A 10Gb/s PI-CDR design

Msc. thesis

by

Achraf Gardouh

TU Delft

Abstract

In this thesis the design and analysis of a dual-loop phase interpolator(PI) clock and data recovery(CDR) with a Delay locked loop (DLL) as a reference loop will be discussed. In the first chapter CDR basics will be explained . The second Chapter will discuss the design and optimization of the DLL circuit. Finally all this will be brought together in the Dual loop CDR to perform its operation

Acknowledgements

I am immensely grateful for the opportunity to work on this thesis project. First and foremost, I extend my appreciation to Dr. Morteza Alavi for his support and invaluable guidance throughout the duration of this project.

I would also like to express my gratitude to Dr. Masoud Babaie and Dr. Francesco Fioranelli for their involvement as committee members and for providing their assessment of the project.

A special thanks goes to Anil Kumaran, a PhD student, whose assistance during my Layout endeavor was truly invaluable.

I am indebted to PhD student Jun Feng for his assistance and guidance, which played a pivotal role in jumpstarting this project.

Additionally, I would like to acknowledge all the professors in the microelectronics department from whom I have gained immense knowledge and skills.

Finally, I want to extend my deepest appreciation to my parents and friends for their unwavering support and encouragement, always pushing me to strive for excellence.

Contents

1
Introduction

Wireline communication has become an essential part of today's world. It is often used where high-speed links are needed, such as in big data centers and high-speed internet connections, in which many wireline links are exploited in parallel to reach extremely high-speed throughput. For example, the link presented in [4] can go up to 136 Gb/s.

Clock and data recovery (CDR) is an essential circuit of the wireline transceivers. This block extracts the clock from the data and then utilizes it to sample it to recover the original data. This thesis project uncovers a 10Gb/s CDR circuit with a 2.5GHz accompanying clock signal. The proposed CDR will be employed in the receiver block of a high-speed serial link enabling real-time operation of high-speed bits-in RF-out transmitter.

1.1. SERDES

SerDes stands for Serializer/deserializer. This block facilitates high-speed serial data communication. Many parallel data lines exist as inputs, which will be combined and represented with only one signal at its output. There are several reasons why transmitting data in serial may be advantageous over its parallel counterpart:

- Efficiency: Serial data transmission is often more efficient because it requires fewer wires or channels to transmit the same data. This reason is particularly valid for high-speed data transmission, where the cost of additional wires or channels can be prohibitive.

- Longer transmission distances: Serial transmission can often transmit data over longer distances than parallel configuration, as the signals can be better protected from noise and interference.

- Clock synchronization: Serial transmission is often exploited in systems where synchronization between the transmitter and receiver is difficult to achieve. In serial transmission, the clock can be embedded in the data stream, eliminating the need for a separate clock signal.

- Scalability: Serial transmission is more scalable than its parallel companion because the number of channels or wires required for serial transmission does not increase as rapidly as in the case of parallel configuration. This arrangement enables high-speed data transmission.

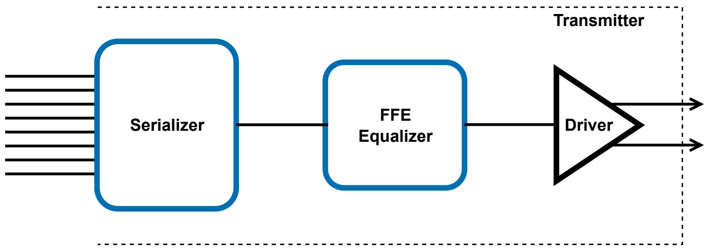

Figure 1.1: General SERDES TX

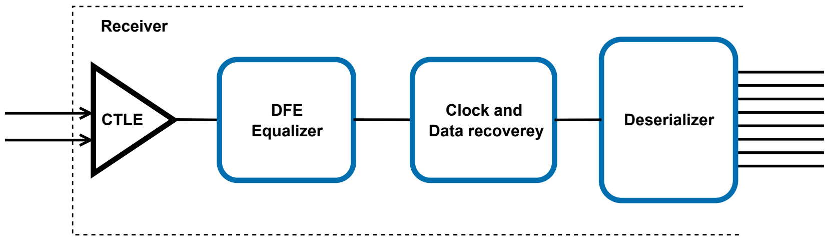

Figure 1.2: General SERDES RX

The originally sent data send and the received data must ideally be the same. However, unfortunately, this is not the case in SerDes communication, whose channel always introduces unwanted effects such as intersymbol interference(ISI), noise, and signal attenuation. To scale the SerDes systems to higher data rates, those undesired signals and effects must be addressed appropriately.

The wireline transmitter (TX) and receiver (RX) operation is as follows. As discussed earlier, the data needs to be serialized to fulfill the communication through fewer channels by exploiting a multiplexer. The multiplexer takes X inputs and turns them into a single output. This high-speed serialized output is transmitted through the communication channel and eventually arrives at the RX, where it will be deserialized again using a demultiplexer. In this context, the CDR block in the RX recovers the original data and its clock.

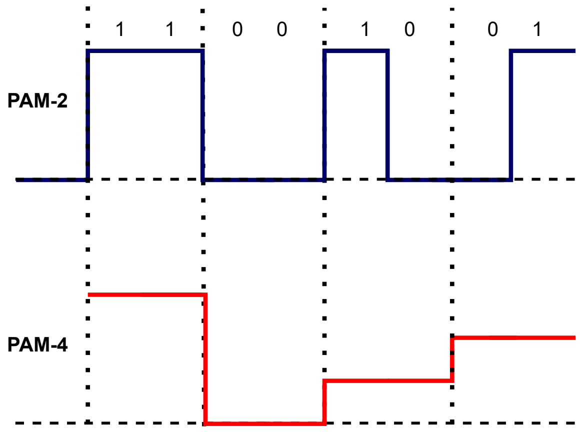

Figure 1.3: Example of PAM and PAM4 signaling

Currently, high-speed serial links utilize predominately two modulation schemes. These are pulse amplitude modulation (PAM)-2, also called NRZ, and PAM-4. The former is a signal with only two different amplitudes representing a single bit. This signal occupies twice its bit rate with a relaxed

SNR requirement. This condition differs in the PAM-4 modulation scheme since it requires a higher SNR while occupying half the bandwidth of the PAM-2 counterparts. In this context, PAM-4 transmits 2 bits with four amplitude levels in 1 symbol (figure 1.3). This feature enforces using pre/post-cursor ISI equalization techniques in the TX/RX chain to mitigate the influence of ISI and channel losses. Pre-cursor ISI is the ISI that influences past information sent, while post-cursor ISI addresses future data. In figures 1.1 and 1.2, three types of equalization can be seen, which are commonly used in SERDES wireline transceivers, linear equalization, Feedforward equalization(FFE) and Decision feedback equalization (DFE).
Equalization is a technique used in receivers and transmitters to lower the influence of ISI and channel losses. There is two different types of ISI, pre and post cursor. Pre cursor ISI is the ISI that influences past information send while post cursor ISI future information. In figure 1.1 and 1.2 3 types of equalization can be seen which are commonly used in SERDES wireline transceivers, linear equalization, Feed forward equalization(FFE) and Decision feedback equalization (DFE).

Linear equalization is used in most front ends of wire-line RXs. It is a circuit with a high-pass characteristic. Linear equalization is exploited to decrease ISI by employing the inverse function of the channel. An equalizer that reduces the ISI to zero is called a zero-forcing linear equalizer. The transfer of a linear equalizer and the transfer of the channel must look like a transfer function with a Nyquist characteristic [3].
In this regard, most non-idealities introduced by the channel and the transmitter are included. Except for one last issue that the CDR block will resolve. To recover the high-speed data successfully, the sampling must happen in the middle of the bit period (data eye), preventing data/clock jitters and data's fall/rise times from affecting CDR operation. Otherwise, the sampling operation close to the edges of the data most likely gives rise to an error. The CDR operation will be the main focus of the thesis.
Linear equalization is present in most front ends of wire-line receivers. It is a circuit with a high pass characteristic. Linear equalization should be there to decrease ISI and do the inverse function of the channel. A equalizer that would reduce the ISI to zero is called a zero forcing linear equalizer. The transfer of a linear equalizer together with the transfer of the channel together would have to look like transfer with a nyquist characteristic [3].

With thebefore-mentionedd circuits most of the non-idealities introduced by the channel and the transmitter can be dealt with. Except one last issue which will be resolved by the clock and data recovery(CDR). This is that the sampling has to happen in the middle of the bit period, due to the jitter, fall and rise time of the data. These non-idealities make it such that the timing of sampling matters since close to the edges of the data an error most likely will occur. A CDR circuit will take the incoming data, find the corresponding clock and recover the data. This will be the main focus of the thesis.

1.2. Thesis Objectives and Specifications

A 10Gb/s wireline TX was realized in 2020 [6][5], whose outputs were 10Gb/s serial data together with a 10GHz reference clock. The TX generates less than 1.5ps random jitter, and its SNR is 26.3dB. This thesis project aims for the companion wireline RX part operating at 10Gb/s. The CDR circuitry is the most critical block of the entire RX chain, whose inputs are 10Gb/s data together with a 2.5GHz reference clock, and its outputs are the deserialized original data driving the presumed inputs of a wireless digital transmitter (DTX). In this regard, the DTX can be evaluated using a real-time long-run OFDM signal. The proposed CDR comprises a delay-locked loop (DLL), a phase interpolator, a 2-bit time-to-digital converter (TDC), a phase detector, and a digital control loop. This thesis aims to recover 10Gb/s data using our proposed CDR circuitry in a 40nm CMOS technology with a 1V supply voltage. The aimed power consumption is 20mW with less than 2ps random jitter. The thesis specifications are highlighted in Table 1.1.

Table 1.1: System specification

	Target
Technology	40 nm CMOS
Supply	1 V
Modulation	PAM-2
Speed	10Gb/s
Power consumption	<20mW (<2pJ/b)

1.3. Thesis overview

This thesis project proposed a 10Gb/s CDR circuitry. In Chapter 2, CDR's state-of-the-art architectures and their related performances will be reviewed. In this context, the focus will be on phase detection and various jitter types of CDR circuitry. Chapter 3 explains the design procedure of the proposed delay-locked loop (DLL) since the proposed 10 Gb/s CDR operates using a 2.5GHz reference clock. The proposed DLL generates multiple clocks with different phases that drive the following phase interpolator. In Chapter 4, the last part of the CDR block will be completed comprising a phase detector and digital control loop. The goal is that the data detection occurs in the middle of the eye to warrant successful data and clock recovery. The argumentation behind certain decisions, like the number of phases generated by the phase interpolator, will be clarified. Chapter 5 summarizes the results, and some suggestions will be presented.

2
Clock and Data Recovery

As mentioned, CDR is the most crucial building block of the entire wireline RX chain. The CDR fulfills two essential functions. It must recover both data and clock from a jittery distorted random input data, as shown in Figure 2.1. In this regard, the incoming data is retimed in the middle of its bit period employing a (reference full-rate) clock. Moreover, clock recovery can be performed in various fashions. The basic architectures of these CDR circuits will be discussed in this chapter, with some insight into the state-of-the-art CDR design.

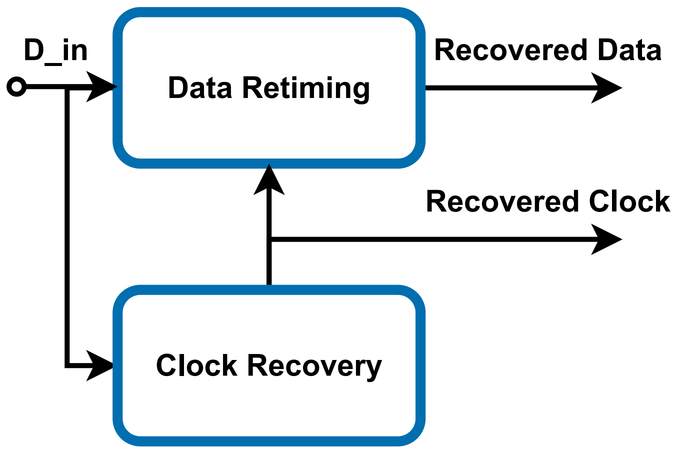

Figure 2.1: CDR basic functionality

2.1. Various CDR architectures

CDR can be categorized into two main architectures, namely:

- Oversampled CDR or phase interpretor-based CDR. It can be inferred as oscillator-less CDR architecture, i.e., CDR operating with a reference clock. In this architecture, the CDR comprises A DLL or a phase interpolator (PI).

- PLL-based CDR, also called reference-less CDR. In this context, the clock has not been forwarded from the wireline TX. Otherwise stated, the CDR does not know at which clock rate the incoming data must be retimed and recovered. Moreover, the NRZ data has no spectral lines at their (multiples) operating data rate due to frequency nulls occurring at the multiples of the NRZ data rate (the Sinc function nulls of random square-wave NRZ data). This feature indicates the CDR loop must first detect the frequency and operation. To achieve this, it generates a spectral line at the operational frequency (clock recovery) using an edge detector and then retimes the incoming data using the recovered synchronous clock. This classic PLL-based CDR will be discussed in the next section.

2.1.1. PLL-CDR

A reference-less CDR architecture employs a PLL to recover its clock. Therefore, the CDR's loop dynamics are similar to its PLL counterpart. Hence, the most significant advantage of PLL-based CDR systems is their ability to operate over a wide frequency range, leading to applications whose data rate requires a large dynamic range [10]. In this configuration, similar to a PLL system, the voltage-controlled oscillator (VCO) is controlled by a phase difference detected by the phase detector (PD), meaning the recovered synchronized clock is the output of the VCO.

The PLL-based CDR circuit is at its name describes a CDR circuit that operates very similar to a PLL. As in that, its loop dynamics are similar. The biggest advantage of a PLL-based CDR being is its ability to do the CDR operation over a wide range of frequencies. This makes the PLL-CDR a good alternative when for applications with dynamic data rates[10]. The PLL-CDR operates like a PLL where the voltage-controlled oscillator is controlled by the phase difference detected by the PD. This means until the clock is synchronized to the output of the VCO

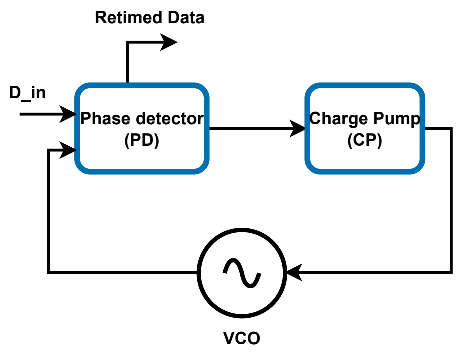

Figure 2.2: Basic PLL Based CDR

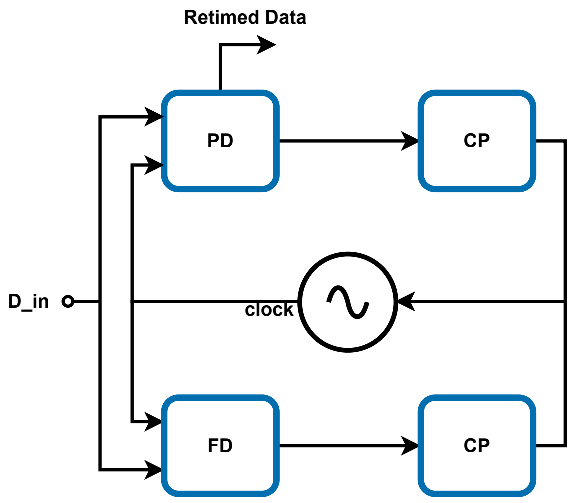

Figure 2.3: Dual loop PLL CDR with frequency Detection

Another alternative is a dual-loop CDR. Instead of having a PD loop in CDR as in a PLL, this architecture adds a second loop where a frequency detector (FD) controls the same VCO. For a referenceless CDR, this arrangement leads to superior robustness and improved stability [19]. This FD is not a typical quadri-correlator exploited for frequency detection. Instead, an edge detector is employed to recover the spectral line (frequency content) of the data required for the quadri-correlator [19] It is worth mentioning the NRZ data do not have a spectral line at the frequency of operation due to exhibiting frequency nulls at the integer multiples of the NRZ data rate.

An example of a state-of-the-art wireline receiver utilizing a PLL-based CDR can be found in the study referenced as [15]. The receiver demonstrates the capability to operate at 80 Gb/s with a PAM-4 modulation scheme. Interestingly, the VCO operates at 20 GHz, while the data is sampled at a quarter rate of 10 Gb/s. This is achieved by dividing the 20 GHz clock into two 10 GHz phases, which are further phase-interpolated into four different phases for sampling the incoming signal.

2.1.2. Delay locked loop CDR

Figure 2.4 exhibits a DLL-based CDR. In the architecture, a reference clock that is forwarded from a wireline TX is applied to a DLL loop whose output drives a PD. The other input of the PD is the incoming random data. The detected phase difference is applied to a charge pump (CP) whose output is the voltage-controlled delay line (VCDL) to adjust the phase of DLL clocks. Meanwhile, the data will be retimed and recovered by the DLL's recovered synchronous clock.

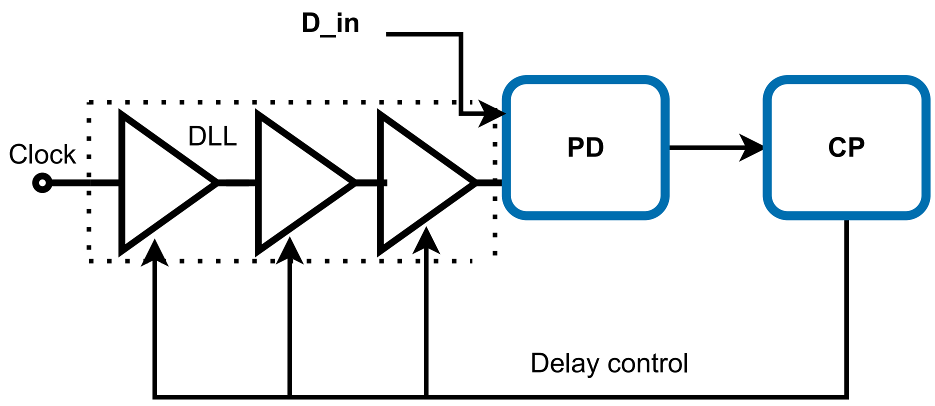

Figure 2.4: DLL CDR diagram

DLL-Based CDR architecture features many advantages. First, the CDR does not exhibit data-dependent jitter. Second, in applications involving many serial lines, using one reference clock for retiming is straightforward, simple, and efficient. In this way, the system does not require a VCO per serial lane that consumes more power and a large silicon area. Third, in contrast to the PLL-based CDR system employing a VCO, the DLL is a feedback system with only one pole, giving rise to stable loop operation.

2.1.3. Phase interpolator CDR

This block includes a reference clock, DLL, phase interpolator (PI), and phase detector (data recovery) loop, forming a dual-loop CDR architecture with multiple phases generated by the first DLL loop. In this context, depicted in Figure 2.5, the PI requires two clocks to generate in-between phases. One of its main features is that the loop where the reference is generated and the data recovery loop are separate, allowing independent operations of phase tracking in the data recovery loop and the phase generation in the reference loop. This aspect also indicates that only one reference loop is needed for multiple CDRs to operate on the same chip, as in the DLL-based CDR.[1]

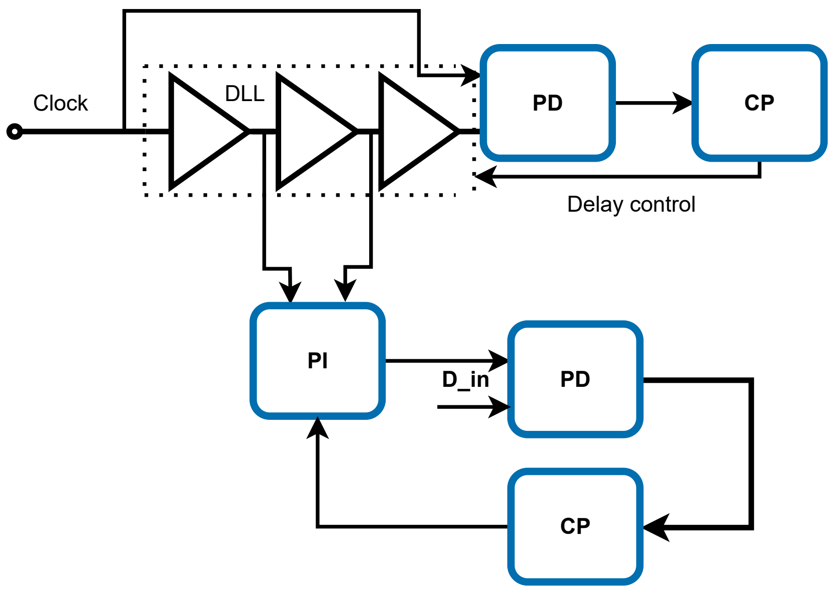

Figure 2.5: PI CDR diagram

Phase interpolator CDR's are widely employed in wireline receivers due to their efficient and fast operation. An intriguing study focuses on a wireline receiver operating at 56 Gb/s, as discussed in [23]. In this wireline receiver, The data rate has been able to be pushed this high by a lot of the concepts described earlier. By utilizing a PAM-4 modulation scheme, the PI-CDR operating at 1/8th of the baud-rate (3.5GHz) and the various equalization mentioned in the introduction the utilized speeds have been able to get pushed to 56 Gb/s.

2.2. Phase detection in CDR

In CDR architectures, there are two types of phase detectors, i.e., linear phase detectors and non-linear phase detectors, which will be reviewed in the following.

The Hogge phase detector (HPD), depicted in Figure 2.6, is the most straightforward CDR linear phase detector. It comprises two XOR gates and two DFFs. As stated, reference-less CDRs require spectral line detection using an edge detector. HPD enables simultaneous phase and edge detections, which is impossible using a stand-alone XOR phase detector due to false locking.

It is worth mentioning that using only a combination of an XOR and a DFF leads to a false locking due to generating the same DC average value for different phase errors and transition density. To address this issue, a reference pulse is generated by employing a set of XOR/DFF right after the first DFF (See Figure 2.6). Here, the output of the second XOR generates pulses with a fixed pulse width, i.e., one period of a full-rate clock, whenever data edges occur [18].

