Hybrid Baud-Rate CDR and Clock Distribution Techniques for High-Speed 100Gbps Wireline Receiver

Yi-Hsuan Shih

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-165
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-165.html

August 15, 2025

Hybrid Baud-Rate CDR and Clock Distribution Techniques for High-Speed 100Gbps Wireline Receiver

By

Yi-Hsuan Shih

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Vladimir Stojanovic, Co-Chair
Professor Borivoje Nikolic, Co-Chair
Professor Martin White

Summer 2025

Hybrid Baud-Rate CDR and Clock Distribution Techniques for High-Speed 100Gbps Wireline Receiver

Abstract

Hybrid Baud-Rate CDR and Clock Distribution Techniques for High-Speed 100Gbps Wireline Receiver

by

Yi-Hsuan Shih

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Vladimir Stojanovic, Co-Chair

Professor Borivoje Nikolic, Co-Chair

With increasing demands in AI and MIMO systems, wireline link speeds have surged beyond 100Gbps, intensifying inter-symbol interference (ISI), and challenging traditional feedback-based equalization techniques like decision feedback equalizers (DFE) due to timing constraints. To overcome feedback loop limitations, feedforward MLSE was proposed, avoiding complex feedback but still introducing significant computational complexity.

This thesis presents a new hybrid clock data recovery algorithm (CDR) that leverages the complexity of the feedforward MLSE to improve the CDR locking performance and robustness to channel variations. Our approach combines the Mueller-Muller algorithm with a data-level maximization algorithm, achieving baud-rate CDR locking with reduced hardware overhead and steady-state stability without additional dithering.

An evaluation framework is developed to compare various CDR algorithms (Mueller-Muller, dLev maximization, and hybrid CDR) in terms of locking position and steady-state jitter by Markov chain analysis. The results indicate that hybrid CDR outperforms in locking accuracy and jitter resilience.

Our hybrid CDR, implemented in 16nm technology node, includes an innovative octature generator with an injection-locked oscillator for clock distribution. Verification using a digital SystemVerilog environment improves simulation accuracy and reduces overall verification time for the receiver system.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

I feel deeply fortunate to have had the opportunity to pursue my PhD at University of California, Berkeley and conduct research in Berkeley Wireless Research Center (BWRC). This has been a long and challenging journey, but I have been lucky to receive the guidance and support of exceptional faculty and talented peers.

First and foremost, I would like to express my sincere gratitude to my advisors, Professors Vladimir Stojanovic and Elad Alon. During the past few years, Professor Vladimir helped me learn to see the whole picture and think from a system perspective technically. I learned from him how to manage the research and tape-outs strategically, especially when the system is too large for a single individual. I am really grateful that he always motivated me and recognized the value of my work. He has been supportive not only in my research but also in the challenges that I face in life. I cannot thank Professor Vladimir enough for his warm support and understanding during my most difficult times. Professor Elad introduced me to the world of high-speed wireline links and guided me through the initial phase of exploring the field and defining my own research topics. He always provided guidance with his sharp intuition for circuits and profound technical understanding whenever I felt stuck.

I also thank Professors Borivoje Nikolic and Martin White for serving on my qualifying exam and dissertation committees. I am grateful for their invaluable feedback and guidance on my research. I greatly appreciate Professor Bora's technical advice, which significantly improved the quality of my dissertation.

I would like to extend my appreciation to Farhana Sheikh and Intel's University Shuttle Program for granting me the opportunity to implement my research work in advanced technology nodes.

During the PhD journey, I am grateful to have met and learned from many brilliant people. I would like to especially thank Paul Kown, Ayan Biswas, Wahid Rahman, Kunmo Kim for working together as a team during multiple tape-outs. I would also like to thank Zhongkai Wang, Bob Zhou, Zhaokai Liu, Kwanseo Park, Zhenghan Lin, Yikuan Chen, Yu-Chi Lin, Hesham Beshary, Rohit Braganza for their tape-out and testing assistance. My sincere thanks also go to Rebekah Zhao, Yue Dai, Meng Wei, Rozhan Rabbani, Bozhi Yin, Ruocheng Wang for their friendship that made my graduate life more colorful. A special thanks to the BWRC staff and EECS department staff for their support.

Last but not least, I owe my deepest thanks to my family for their unwavering love, patience, and encouragement throughout my life. Their belief in me has been the foundation that supports me through the tough PhD life. Without them, I could not have navigated the challenges in my journey.

Chapter 1

Introduction

1.1 Background

Today, data are being created and transmitted on a massive level as a huge wave of connectivity everywhere through a multitude of modern devices such as wearables, smart home appliances, connected cars, medical devices, fitness bands, etc. Furthermore, the surge of artificial intelligence (AI) requires high computational power, which requires integration and cooperation between multiple chips.

Among these explosive growing connections, high-speed links have been widely used to provide wide-bandwidth wireline connectivity on various scales, ranging from on-chip to chip-to-chip and system-to-system interconnects. The data rate of the common I/O standards is shown in Figure. 1.1 reflects the constantly increasing demand for a higher data rate in recent years. Most recent publications in the wireline domain focus on data rates exceeding 100Gbps, reflecting this shift. However, achieving these rates presents challenges. Data rate capabilities are restricted not only by the process node but also by packaging constraints, which can limit performance and reliability. Our research group is addressing these challenges by targeting 100Gbps, 128Gbps, and 160Gbps designs by using a 16-nm process node to meet this demand.

In contrast to this rapid increase in bandwidth, the power consumption becomes the bottleneck to overcome due to the severe thermal restrictions of the entire system. In the vision for the future technology stated in the IRDS roadmap [22], it is expected that the power consumption of communication and routing will be reduced by a factor of 10 while the data will increase by a factor of more than 10. Therefore, the energy efficiency of high-speed links should be constantly improved as the data rate increases. While the bandwidth and power constraints are very challenging by themselves, to make matters worse, the scaling of the CMOS process is slowing down and the performance gain from scaling is not as dramatic as in the past. Therefore, innovative design techniques that push the limits of electrical signaling are essential to meet those bandwidth and energy efficiency goals in advanced CMOS processes.

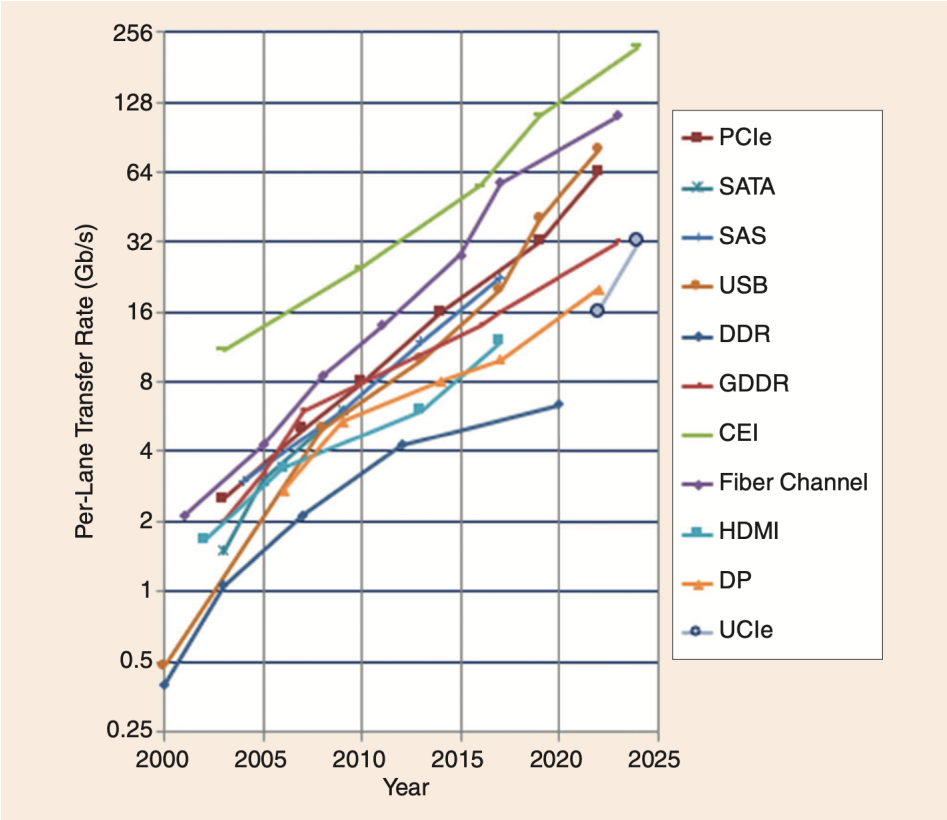


Figure 1.1: Per-lane data rate vs. year for a variety of common I/O standards[34].

The chiplet is a way to implement a system-in-package (SIP) [46] design, using a modular approach that partitions functional blocks into different dies and connects them with a standardized interface, i.e., die-to-die communication. Shown in Figure. 1.1, Universal Chiplet Interconnect Express (UCIe) [45][36][44], is an open industry standard developed to establish a ubiquitous interconnect at the package level and covers the die-to-die I/O physical layer. Instead of having a large monolithic die that includes all functions of the system, we can use several small dies that perform different functions and connect them together in the package. Shown in Figure. 1.3, the die-to-die communication includes the transceiver (TRX) on each die and the connection (channel) between the dies. In this way, the area of the individual die is reduced so that the yield can be improved with similar defect rates, ultimately lowering the cost (Figure. 1.4).

The AMD EPYC CPU[15] for the data center and the Apple M2[17] ultra for the Mac Studio laptop are the real-world example of the chiplet implementation (Figure. 1.5). 192 cores and 384 threads are enabled by 16 CPU dies and 1 IO die connected with Infinity fabric packaging in Turin CPUs (5^{th} Gen EPYC). The Apple M2 Ultra SoC is built with an UltraFusion silicon interposer packaging architecture to connect the die of two M2 Max

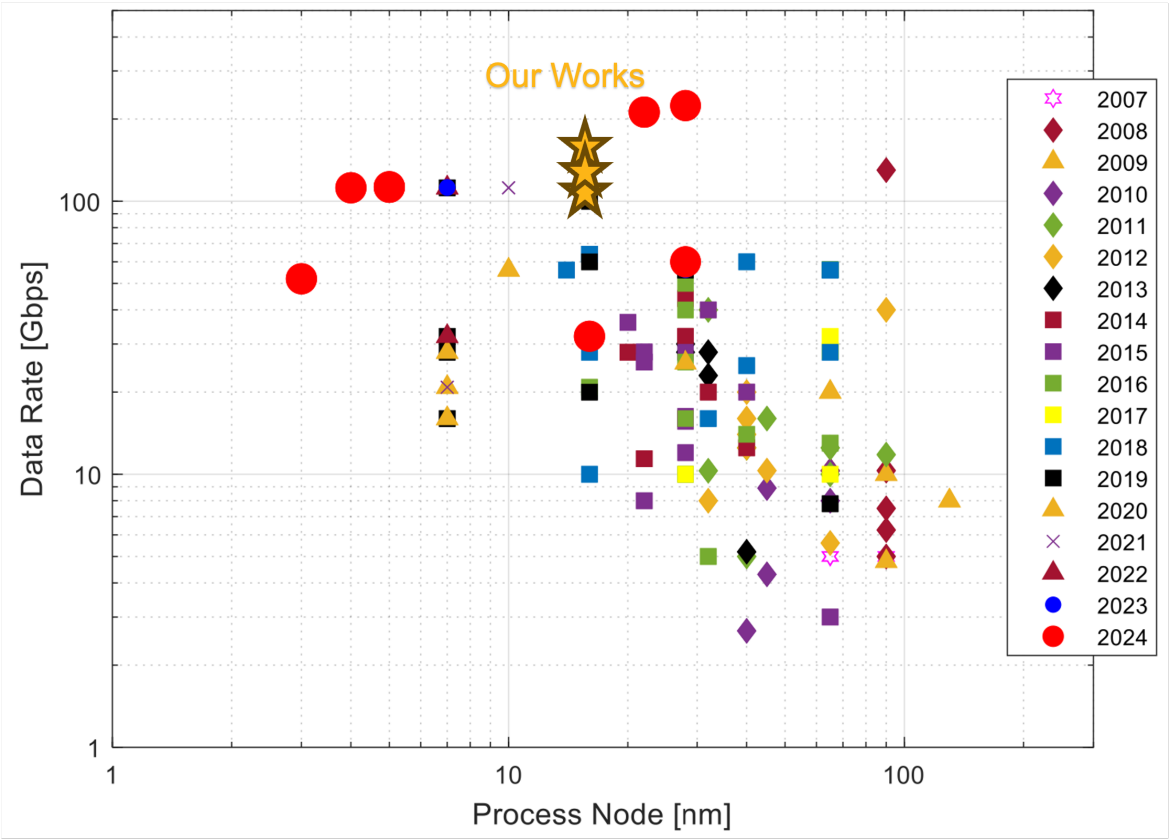


Figure 1.2: Wireline data-rate trend[34].

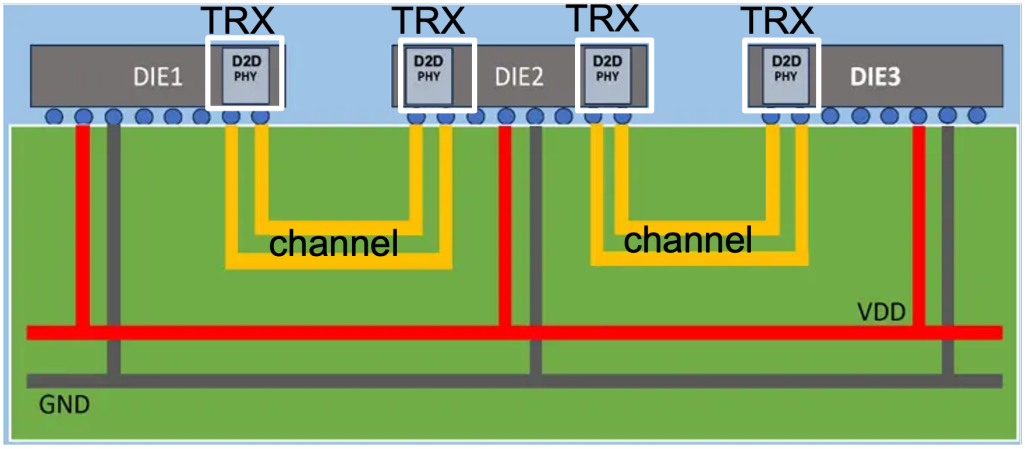


Figure 1.3: Die-to-die communication [25].

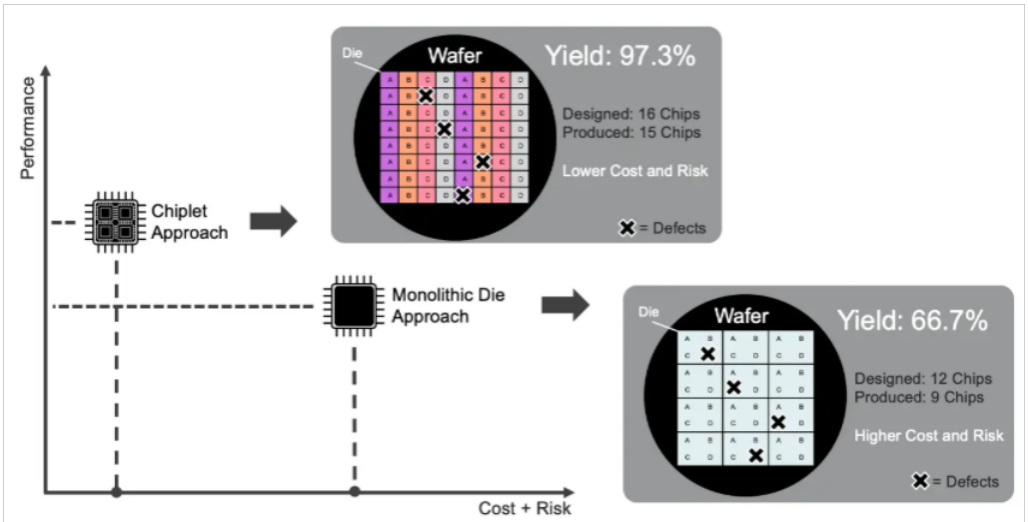
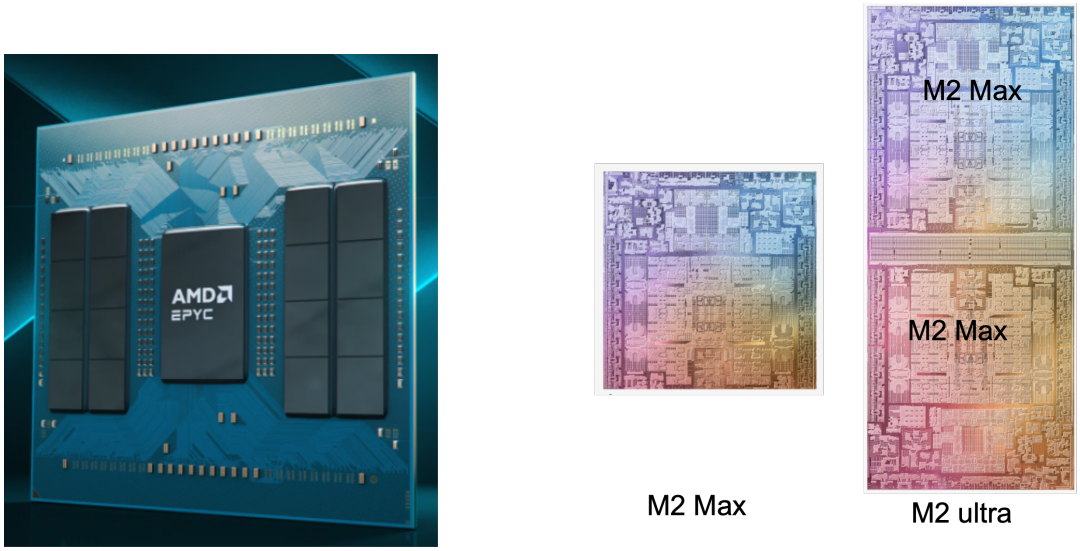


Figure 1.4: Monolithic die vs. chiplet [25].

chips to achieve twice the performance of one M2 Max chip.



(a) AMD Truin CPU [15].

(b) Apple M2 MAX and M2 Ultra[17].

Figure 1.5: Examples of chiplet design

Another example is high-performance computing (HPC). It offers reduced power consumption by connecting the compute node via 112GT/s, 1mm reach, and 1pJ/bit XSR standard electrical link within the electro-optic engine for HPC [22].

Wireline Transceiver

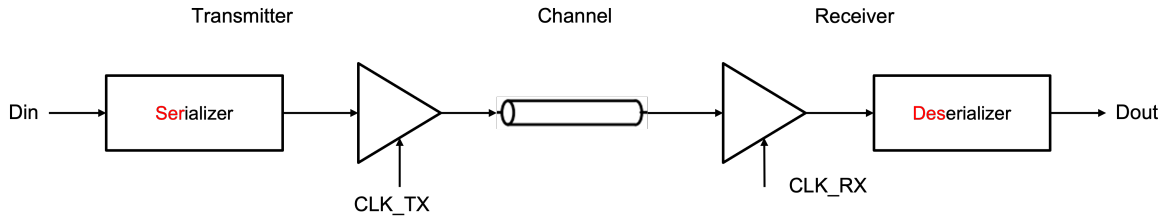


Figure 1.6: High-speed link block diagram.

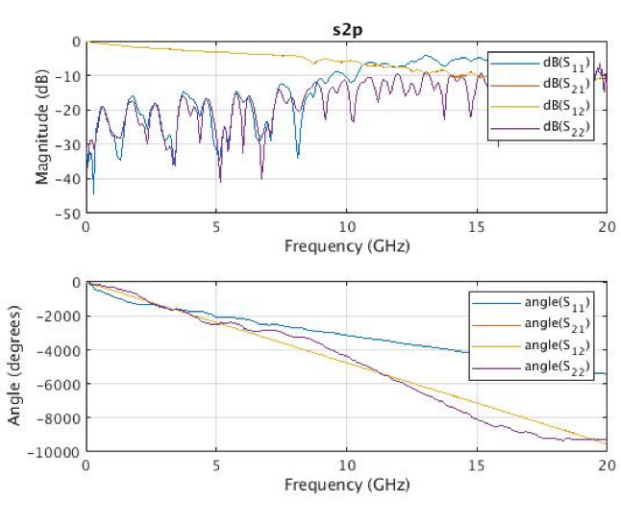
A typical high-speed link is shown in Figure 1.6, consisting of transmitter and receiver circuits and the wire as a channel. At high speeds (> 3Gb/s), the bandwidth of the wire begins to limit the operating speed of the electric links [42]. Equalization techniques have been developed to compensate for the low-pass filtering effect of the channel (Figure 1.7). Multi-level signaling schemes are also proposed for higher throughput at the same bandwidth. Timing in the receiver is especially important to sample the data correctly in terms of frequency and phase.

