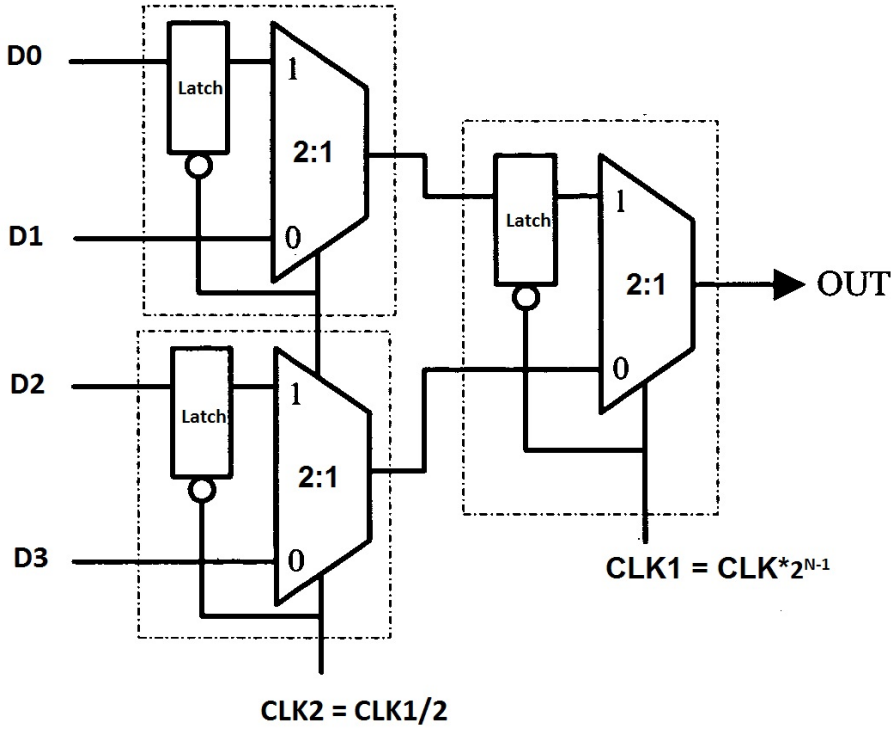

Figure 2.7: A binary tree-based serializer

the select lines configured to change 2^N times per clock period, i.e., the select lines can be considered to be the output of an N-bit counter which works at 2^N times the clock frequency. This implementation is shown in Figure 2.6. There are several problems with this approach:

- A new clock will have to be generated with a frequency of 2^N times the clock frequency. For the case of N = 3, and a 1 GHz system clock, we need to generate a new clock at 8 GHz. Designing phase locked loops (PLLs) to work at such a high frequency is often cumbersome. Also, as N becomes bigger, the problem blows up exponentially.

- Even if we successfully generate the new clock, we would still have to design the 2^N : 1 multiplexer fast enough to transmit a new data every $T/2^N$ seconds, where T is the system clock period. For the previously described example, this would come down to about 125 ps. Designing multiplexers with such small delays could be extremely expensive and sometimes even impossible, as is the case when N becomes larger.

- Working at such a high frequency drastically increases power consumption since the dynamic power is directly proportional to the frequency. The problem is only made worse by the circuits being large to accommodate for short delay targets.

In order to overcome the above issues, a tree-based topology is used [15]. In this topology, the large 2^N : 1 multiplexer is divided into N stages of 2:1 multiplexers, with the final stage operating at 2^{N-1} times the clock frequency and each predecessor stage operating at one half of the frequency of its successor. This distributed approach reduces the delay targets for each stage and thus allows much smaller circuits. In addition, since a large part of the design operates at much lower than peak frequency, there is a great reduction in power consumed. The tree-based design is shown in Figure 2.7 and is the most commonly used topology today. Latches are used to hold the data between stages.

2.3.2 Driver amplifier

The driver amplifier is used both at the transmitter and the receiver. At the transmitter, it is used to amplify the input serial bit stream before it is sent through the channel. Sometimes, a pre-emphasis block is included in the transmit driver amplifier to boost certain components of the signal which are liable to face high attenuation. Another important purpose served by the transmit driver amplifier is to provide impedance terminations to terminate the channel with its characteristic impedance which is typically 50 Ω. This eliminates reflections in the channel and improves the integrity of the transmitted signal.

2.3.3 Phase Locked Loop (PLL)

A PLL is a negative-feedback system whose purpose is to take an input reference clock with frequency f_{in}, and produce an output clock with frequency f_{out} , such that $f_{out} = \alpha f_{in}$, where $\alpha(> 1)$ is the multiplication factor. PLLs are needed since crystal oscillators can provide a high spectral purity reference clock only up to a frequency of about 200 MHz. At the microwave frequencies, crystal oscillators produce intolerable amounts of jitter which

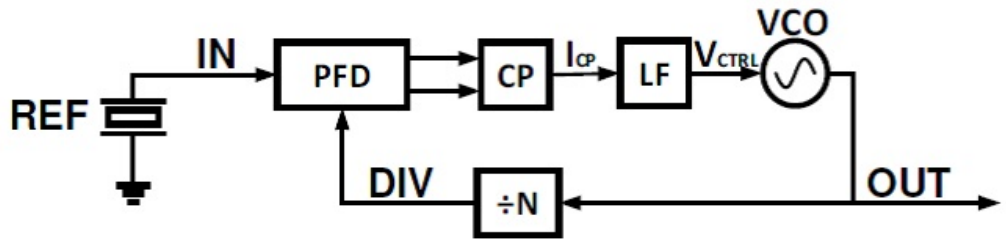

Figure 2.8: Block diagram of a typical PLL

render them unusable. Thus, the most important function of the PLL is to produce clock signals with minimal timing noise, i.e., jitter (in time domain) and phase noise (in frequency domain). Figure 2.8 shows the typical block diagram of a PLL system. The phase-frequency detector (PFD) compares the frequency and phase of the divided version of the generated clock with the reference clock. The PFD produces pulse width modulated (PWM) outputs which are used to drive a charge pump (CP). Depending on the PWM signals, the CP pumps or drains charge into/from the capacitor in the loop filter (LF). The LF is usually second order low pass filter (LPF), which filters out the high frequency components in the output of the PD and provides it to the voltage controlled oscillator (VCO). The VCO produces a clock waveform whose frequency is proportional to the control voltage applied. The divider divides the frequency of the generated clock by a factor N, where $N = \alpha$. Thus, the steady state of the system is one in which the generated clock has a frequency of αF_{in} and is exactly in phase with the reference clock.

2.3.4 Channel

The channel is the physical medium that transports the signals from the transmitter to the receiver. As discussed previously, a good example of a serial link is the channel between two line cards on a backplane. Figure 2.9 [12] shows the S-parameter S_{12} (attenuation) of the various components of the channel. An important observation is that the attenuation gets worse as the frequency increases. Thus, as data rates increase, the degradation suffered by the signal gets worse. A good way to visualize the effect of the channel in the time domain is to look at eye diagrams. An eye-diagram is a synchronized superposition of all possible realizations of the signal of interest,

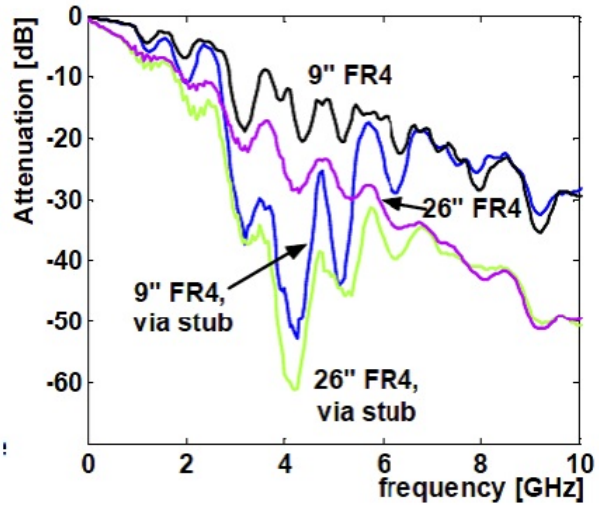

Figure 2.9: Frequency response of the various components of the channel

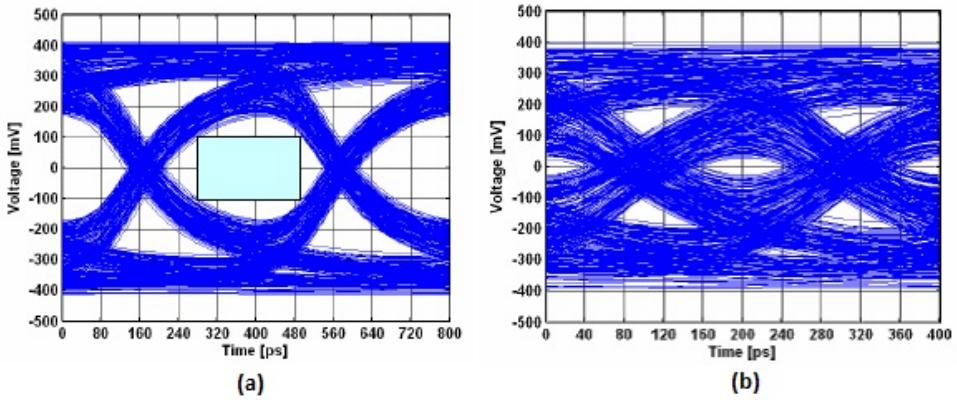

Figure 2.10: Eye diagrams at the output of the channel for operation at (a) 2.5 Gb/s and (b) 5 Gb/s

viewed with a particular signaling interval. Figures 2.10(a) and 2.10(b) show the eye diagrams at the output of the channel when operating at 2.5 Gb/s and 5 Gb/s [16].

Clearly, for the first case, the eye is sufficiently open, i.e., we have sufficient voltage and timing margin to detect 0's and 1's accurately. For the second case, the eye is almost completely closed, and the window to sample the data is very narrow, with the difference in amplitudes between 0's and 1's being negligible. This leads to gross bit errors in the receiver, which is generally unacceptable in high speed serial links. The above effect can be attributed to several factors such as attenuation, dispersion, reflections and ISI, all of

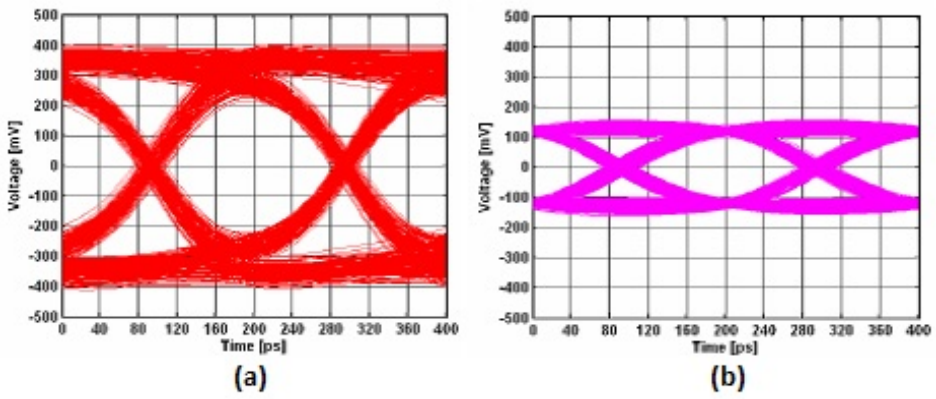

Figure 2.11: Equalized eye-diagrams for the case of (a) boosting high frequencies and (b) suppressing lower frequencies

which become even more pronounced at higher frequencies. Thus, the circuit designer has to accommodate these imperfections in the design, which makes the SERDES design even more challenging at higher frequencies.

2.3.5 Equalizer

As discussed in the previous section, channel behavior distorts the transmitted pulses, leading to increasing bit errors as the data rates increase. One of the circuit techniques used to combat this behavior is equalization at the receiver. A commonly used type of equalization is to provide a response that directly compensates for the channel's frequency response. Since we know that the channel attenuation is larger at higher frequencies, one possible equalization solution is to boost the higher frequency components which have been severely attenuated. Another solution is to suppress the lower frequency components without altering the higher frequency components. The effect of both techniques on the 5 Gb/s eye diagram shown previously is presented in Figures 2.11(a) and 2.11(b) respectively [16].

2.3.6 Clock and Data Recovery (CDR) circuit

In high speed serial links, the transmitter clock is not forwarded. Instead, the clocking information is embedded in the transmitted data stream and the receiver is expected to extract the clock from the received data and use it to sample the data stream. A CDR circuit serves this purpose and a generic

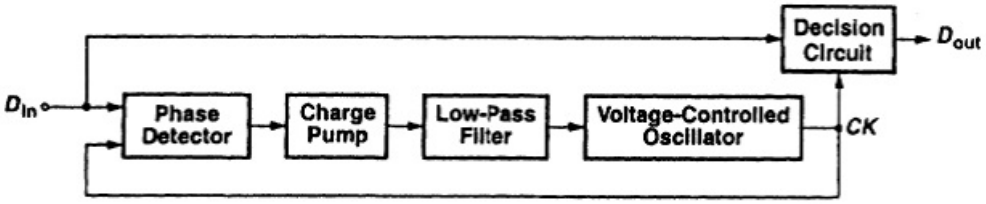

Figure 2.12: Block diagram of a typical CDR system

block diagram of the same is shown in Figure 2.12. Comparing the block diagrams of the CDR and the PLL, we find that they are similar in several aspects. However, a CDR is typically not used for frequency multiplication. It is used to extract the clock from the received data at just the right frequency and phase to sample the data optimally. Thus, the phase detector for the CDR is significantly different from that of the PLL. While the PLL PD is used to ensure that the phase difference between the reference and generated clock is zero, the CDR PD is used to ensure that the phase difference is at a constant value that allows optimal sampling, i.e., the center of the eye. The other blocks in the CDR are very similar to that of the PLL, although there is no divider in CDR or in other words, the generated clock is divided by 1. Once the clock is extracted, we can use a decision circuit such as a sense amplifier flipflop to sample the incoming data stream. We will explore the various blocks in more detail in the following chapters as the CDR design is the focus of this thesis.

2.3.7 Deserializer

The deserializer performs the complementary function of the serializer. Since the data at the receiving end needs to be processed in terms of words of a specific length, it is important to convert the serial data stream back to its native parallel form. This function is served by the deserializer. Thus, in its most basic form, the deserializer is a $1 : 2^N$ demultiplexer. However, as discussed in the serializer section, this topology has several disadvantages and therefore a tree-based deserializer topology is preferred.

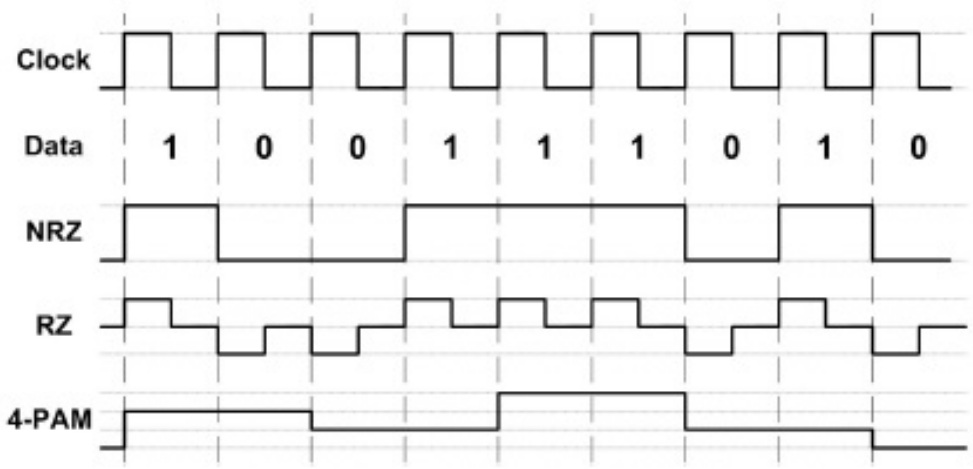

Figure 2.13: Different signaling schemes used in serial links

2.3.8 Encoding and signaling

Encoding refers to the process in which the data to be transmitted is mapped onto a different set of data in a reversible manner. This is done since the new set of data has several advantages over the previous one. For example, one of the purposes of encoding is to ensure that there are no long streams of consecutive 0's or 1's since the CDR ideally needs a transition every clock cycle to work perfectly. Encoding can also allow for DC balance by maintaining a roughly equal number of 0's and 1's. Another important purpose served by encoding could be error detection and correction. Whatever the reason for encoding, it is also critically important that the process is reversible by a decoder at the receiver in order to reclaim the original data. A commonly used encoding scheme is the 8B/10B scheme, where 8 bits of incoming data are mapped to 10 bits of the output data which allows a reduction in the bit error rate (BER).

In addition to encoding, it is also important to describe how the binary digits of 0's and 1's are electrically represented in the channel, and this process is known as signaling. The most commonly used signaling protocol is the non-return to zero (NRZ) signaling where a 1 is represented by a constant high voltage and a 0 is represented by a constant low voltage. Other signaling techniques, such as the PAM-4, allow a reduction in bandwidth by utilizing 4 different voltage levels. Some of the commonly used signaling techniques are shown in Figure 2.13. It is important to remember that a complex signaling

technique requires an equally complex receiver design.

2.3.9 HSSL figures of merit

The performance of a HSSL system depends on both the channel characteristics as well as the circuit design. As data rates steadily climb up, the channel degrades the data even more severely, and it is becoming an increasingly complex task to design HSSL systems. A key concern for HSSL systems is robustness, and several metrics are used to characterize the link including bit error rate (BER), jitter and cross-talk [17].

BER in modern HSSLs is typically between 10^{-12} and 10^{-15} and it is the main metric used to gauge the integrity of the received data. A BER of 10^{-12} implies that there is only a single bit error when receiving 10^{12} bits. Measurement/simulation of BER is one of the major challenges faced by designers, since in order to accurately conclude that a link has a BER of 10^{-12}, we would have to actually transmit/simulate 10^{12} bits, which is almost impossible in state-of-the-art simulators/equipment. Therefore, most simulators use statistical methods to collectively analyze the effects of various deterministic noise sources such as ISI, supply-noise, timing jitter, etc., as well as random noise sources such as white-thermal noise and random jitter when estimating the BER.

Another metric that is often used is eye-diagram masks. Each link standard has a certain characteristic mask, such as the OC-48 mask used for SONET (Synchronous Optical NETworks) systems which is shown in Figure 2.14. The eye diagram needs to have a predetermined width, height, jitter, SNR, etc., to meet the mask specifications and any HSSL system that meets these specifications will be compatible with the other links in the system.

Finally, the last major metric in calculating the timing margin of a HSSL is the jitter. Characterization of deterministic as well as random timing jitter in a clock output is very important to a link designer. Essentially, jitter is the time-domain variation in the clock-signal as shown in Figure 2.15 [18]. A commonly used method for jitter calculation is to close either side of the eye horizontally by the amount of peak clock jitter. While this method can be helpful in evaluating the effects of jitter at the receiver end, it is often an overly optimistic approximation of noise margin degradation for transmitter

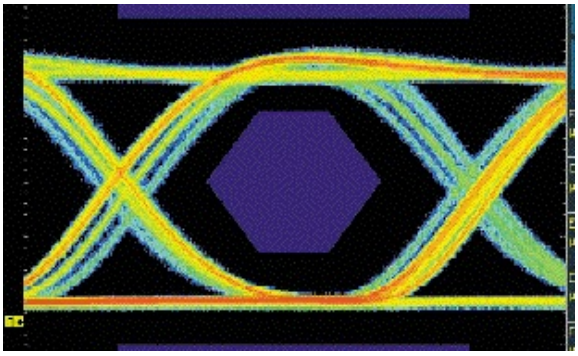

Figure 2.14: The OC-48 eye mask

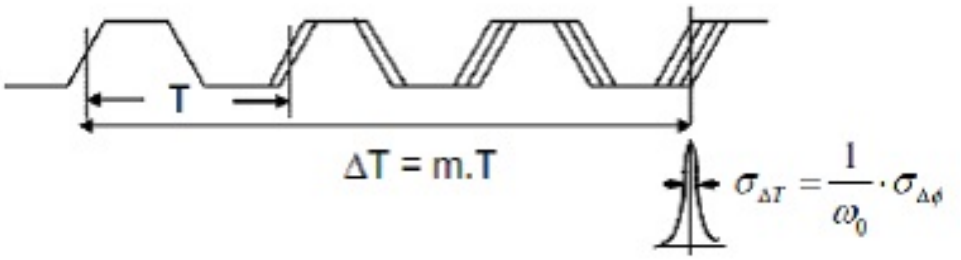

Figure 2.15: Timing jitter example

jitter. Due to the need for integration of clock generators such as PLLs in large digital chips, clock jitter is dominated by power-supply and substrate noise, neither of which scales with technology. Therefore, as data rates increase, bit-periods become shorter and the performance of multi-gigabit links will be limited by the clock jitter, thereby initiating the importance of accurately analyzing the effects of clock jitter on high-speed serial links. Figure 2.16 provides a summary of common jitter profiles in a typical serial link.

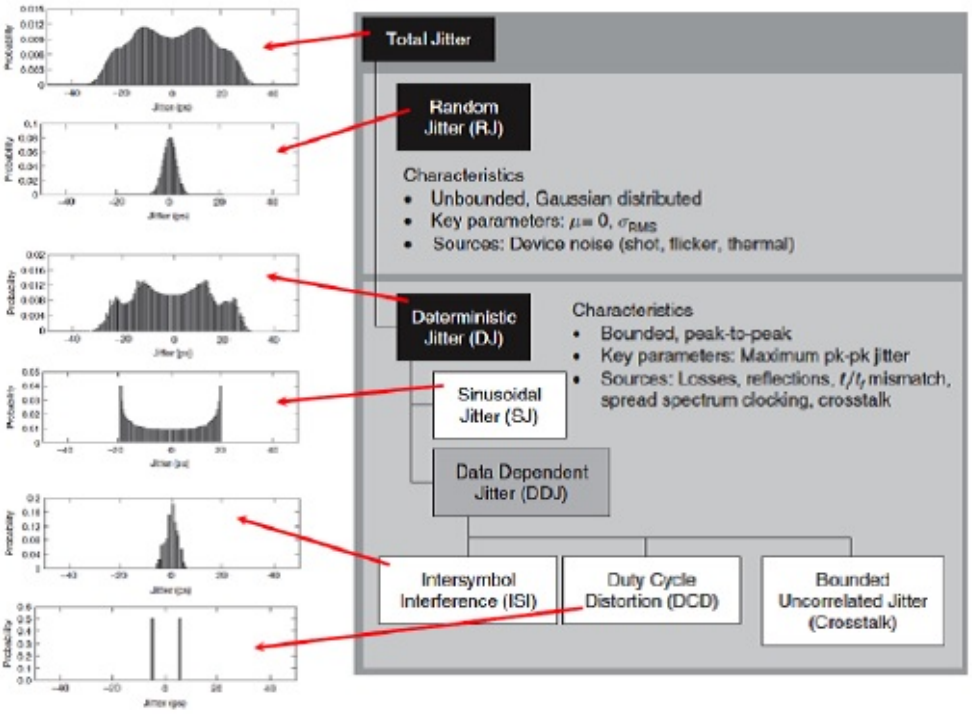

Figure 2.16: Summary of common jitter profiles

CHAPTER 3

CDR THEORY AND ANALYSIS

The clock and data recovery (CDR) circuit is the most important component of the receiver and probably the most critical one in terms of performance. Over the years, a tremendous amount of research has gone into the design and analysis of this delicate block that has a profound effect on the performance of a serial link. Along with channel imperfections, the performance of the CDR is a key factor that limits the data rates in a HSSL. In this chapter, we begin with a detailed look into the various components of the CDR block and conclude with the rigorous analytical treatment of the same.

3.1 CDR building blocks

The block diagram of the CDR system is shown in Figure 3.1. We now look at each of the blocks in detail.

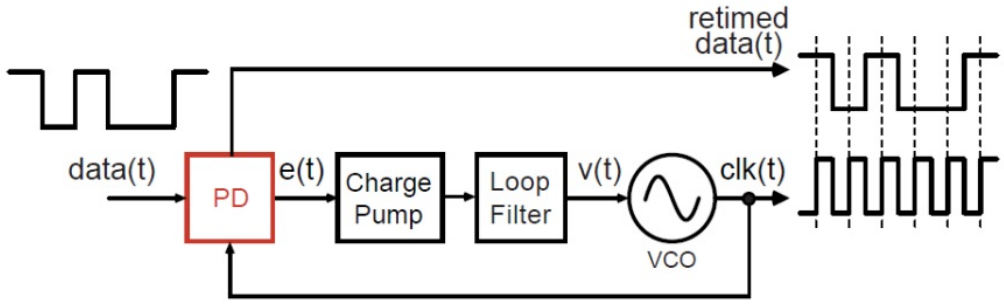

Figure 3.1: Block diagram of a typical CDR system

3.1.1 Phase detector

In a CDR, unlike many other feedback systems, the variable of interest changes dimension around the loop: It is converted from phase to voltage by the PD, from voltage to current by the CP, from current to voltage by the LF and from voltage to phase by the VCO. In the locked state, the phase relation between the incoming data and the generated clock is at a constant value, irrespective of the magnitude of the loop gain. It is also important that this constant value is such that we sample the data at the most optimal point, i.e., the center of the eye.

The phase detector, as the name implies, compares the phase of the incoming data and the generated clock and generates an output signal $e(t)$ that is directly proportional to the phase error ϕ_e. It serves as the error amplifier in the feedback loop, minimizing the phase error and driving the loop to the locked state. The loop is said to be locked when the phase error ϕ_e is a constant. The design philosophy of the CDR should be to minimize the phase error in the locked state, i.e., ideally we want the phase error to be zero. The locking behavior of the CDR can be explained as follows. The PD produces signal(s) whose DC value is directly proportional to the phase error ϕ_e. The LF filters out the high frequency components of the PD output, allowing the DC value to control the VCO frequency. Whenever the VCO frequency is equal to the frequency of the incoming data but the phase error ϕ_e has not established the required control voltage for the VCO, the loop will continue the transient, temporarily making the frequencies unequal again, and this process will continue until the phase error establishes the required control voltage for the VCO. In other words, both frequency acquisition and phase acquisition must be completed for the loop to lock [8].

It is important to note that while the above locking mechanism is straightforward, a simple PD cannot track step changes in the frequency of the input data stream. Thus, in variable data rate applications, a second CDR loop is used for coarse frequency acquisition, while the main CDR loop is used for fine frequency and phase acquisition [19]. In this thesis, a single loop CDR will only be considered since we are interested in constant data rate systems only.