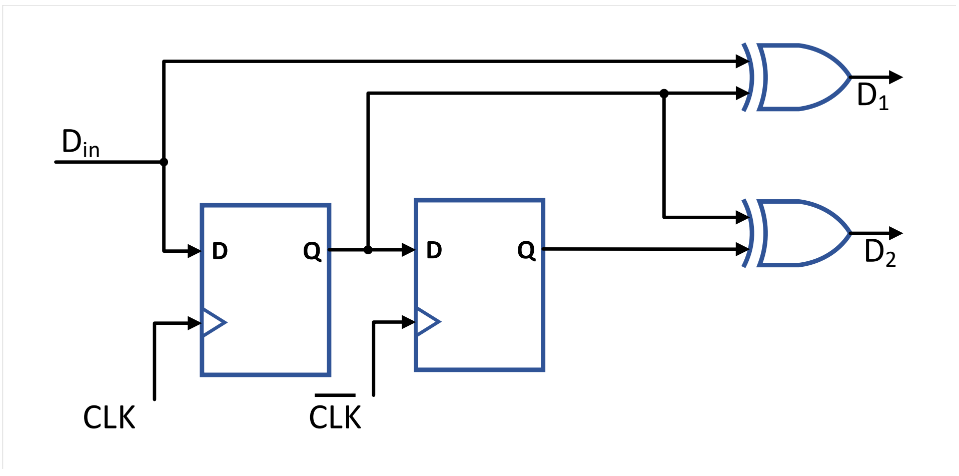

Figure 2.6: Hogge phase detector

However, the HPD exhibits several drawbacks. First, when the incoming random data experiences a long run with no data transition, the VCO control voltage lowers, resulting in false locking and errors in the retiming data. Moreover, due to DFF's clock-to-Q (CKQ) delay, the detector produces a pulse whose width is equal to CKQ, degrading the clock phase margin and increasing the CDR jitter. This issue can be mitigated by inserting the same delay in the reference (bottom) path. The advantage of this skew-free phase detector is that it produces zero average output at the locking condition, resulting in a minimal change in the VCO control voltage.
When using the HLPD there are multiple drawbacks. The most obvious one is what will happen if there is no data transitions in a while this will lower the VCO control voltage and result in errors in the coming data. When using a D-flipflop there is a clock to Q delay which will also influence the average output of the HLPD and thus result in errors. Designing around these issues is possible but can make the design very complex compared to the BBPD[18]. The HLPD does have the advantage of when the phase difference is close to zero that the average output is very low thus resulting in a smaller change in the VCO control voltage.

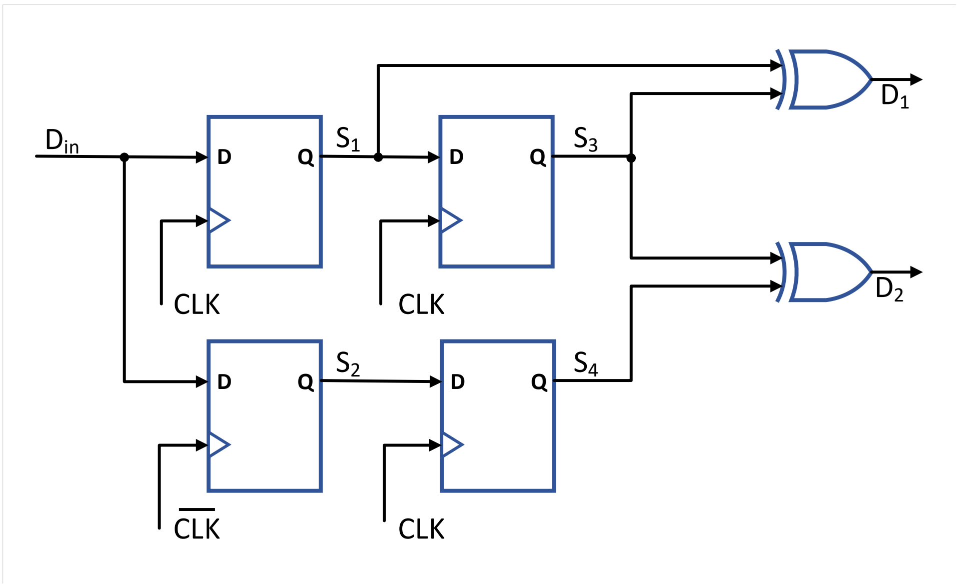

Figure 2.7: Alexander BBPD

Figures 2.8 and 2.9 show the different waveforms inside this detector in case of a late and early clock.

This detector has three different scenarios, which can all be observed in the figures.

- The clock is late thus resulting in D2 being high.
- The clock arrives early resulting in D1 also being high.
- No Data transition, none of the outputs become high.

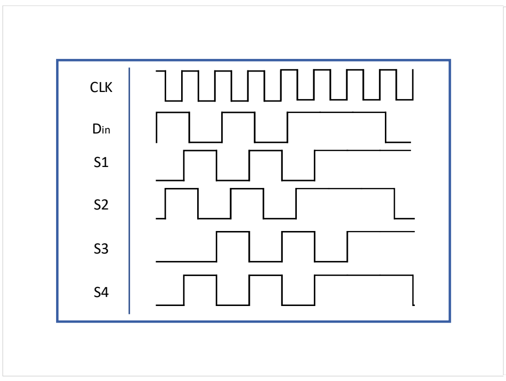

Figure 2.8: Alexander phase detector waveforms with late clock

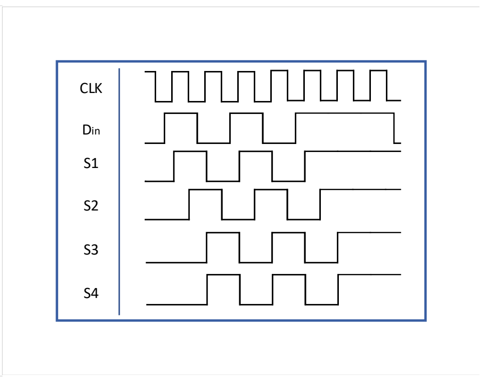

Figure 2.9: Alexander phase detector waveforms with early clock

The Alexander phase detector is much simpler to design than HPD. It alleviates the problem of having no data transitions because, in an Alexander phase detector, the control voltage does not change when there is no transition. The Alexander phase detector lets the VCO know if the clock is leading or lagging. So when the phase difference is close to zero, the change in VCO control voltage does not change but is the same as when the difference is significant, and the flipflops in the Alexander phase detector will be in the meta-stable region, which results in more significant random jitter due to the Phase Detector[8].

2.3. Jitter and noise considerations

Jitter and phase noise are two of the most significant challenges in CDR design. A clock jitter is a variation in the period of a clock signal caused by noise and other disturbances in the system. Phase noise is the variation in the phase of a clock signal caused by random fluctuations in the clock oscillator. They can degrade the performance of a CDR circuit and cause errors in the recovered data. The CDR Jitter generated by the clock comes from two primary sources.

- The jitter from the input data
- The jitter generated from the CDR circuit

The loop filter parameters primarily determine the jitter transfer from the input data jitter in a PLL CDR circuit. In this context, the loop resistance, capacitance, and charge-pump current determine the bandwidth of the jitter transfer, stability, and peaking transfer of the loop [2]. The jitter from within the CDR is mainly due to the ripple in control voltage and the VCO phase noise. When using a DLL-based CDR, there is also the input clock jitter, which is unrelated to the incoming data jitter. In this regard, the two main sources of jitter will be primarily:

- The jitter from the input clock
- The jitter generated from the CDR circuit

One of the main approaches to evaluate a CDR's jitter performance is the jitter tolerance, which shows the maximum allowable input jitter for the specified BER at different frequencies. For lower frequencies, this tolerance is higher, i.e., at lower frequencies, the jitter can be tracked by the CDR and thus result in better jitter tolerance. While at higher frequencies, the jitter transfer becomes lower, and so does the jitter tolerance because the jitter can not be tracked anymore. [2].

3 Delay locked loop Design

In this chapter, we delve into the design of the Delay-Locked Loop (DLL). We begin by providing an overview of the DLL's fundamental operation and dynamics. Subsequently, we delve into a detailed discussion on the design aspects of each component comprising the DLL. These components include the voltage-controlled delay element, the phase detector, and the charge pump. Finally, we present the results obtained from simulating the DLL to assess its performance and functionality.

3.1. Delay locked loop

DLL is a circuit similar to a Phase-Locked Loop (PLL) whose output phase is locked to its input phase, utilizing a feedback loop. PLLs employ a voltage-controlled oscillator (VCO) to enable phase locking. In a DLL, the delay of an inverter chain is varied until the input and output phases are equal, and thus the delay from input to output becomes one clock period.

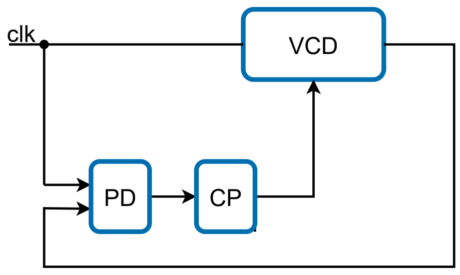

Figure 3.1: DLL of type-I

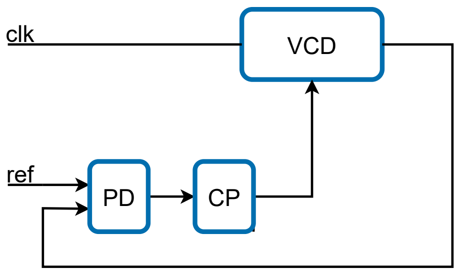

Figure 3.2: DLL of type-II

Figure 3.1 exhibits a block diagram of a DLL. In this DLL, an input reference clock from a PLL or a crystal drives the delay-cell elements. This clock goes through a voltage-controlled delay (VCD), after which a phase detector (PD) and a charge pump (CP) facilitate the phase locking between the input and output of the DLL chain.

There are two types of DLL, which are called type-I and type-II DLL. A generic schematic of the two can be seen in Figure 3.1 and 3.2. In a type-I DLL, the Phase Error Detection is done between the clock and the delayed clock, which makes the delay of the VCD equal to T_{clk}. In a type-II DLL, this error detection is done between the output of the VCD and another reference. In the case of a DLL CDR, this would be the input data, for example. In this project, the only DLL that will be looked at is the type-I DLL because the CDR designed will be a PI-CDR which will thus need a Type-1 DLL as a reference loop.

There are two types of DLL: type-I and type-II DLL. Figures 3.1 and 3.2 show a generic schematic of them. In a type-I DLL, the phase error detection is executed between the clock and the delayed clock, which makes the delay of the VCD equal to Tclk. In a type-II DLL, this error detection is performed between the output of the VCD and another reference which would be the input data in the case of a DLL-based CDR. In this project, the only DLL that will be locked is the type-I DLL because the CDR designed will be a PI-CDR, thus needing a Type-1 DLL as a reference loop.

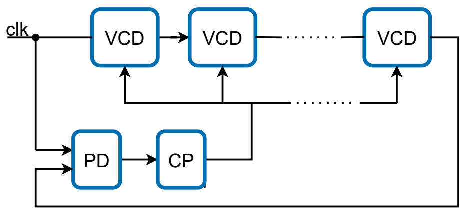

Figure 3.3: Multiple phases generation with a DLL

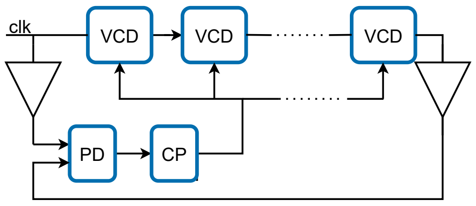

Figure 3.4: Multiple phases with equal load

To generate multiple phases, more VCD elements must be incorporated into the DLL. The number of phases is equal to the number of delay elements shown in Figure 3.3. The issue with this structure is that not all delay elements are equally loaded and, thus, are not exactly the same. To circumvent this, aside from using a buffer for all the phases, the input of the PD will also be buffered so that almost all the elements are equally loaded, as seen in Figure 3.4."

For the loop dynamics, since the type-II DLL has not been considered in this thesis, only the type-I DLL is investigated. The following linear phase domain model can be inferred from Figure 3.1 for a type-I DLL. The phase detector is modeled as a summation point in the phase domain representation since the difference between the input and output are measured. While the charge pump and its output capacitance are modeled as the charge-pump current divided by the capacitance. The delay line gain is modeled as KD. This gain is the gain from the control voltage to the corresponding "time gain." The summation point at the output of the model is thus where the phase of the input and the delay line are added, which creates the output phase.
From this model, it can be derived that the relationship between the input is the following:

$$Phase_{out} = Phase_{in} + (Phase_{in} - Phase_{out})\frac{I_{CP} \cdot K_D}{CS}$$

Thus, this relationship indicates that the output phase approaches the input phase.

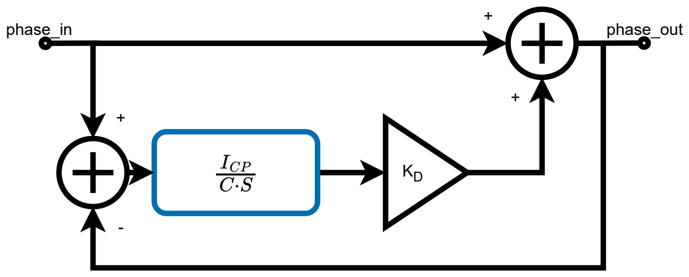

Figure 3.5: Phase domain model type-I DLL

3.2. Phase detector & Charge pump

The phase detector (PD) is a circuit that translates the phase difference of two signals into an output signal. The PD is an integral part of the design since having a phase detector with a dead zone will create an offset error in the DLL. There is also the speed of the incoming clock of the DLL to consider at 2.5 GHz. The PD used for his DLL can be seen in Figure 3.6.

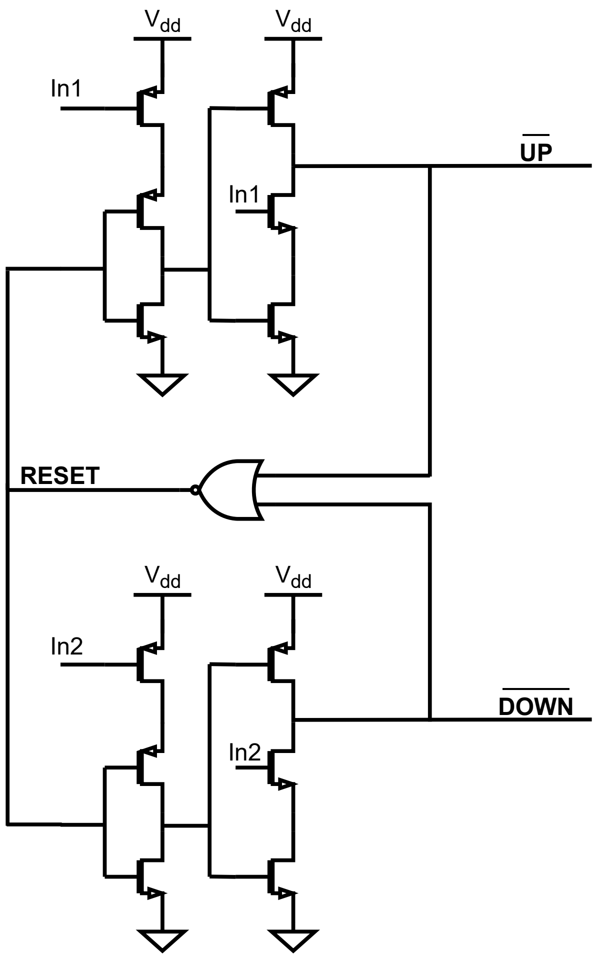

Figure 3.6: TSPC based phase detector [14]

This phase detector (PD) is employed for two main reasons. Since it comprises two "true single-phase" latches, the speed of this PD is faster than the basic PD structure. Moreover, there is a minimum output pulse even when there is no phase difference at the input. Since the circuit is symmetrical, this can be explained by examining one-half of the circuit. When the input is ONE, and the reset input is ZERO, the output UP is high. When input 1 becomes high, UP becomes low. Up only will be high again when the reset becomes high. When the reset is high, it takes about two gate delays before UP is high again. In total, it takes a minimum of three gate delays for the UP signal to become high again, even when both signals arrive simultaneously.

Nevertheless, the charge pump still generates a pulse even when there is no phase difference, which must be carefully investigated. The charge pump, whose inputs are controlled by the UP and DOWN signals, creates a current that produces the output control voltage. When the pulse generated by the phase detector gets close to ZERO, the charge pump should not produce a current since the UP and DOWN signal will quickly fall up/down.

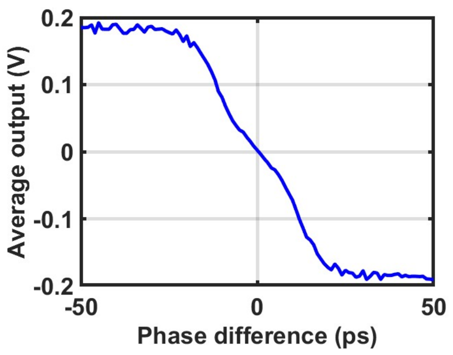

Figure 3.7: PDF characteristics

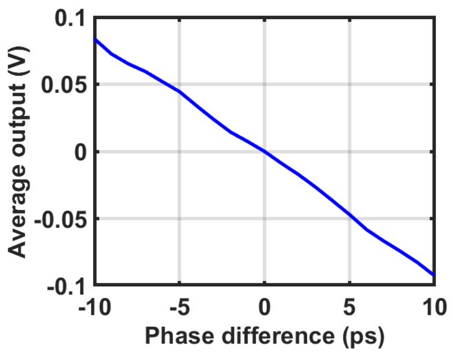

Figure 3.8: Average output voltage around zero phase difference

An essential feature of the charge pump is the mismatch between the charge and discharge current. In a DLL, this issue leads to an offset error in the loop, which is undesirable; how this mismatch exactly works can be explained using Figures 3.9 and 3.10.

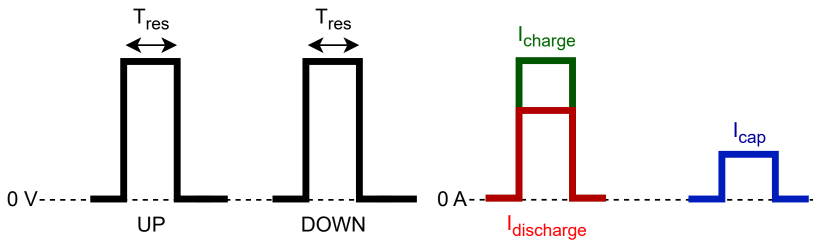

Figure 3.9: Current mismatch without a phase difference

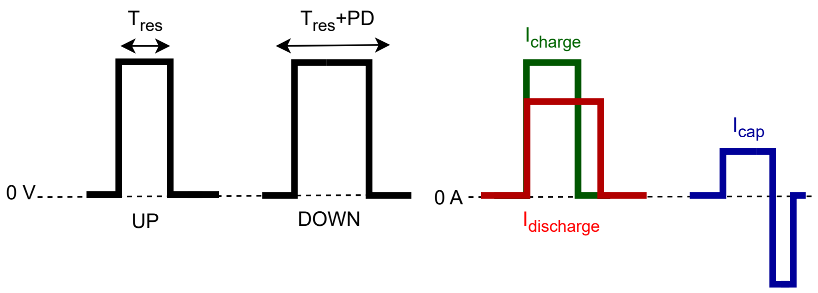

Figure 3.10: Current mismatch with a phase difference

In Figures 3.9 and 3.10, all four pulses are shown. In the first case, there is no phase difference at the input of the phase detector, as shown in Figure 3.9. The first two black-colored signals are the UP and DOWN signals from the phase detector, which is the case with equal three gate delays, as discussed earlier, called Tres. When there is a mismatch between the charging and discharging of the capacitor, there will be a net current flowing through the capacitor at the output of the charge pump, thus changing the control voltage even if there is no phase difference. In the case of Figure 3.10, there is a phase difference, but due to the current mismatch, the capacitor will be charged and discharged again so that the control voltage at the output does not change. These charging and discharging schemes demonstrate the current mismatch and the offset error relationship.

$$T_{res} * I_{charge} = (T_{res} + PD) * I_{discharge}$$

$$\frac{I_{charge}}{I_{discharge}} = \frac{(T_{res} + PD)}{T_{res}}$$

$$\frac{I_{charge}}{I_{discharge}} = 1 + \frac{PD}{T_{res}}$$

$$PD = Mismatch * T_{res}$$

The minimum pulse introduced by the phase detector also introduces an offset error when there is a current mismatch. The mismatch in the charge pump currents has to be diminished to minimize the error.

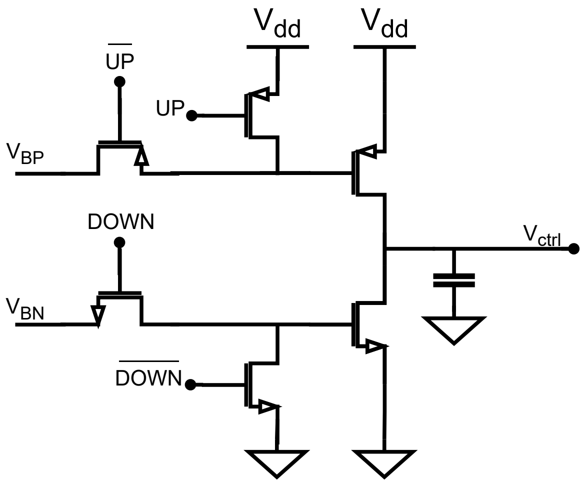

Figure 3.11: A Gate switched charge pump [17]

The chosen charge pump structure is the gate-switched charge pump, depicted in Figure 3.11. This structure is preferred due to only having one NMOS and one PMOS at the output stage, which will be helpful in nanoscale CMOS processes with limited voltage headroom.
In designing this charge pump, many important issues had to be taken into account. The current mismatch, as described earlier, substantially impacts the offset error of the DLL. Also, the speed at which the charge pump operates must be considered. To decrease the mismatch due to channel length modulation, the output stage transistor size is usually increased, which is impossible in this case because making the CMOS transistors too large leads to the charge pump not operating correctly at the speed of 2.5 GHz.