Link Environment: Channel

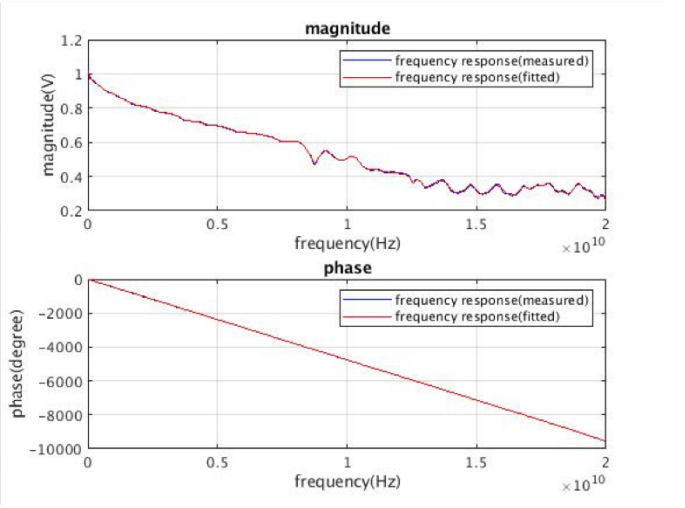
As the data rate reaches the 10Gb+/s range specified by the per-lane transfer rate of UCIe, the non-ideality caused by the channel has become a critical issue for signal integrity. A typical channel characteristic is shown in Figure 1.7 [19]). The channel is usually characterized as the s parameter (Figure. 1.7a). S21 or the corresponding frequency response shows the low-pass feature with notches at some frequencies in Figure. 1.7b. The notch and highly reflective transients in the pulse response could be generated from impedance discontinuities in complex PCB structures, such as packages and connectors. The corresponding time-domain pulse response is obtained from passing a 1 UI (unit interval, i.e. 1/data rate) square pulse as input to the channel. With low-pass characteristics, the pulse response has a lower peak amplitude and spread wider than 1UI (Figure. 1.7c. The inter-symbol interference (ISI) is the value measured in time that space nUI (n is nonzero integer) from the peak and are denoted in Figure. 1.7d.

Signaling: Signaling Scheme and Equalizers

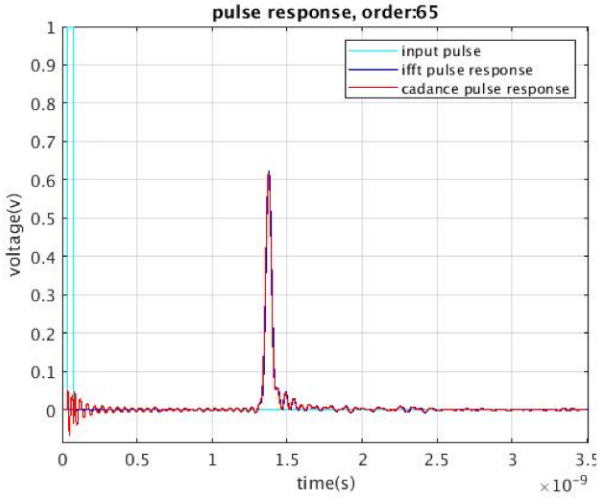
Non-return-to-zero (NRZ) is the intuitive and simple way to transmit the bit streams with two levels for 0 and 1 respectively. To increase spectral efficiency, complex modulation schemes, such as the Pulse Amplitude Modulation 4-level (PAM4) scheme, have been widely explored in recent transceiver designs to relax the baud rate requirements. In Figure. 1.8, the waveform of the ideal levels of PAM4 and NRZ with the input data streams is shown



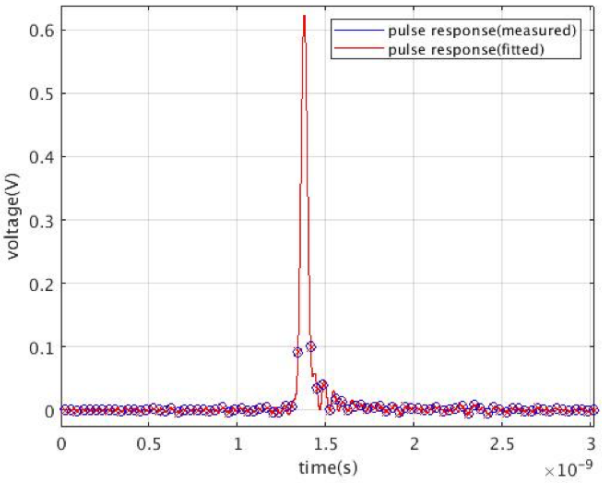
(a) S-parameters.



(b) Frequency response.



(c) 1UI input pulse and pulse response.



(d) Pulse response with ISI denoted.

Figure 1.7: IEEE P802.3ba 8dB nAUI channel characteristics [19].

on the left, while the corresponding eye diagram is shown on the right. Although PAM4 transceivers are capable of handling high-loss channels, most PAM4 designs require error coding and/or complex digital equalization due to a lower signal-to-noise ratio (SNR) and a higher impact of residual ISI and nonlinearity on eye opening, which typically leads to their power consumption being higher than the Non-Return-to-Zero (NRZ) counterpart. For example, PCIe 6.0 required forward error correction (FEC) to compensate for the loss of 9.5 dB SNR due to the lower eye height (1/3 of the NRZ eye). For the XSR link, NRZ is often preferred for its simplicity and power efficiency, while PAM4 provides an option to

realize a higher date rate at the same bandwidth at the cost of increased power and design complexity for signal integrity issue.

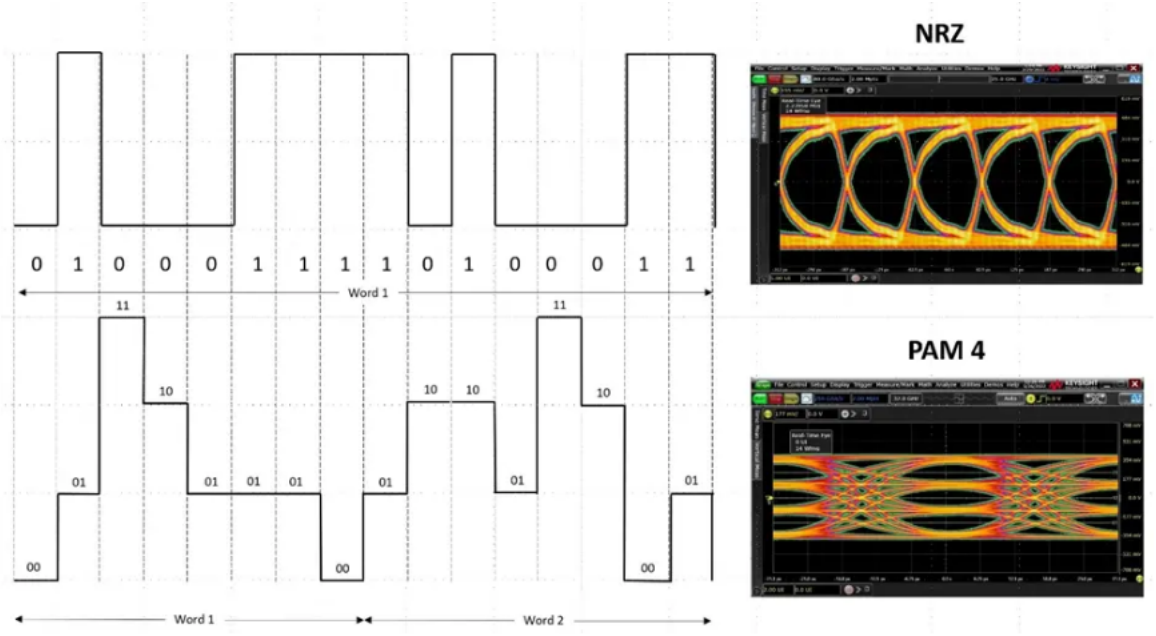
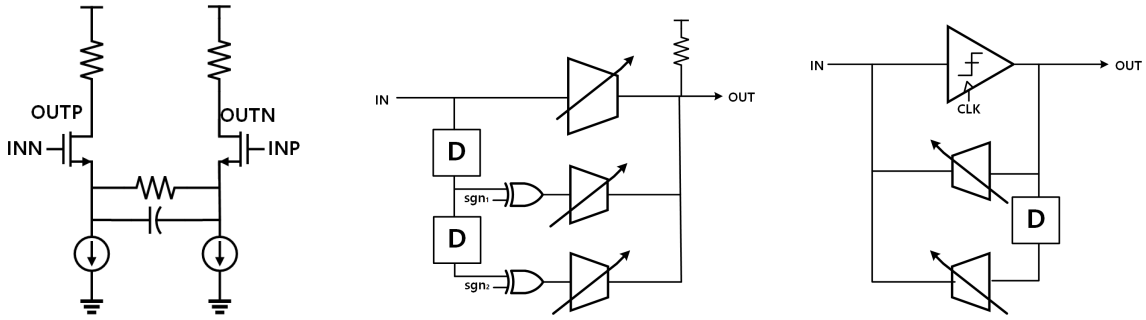


Figure 1.8: Comparisons of different signaling schemes [33].

Significant ISI due to the channel effect contributes to higher bit error rates (BER). Equalizers in transmitter and/ or receiver are implemented to compensate ISI and meet the BER specs at high data rate. If the amount of channel loss remains within a reasonable range (<25dB for NRZ and <15dB for PAM4), which is the short-link case we are targeting, analog and mixed signal equalizers are preferred over digital approaches to address ISI. Conventionally, various analog and mixed-signal equalizers have been implemented, including continuous-time linear equalizers (CTLE), feedforward equalizers (FFE), and decision-feedback equalizers (DFE).

FFEs (Figure 1.9b) are often used to correct pre-cursor ISI and usually perform at the transmitter side of the mixed-signal based link for simplicity of implementation. Since the transmitter output power is limited, the summation of the FFE taps has to be fixed, and therefore the signal power is decreased. The reduced signal power will cause noise enhancement. Both DFE and CTLE are usually placed on the receiver side. DFE(Figure 1.9c) outperforms CTLE for canceling major post-cursor ISI near the main cursor since DFE summer taps do not have the linearity constraints present in CTLE, and are more tolerant to noise. Furthermore, adapting DFE taps for irregular pulse responses is easier than CTLE. CTLE(Figure 1.9a) is commonly placed in the first few stages at the receiver and configurable to cancel both pre-cursor and post-cursor ISI. CTLE uses frequency-dependent gain degradation of the common source (CS) amplifier to create peaking around the channel

bandwidth to extend the bandwidth. In contrast to the finite impulse response filter (FIR)-based FFEs and DFEs, CTLE serves as an infinite impulse response filter (IIR) and can be used for long-tail cancellations.



(a) CTLE (b) FFE (c) DFE

Figure 1.9: Structures of commonly used equalizers.

Timing: Clock and Data Recovery Circuits (CDRs)

To correctly decide the data transmitted on the receiver side, it is extremely important to sample at the correct timing, i.e., accurate spacing of transmitted data symbols and sampling of the signal waveforms at the receiver. Clock and data recovery (CDR) is used to recover the clock with the received data and align the recovered clock to the locking position decided by the CDR loop. The CDR loop compensates for the phase skew between the received signal and the receiver's internal clock, and filters out the input jitter which causes the sampling uncertainty at the sampler. This is the main topic we will focus on in the following chapters.

Berkeley Analog Generator (BAG)

The wireline links used for different applications will have different specs to meet. Even developing a link with higher data rates based on existing links requires essential efforts to redesign each block and verify performance. The subtle change in the routing or via style can have a non-negligible degradation of performance. Thus, substantial layout iteration would be needed in the design process. To make matters worse, the design rule sets are more complicated in the advanced nodes, making the layout iteration more time consuming. These are the non-returning engineering (NRE) costs when the specs evolve. It is crucial to reduce design costs to increase profit margins for high volumes and enable custom design for low volumes.

Berkeley Analog Generator (BAG) [20] is an open-source hierarchical Python framework that allows the design procedure to be specified in executable manners. It defines and implements a process-portable interface to layout [29][2]. It has already been used mainly

for analog and mixed signal (AMS) layout automation [6][50]. BAG is a software wrapper for hardware design that focuses on reusing the unit design hierarchically instead of designing from scratch. Instead of doing manual layout with graphical polygons, circuit designers write executable generator in Python which capture designer's knowledge. In other words, the generator itself is an implementation of the circuit design methodology of designers.

The generator nomenclature is as follows:

- Schematic generator

  Produces schematic/ netlist based on low-level parameters (width, length, number of finger, etc.)

- Layout generator

  Produces DRC/ LVS clean layout based on low-level parameters (width, length, number of finger, etc.)

- Model generator

  Produces SystemVerilog model based on low-level parameters, netlist of the schematic, and paramter characterized from the simulation (delay, rising time, falling time, setup time, hold time, etc.)

With generator-based design methodology, it can truly close the entire design loop from design (schematic and layout) to verification (behavioral model and testbench).

The other advantage of automatic layout is the manageable layout process in advanced technology and portability among different technologies. The design rules are different from technologies to technologies, which makes porting an existing design from one technology to another not a trivial work. In fact, the complex design rules of the FinFET process further reduce layout portability and increase the time required to port the design. Within the BAG framework, the complicated design rules, such as the quantized width and spacing of the metal in different layers, are parameterized and assigned to the technology files. In this way, circuit designers do not need to bother knowing each design rules and can easily port one design to different technology nodes. With BAG, productivity gains come from parameterization, incremental extension, and process portability.

1.2 Thesis Organization

There are a variety of data-rate target die-to-die XSR link projects included in the thesis. The main contributions in these projects are outlined in Figure. 1.10. The detailed information and contributors of the projects are listed in Table. 1.1

In Chapter 2, the clock path targeting multiple data-rate links has been explored. The building blocks of clock generation and distribution are explained. The reconfigurability of the generator-based design makes it possible to adopt a circuit topology similar to the different target data rates in short cycle time and resources.

Chapters 3 to 6 focus on the theory and implementation of the baud rate CDR algorithm. Chapter 3 starts with a review of the literature on the CDR algorithm and the theory of the hybrid CDR algorithm. Statistical analysis is used to evaluate the performance of the innovative CDR algorithm and also serves to compare with the state-of-the-art CDR algorithm. In Chapter 4, the focus is on the implementation of the hybrid CDR algorithm in the design of the 100Gbps XSR receiver. 200Gbps receiver in [31] is modified for the hybrid CDR proposed in Chapter 3. The change in the datapath and the clock path achieves a more compact design and lower power consumption. Next, Chapter 6 describes the integration flow of the 100Gpbs receiver with hybrid CDR design and the packaging, PCB design for the testing in lab. The testing setup and results are discussed at the end of Chapter 6.

Finally, Chapter 7 concludes the thesis with the summary and potential future work based on the proposed work.

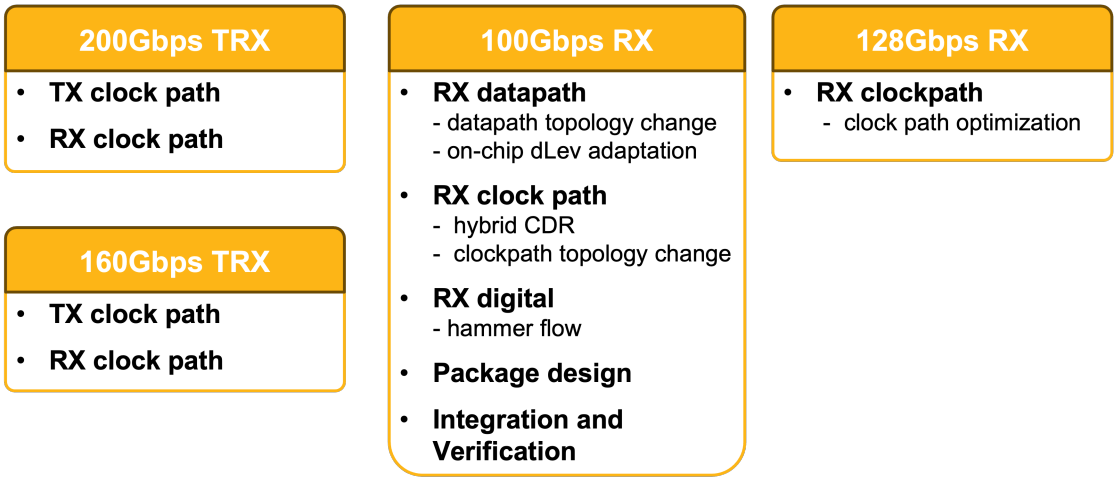


Figure 1.10: Project development.

Projects	200Gbps TRX	160Gbps TRX	100Gbps RX	128Gbps RX
Technology	22nm FinFET	16nm FinFET	16nm FinFET	16nm FinFET
Taped out time	2021 Nov.	2022 Aug.	2023 Dec.	2024 Nov.
TX datapath	Ayan Biswas[3]	Ayan Biswas[3]	-	-
TX clock path	Yi-Hsuan Shih[1]/ Wahid Rahman	Yi-Hsuan Shih[1]	-	-
RX datapath	Paul Kwon[4]	Paul Kwon[4]	Paul Kwon[4] / Yi-Hsuan Shih[2]	Kunmo Kim
RX clockpath	Yi-Hsuan Shih[1]/ Wahid Rahman	Yi-Hsuan Shih[1]	Yi-Hsuan Shih[2]	Yi-Hsuan Shih[1]/ Kunmo Kim
Digital Backend	Paul Kwon[4]	Paul Kwon[4]	Paul Kwon[4]/ Yi-Hsuan Shih[2]	Kunmo Kim/ Sunjin Choi
Passive Frontend	Kunmo Kim	Kunmo Kim	Kunmo Kim	Kunmo Kim

[1]Chapter 2

[2]Chapter 5 and Chapter 6

[3]Ayan Biswas thesis [18]

[4]Paul Kwon thesis [31]

Table 1.1: Project information and contributors.

Chapter 2

Clock Generation and Distribution for 100+ Gb/s Transmitter

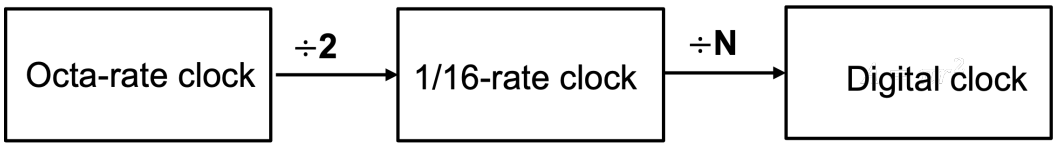
2.1 Clock Path Overview

The basic topology of clock distribution and the building block layout generator are inherited from [47] designed by Zhongkai Wang. The prior work [47] was done in 28nm planar technology and the design is listed in Figure. 1.10 were done in 16nm/22nm FinFET technology. Key changes were made to adapt to the change from quadrature path to octature path due to the double interleaving of the datapath to push the data rate limited by the technology.[1] The octature generator and the reset synchronizer are newly added to adapt to the change in interleaving. The duty cycle correction is added in the delay line and the change in the current DAC topology to provide a linear delay step. The clock path design for the 160Gpbs transceiver, the 128Gbps receiver, and the 100Gbps receiver will be discussed in this chapter. The clock path of links with different data rates shares a similar structure as summarized in Figure 2.1. The design is built hierarchically with Berkeley Analog Generator (BAG) which gives portability and flexibility to configure with different data-rate of the building blocks and the system.

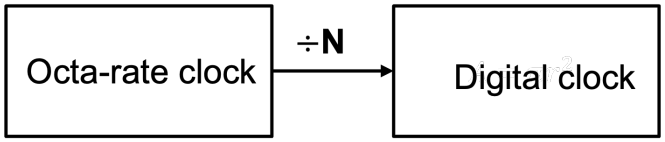
For 160Gbps and 128Gbps links, the two main analog clock domains are used considering the fanout and the speed limit of the latches as shown in Figure. 2.1a. For the 100Gbps link, the computation can be resolved without further deserialization and thus only one analog clock domain (Figure. 2.1b) is used for simplicity and compactness of the layout.

Both of the clock distribution topology further divide the analog domain clock to the diigital clock along with the deserialization after the MLSE computation.

[1] The initial clock path design and the newly added blocks (octature generator, reset synchronizer, global buffer, duty cycle correction) for 200Gbps TRX clock path is in collaboration with Wahid Rahman. The 160 Gbps TRX follows essentially the same structure, but with an improved layout and an added debug clock.



(a) Two main analog clock domains.



(b) One main analog clock domain.

Figure 2.1: Clock distribution.

160Gbps TRX Clock Distribution Overview

The clock distribution structure for the transmitter (TX) and receiver (RX) is designed similarly with minor modifications devoted to specific needs, as shown in Figure 2.2. This simplifies the design, and the layout generator can be reused with different configurations.