Several different types of phase detectors are proposed across the literature. Generally speaking, these can be classified into two categories: linear PD and

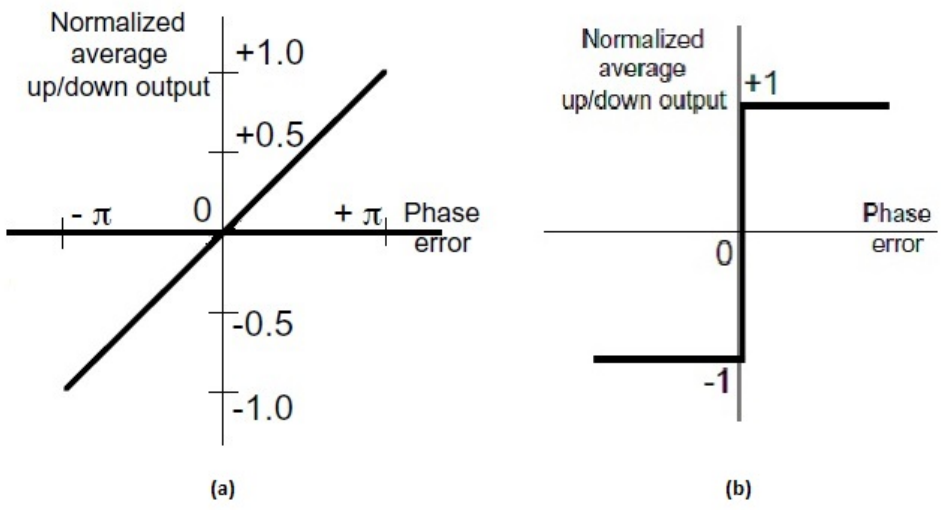

Figure 3.2: Phase transfer characteristics of (a) linear phase detector and (b) binary phase detector

binary PD (or non-linear). A linear PD produces an error signal whose DC magnitude is directly proportional to the phase error ϕ_e. On the other hand, a binary PD produces an error signal whose value depends only on the sign of the phase error ϕ_e. Figures 3.2(a) and 3.2(b) show the phase transfer characteristics of the linear PD and binary PD respectively. The linear PD has a linear phase transfer characteristic and the slope of the straight line depends on the data transition density. If we have a data transition density of 1, i.e., a transition every clock cycle, then the slope of the line is unity. On the other hand, the binary PD has only two output levels: if the phase error is positive, a high voltage is generated, if the phase error is negative, a low voltage level is generated. Each of the two configurations has its own merits and demerits. The biggest advantage of a linear PD is the low jitter generation since the average output only changes slightly when the phase error fluctuates around zero. However, it suffers from limited bandwidth and a static phase offset error due to mismatch in up and down paths. A binary PD typically has a higher PD gain and bandwidth, but is characterized by high output jitter generation. This is because the average output changes vastly when the phase error fluctuates around zero. Binary PDs work best when employed with all-digital CDRs. In this thesis, an analog CDR is considered and hence a linear PD will be used.

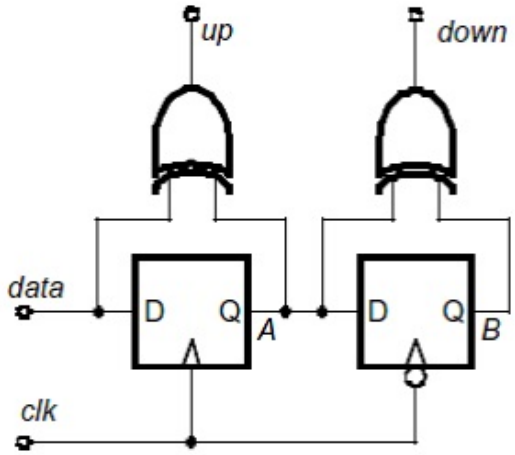

Figure 3.3: Circuit diagram of a Hogge phase detector

Hogge phase detector

The Hogge phase detector [20] shown in Figure 3.3 is a linear phase detector and will be the focus of this thesis. It consists of a positive edge-triggered flipflop, a negative edge-triggered flipflop and two XOR gates. The data input is served by the incoming serial data stream and the clock input is served by the generated VCO clock. The PD indicates the phase error using two output signals UP and DOWN. The DOWN signal is a reference signal that produces a periodic pulse of a fixed width, i.e., one half of the current clock period. The UP signal depends on the phase difference between the generated clock and the incoming data. If the positive-edge occurs at the center of the eye, the UP pulse will have the same width as the DOWN pulse, i.e., one half of the current clock period. Depending on whether we sample at the left or right of the eye, the UP pulse will have a lesser or greater width. Thus, the PD produces pulse width modulated signals that indicate the phase of the generated clock relative to the incoming data. In the locked state, the UP and DOWN pulses should have equal width and hence the average PD output is zero. The above described operation is shown graphically in Figures 3.4(a) and 3.4(b) for the cases where the clock rising edge is at the data center and the clock rising edge is to the right of the data center, respectively. It is important to note that if we have consecutive 0's or 1's, both the UP and DOWN pulses are not generated. While this may seem enough to maintain the charge of the LF capacitor at a constant value, leakage current almost

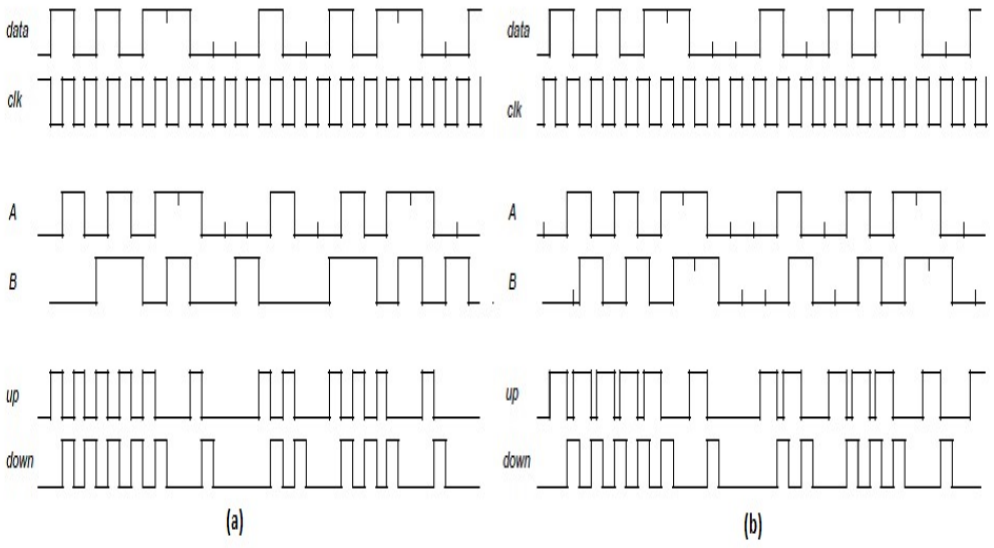

Figure 3.4: Waveform showing the operation of the Hogge PD when the clock samples (a) at the center of the eye and (b) to the right of the center

always causes the charge to drain. Thus, a CDR has a limit on the maximum number of 0's or 1's that are allowed in the incoming data and this is taken into account at the transmit encoder.

The input-output relationship can be expressed as:

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

where K_{PD} is the phase detector gain and is given by:

$$K_{PD} = \frac{TD}{\pi} \tag{3.2}$$

where TD is the transition density and can be assumed as 0.5 for random data.

3.1.2 Charge pump

A charge pump is a circuit that pumps or drains charge depending on the value of the input signals. As discussed in the previous section, the PD generates two pulse width modulated signals UP and DOWN. It is essential to convert these signals into a voltage since the only way to control a VCO is to adjust its input voltage. A charge pump serves this purpose. A conceptual

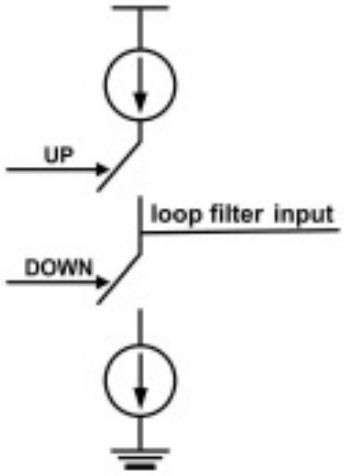

Figure 3.5: Conceptual circuit diagram of a charge pump

diagram of the charge pump is shown in Figure 3.5. It consists of two current sources, one to pump charge to the loop filter capacitor and the other to drain charge from it. The pump/drain action is initiated by opening or closing an electronically controlled switch using the UP and DOWN pulses. Thus, when the UP pulse is high, the upper switch is ON and charge is pumped into the capacitor. When the DOWN pulse is high, the lower switch is ON and charge is drained from the capacitor. It is worth mentioning that when UP and DOWN are both high or both low, no net charge should be pumped or drained from the capacitor.

The electronically controlled switches are realized using transistors. It is important to ensure that the two current sources are exactly equal in magnitude to ensure that there is no mismatch which will lead to a phase offset. While this may sound trivial, ensuring this requirement at the circuit level is a herculean task.

Together with the phase detector, the S-domain transfer function of the charge pump becomes the following for the case of a random data input:

$$H(s) = K_{PD} = \frac{i_{cp}}{2\pi} \tag{3.3}$$

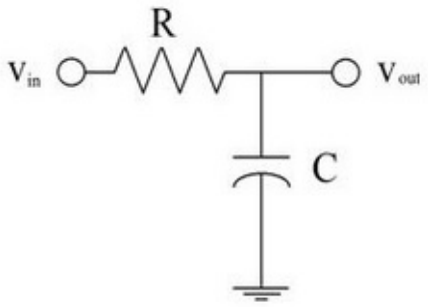

Figure 3.6: A simple passive first order RC filter

3.1.3 Loop filter

Loop filters are low pass filters which are used to filter out the high frequency components in the PD output. Typically, loop filters are realized using passive *RC* networks. The capacitor in the loop filter also serves as the reservoir into which the charge pump pumps or drains charge. Figure 3.6 shows a simple first-order low pass filter. Typical CDR implementations use higher order filters to track both frequency and phase accurately.

3.1.4 Voltage Controlled Oscillator (VCO)

VCOs are the most important and complex component of the overall CDR design. The essential idea behind a VCO design is to generate a clock signal based on the Barkhausen criteria for oscillation, which state that the magnitude of the VCO transfer function at the oscillation frequency is 1, while the phase is -180 degrees. Two of the most popular VCO topologies are ring-oscillator based and LC-tank based. The ring-oscillator is a digital circuit which consists of a cascade of odd number of inverters, arranged in a feedback path. By utilizing the fact that the delay of each inverter depends upon the amount of current it can sink in, which in turn can be made to depend on the control voltage, the frequency of oscillation can be controlled. Figure 3.7 shows a simple ring oscillator with 3 stages.

The LC-tank based VCOs utilize the resonant frequency of a series or parallel resonant circuit to produce oscillations at that frequency. At the resonant frequency, the energy lost from the capacitor is completely transferred to the inductor and vice-versa, and this leads to sustained oscillations.

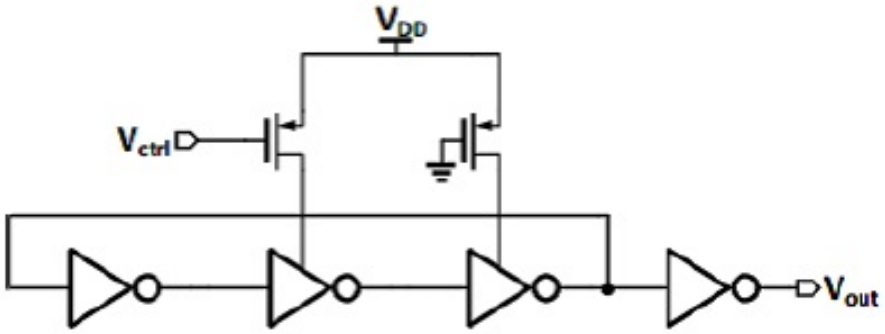

Figure 3.7: A typical ring oscillator based VCO

The resonant frequency is given by:

$$\omega_r = \frac{1}{\sqrt{LC}} \tag{3.4}$$

By utilizing a varactor diode capacitor, the capacitance can be made to be dependent on the control voltage and the resonant frequency of the circuit can be controlled.

VCO is the device that generates the target clock. Ideally, its output frequency should be linearly related to the input control voltage. The Laplace transform function of the VCO is derived as follows:

$$\omega_{out}(t) = K_{VCO} v_{ctrl}(t) \tag{3.5}$$

$$\mathcal{L}[\omega_{out}(t)] = \omega_{out}(s) = K_{VCO} v_{ctrl}(s) \tag{3.6}$$

$$\phi_{out}(t) = \int_0^t \omega_{out}(\tau) d\tau = \int_0^t K_{VCO} v_{ctrl}(\tau) d\tau \tag{3.7}$$

$$\mathcal{L}[\phi_{out}(t)] = \phi_{out}(s) = \frac{\omega_{out}(s)}{s} = \frac{K_{VCO} v_{ctrl}(s)}{s} \tag{3.8}$$

Thus, the Laplace transfer function of the VCO is:

$$H_{VCO}(s) = \frac{\phi_{out}(s)}{v_{ctrl}(s)} = \frac{K_{VCO}}{s} \tag{3.9}$$

where K_{VCO} is the VCO gain.

3.2 Analysis of a simple CDR in locked state

In this section, we obtain the transfer function of a simple CDR with a first order loop filter. The open loop transfer function of the CDR is equal to $H_O = K_{PD}G_{LPF}(s)\frac{K_{VCO}}{s}$, yielding a closed-loop transfer function of $H(s) = \frac{\phi_{out}(s)}{\phi_{in}(s)} = \frac{K_{PD}K_{VCO}G_{LPF}(s)}{s+K_{PD}K_{VCO}G_{LPF}(s)}$. In its simplest form, the loop filter is a first order filter which has a transfer function of the form $G_{LPF}(s) = \frac{1}{1+\frac{s}{\omega_{LPF}}}$, where $\omega_{LPF} = \frac{1}{RC}$. Thus, the closed loop response of the CDR is obtained as $H(s) = \frac{K_{PD}K_{VCO}}{\frac{s^2}{\omega_{LPF}}+s+K_{PD}K_{VCO}}$, indicating that the system is of second-order, where one pole is contributed by the VCO and the other by the LPF. Here, $K = K_{VCO}K_{PD}$ is termed the loop gain and is expressed in rad/s. In order to understand the dynamic behavior of the CDR, the denominator of the second-order closed-loop response is converted to a form commonly used in control theory: $s^2 + 2\zeta\omega_n s + \omega_n^2$, where ζ is the damping factor and ω_n is the natural frequency of the system. Therefore, the closed-loop response can now be expressed as $H(s) = \frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$, where $\omega_n = \sqrt{\omega_{LPF}K}$ and $\zeta = \frac{1}{2}\sqrt{\frac{\omega_{LPF}}{K}}$. Note that ω_n is the geometric mean of the -3dB bandwidth of the LPF and the loop gain. Typically, in a well designed second order system, ζ is usually greater than 0.5 and preferably equal to $\frac{1}{\sqrt{2}}$ so as to provide an optimally flat response. Thus K and ω_{LPF} cannot be chosen independently; for example if $\zeta = \frac{1}{\sqrt{2}}$, then $K = \frac{\omega_{LPF}}{2}$. If $s \to 0$, we note that $H(s) \to 1$; i.e. a static phase shift at the input is transferred to the output unchanged. We can examine the "phase error transfer function" defined as $H_e(s) = 1 - H(s) = \frac{\phi_e(s)}{\phi_{in}(s)} = \frac{s^2+2\zeta\omega_n s}{s^2+2\zeta\omega_n s+\omega_n^2}$, which drops to 0 as $s \to 0$, thereby achieving phase and frequency lock.

3.3 Analysis of a CDR with a second order loop filter

Now we derive the transfer function of a CDR with a second order loop filter such as the one shown in Figure 3.8 . This will be the type of CDR designed in this thesis.

The open loop transfer function of the CDR is given by:

$$H_O(s) = K_{PD}G_{LPF}(s)\frac{K_{VCO}}{s} \tag{3.10}$$

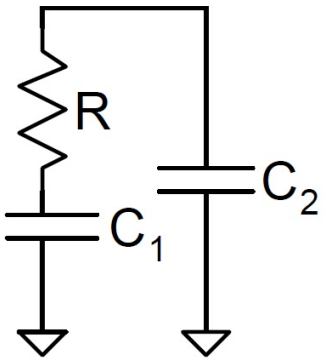

Figure 3.8: A second order low pass filter

This gives a closed loop transfer function:

$$H(s) = \frac{\phi_{out}(s)}{\phi_{in}(s)} = \frac{H_O(s)}{1 + H_O(s)} = \frac{K_{PD}K_{VCO}G_{LPF}(s)}{s + K_{PD}K_{VCO}G_{LPF}(s)} \tag{3.11}$$

The transfer function of the second order LPF is given by:

$$G_{LPF} = \frac{s + \frac{1}{RC_1}}{C_2 s(s + \omega_{LPF})} \tag{3.12}$$

where $\omega_{LPF} = \frac{1}{RC_{eq}}$ is the -3dB bandwidth of the LPF and $C_{eq} = \frac{C_1C_2}{C_1+C_2}$.

Thus, the closed loop transfer function of the CDR is given by:

$$H(s) = \frac{K(s + \frac{1}{RC_1})}{C_2 s^3 + \omega_{LPF}C_2 s^2 + Ks + \frac{K}{RC_1}} \tag{3.13}$$

where $K = K_{VCO}K_{PD}$ is the loop gain.

Since the coefficient of the s^3 term is C_2, which is of the order of $10^{-12}F$ typically, the s^3 term can be neglected without any significant loss in accuracy.

Thus, the closed loop transfer function of the CDR can be approximated in the standard form for a second order system with one zero:

$$H(s) = \frac{\frac{\omega_n^2}{\alpha}(s + \alpha)}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{3.14}$$

where $\omega_n = \sqrt{\frac{K}{C_1+C_2}}$, $\alpha = \frac{1}{RC_1}$ and $\zeta = \frac{1}{2}\frac{\omega_n}{\alpha}$.

We now show that the CDR can track step changes in the input frequency. Suppose a frequency step is applied at the input, i.e., $\omega_{in}(s) = \frac{\Delta\omega}{s}$. Then $\phi_{in}(s) = \frac{\Delta\omega}{s^2}$. The phase error transfer function defned as $H_e(s) = 1 - H(s) = \frac{\phi_e(s)}{\phi_{in}(s)} = \frac{1}{1+H_O(s)}$.

Applying the final value theorem, we find the steady-state error to be:

$$\begin{aligned}
\phi_{ss}^{F_{step}} &= \lim_{s\to 0} s.H_e(s).\phi_{in}(s) && (3.15a)\\
&= \lim_{s\to 0} s.\frac{1}{1+H_O(s)}.\frac{\Delta\omega}{s^2} && (3.15b)\\
&= \lim_{s\to 0} \frac{[RC_1C_2s^2 + (C_1+C_2)s]\Delta\omega}{RC_1C_2S^3 + (C_1+C_2)s^2 + Ks + 1} && (3.15c)\\
&= \frac{0}{1} && (3.15d)\\
&= 0 && (3.15e)
\end{aligned}$$

The above equations prove that the CDR with a second order LPF can track step changes in the input frequency and establish a relock with zero steady-state phase error. This would not have been possible with a first order LPF which can only track step changes in phase.

3.4 Loop design procedure

Now that we have studied a mathematical treatment of the CDR blocks, we should be able to suitably select the loop parameters such that we obtain a stable system that rapidly locks on to the incoming data stream. In the previous section, we derived the closed loop transfer function of the CDR in the form of a standard second order system with one zero given by equation 3.14. The phase margin of such a system is given by:

$$\phi_M = \arctan\left(\frac{\omega_{ugb}}{\alpha}\right) - \arctan\left(\frac{\omega_{ugb}}{\omega_{LPF}}\right) \tag{3.16}$$

where ω_{ugb} is the unity gain bandwidth.

One way to maximize the phase margin is to place the zero (α) below the unity gain bandwidth and place the pole(ω_{LPF}) above the unity gain bandwidth by the same factor. Let the scaling factor be β. Thus, $\alpha = \beta\omega_{ugb}$

and $\omega_{LPF} = \frac{\omega_{ugb}}{\beta}$. The phase margin can now be expressed as:

$$\phi_M(\beta) = \arctan(\beta) - \arctan(\frac{1}{\beta}) \tag{3.17}$$

The design procedure is outlined as follows:

1. Choose the desired unity gain bandwidth (ω_{ugb}) , charge pump current (i_{cp}), VCO gain (K_{VCO}) and phase margin (PM)
2. Using Eqn 3.17, obtain the value of β that gives the desired phase margin
3. Calculate the loop gain $K = K_{PD}K_{VCO}$, where $K_{PD} = \frac{i_{cp}}{2\pi}$
4. Calculate the value of C_2 given by $C_2 = \frac{K}{\beta\omega_{ugb}^2}$
5. Calculate the value of C_1 given by $C_1 = \frac{(\beta^2-1)K}{\beta\omega_{ugb}^2}$
6. Calculate the value of R given by $R = \frac{\beta}{\omega_{ugb}C_1}$

The values of K_{VCO} and i_{cp} are determined by the limitations of the circuit. For example, the value of the charge pump current is determined by the charge pump design. It is desirable to choose the maximum possible value of charge pump current that can be maintained for the entire range of operating voltages without suffering any mismatch between the up and down currents. Although this improves the phase detector gain, this will lead to increased power dissipation. Thus, it is up to the circuit designer to find the sweet spot that achieves the best tradeoff between power and performance. Similarly, K_{VCO} is determined by the VCO architecture. Typical values for F_{ugb} are between $\frac{F_{REF}}{20}$ and $\frac{F_{REF}}{15}$, where F_{REF} is the reference frequency, i.e., the frequency of the incoming data stream. The phase margin determines the settling time of the system. A positive value of phase margin is required for a stable system. Theoretically, a phase margin of about 55° will give the least settling time. However, several circuit non-idealities often cause an offset and the optimum value of phase margin is often quite different.

Thus, several iterations of the above-described design procedure might be required; therefore, the procedure is implemented as a MATLAB function to automate the process.

CHAPTER 4

BEHAVIORAL MODELING OF THE CDR

In the previous chapters, we have analyzed the behavior of the CDR from a mathematical as well as from a qualitative point of view. The mathematical analysis allows us to formulate a systematic design procedure for CDR loop design. Although the mathematical model is fairly accurate, the loop parameter design often requires several iterations before we can obtain realistic values. To assist with the iterative design, we seek the help of behavioral modeling.

4.1 Why behavioral modeling?

Traditionally, transistor level circuit simulations have been carried out with SPICE (Simulation Program with Integrated Circuit Emphasis). SPICE employs a form of nodal analysis, applying Kirchhoff's current law at the various nodes in the circuit. The equation is often expressed in the standard matrix form representation of a system of linear equations. Thus, the solution requires matrix inversion, which is a fairly complex operation with a computational complexity between $O(n^2)$ and $O(n^3)$. Consequently, SPICE simulations take rapidly increasing computation time as the number of nodes in the circuit increases. Therefore, it is desirable to have a faster method of simulation to perform back-of-the-envelope calculations for transistor-level circuit design. Behavioral modeling serves this purpose.

Behavioral modeling is not a completely new paradigm since behavioral modeling languages such as Verilog and VHDL have been in use for the past 25 years to model digital circuits. This has rapidly diminished the turnaround time for digital circuit designs and allowed increasing levels of design automation. In fact, a majority of digital design today is completely automated, with the user having to specify only the behavioral model of

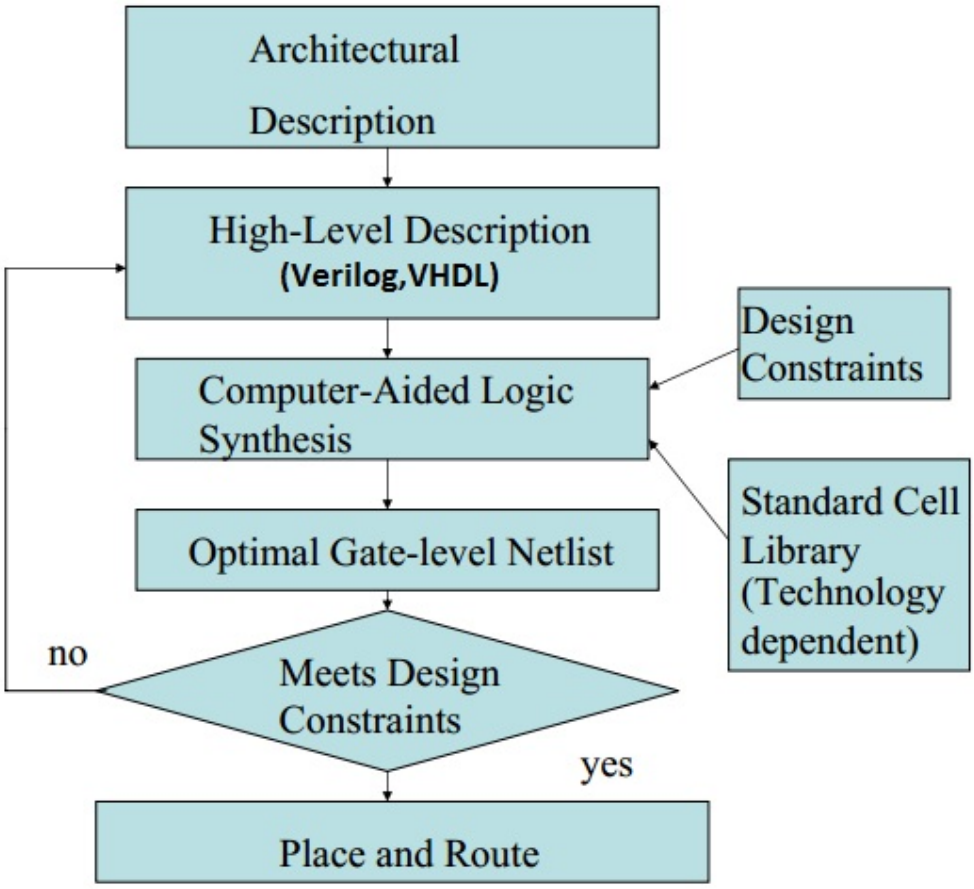

Figure 4.1: Typical digital circuit design flow

the circuit using a hardware description language (HDL). This model is often termed the register transfer level (RTL) model, since the behavior is described as the movement of data between registers, while being operated on by intermediate logic. This RTL model can be directly translated to transistor-level circuits by a process known as logic synthesis. This requires a library of transistor-level gates which can be used in the design and some design constraints such as timing and power constraints, in addition to the RTL model of the circuit. Figure 4.1 shows the digital circuit design flow using behavioral modeling and logic synthesis.

In recent years, there has been an industry-wide trend to extend the concept of behavioral modeling to the design of analog and mixed signal circuits, to harvest the same benefits obtained from the abstraction of digital circuit designs. An important point to note is that the behavioral models for analog and mixed signal circuits are not synthesizable by CAD (computer aided

design) tools. Instead, they are used for blackbox modeling of these circuits, allowing circuit designers to emulate the functionality of an analog block without actually getting into the nitty-gritties of the transistor-level design.

Such modeling techniques have several advantages:

- They are several times faster to simulate than traditional SPICE simulations since there are no numerically complex operations being performed.

- They offer powerful techniques for system level design. In the case of the CDR, they allow different loop configurations such as the charge pump current, VCO gain, etc., to be tested quickly and iteratively. This allows system designers to arrive at the optimum loop parameters efficiently.

- Behavioral modeling can be easily interfaced with traditional SPICE simulations, allowing unique design optimization techniques. For example, it is possible to test a new architecture for the phase detector by using a transistor level model for the phase detector along with behavioral models for the other blocks in the system. This technique also has immense value for debugging individual blocks.

- They offer a higher level of abstraction which leads to much faster turnaround time. For example, the charge pump current can be altered by changing the value of a variable in the design, whereas in traditional SPICE simulations, the entire circuit will have to be redesigned and resized to allow for a new current.

- The transistor-level models of a circuit are highly dependent on the technology used. Thus, as we move across technology nodes, a complete redesign of the circuit is required. This is not the case with behavioral models, which can be easily ported to different technologies.

In view of these advantages, we will first perform a behavioral modeling of the CDR system and obtain the various loop parameters. These will be used as starting points for the actual transistor-level design of the CDR.

4.2 Verilog-AMS

Verilog-AMS [21] is a high-level hardware description language (HDL) used to describe the structure and behavior of analog and mixed-signal systems. It is an extension to the IEEE 1364 Verilog HDL standard and is very powerful in providing fast prototyping capabilities for mixed-signal systems. The key advantage of circuit modeling using Verilog-AMS is that it provides a single language and simulator ecosystem that can be shared between analog, digital and system-level designers. Verilog-AMS leverages the superior speed and capacity offered by traditional Verilog and allows event-driven capabilities within analog model simulation, making it an attractive choice when simulating highly complex mixed-signal circuits such as PLLs, CDRs, ADCs, and DACs. The only pitfall of using Verilog-AMS is that it cannot replace traditional transistor-level SPICE simulation completely as it does not have synthesis capabilities like its digital counterpart Verilog. However, at the onset of the design phase, using Verilog-AMS for circuit modeling is very powerful for a mixed-signal circuit/system design engineer as it offers fast prototyping/verification for behavioral level simulation, thereby expediting the time-to-market for the system.