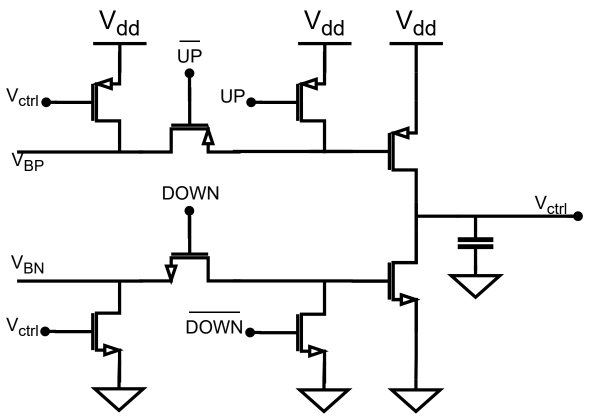

Figure 3.12: A Gate switched charge pump with Feedback

So to decrease the mismatch instead of increasing the output stage transistor lengths another method has to be used which in this case will be by using feedback. This feedback has been implemented using 2 extra transistors which can be seen that have been added in figure 3.12. As can seen in figure 3.13 Without any feedback you can see the effect the channel length modulation which is that the discharge and charge current are not constant with respect to the control voltage.
Hence, a feedback technique is incorporated to decrease the mismatch instead of increasing the output stage transistor lengths. This feedback has been implemented using two extra transistors, shown in Figure 3.12. As can be seen in Figure 3.13 , without any feedback, the effect of the channel length modulation is significant, in which the discharge and charge currents are not equal with respect to the control voltage.
To minimize the mismatch, the feedback transistor adjusts the biasing voltage further up or down depending on the control voltage. When the control voltage is low, the discharge current is much lower than the charging current due to channel length modulation and low control voltage. Due to the lower control voltage, the NMOS in the output stage falls out of saturation and slowly turns off the lower control voltage. The same happens when the control voltage rises and comes closer to the supply voltage of the PMOS transistor.
The feedback thus tries to compensate for this issue and make the charge pump's operating range larger where the mismatch is acceptable. Therefore, when the control voltage decreases, the discharge current becomes smaller, forcing the charging current to reduce to match the discharge current. The feedback transistor thus lowers the charging current by increasing the biasing voltage. While the control voltage comes closer to the supply, the discharge current has to be lower, which is executed by the NMOS pulling transistor, in which the biasing voltage is lower depending on how high the control voltage gets.

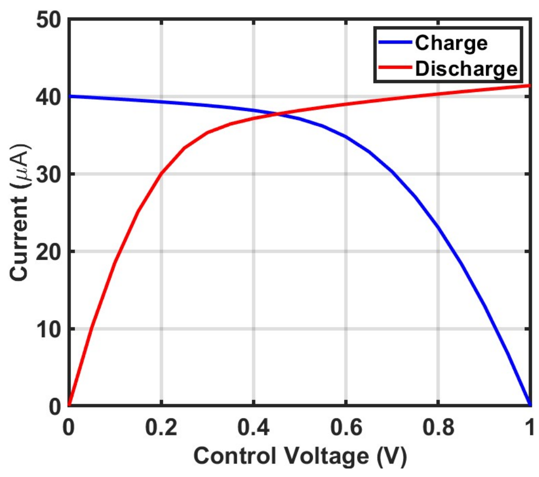

Figure 3.13: (Dis) Charging current without feedback

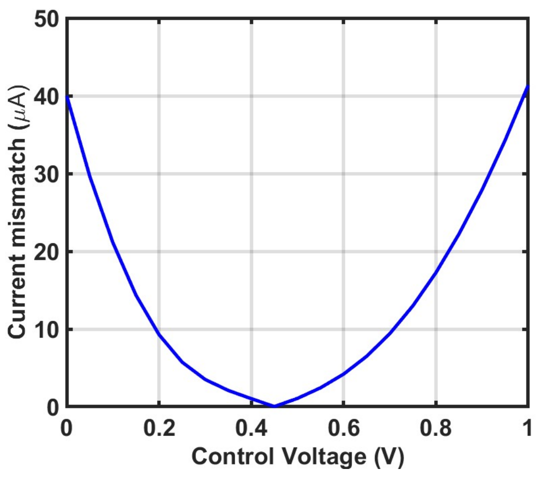

Figure 3.14: Current mismatch with feedback

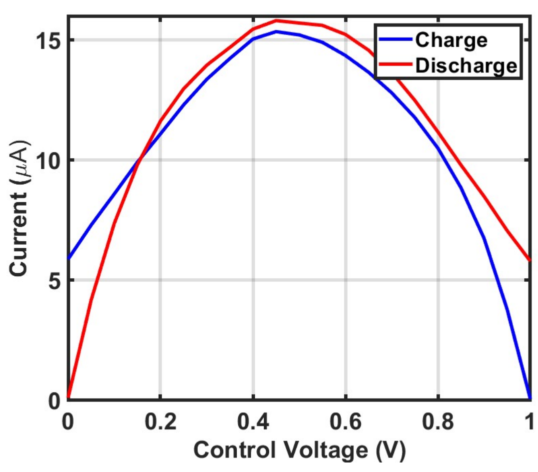

Figure 3.15: (Dis) Charging current without feedback

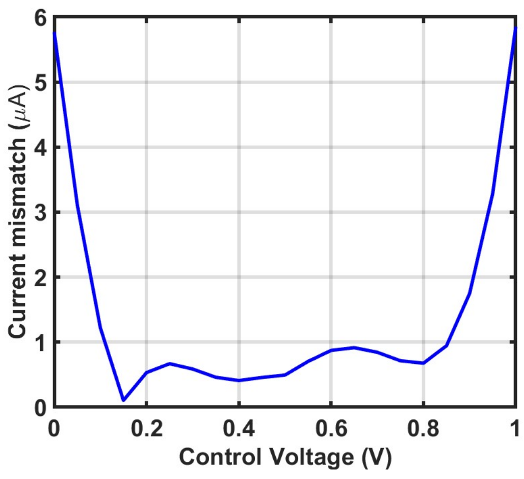

Figure 3.16: Current mismatch with feedback

3.3. Voltage controlled delay

Many different circuitry can be used to create delays, such as comparator-based delays, differential delay elements, thyristor-based delay elements, and inverter-based delay elements. Inverter-based delays are the most popular ones among these choices because of their simplicity and low power consumption is also the main reason it will be used in this DLL [20].

The inverter-based controlled delays can be realized in two fashions: Analog and digitally controlled inverter delays. As the name suggests, the analog-controlled delays are controlled by an analog input voltage. The digitally controlled delay will be controlled with control bits. Digitally controlled delays depend on the number of control bits, the resolution of the delay element for the delay range, and the accuracy of these delays.

The voltage-controlled delay (VCD) element is being discussed at the end because essential information was required in the previous section on the charge pump. Based on the charge pump discussion, the minimal current mismatch occurs when the control voltage is approximately half the supply voltage. Thus, the VCD elements are designed based on the criterion mentioned above.

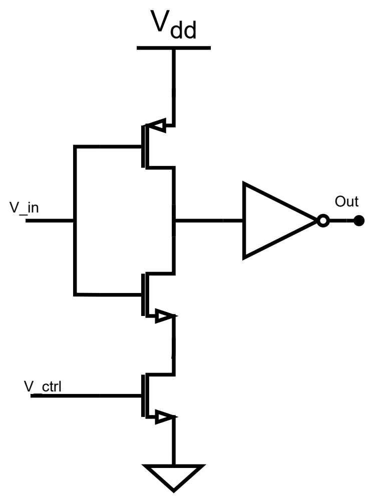

Figure 3.17: NMOS Current starved inverter delay

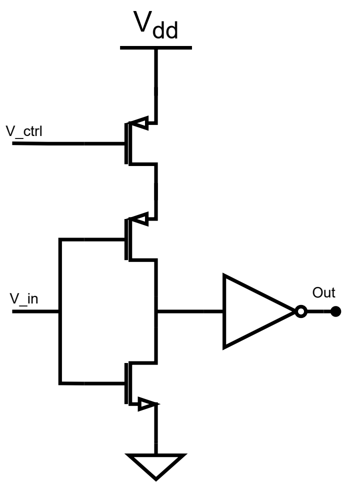

Figure 3.18: PMOS Current starved inverter delay

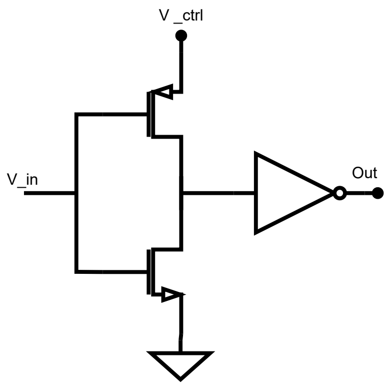

Figure 3.19: Power supply controlling inverter delay

When considering an inverter-based delay element, the first candidate is the current starved inverter. However, it has a critical issue that relates to its control voltage with relatively small ranges

- From VSS to (VDD-Vth) for PMOS current starved inverter delay (shown in Figure 3.18)
- From Vth to (DD) for NMOS current starved inverter delay (shown in Figure 3.17)

This aspect depends on whether the current starved path will be the charge or discharge path. The issue is that the control voltage range is small, and the control voltage does not include half the supply voltage in either of the voltage ranges with a supply voltage of 1 V.

Another possibility for the delay element (Figure 3.19) is the power supply controlling inverter. With this structure, the controlling voltage will usually be where the voltage source of the inverter is. The delay of the element decreases by lowering the inverter's supply voltage. The issue with this approach is that the delay range is also relatively small, which is a critical issue since, with PVT variations, there will be no guarantee that the required delay will be in the control range of the element.

Since these straightforward delay elements can create significant errors, they are not suitable choices employed in a DLL where the error is directly proportional to the CDR error phase. Therefore, to design a delay element whose follow-up charge pump current mismatch is sufficiently minimized, as seen in the charge pump section, the delay element should be designed in such a way that its desired delay will be as close as possible to half of the supply (0.5V).

For this to be feasible, a rail-to-rail voltage-controlled delay element must be adopted to address the required half-supply delay condition, which is impossible using a simple current-starved inverter.

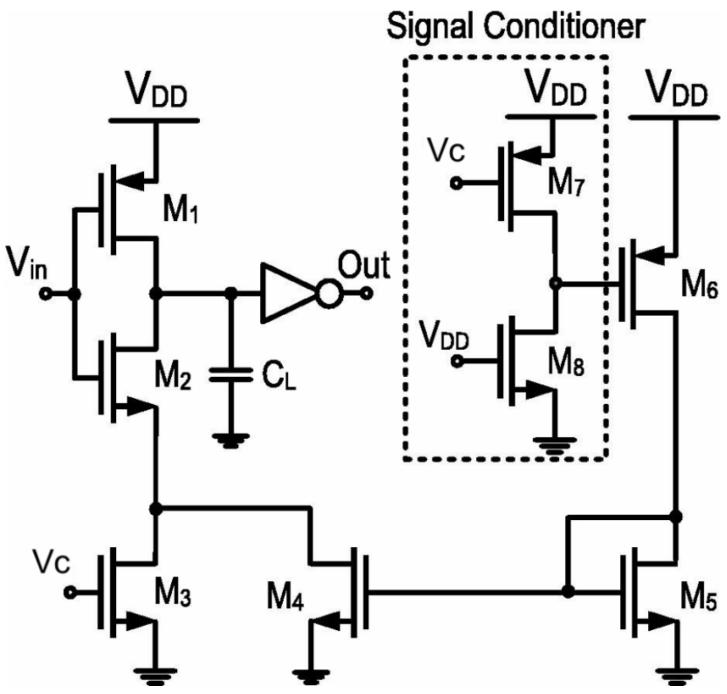

Figure 3.20: Linear voltage controlled delay element[20]

To understand the delay element used in this project, let's review the principle operation of the selected delay element, which is shown in Figure 3.20. In this context, transistors M1 to M3 create the current starved delay of Figure 3.17. Thus, M4 ensures current flows through the inverter when M3 is turned off. This condition is met using a signal conditioner highlighted in the figure. This signal conditioner exploits the control voltage to generate a current through M6, which gets mirrored into M4.

The issue with the delay element in Figure 3.20 is that the delay is only being applied to one edge of the signal. In a DLL generating multiple phases, this issue leads to more duty cycle errors. To make the clock signals more symmetric, a modified version of the design proposed in [7] has been adopted in the proposed DLL as its delay element.

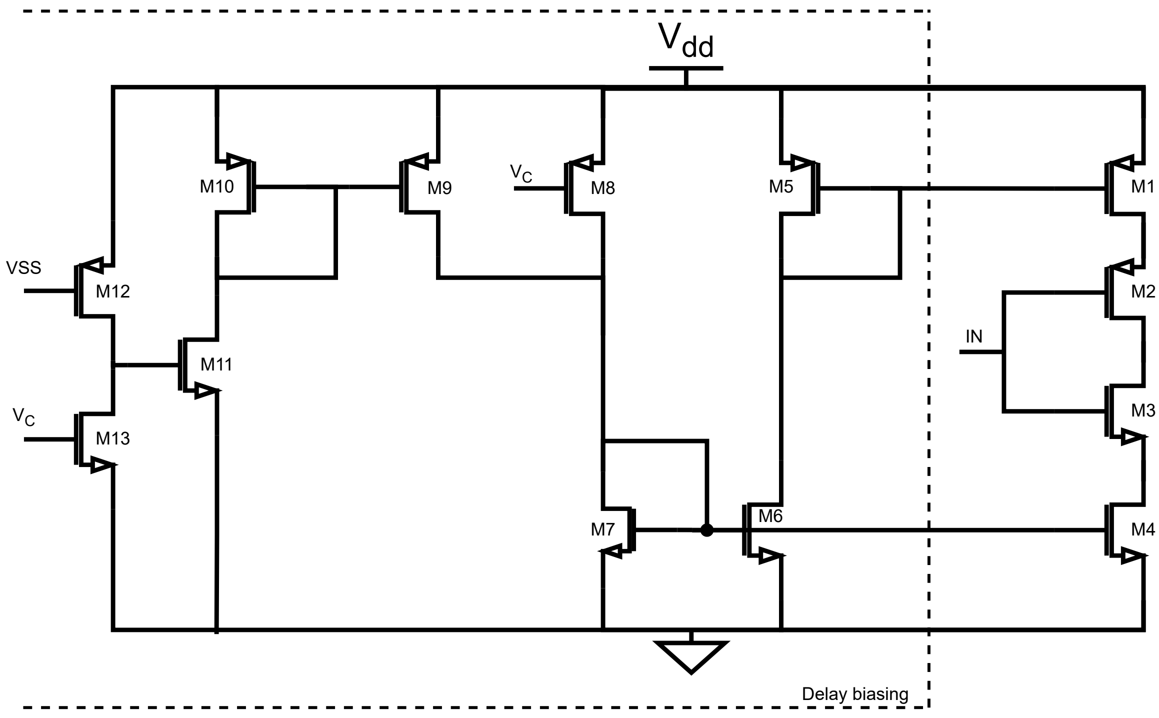

Figure 3.21: Linear voltage controlled delay element

As seen in Figure3.21, both edges are delayed in the new configuration. The delay element has the delay biasing and the current starved inverter parts. In this regard, the proposed DLL comprises multiple similar delay elements. They require one biasing circuit. Thus, all these different delay elements will be biased similarly and loaded equally to produce precise delay leading to the generation of multiple clock signals with accurate phase differences.

Another critical issue considering the delay elements and the DLL is PVT variation. In different process corners, the designated delay range and so-called "delay gain" would not be sufficient. With a smaller range, the delay element will not be suitable for phase locking. With a larger range, the DLL might experience false locking where the phase detector observes no phase difference, but the DLL is locked on a wrong delay. Thus, two ways to circumvent this problem are to make a wide range delay with false lock detection or create delay elements with biasing for different process corners and make the range large enough to account for voltage and temperature variation. The delay element in the proposed DLL is a delay element with dedicated calibration schemes to alleviate the need for false lock detection.

The calibration that has been added to the delay element can be seen in Figure 3.22. This calibration was inspired by the digitally controlled delay element in [16]. Comparing Figures 3.21 to 3.22 , there are 4 transistors added: M14 to M17. The first two important changes are M14 and M15, employed so that when they are active and connected to M5, the current through M5 will be larger, resulting in a smaller delay at the output node. This calibration can thus be fulfilled using even more transistors so that the delay element can be calibrated for whatever process variation. Subsequently, for the voltage and temperature variations, the delay element needs to have a wide enough range of delays to accommodate those.

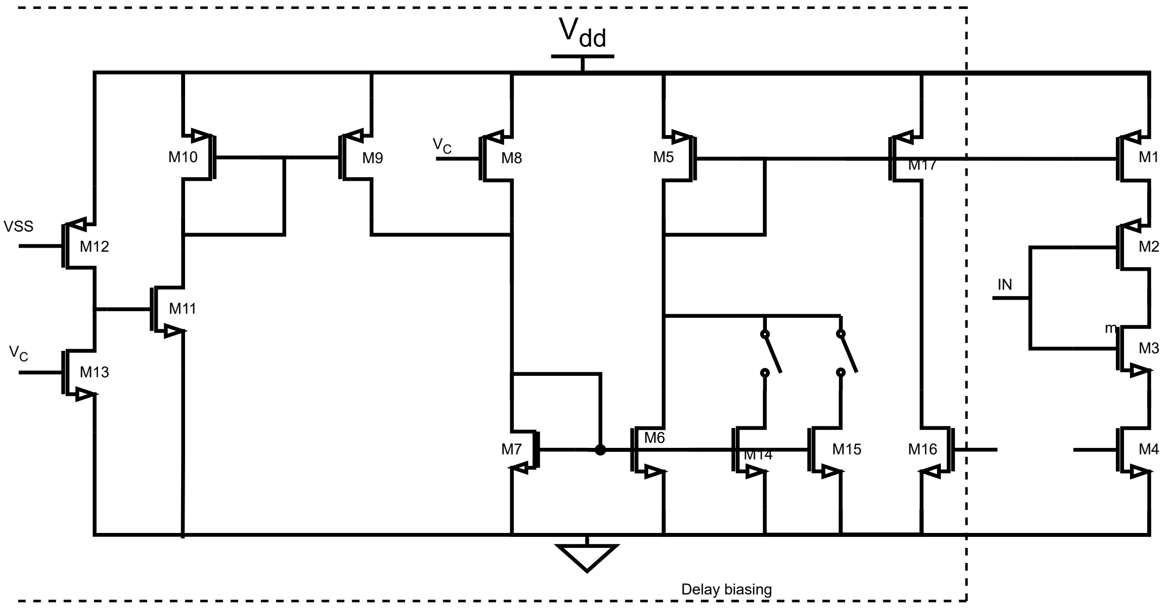

Figure 3.22: Linear voltage controlled delay element with calibration added

This calibration has been tested in two different corners, namely the SS and FF corners since these are the fastest and slowest MOS transistors. In Figures 3.23, 3.24, and 3.25, the resulting delay can be seen where the range is kept almost equal and where the desired delay of 50ps required a control voltage of 0.5V so the current mismatch does not significantly impact the offset error.

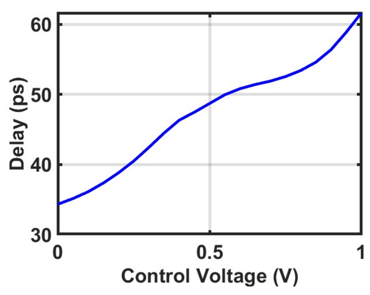

Figure 3.23: Linear delay element result in typical condition

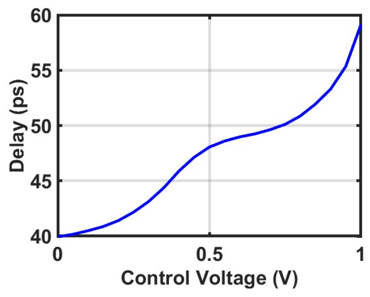

Figure 3.24: Linear delay element result in SS corner

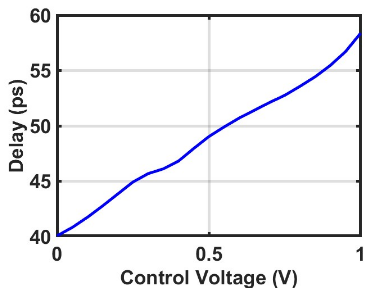

Figure 3.25: Linear delay element result in FF corner

3.4. Duty cycle correction

The duty cycle of a clock signal refers to the ratio of the time the signal spends in the high state to the time it spends in the low state. Generally, the duty cycle should be 50%, meaning the signal is ONE (logic level) for precisely half of the period and ZERO (logic level) for the other half. However, due to various factors such as noise, asymmetrical rise/fall times, and device mismatches, the duty cycle of a clock signal can deviate from 50%.

Due to multiple phases of the output clock signal generation, the duty cycle of the reference signal of the DLL can be affected due to a phenomenon called duty cycle distortion. This feature occurs when the DLL locks onto the rising edge of the reference signal and generates multiple phases from that point without considering any differences in the duty cycle of the reference signal.

As indicated in the DLL, each output phase is made by a separate delay line, and the delay in each line is tuned separately to generate the desired phase shift, indicating that each delay element can add some duty cycle distortion that will accumulate. One of the main reasons for duty cycle distortion in the current design of the DLL is the uneven rise and fall times in the used delays. This uneven rise and fall time can contribute to duty cycle distortion, which can be explained using Figure 3.26. This figure shows a highly exaggerated uneven rise/fall time signal. The signal underneath that will be the signal after inversion. If the inverter has a more even rise/fall time than the delay element, the inverted signal will have an altered duty cycle. In the case of the current DLL, since the delay elements are inverting elements, a second inverter is exploited to make it a non-inverting delay stage. This configuration makes the delay element's rise/fall time difference, leading to a duty cycle difference.

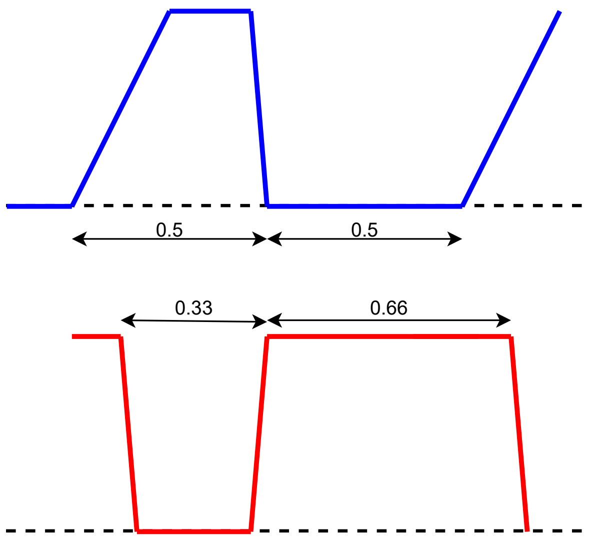

Figure 3.26: Duty cycle distortion due to uneven rise and fall times

To address this issue, a digital duty cycle correction (DCC) has been employed to use a half-length DLL to generate a 50% duty cycle. In this approach, since DLL comprises an even number of delay elements, the digital DCC utilizes a pair of complementary clock signals (two clock signals 180 degrees apart from each other), as depicted in Figure 3.27.