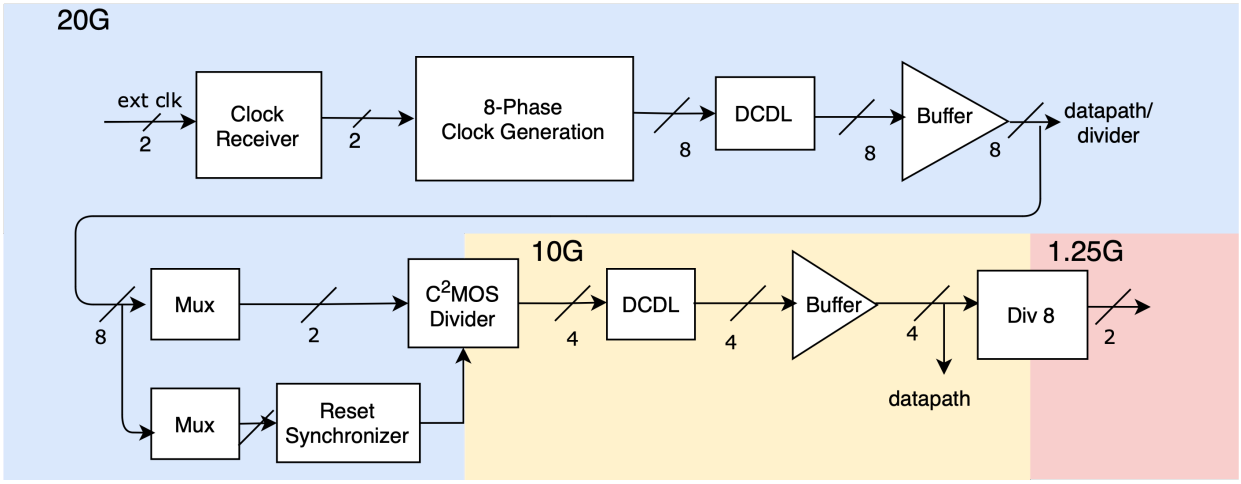
There are two clock domains (20G and 10G) in RX and three clock domains (20G, 10G, 1.25G) in TX, respectively. The 20G 8-phase clock domain contains the clock receiver, the octature generator, the current control delay line, and the global buffer for both TX and RX.

Four quadrature divider divide the 8 phase 20G clock to the 16 phase 10G clock in RX. The 10G clocks then feed into the phase rotator for the coarse tuning and the delay line for the fine tuning.

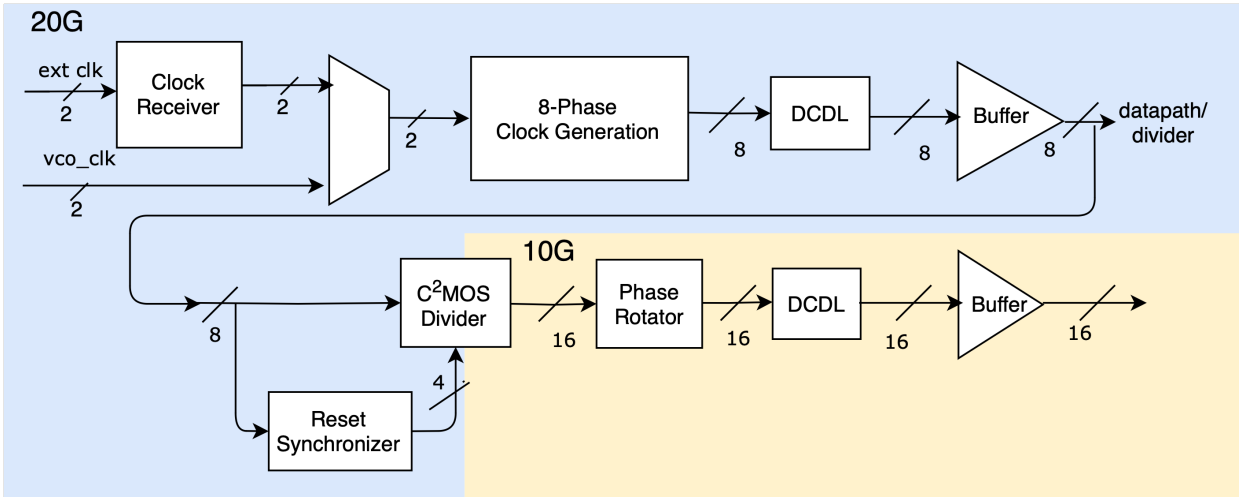
In TX clock path, only 4-phase is needed in 10G domain, thus only one quadrature divider is used. The mux is used to choose the clock phase for the divider input and the reset path to compensate for the delay across the corner. DCDL is enough to correct for the 4 phase skew compared to the RX counterpart. The divided clock followed by the Div-by-8 divider is generated by the clock path and sent to the TX datapath.

128Gbps RX Clock Distribution Overview

The 128Gbps RX clock distribution (Figure. 2.3) is built on the basis of the topology of the 160Gbps RX clock distribution (Figure. 2.2b).The main modification is the sizing and stacking change of the oscillator inside the 8-phase clock generation to achieve 128 GHz frequency with decent phase error across PVT corners. The C^2MOS divider is redesigned for robustness to the input clock skew which causes failure in the 160GHz RX measurement.



(a) Transmitter (TX) clock path.



(b) Receiver (RX) clock path.

Figure 2.2: 160GHz transceiver clock path.

100Gbps RX Clock Distribution Overview

The 100Gbps RX clock path (Figure. 2.4) follows the topology of Figure 2.1b where only one main analog clock domain is presented due to the relaxation of the speed limit from the down scale of the data rate. Since the sampler can be clocked at 12.5GHz, there is no need for the divided 1/16 rate clock. The change simplifies the design of the clock path and reduces the area of the 16-phase 1/16 rate clock distribution. The other difference is that the input of the 8-phase generation comes from either an external clock or the VCO clock generated by the LC VCO on chip controlled by the CDR logic in the digital domain.

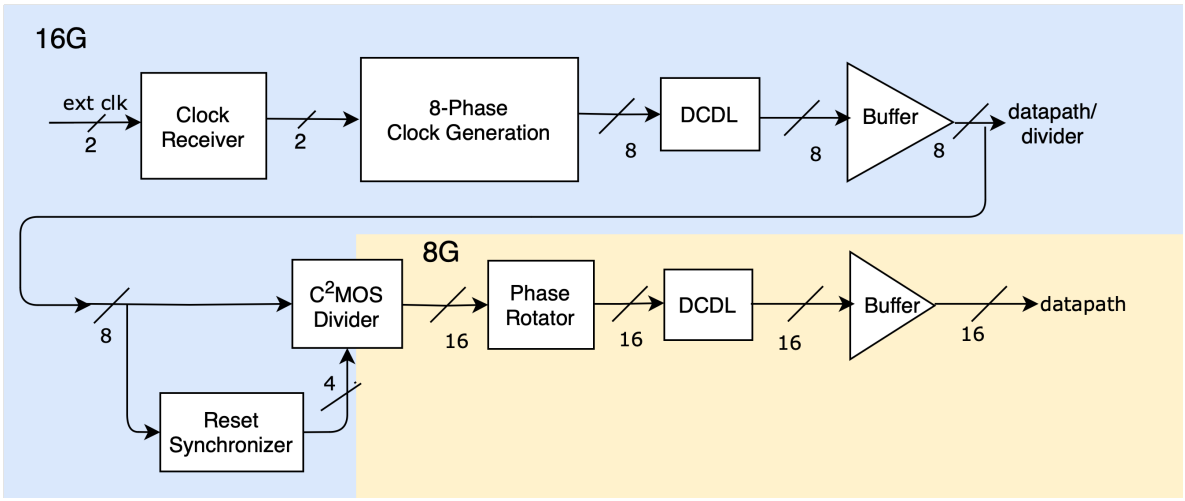


Figure 2.3: 128GHz receiver clock path.

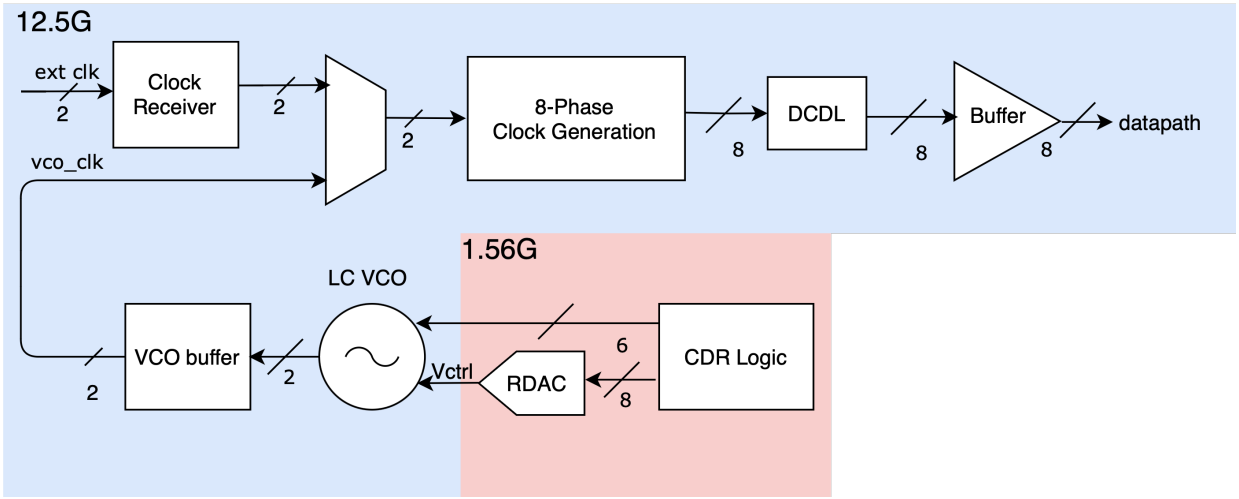


Figure 2.4: 100GHz receiver clock path.

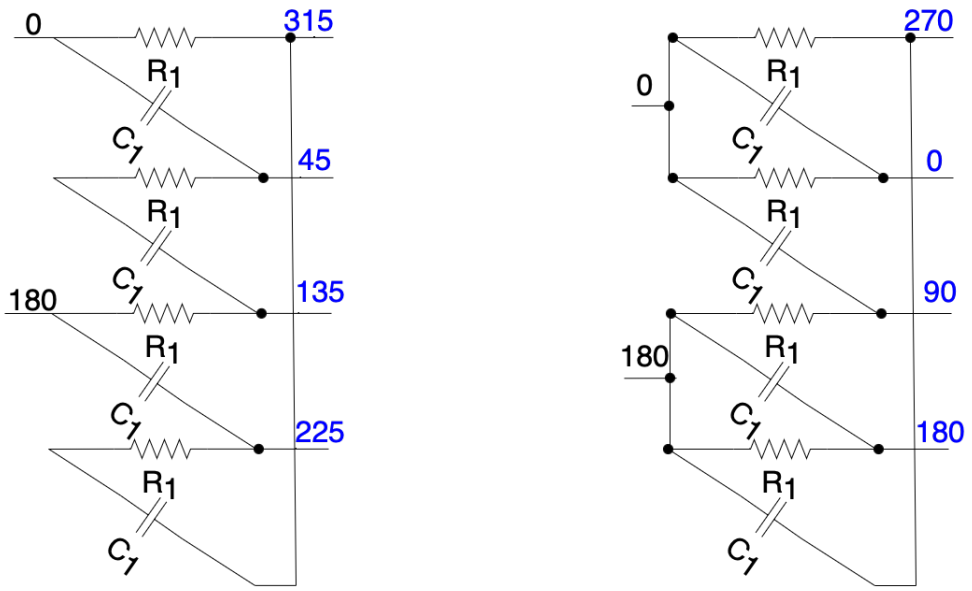
2.2 Octa-Rate Clock Distribution

Octature Generator

Multiphase generation

Multiple phase clocks are essential for the high-speed SerDes to enable the interleaving path in the datapth. There are several ways to generate the multiple phase clocks, which can be mainly categorized to closed-loop and open-loop methods. For closed-loop method, DLL and PLL can be used for multiphase clock generation, but the power consumption is high, and large area and long lock-in time are required.

On the other hand, an open loop inverter-based ring oscillator and polyphase filter can be used to generate multiphase clocks[7]. There are two types of quadrature generation of polyphase filters depending on the connection of the input clocks. Both types provide a 90 degree phase shift and a matched amplitude at the pole frequency ($\omega = \frac{1}{RC}$). Type 1 is constant phase, which means that the phase shift is constant across the frequency, while the amplitude of the outputs only matches at pole frequency. Type 2 in contrast is constant-amplitude which has constant matched amplitude response across the frequency while the phase shift is 90 degree only at pole frequency. The sensitivity to component mismatch (ΔR, ΔC) of Type 1 is twice higher than of Type 2 [40] and the phase error is tunable in the following delay line; we chose the Type 2 polyphase filter for quadrature and octature generation.



(a) Type 1 polyphase filter.

(b) Type 2 polyphase filter.

Figure 2.5: Polyphase filter for quadrature phase generation.

To implement octature generation with Type 2 polyphase filter, we can use the phase shift property of the following stage to achieve a 45 degree phase shift. The octature generator is shown in 2.6 where 8 phases are generated by one-stage and two-stage polyphase filters, respectively. To determine the values of the components, we rewrite the relation between R_1 and R_2 as

$$R_2 = \alpha R_1 \tag{2.1}$$

Since the pole frequency needs to be the target frequency for exact phase shift, we have

$$R_1 C_1 = R_2 C_2 = \frac{1}{\omega} \tag{2.2}$$

Therefore,

$$C_2 = \frac{C_1}{\alpha} \tag{2.3}$$

Assuming that the loading of each phase is C_L and $\frac{1}{\omega C_L} >> R_1, R_2$, we have the following.

$$|V_{o1}| = |V_{in}| \tag{2.4}$$

$$|V_{o2}| = \sqrt{2}|V_m| = \sqrt{2}(\frac{\alpha}{\alpha+1}|V_{in}|) \tag{2.5}$$

To match V_{o1} and V_{o2},

$$\frac{\alpha}{\alpha+1} = \frac{1}{\sqrt{2}} \tag{2.6}$$

By solving the above equation, we have

$$\alpha = 2.414 \tag{2.7}$$

Let us take 25GHz clock generation for example. If we have $C_L = 6fF$ for each phase, we can implement the octature generator by choosing $R_1 = 93\Omega, C_1 = 68.5fF, R_2 = 217\Omega, C_2 = 29.3fF$. With an input clock of a perfect 25GHz differential sine wave, the output schematic simulation results are shown in Figure 2.7. However, if the input is not a sinusoidal wave, then the output has significant distortion and phase error.

The polyphase filter only consists of passive circuits, and thus has the advantage in terms of the power consumption. However, the narrow-band frequency filtering characteristics strictly require the input clock to be a sinusoidal wave. For square waves containing harmonic frequency components, the phase shift will not be the same as the fundamental frequency, causing the phase error and the distortion of the output clocks. The input frequency range can be extended by cascading more stages of the polyphase filter. However, the signal attenuation will be significant with more stages. To ensure sinusoidal input, the resonant buffer is needed before the polyphase filter, increasing the area and the power consumption.

The inverter-based injection-locked ring oscillator is an alternative open-loop multiphase generation. It consumes more power than the polyphase filter due to its active circuit nature.

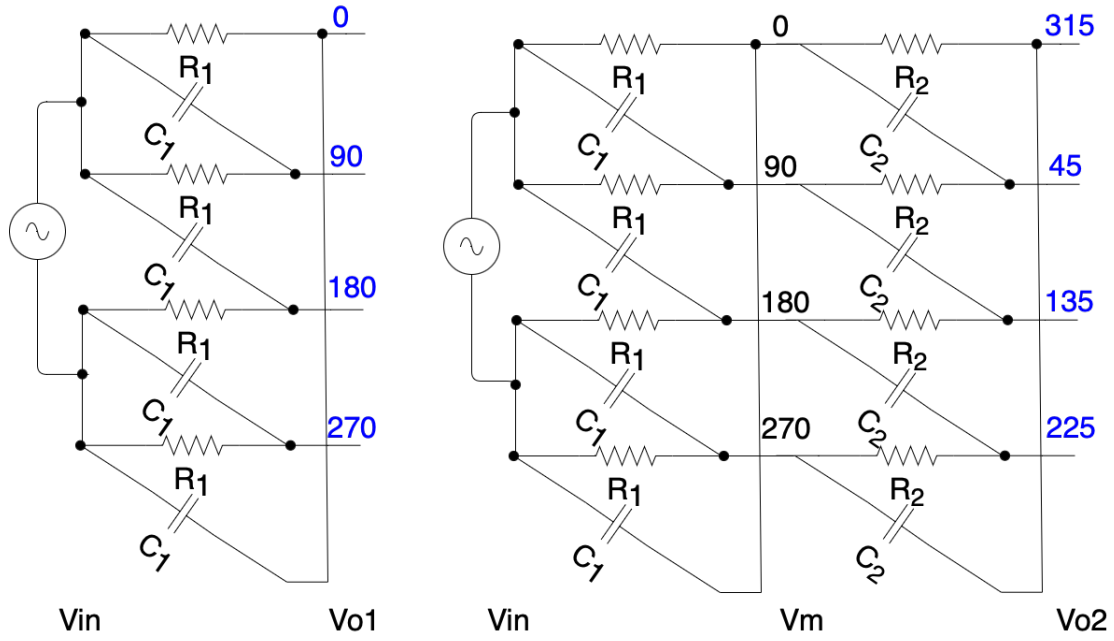


Figure 2.6: Polyphase filter for octature clock generation

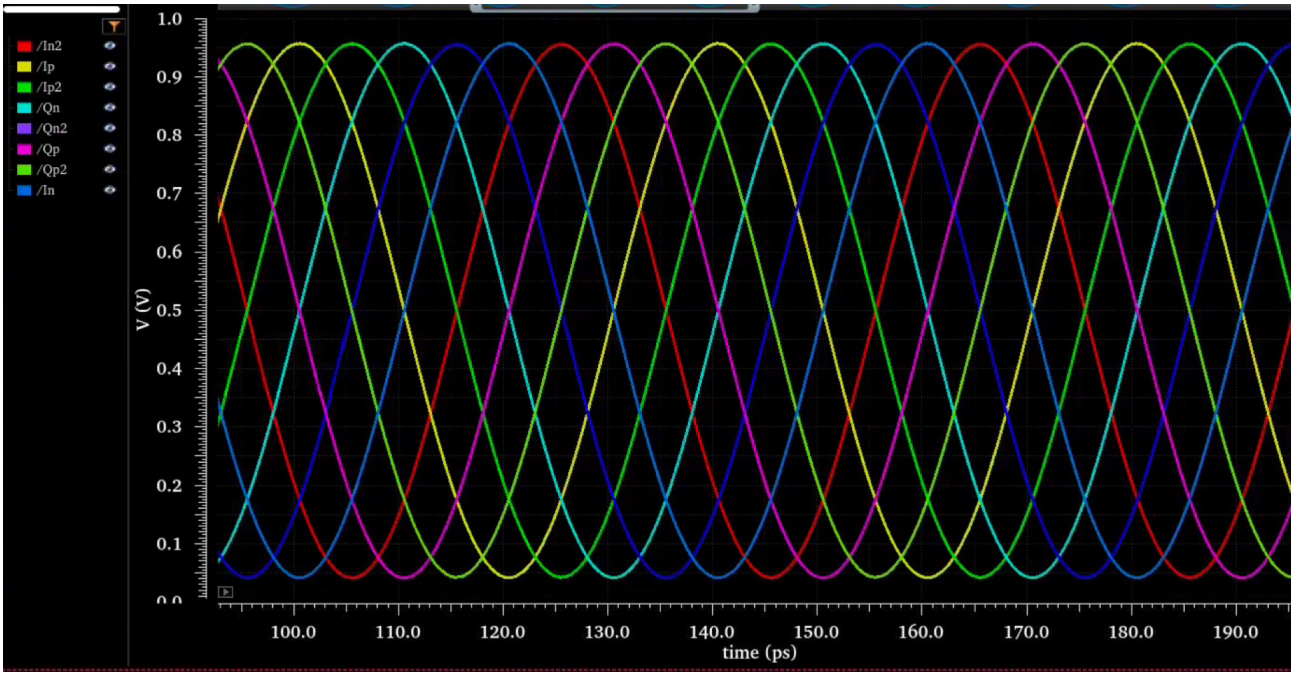


Figure 2.7: Schematic simulation of polyphase filter octature clock generation.

However, it provides a wide frequency locking range with sufficient injection strength. The MOSFET-based layout is compacter than that of the passive resistor and capacitor. The quadrature generator based on the injection-locked ring oscillator is proposed in [8] and used in [32] [48][51]. Due to the wide-bandwidth input and wide locking range, the inverter-based structure is chosen to be the multiphase clock generation structure. It is intuitive to generate the octature generator from the single ring oscillator by having 8 inverters connected in one loop. However, it is difficult to achieve high speed due to the process-dependent minimum inverter delay.

Phase Interpolation

Suppose that the clock waveform is a sinusoidal wave with amplitude A, radian frequency ω, and phase angle ϕ to allow for any position of the time origin. The clock waveform can be written in the following general form.

$$x(t) = Acos(\omega t + \phi). \tag{2.8}$$

For two clock waveforms with the same amplitude and radian frequency but different phases, we can express them as follows.

$$\begin{aligned} x_1(t) &= Acos(\omega t + \phi_1) \\ x_2(t) &= Acos(\omega t + \phi_2). \end{aligned} \tag{2.9}$$

The corresponding non-rotating phasors taken by taking a snapshot of t=0 of both clock waveforms are

$$\begin{aligned} \underline{X_1} &= A\angle\phi_1 = A(cos\phi_1 + jsin\phi_1) \\ \underline{X_2} &= A\angle\phi_2 = A(cos\phi_2 + jsin\phi_2). \end{aligned} \tag{2.10}$$

To interpolate between two clock phases, we weighted the sum of two clock waveforms. Assuming that the weighting is the same for both waveforms, i.e. $w_1 = w_2 = \frac{1}{2}$, then we have

$$\begin{aligned} \underline{X_3} &= w_1\underline{X_1} + w_2\underline{X_2} \\ &= \frac{1}{2}(\underline{X_1} + \underline{X_2}) \\ &= \frac{1}{2}A[(cos\phi_1 + cos\phi_2) + j(sin\phi_1 + sin\phi_2)] \\ &= A_3\angle\phi_3. \end{aligned} \tag{2.11}$$

In Equation. 2.11, A_3 and ϕ_3 are the interpolated amplitude and the phase, respectively. We would like to rewrite it using the input amplitude A and the input phases ϕ_1 and ϕ_2.