Verilog-AMS combines both Verilog-D and Verilog-A including a few additional mixed-signal constructs to provide a HDL language capable of performing truly mixed-signal simulation. Cadence has been the front-runner in promoting the language, making it an industry standard, and has led the majority of the advancement efforts ever since its release in 2003. The power of Verilog-AMS simulator in Cadence Virtuoso is that it can perform co-simulation among behavioral analog/digital blocks described by corresponding Verilog-A and Verilog-D models, respectively, as well as transistor-level circuit blocks by running the Spectre simulation. When a circuit consisting of transistor-level circuit elements, analog behavioral modules written in Verilog-A and digital behavioral modules written in Verilog-D is simulated, the AMS simulator in Cadence partitions the testbench into analog and digital components. The simulator then merges the analog simulation results from Spectre with the digital simulation results from NC-SIM and the resulting output is plotted just like that in the case of traditional Spectre simulation.

```
//Verilog-AMS HDL for "cdrbehav", "pd" "verilogams"

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

module pd (up,down,clk,D);
input clk,D;
output up,down;
reg q0,q1;
always @(posedge clk) begin
q0 <= D;
end
always @(negedge clk) begin
q1 <= q0;
end
assign up = D^q0;
assign down = q0^q1;
endmodule
```

Figure 4.2: Behavioral modeling of the Hogge phase detector

4.3 Behavioral modeling of the CDR blocks

We look at the behavioral modeling of the various CDR blocks and model them for 2 Gbps operation.

4.3.1 Hogge phase detector

The working of the Hogge phase detector was detailed in section 3.1.1. It is important to note that the Hogge phase detector is a completely digital circuit and is therefore modelled using traditional Verilog or Verilog-D. The two flipflops are instantiated using the "reg" specifier of the Verilog language and the XOR gates are represented through the ^ operator. The Verilog AMS model of the Hogge phase detector is shown in Figure 4.2 and the testbench is shown in Figure 4.3.

Figures 4.4 and 4.5 show the simulation waveforms for the case when the clock samples to the left and right of the center of the eye, respectively. In this testbench, the clock has completed frequency acquisition and only suffers from a phase misalignment. It can be clearly seen that the width of the UP

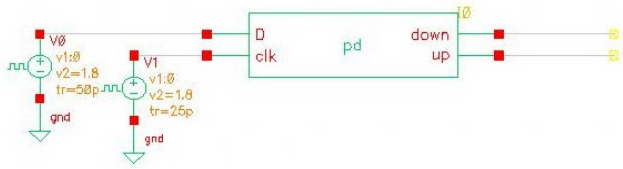

Figure 4.3: Testbench used for the phase detector

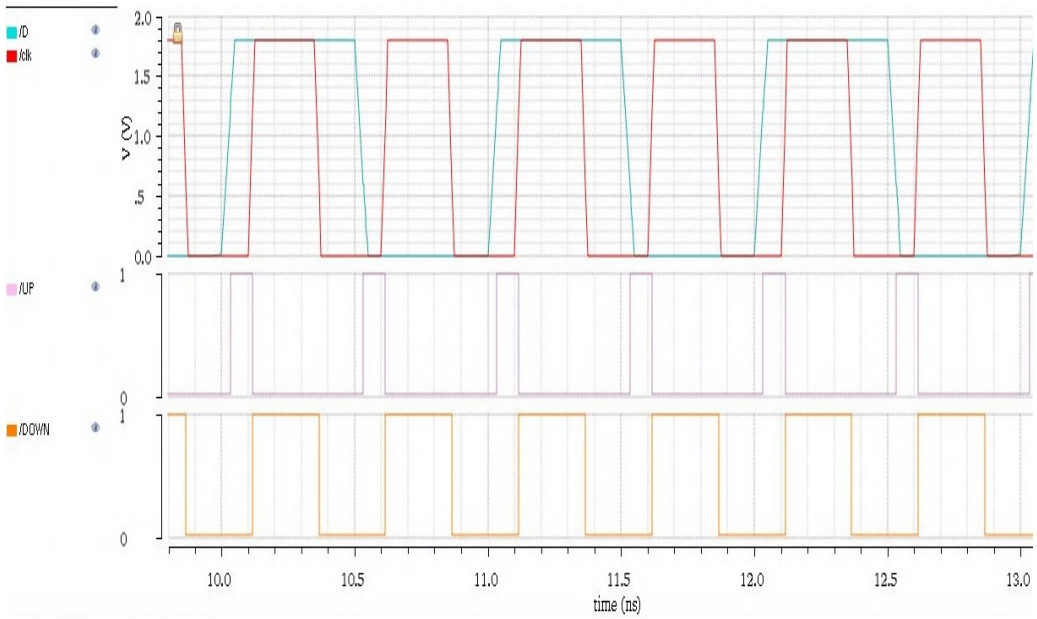

Figure 4.4: Waveforms showing the PD operation when the clock samples to the left of the center of the eye

pulse reflects the phase relation between the data and the clock, whereas the DOWN pulse has a constant width and is used as a reference.

4.3.2 Charge pump

The working of the charge pump was described in section 3.1.2. The charge pump is a mixed signal circuit. It accepts digital inputs in the form of the UP and DOWN pulses and provides an analog output, i.e., the current. Therefore, the charge pump must be modeled using Verilog-AMS constructs as shown in Figure 4.6. The first thing to note here is that the charge pump current is parameterized and is set by the real variable "curr". By changing the value of the variable, different charge pump currents can be obtained.

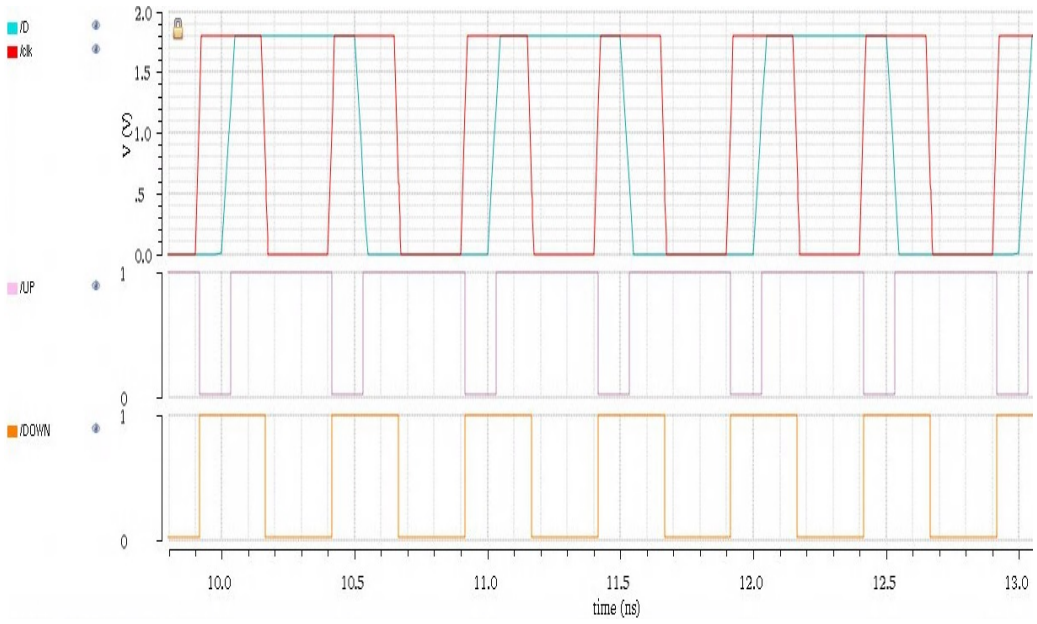

Figure 4.5: Waveforms showing the PD operation when the clock samples to the right of the center of the eye

The code itself is fairly similar to Verilog-D and is straightforward to understand. The key analog statement is the transition function, which is used to determine how the current will transition from one value to another. In this case, the rise time and fall time for the transition are specified as 10 ps. Figure 4.7 shows the testbench which includes the phase detector described previously.

Figures 4.8 and 4.9 show the simulation waveforms for the case when the clock samples to the left and to the right of the center of the eye, respectively. As seen previously, the phase detector modulates the width of the UP pulse based on the phase relation between the data and the clock. The charge pump pumps out current when UP is high and pulls in current when DOWN is high. When UP and DOWN are both high or both low, the net current coming out or going in to the charge pump is zero.

4.3.3 Filter

The filter is a completely analog circuit made of passive components, i.e., resistors and capacitors. Thus, we implement the filter with actual resistors and capacitors, rather than using any sort of behavioral model. This

```
//Verilog-AMS HDL for "cdrbehav", "cp" "verilogams"

`include "constants.vams"
`include "disciplines.vams"
`timescale 10ps / 1ps
module cp ( pout, nout, up, down);
parameter real curr = 500u;
input up,down;
output pout,nout;
electrical pout,nout;
real out;

analog begin
    @(initial_step) out=0.0;
     if(down && !up)
           out = -curr;
     else if(!down && up)
           out = curr;
     else out = 0;
           I(pout,nout) <+ -transition(out, 0.0, 10p, 10p);
     end
endmodule
```

Figure 4.6: Behavioral modeling of the charge pump

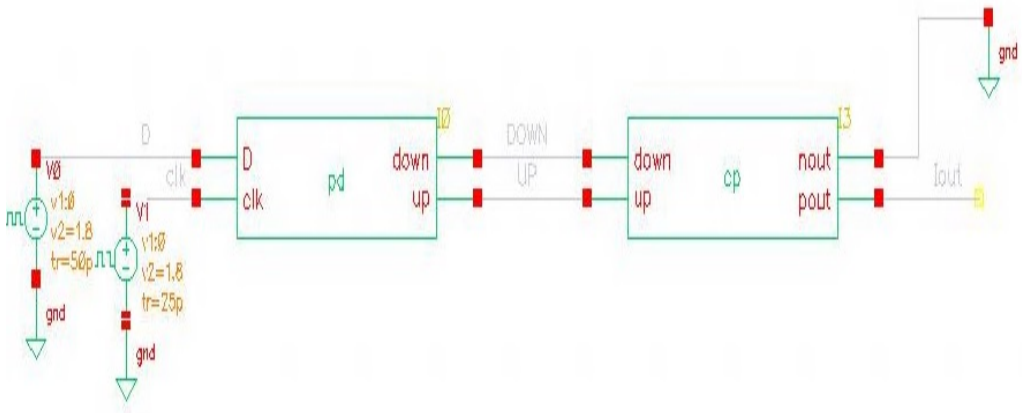

Figure 4.7: Testbench used for the charge pump

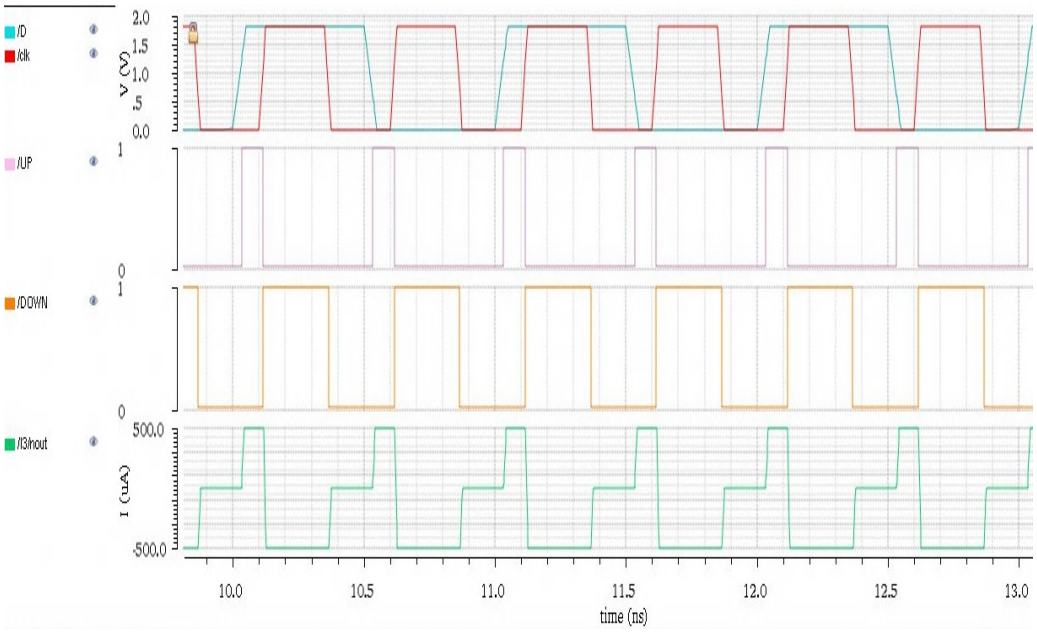

Figure 4.8: Waveforms showing the operation of the PD+CP when clock samples to the left of the center of the eye

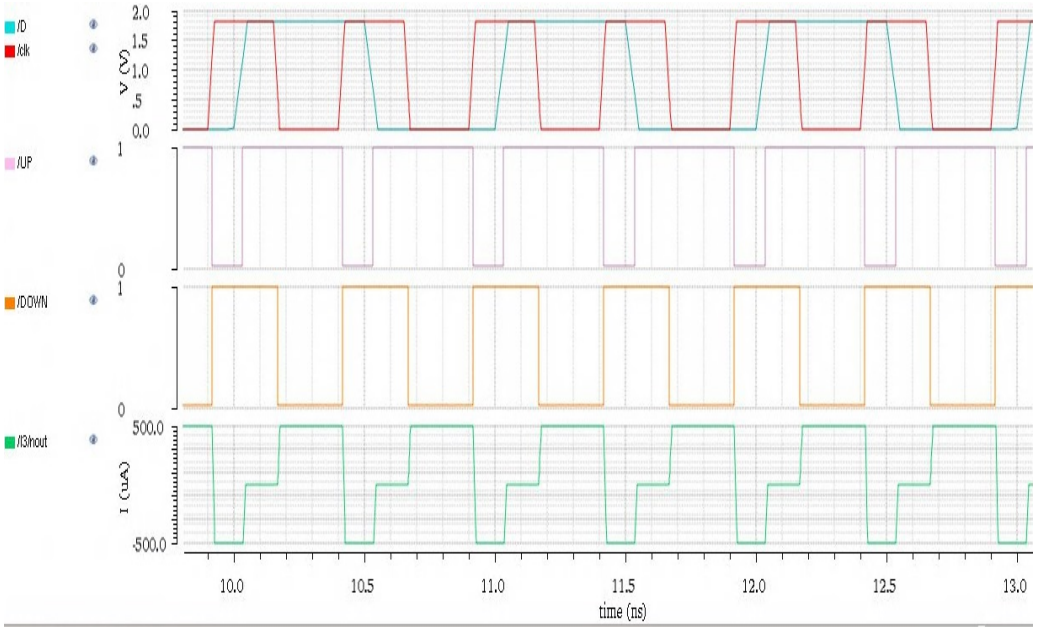

Figure 4.9: Waveforms showing the operation of the PD+CP when clock samples to the right of the center of the eye

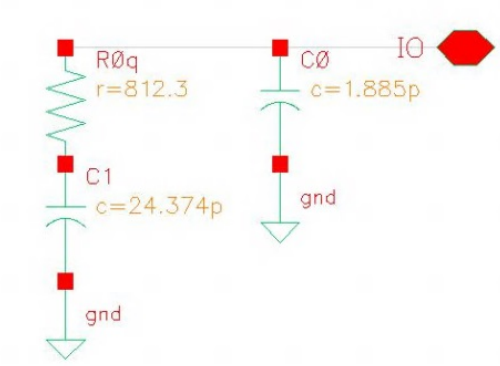

Figure 4.10: The low pass filter used in the design

is an important advantage of Virtuoso's AMS simulation engine which lets you mix and match physical and behavioral components. The values of the filter elements are chosen according to the design procedure outlined in the previous chapter. Figure 4.10 shows the filter that was used.

4.3.4 Voltage Controlled Oscillator (VCO)

The VCO is the most critical component of the CDR system and perhaps the most difficult to model. This can be attributed to the VCO being a completely analog circuit and the need to model jitter as accurately as possible, including random jitter such as white noise jitter. Thus, the VCO is modelled using Verilog-A constructs which are a subset of Verilog-AMS and is shown in Figure 4.11.

As in the case of the charge pump, the key variables such as frequency and voltage ranges are parameterized and hence can be easily changed. The analog portion of the code is used to model the behavior of the VCO, i.e., how the voltage controls the frequency, how much jitter it accumulates, when the zero crossings occur, etc.

Figure 4.12 shows the testbench for the VCO. Figures 4.13 and 4.14 show the waveforms for control voltages of 500mV and 950mV respectively. Clearly, the latter has a greater frequency and the VCO is functioning correctly.