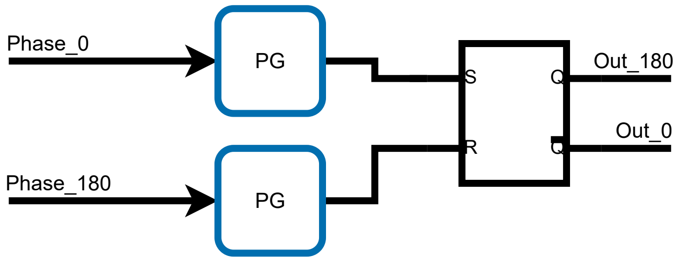

Figure 3.27: Duty cycle correction circuit [24]

As shown in Figure 3.27 a pair of complementary clock signals, with a duty cycle larger/smaller than 50%, are first applied to two pulse generators (PGs) whose outputs are Set (S) and Reset (R) inputs of an SR latch. Eventually, the output signals of an SR latch are 50% complementary clock signals.

One issue with this DCC circuit is that the S-to-Q and R-to-Q delays are unequal. This feature leads to a duty cycle error equal to the difference between these delays. In other words, this issue gives rise to unusable complementary output due to larger signal traverses for S-to-Q and R-to-S paths. Consequently, the latch must be designed to minimize this difference so that the duty cycle can be closer to 50% . Hence, we must employ two consecutive SR latches to equalize the delays leading to 50% duty cycle clocks.

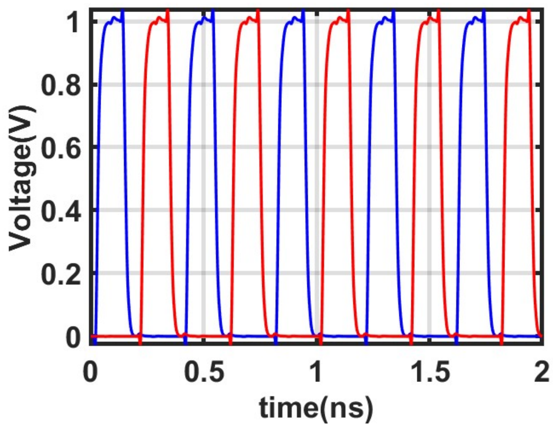

Figure 3.28: Input clocks DCC

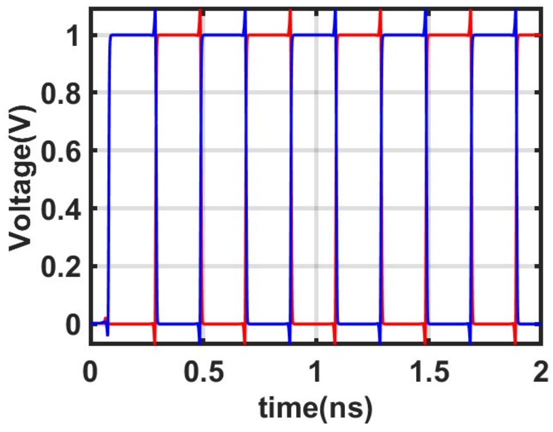

Figure 3.29: Output of the DCC

A simple test of this DCC circuit has been fulfilled to verify its operation. This simulation used a 2.5 GHz clock signal with a duty cycle of 30% at the input of two different DCC units. This arrangement is due to the modification of the latch in the DCC. Only one of the outputs will have an adequately corrected output.

3.5. Results DLL

Multiple simulations have been executed to verify the DLL performance. First, a transient simulation was performed where the transient response of the control voltage and the phases after the DLL has been settled can be seen in Figures 3.30 and 3.31.

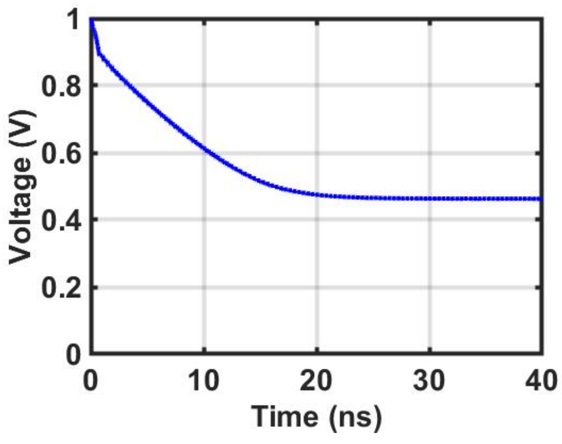

Figure 3.30: Transient response of Vctrl

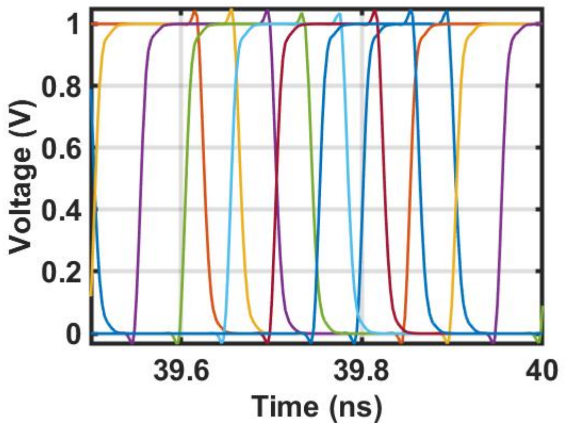

Figure 3.31: Rising edges of 8 DLL phases

Because of the uneven rise and fall times, the duty cycle of the signals becomes relatively small, and the effect of the DCC circuit impacts the duty cycle of the DLL output phases. According to Figures 3.30 and 3.31 the duty cycle of each phase becomes lower and lower because of the effect described in the previous section.

Because of the uneven rise and fall times, the duty cycle of the signals becomes relatively small, and the effect of the DCC circuit impacts the duty cycle of the DLL output phases. According to Figures 3.35 and 3.35 the duty cycle of each phase becomes lower and lower because of the effect described in the previous section.

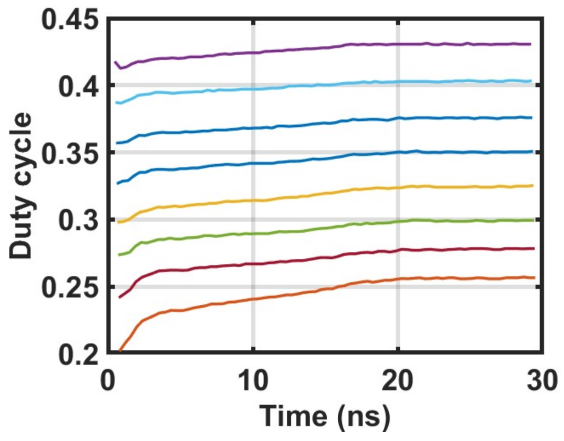

Figure 3.32: DLL output phases duty cycle

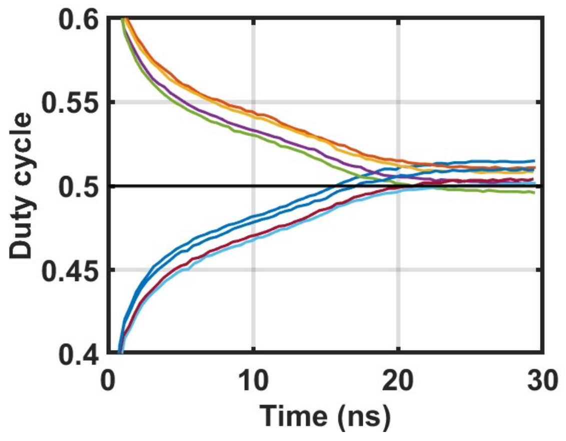

Figure 3.33: Duty cycle at the output of the DCC

The DLL also has been tested across multiple input frequencies to test its offset voltage and its settling voltage. When looking at the shape of the offset error of the DLL this is fairly similar to the shape of the measured current mismatch in figure 3.16 which also is in agreement with the fact that the current mismatch is thus proportional to the offset error.

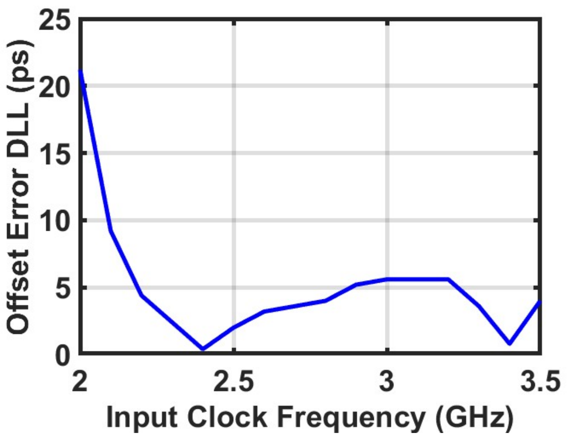

Figure 3.34: Offset error of the DLL

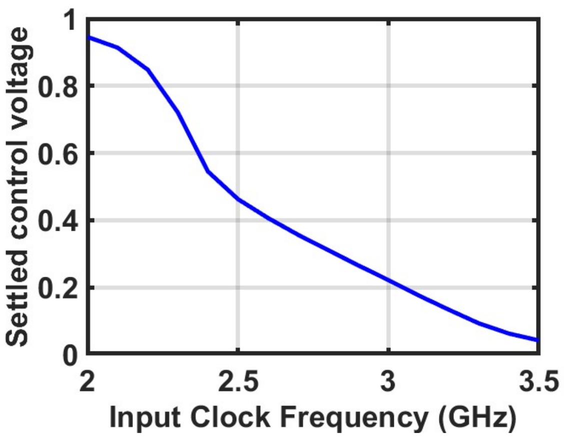

Figure 3.35: Control Voltage of the DLL

At 2.5GHz the offset error is equal to 3ps, this results in some error in the phases generated by the DLL. This error can be observed in figures 3.37 and 3.36. There is a peak that can be seen in the DLL phase error which is due to the offset error. The error between the phase difference between the first and last phase is always about equal to the offset error. while the rest of the differential errors are fairly similar due to loading each delay element with a similar load.

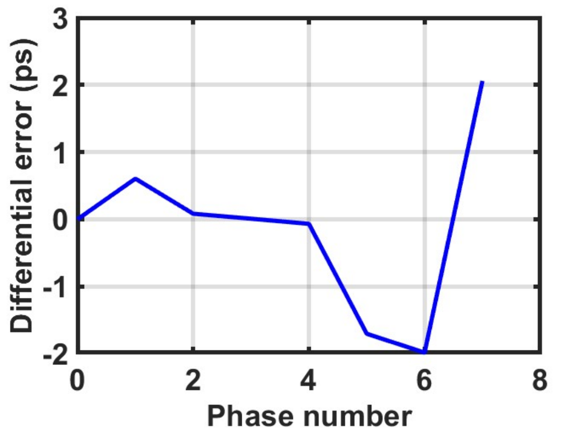

Figure 3.36: Differential phase error

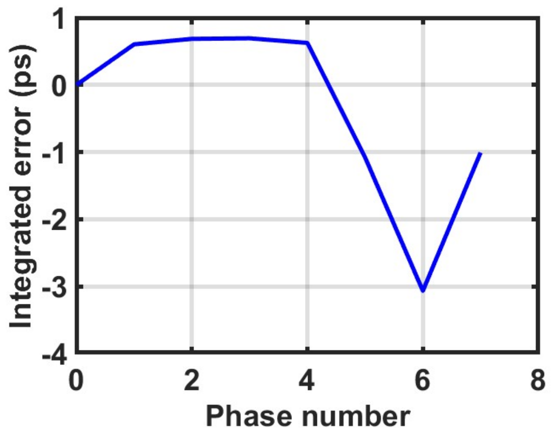

Figure 3.37: Integrated phase error

In the final outcome, the designed DLL generates 8 phases with an operating range from 2 to 3.5GHz. Due to eliminating the Phase Detector dead zone and minimizing the current mismatch in the Charge Pump the offset error at the operating frequency has been measured to be 4.5 ps.
Overall, this DLL's successful design and evaluation highlight its potential as a reliable component in high-frequency applications, exhibiting excellent phase alignment and minimal offset error.

4
CDR Design

As mentioned in previous chapters, the proposed and selected clock and data recovery (CDR) architecture is a Phase Interpolated(PI)-based CDR. In this chapter, we will discuss the design of the phase interpolator and its related DLL and the digital control circuit of the CDR. A simple schematic of the CDR is shown in Figure 4.1. Here the incoming data operate at a data rate of 10Gb/s, and an incoming clock reference runs at 2.5Ghz. A quarter rate architecture has been designed to enable optimum sampling and retiming of the random NRZ data

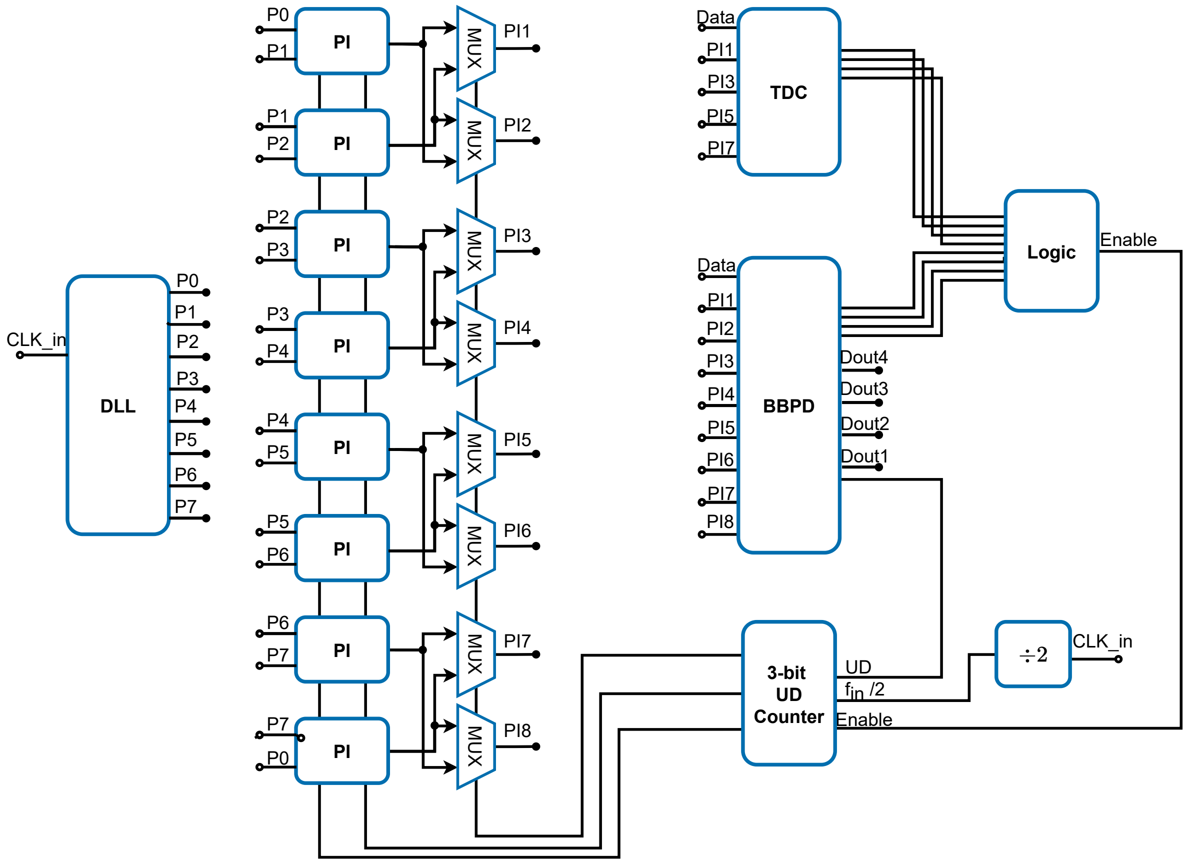

Figure 4.1: System Schematic of the complete CDR in this work

A quarter rate design indicates that the circuit operates at a clock frequency of one-quarter of the data rate. Since the data rate is 10 Gb/s, the clock frequency would be 2.5 GHz. In other words,

operating at a lower clock frequency can reduce power consumption and improve the stability of the CDR.
However, it also requires more complex circuitry to interpolate the clock phases and synchronize the incoming data. In this approach, the digital circuit must generate four clock edges for every incoming data bit to achieve a 10 Gb/s data rate with a quarter rate design
The phase interpolater (PI) CDR consists of two loops. In the clock reference loop, reference clocks with different phases are generated. This loop is usually a DLL or a PLL that generates multiple phases. The second loop is the CDR loop, where the clock will be recovered, and the data will be sampled. The CDR loop will be discussed in this chapter, while in the previous chapter, the DLL has already covered.

The PI-based CDR is one of the CDR architectures which uses a reference clock. One of the advantages the PI-based CDRs compared to continuous-time CDRs is that there is no on-chip clock generation (oscillator). This feature thus entails less power consumption and a smaller area. A forward or on-chip PLL generally generates the required reference clock signal. If this incoming signal does not have multiple phases (edges), these phases must be produced using a DLL first.

The complete CDR system is illustrated in Figure 4.1. The figure depicts the phases generated by the Delay-Locked Loop (DLL), which are then transmitted to the phase interpolator. The phase interpolator performs interpolation on the incoming phases based on the input bits. The mux chooses which of these PI outputs is the sampling clock and which are the extra signals generated for the Bang Bang Phase Detector (BBPD). The MUX outputs are thus sent to the Time to Digital converter (TDC) and the BBPD will combine their outputs to create a enable signal which will control whether the counter will go to the next phase or will lock into place. Additionally, there is another control dimension represented by the Up/Down (UD) signal, determining whether the counter increments or decrements. The BBPD possesses the capability to detect whether the clock is leading or lagging in relation to the desired sampling moment, enabling faster phase locking. All these parts will be discussed separately in the following sections.

4.1. The PI phase resolution

The first design choice is PI resolution or PI number of different phases. This design choice impact significantly the BER of the CDR system. A MATLAB simulation has been performed to elaborate on the selection of PI resolution.

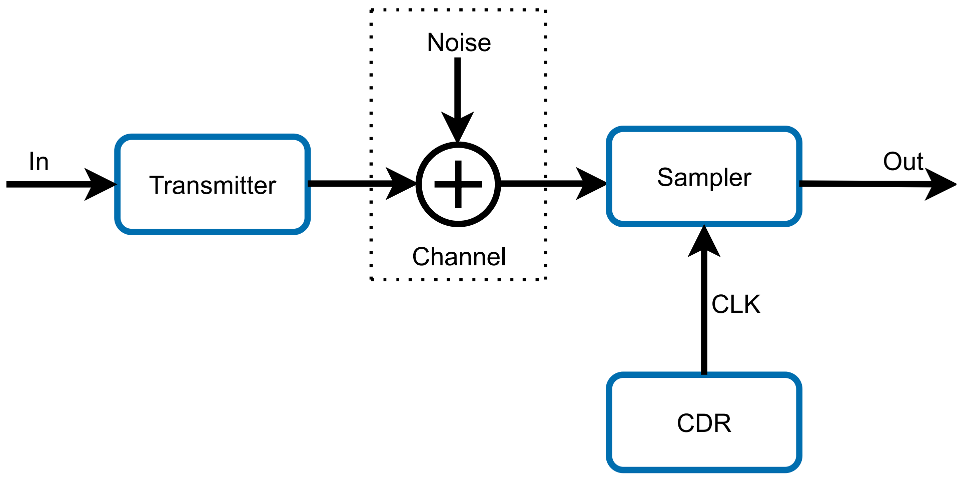

Figure 4.2: Basic schematic to estimate the phase resolution

Figure 4.2 exhibits the basic schematic used for estimating the BER. Here, the transmitted signal first pass through a noisy channel, degrading its signal-to-noise ratio (SNR). This channel will also have low pass characteristics. Subsequently, this signal is sampled with a clock signal chosen from the CDR.

In the MATLAB estimation of Figure 4.2, a few nonidealities, such as transmitter and channel noises, have been incorporated to determine how many phases are needed. The deterministic errors due to insufficient phase resolution and CDR nonidealities deteriorate the CDR's BER performance. If the middle of the bit period is precisely in between two phases generated by the CDR, the maximum sampling error will be half of the phase resolution of the CDR.

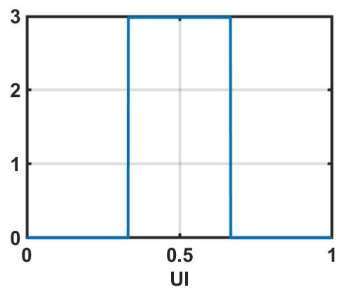

Figure 4.3: PDF sampling timing of the sampling error

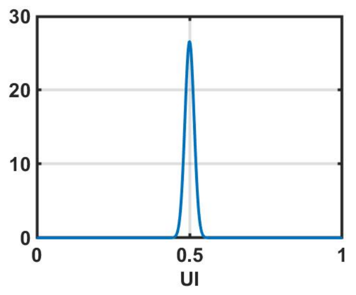

Figure 4.4: PDF sampling timing of the random jitter

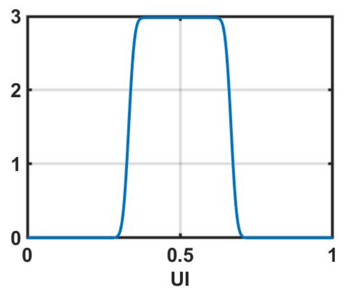

Figure 4.5: PDF sampling timing of Addition of RJ and sampling

Due to the PI's phase quantization error and clock jitter, CDR experiences a timing error in sampling the data. Moreover, the incoming data exhibit a random jitter of 1.5ps [6][5], as mentioned in Chapter 1. While the sampling error will be somewhere between: $-(2 * \#phases * (10e9))^{-1}$ and $(2 * \#phases * (10e9))^{-1}$, assuming the phases are accurately generated, and the phase closest to the middle eye has been chosen. Aside from the jitter of the incoming data and the sampling error, the jitter caused by the CDR was not known before the design, so the number of phases has been estimated without including the latter issue.

The probability density function(PDF) of sampling at a specific point has been created in MATLAB to estimate the PI's phase resolution. For the sampling error, this is a continuous uniform distribution in which the probability in the whole range is equal due to entirely random NRZ data. The range for this deterministic error is similar to the resolution of the phase interpolater. Figure 4.3. depicts the sampling error.

The random jitter of the clock is considered a normal distribution indicated in Figure 4.4. Now, since the sampling error and the error due to random jitter are not correlated, the PDF of the sampling error is the convolution of the two PDFs. This aspect is because the absolute timing error comprises the sampling and the random jitter. This addition translates to a convolution of the two PDFs, shown in Figure 4.5.