1. find A_3

$$\begin{aligned} A_3^2 &= ||\underline{X_3}||^2 = \underline{X_3}\underline{X_3^*} \\ &= (\frac{A}{2})^2[(cos\phi_1 + cos\phi_2)^2 + (sin\phi_1 + sin\phi_2)^2] \\ &= \frac{A^2}{4}[(cos^2\phi_1 + sin^2\phi_1 + cos^2\phi_2 + sin^2\phi_2) + 2cos\phi_1 cos\phi_2 + 2sin\phi_1 sin\phi_2] \\ &= \frac{A^2}{4}[2 + (cos(\phi_1 + \phi_2) + cos(\phi_1 - \phi_2)) - (cos(\phi_1 + \phi_2) - cos(\phi_1 - \phi_2))] \\ &= \frac{A^2}{4}[2 + 2cos(\phi_1 - \phi_2)] \\ &= \frac{A^2}{2}[1 + cos\Delta\phi] \end{aligned} \tag{2.12}$$

2. find ϕ_3 To easily calculate ϕ_3, we can rotate both ϕ_1, ϕ_2, ϕ_3 by $-\phi_2$.

$$\begin{aligned} \underline{X_3'} &= w_1\underline{X_1'} + w_2\underline{X_2'} \\ &= \frac{1}{2}A[(cos(\phi_1 - \phi_2) + cos0) + j(sin(\phi_1 - \phi_2) + sin0)] \\ &= \frac{1}{2}A[(cos\Delta\phi + 1) + jsin\Delta\phi] \\ &= A_3\angle(\phi_3 - \phi_2) \end{aligned} \tag{2.13}$$

We can take arctan to calculate $\phi_3 - \phi_2$.

$$\begin{aligned} \phi_3 - \phi_2 &= tan^{-1}\frac{sin\Delta\phi}{cos\Delta\phi + 1} \\ &= \frac{\Delta\phi}{2} \qquad \forall -\frac{\pi}{2} < \frac{\Delta\phi}{2} < \frac{\pi}{2} \\ \phi_3 &= \frac{\Delta\phi}{2} + \phi_2 \\ &= \frac{\phi_1 - \phi_2}{2} + \phi_2 \\ &= \frac{\phi_1 + \phi_2}{2} \qquad \forall -\pi < \Delta\phi < \pi \end{aligned} \tag{2.14}$$

Here is the caveat that the difference of the phase must be within $-\pi$ to π to use the interpolation of the phase. In other words, the phase has to be wrapped within $-\pi$ to π for Equation 2.14 to hold.

Octature Generation Theory

Based on the theory of coupling between two rings from [39], we designed the octature generator consisting of two coupled ring oscillators that generate quadrature clocks individually. By doing so, we can achieve almost 2x the speed that a single ring oscillator can achieve. The phase interpolation between the two quadrature generators creates a 45-degree phase shift between them. The phase of the ring oscillator is decided by the input of the ring inverters, the input of the coupling inverters, and the condition to make the ring oscillator oscillate.

Let us assume that there are two identical rings that oscillate at the same frequency. Without any coupling, the two rings will oscillate independently and have the phase shift ϕ shown in Figure. 2.8 which is undetermined and unpredictable. The cross-coupled pair within the ring to ensure the oscillation of even stages oscillator is omitted in the figures just for simplicity.

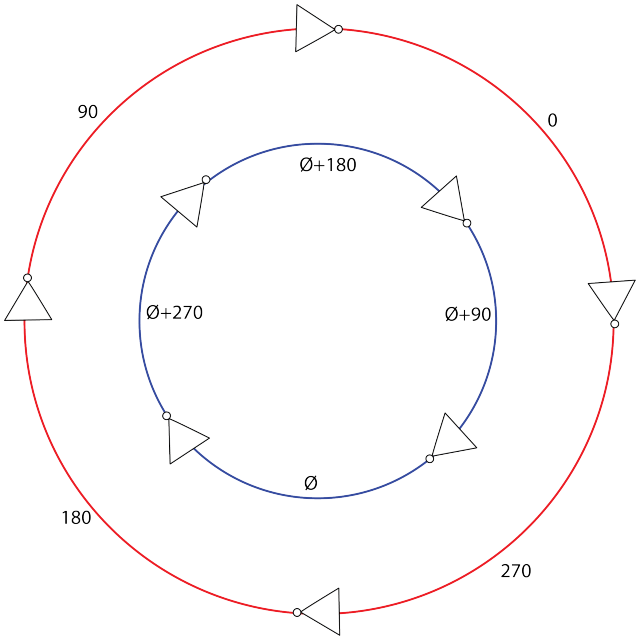


Figure 2.8: Two rings oscillate independently.

To make the ring oscillate, the phase shift of the whole ring should be 360 degrees. Therefore, the phase shift of an inverter within the ring should be $\frac{360}{N}$ for N stages. For the inverter between rings without phase interpolation with the other inverter output, the phase change should be 180 degrees, as shown in Figure 2.9a. Based on the inverter phase relation, we can zoom in on a part of the ring and analyze the phase relation between the two rings after adding the coupling inverter. In Figure 2.9b, we decompose the phase interpolation into two parts, the input from the previous ring inverter and the input from the coupled inverter. The two inputs are combined at the input of the ring inverter and then experience

a $\Delta\phi$ phase shift to the output of the ring inverter. For the ring oscillator to oscillate, the output of each stage must have a phase shift of -90 degrees. In Figure 2.9b, the output of inverter A is rewritten as -90 from 270 to ensure the -90-degree shift of the ring oscillator. We can find the phase relation ϕ between two rings by examining the phase summation and the change in inverter A and inverter B. Assuming that the coupled inverter has the same driving strength as the ring inverter, we can use Equation 2.14 for the input summation.

Inverter A has the phase relation as the following equation.

$$\frac{1}{2} \times 0 + \frac{1}{2} \times (\phi - 90) + \Delta\phi = -90 \tag{2.15}$$

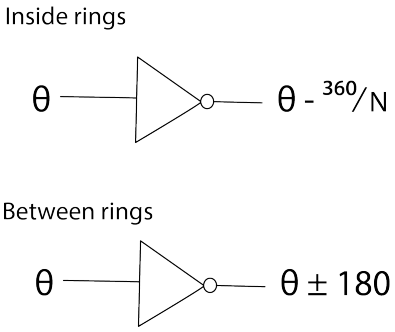
Using the same analysis, the phase relation at inverter B can be written as follows.

$$\frac{1}{2} \times (\phi + 180) + \frac{1}{2} \times (180) + \Delta\phi = \phi + 90 \tag{2.16}$$

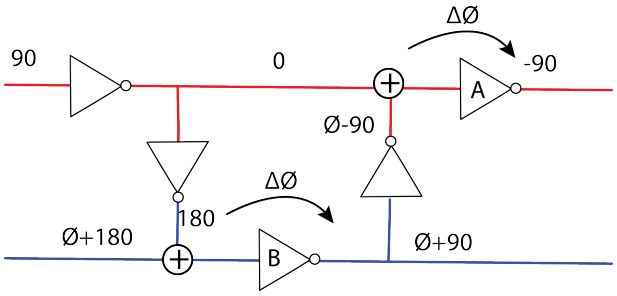
By solving Equation 2.15 and Equation 2.16, we have

$$\begin{aligned} \phi &= 45 \\ \Delta\phi &= -112.5 \end{aligned} \tag{2.17}$$

The results in Equation 2.17 show that with the same driving strength coupling between two rings, we can make the phase shift between two rings exactly 45 degrees apart. If we place the coupling inverter around the full ring oscillators, we can get the coupled ring oscillator shown in Figure 2.10 running at the same frequency as the quadrature generator, but generate 8 phases with 45 degrees apart.



(a) phase relation of inverter.



(b) phase relation of part of the coupled oscillator.

Figure 2.9: Phase relation of input and output of the inverter. N: number of stages in the ring oscillator.

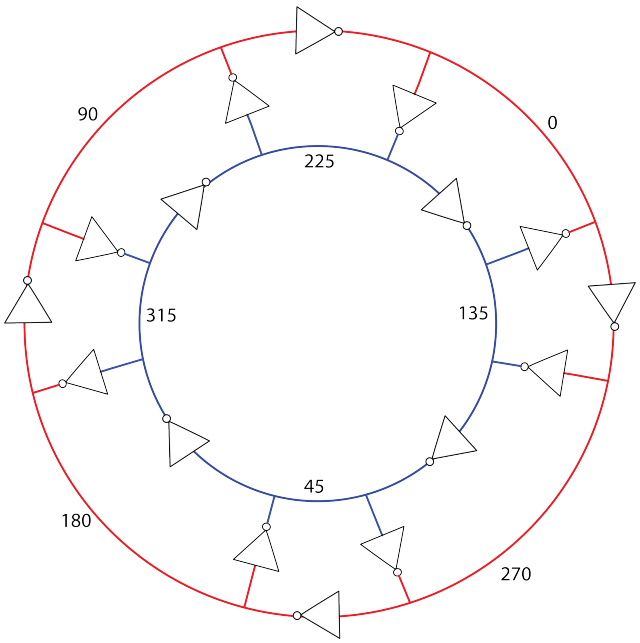
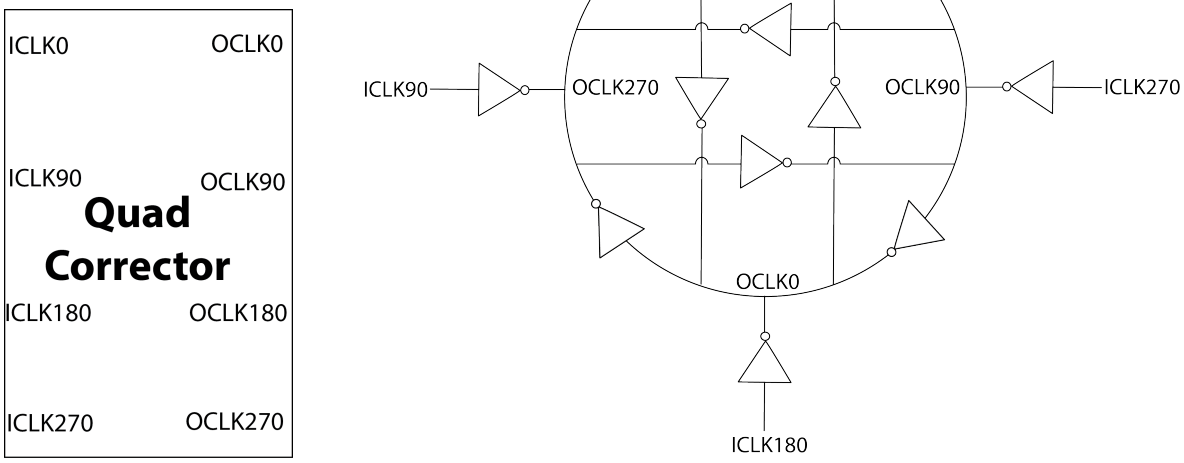


Figure 2.10: Two rings oscillate coupled with each other.

Quadrature generation

The quadrature generation can be achieved with the quadrature correlator proposed in [8]. It contains the main structure of the ring, the cross-coupled inverters inside the ring, and the injection inverters shown in Figure 2.11. The four inverters form the ring structure and generate four-phase clocks ($OCLK0$, $OCLK90$, $OCLK180$, $OCLK270$) when entering oscillation states. The cross-coupled pairs are set to be half of the main ring strength to help prevent metastability. The input clocks ($ICLK0$, $ICLK90$, $ICLK180$, $ICLK270$) inject through the injection inverters to the ring to lock the frequency and the phases are interpolated between the input clocks and the 90 phase shift inside the ring.



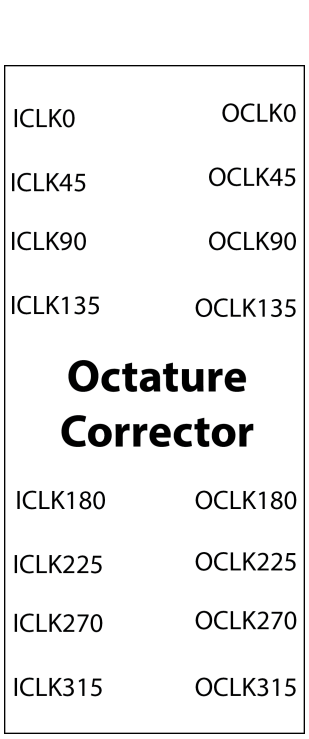
(a) Symbol. (b) Schematic.

Figure 2.11: Quadrature corrector.

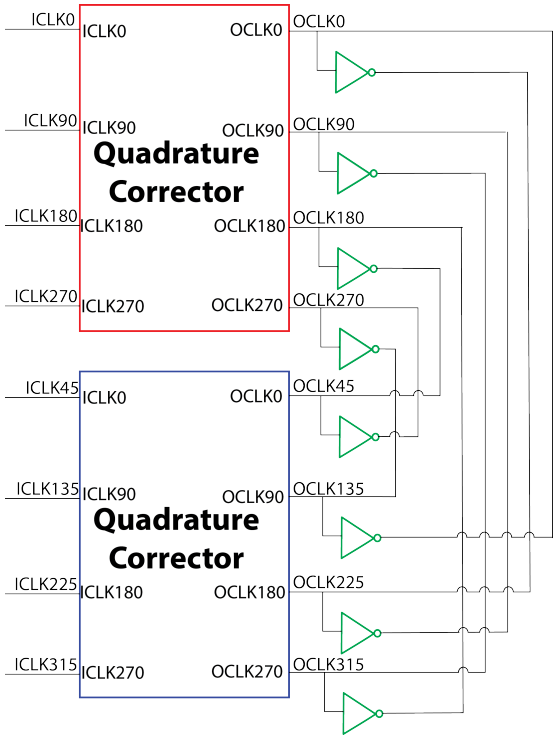
In [8], the structure of the quadrature generator is based on the cascading quadrature correlator to convert the differential clock to the quadrature clocks. The phase error can be minimized when more stage of quadrature correctors are cascaded with the expanse of power.

Octature genreator structure

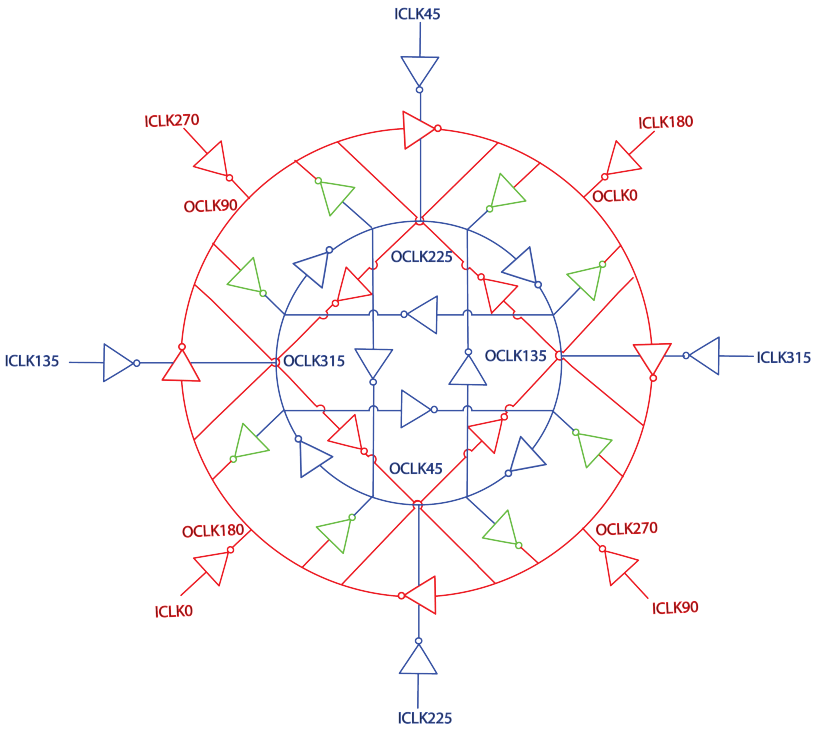
In the octature generation theory section, we explored a way to interpolate two quadrature rings to generate eight phases with a 45-degree phase shift. We can apply the theory to the quadrature corrector we introduced in the previous section. The conceptual schematic of the implementation is shown in Figure 2.12b. The outputs of the two quadrature correctors are coupled through the coupling inverters in green to achieve phase interpolation between two quadrature rings. The detailed schematic is shown in Figure 2.12c with the two ring oscillators shown in red and blue ,respectively. The inverter strength of ring 1 and ring 2 is shown in Figure 2.12d and Figure 2.12e with the same philosophy as stated in the previous section. The coupling inverters shown in Figure 2.12f have the same driving strength as the main ring structure in ring 1 and ring 2 to perform the octature generation.



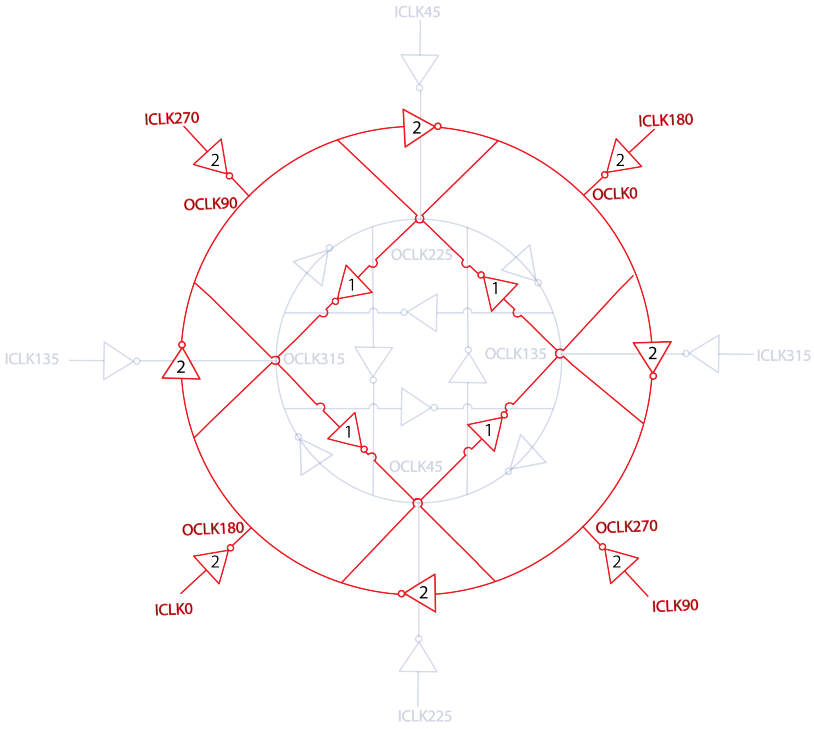
(a) Symbol.



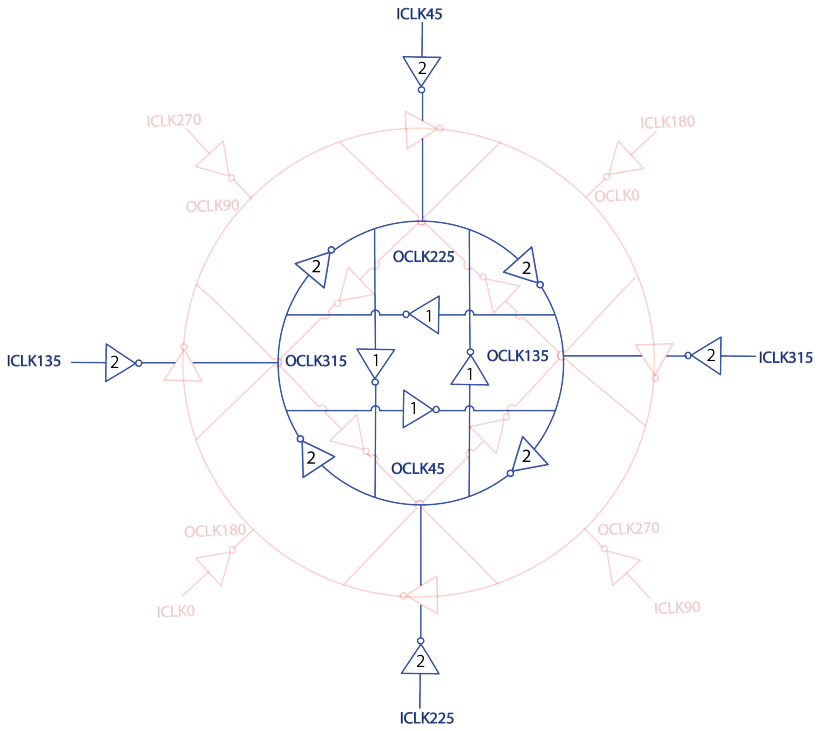
(b) Schematic.