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

module vco(vin, out);
/* I/O Declarations */
input vin ;
output out ;
electrical vin ;
electrical out ;
/* Parameter Declarations */
parameter real Vmin=0;          // Minimum input voltage
parameter real Vmax=1 from (Vmin : inf) ;       // Maximum input voltage
parameter real Fmin=    1.75e9 from (0 : inf) ;         // Minimum output frequency
parameter real Fmax=2.25e9 from (Fmin : inf) ; // Maximum output frequency
parameter real Vamp = 1.8 from [0 : inf) ; // Output sinusoid amplitude
parameter real ttol =1u/Fmax from (0 :1/Fmax) ; // Crossing time tolerance
parameter real vtol = 1e-9; // Voltage
// Minimum number p o i n t s p e r p e r i o d f o r u p d a t e
parameter integer min_pts_update =32 from [2 : inf);
// Transition time for square output
parameter real tran_time = 10e-12 from (0:0.3/Fmax);
// Std d e v i a t i o n o f p ha s e j i t t e r ( UI )
parameter real jitter_std_ui = 0 from [0:1) ;
/* Internal Variables */
real freq ;
real phase ;
integer n ;
integer seed ;
real jitter_rad  ;
real dPhase ;
real phase_ideal ;
analog
begin
        @(initial_step)
        begin
                seed = 671;
                n = 0;
                dPhase = 0 ;
                jitter_rad = jitter_std_ui*2*`M_PI ;
        end
        // compute the freq from the input voltage
        freq = ((V(vin) - Vmin )*(Fmax - Fmin)/(Vmax - Vmin )) + Fmin ;
        $bound_step(1/(min_pts_update*freq)) ;
        if (freq > Fmax)        freq = Fmax ;
        if (freq < Fmin)        freq = Fmin ;
        phase_ideal = 2*`M_PI*idtmod(freq,0.0,1.0,-0.5) ;
        phase = phase_ideal + dPhase ;
        @(cross(phase_ideal + `M_PI/2 , +1, ttol, vtol)
        or  cross(phase_ideal - `M_PI/2 , +1, ttol , vtol))
        begin
        dPhase = $rdist_normal(seed, 0, jitter_rad) ;
        end
        @(cross( phase + `M_PI/2 , +1, ttol , vtol)
        or cross( phase - `M_PI/2 , +1, ttol, vtol))
        begin
        n = (phase >= -`M_PI/2)&&(phase < `M_PI/2);
        end
        // generate the output
        V(out) <+ transition (n?Vamp:0, 0, tran_time);
end
endmodule
```

Figure 4.11: Behavioral modeling of the VCO

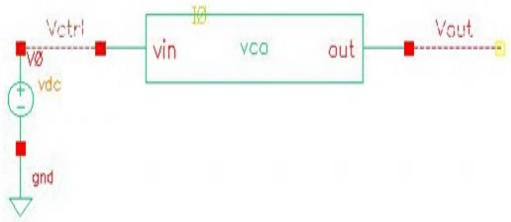

Figure 4.12: Testbench used for the VCO

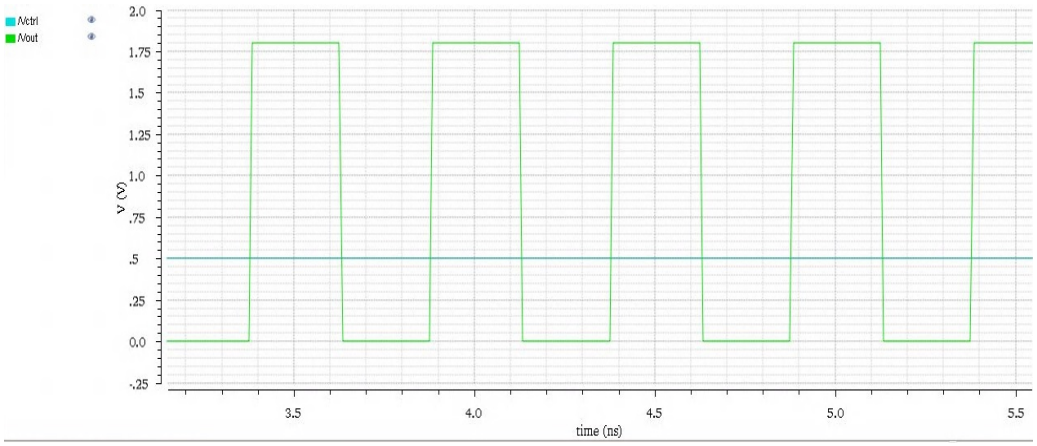

Figure 4.13: Waveforms showing the operation of the VCO for a control voltage of 500 mV

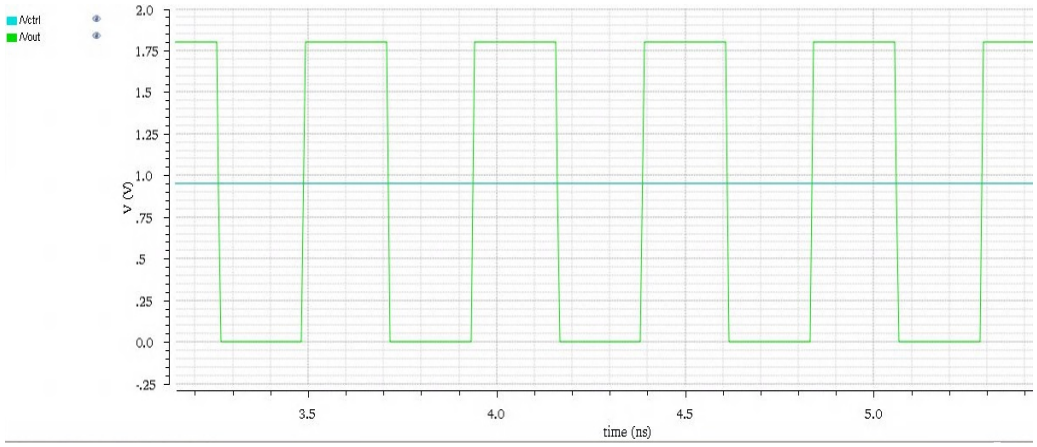

Figure 4.14: Waveforms showing the operation of the VCO for a control voltage of 950 mV

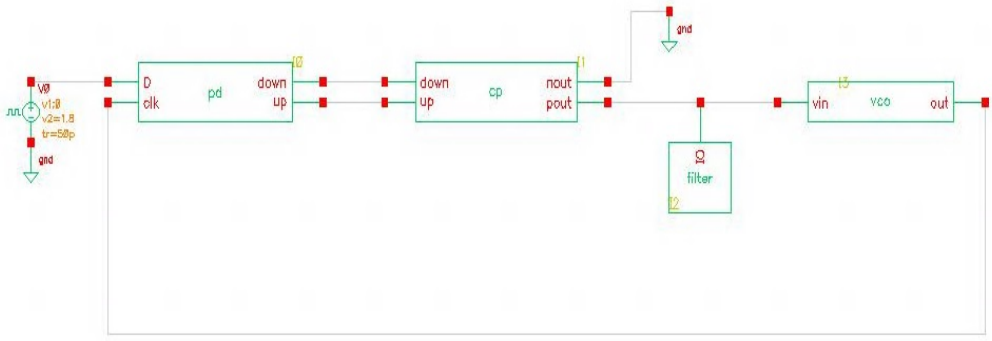

Figure 4.15: Testbench for the complete CDR simulation

4.4 Complete CDR simulation with jitter

Now that we have verified the functionality of the various blocks involved in the CDR, it is possible to simulate the complete system, all using behavioral models. Such simulations are useful to get some early feedback and allow rapid prototyping of the design. It also helps to compare different architectures, loop parameter configurations, etc. Figure 4.15 shows the test bench used to simulate the CDR system.

Figures 4.16 shows the simulation waveform of the control voltage settling to the correct value to generate a clock at 2 GHz to sample the 2 Gbps data stream, which is 500 mV in this case. It is important to note that the control voltage does not actually settle at 500 mV but has a 50 mV ripple with a DC value of 500 mV. The ripple can be attributed to the charging and discharging of the capacitor in the loop filter which causes small changes in the control voltage. This ripple effect is not desirable and will affect the VCO by causing jitter at the output.

Figure 4.17 shows the waveforms after lock has established. Clearly, the rising edge of the clock samples the data at the center of the eye, which is what we desired. The bottom trace shows the charge pump current waveform. It is obvious that an equal amount of current is being pushed and drained by the charge pump, therefore keeping the control voltage almost constant, except for a small 50 mV ripple.

Figure 4.18 shows the eye diagram of the generated clock, measured after lock acquisition. The width of the transition region gives the peak jitter of

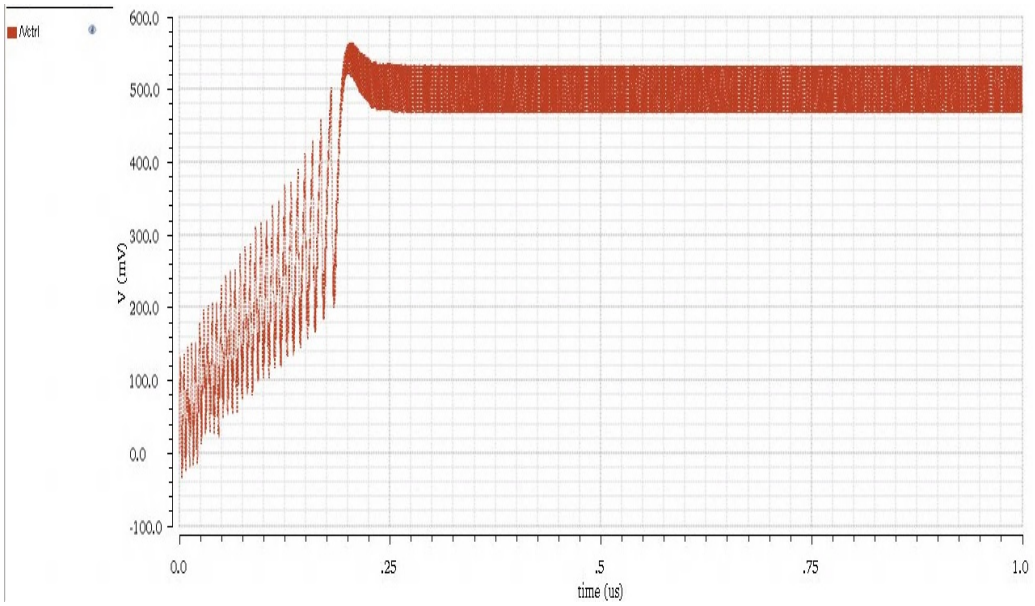

Figure 4.16: Waveform showing the locking behavior of the control voltage

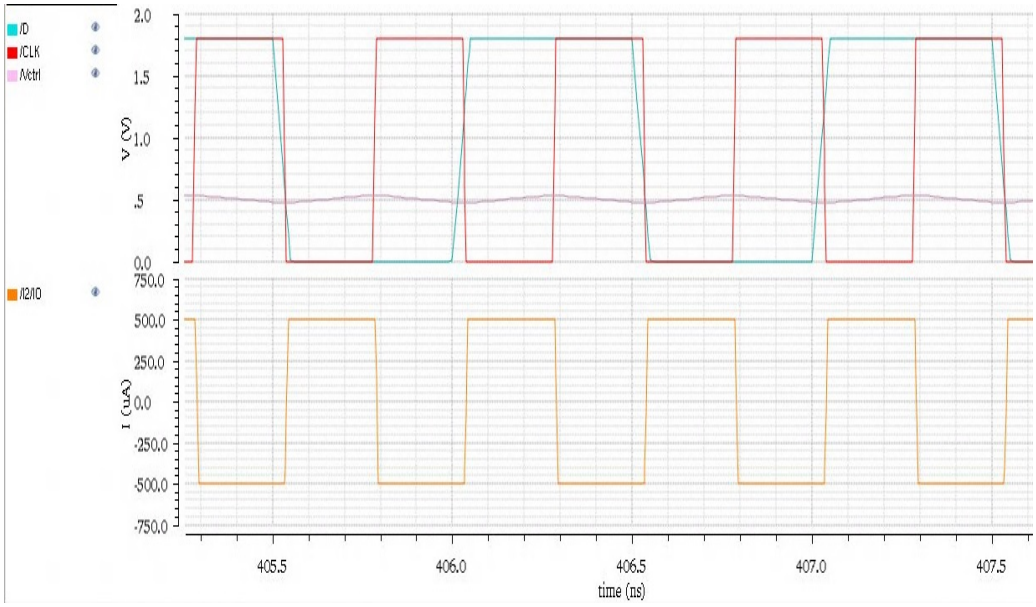

Figure 4.17: CDR Waveforms after frequency and phase lock has been established

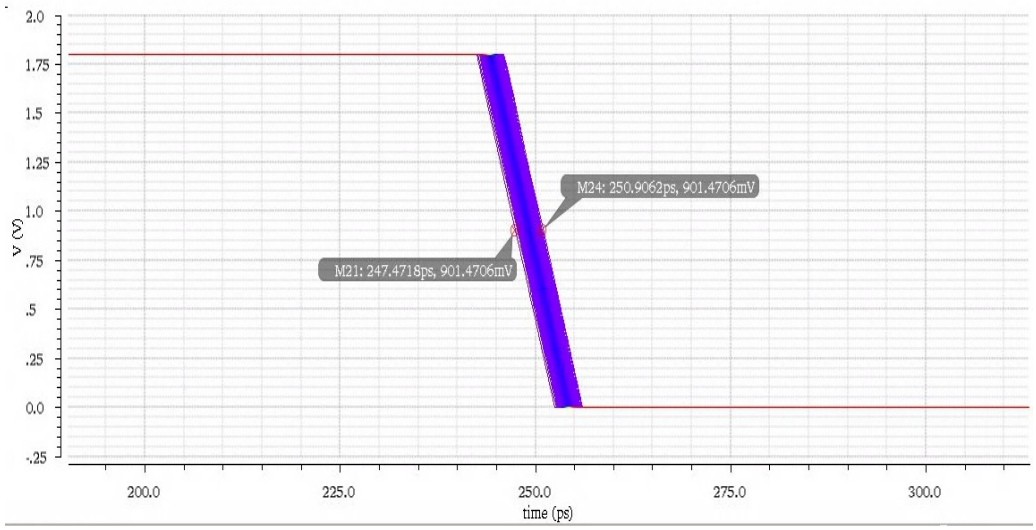

Figure 4.18: Eye diagram showing jitter in the generated clock after lock acquisition

the circuit and in this case it is 3.434 ps. Note that this value is obtained from random jitter due to white noise alone without considering other sources of jitter. Once we include other effects such as transistor-level jitter, source-side jitter, etc., the value is expected to be much higher.

Thus, the behavioral model has verified the functionality of our CDR and given us starting values for the various loop parameters which will be used to design the actual transistor level circuits. However, this model does not account for any non-idealities that are present in an actual circuit, which are bound to affect the operation of the CDR. Thus, manual tuning is inevitable in the design of analog mixed signal circuits.

CHAPTER 5

SINGLE-ENDED CDR DESIGN

In this chapter, we discuss the transistor level design of the single-ended SERDES system depicted in Figure 5.1. In addition to the CDR, several other blocks such as the transmit driver, channel, receiver amplifier, etc., are also designed in order to form a complete system whose performance can be fairly evaluated. The serializer and deserializer blocks are not discussed and it is assumed that a serialized bit stream is available at the transmitter side. The data recovered by the CDR can be deserialized using the generated clock.

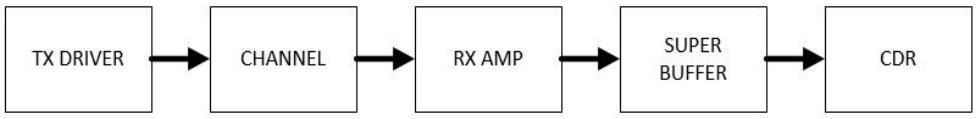

Figure 5.1: The single-ended SERDES system

5.1 Transmit driver

As discussed previously, the driver amplifier is used to amplify the serial bit stream to make it suitable for transmission over the channel. In addition, it also provides 50 Ω terminations for the channel to eliminate reflections. The input to the driver comes from the serializer whose last stage is typically a current mode logic (CML) circuit which has differential outputs. In this case, the input signal to the driver is assumed to have a voltage swing of 500 mV, with logic high being represented by 1.8 V and logic low being represented by 1.3 V respectively.

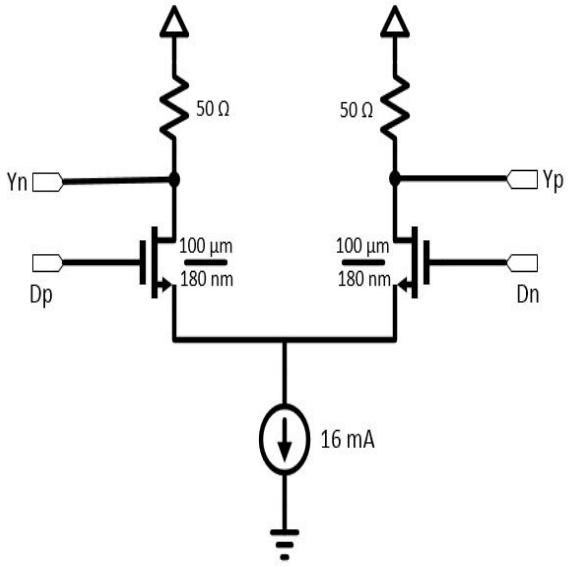

Figure 5.2: Transistor level schematic of the transmit driver

The transistor level schematic of the driver amplifier is shown in Figure 5.2. It is essentially a differential amplifier, with the pull up resistors being 50 Ω in order to provide impedance matching with the channel. It is important to note that only the Yp output is used to drive the channel, since we are dealing with a single-ended channel. It is assumed that the signal transmitted through the channel will need a voltage swing of 400 mV, i.e., the difference between the high and low logic levels. This is often determined by the type of channel, the sensitivity of the receiver, etc., and therefore a reasonable value is chosen.

All current sources depicted in this thesis were designed using current mirror circuits to accurately model the effects of tail current variation due to channel length modulation etc. It is assumed that the logic high voltage will be the same as the supply voltage for the 180 nm technology, i.e., 1.8 V. Thus, the voltage level for logic low should be 1.4 V for a 400 mV swing. The value of the current source is chosen as follows:

By Ohm's law, $\Delta V = IR$, where R is effective resistance seen between VDD and the output node. In this case, this consists of a 50 Ω resistor at the transmitter side in parallel with a 50 Ω resistor in the receiver side termination as shown in Figure 5.3. Thus, R = 25 Ω and $\Delta V = 400mV$. Consequently, $I = 16mA$.

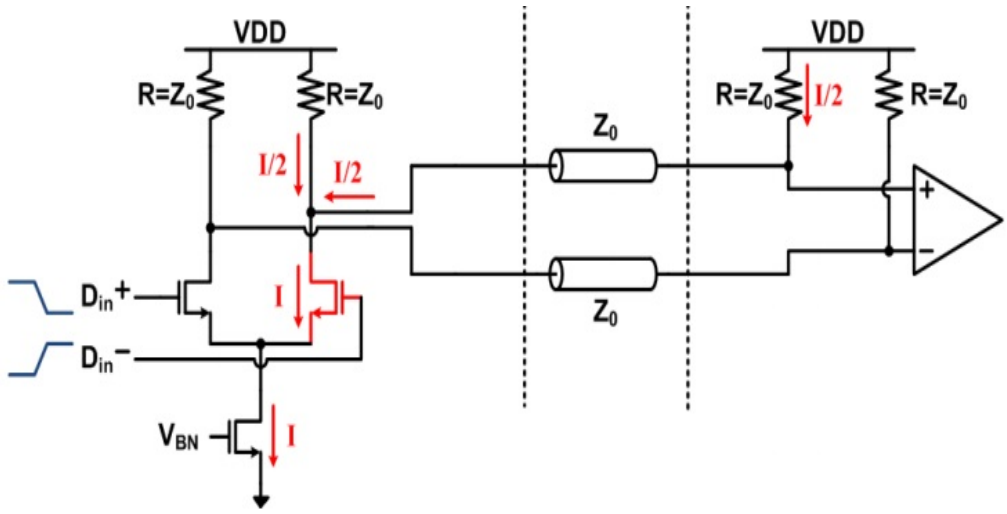

Figure 5.3: Operation of the current mode transmit driver

The transistor sizes are chosen in such a way that they can sink 16 mA of current, with the given input and output voltage specifications, while remaining in the saturation region. It is also important to consider the fact that, at any given point, only one of the transistors needs to be conducting, while the other must be cutoff to ensure that the logic levels do not deteriorate. Lastly, it is also necessary to consider the minimum voltage required by the current mirror to provide 16 mA of current and ensure that the voltage at the node does not fall below the minimum value.

5.2 Channel

The channel is the actual physical medium through which data is transferred from the transmitter to the receiver. In this work, the channel is a simple microstrip line on a PC motherboard. The channel is modelled using Q3D, a 2D electromagnetic field solver, which analyzes the geometry of the channel and solves Maxwell's equations numerically to obtain a circuit model of the channel in terms of RGLC elements as a function of frequency. The circuit model can be directly used in a circuit simulator such as Cadence Spectre or to generate the S parameters of the channel, which can also be used in Cadence Spectre. More complex channel geometries can be modelled using tools such as ANSYS HFSS which support full 3D modelling and directly

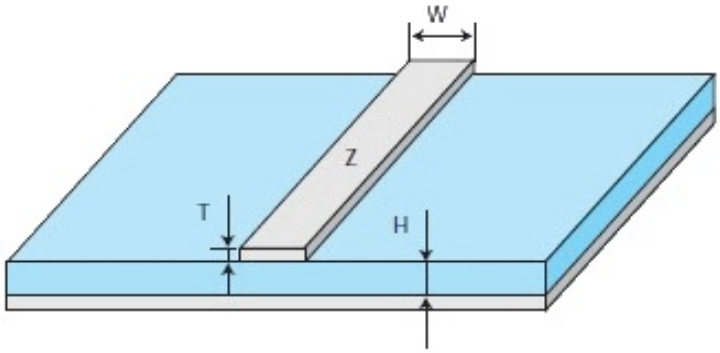

Figure 5.4: An illustrated 3-D view of the channel

Substrate	ε_r	Loss/inch per gigahertz	Z = 50 Ω, T = 35 µm			
			H = 1.6 mm		H = 0.8 mm	
			W	ε_{eff}	W	ε_{eff}
FR4	4.5 ± 10%	0.08 dB	2.95 mm	3.38	1.45 mm	3.38
RO4003	3.38 ± 1.5%	0.02 dB	3.30 mm	2.65	1.65 mm	2.65

Figure 5.5: Different PCB stack ups to obtain a 50 Ω microstrip line

provide the S parameters of the channel.

Figure 5.4 shows a view of the channel. For the single-ended case, a single microstrip line with a characteristic impedance of 50 Ω is used. The channel design is accomplished using the following equations [22]:

$$\mathbf{Z} = \frac{87}{\sqrt{\epsilon_r + 1.41}} . \ln[\frac{5.98H}{0.8W + T}] \tag{5.1}$$

$$\textbf{if } \mathbf{W}>\mathbf{T} : \epsilon_{eff} = \frac{\epsilon_r + 1}{2} + \frac{\epsilon_r - 1}{2} . (1 + 12(\frac{H}{W}))^{-0.5} \tag{5.2}$$

Figure 5.5 shows a variety of different PCB stack-ups that accomplish the required characteristic impedance [23]. In this work, an FR4 substrate with 0.8 mm thickness is used to design an 8 inch long channel.

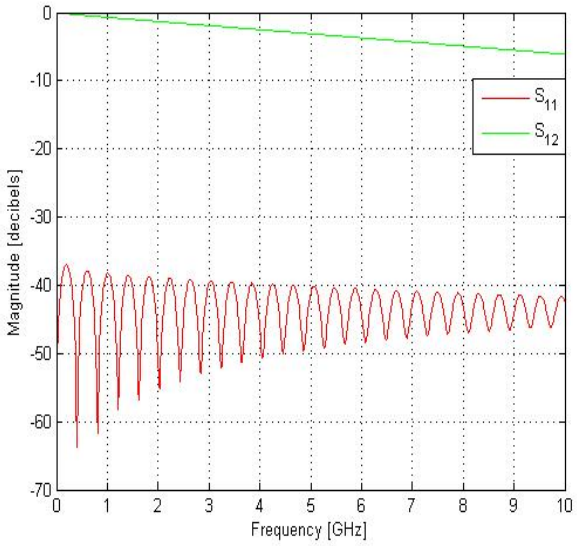

Figure 5.6: The frequency response of the designed channel

Figure 5.6 shows the channel response, i.e., the S parameter S_{12} which represents the loss or attenuation of the channel. Clearly, as the frequency increases, the attenuation gets larger and equalization techniques have to be used. Since we are dealing with a 2 Gbps system, we are concerned with the channel attenuation at 1 GHz, which is about 2 dB, and therefore equalization is not needed.

5.3 Receiver

The receiver interface circuit consists of an amplifier and a superbuffer.

5.3.1 Receiver amplifier

The receiver amplifier is used to amplify the swing of the incoming signal to make it suitable for the receiver circuitry such as the CDR. It is typically a high gain differential amplifier consisting of multiple stages. As shown in Figure 5.7, we have a 3-stage differential amplifier that is used to increase the swing from 400 mV to a full swing signal that can be processed by CMOS logic circuits. The first stage is used to convert the single ended signal to

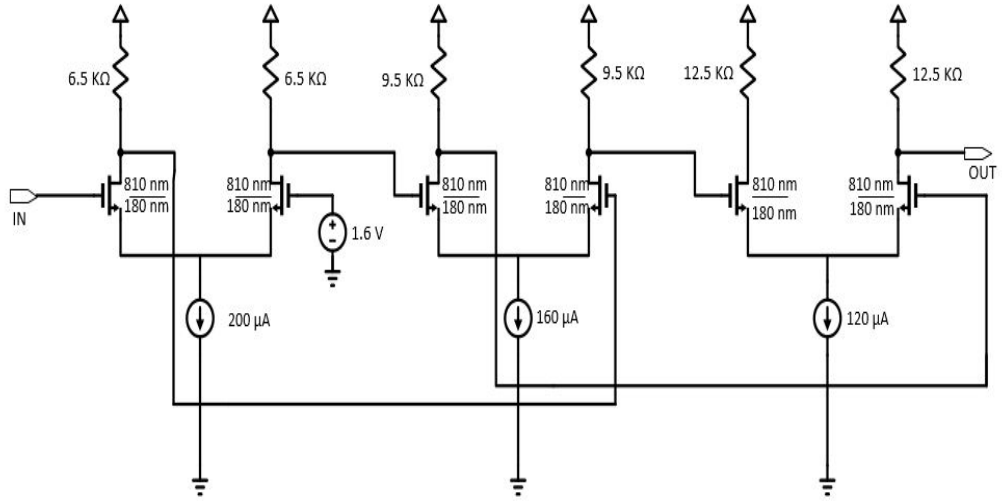

Figure 5.7: The 3-stage differential amplifier used in the receiver

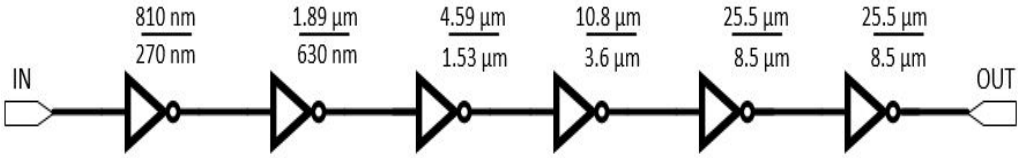

Figure 5.8: The super buffer used in the design

differential and to suppress the common mode from 1.6 V to about 1 V. The subsequent two stages are used to amplify the signal while ensuring that there is no significant rise time or fall time degradation. For further details on the design of differential amplifiers, the reader may refer to extensive work in the literature [24, 25].

5.3.2 Superbuffer

A superbuffer is a chain of cascaded inverters, which is often used to drive a large load with minimal delay and rise/fall time degradation. In this case, the CDR circuit offers a huge capacitive load that cannot be directly driven by the amplifier stage. Hence, we use a chain of inverters, with gradually increasing size, which can drive the load more efficiently as shown in Figure 5.8. The dimensions shown represent the width of the PMOS and NMOS

transistor for each inverter. All transistors use a minimum length of 180 nm. The inverters are sized to have equal delays for rising and falling transitions. For more details on the design on superbuffers, the reader may refer to [26].

Figure 5.9 shows the waveforms from the receiver circuitry. Clearly, the differential amplifiers amplify the input signal from the channel to CMOS logic levels through the 3 stages. The superbuffer is then used to drive the CDR, while also providing full voltage swing and sharpening the input pulses so that the CDR can function efficiently.

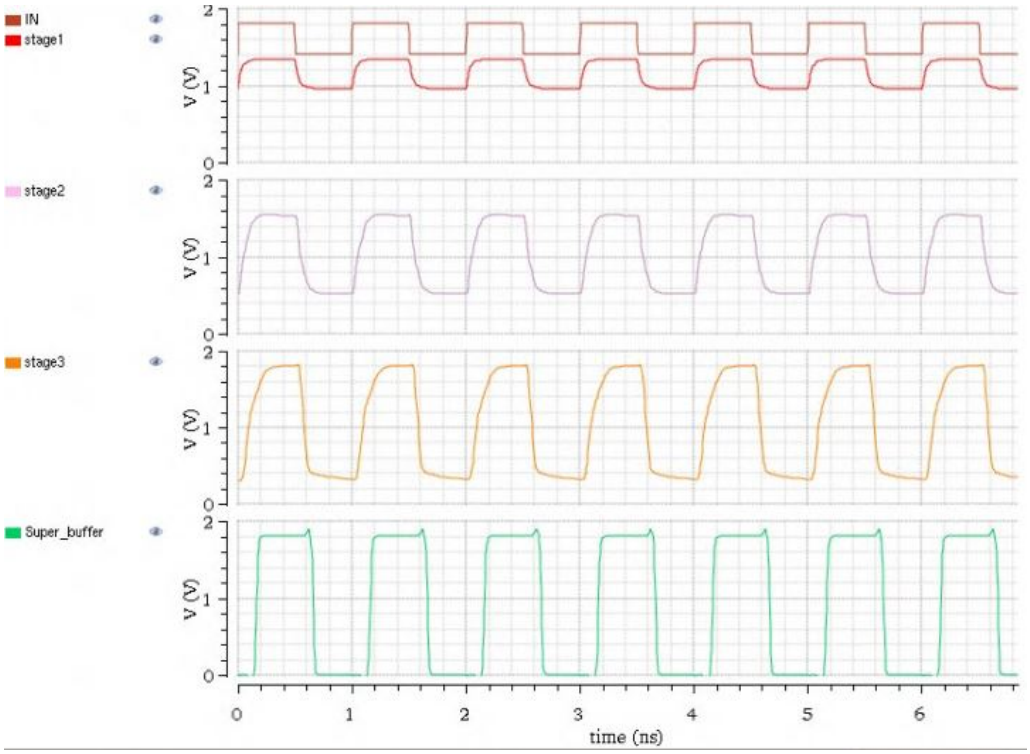

Figure 5.9: Waveforms showing the operation of the receiver amplifier

5.4 Clock and data recovery (CDR) circuit

Figure 5.10 shows the block diagram of the CDR circuit with the incoming data coming from the superbuffer. The design methodology used is the logical effort technique [27], where we attempt to distribute the electrical effort across the various blocks in order to reduce the delay and area of the circuit. A key detail is that we attempt to design the block in a bottom-up manner,

starting from the output. This way, we know the load that each block needs to drive and can therefore size the block accordingly. We now look at the design of each of these blocks in detail.

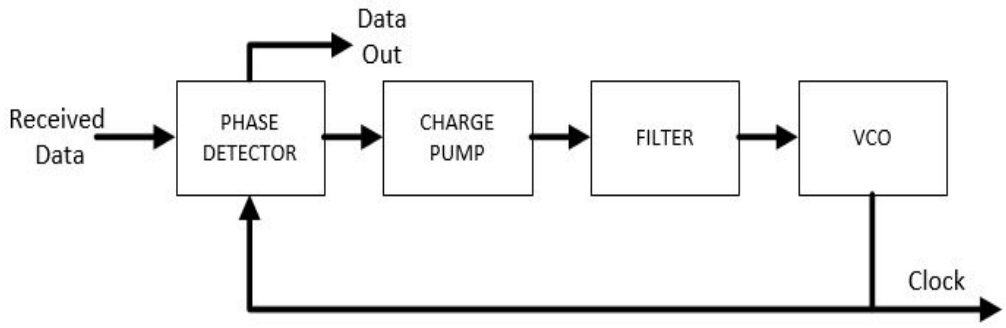

Figure 5.10: Block diagram of the CDR system

5.4.1 Filter

The values of the loop filter elements are chosen by using the design procedure outlined in section 3.4 as a guideline. Manual tuning of the parameters was done to account for the non-idealities in the circuit. Figure 5.11 shows the loop filter circuit.