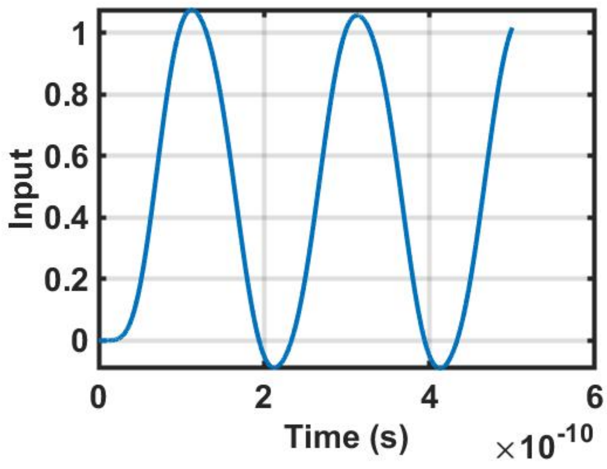

Figure 4.6: Filtered data used to calculate BER

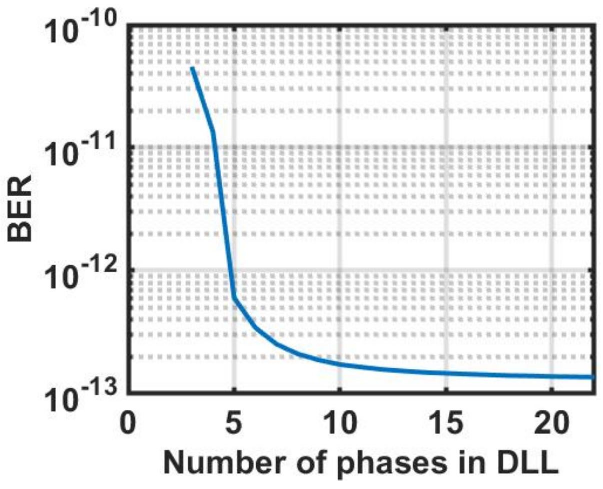

Figure 4.7: BER against the amount of phases at 10Gb/s with a SNR of 22dB

Using the PDFs shown in the aforementioned figures the BER can be estimated by applying this convoluted PDF to an incoming filtered signal. To perform this estimation, two scenarios have been considered: Sampling a '0' bit or sampling a '1'. When Sampling a 1 there is an error when the sampled signal is below the threshold which in this case is assumed to be 0.5. While in the case of a 0, there is an error when the signal gets above this threshold. The estimation of the BER takes into account that each bit has an equal probability of occurring. Consequently, the probability of an error can be calculated as follows:

$$P(Error) = \frac{1}{2} * P(error|D = 0) + \frac{1}{2} * P(error|D = 1)$$

Applying everything assumed so far the BER can be calculated and subsequently plotted against the number of phases at 10GHz, which can be seen in figure 4.7. From this figure can be seen that the highest gain in BER is taken when going from 0 phases to about 6 phases. Here it starts to flatten out. As a result, for this particular project, a total of 8 phases at 10 GHz have been selected, considering a quarter-rate design that would involve 32 phases.

4.2. Digital Control Circuitry

A digital circuit that controls a PI-based CDR is a crucial component. It compares the incoming random data phase with one of the PI's phases generated by DLL. A phase detector performs this comparison. The phase detector determines the phase error between the two signals and indicates whether this error is within the DLL/PI's phase range.

If the phase difference exceeds the PI's resolution, the digital circuit will go to the next phase using a counter with an enable signal. This counter is employed to generate a new phase interpolated signal closer to the incoming signal's phase. This process is repeated until the phase difference is within the PI's phase range.

The phase interpolation is realized using an inverter structure. These inverters are structured in a way that the weighting of the PI can be changed with two or more control bits.

4.2.1. Counter

The counter used in the digital control circuit is a 3-bit asynchronous counter. This counter should be able to count from 0 to 7 in a binary fashion, which is enough when looking at the number of phases generated and the fact that a quarter-rate design has been selected. The 32 phases generated at 2.5GHz is thus equal to having eight phases equally spaced apart in each period of a 10 Gb/s data signal.

Table 4.1: Truth table of the JK-flipflop

clock	J	K	Q
↑	0	0	Latch
↑	1	0	1
↑	0	1	0
↑	1	1	Toggle

A JK flip flop (FF) is adopted as a 3-bit asynchronous counter whose truth table is shown in Table 4.1. According to its property, a JK-FF can be configured as a counter when the J and K inputs are high in the output toggles. Thus, it is being used like a toggle flipflop.

As shown in Figure 4.8, when counting in a binary fashion, the LSB is constantly toggling, which is why the first JK-flipflop is connected to VDD. The second bit always toggles when the previous bit (the LSB) becomes high. Furthermore, the bit after (last bit) starts toggling when the first two bits become high. This function is realized using a AND gate in Figure 4.8.

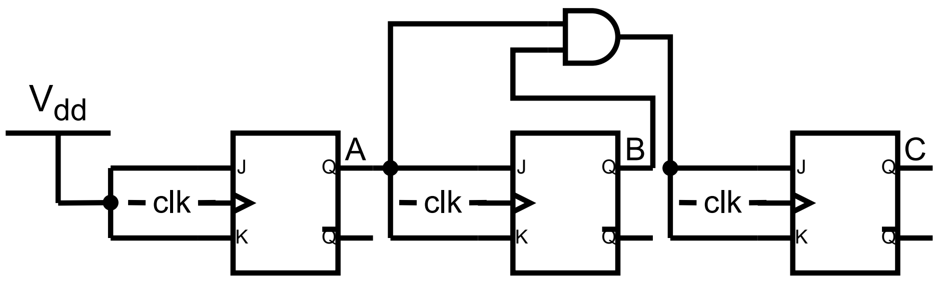

Figure 4.8: 3-bit counter with JK-fliplops

Figure 4.8 shows a simple counter that can only count upwards and has no enable signal. While for the control circuitry, both are needed since the incoming random data can be leading or lagging the clock. Thus to achieve fast locking, counting both ways is necessary. Also, the CDR must manage a low-frequency jitter by accomplishing both up/down counting operations.

To implement the up/down count functionality, we can use two additional control inputs: up/down. The way the counter counts up has already been discussed earlier. In the case of a count-down operation, instead of looking at the previous bits to see if they are a logic ONE, it is essentially the same but looks at whether it is a logic ZERO.

When the up/down input is high, the counter will increment the count on each clock cycle, and when it is low, the counter will decrease the count. This approach is achieved by using the output for counting up and its complement to examine if it needs to be counted down. Thus, when the up/-down signal is low and all the complementary outputs are high, the counter counts down, and when all the outputs are high, and the up/down signal is high, the counter counts up. This functionality is realized as depicted in Figure 4.9.

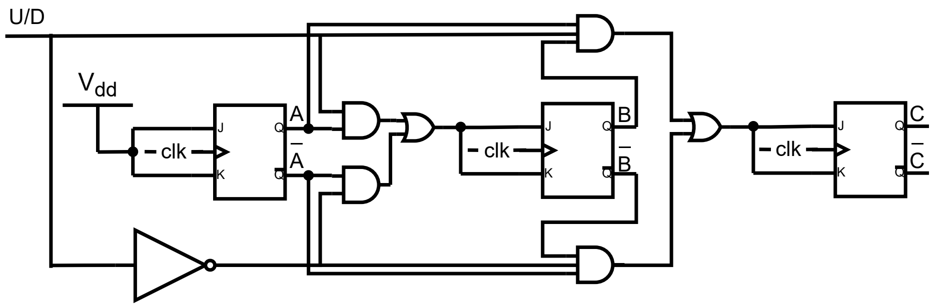

Figure 4.9: 3-bit counter with an Up/Down signal

The enable signal is used to control the operation of the counter. When the enable signal is high, the counter is allowed to operate, and when it is low, the counter is disabled, and the current status is held. This arrangement is being implemented by putting the clock signal through an AND-gate with an enable at the other input so that the reference clock input is blocked when the enable signal is low.

4.2.2. Multiplexer

A multiplexer (mux) is a digital circuit that selects one of several input signals and directs it to the output. There are several different types of muxes, including NAND-based muxes, transmission gate muxes, and cascaded NAND gate muxes. As the name suggests, a NAND-based mux uses NAND gates to implement the selection function.
The input ports to the mux are connected to the input ports of the NAND gates, and the output port of each NAND gate is connected to the input port of another NAND gate, forming a series of gates that produce the output signal. The advantage of this mux type is that it is relatively simple to implement and requires only a few components. However, it can suffer from issues with signal integrity, such as noise and delay, which can affect the quality of the output signal, and in the case of the CDR it delays the selection process.

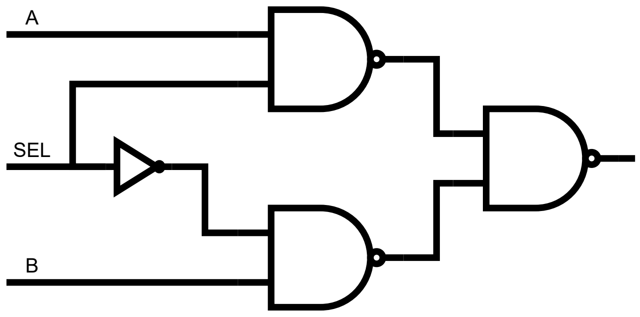

Figure 4.10: simple 2:1 NAND mux

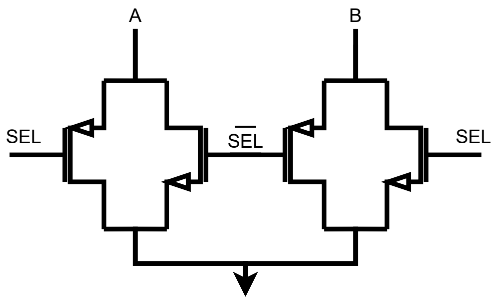

Figure 4.11: simple 2:1 transmission gate mux

A transmission gate mux uses MOSFETs to implement the selection function. Specifically, the input ports to the mux are connected to the MOSFETs' gates, and each MOSFET's output port is connected to the output line. When the select signal is high, the MOSFET conducts, and the corresponding input signal is transmitted to the output line. The advantage of this type of mux is that it has high signal integrity and relatively small delay, which can result in a high-quality output signal. However, it is more complex to implement than a NAND-based mux and requires more components.
A cascaded NAND gate mux is a type of mux that uses multiple levels of NAND gates to implement the selection function. Specifically, the inputs to the mux are connected to the first NAND gate stage, and the output of each NAND gate is connected to the input port of another NAND gate stage. The

last NAND gate stage produces the eventual output signal. The advantage of this mux type is that it can be more flexible and versatile than a simple NAND-based mux and can support a larger number of input signals. However, it can also be more complex and require more components than a simple NAND-based mux.

When creating a mux for the CDR circuit, a NAND-based mux has been chosen because the transmission gate mux does not have a path to the supply. Thus, when cascading multiple transmission gates to each other, the signal quality can be degraded too much.

4.2.3. Time to Digital Converter (TDC)

One of the most critical circuits in the CDR's digital control will be the input data's phase detection. The goal is for the signal to be as close as possible to the middle of the data eye. As discussed in the second chapter, the main PD used in CDR circuits is the bang-bang phase detector because of its straightforward design, and it also alleviates the problem of having no data edges. Using only a BBPD doubles the maximum error of half the distance between phases due to the toggling behavior of the BBPD. The chosen signal toggles between the two phases that are closest to the middle of the data eye. Thus, what happens when one of the phases is exactly in the middle of the data?

When one of the phases is in the middle of the data and the BBPD is still toggling this means that it will toggle between a phase exactly in the middle of the data and a phase that is one phase resolution away from the middle. This means that the maximum timing error will be twice the maximum timing error when you can lock onto a phase. To alleviate this problem, a TDC will be added next to the BBPD to make the CDR lock onto the closest data edge possible.

The BBPD will be used to see if there is a data edge. Also, will the BBPD be used to see if the chosen phase is leading or lagging from the middle of the data eye? But the TDC will make the whole CDR lock into one phase.

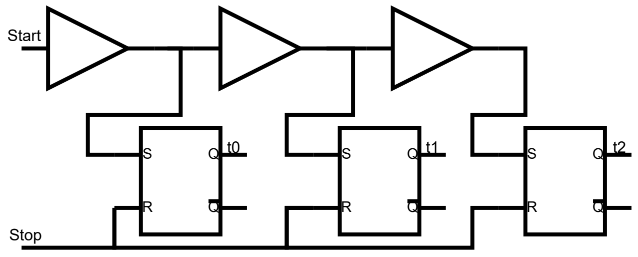

Figure 4.12: Example of an delay line TDC

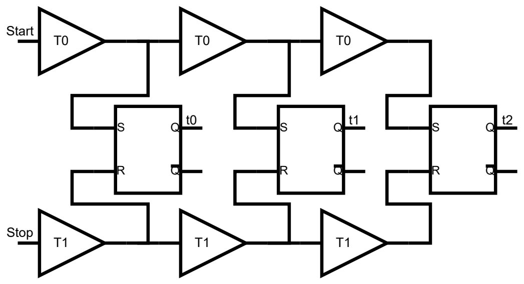

Figure 4.13: Example of a vernier-TDC

The phase detector is a quarter rate 2-bit vernier time to digital converter (TDC). A TDC is a circuit that translates a time difference into a digital signal. This functionality is usually fulfilled using delay cells. Employing a delay cell and a latch, the difference between two signals can be measured to be larger or smaller than the delay depending on the output of the latch. This type of TDC is called a delay line TDC as depicted in Figure 4.12.

In this context, a vernier TDC (figure 4.13) exploits two delay lines to create more possibilities for measuring different delays. The Vernier TDC enables measuring smaller resolutions and measuring 'negative' delays. The 'negative' delay signifies that the start signal can lead and lag the stop signal. This negative delay attribute can thus be used to choose the correct phases in the CDR.

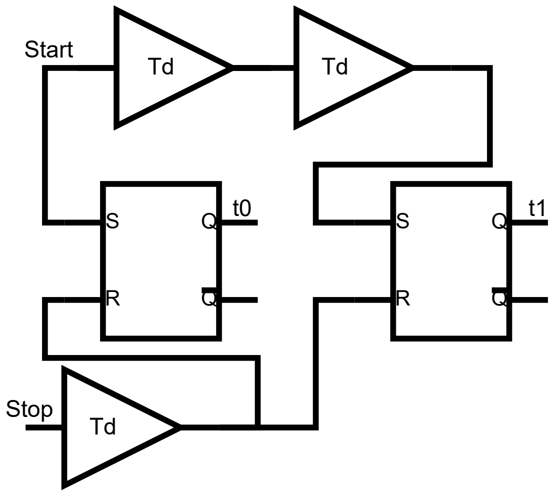

Figure 4.14: TDC used in the CDR

In Figure 4.14, the TDC that is used can be seen. This TDC has two outputs which are t0 and t1. T0 is the output indicating the negative delay mentioned earlier. Thus, when the t0 is high, the start input delay is smaller than the buffer delay. Moreover, when t1 is high, the stop input is delayed by a bigger buffer delay. In the use case of the CDR, the start input will thus be one of the phases that have been chosen, and the stop will be the incoming data.

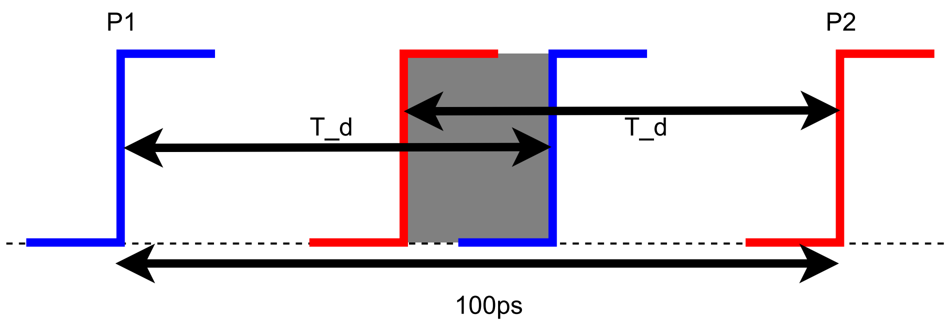

Figure 4.15: Timing of in design TDC

Figure 4.15 is presented to understand how this TDC selects a proper phase for the CDR. In this figure, two phases are shown of the four total to make the figure easier to digest. In the whole CDR, all four phases will have their own TDC. Between phase 1 and phase 2, a data edge will be possible. When there is a positive edge between p1 and p2, the system will thus focus on two different signals: the t0 of p2 and t1 of p1. This arrangement can be used since when the t0 of p2 is low and the t1 of p1 is high, the data edge will be in the grey area in Figure 4.15. This grey area can also be called the locking area, where the CDR locks to the chosen phase.

To realize this, the buffer delay has to be chosen so that the locking area is large enough for all phases to lock but not so large that unnecessary errors could be made. The minimum delay is half the bit period since there will be no locking area if it is smaller than half the bit period. To make the locking area equal to the phase difference between the phases, the delay must be equal to half the bit period plus half the difference between the phases, leading to a delay of 56.25 ps in this CDR.

The advantage of using this TDC is that the clock signal will not be toggling between the closest two phases, which has been discussed earlier. Another advantage is that the TDC delays do not need to be equal to the phase resolution of the CDR. This feature is due to retiming the incoming random data at the middle of the data eye, which simplifies the TDC enormously since instead of trying to design a TDC with a resolution equal to half the phase resolution, i.e., 6.25ps, the delays are much larger and easier to achieve.

A drawback of the TDC used in the CDR is its limited operating range. If the CDR is required to operate at a data rate of 9 Gb/s, the delay should be adjusted to approximately 69.4 ps instead of the default 56.25 ps to ensure a proper locking area at this specific data rate. To accommodate different data rates, the current-starved delay element from the DLL (as shown in Figure 3.22) is employed. This current-starved inverter can be designed with a slightly larger load compared to the DLL, resulting in a delay increase of approximately eight of the original delay, bringing it to 56.5 ps at the operating frequency. By utilizing the same biasing element as the DLL, the issue of delays is effectively resolved. As the DLL locks onto the correct delay for a broad range of frequencies, this delay element can be controlled to adjust the locking area for different data rates.

To verify if the TDC is working as intended the TDC has been tested with data arriving before the clock and the data arriving after the clock. When the data arrives early(figure4.16) t0 toggles when the data arrives somewhere between 57 and 58ps. This is a bit larger than needed but is not a major problem. When the data arrives late can be observed in Figure 4.17. This will result in a slightly larger locking range than necessary.

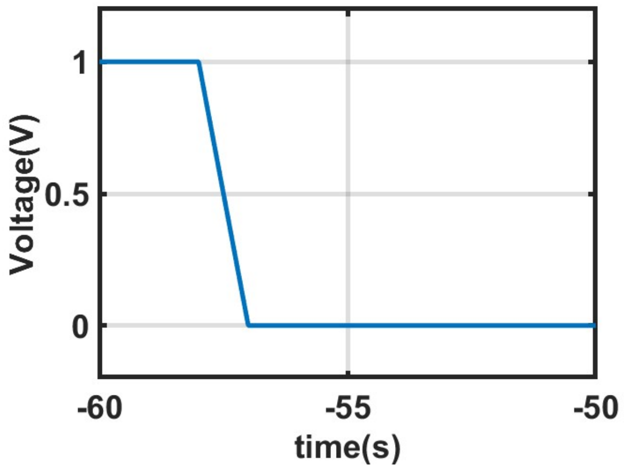

Figure 4.16: Data Arriving before clock (t0)

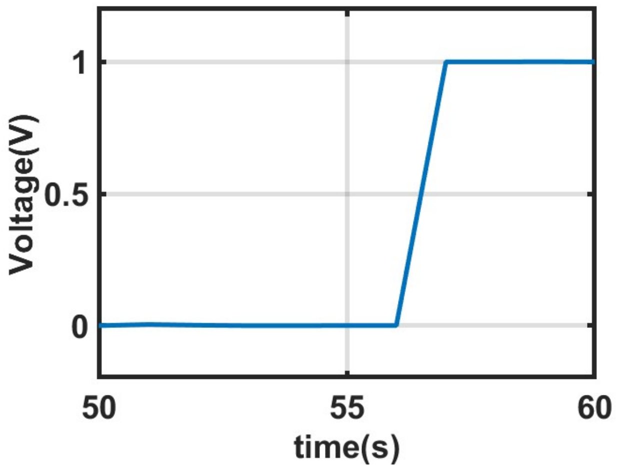

Figure 4.17: Data arriving after clock (t1)

4.2.4. BB phase detector

As discussed in the previous section, apart from the TDC that is being used, a BBPD is exploited so that the CDR stays in a locked state even if there is no data edge and to detect whether the signal is leading or lagging with respect to the middle of the data eye. This aspect is important because the PI-based CDR with few phases can get very far away from the center of the data eye in a few cycles of long-run data.

Two issues with the Alexander BBPD are shown in Figure 2.7 in Chapter Two. This phase detector does not operate at a quarter rate, so it has to be modified. The other issue is the speed of the latches, which must be fast enough to sample at 2.5GHz.

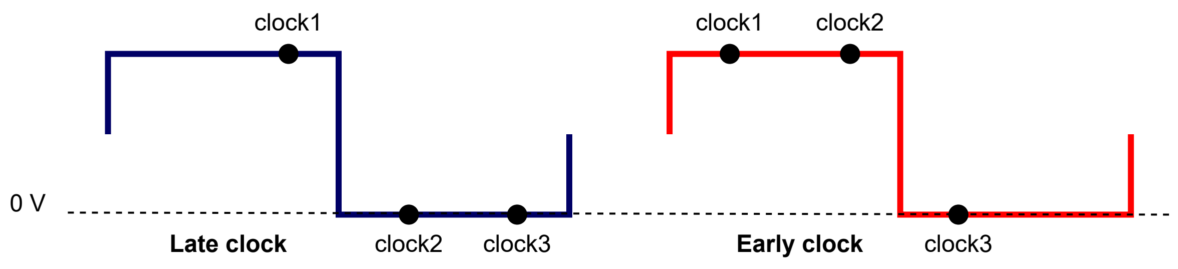

Figure 4.18: Sampling moments BBPD at each bit

To make the BBPD operate with a clock of 2.5GHz, the BBPD must be able to detect different bit periods, which immediately causes the BBPD to be four times larger. Another issue operating at the quarter rate of the bit rate is that 8 phases are needed because, for the BBPD discussed in 2.7, two phases are required for phase detection. Converting this to quarter rate operation requires eight phases.

To understand why 8 phases are needed and to have a better understanding of the BBPD the moments of sampling in the BBPD have to be studied. In figure 4.18 can be seen when the bit is getting sampled in case of a data edge with a late clock(blue signal) and an early clock (red signal). When comparing clocks 1 and 3 with each other these will always be different in the case of a data edge because the phase difference between these two will be a one-bit period of the data rate. This is how the BBPD detects whether there is a data edge or not.