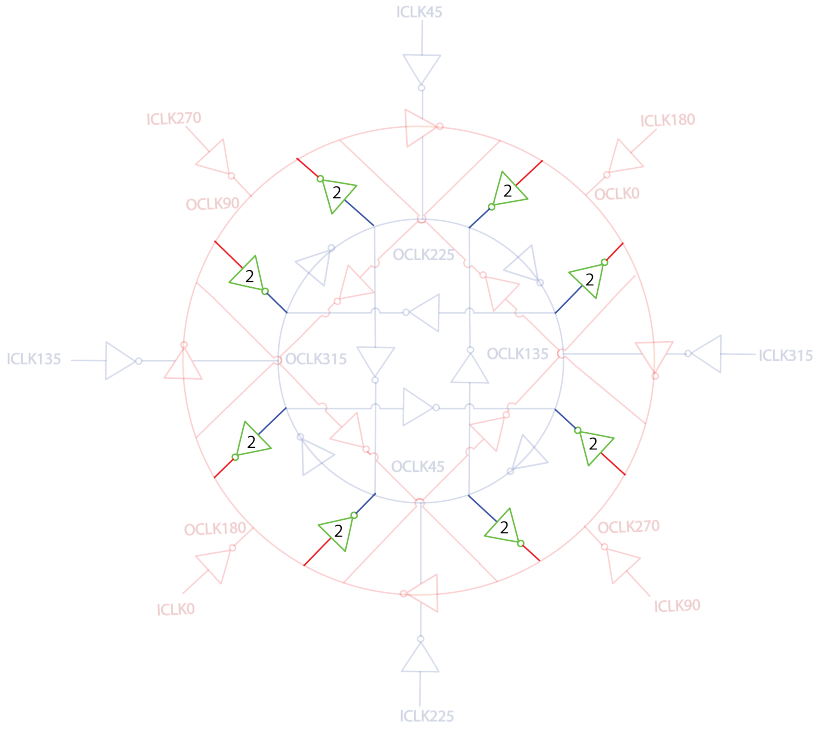
(c) Detailed schematic.



(d) Ring 1, with inverter strength annotated.



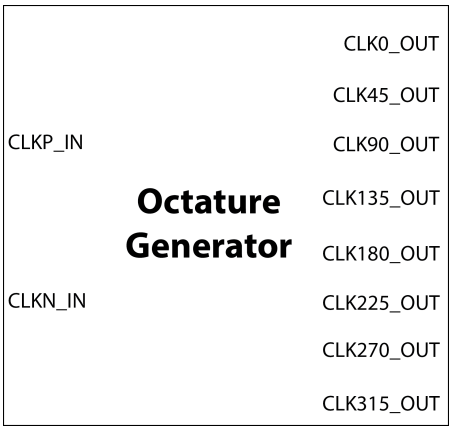
(e) Ring 2, with inverter strength annotated.



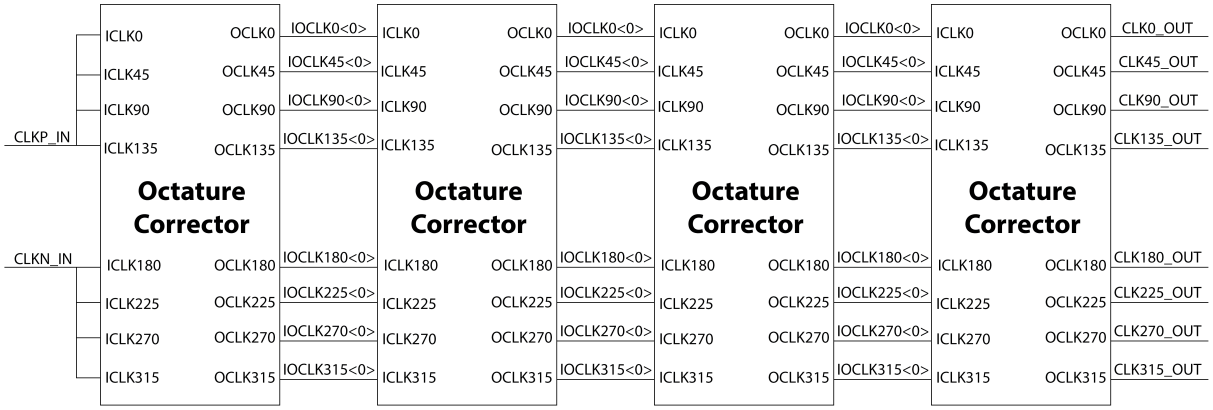
(f) Coupling inverters between two rings with inverter strength annotated.

Figure 2.12: Octature corrector.

The octature generator shown in Figure 2.13 is implemented by cascading 4 stages of the octature correctors (Figure 2.13b. The input of the octature generator is the differential clock $CLKP\_IN$, $CLKN\_IN$. The octature corrector is locked to the frequency of the input differential clocks. The phase of the output clocks of each octature generator is the interpolation of the 45 phases shift inside the quadrature rings and the input of the octature corrector. Thus, the output clocks would be closer to the ideal 8-phase clocks with more octature correctors cascading in series. The layout strategy is to have the two quadrature



(a) Symbol.



(b) Schematic.

Figure 2.13: Octature generator.

generators placed on top and bottom, which make one stage of the octature corrector. Then, the next stage can be cascaded by placing on the right. The size of each stage and the number of stages are configurable using BAG. Figure. 2.14 shows the simulation results of the octature clocks operated at 25GHz. The design includes four-stage quadrature correctors.

The solid clock waveforms include the power grid in the RC extration while the dotted clock waveforms only include block-level RC extraction. The performance of 25GHz is summarized in Table. 2.1. Here, we only show the representative simulation results for 25GHz used in 200Gbps TRX since it was the most challenging specs among the TRX clock path in Figure. 1.10 we have built. The phase error is less than 3 degrees and the locking range is larger than 20%.

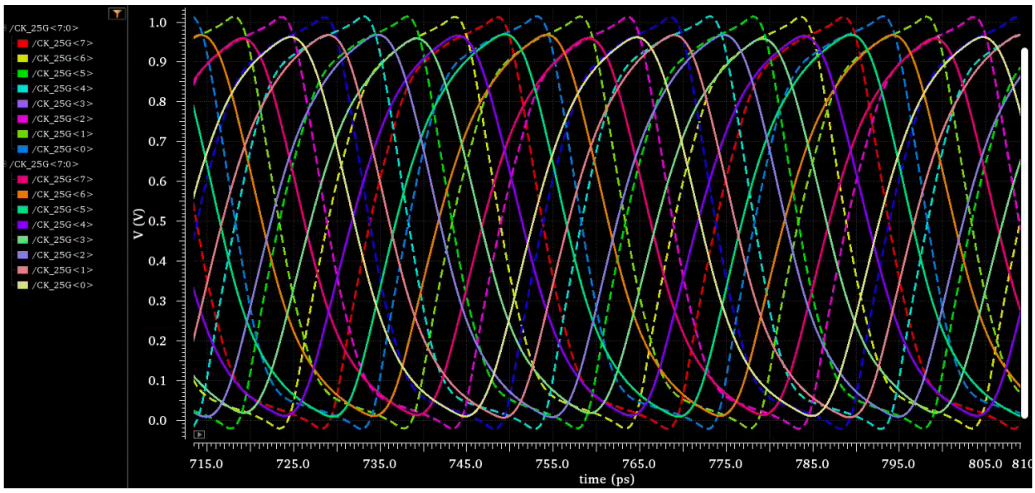


Figure 2.14: Octature clocks post-layout simulation results for 25GHz clock.

Phase error	Locking range	$V_{se,pp}$	Duty cycle	Power	Jitter
$< 3^o$	$> 20\%$	$0.8V_{DD}$	$48\% \sim 52\%$	35mW	23.7fs

Table 2.1: 25GHz octature clocks performance (simulation).

Digitally Controlled Delay Line (DCDL)

The digitally controlled delay line (DCDL) is used to adjust each phase of the clock to minimize the phase error. The basic structure of the delay element in DCDL is the current-starved inverter. Current-starved devices are on top and bottom of the inverter with the biases controlled by the current DAC. The current is adjusted by controlling the gate voltage of current-starved devices, which change the speed of charging and discharging the cap. The slower the charging and discharging, the longer the output delay. Extra pull-up and pull-down devices in parallel with current-starved devices are used to fine-tune the duty cycle with the digital input signal at the gates.

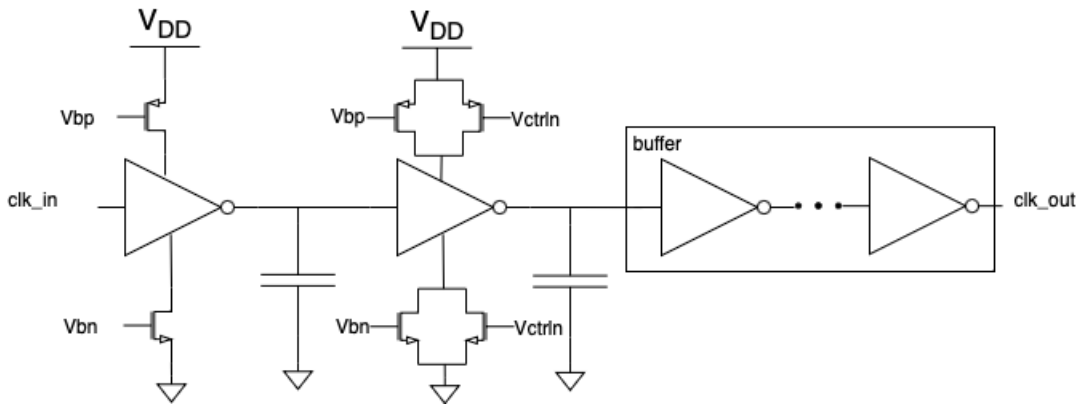


Figure 2.15: Delay line structure.

The conventional current DAC structure is shown in Figure 2.16a, where the reference branch in blue is fixed and the output current is decided by how many unit branches are turned on of the output branches N in red. Suppose that the unit output branch is the exact same size as the reference branch. In this case, the output current is proportional to the input code assuming that the input code directly indicates the number of branches turned on. If n represents the input code and $I_{refp} = \alpha_{ref} I_{REF}$ the relationship between the output current and the input code can be written in the following equation.

$$I_{out} = nI_{refp} = n\alpha_{ref} I_{REF} \tag{2.18}$$

Assuming the capacitance load for the delay line is C, the voltage swing of the delay line is ΔV, and the current mirror ratio between the current-starved device in the delay line and the output branch is α_{DAC}, the delay D can be represented as

$$D = \frac{C\Delta V}{n\alpha_{ref}\alpha_{DAC} I_{REF}} \propto \frac{1}{n}. \tag{2.19}$$

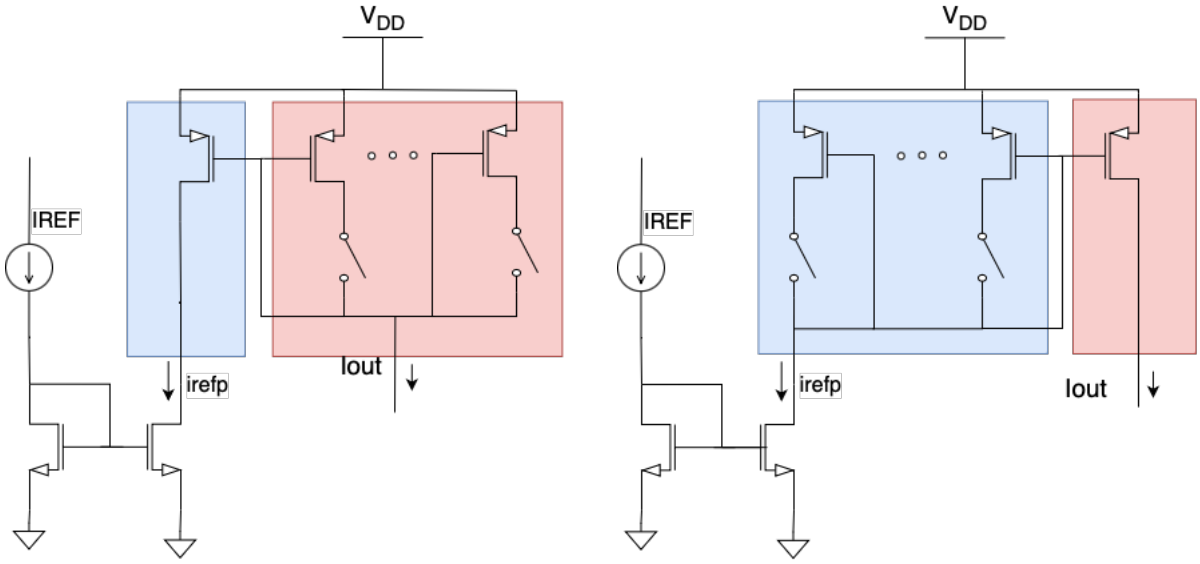
As shown in Equation 2.19, the input code is inversely proportional to the delay generated by the delay line.

To have a linear delay step on the delay line, the current DAC for the delay line is designed with the input code proportional to the delay instead of inversion proportional to the delay shown in Figure 2.16b. Given the same sizing condition as that for the unit branches, the output current can be written as

$$I_{out} = \frac{I_{refp}}{n} = \alpha_{ref}\frac{I_{REF}}{n} \tag{2.20}$$

Using the same symbol as before, we can find the delay D of the delay line as

$$D = \frac{nC\Delta V}{\alpha_{ref}\alpha_{DAC}I_{REF}} \propto n \tag{2.21}$$



(a) Current DAC with linear current step.

(b) Current DAC with linear delay step.

Figure 2.16: Current DAC structures.

The connection between the current DAC and the delay line is shown in Figure 2.17. For the 8-phase clock domain, there are eight copies of the current DAC and delay line dedicated for the tuning of each phase. Delay line for each phase is controlled by the current DAC with the topology in Figure. 2.16b shown in the green block in Figure. 2.17. The output current of the current DAC then mirrors the bottom and top current-starved devices to control the delay of the inverters shown in the yellow block.

The example simulation results are shown in Figure. 2.18 with 500 μA reference current and 7 ps tuning range (0.7UI). The nonlinearity occurs mainly in MSB switching due to the binary DAC structure. The step size is around 150fs. By increasing the reference current, the step size and tuning range will decrease accordingly.

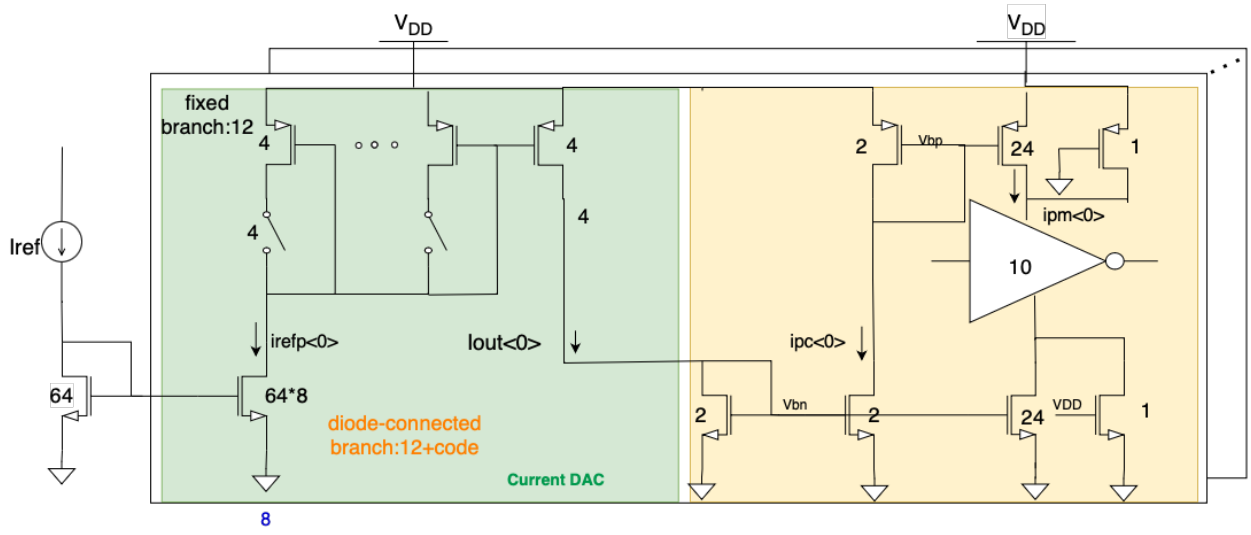


Figure 2.17: Current and delay line branches.

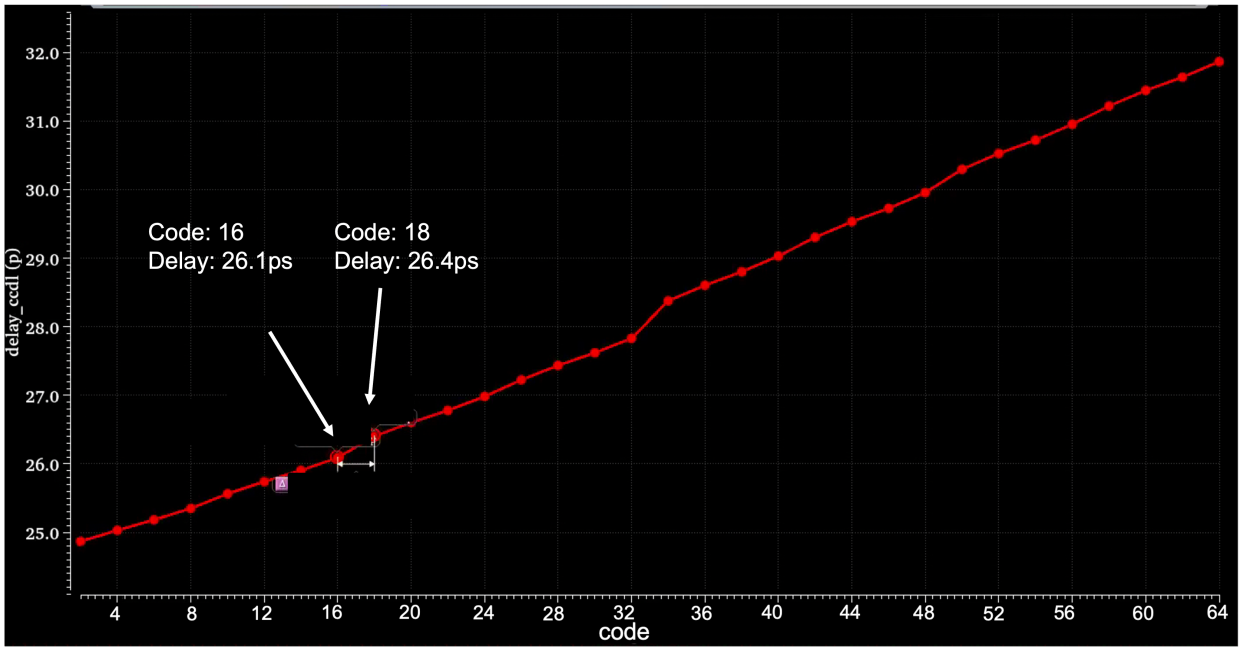


Figure 2.18: Current dac code vs. delay.

Global Buffer

The global buffer is used to drive the datapath loading and the divider to generate the C16 clocks. The resonant buffer could be used with the inductor to resonate out the loading cap to reduce the needed driver size. However, the inductor will take up a substantial area, which may make routing longer and more complicated. For a high-speed circuit, the more compact the layout, the less routing is involved, leading to lower parasitic capacitance and resistance. For the matching of the clocks, the parasitic resistance is desired to be reduced as much as possible. Even if we can match the routing length so that the capacitance matched for all clock phases, it is more difficult to make the routing resistance the same for all. Therefore, to make the overall layout more compact, we place the clock path close to the datapath to minimize the routing parasitic. With less routing capacitance, we can eliminate the need for a resonant buffer and implement the differential pair of inverters with cross-coupled inverter pairs in between as the global buffer.

The schematic of the global buffer is shown in Figure 2.19. The cross-coupled pair inverters are sized much smaller than the main forward inverters so that the main inverter still dominates the signal path. The positive feedback cross-coupled pair adds hysteresis to the outputs that helps sharpen the edge and also couples the two outputs differentially. Positive feedback at the output helps the main inverter drive a larger capacitance load than the typical fanout. However, it also loads the previous stage more, i.e. the previous stage needs more driving power to fight with the cross-coupled pairs.

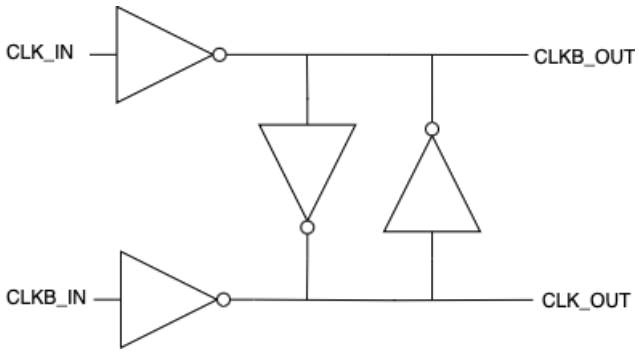


Figure 2.19: Global buffer schematic.

Figure 2.20 shows one of the four pair connections between the DCDL and the global buffer. The inverter chain serves as the buffer in between to be size with the global buffer.