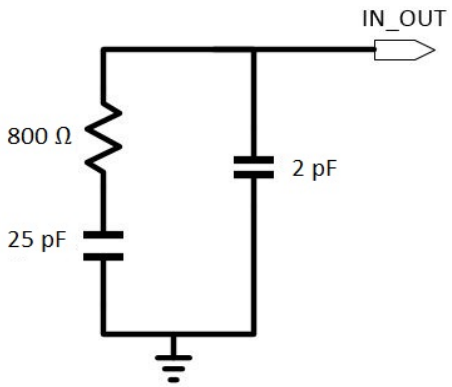

Figure 5.11: The loop filter used in the design

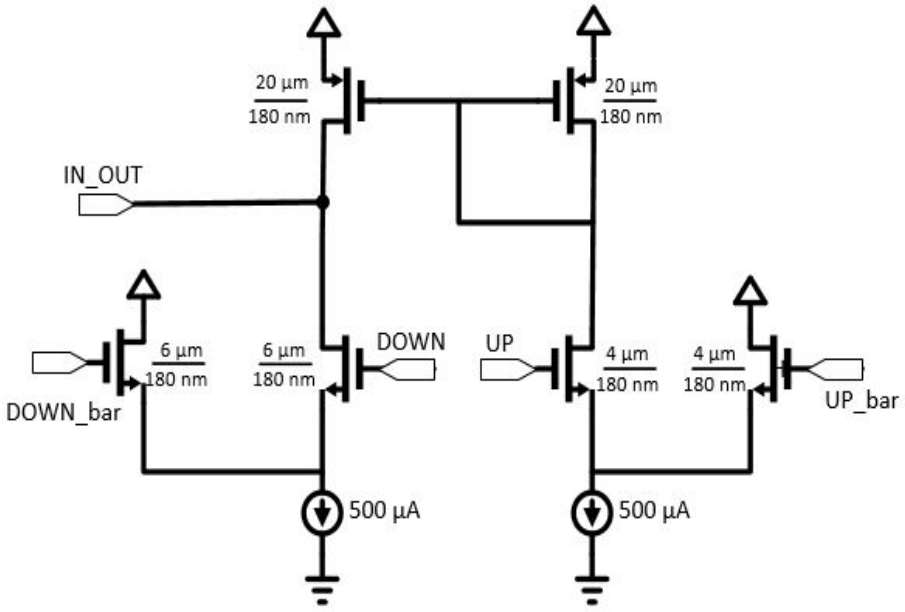

Figure 5.12: A current steering charge pump with NMOS switches only

5.4.2 Charge pump

The charge pump is used to pump/drain charge to/from the loop capacitor. The charge pump can be controlled electronically using the UP and DOWN signals which are generated by the phase detector. The charge pump was designed to have a pump/drain current of 500 μA. We make use of a differential current steering topology for the charge pump as shown in Figure 5.12.

One of the key requirements of the charge pump is to maintain symmetry between the push (UP) and pull (DOWN) currents across the entire range of operating voltages. The differential current steering topology is advantageous since we use NMOS transistors for both UP and DOWN currents, resulting in a good matching of the currents. However, other effects such as channel length modulation often cause a mismatch between the two currents resulting in a static phase error, i.e., the clock may not sample the data at the center of the eye. Thus, it is important to study the charge pump characteristics across a full range of operating voltages. Figure 5.13 shows a plot of UP and DOWN currents plotted against the voltage at the IN_OUT node.

We observe that the currents are perfectly matched only at about 906 mV. The UP current decreases with the voltage of the IN_OUT node and the

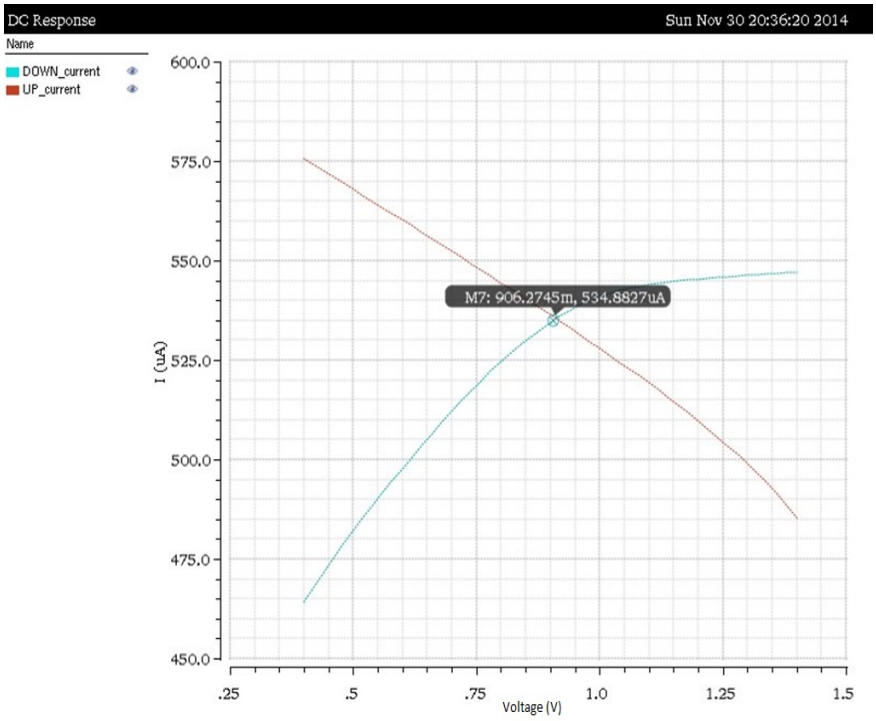

Figure 5.13: Variation of UP and DOWN currents as a function of the output node voltage

DOWN current increases due to channel length modulation. If we define a tolerance of 50 μA (10%), we see that the range of IN_OUT voltage must be restricted to between 600 mV and 1.2 V after lock-in, in order to avoid any significant static phase error.

The sizing of the transistors is done so as to maximize the range of voltages for which the charge pump can be used. A key concern is to provide sufficient voltage drop to the current mirror circuit in order to provide roughly 500 μA current.

5.4.3 Phase detector

As discussed previously, the phase detector is used to provide a measure of the phase relation between the generated clock and the incoming data. We implement a Hogge phase detector, which is a linear phase detector consisting of flipflops and XOR gates. Figure 5.14 shows the block diagram of the Hogge phase detector, which provides the phase information in the form of UP and

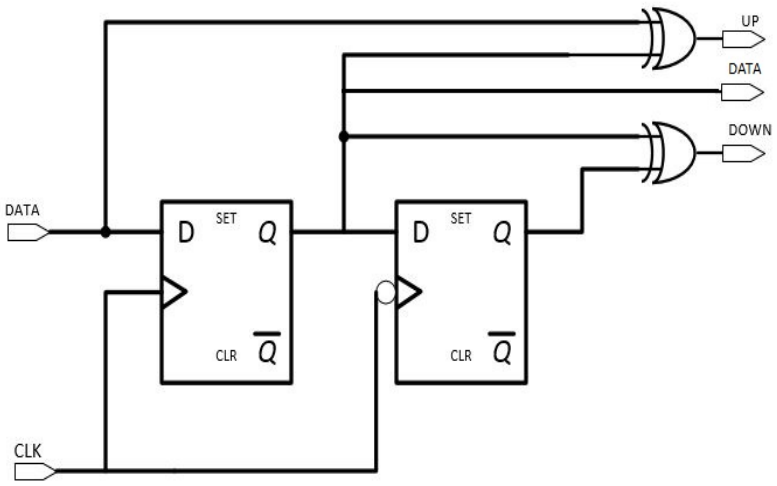

Figure 5.14: Block diagram of the phase detector

DOWN pulses which can be used to control the charge pump. We will now look into each block of the phase detector in detail.

Positive edge-triggered flipflop

The positive edge-triggered flipflop is realized using the True Single Phase Clocked (TSPC) register architecture. This has the advantage of requiring only a single phase of the clock as opposed to differential clock signals. In addition, it requires fewer transistors than other flipflop designs. Figure 5.15 shows the implementation. Since the positive edge-triggered flipflop has a bigger load, it is sized larger than the negative edge-triggered flipflop. A PMOS sizing factor of 2 is used in the design.

Negative edge-triggered flipflop

The TSPC architecture is again used to implement the negative edge-triggered flipflop, which is significantly smaller and is shown in Figure 5.16.

XOR gate

The XOR gate is realized using a transmission gate topology and is shown in Figure 5.17. Since the charge pump requires differential inputs, two copies

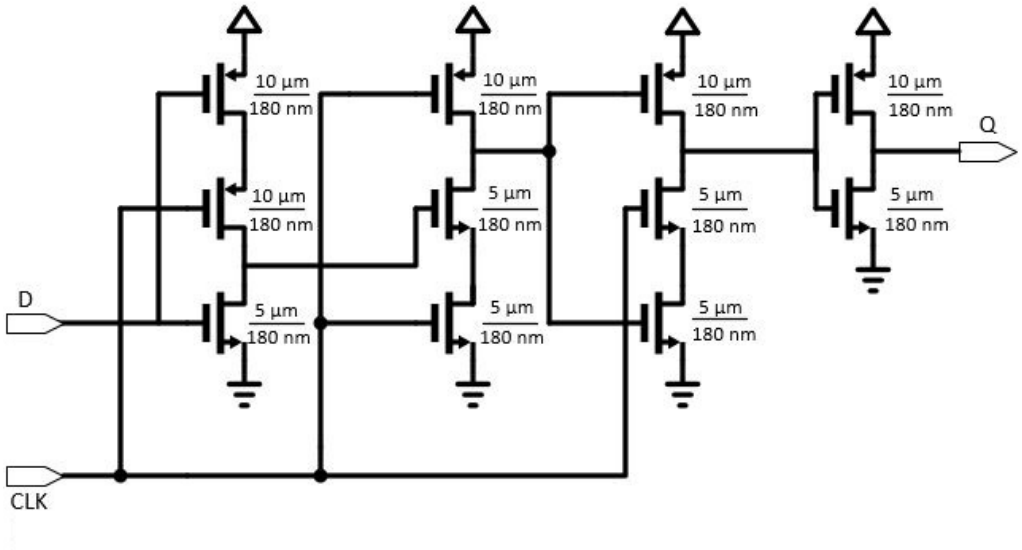

Figure 5.15: A TSPC positive edge-triggered flipflop

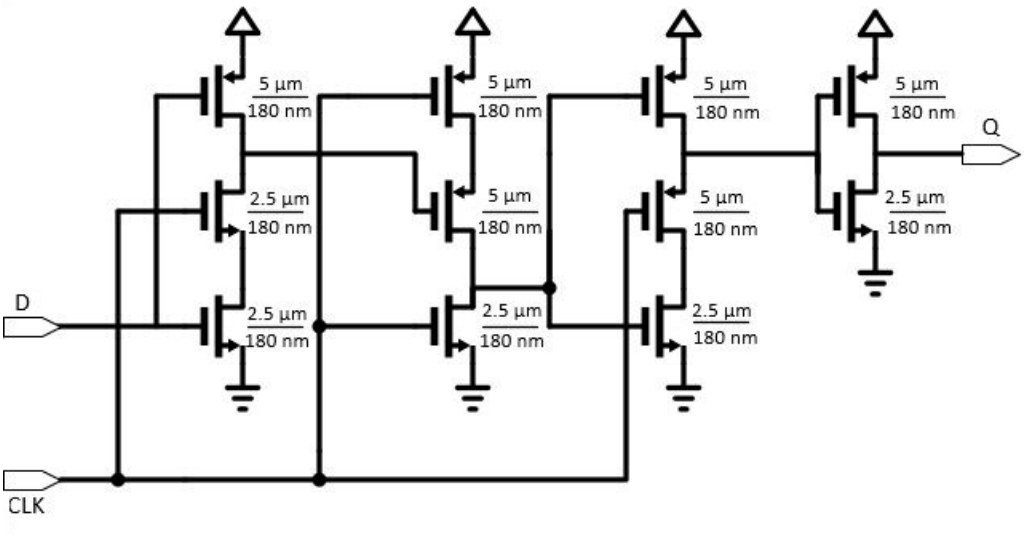

Figure 5.16: A TSPC negative edge-triggered flipflop

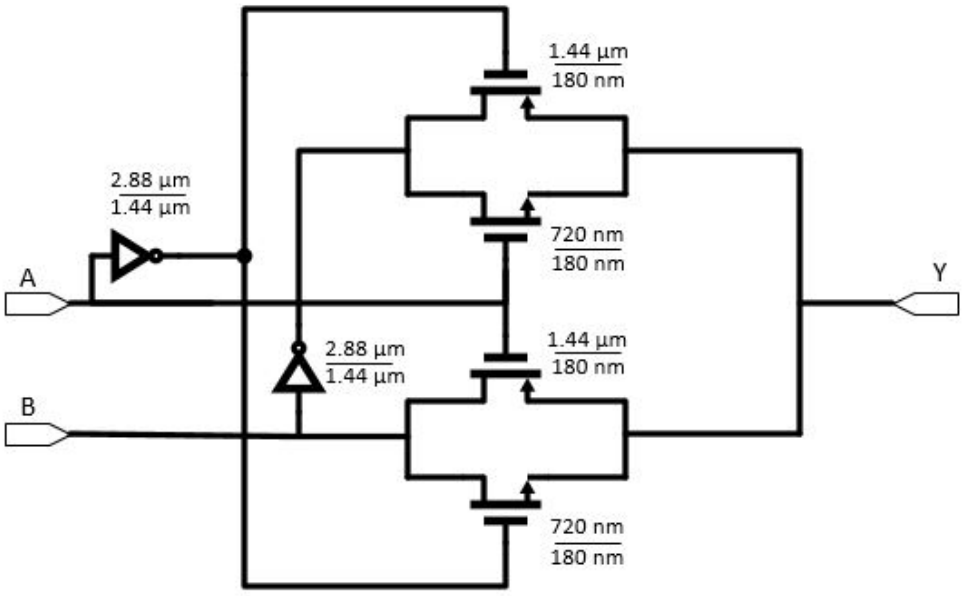

Figure 5.17: Transmission gate XOR circuit

of the shown topology are used for each XOR gate with one copy performing the XOR function and the other performing the XNOR function. The XNOR function can be obtained by flipping the two drain inputs of the transmission gate. This ensures that the skew between UP (DOWN) and UP_bar (DOWN_bar) is kept to a minimum.

A vital step during the transistor level design of the each circuit is to ensure that it behaves exactly the way we want it to, by testing the circuit using test waveforms. Each of the above circuits was rigorously tested to ensure the same behavior as that obtained from the behavioral modeling discussed in Chapter 4 to ensure correct system level functionality. The results are presented in Chapter 8.

CHAPTER 6

COMPLEMENTARY LOGIC CDR DESIGN

In this chapter, we explore the design of the CDR using complementary logic, i.e., a logic family in which a single value is represented by two lines, true and complement, also referred to as plus and minus. Although it is tempting to think of this logic family as differential logic family, it cannot be treated as a truly differential logic family since the voltage levels for the high and low levels are designed to be VDD and GND respectively and the signal is not represented as the difference between two voltages. However, the topology is still advantageous from a signal integrity point of view since the channel is still differential, allowing the common mode noise and interference to cancel each other out. The block diagram of the system is shown in Figure 6.1. As before, several other blocks are also designed to completely evaluate the system and the serializer-deserializer blocks are not considered. It is important to note that each arrowhead in Figure 6.1 represents two lines: true and complement.

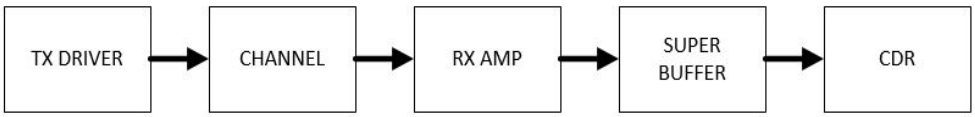

Figure 6.1: Complementary logic SERDES system

6.1 Transmit driver

The transmit driver amplifier is shown in Figure 6.2 and is very similar to the one discussed for the single ended case in Section 5.1. The only difference is that both Yp and Yn outputs are used to drive the two conductors

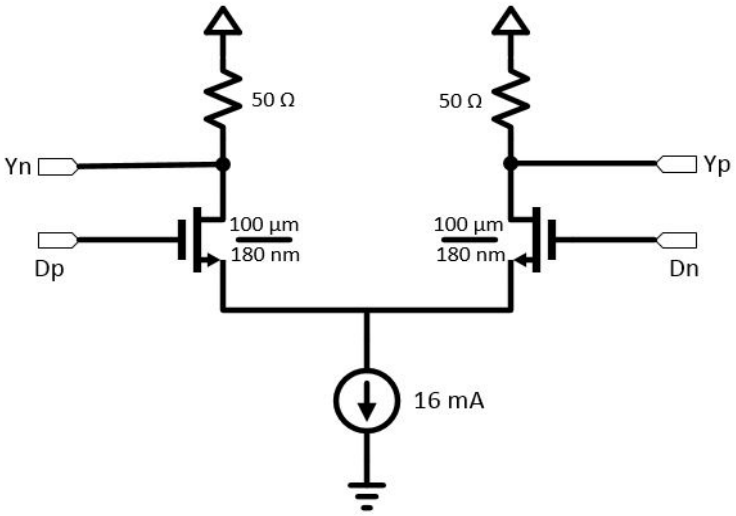

Figure 6.2: Circuit diagram of the transmit driver

of the differential channel, thereby providing better noise immunity. The design procedure is identical to the single-ended scenario. The driver amplifier terminates both the conductors with a 50 Ω impedance to minimize reflections. In addition to matching the single-ended impedance, it is also crucial to match the differential impedance of the channel in order to truly eliminate any reflections. Since it is not straightforward to accurately control the differential impedance of the amplifier, we design the channel to ensure that the differential impedance of the channel is identical to that of the amplifier.

We first measure the differential impedance of the amplifier using Spectre simulations using the testbench shown in Figure 6.3. We first bias the circuit at the typical operating point, i.e., INp at high (1.8 V) and INn at low (1.3 V). We now connect two test sources at the outputs, each with small signal AC amplitudes of 1 V and -1 V, respectively, via switches which block DC components and allow only AC components. Now we perform an AC simulation and measure the currents through both test sources at 1 GHz, which is our frequency of interest. Ideally, these two currents should be the same, but due to transistor non-idealities, they are slightly different and hence we take the average of the two currents. We then compute the differential impedance of the amplifier using the expression:

$$Z(diff) = \frac{V(diff)}{I_{one}} \tag{6.1}$$

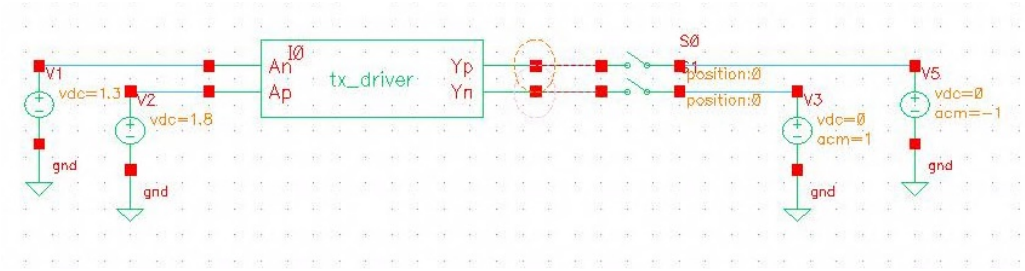

Figure 6.3: testbench used to measure the differential impedance of the transmitter

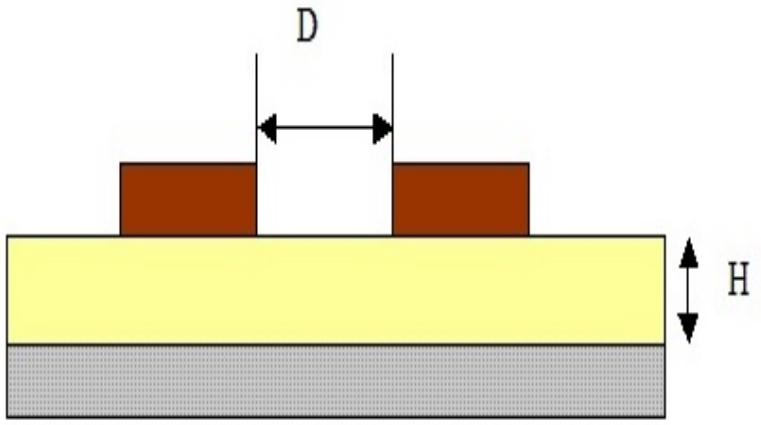

Figure 6.4: Cross-section of the differential channel

The differential impedance of the amplifier was found to be approximately 91 Ω.

6.2 Channel

The channel is a pair of differential microstrip lines on a PC motherboard. The channel is modelled using the 2D electromagnetic field solver tool Q3D. Each of the two traces has a characteristic impedance of 50 Ω and is designed using the same parameters as the single-ended channel in Section 5.2. The new variable in this design is the spacing between the two lines, which affects the coupling between them and hence the differential impedance. Since we want to match the differential impedance of the channel to that of the amplifier, we design the channel to have a differential impedance of 91 Ω. The

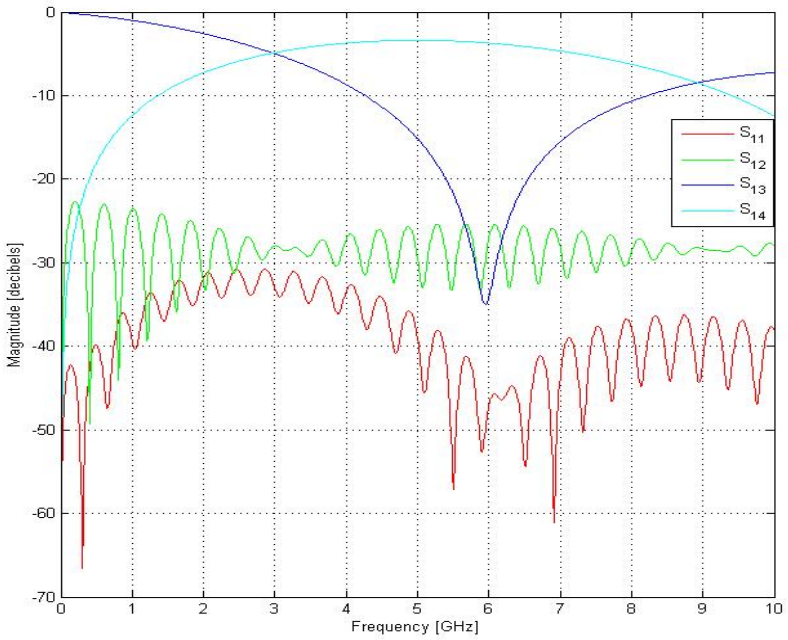

Figure 6.5: S parameters of the designed differential channel

differential impedance of the channel is given by [28]:

$$Z_{diff} = 2Z_O(1 - 0.48e^{\frac{-0.96D}{H}}) \tag{6.2}$$

where Z_O is the characteristic impedance of each single-ended line which is 50 Ω in this case, D is the distance between the two conductors and H is the height of the dielectric substrate.

Thus, for our PCB stack-up using a 0.8 mm thick FR4 substrate, we obtain the spacing to be 1.394 mm. A cross-section of the channel is shown in Figure 6.4. The frequency response of the channel is shown in Figure 6.5

6.3 Receiver

The receiver again consists of an amplifier to increase the swing of the signal and a superbuffer to drive the big load offered by the CDR circuit. Figure 6.6 shows the amplifier circuit, which is very similar to the one used in the single-ended case in Section 5.3.1. The only difference is that the first stage

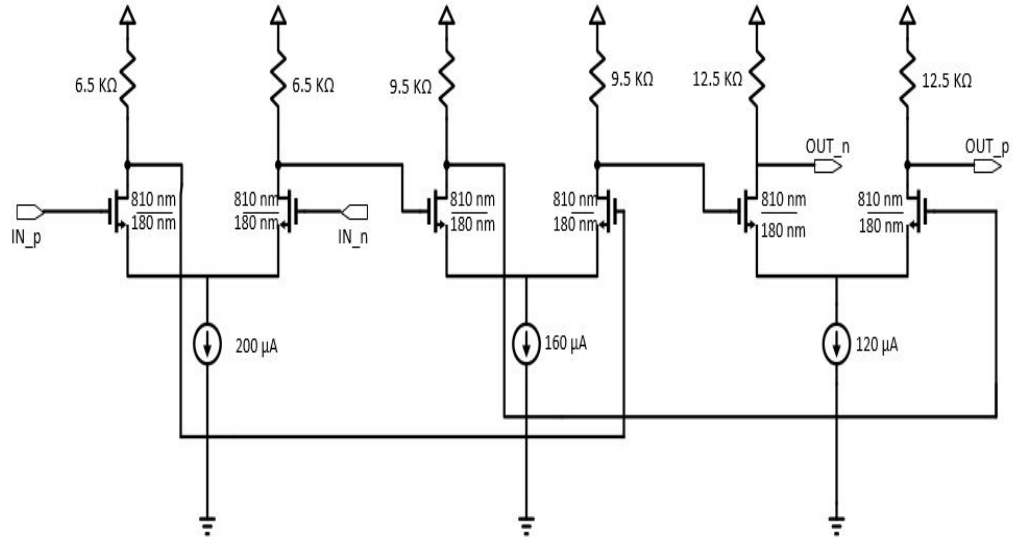

Figure 6.6: The 3-stage differential amplifier used in the design

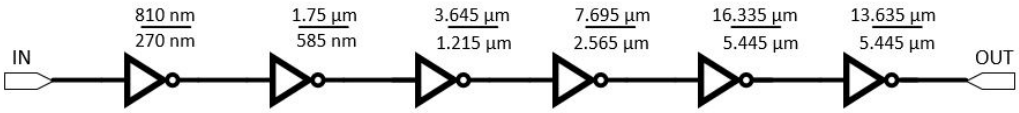

Figure 6.7: Super buffer circuit used to drive the CDR

has both inputs fed by the incoming differential signal instead of a single-ended signal and a constant bias. Figure 6.7 shows the superbuffer circuit. It is essential to realize that two copies of the circuit are used to drive the two complementary signals to the CDR. The inverters in the superbuffer circuit are sized to have equal rise and fall delays, in order to minimize the skew between the two complementary signals.

6.4 Clock and data recovery (CDR) circuit

Figure 6.8 shows the block diagram of the CDR circuit, with the incoming data coming from the superbuffer. Once again, the logical effort technique is used to minimize the delay and area of the circuit, and a bottom-up design methodology is used. Each arrowhead in Figure 6.8 represents two lines: true and complement. The CDR loop parameters are chosen to be identical with

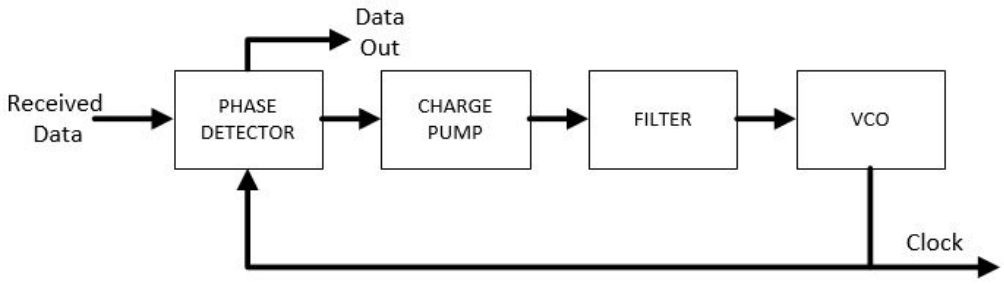

Figure 6.8: Block diagram of the CDR system

the single-ended case in order to provide a fair comparison. We will now look at each block in detail.

6.4.1 Filter

The filter is identical to the one used in Section 5.4.1 for the single-ended case.

6.4.2 Charge pump

The charge pump is also identical to the one used in Section 5.4.2 for the single-ended case. However, in this case the complementary signals are directly available and hence have much smaller skew, thereby making the charge pump more effective.

6.4.3 Phase detector

We implement a Hogge phase detector using complementary logic signals, which is a linear phase detector providing complementary UP and DOWN signals that reflect the phase relation between the clock and the incoming data stream. It consists of flipflops and XOR gates as shown in Figure 6.9.

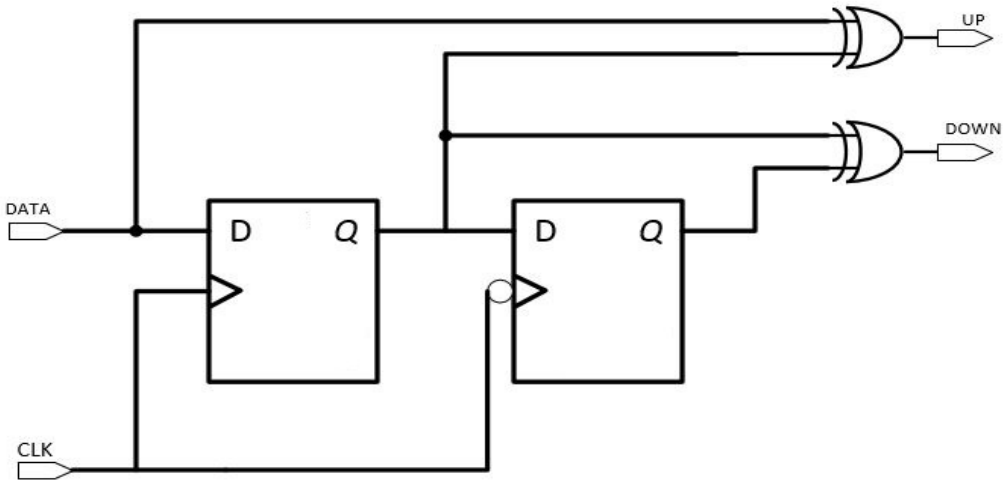

Figure 6.9: Block diagram of the Hogge phase detector

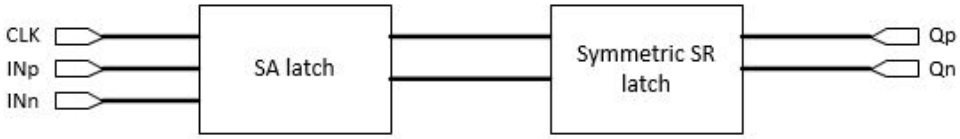

Figure 6.10: Generic realization of a sense amplifier flipflop circuit

Positive edge-triggered flipflop

A flipflop typically consists of two latch stages: a master latch and a slave latch. We make use of a sense amplifier (SA) latch for the master stage. For the slave stage, we make use of a symmetric SR latch. The block diagram of the flipflop is shown in Figure 6.10.

SA latch The transistor-level schematic of the SA latch is shown in Figure 6.11. It is a precharge type of latch, whose operation can be explained using Figure 6.12. When CLK is low, the output nodes are precharged to high. When CLK is high, the pull-down network is activated and depending on the value of the inputs, one of the two output nodes is pulled down to low faster than the other. This is referred to as the sampling phase. Once the first output node is pulled down below a certain minimum value the cross coupled inverters reinforce the change of state and cause that output node

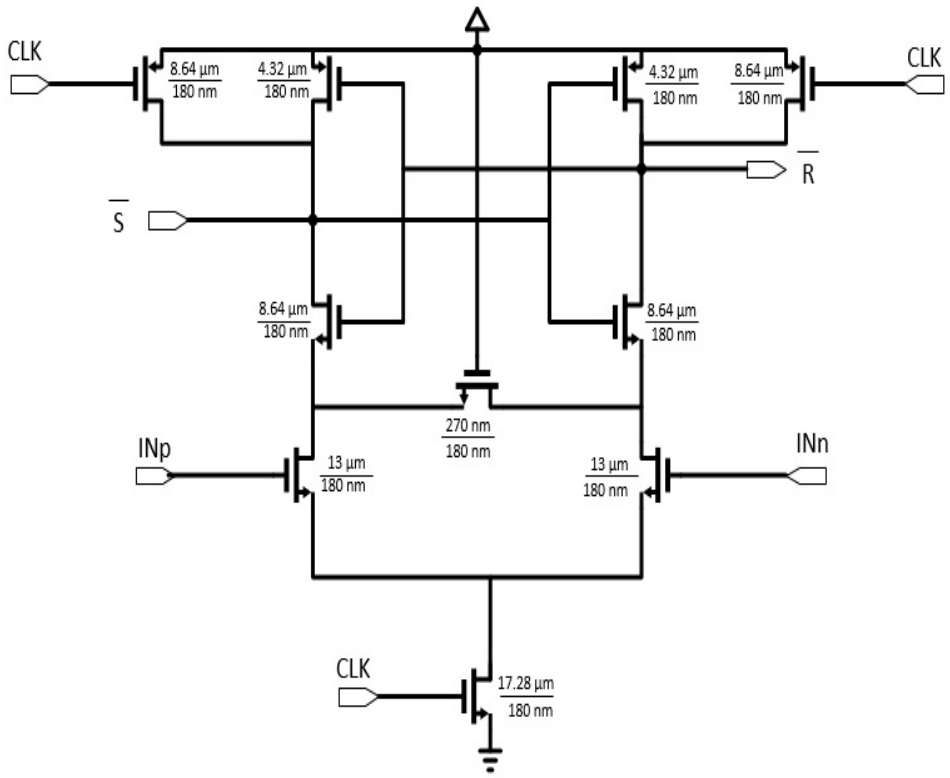

Figure 6.11: Circuit diagram of a sense amplifier latch

to be pulled down faster, while pulling up the other output node. This is referred to as the regeneration phase. Once the output nodes have been pulled above or below the logic thresholds, subsequent circuitry can process the information. This is referred to as the decision phase. The SA latch often has a negative setup time, i.e., the input can change even after the CLK goes high until a certain time. However, this will have an adverse impact on the delay of the flipflop.

Symmetric SR latch The output of the SA latch can be processed by a simple SR latch to produce complementary outputs Q and $\bar{Q}$. However, traditional SR latches have asymmetric delays between Q and $\bar{Q}$, which cause skew between the signals and can adversely affect the operation of subsequent circuits. Thus, we make use of a symmetric SR latch topology proposed by Nikolic and Oklobdzija [29] which is shown in Figure 6.13. The SA latch stage produces the outputs $\bar{S}$ and $\bar{R}$, and we make use of inverters to produce the signals S and R which are not shown in the figure. The function of the SR

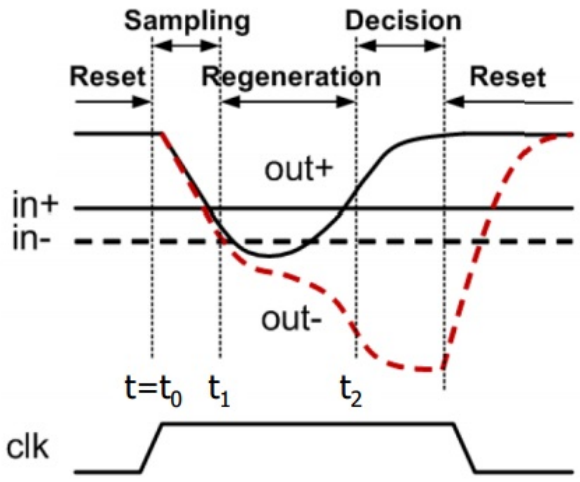

Figure 6.12: Waveform showing the operation of the sense amplifier latch

latch can be explained through the truth table shown in Table 6.1. When both $\bar{S}$ and $\bar{R}$ are high, the latch retains its previous value. Depending on which one of $\bar{S}$ and $\bar{R}$ is pulled low, either Q or $\bar{Q}$ is pulled high, respectively. Having both $\bar{S}$ and $\bar{R}$ at low forces the latch into race condition, which results in metastability, and this condition is avoided by careful design of the SA latch.