To see if the clock is late or early from the middle of the eye a phase which is half a bit period away is needed from the sampling moment. In figure 4.18 this is done by clock 2. In the case the clock is late, clock 2's sampled signal will be the same as the sampled data with clock 3 and in case the clock is early they will be different.

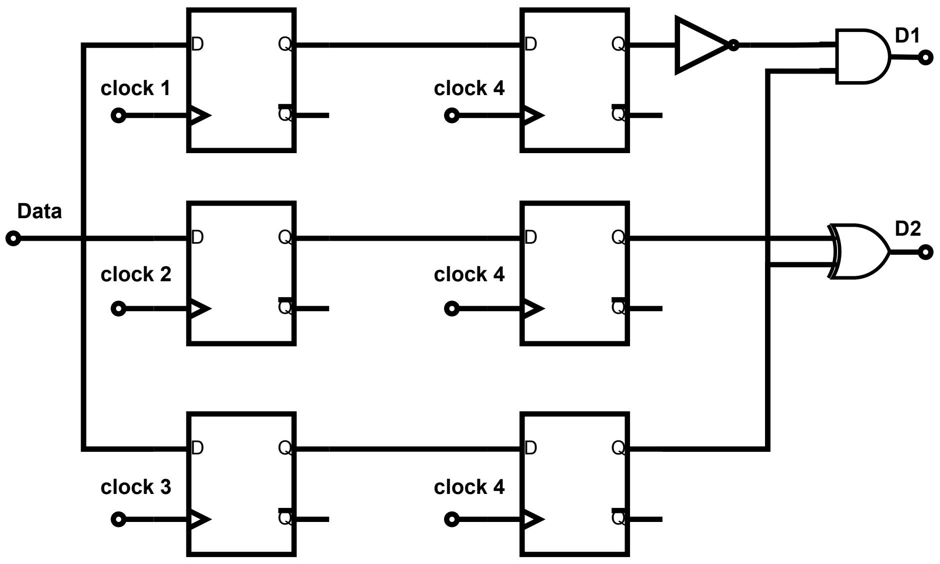

Figure 4.19: Quarter rate BBPD

In figure 4.19 the BBPD used can be seen. This BBPD for the D1 output instead of doing a XOR operation like the usual BBPD in figure 2.7 there is an AND operation. This is so that this BBPD only counts for rising edges in the data. This is done because the TDC has only been tested to work for rising edges.

When the clock is late the counter needs to count down. which means that in this CDR the D2 will be the UD control signal for the counter. This will be generated by doing an XOR operation between clock 2 and clock 3 because when the clock is late the output will be 0 thus the counter will count down and when the clock is early D2 will be high making the counter count up.

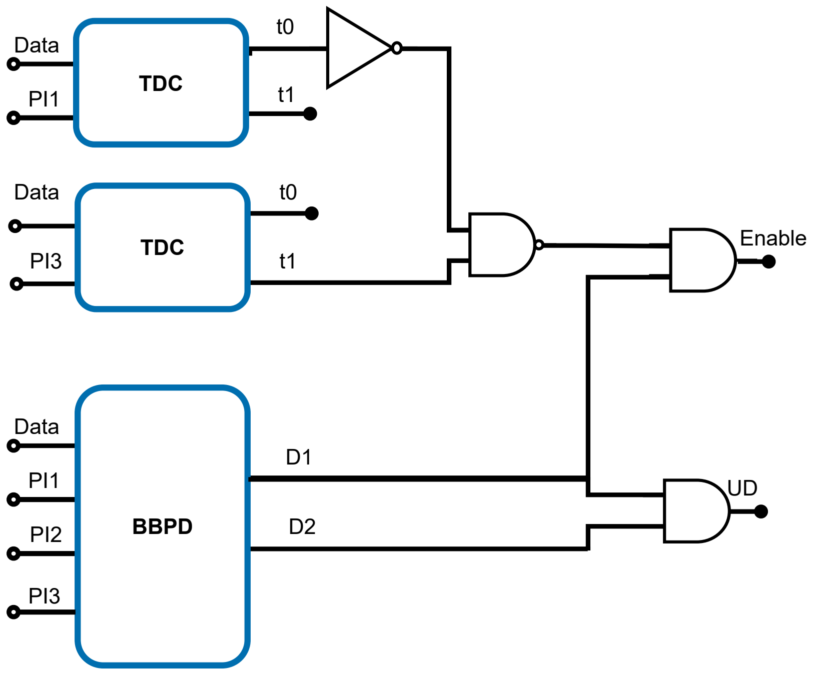

Figure 4.20: Control of the clock

The generation of the control signals for the counter is depicted in Figure 4.20. The figure shows that the enable signal is generated by utilizing the BBPD D1 signal along with the combined outputs of the TDC. This configuration ensures that the CDR only transitions to the next phase when an edge is detected by the BBPD and the TDC does not detect the lock condition. In all other cases, the counter remains locked in place.

Regarding the UP/DOWN (UD) signal, the BBPD combines the D1 and D2 outputs, resulting in the U/D signal changing to count up only when an edge is detected. This mechanism ensures that the counter's counting direction is altered appropriately based on the detected edges.

Due to the quarter-rate design and the CDR's operation exclusively at rising edges of the data, a maximum of two out of the four TDC and BBPD combinations propagate to the output. To make sure there is proper control of the counter, these output signals need to be combined into a single signal. The four enable signals are combined using an OR operation, resulting in a combined enable signal that controls the counter. Similarly, the four U/D signals are combined using an OR combination to generate a unified U/D signal. This approach is adopted because both the enable and U/D signals increment only when an edge is detected. Therefore, if there is only one data edge, one of the four outputs will contain the required control information. And in the case there are two edges as mentioned earlier t the operation will not change

An important issue that arose from this design is the long delays in the TDC, resulting in an input-to-output delay of approximately 250ps. This delay can potentially cause complications when operating the digital control at 2.GHz. The synchronization between the TDC and the BBPD becomes critical, particularly when it aligns closely with the counter's up/down counting phase. This can impact the decision-making process in the subsequent phase. To address this problem, a counter operating at half the speed is implemented, ensuring that this synchronization issue does not occur.

4.3. Phase interpolator

Phase interpolation is a process where two phases are taken and interpolated to generate an in-between phase. Phase interpolation is implemented in two different manners. There is an inverter-based interpolation, and there is differential pair phase interpolation.

The simplest form of an inverter-based inverter is shown in Figure 4.29. Let's assume the phase interpolated inputs are clock-1 and clock-2 ports. If both inverter strengths are the same and the input slope is not too steep, a phase between clock-1 and clock-2 is generated at the output. This function is accomplished because when clock-1 becomes high before clock-2, the two inverters operate against each other. This arrangement enables the output signal not to become high immediately and takes a bit longer to generate the phase between the two. When the strength of the two inverters is the same, the phase in between will be generated with a phase resolution of half of their phase difference. However, if more phases are required, the strength of the inverters can be changed to generate a finer resolution of phases. The disadvantage of using the inverter-based PI is that PVT variation negatively impacts the accuracy of this PI. The inverter-based PI is an excellent low-power and compact phase interpolation alternative.

There are various methods to realize an inverter-based PI, including a chain of matched inverter-based PI or a chain of weighted inverter-based PI.

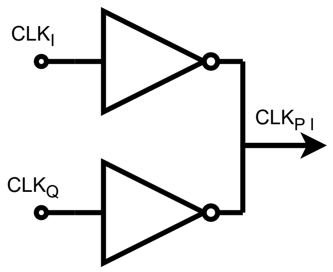

Figure 4.21: simple inverter based PI

The differential pair PI is generally employed in the PI-based CDR due to its more accurate phase interpolation when scaling the number of phases up. Figure 4.23 depicts an example of a differential pair phase interpolater. The main idea behind this phase interpolater is that the different input clock signals will charge the output node with different currents depending on the control voltage. Thus, when the two currents of the branches are equal, the PI signal will be in the middle of clock-1 and clock-2.

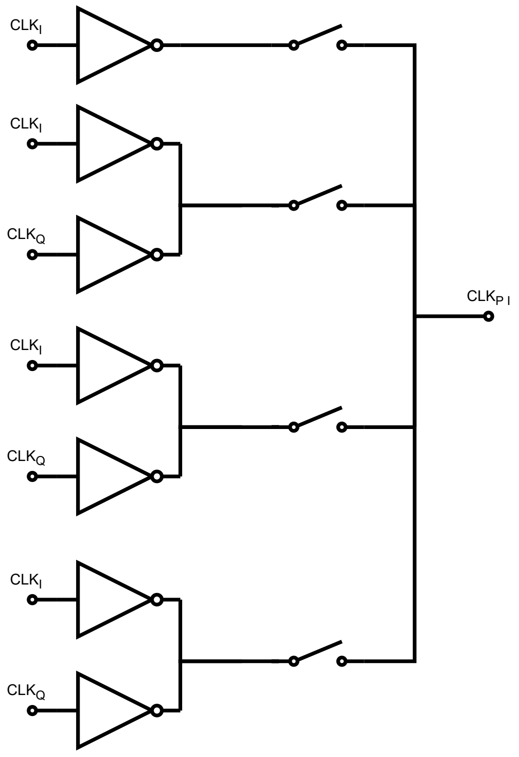

Figure 4.22: Straightforward 2-bit inverter based PI

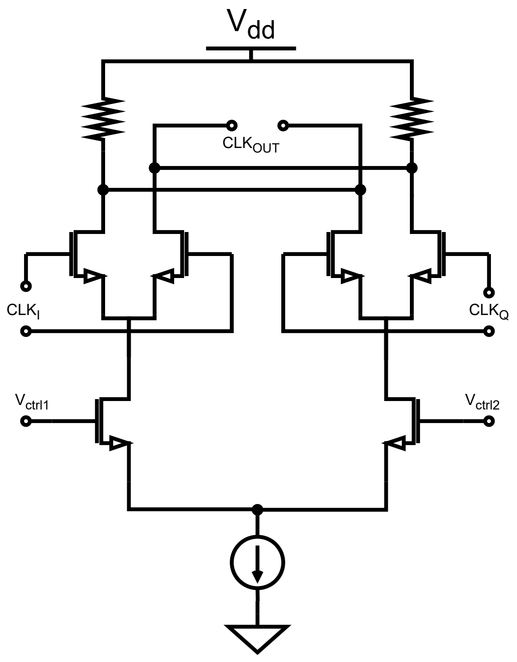

Figure 4.23: Differential pair phase interpolator

Because the phases have been reduced significantly from a usual PI-based CDR, instead of designing a differential pair PI an inverter-based PI is selected to decrease the power consumption and size of the PI and complete CDR. The implementation of this 2-bit is slightly different from the straightforward design of a two-bit PI shown in Figure 4.22. In this context, there are four different inverter-based PI where the inverter strength is adjusted to generate the four desired phases. More inverters are being used in this work, which is more than the required phases. To decrease the number of inverters needed, one can only use four inverters to generate a 2-bit PI [22], as shown in Figure 4.29.

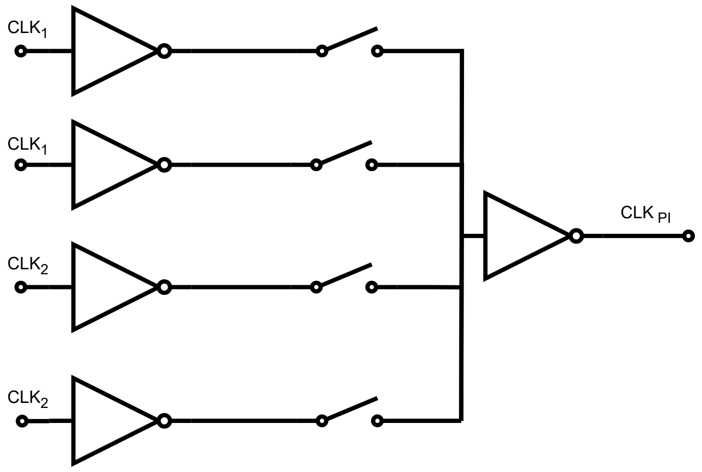

Figure 4.24: 2-bit inverter based PI ([22])

All inverters are connected to the output, of which one DLL clock signal is connected to clock one input, and the other immediate DLL clock signal is connected to clock two input. By combining these different strength inverters, multiple phases are created. For example, when both clock-one ports of the inverter are connected to the output, the earliest signal will be generated. But if an inverter of clock one and an inverter of clock two is connected, then a delayed clock phase will be generated at the output. This PI has been implemented in the current CDR design, of which the four phases generated are shown in Figure **??**.

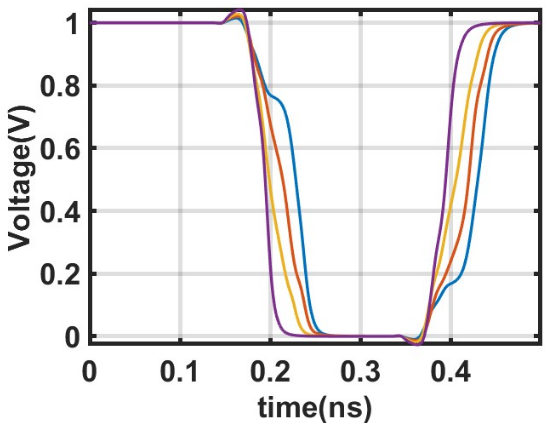

Figure 4.25: PI signal before the inverter

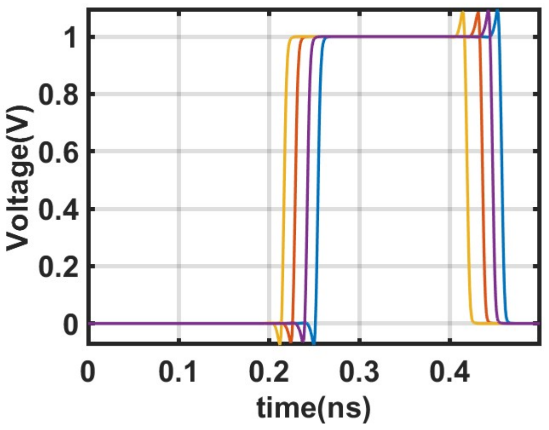

Figure 4.26: Phase interpolated signal

To further show the performance of this PI, all 32 phases have been simulated by using the previously designed DLL and feeding its output phases to the phase interpolator. This process resulted in a relatively accurate phase generation. The combined operation of a DLL and PI, whose phases are chosen to sample the data, is similar to a digital-to-phase circuit operation. To test this circuit, its INL and DNL of all the generated phases have been measured, shown in Figures4.28 and 4.27. This simulation shows DNL is approximately 0.1 LSB(LSB = 12.5ps), leading to a PI error of between 1 to 2 ps. There is one large spike in the DNL and INL due to the offset error in the DLL which will make one of the phases generated deviate by about 3 to 4 ps.

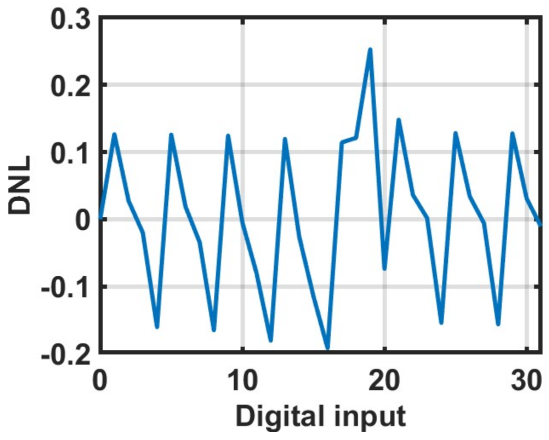

Figure 4.27: PI signal before the inverter

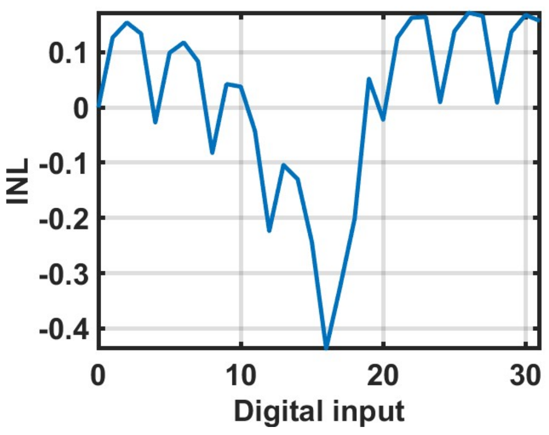

Figure 4.28: Phase interpolated signal

The following figure 4.29 is a polar plot of all phases that can be generated with the combination of the PI and the DLL. In the polar plot in the center, you can see all the phases rotate till they get to their locked state. while with the naked eye, the errors mentioned in the INL and DNL plot are not easily observed.

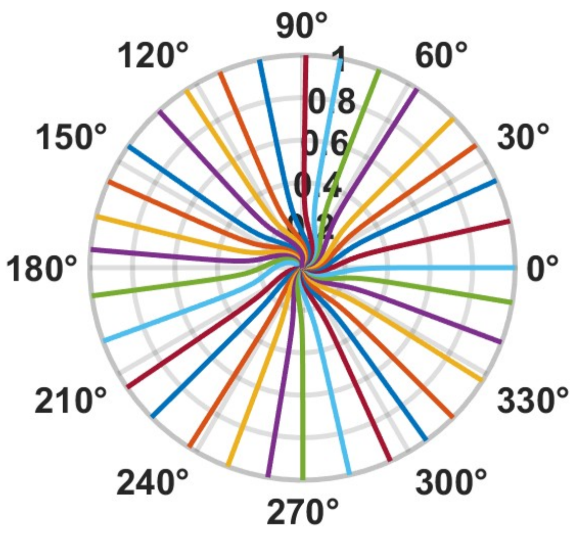

Figure 4.29: All phases generated with the DLL and PI

4.4. Results CDR

Multiple simulations have been performed to validate the CDR design. Unfortunately, the Jitter generation and JTOL simulations were not run due to insufficient time and needing more experience with the tools at my disposal. Instead, the transient simulation of the CDR has been run in various conditions to demonstrate the results and operation of the CDR.

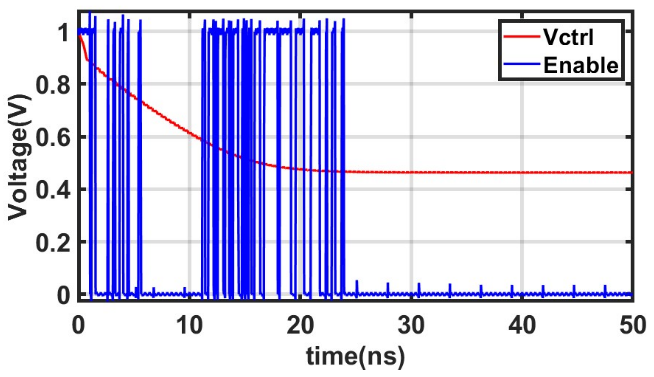

Figure 4.30: Counter enable and DLL control voltage

The first simulation to see if the CDR can make proper clock recovery is to see if the CDR can lock to a phase that is as close as possible to the middle of the data. Thus, This procedure can be accomplished by the clock-enabling signal generated by the TDC and BBPD. When the CDR enables the clock stop-counting signal, the chosen phases should be as close as possible to the middle of the data eye.

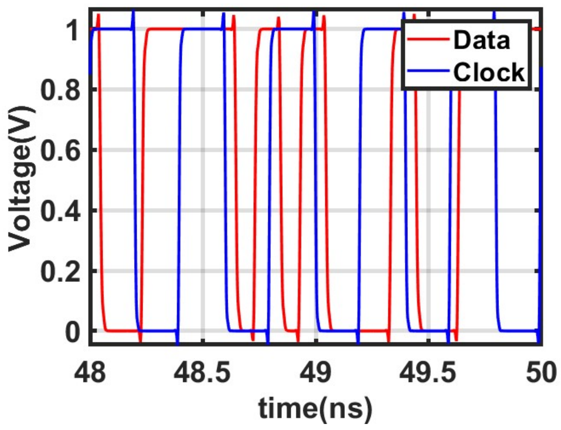

Figure 4.31: First clock phase and data signal

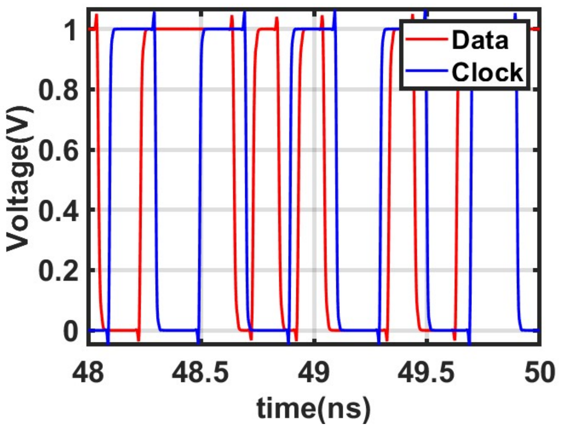

Figure 4.32: Second clock phase and data signal

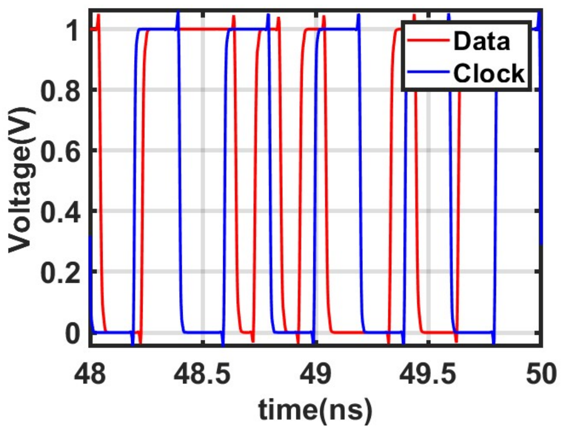

Figure 4.33: Third clock phase and data signal

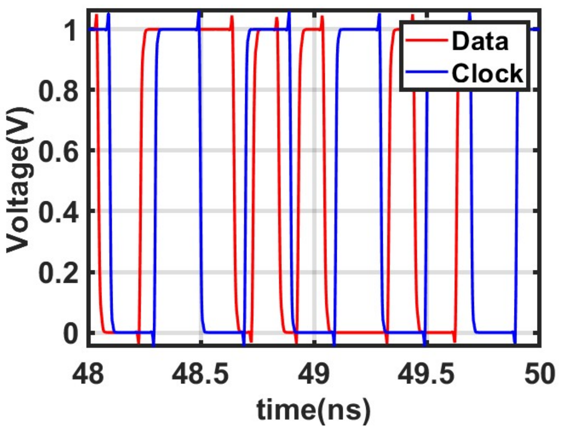

Figure 4.34: Fourth clock phase and data signal

Now from the previous results can thus be seen that the phases chosen by the CDR indeed are close to the middle of a bit period. To further verify its conditions the CDR has been tested while filtering the input data to replicate the low pass behavior of the transmission channel(figure). The corner frequency of this low pass filtering has been swept till the point of failure of the CDR has been found.