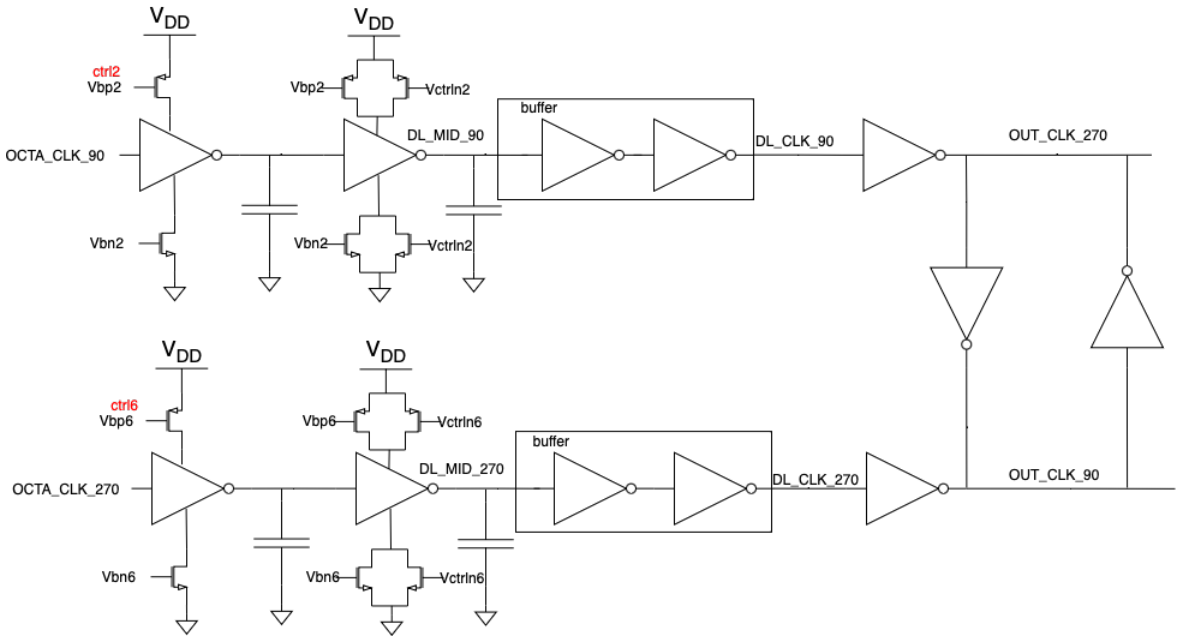


Figure 2.20: One pair of CCDL and global buffer.

2.3 1/16-Rate Clock Distribution

The 16-phase clocks are required for the buffer array and the MLSE slices on the datapath. Four quadrature C^2MOS dividers with the reset synchronizer are implemented to generate a 16-phase clock from the C8 distribution. The 16 phase clocks are buffered with the inverter chain array and feed into the phase rotator and delay line for coarse and fine control of the phase skew. The last stage is the global buffer array to provide enough driving strength for the datapath loading stages.

Divider

The quadrature divider is implemented by connecting the C^2MOS latch back-to-back as shown in Figure. 2.21. The input of the divider is differential clocks which are divided to quadrature output clocks. C^2MOS frequency divider is able to provide large output swing and eliminate the need for the CML-to-CMOS converter. To generate 16 phase clocks, we need 4 quadrature dividers to start dividing in the appropriate order. The reset signal is added to ensure that the divider starts with an initial known state and can be manipulated to have correct orders in the case that k quadrature dividers are used to generate $4k$ phase clocks. In our case, there are four quadrature dividers to divide 8-phase clocks into 16-phase clocks. The reset synchronizer serves this purpose to make sure that the dividers are reset in the correct order to generate the 16 phases correspondingly.

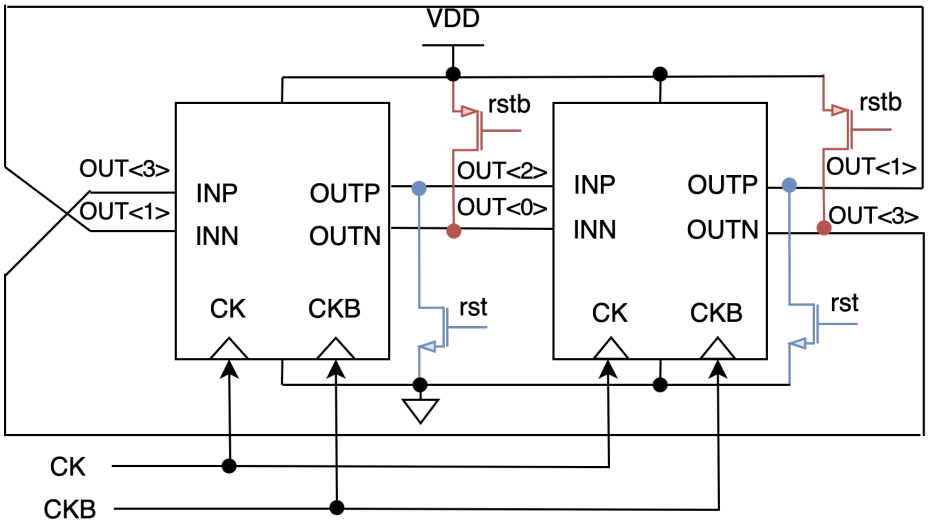


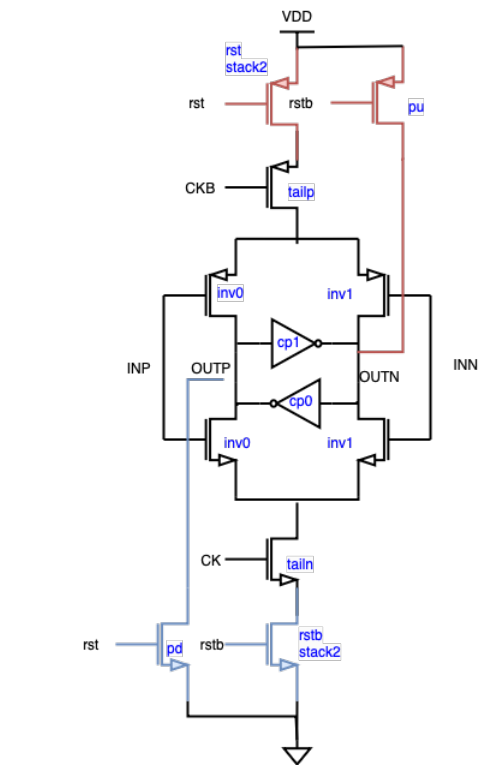
Figure 2.21: Quadrature divider schematic.

In the 160 Gbps transceiver design [31], the divider is followed by a cascade of C^2MOS latches with devices clocked outside the signal path. While measuring the 160 Gbps chip, we

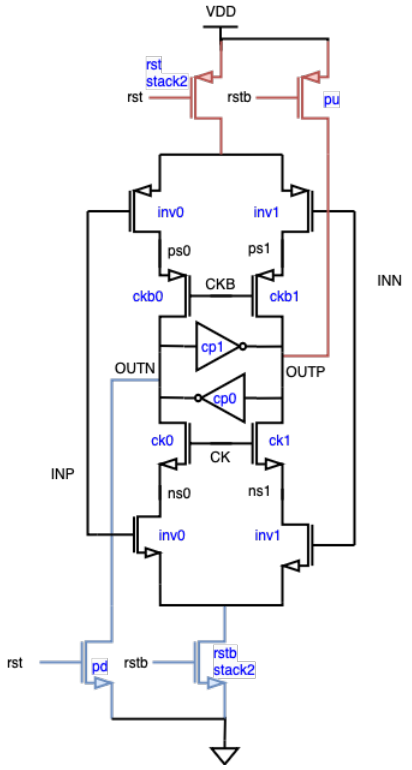
found that the 1.25 GHz digital clock divided from the main clock path fails at the nominal operating speed, but can be observed at lower speed with the tuned current DAC code[31]. The phase skew of the differential phases at the input of the divider is suspected to cause the error. This is the known issue described in [38]. This topology can achieve higher frequency but suffers from charge sharing. The charge-sharing problem induces the kink in divided clocks which is more sensitive to the phase skew of input clocks[38]. Although the kink disappears at very high speeds, it is suspected to cause the failure in the 160 Gbps RX clock path [31].

The original divider structure used in 160Gbps link can support higher speed but is sensitive to clock phase skews. To enhance robustness against clock phase skews, clock devices are moved in the signal path to prevent distortion due to charge sharing for 128Gbps link.

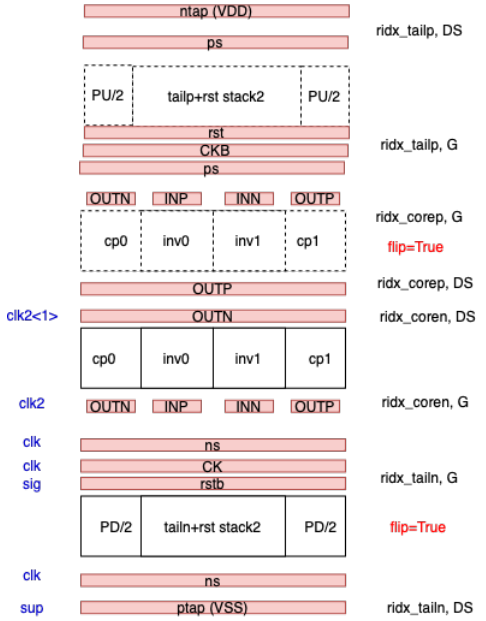
Figure. 2.22 shows the schematics of the original and new dividers and the corresponding floorplan. The layout generator can be modified without significant change to generate new divider layout. The new divider layout is verified to be robust to input clock phase skew.



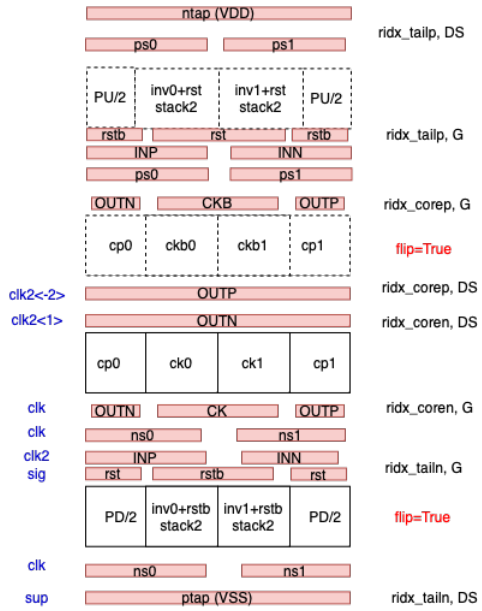
(a) Original schematic.



(b) New schematic.



(c) Original floorplan.



(d) New floorplan.

Figure 2.22: C^2MOS latch schematic and floorplan.

Reset Synchronizer

To generate C16 from C8 clocks, four C^2MOS dividers are needed to generate 4 pairs of the 4 phases clocks in the C16 domain from differential clocks in the C8 domains. Without proper reset signals, the initial state of the divider is not predictable. The reset signals define the initial state of the output of the dividers and thus make the phase alignment between the C8 and C16 domains more controllable. However, the asynchronous reset from the scan chain has an unknown relationship with the clock phases, and some dividers could potentially enter the metastability. In other words, the dividers run independently with the divided clock occurring at two possible states that are 180 degrees apart. The purpose of the reset synchronizer is to generate the synchronized reset signals with the input of the asynchronized reset from the scan chain for each divider in specified order and spacing in time so that the dividers can be reset and started in desired order. All reset signals are synchronized with clock phase 0 to preserve clock phase order after the dividers.

A typical reset synchronizer contains two flip-flops as shown in Figure. 2.23 The first flip-flop synchronizes the reset signal with the clock edge, while the second flip-flop removes the potential metastability during activation of the asynchronous active low reset (AsyncRSTB), where AsyncRSTB goes from high to low. The corresponding output is the assertion of the synchronized reset (SyncRST) that goes to the clock dividers.

The timing diagram of the standard reset synchronizer is shown in Figure 2.24. If the transition of AsyncRSTB from high to low does not violate the timing of the flip-flop, then node mid_rest will be synchronized with the following rising clock edge. However, when the transition of AsyncRSTB from high to low violates the timing of the flip-flop, it will cause the metastability issue in node mid_rst and it is no longer clean synchronized reset. The second flip-flop in Figure 2.23 is used to remove the possible metastability in mid_rst. If the metastability fails to resolve before the next clock cycle, then multiple flip-flops need to be added to remove the metastability. In this case, the depth of the reset synchronizer must be greater than two flip-flops to ensure the generation of the clean synchronized reset shown in Figure 2.25.

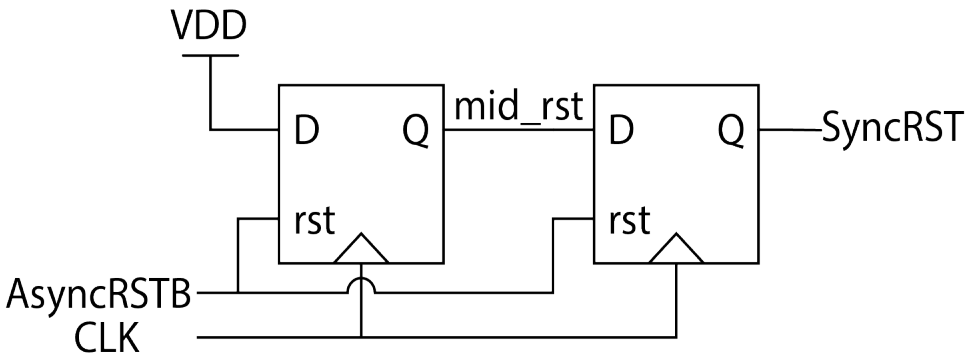
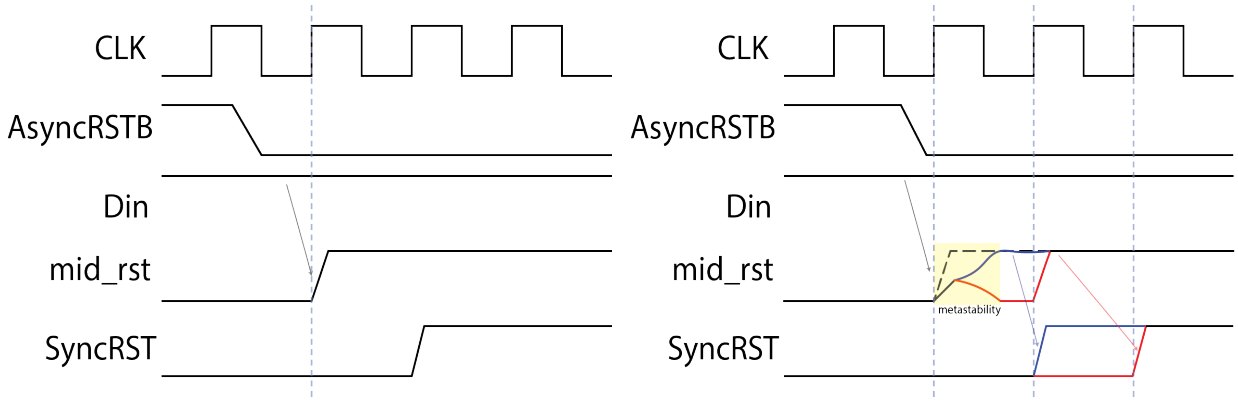


Figure 2.23: Standard unit reset synchronizer.



(a) AsyncRSTB not violate timing.

(b) AsyncRSTB violates timing.

Figure 2.24: Standard reset synchronizer timing diagram.

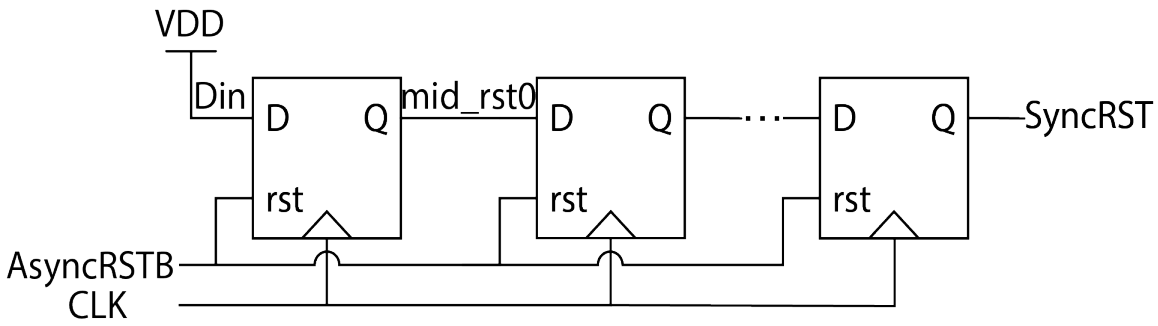


Figure 2.25: Unit reset synchronizer with depth larger than 2.

The reset synchronizer consists of four stages of the unit reset synchronizer to generate four reset signals to the four quadrature dividers shown in Figure 2.26. Each stage requires CLK and CLKB that correspond to the input of each divider. The first stage contains four flip-flops cascaded in series (Figure. 2.25) to ensure that the timing of the last flip-flop is met with the asynchronous reset as the input. The Din of the following stages would be the synchronized reset of the previous unit reset synchronizer. The synchronized reset transition from low to high occurs with at least 4 C8 clock delay (depth of the first stage) after the AsyncRSTB transition from high to low. Therefore, a standard two flip-flop typical unit reset synchronizer (Figure 2.23) in series is enough for metastability and creates 2 C8 clock delays between each reset signal. To ensure a match of loading between the C8 clocks, the CLK/CLKB input of the stage 2 3 are also connected to the dummy load to compensate for the extra flip-flop load in the first stage.

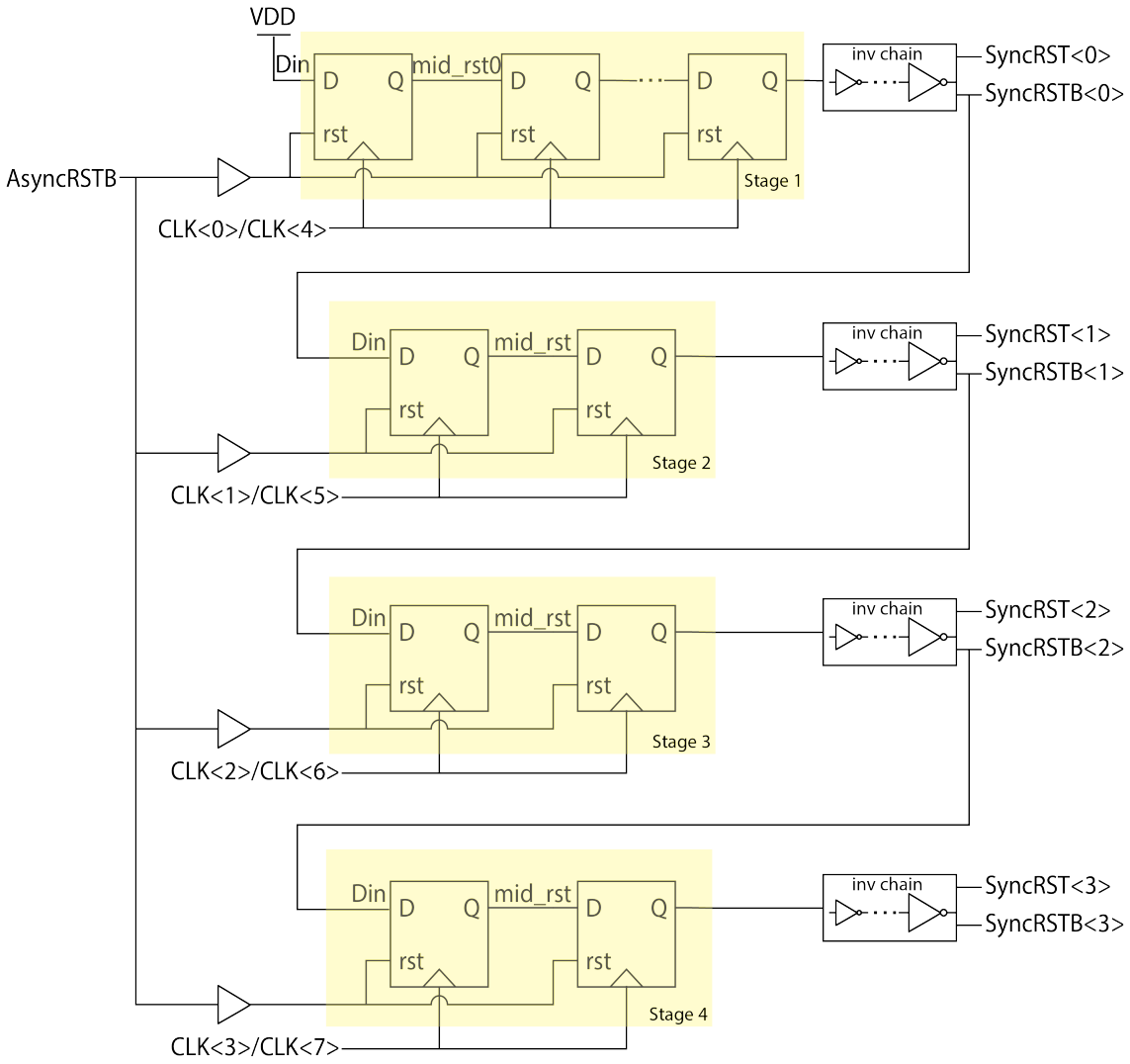


Figure 2.26: 4-stage reset synchronizer for the dividers.