Table 6.1: Truth table showing the operation of the symmetric SR latch

$\bar{\mathbf{S}}$	$\bar{\mathbf{R}}$	**Q**	$\bar{\mathbf{Q}}$	**STATE**
0	0	X	X	Race condition
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_{prev}	$\bar{Q}_{prev}$	Memory

Negative edge-triggered flipflop

The negative edge-triggered flipflop is constructed in a manner similar to that of the positive edge-triggered flipflop. The SA latch used in this case is now negative level sensitive and is the exact dual of the previously described circuit and is shown in Figure 6.14. The SR latch is also very similar to the

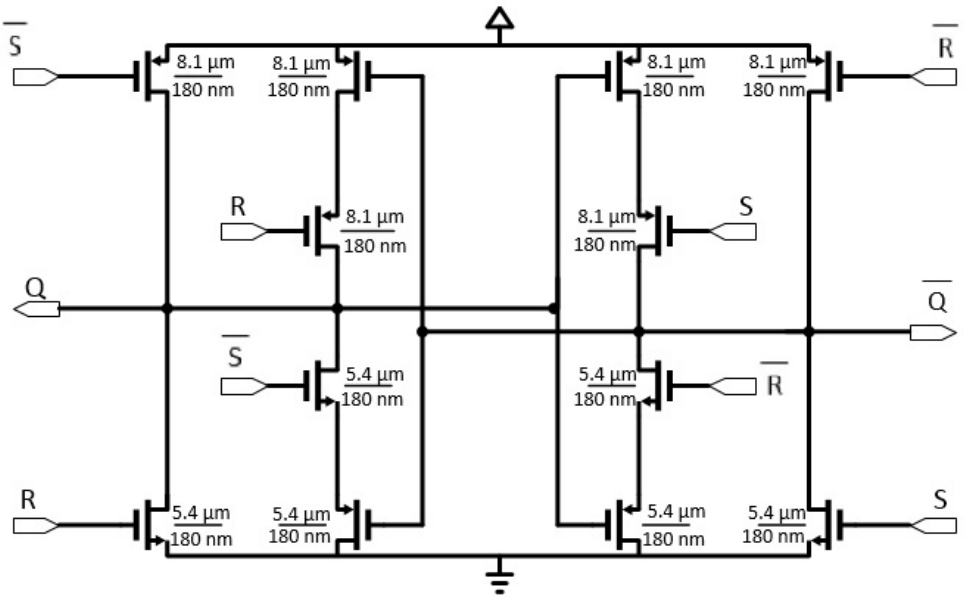

Figure 6.13: Circuit diagram of the symmetric SR latch

previously described circuit and is shown in Figure 6.15. It is to be noted that the transistor sizes are much smaller than the positive edge-triggered case, since the load is much smaller in this case.

XOR gate

We employ the same transmission gate XOR topology previously described for the single-ended case in Section 5.4.3. The key difference is that we do not need inverters to generate complementary signals since we use complementary logic in the preceding stages. The transistor-level schematic is shown in Figure 6.16. Once again, we utilize two copies of the circuit to produce complementary outputs, i.e., the XNOR function is implemented by swapping the drain inputs of the transmission gates.

As before, the circuits are rigorously tested to ensure correct functionality with reference to the behavioral model to ensure correct system level functionality. The results are presented in Chapter 8.

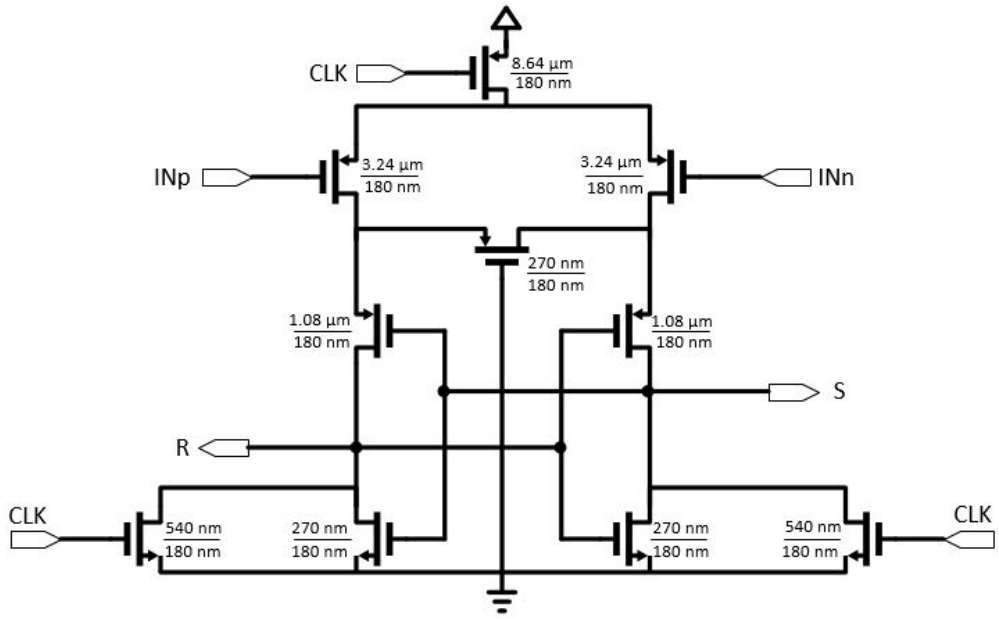

Figure 6.14: Circuit diagram of the sense amplifier latch(neg)

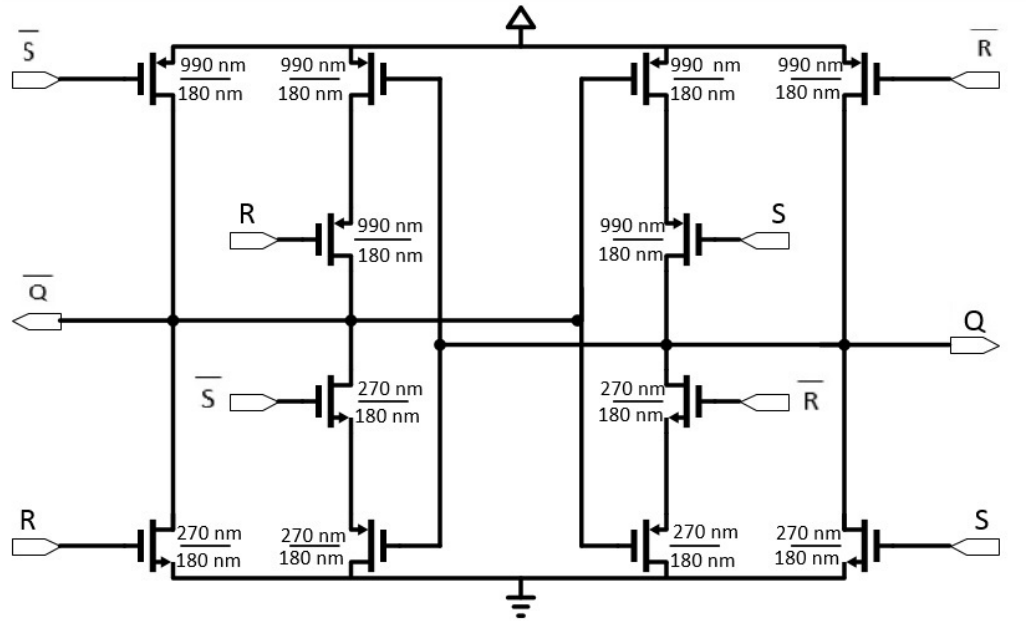

Figure 6.15: Circuit diagram of the symmetric SR latch(neg)

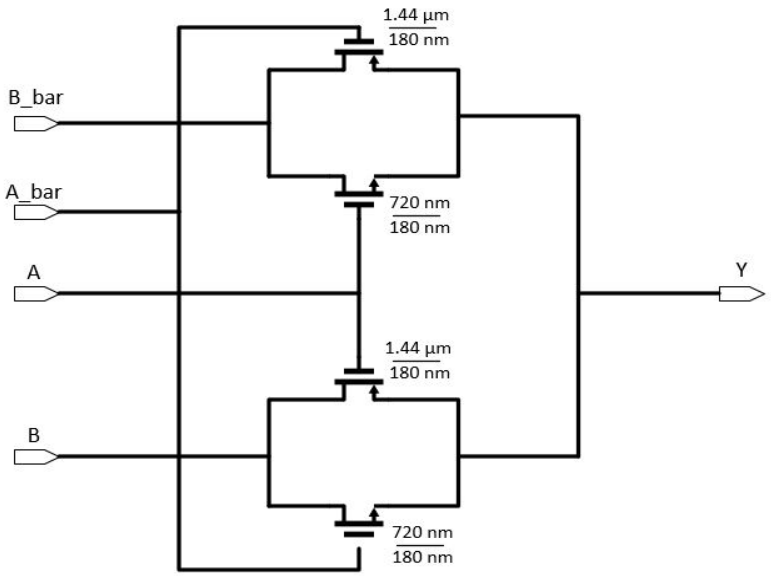

Figure 6.16: Circuit diagram of the transmission gate XOR

CHAPTER 7

CURRENT MODE LOGIC CDR DESIGN

In this chapter, we look at the design of the CDR using the current mode logic (CML) family. CML circuits are essentially analog circuits implementing digital functions. They work by steering the current between two different arms based on the value of inputs, providing differential outputs. Thus, CML is a completely differential circuit, where each signal is represented by the difference between two lines. It is different from complementary logic in the sense that the voltage levels on the two lines are not full swing CMOS voltages. Instead, the difference in voltage levels is much smaller, allowing the circuit to switch much faster. The higher speed comes with the drawback of increased power consumption since there is always some current flowing between VDD and GND. Thus, CML is used in high performance circuits where speed is critical and larger power consumption can be tolerated. Thus, CML is a good candidate for CDR circuits, especially with increasing data rates. The block diagram of the CML SERDES system is shown in Figure 7.1. Since CML circuits can work with smaller voltage swing, the need for an amplifier at the receiver is eliminated, thereby providing some power savings. We now look at each block in detail.

Figure 7.1: Block diagram of the SERDES system

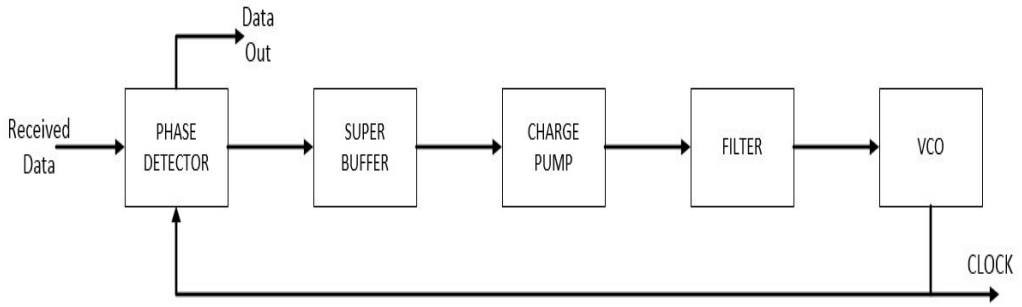

Figure 7.2: Block diagram of the CDR system

7.1 Transmit driver

The driver amplifier is identical to the one used for the complementary case in Section 6.1. This is because the transmit driver used for the previous two cases is essentially a current mode driver, which works by steering current between the two arms. The voltage level for high is 1.8 V and that for low is 1.4 V. Thus, a logic high is represented by a 400 mV difference signal and a logic low is represented by a -400 mV difference signal. We design the CDR to work with the same voltage swing.

7.2 Channel

The channel is identical to the one used for the complementary case in section 6.2, since we only require a differential channel and the transmit drivers are identical.

7.3 Clock and data recovery

The block diagram of the CDR circuit is shown in Figure 7.2. The major difference from the previous architectures is that we now have a superbuffer to drive the charge pump. This allows the phase detector to be much smaller and have smaller drive currents, thereby saving power. We now look at each block in detail.

7.3.1 Filter

The filter is identical to the ones used in the previous two cases since we have the same CDR loop parameters.

7.3.2 Charge pump

The charge pump is also identical to the ones used in the previous two cases since we made use of a fully differential charge pump which is compatible with CML voltage levels.

7.3.3 Superbuffer

Since the charge pump circuit offers a big capacitive load, we use a superbuffer to drive the signals from the phase detector. This allows the phase detector to be much smaller, thereby reducing the drive currents in the CML logic cells, which helps to save power. It also reduces the load offered to the channel.

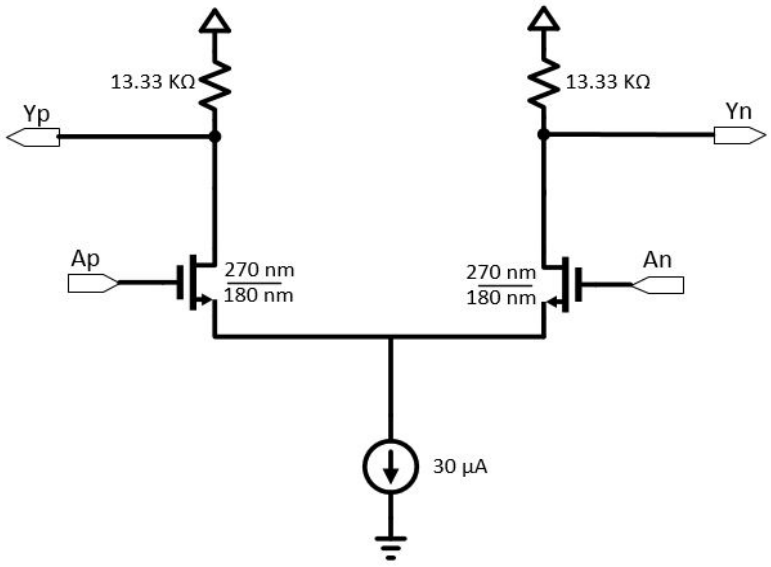

Figure 7.3: Circuit diagram of a minimum drive strength CML inverter

A superbuffer is a chain of cascaded inverters of progressively increasing size or drive strength. In this case, the inverters are CML inverters. To understand the operation of CML circuits, it is crucial to understand the

operation and sizing of the CML inverter. Once the inverter is sized correctly, other circuits can be sized using the equivalent inverter size method [30]. Figure 7.3 shows a CML inverter of minimum size and drive strength, which will be referred to as INVD1, where INV stands for inverter and D1 stands for unit drive strength.

The current source is realized with a current mirror circuit which is not shown here. Clearly, regardless of the value of the inputs, there is a 30 μA current flowing between VDD and GND. This leads to much higher power consumption than static CMOS, where there is no direct path current at steady state. Depending on which of Ap and An has a higher voltage, the current is steered predominantly to that arm. Thus, if Ap is at 1.8 V and An is at 1.4 V, the majority (if not all) of the current will flow through the left branch, resulting in a much greater voltage drop across the resistor on the left than that on the right. Thus, Yp will have a smaller voltage than Yn and the signal has been inverted.

The sizing of CML circuits is very similar to the sizing of differential amplifiers. The values of the resistors, transistor widths and the current source have to be chosen in accordance with the input and output voltage swings. Let us assume that we want the output to have the same swing as the input i.e. between 1.4 V and 1.8 V. In this case, we are essentially designing a unit gain differential amplifier. First, we select the value of the current source to

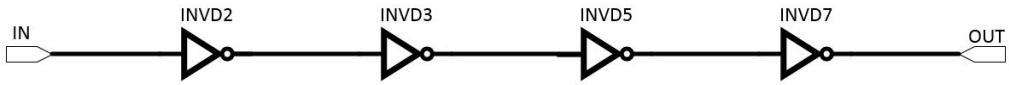

Figure 7.4: The superbuffer circuit used to drive the CDR

be a reasonable value that can drive a minimum size inverter, i.e., 30 μA. By using Ohm's law, we can directly compute the value of the resistor:

$$\Delta V = IR \tag{7.1}$$

where ΔV is 400 mV and I is 30 μA. This gives the value of the resistor as 13.33 KΩ.

Now we choose the width of the transistors such that one of them (the ON transistor) will be in saturation region and the other will be cutoff. Another

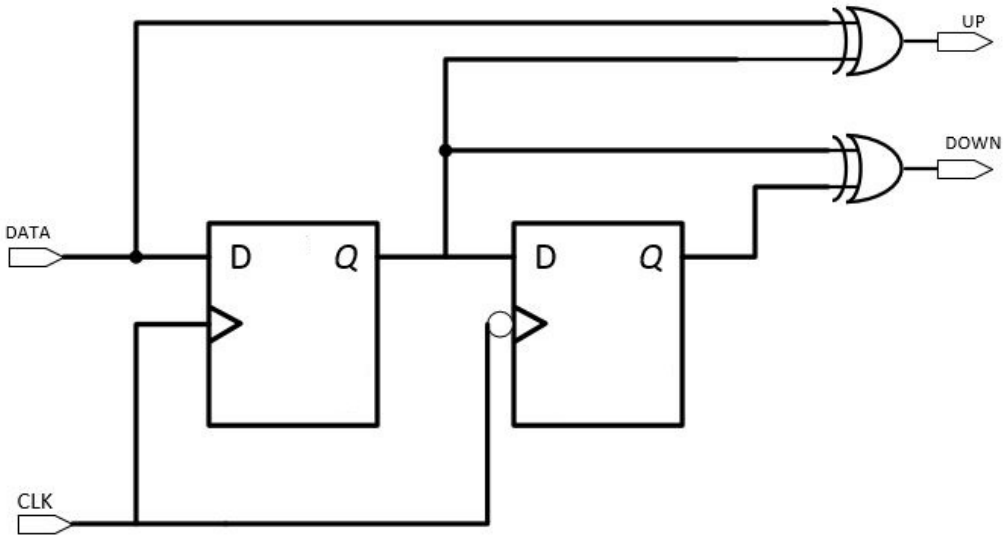

Figure 7.5: Block diagram of the Hogge phase detector

consideration is to provide the minimum voltage required by the current mirror to provide 30 μA of current. A value of 270 nm is found to satisfy the requirements and is therefore used for this minimum drive strength CML inverter.

Once we design INVD1, designing higher drive strength inverters is very straightforward. For example, INVD2 can be designed with double the transistor widths and drive current, with one half the resistor value. Thus, we are now in a position to design an inverter of any required drive strength and can therefore design the superbuffer.

The CML superbuffer chain is designed using standard supersbuffer sizing techniques and is shown in Figure 7.4. Again, each line is a differential pair with plus and minus signals. We use two copies of the circuit: one for the UP signal and one for the DOWN signal.

7.3.4 Phase detector

Once again, we make use of the Hogge phase detector, which is a linear phase detector that provides the UP and DOWN signals to control the charge pump. As shown in Figure 7.5, it consists of flipflops and XOR gates, all of which are realized using CML, as discussed below.

Positive edge-triggered flipflop

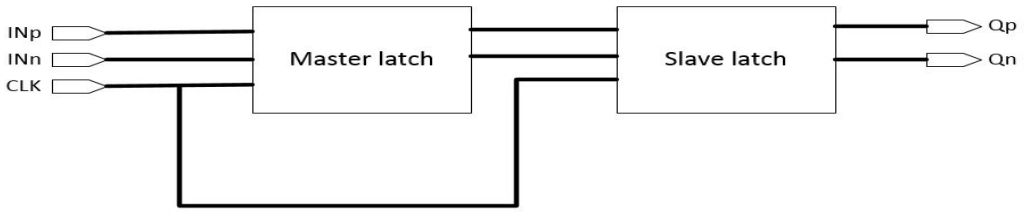

Figure 7.6: Master-slave realization of a flipflop

The positive edge-triggered flipflop is constructed using the traditional master-slave topology which is shown in Figure 7.6. This topology is slightly different from the sense amplifier latch discussed in Chapter 6 in the sense that both latches are clocked. The first latch is a negative level-sensitive latch and second latch is a positive level-sensitive latch with respect to the clock. Thus, the entire assembly behaves as a positive edge-triggered flipflop.

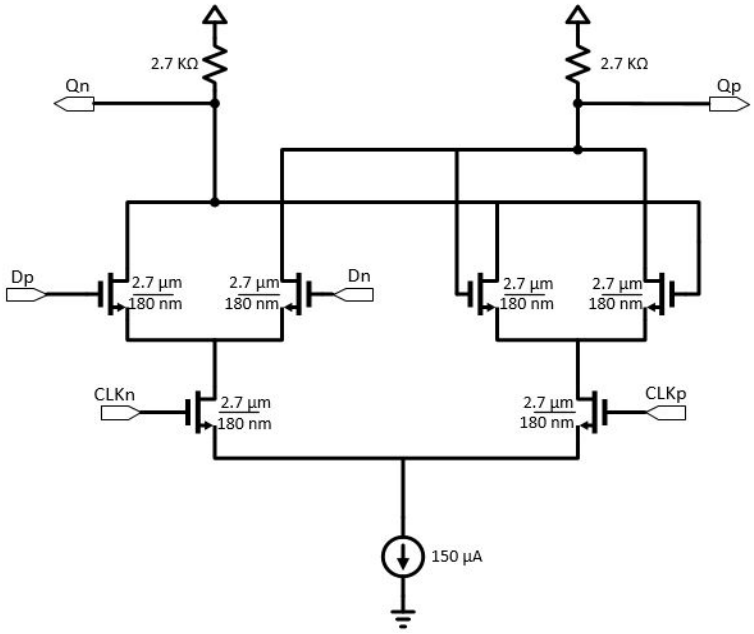

Figure 7.7: A CML latch circuit

Figure 7.7 shows the master latch circuit. When CLKn is high, the branch on the left gets activated and depending on the value of the inputs Dp and

Dn, the output states change. Once CLKn goes low, CLKp goes high, and the values at the output nodes are retained using a cross-coupled inverter configuration seen on the right. Thus, the circuit behaves as a negative-level sensitive latch.

We size the latch according to the equivalent inverter width method. We first decide to have a latch with a driving strength of 5X. Therefore, we use a current source of 5 x 30 μA = 150 μA. Also, we need transistors with an equivalent width of 5 x 270 nm = 1.35 μm. Since each branch of the latch has two NMOS transistors in series, the equivalent width of each branch will be half the width of each transistor. Thus, we obtain the width of each transistor as 2.7 μm. Lastly, the value of the resistor is obtained by dividing the value of the resistor used in the inverter by 5, yielding about 2.7 kΩ.

The slave latch is identical to the master latch in terms of sizing and topology. However, the CLKp and CLKn inputs are swapped to ensure that the slave latch is positive level sensitive to the CLK.

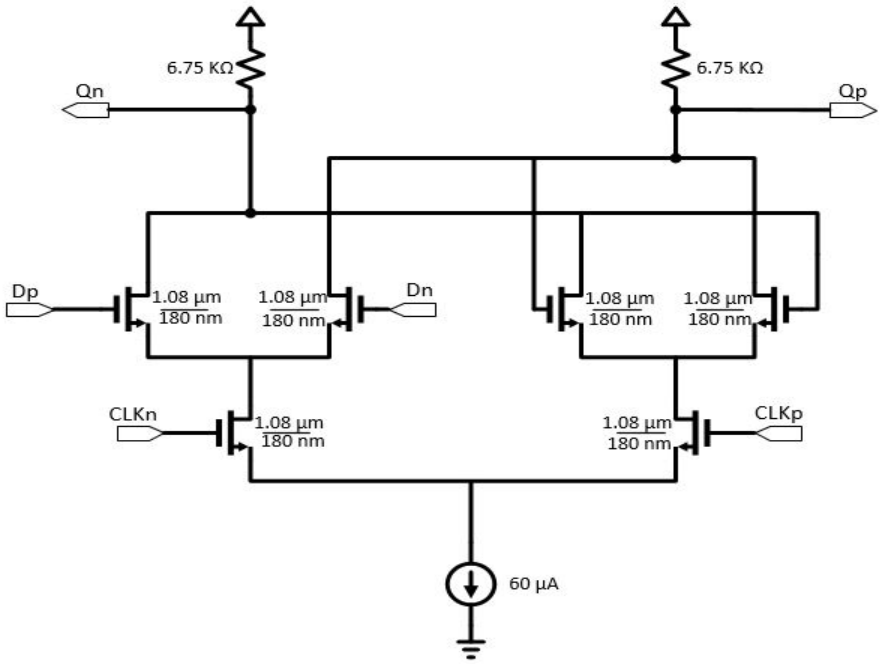

Figure 7.8: The CML latch used in the negative edge-triggered flipflop

Negative edge-triggered flipflop

We again go for the master slave topology to implement the flipflop. The difference is that the master latch is positive level sensitive and the slave latch is negative level sensitive to the CLK. This ensures that the circuit acts as a negative edge-triggered flipflop. Figure 7.8 shows the latch circuit used. As expected, the circuit is much smaller than the one used for the positive edge-triggered flipflop since the load is much smaller.

XOR gate

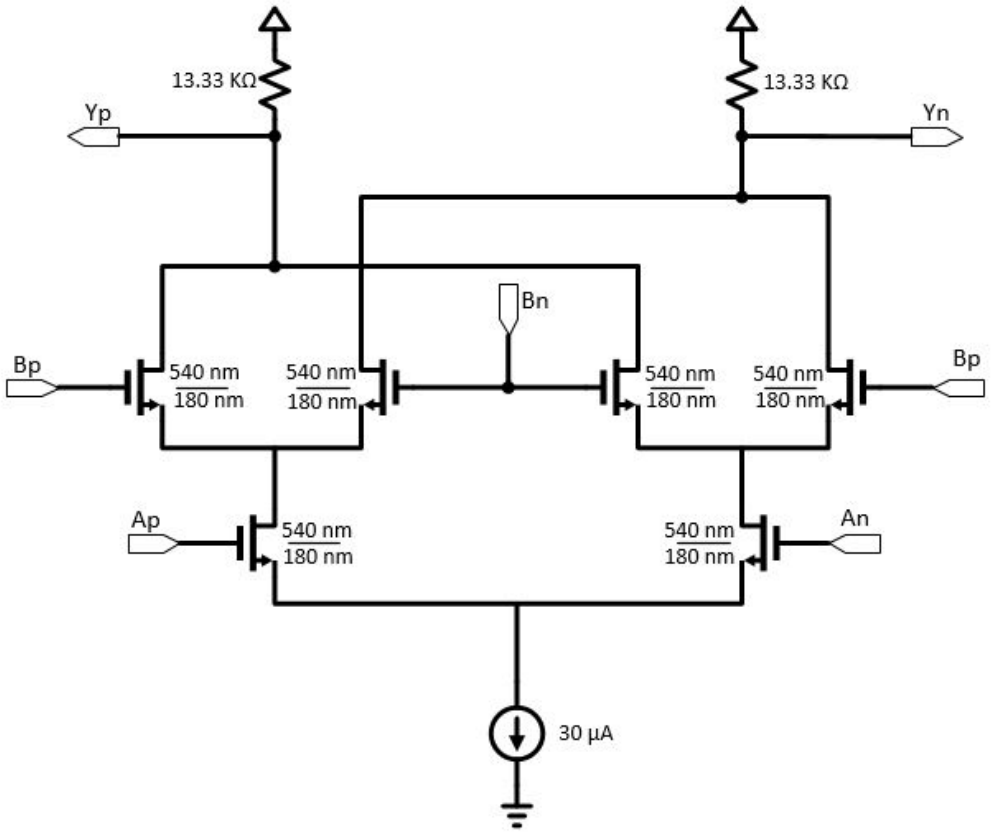

Figure 7.9: The CML XOR gate

Figure 7.9 shows the CML XOR circuit used in the design. The circuit topology is very similar to the one used for the latch. This is because both are implemented using a MUX circuit and the inputs are chosen accordingly. Since a 2:1 MUX can be used to implement any logic function, we can there-

fore design any arbitrary Boolean function using CML logic gates. The drive strength of the gate can be easily modified by scaling the values of the current source, transistor widths and the resistors accordingly. In this case, a minimum drive strength XOR gate is used since we only need to drive the superbuffer circuit which will drive the much bigger charge pump circuit.

As always, each block is verified to have the same behavior as that obtained from the behavioral level models in order to ensure correct system-level functionality. The results are presented in Chapter 8.

CHAPTER 8

RESULTS

In this chapter, we present the results obtained from simulating the three CDR designs. We also provide a comparison of the architectures which can help to evaluate design tradeoffs. The designs are tested using a pseudo-random bit sequence (PRBS) of length $2^{32} - 1$. Since we are operating at 2 Gb/s with a bit period of 500 ps, simulating the entire length of the PRBS will take an unreasonable amount of time. Therefore, we simulate only 20000 cycles of the sequence, i.e., 10 μs. This is sufficient to characterize the system with reasonable accuracy.

8.1 Single-ended CDR

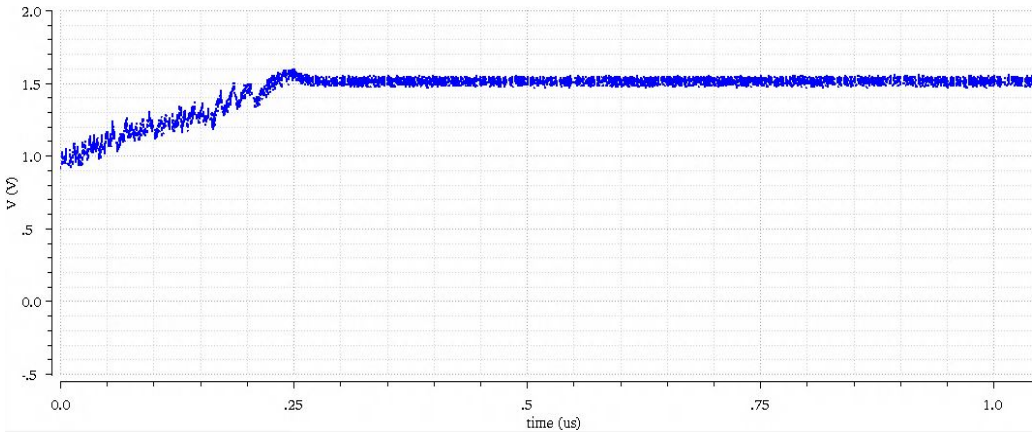

Figure 8.1: Locking behavior of the control voltage

It is important to verify that the CDR actually locks to the frequency of the incoming data stream and also locks at the right phase for optimum sampling. This can be accomplished by looking at the control voltage as a function of time as shown in Figure 8.1. Clearly, the control voltage starts from its initial

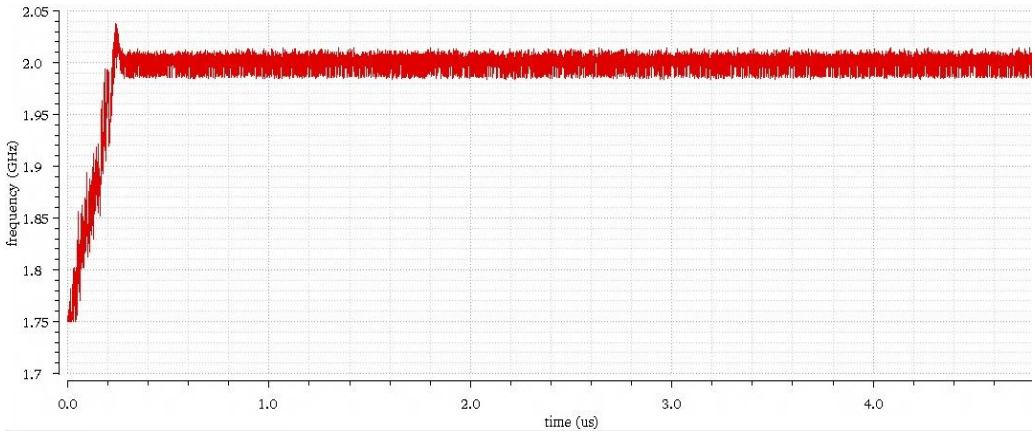

Figure 8.2: VCO frequency as a function of time showing lock acquisition

value and climbs up to the value required to provide the correct voltage for the VCO, which in this case is 1.5 V. Since we have designed a slightly over-damped system, the control voltage quickly settles to the required value in about 250 ns.

The locking behavior can also be verified by looking at the instantaneous frequency of the generated clock as shown in Figure 8.2. The frequency climbs up from 1.75 GHz to quickly settle at the desired value of 2 GHz. We can also see that the ripples in the control voltage cause very slight changes in the frequency of the generated clock, which get translated to jitter at the output. However, the instantaneous frequency hovers closely around 2 GHz, thereby sampling the data correctly.

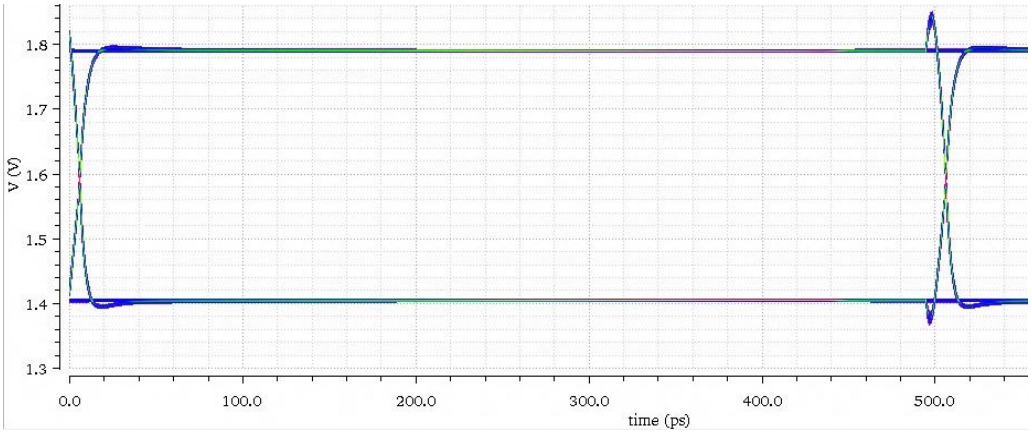

Figure 8.3: Eye diagram at the output of the transmitter