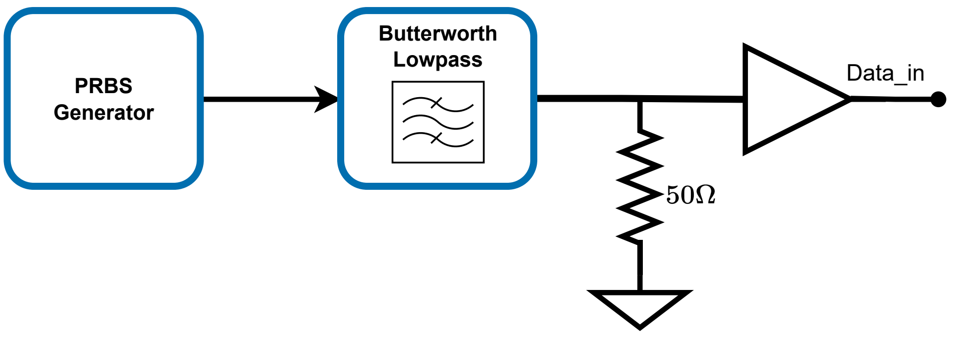

Figure 4.35: Eye diagram input data with LPF

Figure 4.35 displays the test conditions. The PRBS generates a random bit stream, which is then filtered by an ideal filter from the Cadence's "rflib." This filtered signal is terminated with a 50 Ohm resistance and a buffer, representing the buffered data to the CDR. Using this Testbench, we measured the CDR's operational range to have a minimum input data bandwidth of 8 GHz. Below this threshold, the CDR performance is affected by noticeable jitter and ISI induced by the' channel.'

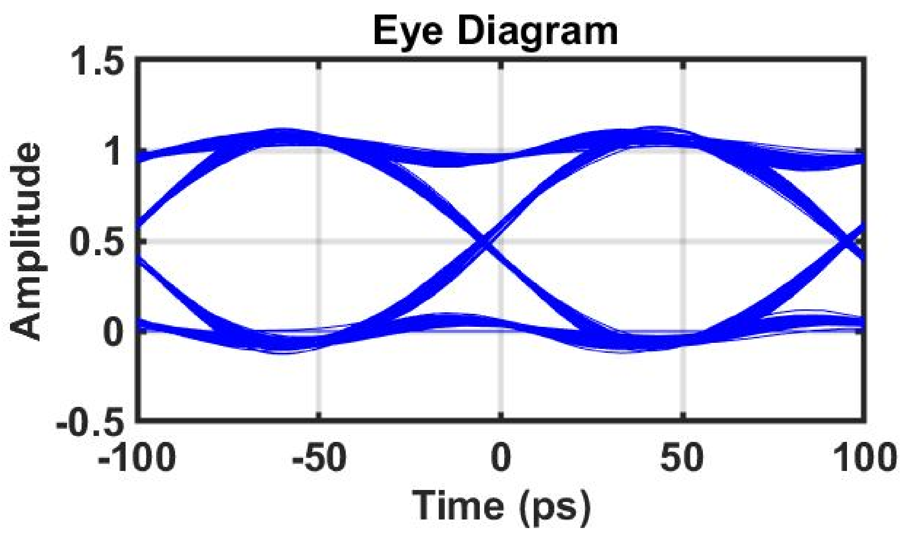

Figure 4.36: Input data eye diagram: 8 GHz bandwidth (10Gb/s)

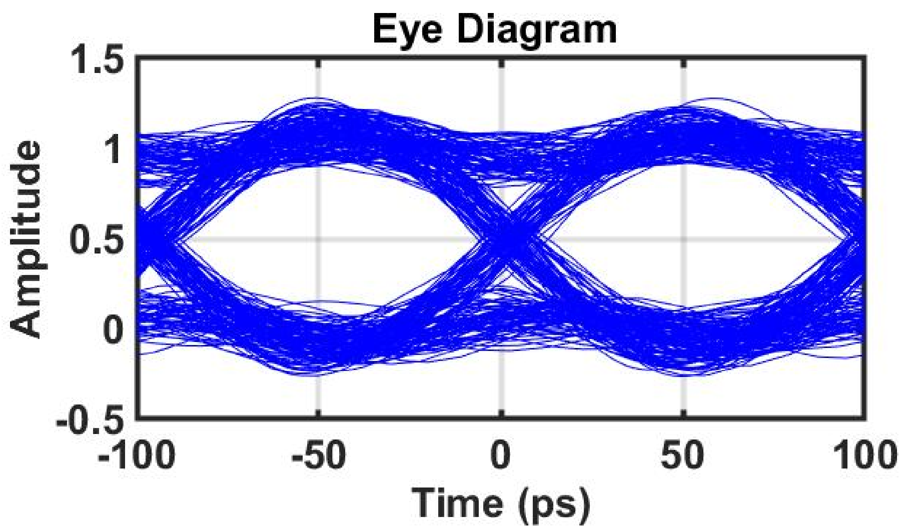

Figure 4.37: Input data eye diagram: 7 GHz bandwidth (10Gb/s)

The minimum bandwidth at which the CDR experiences excessive errors is 8 GHz. Comparing the eye diagram of an 8 GHz bandwidth signal (Figure 4.36) with the eye diagram of a 7 GHz bandwidth signal (Figure 4.37), it becomes apparent that the jitter in the input data significantly increases when the bandwidth drops below 8 GHz. Consequently, the CDR cannot cope with this increased jitter, resulting in substantial error generation.

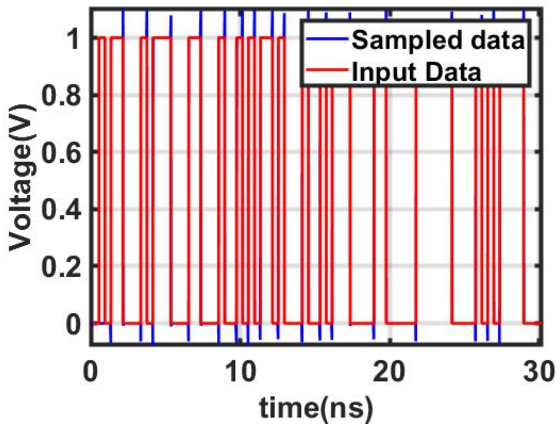

Figure 4.38: D1 and its input data

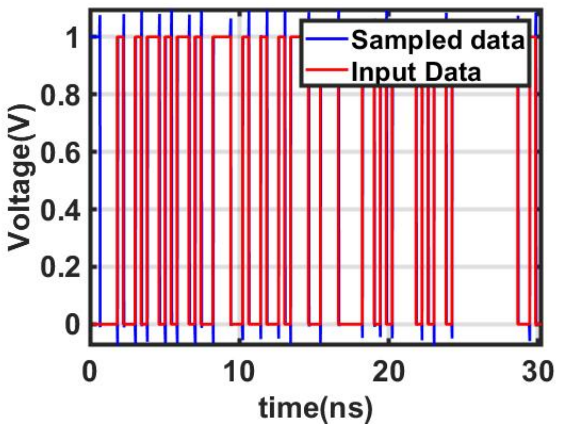

Figure 4.39: D2 and its input data

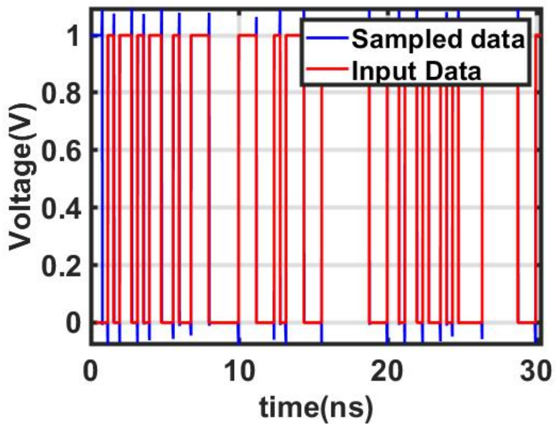

Figure 4.40: D3 and its input data

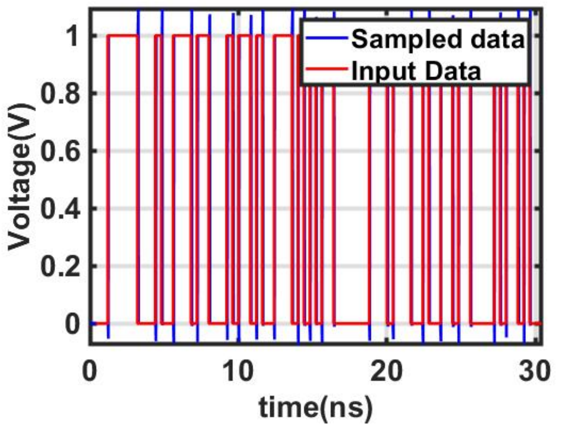

Figure 4.41: D4 and its input data

When the CDR operates with an 8GHz input signal, as exhibited in Figures 4.38 to 4.39, there are no errors when sampling the data in this range of 30 ns after the CDR is locked. The range of the CDR has been tested as well. Due to the DLL's limited locking range and how the TDC works, the data range has been limited to and tested from 8.5GB/s to 11 Gb/s for which the simulations can be seen in the appendix.

Table 4.2: Table of comparison

Reference	This Work	[25]	[21]	[13]	[9]	[12]	[11]	[26]
Data Rate(Gb/s)	10	1-16	5	0.65-8	1-16	20	10.8	12.5
Technology(nm)	40	65	90	65	130	65	110	90
Power Supply	1 V	1.2 V	1.3/1 V	1.2 V	1.5 V	1.2/1.5 V	1.5 V	1.2V
Power efficiency (mW/Gb/s/lane)	1.13	5.47	2.62	11.1	10	8.46	20.4	6.72

Through comprehensive transient simulations, the CDR operation was thoroughly evaluated, with individual component testing followed by successful recovery of all input data at the output. Notably, the CDR achieved this with a low power consumption of 11mW and generated a jitter of 5/14.1 ps (rms/pp). The dual-loop CDR design exhibits excellent power efficiency compared to the CDRs listed in Table 4.2. This outcome validates the effectiveness of the implemented measures to minimize power consumption.

5 Conclusion

In conclusion, this thesis has presented the design and analysis of a quarter-rate Dual Loop Phase-Interpolator Clock and Data Recovery (CDR) circuit with a Delay-Locked Loop (DLL) for efficient data recovery in high-speed communication systems. This design and analysis of the dual loop CDR were divided into two main chapters (Chapters 3 and 4).

The first chapter gives a general introduction to CDR architectures and considerations. The fundamental operation of a PLL-CDR, DLL-based CDR, and PI-based CDR has been discussed. While also providing examples of how these architectures are exploited in state-of-the-art wireline receivers.

In the second chapter, the design and optimization of the DLL were explored. The optimization was performed by designing and carefully selecting the voltage-controlled delay elements, the phase frequency detector, and managing the current mismatch in the charge pump. Through analysis and simulations, the effectiveness of the proposed design was demonstrated, showcasing its ability to achieve accurate phase alignment with an offset of 2ps at the operating frequency of 2.5GHz.

The third chapter discusses the design and evaluation of the CDR loop. The optimization of the CDR loop included the choice of the number of phases. The number of total phases generated has been chosen to be 32. These phases have been generated by interpolating the DLL's output designed earlier. Using the phase interpolated phase and the incoming data, the CDR's BBPD and TDC decide to go to the next phase or stay at the current phases. This mechanism resulted in an efficient CDR circuit that operates at 1.14mW/Gb/s.

Overall, the successful implementation and evaluation of the quarter-rate Dual Loop Phase Interpolator CDR circuit, as presented in this thesis, holds promise for enhancing the performance of high-speed communication systems, resulting in the deserialization of more than 10Gb/s random NRZ data.

5.1. Fututre Work

In order to build upon this project, several future projects can be recommended. Firstly, it is important to focus on the physical implementation of the Clock and Data Recovery (CDR) system. Developing a prototype will be crucial to validate its performance in practical applications.
Additionally, exploring the integration of the CDR within a complete wireline receiver system presents an interesting topic. Investigating the compatibility and performance of the CDR when incorporated into a broader receiver framework would provide valuable insights.

Another promising project involves extending the range of the digital delay-locked loop (DLL) to further enhance the range of the CDR. By increasing the DLL's capabilities, the CDR's overall range can be expanded, contributing to improved data recovery.
Furthermore, it would be worthwhile to assess the viability of the current CDR concept when applied to Pulse Amplitude Modulation-4 (PAM-4) modulation, with slight modifications. Evaluating how the underlying idea behind the CDR performs in conjunction with PAM-4 modulation can offer valuable insights into its adaptability and potential enhancements.
By focusing on these recommended future projects, we can advance the understanding and practical applications of CDR technology.