Phase Rotator and Delay Line

The phase rotator and the digitally controlled delay line (DCDL) are implemented as coarse and fine control of the phase errors, respectively. The phase rotator is inspired by [30] to adjust the clock phases due to the common skew among the 16 phases. Instead of using the phase interpolator (PI), which mixes and weighted two phases to get the interpolated phase, we directly apply 4-to-1 mux to choose the clock phase with minimum error that compensate over PVT corner. The simplicity of the 4-to-1 mux is more power and area efficient than the phase interpolator. The phase skew is minimized by relabeling the C16 clocks generated from the quadrature divider to align the C8 clocks, that is, phase n of C16 aligns with phase

n of C8, to compensate the logic delay and the routing delay in the post-layout simulation. Then the 4 phases ($n-k$, n, $n+k$, $n+2k$) of the 16 phases centered on the default phase n are chosen to cover the PVT corner. The detailed schematic is shown in Figure 2.27, where n is the index of the clock phase ($0 < n < 15$) and k is the select phase step to cover all variation of PVT. The delay line after the phase rotator has a linear relationship between the delay and the digital code that can be used to fine-tune the phase error between the C16 clocks and the alignment between C8 and C16.

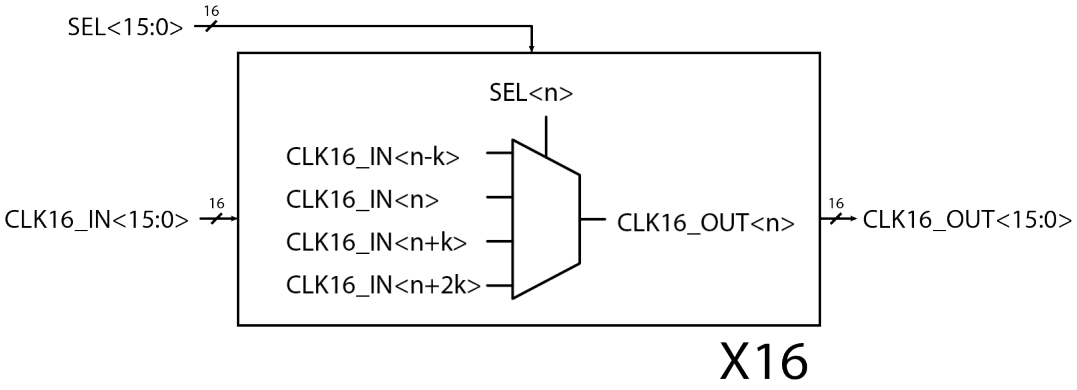


Figure 2.27: Phase rotator schematic.

Digitally Controlled Delay Line (DCDL)

The DCDL is basically the same as the one implemented in the C8 clock domain. Since the speed is twice slower than the C8 clock, the duty cycle due to the NMOS and PMOS driving strength is relieved. The duty cycle correction is therefore not included in the C16 CCDL for simplicity. The number of phases that must be tuned is double in the C16 domain, and 16 DCDLs and current DACs dedicated to each phase are needed. The tuning range needs to cover the phase step in the phase rotator (at least k UI is needed) and the DCDL step is also linear.

Global Buffer

The global buffer structure is the same as the C8 global buffer shown in Figure 2.19. The total number of the global buffer is doubled to deal with twice clock phases in the C16 clock domain.

Chapter 3

MLSE and CDR Overview

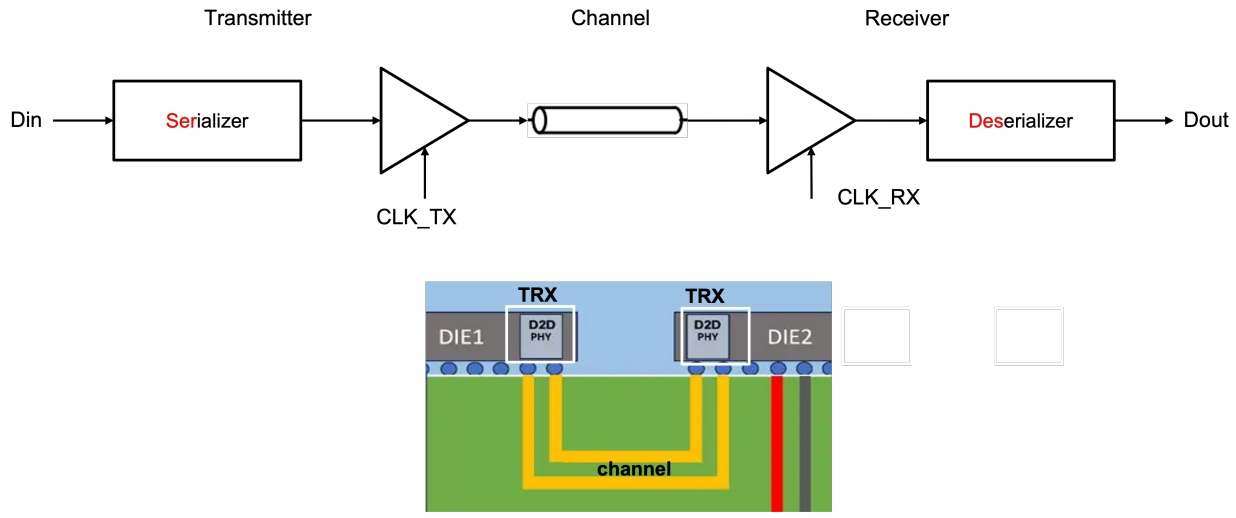


Figure 3.1: High-speed link block diagram[25].

3.1 Inter-symbol Interference (ISI) and Equalization

Inter-symbol Interference (ISI)

Inter-symbol interference (ISI) is a form of distortion of a signal consisting of a sequence of data symbol where one symbol is affected by other symbols in the sequence. The impulse digital input 0, 0, ...,0, 1, 0, ...0 before the serializer in the TX ideally converts into the 1-UI square pulse at the input of the receiver. However, the pulse response will smear out due to the finite bandwidth of the TX output driver and the low-pass channel between the TX and the RX. In Figure. 3.2, we take the pulse response of 200Gbps PAM4 TX in [47] as an

example. We define the cursor as $h[0]$ in the pulse response. The pre-cursor ($h[-1]$) and the post-cursor ($h[1], h[2], ...$) are the integer multiples of the UI before and after the cursor of the pulse response, respectively. ISI of the pulse response includes both pre-cursor and post-cursor. For a sequence of digital input data $Din[n]$, the input voltage received at the input of the RX $V_{in}[n]$ can be derived by convolution of the input data and the cursor and ISI of the pulse response.

$$v_{in}[n] = \Sigma_{k=-\infty}^{\infty} D_{in}[n-k]h[k] \tag{3.1}$$

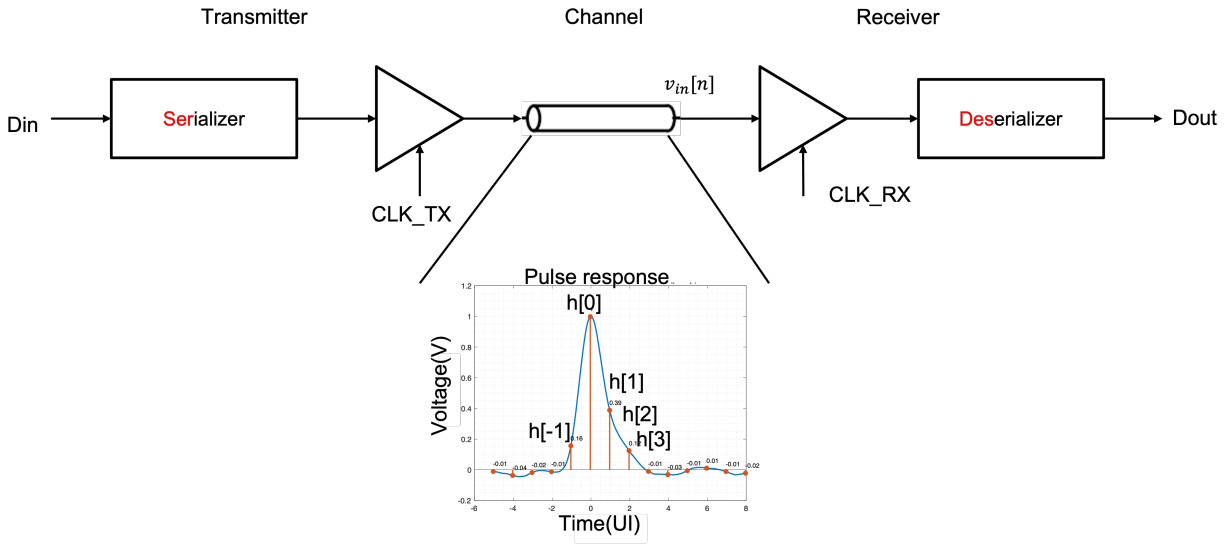


Figure 3.2: High-speed link with low-pass channel.

Equalization in the Receiver

With the ISI created by the bandwidth-limited output driver and channel, the eye opening is reduced or even close in the eye diagram (Figure. 3.3. To cancel out the ISI, an equalizer is needed to open the eye before the sampler in the receiver. As previously described in Chapter 1, the popular equalizer is summarized in Figure. 1.9. CTLE and DFE are commonly used in RX. A linear CTLE is usually used to cancel the pre-cursor and the long tail but can cause noise enhancement and may not be sufficient when the speed scales up. DFE is good at canceling post-cursor without noise enhancement, but suffers from feedback loop timing constraint that limits the speed it can achieve.

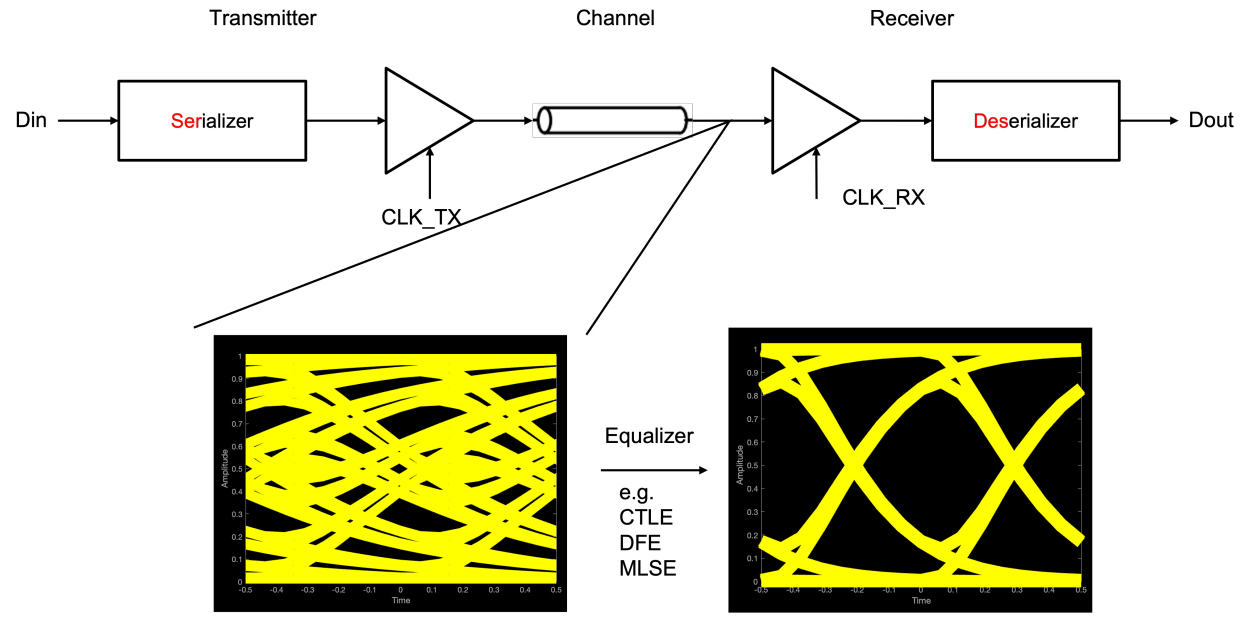


Figure 3.3: High-speed link with equalizer.

Maximum Likelihood Sequence Estimation (MLSE)

Introduction

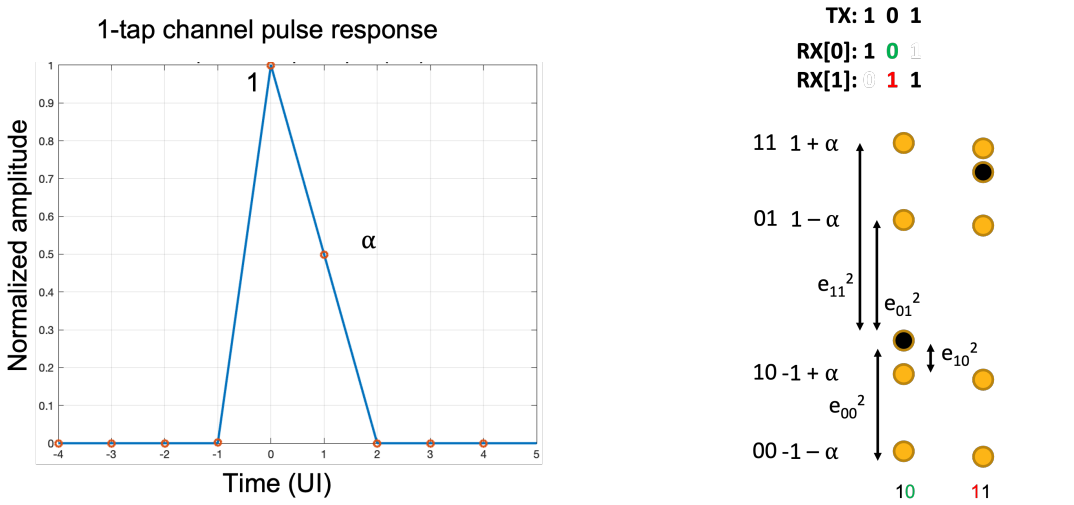
Maximum Likelihood Sequence Estimation (MLSE) uses ISI energy to recover a sequence of symbols. Take, for example, an ideal channel pulse response with 1-tap post-cursor (Figure. 3.4a). The received symbol would be compared with all possible data levels and then be decoded with the data level that introduces fewer errors.

MLSE provides better error statistics than conventional equalizers and is robust to thermal noise. The conflicts between the symbols could occur with noise; for example, the middle bit is decoded as 0 while 1 is decoded in the next window. MLSE gives optimal performance with better error statistics, but is difficult to implement due to its high complexity. If the complexity can be reduced and reused in the clocking scheme, the MLSE would be promising to enable further speed scaling. The pros and cons of the RX equalizer used in RX are summarized in Table 3.1.

Conflict Resolution: Single Window[31]

Conflicts between windows of MLSE are traditionally solved by Viterbi algorithms, which also introduce a feedback loop. A single-window strategy [31] is used to resolve conflicts that enable the purely feedforward system.

First, the window length is defined as the number of UIs over which we accumulate errors. If the window length is 2, that is, we accumulate the error over 2 UIs, we can have



(a) One-tap ideal pulse reponse.

(b) MLSE decoding scheme.

Figure 3.4: MLSE illustration.

	Pros	Cons
CTLE	Adaptive	Noise enhancement
DFE	No noise enhancement	Feedback loop timing constraint
MLSE	Optimal performance	High complexity

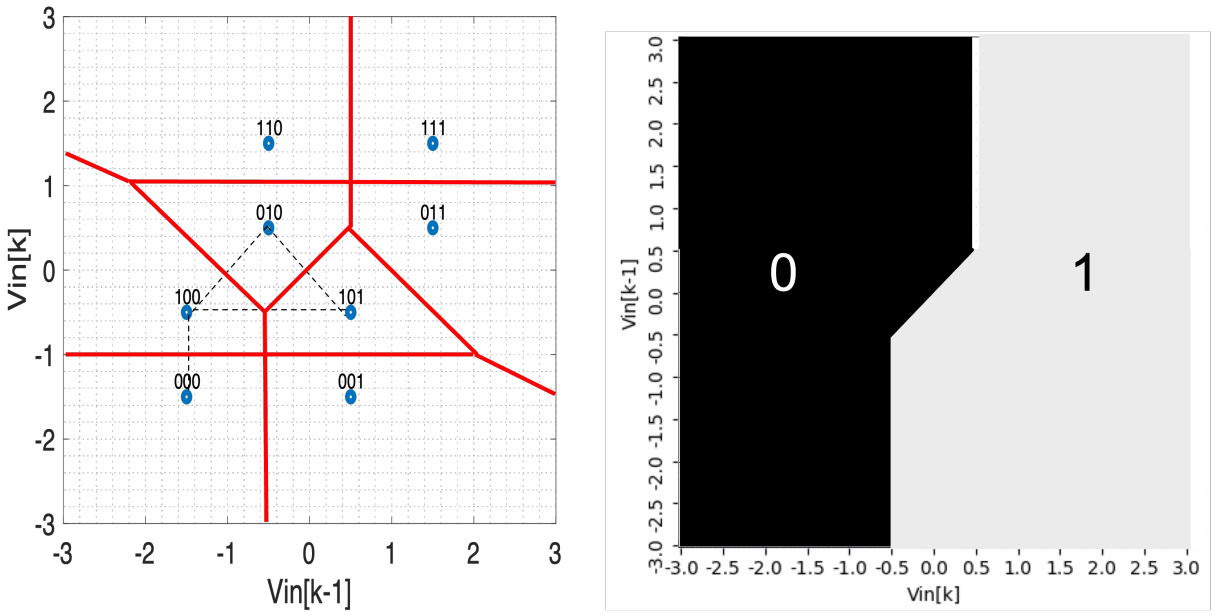
Table 3.1: RX equalizer comparison table.

three consecutive bits, which translates to two consecutive input voltages $V_{in}[k-1:k]$. We can map the two consecutive input voltages in the 2-dimensional graph with $V_{in}[k]$ as the x axis and $V_{in}[k-1]$ as the y axis shown in Figure. 3.5a. The previous input voltage $V_{in}[k-1]$ can be decided by the first two bits ($D[k-2:k-1]$) which have 4 levels $(-1-\alpha, -1+\alpha, 1-\alpha, 1+\alpha)$ due to the 1-tap post-cursor (Figure. 3.4a). Similarly, the current input ($V_{in}[k]$) can be mapped by the last two bits ($D[k-1:k]$). Assuming that the ratio of the first post-cursor to cursor is 0.5 (α=0.5), the input voltage corresponding to 3 consecutive digital inputs can be summarized in Table. 3.2. The way to decide the border of the two dots on the graph is to draw the orthogonal line that gives the largest error margin. If we draw the orthogonal line between every two dots on the graph, we can get the 2D plot with 8 regions.

It is difficult to implement the partition of 8 regions for a window length of 2. However, if we only consider the last bit ($D[k]$) which represents the current bit, then we get a simple two partitions with three piecewise linear segments. This is called a single-window strategy.

D[k-2:k]	D[k-2:k-1]	Vin[k-1]	D[k-1:k]	Vin[k]
000	00	-1.5	00	-1.5
001	00	-1.5	01	0.5
010	01	0.5	10	-0.5
011	01	0.5	11	1.5
100	10	-0.5	00	-1.5
101	10	-0.5	01	0.5
110	11	1.5	10	-0.5
111	11	1.5	11	1.5

Table 3.2: TX output and RX input mapping.



(a) Symbol mapping of D[k-2:k].

(b) single-window algorithm

Figure 3.5: MLSE mapping with window length of 2.

The left part of the partition stands for $D[k] = 0$ while the right part of the partition stands for $D[k] = 1$.