An important metric to characterize the various blocks of the SERDES system is the eye diagram. The eye diagram gives important details such as the peak-to-peak jitter and noise levels in the signal. The peak-to-peak jitter can be obtained by looking at the time difference between the earliest and the latest transition at the eye crossing points. The noise level can be obtained by looking at difference between the highest and lowest values for logic low or logic high at the ideal sampling point, i.e., the center of the eye. The eye diagram at the output of the transmitter is shown in Figure 8.3. It is easy to observe that the eye is very clean with minimal jitter and noise. We can also see that there are almost no reflections from the channel, thereby verifying that it is terminated correctly.

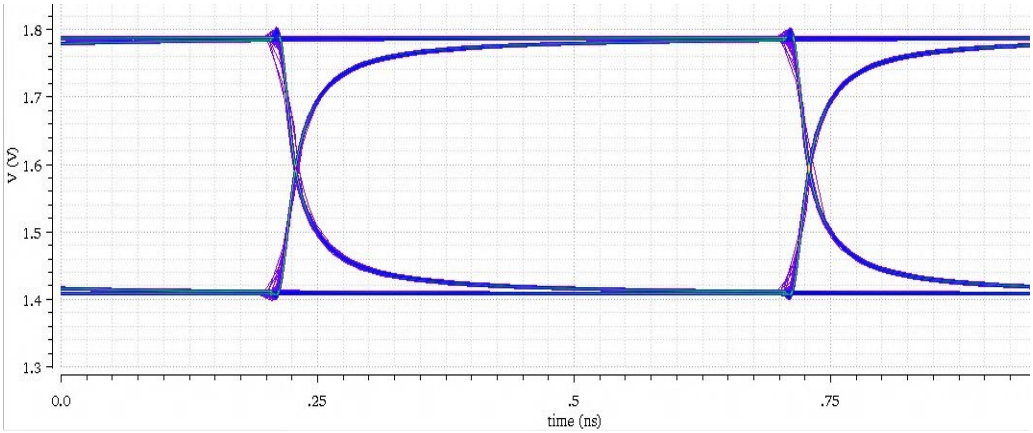

Figure 8.4: Eye diagram at the output of the channel

We now examine the eye diagram at the output of the channel in Figure 8.4. We can clearly visualize the non-idealities of the channel which lead to dispersion, intersymbol interference, rise time degradation, etc., and distort the eye. We can also notice that the channel has caused an increase in the jitter and noise levels of the signal.

To characterize the actual performance of the CDR, we look at the eye diagram of the retimed data which is shown in Figure 8.5. We can see that the CDR has recovered the data correctly while compensating for the non-idealities of the channel. However, this comes at the cost of slightly higher jitter due to the rippling of the control voltage around the final value. One noteworthy feature of the eye is the asymmetry between the rising and falling transitions. This is due to the mismatch between the pull-up and pull-

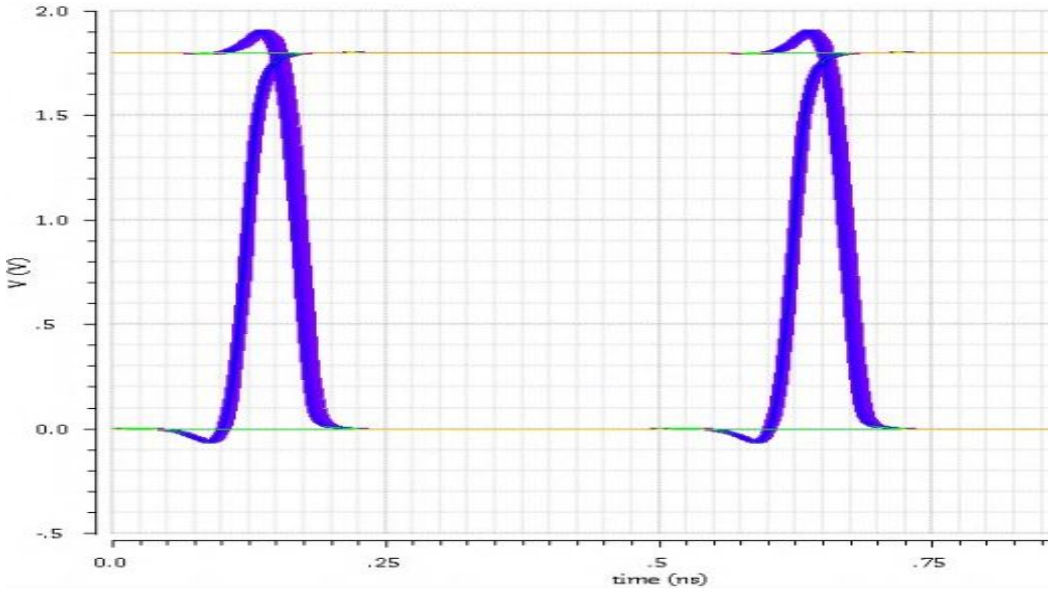

Figure 8.5: Eye diagram at the output of the CDR

down networks and duty cycle distortion caused by the receiver amplifier. This property is undesirable since it leads to loss of integrity of the signal. However, it does not significantly affect the operation of subsequent digital circuitry.

Table 8.1: Summary of eye diagram characteristics for the single-ended design

EYE	HEIGHT (mV)	WIDTH (ps)	NOISE (mV)	JITTER (ps)
Transmitter	380	500.5	8	0.2
Channel	357	494.5	24	5.5
CDR	1800	437	1	27

Table 8.1 summarizes the characteristics of the previously shown eye diagrams and provides values for jitter, noise margin, sampling window, etc. We can also examine the offset from the ideal sampling point of the generated clock as shown in Figure 8.6. We see that the clock samples the data about 80 ps to the left of the center of the eye.

We can estimate power consumption by calculating the average current

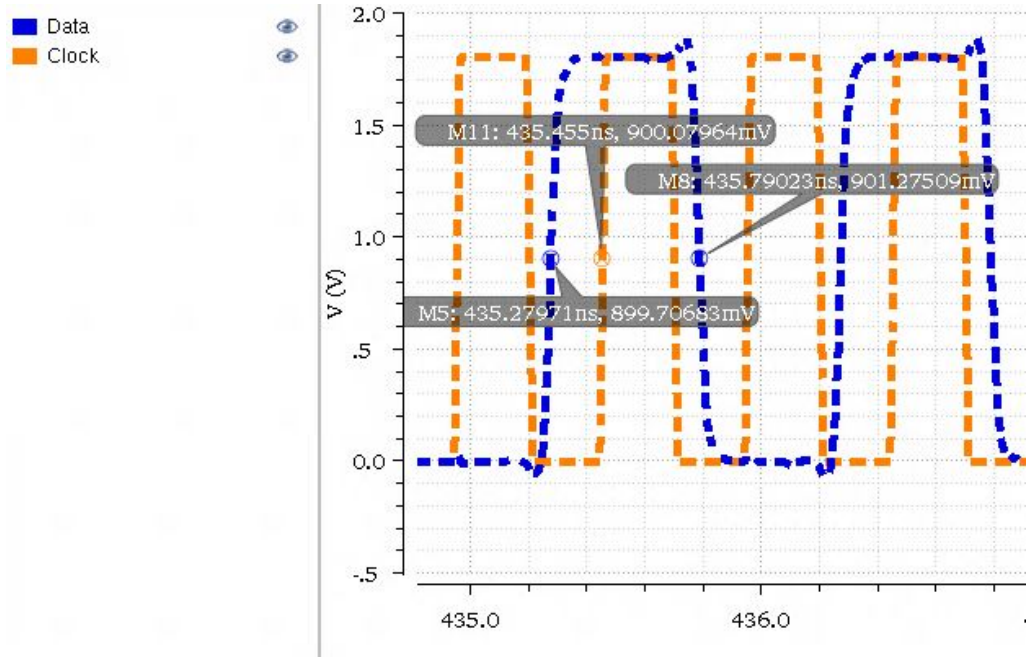

Figure 8.6: Waveform showing the phase offset in sampling the data by the CDR

drawn from the power supplies. The total power consumption of the SERDES system is about 34 mW and that of the CDR alone is about 3.5 mW.

8.2 Complementary logic CDR

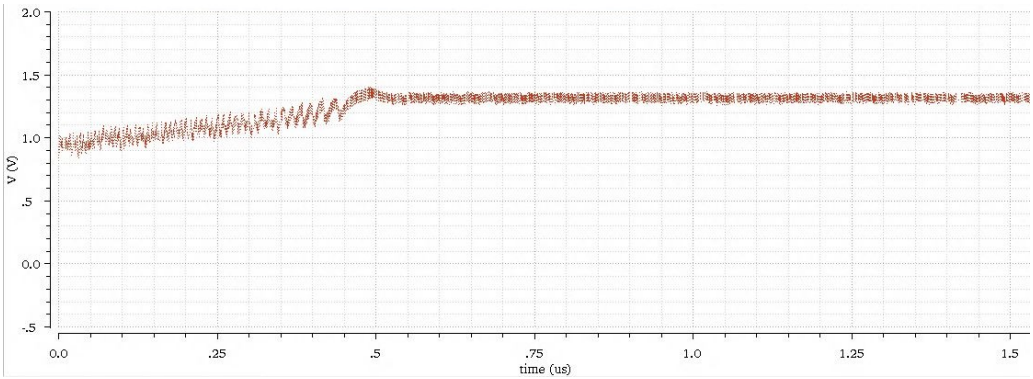

Figure 8.7: Locking behavior of the control voltage

We now look at the performance of the complementary logic CDR. We first

examine the locking behavior of the control voltage as shown in Figure 8.7. Similar to the single-ended case, the control voltage quickly settles to its final value within 500 ns. This is also confirmed by looking at the instantaneous frequency of the generated clock as shown in Figure 8.8 and the results are consistent.

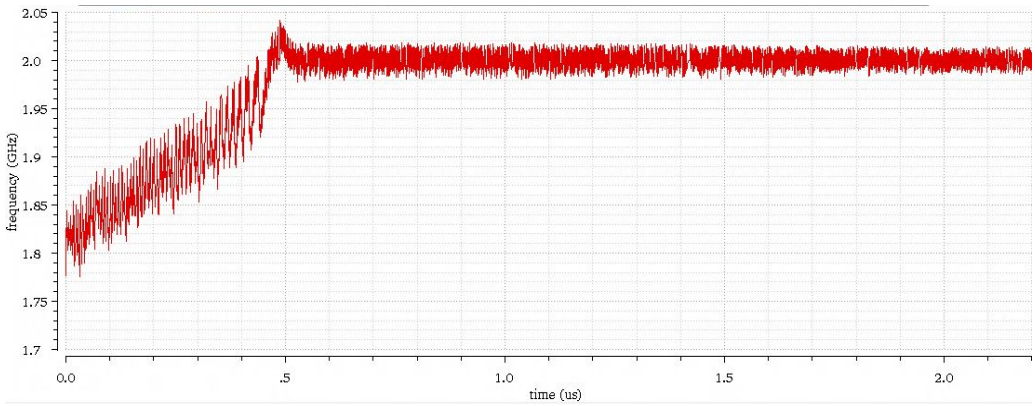

Figure 8.8: VCO frequency as a function of time showing lock acquisition

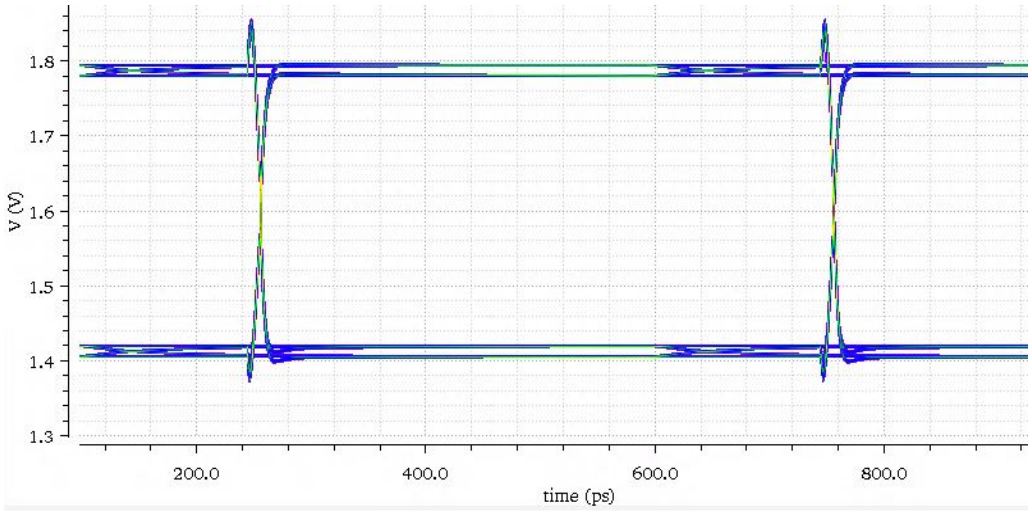

Figure 8.9: Eye diagram at the output of the transmitter

We also present the eye diagrams from the transmitter, channel and the CDR in Figures 8.9, 8.10 and 8.11 respectively. The eye diagram of the transmitter is very similar to the single-ended case although one can see very small mismatches in the differential impedance of the channel. This is well within the accepted limits and is accounted for in the design.

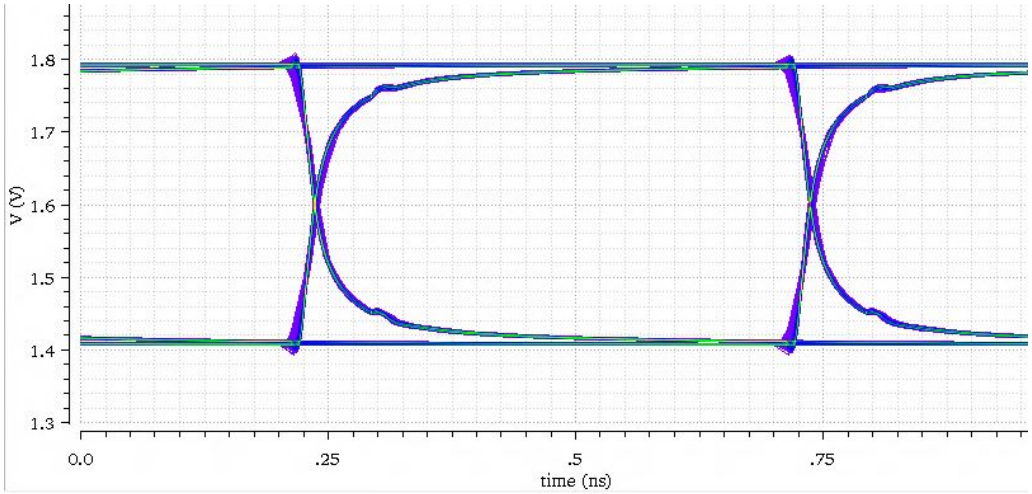

Figure 8.10: Eye diagram at the output of the channel

The eye diagram of the channel shows the signal degradation as previously explained. This leads to increased jitter and noise levels.

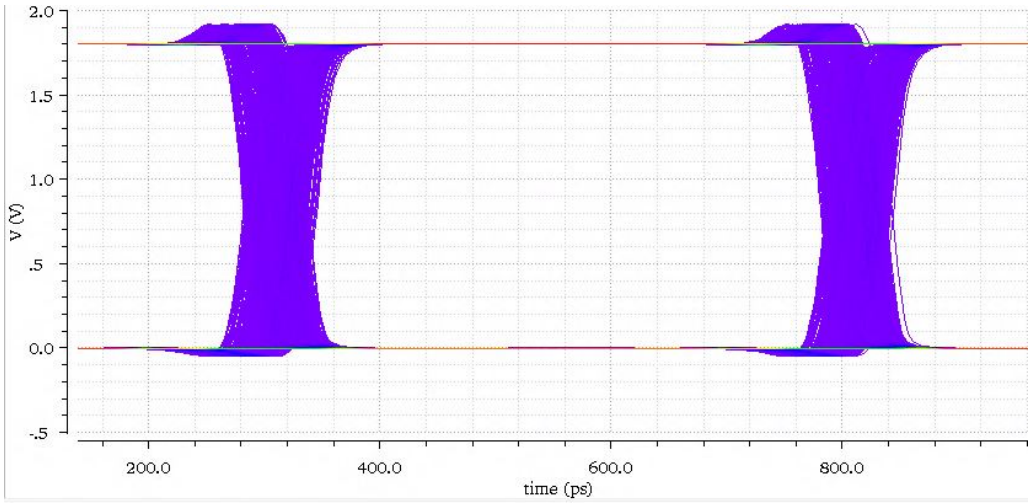

Figure 8.11: Eye diagram at the output of the CDR

Looking at the eye diagram at the output of the CDR (Figure 8.11), we notice that the data has been recovered correctly and we have excellent noise margin. However, we see a significant increase in the jitter of the retimed data which is roughly about 55 ps. This can be attributed to the fact that the CDR is driven by a superbuffer, which is essentially two copies of an inverter chain, and therefore the complementary signals are not exactly synchronized.

This effect is compounded by the ripples in the control voltage, which amplify the jitter in the circuit. However, the jitter is still close to 10% of the bit period and is tolerable in some applications.

Table 8.2: Summary of eye diagram characteristics for the complementary logic design

EYE	HEIGHT (mV)	WIDTH (ps)	NOISE (mV)	JITTER (ps)
Transmitter	360	499.2	16	0.58
Channel	362	493.1	12	6.5
CDR	1800	439	1	55.2

Table 8.2 summarizes the characteristics of the previously shown eye diagrams and provides values for jitter, noise margin, sampling window, etc. We can also examine the offset from the ideal sampling point of the generated clock as shown in Figure 8.12. We see that the clock samples the data about 50 ps to the left of the center of the eye.

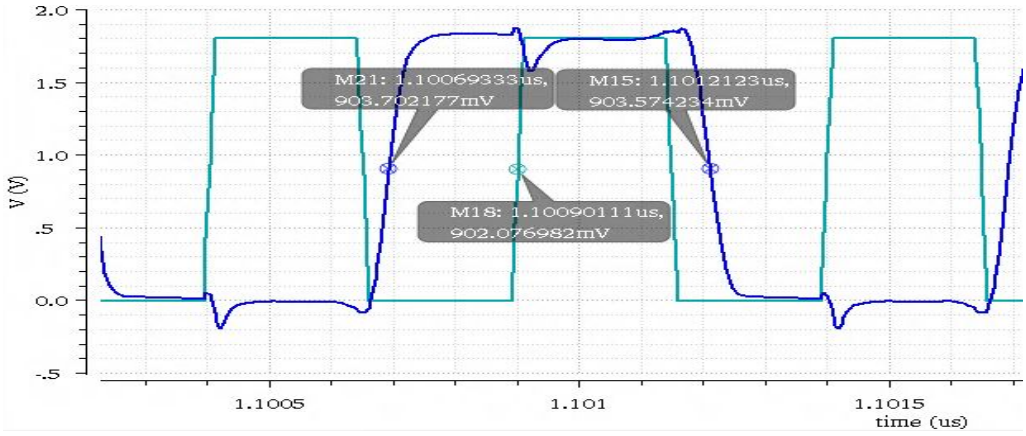

Figure 8.12: Waveform showing the phase offset in sampling the data by the CDR

As before, we can estimate power consumption by calculating the average current drawn from the power supplies. The total power consumption of the SERDES system is about 36 mW and that of the CDR alone is about 4 mW.

8.3 Current mode logic CDR

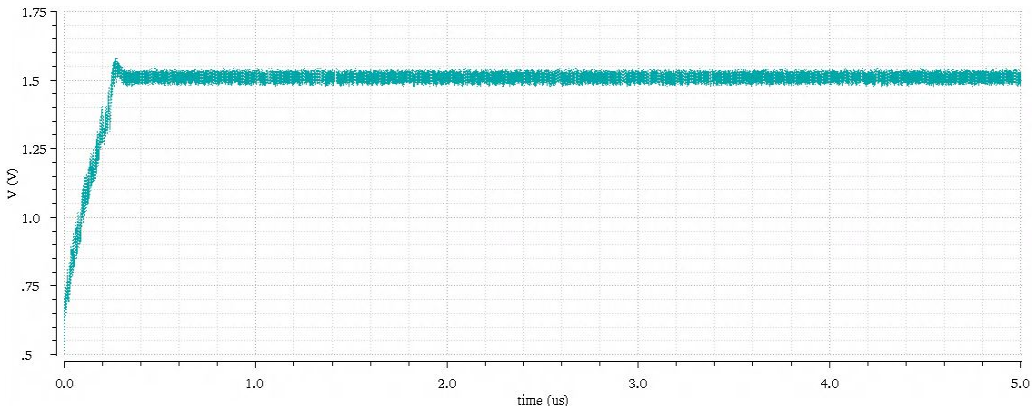

Figure 8.13: Locking behavior of the control voltage

We now look at the performance of the CML CDR. We first examine the locking behavior of the control voltage as shown in Figure 8.13. Similar to the single-ended case, the control voltage quickly settles to its final value within 300 ns. This is also confirmed by looking at the instantaneous frequency of the generated clock as shown in Figure 8.14 and the results are consistent.

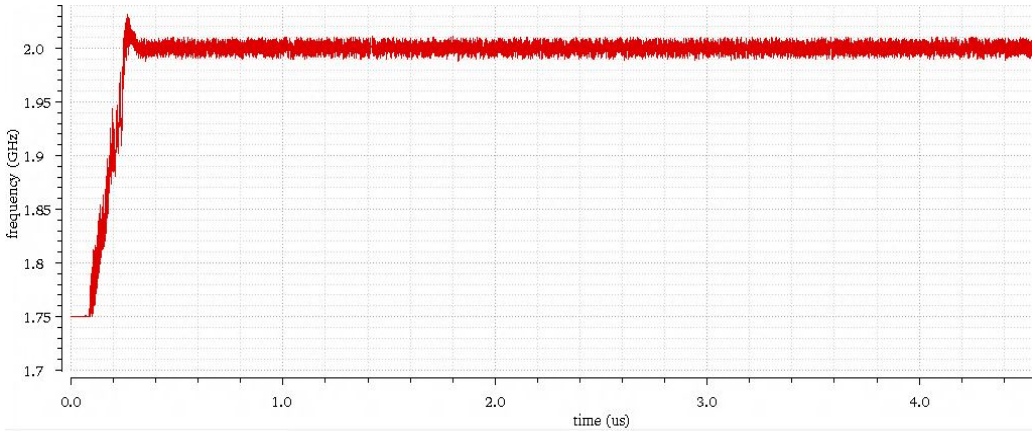

Figure 8.14: VCO frequency as a function of time showing lock acquisition

We also present the eye diagrams from the transmitter, channel and the CDR in Figures 8.15, 8.16 and 8.17. The eye diagram of the transmitter is very similar to the complementary logic case since the same physical channel is used with the same transmitter.

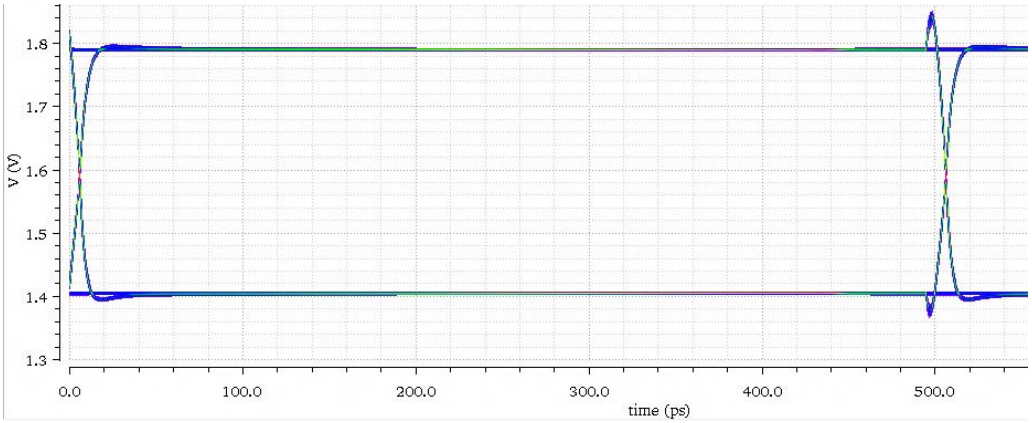

Figure 8.15: Eye diagram at the output of the transmitter

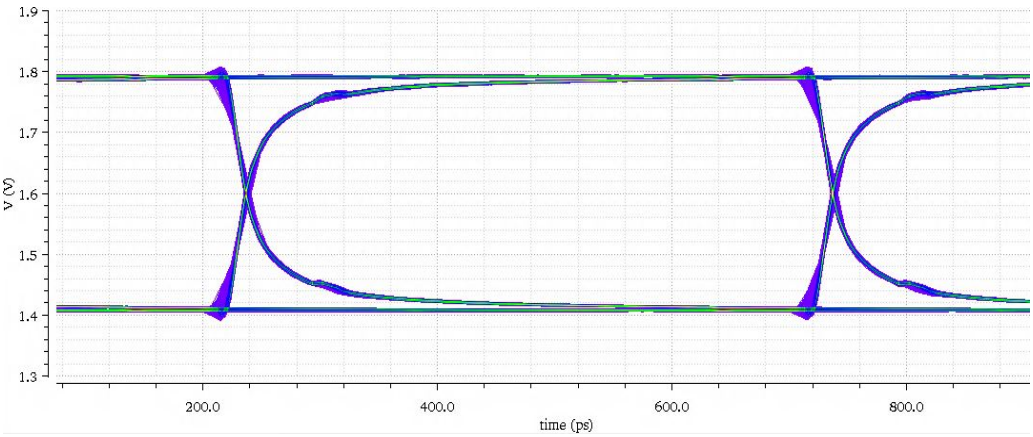

Figure 8.16: Eye diagram at the output of the channel

The eye diagram of the channel shows the signal degradation as previously explained. This leads to increased jitter and noise levels.

Looking at the eye diagram at the output of the CDR (Figure 8.17), we notice that the data has been recovered correctly and we have excellent noise margin. The jitter has been slightly amplified by the CDR and is about 25 ps, which is well within acceptable limits. The eye is perfectly symmetric, which augurs well for the integrity of the signal as well as for subsequent circuitry.

Table 8.3 summarizes the characteristics of the previously shown eye diagrams and provides values for jitter, noise margin, sampling window, etc. We can also examine the offset from the ideal sampling point of the generated clock as shown in Figure 8.18. We see that the clock samples the data about

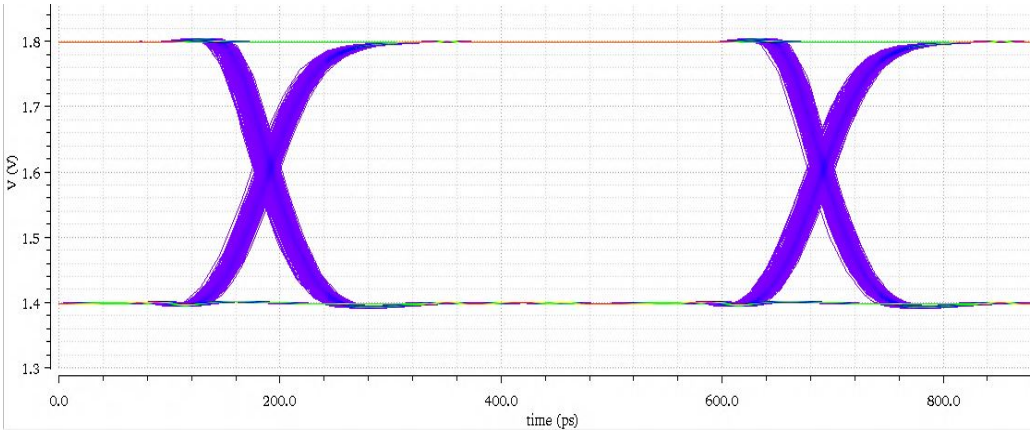

Figure 8.17: Eye diagram at the output of the CDR

Table 8.3: Summary of eye diagram characteristics for the CML design

EYE	HEIGHT (mV)	WIDTH (ps)	NOISE (mV)	JITTER (ps)
Transmitter	360	499.2	16	0.58
Channel	362	493.1	12	6.2
CDR	400	475	1	25

100 ps to the left of the center of the eye.

As before, we can estimate power consumption by calculating the average current drawn from the power supplies. The total power consumption of the SERDES system is about 36 mW and that of the CDR alone is about 6.5 mW.

8.4 Comparison of CDR circuit architectures

We now have sufficient information to compare and evaluate the three different architectures designed in this thesis. Table 8.4 shows a summary of the comparison.

The lock-in time is the time taken by each CDR to achieve frequency and phase lock when fed with identical PRBS sequences of length $2^{32} - 1$. It is important to note that this depends on the initial voltage on the loop

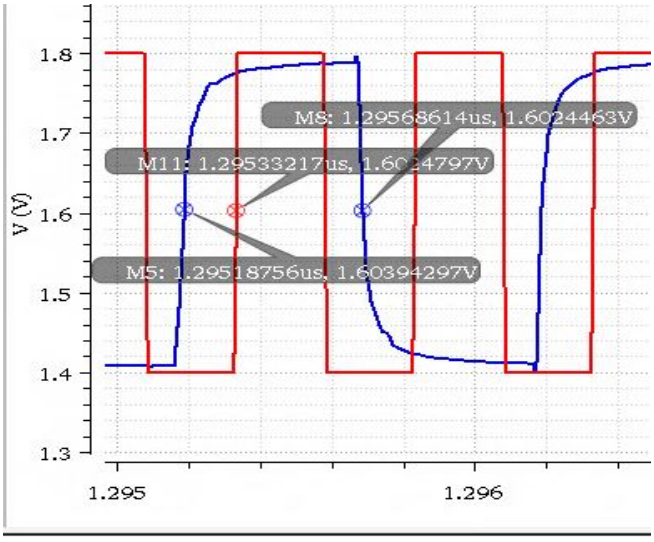

Figure 8.18: Waveform showing the phase offset in sampling the data by the CDR

capacitor and also the final value to which the control voltage settles. Simulators such as Cadence Spectre often cannot accurately predict the initial state of the system and the final state is highly dependent on the VCO design. Thus, this metric can often be misleading. In order to overcome this problem, we introduce a new metric known as lock-in slope - defined as the slope of the straight line that linearly models the rise or fall of the control voltage. Clearly, we notice that the CML architecture settles faster to the required value and can therefore recover data the fastest. It also helps the architecture to quickly establish relock if for some reason the lock is lost.

An important consideration in the present age of mobile devices is power consumption. The transmit powers are almost identical across the three architectures because in all cases the transmitter is identical. The differential channels consume slightly more power since we now have to send data on two lines. We can also see that the single-ended design consumes the least system power whereas the other two architectures consume slightly higher power. When it comes to the CDR, the CML architecture consumes the greatest power since we always have current flowing between VDD and GND, leading to static power consumption. However, this is reasonably compensated by the lack of receiver amplifier in the CML architecture.

We also notice that the CML architecture performs best in terms of jitter

Table 8.4: Comparison of the three circuit architectures designed in this thesis

METRIC	SINGLE-ENDED	COMPLEMENTARY	CML
Lock-in time	200 ns	500 ns	300 ns
Lock-in slope	2.94 MV/s	0.73 MV/s	3.41 MV/s
Transmit power	28.62 mW	28.81 mW	28.81 mW
RX Amp power	1.79 mW	2.043 mW	0
CDR power	3.46 mW	4.07 mW	6.69 mW
System power	33.97 mW	35.76 mW	35.51 mW
Jitter	27 ps	55 ps	25 ps
Sampling window	437ps	439 ps	475 ps
Sampling offset	80 ps	50 ps	100 ps
Number of consecutive 0s/1s	unrestricted	unrestricted	100
Channel degradation	24 mV	12 mV	12 mV
Total transistor width (CDR)	208.42 µm	299.12 µm	126.42 µm
Total transistor width (RX)	337.48 µm	428.18 µm	126.42 µm

and width of the sampling window. The single-ended architecture has similar jitter performance but suffers from asymmetry of rising and falling pulses leading to a smaller sampling window. The CML architecture is completely symmetric and overcomes this problem. The complementary architecture has poor jitter performance and also suffers from a smaller sampling window. However, this architecture exhibits the smallest sampling offset, i.e., the offset from the center of the eye is only 50 ps. The other two architectures have reasonable sampling offsets, and this can be adjusted by designing the VCO to lock closer to the value of control voltage when the UP and DOWN currents are symmetric.

Another important performance metric of the CDR is the number of consecutive 0's/1's it can tolerate without losing lock. Since the single-ended and complementary architectures use logic signals with full CMOS voltage swing, the switches for the UP and DOWN currents of the charge pump are completely turned off when receiving consecutive 0s/1s. Thus, the only way to leak charge from the capacitor is through leakage current which is of the order of 10^{-9} A. Hence, it takes an extremely long time for the voltage on the capacitor to decay, and for practical purposes it can be assumed that there is no siginificant limit on the number of consecutive 0s/1s. In case of the CML architecture, the voltage level for logic low is 1.4 V, which is sufficient to

allow significant current to pass through the transistors in the charge pump, thereby leaking or pumping charge from/to the loop capacitor. Hence, the control voltage will gradually change with time and deviate from the required value to produce a 2 GHz clock. For this specific design, it was observed that we could receive up to a 100 consecutive 0s or 1s without losing frequency and phase lock with the incoming data stream. This is sufficient for most practical applications, especially with the use of encoding to increase the amount of data transitions

It is also interesting to observe that the single-ended channel causes more (2X) degradation of the transmitted data than the differential channels. This is because differential channels allow us to cancel out the common mode noise, whereas in the case of single-ended channels the signal is referred to ground and any noise present will not cancel out. This becomes increasingly important as we go to higher data rates and has mandated the use of low voltage differential signaling (LVDS) as the standard for IO interfaces.

The last comparison metric is related to the area of the design which will directly impact cost. Since the designs were simulated only at the schematic level, it is not possible to accurately obtain the actual silicon area required. This can be obtained only by performing the layout design of the transistors and performing routing, etc. However, to obtain a rough idea, we can look at the total width of the transistors used in each of the designs. All transistors have a length equal to the minimum feature size, i.e., 180 nm. It is clear that the CML architecture is superior both in terms of the CDR area as well as the overall receiver area. This can be attributed to the fact that CML circuits have transistors only in the pull-down network, which results in a significant reduction in area as observed in the case of memory circuits using pseudo-NMOS technology [31]. Secondly, the CML architecture does not require the receiver amplifier and superbuffer stages, resulting in area as well as power savings.