A
CDR bit range tests

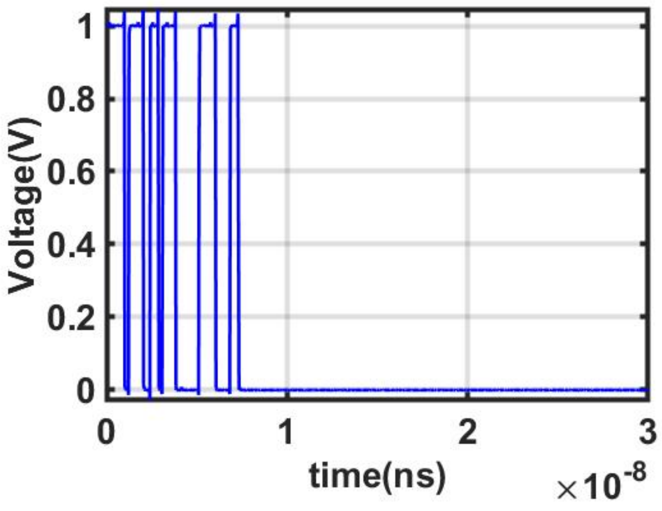

Figure A.1: settling of counter Enable at 8.5Gb/s

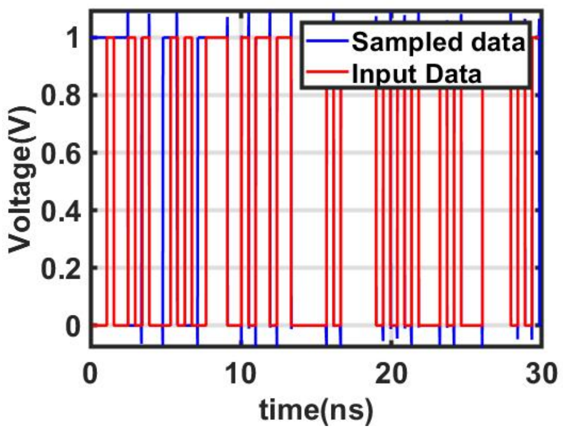

Figure A.2: D1 and its input data at 8.5Gb/s

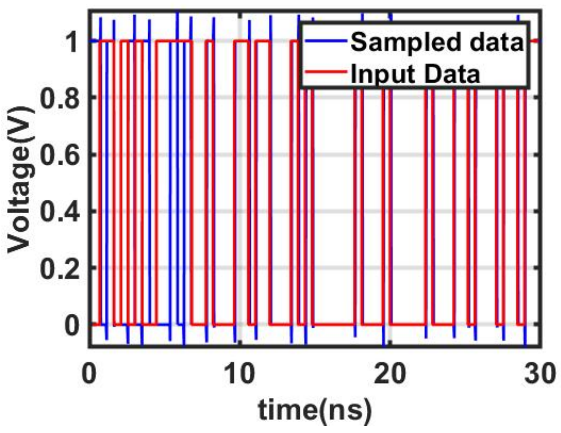

Figure A.3: D2 and its input data at 8.5Gb/s

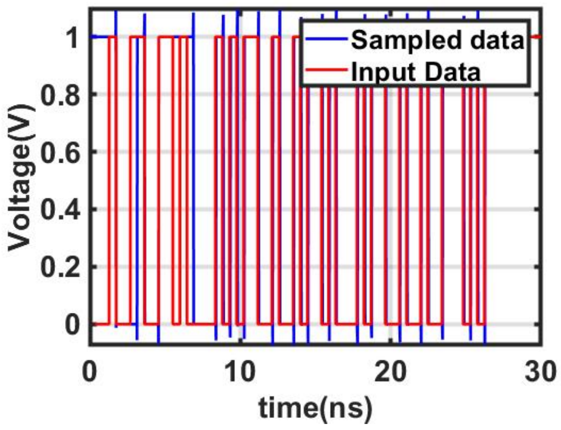

Figure A.4: D3 and its input data at 8.5Gb/s

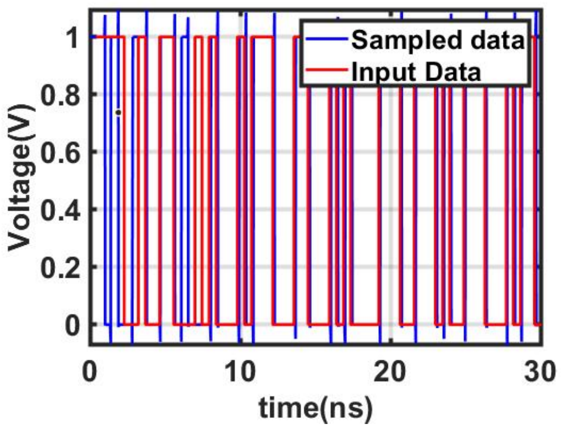

Figure A.5: D4 and its input data at 8.5Gb/s

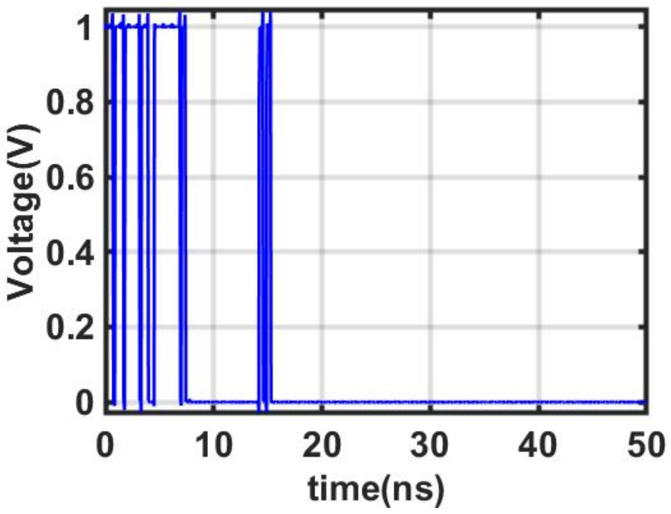

Figure A.6: settling of counter Enable at 11Gb/s

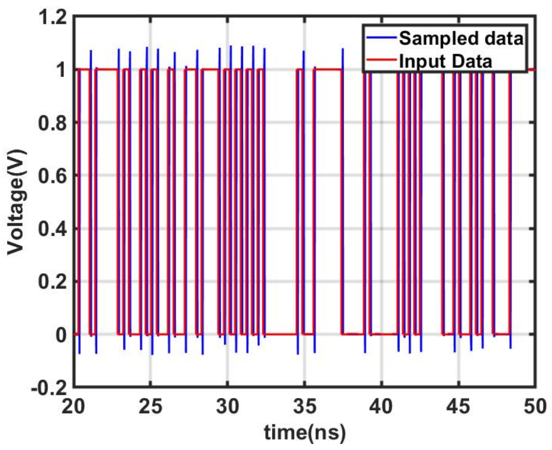

Figure A.7: D1 and its input data at 11Gb/s

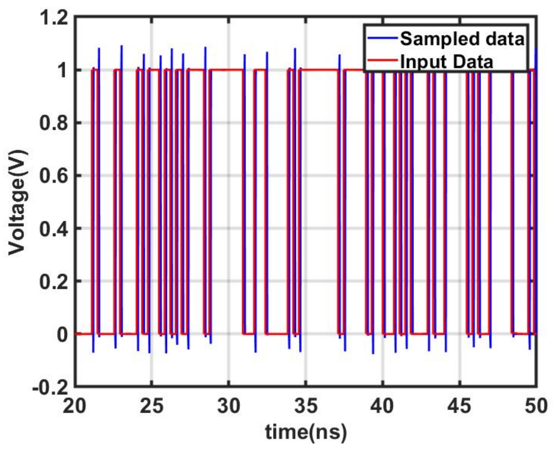

Figure A.8: D2 and its input data at 11Gb/s

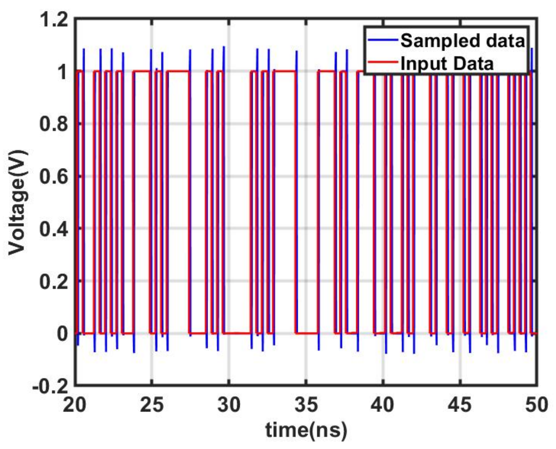

Figure A.9: D3 and its input data at 11Gb/s

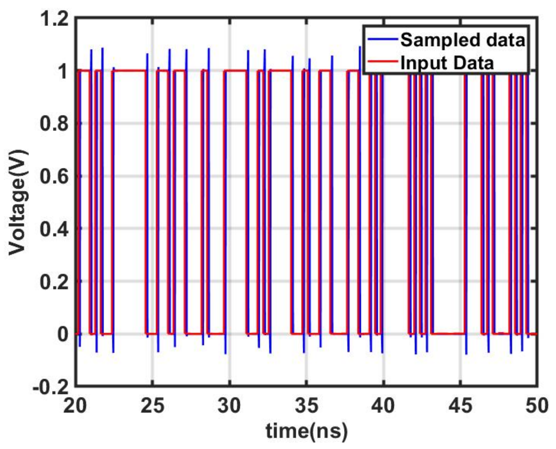

Figure A.10: D4 and its input data at 11Gb/s

B
Testing at the ff and ss process corner

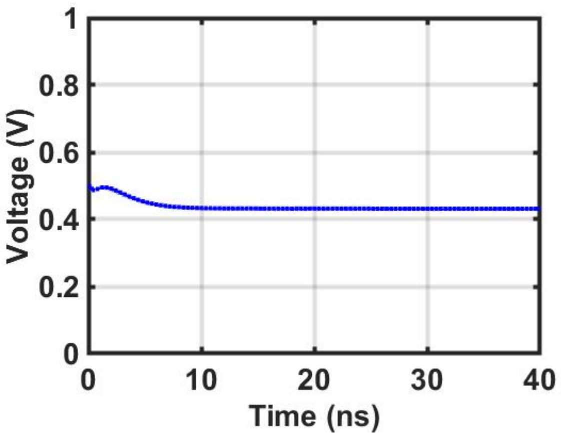

Figure B.1: DLL control voltage at ff corner

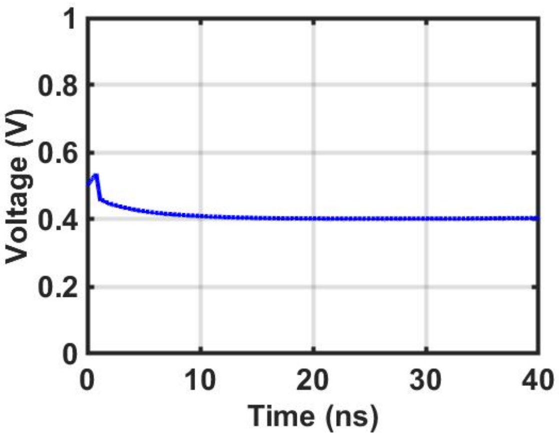

Figure B.2: DLL control voltage at ss corner

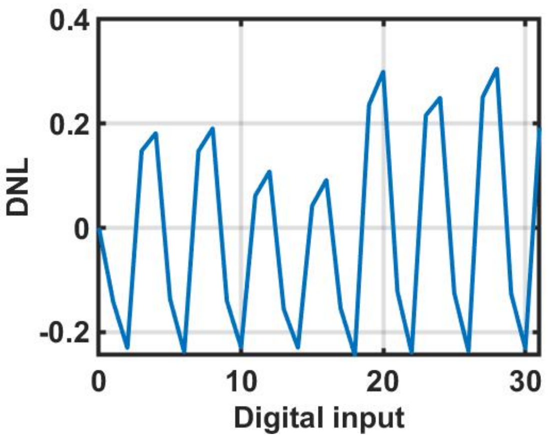

Figure B.3: DNL of phases generated at the ff corner

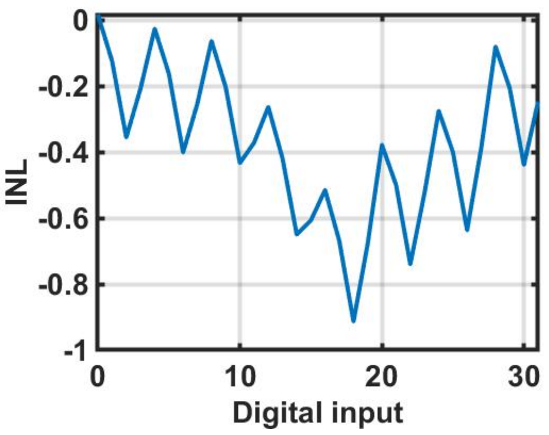

Figure B.4: INL of phases generated at the ff corner

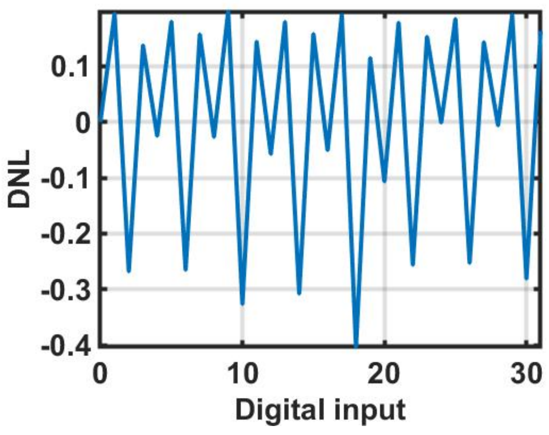

Figure B.5: DNL of phases generated at the ss corner

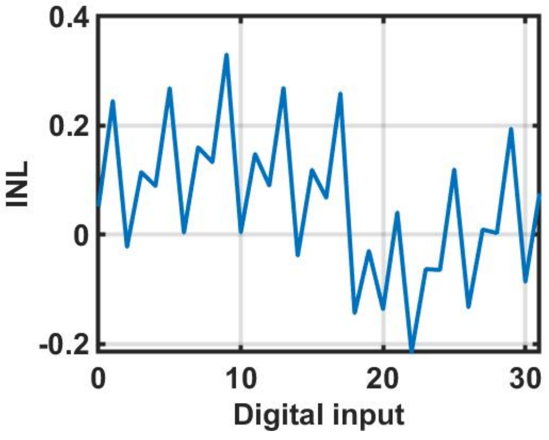

Figure B.6: INL of phases generated at the ss corner

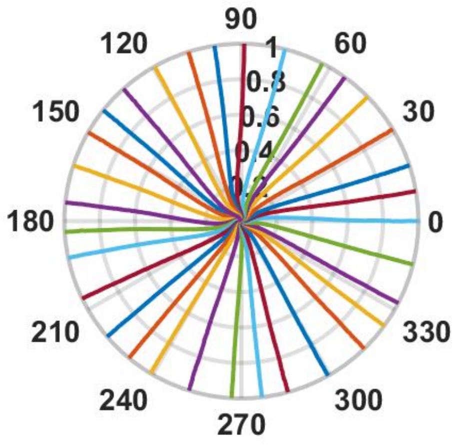
Figure B.7: Polar plot of phases generated at the ff corner

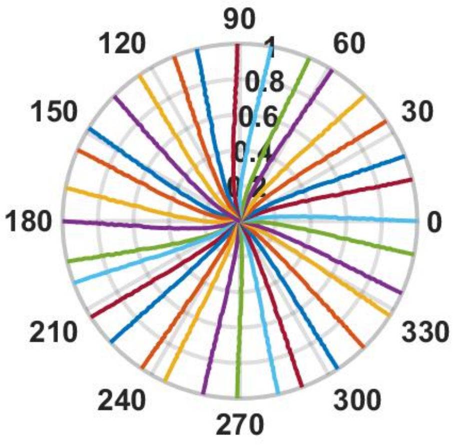
Figure B.8: Polar plot of phases generated at the ss corner

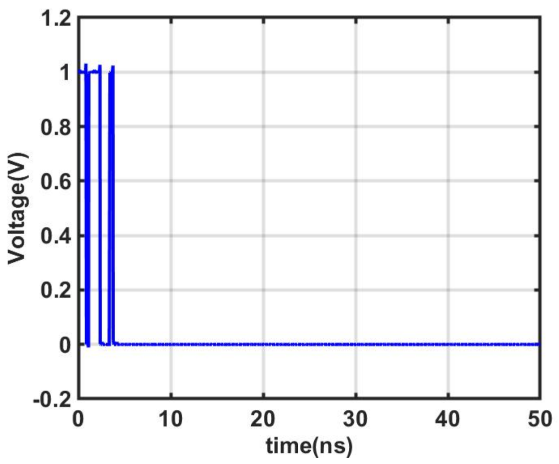

Figure B.9: settling of counter Enable at 11Gb/s

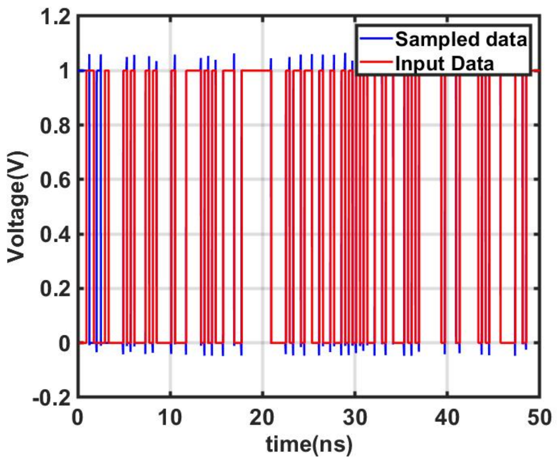

Figure B.10: D1 and its input data at the ff corner

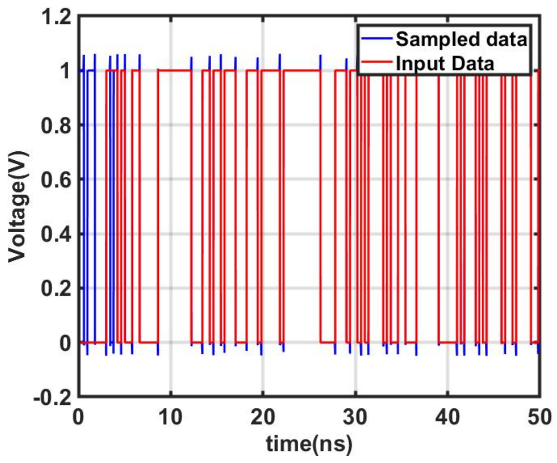

Figure B.11: D2 and its input data at the ff corner

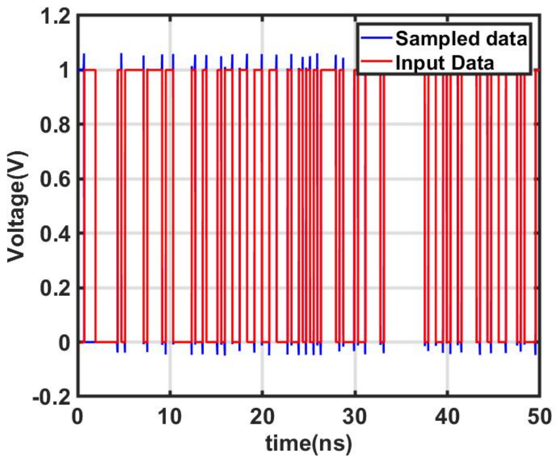

Figure B.12: D3 and its input data at the ff corner

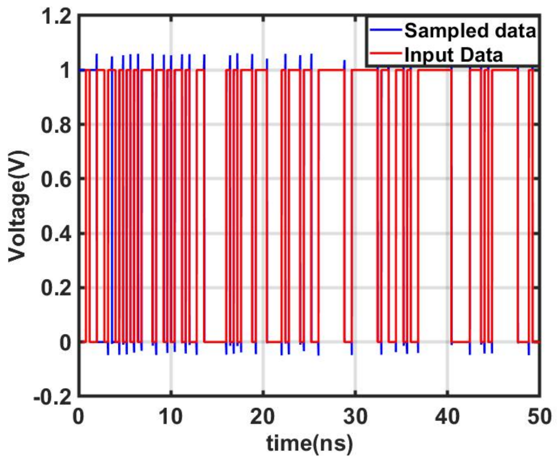

Figure B.13: D4 and its input data at the ff corner

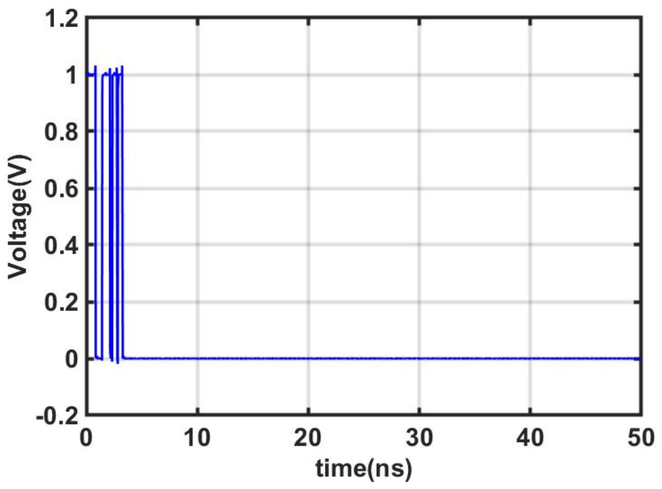

Figure B.14: settling of counter Enable at the ss corner

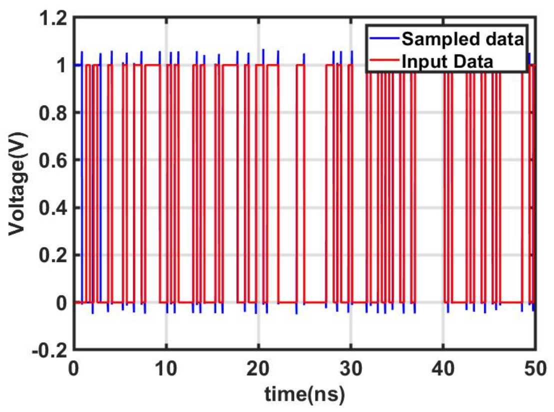

Figure B.15: D1 and its input data at the ss corner

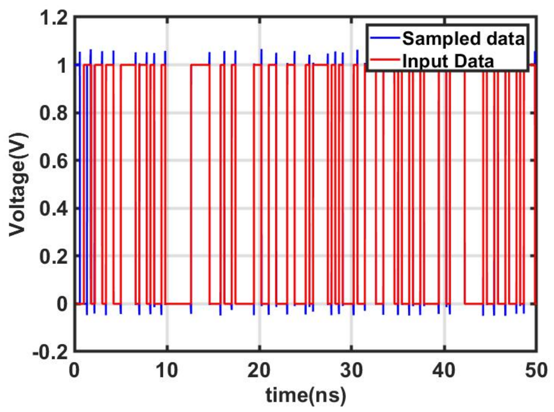

Figure B.16: D2 and its input data at the ss corner

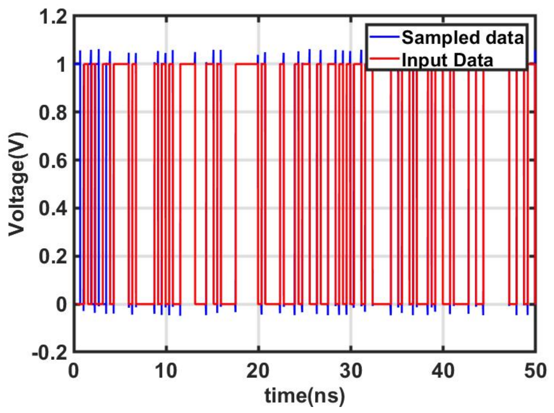

Figure B.17: D3 and its input data at the ss corner

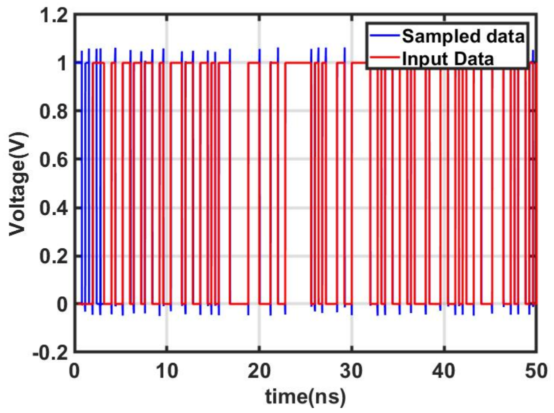

Figure B.18: D4 and its input data at the ss corner

Bibliography

[1] Phase interpolator-based cdr. Rambus Inc. Website. URL https://www.rambus.com/phase-interpolator-based-cdr/. Accessed: June 6, 2023.

[2] A. Amirkhany. Basics of clock and data recovery circuits: Exploring high-speed serial links. IEEE Solid-State Circuits Magazine, 12(1):25–38, 2020. doi: 10.1109/MSSC.2019.2939342.

[3] J. W. M. Bergmans. Digital Baseband Transmission and Recording. Springer Publishing Company, Incorporated, 1st edition, 2010. ISBN 1441951644.

[4] P. W. de Abreu Farias Neto, K. Hearne, I. Chlis, D. Carey, R. Casey, B. Griffin, F. S. F. Ngankem Ngankem, J. Hudner, K. Geary, M. Erett, A. Laraba, H. Eachempatti, J. W. Kim, H. Zhang, S. Asuncion, and Y. Frans. A 112–134-gb/s pam4 receiver using a 36-way dual-comparator ti-sar adc in 7-nm finfet. IEEE Solid-State Circuits Letters, 3:138–141, 2020. doi: 10.1109/LSSC.2020.3007580.

[5] Jun Feng. A 10 gbps wireline transceiver link. Master's thesis, TU Delft, Delft, August 2020.

[6] Jun Feng, Mohammadreza Beikmirza, Mohammadreza Mehrpoo, Leo C.N. de Vreede, and Morteza S. Alavi. A versatile and efficient 0.1-to-11 gb/s cml transmitter in 40-nm cmos. In 2021 18th International SoC Design Conference (ISOCC), pages 41–42, 2021. doi: 10.1109/ISOCC53507.2021.9613887.

[7] Jordan Lee Gauci, Edward Gatt, Owen Casha, Giacinto De Cataldo, Ivan Grech, and Joseph Micallef. Design of a quasi-linear rail-to-rail delay element with an extended programmable range. In 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pages 265–268, 2018. doi: 10.1109/ICECS.2018.8617855.

[8] M. M. Green. An overview on wireline communication systems for high-speed broadband communication. In Proceedings of Papers 5th European Conference on Circuits and Systems for Communications (ECCSC'10), pages 1–8, 2010.

[9] Chang-Lin Hsieh and Shen-Iuan Liu. A 1–16-gb/s wide-range clock/data recovery circuit with a bidirectional frequency detector. IEEE Transactions on Circuits and Systems II: Express Briefs, 58(8):487–491, 2011. doi: 10.1109/TCSII.2011.2158719.

[10] Ming-ta Hsieh and Gerald E. Sobelman. Architectures for multi-gigabit wire-linked clock and data recovery. IEEE Circuits and Systems Magazine, 8(4):45–57, 2008. doi: 10.1109/MCAS.2008.930152.

[11] R. Kreienkamp, U. Langmann, C. Zimmermann, T. Aoyama, and H. Siedhoff. A 10-gb/s cmos clock and data recovery circuit with an analog phase interpolator. IEEE Journal of Solid-State Circuits, 40(3):736–743, 2005. doi: 10.1109/JSSC.2005.843624.

[12] Young-Ho Kwak, Yongtae Kim, Sewook Hwang, and Chulwoo Kim. A 20 gb/s clock and data recovery with a ping-pong delay line for unlimited phase shifting in 65 nm cmos process. IEEE Transactions on Circuits and Systems I: Regular Papers, 60(2):303–313, 2013. doi: 10.1109/TCSI.2012.2215781.

[13] Seon-Kyoo Lee, Young-Sang Kim, Hyunsoo Ha, Younghun Seo, Hong-June Park, and Jae-Yoon Sim. A 650mb/s-to-8gb/s referenceless cdr circuit with automatic acquisition of data rate. In 2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers, pages 184–185,185a, 2009. doi: 10.1109/ISSCC.2009.4977369.

[14] Won-Hyo Lee, Jun-Dong Cho, and Sung-Dae Lee. A high speed and low power phase-frequency detector and charge-pump. In Proceedings of the ASP-DAC '99 Asia and South Pacific Design Automation Conference 1999 (Cat. No.99EX198), pages 269–272 vol.1, 1999. doi: 10.1109/ASPDAC.1999.760011.

[15] Fangxu Lv, Xuqiang Zheng, Ziqiang Wang, Yajun He, Chun Zhang, Jianye Wang, Zhihua Wang, and Hanjun Jiang. Design of 80-gb/s pam4 wireline receiver in 65-nm cmos technology. In 2017 International Conference on Electron Devices and Solid-State Circuits (EDSSC), pages 1–2, 2017. doi: 10.1109/EDSSC.2017.8126514.

[16] M. Maymandi-Nejad and M. Sachdev. A monotonic digitally controlled delay element. IEEE Journal of Solid-State Circuits, 40(11):2212–2219, 2005. doi: 10.1109/JSSC.2005.857370.

[17] B. Razavi. Design of CMOS Phase-Locked Loops: From Circuit Level to Architecture Level. Cambridge University Press, 2020. ISBN 9781108494540. URL https://books.google.nl/books?id=Sm_CDwAAQBAJ.

[18] Behzad Razavi. Design of Integrated Circuits for Optical Communications. McGraw-Hill, Inc., USA, 1 edition, 2002. ISBN 0072822589.

[19] Behzad Razavi. High-Speed CMOS Circuits for Optical Receivers. Springer Science & Business Media, New York, NY, 2002. ISBN 978-0-306-47292-3.

[20] Hassan Rivandi, Sudeh Ebrahimi, and Mehdi Saberi. A low-power rail-to-rail input-range linear delay element circuit. AEU - International Journal of Electronics and Communications, 79: 26–32, 2017. ISSN 1434-8411. doi: https://doi.org/10.1016/j.aeue.2017.04.037. URL https://www.sciencedirect.com/science/article/pii/S1434841116304411.

[21] Guanghua Shu, Saurabh Saxena, Woo-Seok Choi, Mrunmay Talegaonkar, Rajesh Inti, Amr Elshazly, Brian Young, and Pavan Kumar Hanumolu. A reference-less clock and data recovery circuit using phase-rotating phase-locked loop. IEEE Journal of Solid-State Circuits, 49(4):1036–1047, 2014. doi: 10.1109/JSSC.2013.2296152.

[22] Andreas Tsimpos, George Souliotis, Andreas Demartinos, and Spiros Vlassis. All digital phase interpolator. In 2015 10th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS), pages 1–6, 2015. doi: 10.1109/DTIS.2015.7127354.

[23] Dengjie Wang, Ziqiang Wang, Hao Xu, Jiawei Wang, Zeliang Zhao, Chun Zhang, Zhihua Wang, and Hong Chen. A 56-gbps pam-4 wireline receiver with 4-tap direct dfe employing dynamic cml comparators in 65 nm cmos. IEEE Transactions on Circuits and Systems I: Regular Papers, 69(3):1027–1040, 2022. doi: 10.1109/TCSI.2021.3125355.

[24] Yi-Ming Wang and Jinn-Shyan Wang. An all-digital 50 In 2004 IEEE International Symposium on Circuits and Systems (ISCAS), volume 2, pages II–925, 2004. doi: 10.1109/ISCAS.2004.1329424.

[25] Guoying Wu, Deping Huang, Jingxiao Li, Ping Gui, Tianwei Liu, Shita Guo, Rui Wang, Yanli Fan, Sudipto Chakraborty, and Mark Morgan. A 1–16 gb/s all-digital clock and data recovery with a wideband high-linearity phase interpolator. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 24(7):2511–2520, 2016. doi: 10.1109/TVLSI.2015.2508045.

[26] Arash Zargaran-Yazd and Shahriar Mirabbasi. 12.5-gb/s full-rate cdr with wideband quadrature phase shifting in data path. IEEE Transactions on Circuits and Systems II: Express Briefs, 60(6):297–301, 2013. doi: 10.1109/TCSII.2013.2258255.