It is critical to note that the area numbers reported in Figure 21 do not consider the area required to implement the resistors in CML circuits. Implementing polysilicon resistors is often very expensive; therefore, a PMOS load which is biased in the resistive region is used. This provides almost identical performance with a large reduction in area. Secondly, the area required to implement various current sources using current mirror circuits is also not considered. These two effects will reasonably increase the area required by

the CML architecture. It is also worth mentioning that the receiver amplifier blocks in the single-ended and complementary logic architectures also make use of resistors and current sources which have not been included in the reported area numbers.

Overall, we see that the CML architecture provides superior performance while providing significant reduction in area and consuming only slightly more power. Thus, for performance critical high speed serial links, current mode logic should be the circuit architecture of choice.

CHAPTER 9

DISCUSSION

In this thesis we have designed and evaluated three different CDR circuit architectures for 2 Gb/s operation using 180 nm CMOS technology. First, a strong mathematical framework to analyze the CDR is presented and is used to derive the optimal loop parameters. Next, a behavioral model of the CDR is implemented using Verilog-AMS to rapidly prototype designs and arrive at the ideal loop configuration. Finally, the behavioral model is implemented at the transistor level using three different circuit architectures and the results are compared. In this final chapter, we conclude with a discussion of CML architectures and why they are superior and also highlight some of the possible future work along these lines.

9.1 CML vs. single-ended

Based on the results presented in Chapter 8, the front runners in terms of CDR circuit architectures were the CML and single-ended design. Both architectures have almost identical jitter performance although the single-ended case has asymmetric transitions resulting in a smaller sampling window and a loss in the integrity of the signal. However, the single-ended case utilizes CMOS circuits which have only dynamic power consumption and no static power consumption. Thus, it is very power-efficient and this can be confirmed by looking at the CDR power alone in which case the single-ended case is 50% more power-efficient. However, at the system level it is only 4% more efficient due to the large power consumption of the receiver amplifier and the superbuffer blocks. Also, the single-ended architecture requires more area than the CML architecture. Thus, except in the case of power critical systems, the single-ended architecture is not preferable.

Another important reason for choosing the CML architecture is to combat

the increase in data rates. There is an insatiable demand for faster data access which necessitate an increase in the data rate of serial links. Thus, in performance-critical systems, power takes a back seat and it is vital to increase the data rate as much as possible. CML can prove to be extremely superior in this regard since it uses fewer transistors and therefore has a smaller load capacitance, which leads to faster switching. Also, the voltage swing in CML is much smaller than the conventional CMOS case which operates with full CMOS voltage levels. These two factors allow CML circuits to be a significantly faster than their conventional CMOS counterparts. Table 9.1 shows a comparison of unloaded delays of various fundamental digital logic elements in CML as well as conventional CMOS. Each of the elements is sized to have the drive strength of a minimum size inverter in the technology. A TSPC flipflop is used for conventional CMOS whereas a simple master-slave flipflop is used for the CML family. The test waveforms have a rise/fall time of 50 ps and full CMOS voltage levels are used for the conventional CMOS family whereas a voltage swing of 400 mV is used for the CML family. Thus, even though the rise/fall times are identical, the slew rate is much smaller in the case of the CML family, which typically leads to larger delays. From the comparison, it is clear that the CML family outclasses the conventional CMOS family in terms of switching speed. The speed advantage becomes more prominent as we scale down the device length and move to advanced process nodes.

Table 9.1: Comparison of switching speed between CML and conventional CMOS families

Digital Logic Element	Conventional CMOS delay (ps)	CML delay (ps)
Inverter	21.95	8.76
Rising edge-triggered Flipflop	87	62
MUX	85	43

The only potential drawback of the CML architecture is the power consumption. Although CML has a large static power consumption, the dynamic power consumption is much smaller than conventional CMOS due to

the smaller voltage swing. As the frequency of operation increases, dynamic power increases linearly, and this can result in the dynamic power consumption of conventional CMOS circuits being comparable to static power consumption of CML circuits, assuming conventional CMOS circuits are able to operate at such frequencies. Regardless of the power consumption, CML allows us to exploit a power-performance tradeoff which was not possible in the case of conventional CMOS due to fundamental limitations of the technology. As long as we have performance-critical applications such as data centers and servers, there will always be a demand for faster data access, and CML meets that requirement.

It is possible to make the CML technology even faster by reducing the voltage swing. In addition to faster switching speeds, this also provides savings in dynamic power consumption since the capacitors are charged and discharged to a lesser extent. Also, the smaller swing would imply smaller bias current, which leads to less static power and also smaller transistors. Thus, it is possible to make the CML technology faster, more power-efficient and more area-efficient by reducing the voltage swing. However, this affects the noise immunity of the circuit and must therefore be approached with great caution.

9.2 Future work

The biggest takeaway from this thesis is that CML circuits are the way forward in designing superfast serial links. A very basic version of the CDR was implemented to verify the advantages offered by the CML architecture. Several enhancements can be carried out, such as:

- Add a second CDR loop to perform coarse frequency detection. This allows the CDR to work over multiple frequencies in an adaptive manner and can therefore serve multiple applications.

- Implement more complex phase detectors. The Hogge phase detector is a simple linear PD and can be made more efficient. For example, [32] discusses a phase detector using latches instead of flipflops. This allows for less delay in the phase detector as well as less power consumption.

- Implement a half-rate/quarter-rate phase detector. In this thesis, the CDR only sampled on the positive edge of the clock. We can have the CDR sample at both the positive and negative edges, thereby doubling the data rate without increasing the clock frequency. This requires modifications to the phase detector [33]. The concept can be extended and we can have the VCO produce in-phase and quadrature-phase clocks, while sampling at both the positive and negative edges, thereby providing a 4X increase in the data rate without increasing the clock frequency.

- Consider using a charge pump and VCO with differential outputs. This would greatly improve the phase noise and jitter performance of the CDR.

- A completely different approach is to use all-digital CDRs [16]. In this case, the incoming analog signal is first converted to a digital signal using ADCs and then operations such as phase detection are performed in the digital domain. Instead of a VCO, we have a digitally controlled oscillator that produces different frequencies based on the control word. Thus, we eliminate the need for a charge pump and the loop filter. This has tremendous savings in terms of power since the charge pump often has a large current in the order of 1 mA. It also has area savings since the loop capacitor is in the order of $10^{-12}F$ and often requires a large area to implement. However, new challenges such as quantization noise arise and need to be tackled efficiently to provide performance comparable to that of analog CDRs. Another advantage of all-digital CDRs is that they are fully synthesizable and can therefore be highly automated in terms of the design process using hardware description languages (HDLs) and synthesis tools.

REFERENCES

[1] IDC, "IDC Predictions 2013: Competing on the 3rd Platform," Nov 2012. [Online]. Available: http://www.idc.com/research/Predictions13/downloadable/238044.pdf

[2] "Cisco Visual Networking Index: Forecast and Methodology, 2012-2017," White Paper, Cisco, May 2013. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

[3] M. Komorowski, "History of storage cost (update)," Mar. 2014. [Online]. Available: http://www.mkomo.com/cost-per-gigabyte-update

[4] J. Nielsen, "Nielsen's Law of Internet Bandwidth," Apr 1998. [Online]. Available: http://www.nngroup.com/articles/law-of-bandwidth/

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

[6] G. Moore, "Cramming More Components Onto Integrated Circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan 1998.

[7] R. Vetter, D. Du, and A. Klietz, "Network supercomputing," *Network, IEEE*, vol. 6, no. 3, pp. 38–44, May 1992.

[8] B. Razavi, *Monolithic Phase-Locked Loops and Clock Recovery Circuits*. Piscataway, NJ: IEEE Press, 1996.

[9] R. R. Dobkin, A. Morgenshtein, A. Kolodny, and R. Ginosar, "Parallel vs. serial on-chip communication," in *Proceedings of the 2008 International Workshop on System Level Interconnect Prediction*, ser. SLIP '08. New York, NY, USA: ACM, 2008. [Online]. Available: http://doi.acm.org/10.1145/1353610.1353620 pp. 43–50.

[10] J. C. Chen, "Multi-gigabit SERDES: The corner-stone of high speed serial interconnects," 2011. [Online]. Available: http://www.design-reuse.com/articles/10541/multi-gigabit-serdes-the-cornerstone-of-high-speed-serial-interconnects.html

[11] 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, Scotland, UK, 2009. [Online]. Available: theses.gla.ac.uk/707/1/2009assaadphd.pdf

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

[13] R. Kollipara et al., "Design, Modeling and Characterization of High-Speed Backplane Interconnects," presented at the Proc. DesignCon 2003, Santa Clara, CA.

[14] E. Alon, "High-speed electrical interface circuit design: Lecture 1," 2014. [Online]. Available: http://bwrcs.eecs.berkeley.edu/Classes/icdesign/ee290cs11/lectures/Lecture01Intro_2up.pdf

[15] M. Ida, N. Kato, and T. Takada, "A 4 Gbits/s GaAs 16:1 multiplexer/1:16 demultiplexer LSI chip," *Solid-State Circuits, IEEE Journal of*, vol. 24, no. 4, pp. 928–932, Aug 1989.

[16] P. Hanomulu, "Design Techniques for Clocking High Performance Signaling Systems," Ph.D. dissertation, Oregon State University, Corvallis, 2006. [Online]. Available: http://ir.library.oregonstate.edu/xmlui/bitstream/handle/1957/3530/20061016+pavan+dissertation.pdf?sequence=1

[17] S. Palermo, *CMOS Nanoelectronics Analog and RF VLSI Circuits*. New York City, NY: McGraw-Hill, 2011.

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

[19] A. Rezayee and K. Martin, "A 9-16Gb/s clock and data recovery circuit with three-state phase detector and dual-path loop architecture," in *Solid-State Circuits Conference, 2003. ESSCIRC '03. Proceedings of the 29th European*, Sept 2003, pp. 683–686.

[20] C. Hogge, "Carrier and Clock Recovery for 8 PSK Synchronous Demodulation," *Communications, IEEE Transactions on*, vol. 26, no. 5, pp. 528–533, May 1978.

[21] K. Kundert and O. Zinke, *Designer's Guide to Verilog-AMS*. Boston, MA: Kluwer-Academic Publishers, 2004.

[22] E. Hammerstad and O. Jensen, "Accurate models for microstrip computer-aided design," in *Microwave symposium Digest, 1980 IEEE MTT-S International*, May 1980, pp. 407–409.

[23] E. H. Fooks and R. A. Zakarevicius, *Microwave Engineering using Microstrip Circuits*. Upper Saddle River, NJ: Prentice-Hall, 1990.

[24] S.-C. Huang, M. Ismail, and S. Zarabadi, "A wide range differential difference amplifier: a basic block for analog signal processing in MOS technology," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 40, no. 5, pp. 289–301, May 1993.

[25] F. Silveira, D. Flandre, and P. Jespers, "A g_m/I_D based methodology for the design of CMOS analog circuits and its application to the synthesis of a silicon-on-insulator micropower OTA," *Solid-State Circuits, IEEE Journal of*, vol. 31, no. 9, pp. 1314–1319, Sep 1996.

[26] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits, 2/e*. Upper Saddle River, NJ: Prentice-Hall, 2003.

[27] X. Lin, Y. Wang, S. Nazarian, and M. Pedram, "An improved logical effort model and framework applied to optimal sizing of circuits operating in multiple supply voltage regimes," in *Quality Electronic Design (ISQED), 2014 15th International Symposium on*, March 2014, pp. 249–256.

[28] "Differential impedance calculator: Microstrip." [Online]. Available: http://referencedesigner.com/tutorials/si/si_10.php

[29] B. Nikolic and V. Oklobdzija, "Design and optimization of sense-amplifier-based flip-flops," in *Solid-State Circuits Conference, 1999. ESSCIRC '99. Proceedings of the 25th European*, Sept 1999, pp. 410–413.

[30] A. P. Martinez, "Design of MOS Current-Mode Logic Standard Cells," M.S. thesis, École polytechnique fédérale de Lausanne, Lausanne, Switzerland, 2007.

[31] V. Beiu, J. Quintana, and M. Avedillo, "VLSI implementations of threshold logic-a comprehensive survey," *Neural Networks, IEEE Transactions on*, vol. 14, no. 5, pp. 1217–1243, Sept 2003.

[32] J. Savoj and B. Razavi, "A 10-Gb/s CMOS clock and data recovery circuit with a half-rate linear phase detector," *Solid-State Circuits, IEEE Journal of*, vol. 36, no. 5, pp. 761–768, May 2001.

[33] J. Lee and B. Razavi, "A 40-Gb/s clock and data recovery circuit in 0.18-μm CMOS technology," *Solid-State Circuits, IEEE Journal of*, vol. 38, no. 12, pp. 2181–2190, Dec 2003.