Compare to DFE

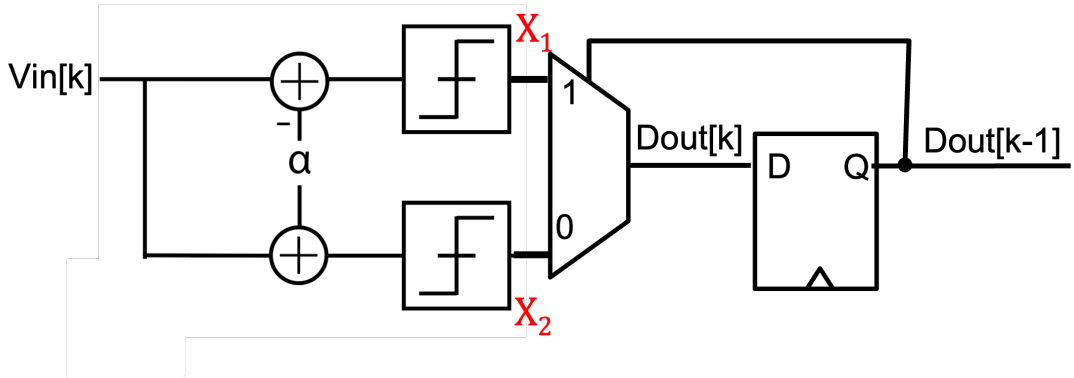


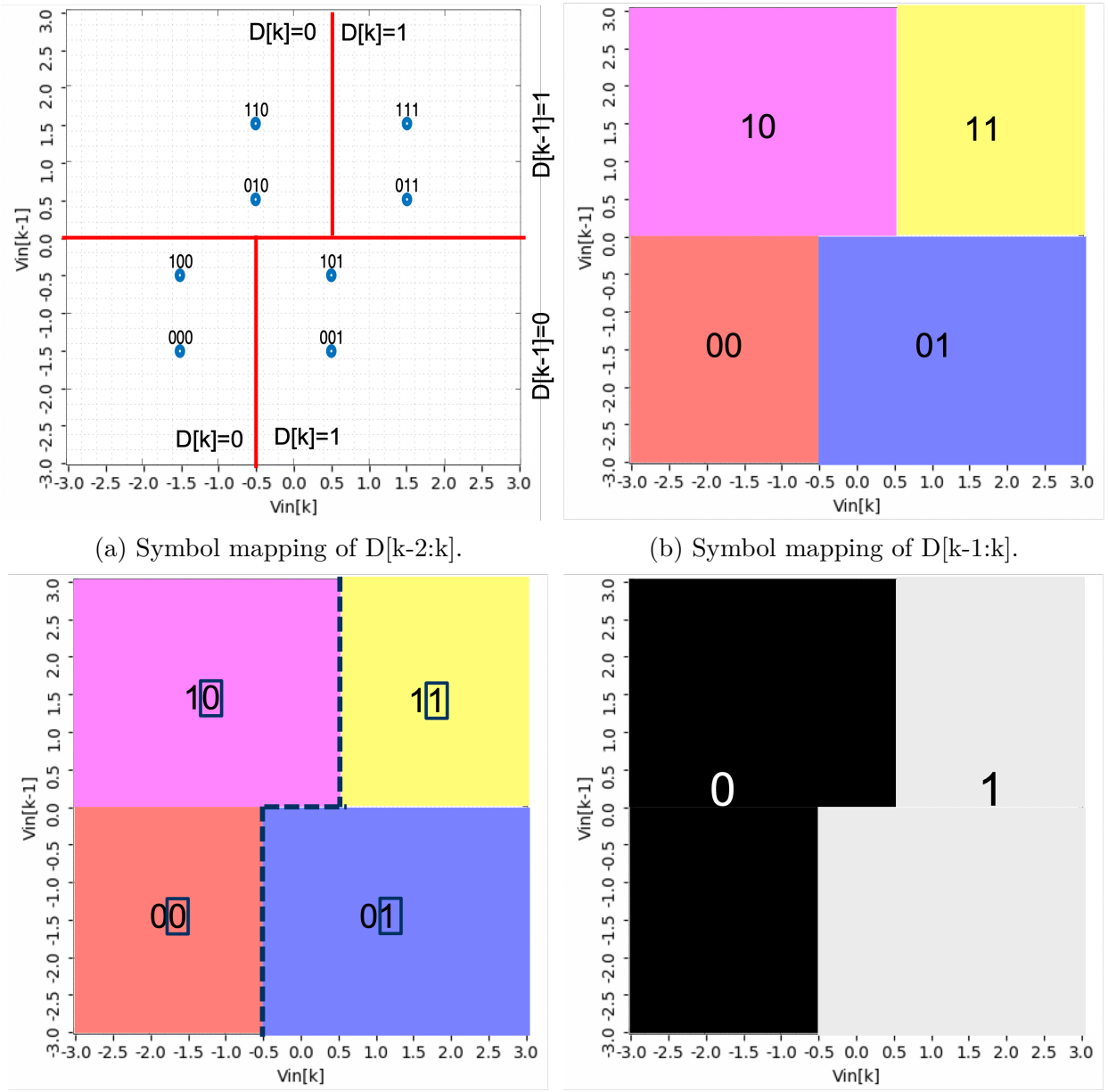
Figure 3.6: DFE circuit implementation.

The conventional one-tap loop-unrolled DFE circuit diagram is shown in Figure 3.6. $D_{out}[k]$ is decoded by different offsets $-\alpha$ and α depending on the previous decoded bit $D_{out}[k-1]$.

Although the loop-unrolling removes the time for the slicer to resolve from the feedback loop. There is still a timing constraint of the DFE as follows.

$$T_{CK-Q} + T_{setup} + T_{MUX} < 1UI \tag{3.2}$$

Assuming we have the same pulse response as in the example in Figure. 3.4a, the similar analysis that we did for the MLSE single-window strategy can be applied for the one-tap loop-unrolled DFE summarized in Figure. 3.7. First, we start with the 8 symbols considering the consecutive input voltage $V_{in}[k-1:k]$ in Figure. 3.7a. Instead of partitioning between each symbol, we would like to partition on the basis of $D[n-1]$ first. The symbols are then divided into two groups: the upper half ($D[k-1]=1$) and the lower half ($D[k-1]=0$). The horizontal line $V_{in}[k-1]=0$ is the border that gives the largest error margin for the two groups. Within each group, we want to find the line to partition based on $D[k]$. The results are the two vertical lines $V_{in}[k]=-\alpha$ and $V_{in}[k]=\alpha$ for $D[k-1]=0$ and $D[k-1]=1$, respectively. With these partitions, the symbol mapping for $D[k-1:k]$ is redrawn in Figure. 3.7b. If we only look for the last bit, as we did in the MLSE single-window strategy, we can obtain the partition of piecewise linear segments that match the circuit implementation shown in Figure. 3.6.



(a) Symbol mapping of D[k-2:k].

(b) Symbol mapping of D[k-1:k].

(c) Symbol mapping of D[k].

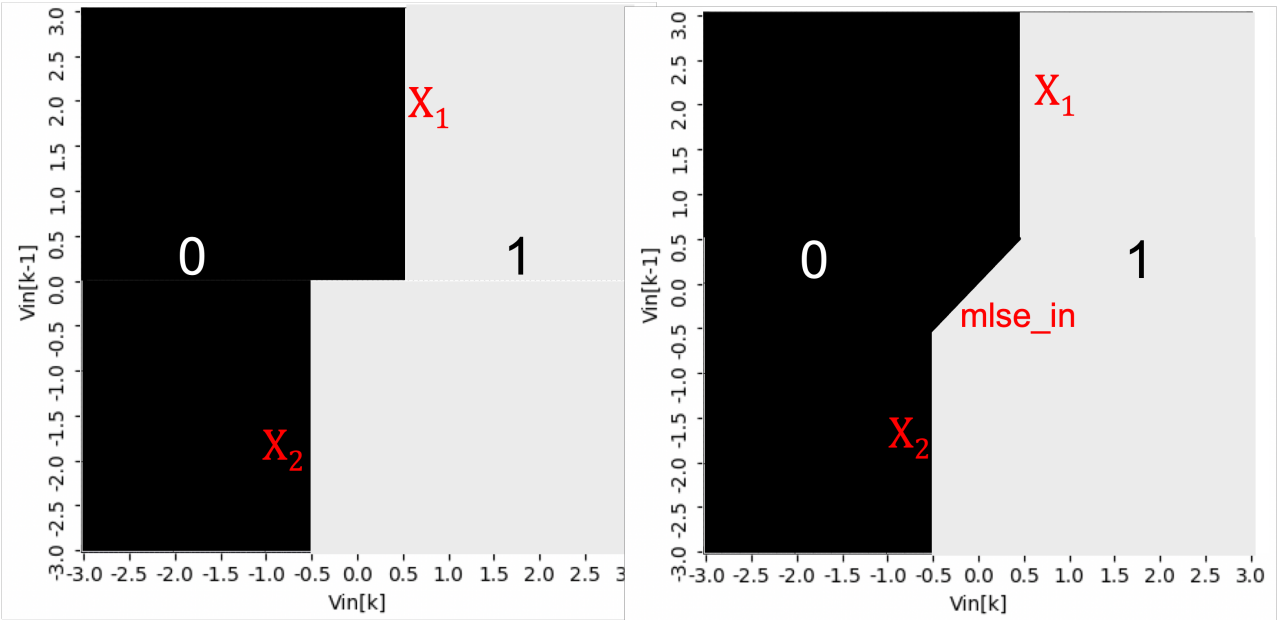
(d) Mapping of D[k].

Figure 3.7: DFE mapping with window length of 2.

Circuit Implementation

From the previous analysis for MLSE and DFE using consecutive symbols with an ideal one-tap channel, we can construct the MLSE circuit based on the one-tap loop-unrolled DFE shown in Figure. 3.6. Observing the mapping for decoding $D_{out}[n]$ in Figure. 3.8, two vertical segments (X_1, X_2) are identical for both equalization techniques. Although MLSE has additional segments $mlse\_in$ to deal with the case that both $V_{in}[k]$ and $V_{in}[k-1]$ are within $\pm\alpha$. In other words, MLSE gives better statistical performance as it can better separate the symbols 010 and 101 in Figure. 3.5b. To implement piecewise linear segments, we can start by defining the variables $X_1, X_2, mlse\_in$.

$$\begin{aligned} X_1 &= V_{in}[k] > \alpha \\ X_2 &= V_{in}[k] > -\alpha \\ mlse\_in &= V_{in}[k] > V_{in}[k-1] \end{aligned} \tag{3.3}$$



(a) DFE. (b) MLSE.

Figure 3.8: DFE/ MLSE mapping.

If we define the slicing function $slice(x)$ [31] to mimic a comparator or slicer function as follows, we can rewrite the variables in Equation 3.3.

$$slice(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases}$$

$$\begin{aligned} X_1 &= slice(V_{in}[k] - \alpha) \\ X_2 &= slice(V_{in}[k] + \alpha) \\ mlse\_in &= slice(V_{in}[k] - V_{in}[k-1]) \end{aligned} \tag{3.4}$$

X_1	X_2	$mlse\_in$	D_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

Table 3.3: Truth table for MLSE decoder.

The truth table with input $X_1, X_2, mlse\_in$ and output D_{out} is derived from Table 3.3. If we group the don't care terms with $X_1\bar{X}_2$ and $\bar{X}_1 X2 mlse\_in$, we can get to the expression for the decoder.

$$D_{out} = X_1 + X_2 \cdot mlse\_in \tag{3.5}$$

The circuit implementation shown in Figure. 3.9 is realized by using minimal circuitry consisting of a few comparators, adders, inverter, NAND gate, and register. An additional 2-tap FIR path that generates $mlse\_in$ is required compared to the DFE circuit in Figure. 3.6.

3.2 CDR Overview

In an embedded clock system, the receiver needs to recover the clock from the input data in order to sample the data in the optimal position. The optimal sampling point should give the best bit error rate (BER), which means that it should be the sampling point that can best differentiate the data level so that the error margin for decision making on the datapath can be maximized in order to reduce the BER. In Figure. 3.10, if there is no feedback CDR loop, then the receiver clock (CLK_RX) could be drifting in frequency and phase and sample the data input early or late compared to the optimal sampling point, which gives the largest opening of eyes.

The CDR locking performance is evaluated using Markov chains assuming that the next step depends only on the previous state. The single-variable Markov chain and the multiple-variable Markov chain are used to calculate the locking position and the jitter distribution

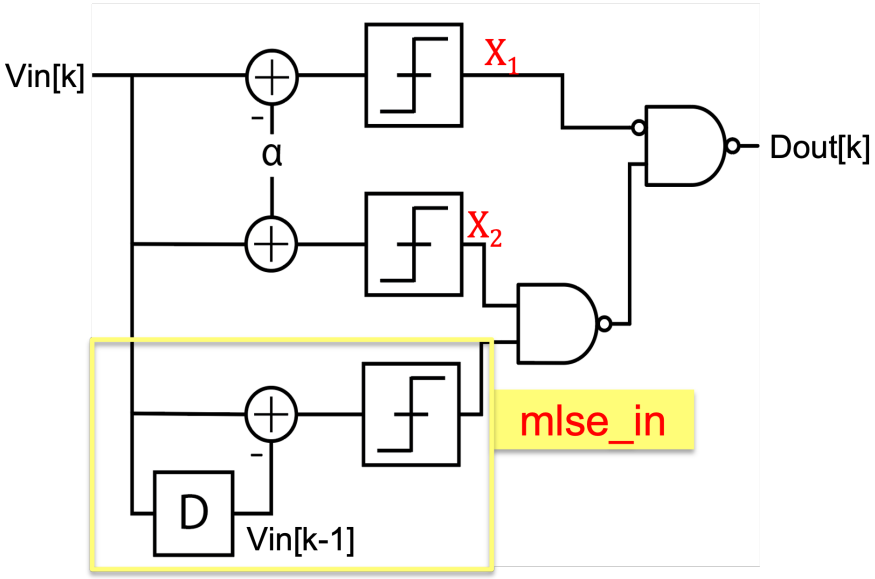


Figure 3.9: MLSE circuit implementation (adapted from Figure. 2.14 in [31]).

while locking. By sensing the difference between the current clock edge and the locking point, CDR can help bring the sampling point to the locking point, which is ideally the optimal sampling point that maximizes the timing margin. The other consideration is that the jitter of the recovered clock should be minimized as much as possible.

Oversampling vs. Baud-rate CDR

The type of CDR can be categorized as oversampling or baud rate depending on how many data samples in 1UI are needed for the CDR algorithm. Baud-rate CDR only takes one data sample in 1UI to recover the clock, while oversampling CDR needs extra data samples for the same purpose.

Bang-bang CDR (Alexander) is a commonly used oversampling CDR that requires additional edge sampling points. When there is a data transition between $D[n]$ and $D[n+1]$, the polarity of the edge sample $E[n]$ can be used to detect whether the current clock is early or late. If the edge sample and the current data sample have the same polarity, the clock is EARLY and the phase detector output (PD_{out}) will be assigned 1. On the other hand, the clock is LATE and phase detector output will be assigned -1 if the edge sample and the next data sample have the same polarity.

```
if E[n]==D[n]:
        PD_out = 1 (Clock is EARLY)
else:
        PD_out = -1 (Clock is LATE)
```

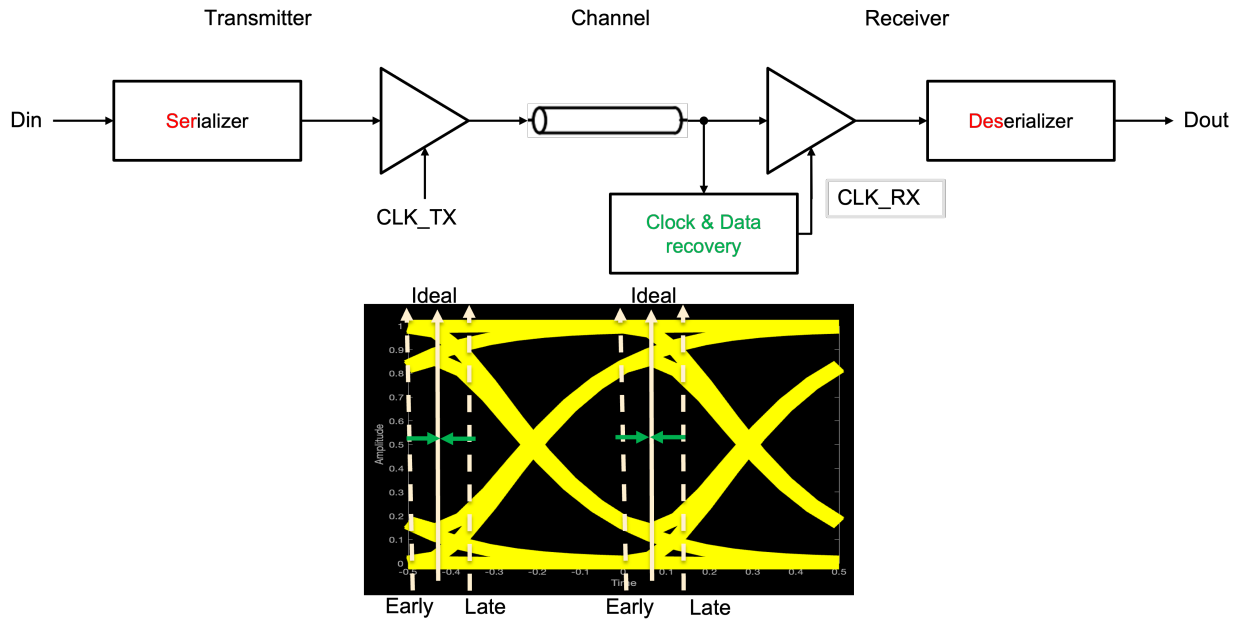


Figure 3.10: Function of the CDR.

This CDR algorithm is more robust to pulse response variation, but requires additional slicer and clock phase to produce the edge sample, which is expensive at data rates close to the limit of the process technology.

The baud-rate CDR eliminates the need for the edge sample and is thus preferred in the speed target close to the limit of the process technology. The power efficiency is attractive for people to explore baud-rate CDR over oversampling CDR in recent years. However, without the additional edge samples, the information for recovering the clock is also reduced and has more dependency on the pulse response. Our goal is to push the speed limit of the process technology; therefore, baud-rate CDR is chosen over oversampling CDR. Multiple state-of-the-art baud-rate CDRs will be described in the following sections.

Mueller-Muller CDR

Timing Recovery Theory[35]

The most popular baud-rate CDR is the Mueller-Muller CDR proposed in 1976 [35]. In the original paper, fast-converging timing recovery methods with input signal sampled at baud rate were investigated. Two timing functions are discussed for the timing recovery extraction related to the pulse response. The results of [35] are summarized below.

First, we consider a continuous signal received by the receiver front-end as $V_{in}(t)$ with a sequence of random data symbols $D_{in,k}$ transmitted in data rate $\frac{1}{T}$ with overall pulse response $h(t)$.

$$V_{in}(t) = \Sigma_k D_{in,k} h(t - kT) \tag{3.6}$$

Suppose that the signal is sampled at instants $t = \tau + mT$. The timing function $f(\tau)$ will determine the transfer characteristic of the CDR control loop. The resulting steady-state timing phase (τ_{ss}), which is also called the locked sampling phase, will force the timing function to 0.[35] as in Equation. 3.8.

$$f(\tau_{ss}) = 0 \tag{3.7}$$

The timing function $f(\tau)$ is limited to linear combinations of the samples (ISI) of the pulse response:

$$f(\tau) = \Sigma_{i=1}^{L} u_i \tau_i, \text{ where } \Sigma_i u_i = 1 \tag{3.8}$$

Among all possible timing functions in Equation 3.8, there are two special cases as shown in Equation. 3.9 is analyzed. Type A is used in most published papers using Mueller-Muller CDR. It locks when the first pre-cursor and first post-cursor are the same (Figure. 3.11. Type B is a zero forcing of the first post-cursor h_1. This will take the first zero crossing after the main pulse as a timing reference (h_1). Although only type A and B are shown, the linear combination of ISI is forced to 0 (Equation. 3.8) would be the generalized form of the Mueller-Muller CDR.

$$\begin{aligned} &\textbf{Type A: } f(\tau) = 0.5(h_1 - h_{-1}) = 0.5[h(\tau + T) - h(\tau - T)] \\ &\textbf{Type B: } f(\tau) = h_1 = h(\tau + T). \end{aligned} \tag{3.9}$$

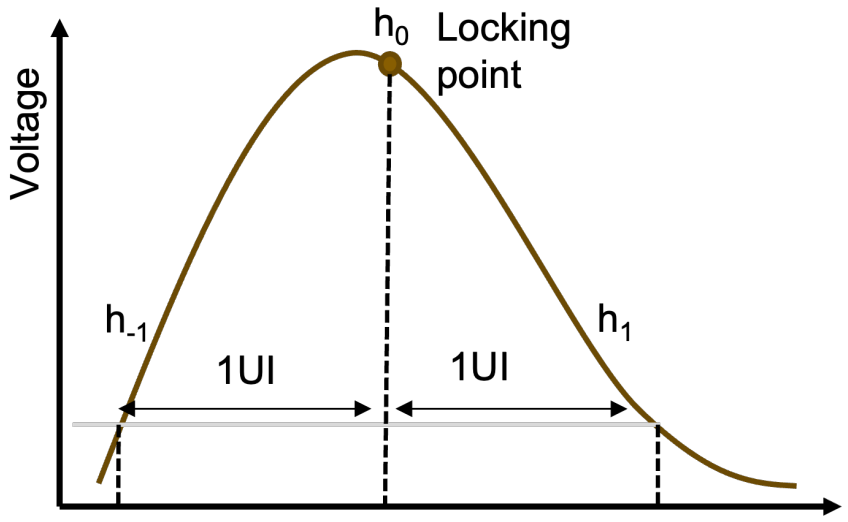


Figure 3.11: Type A Muller-Mueller CDR.

Variants of Mueller-Muller CDR

Sign-sign Mueller-Muller CDR proposed in [3] is a simplified version of the Type A timing function in Equation. 3.9. The 1-bit sign-sign phase detector is used to replace the original quantizer proposed in [35]. The desired reference level ($VREF$) which is also called the data level ($dLev$) that is adapted to be the sample at the desired sampling phase. The phase detector compares the current sample versus $VREF$ to generate the error sample ERR_n. By correlating ERR_n with D_{n-1} and ERR_{n-1} with D_n, we can obtain the information of h_1 and h_{-1} respectively.

The hardware overhead of this method includes additional error samplers[3] and $VREF$ adaptation. The error samplers are also used in the datapath for DFE but reusing the error sample in clock recovery requires the error sampler to be active at all time, which is not a requirement for the bang-bang PD.

The timing function will change with the use of DFE, and thus move the sampling position accordingly. Since the DFE zero forces the first post-cursor tap (h_1) to zero, the Type A timing function in Equation. 3.9 reduces to zero first pre-cursor criteria as Equation. 3.10. This moves the sampling point to the left (lead) of the original pulse response peak. The shift can be reduced with higher pre-emphasis at a price of increasing the second pre-cursor ISI (h_{-2}) [41]. Without the pre-cursor undershoot, the slope near $h_{-1} = 0$ would be low, suggesting a small PD gain and susceptible to noise and degrading jitter on the recovered clock.

$$f(\tau) = h_{-1} = h(\tau - T) = 0 \tag{3.10}$$

To solve the left-shifting issue caused by the zero-forcing of DFE, gain and offset adjustment to the timing function is proposed in [11] [10]. The unequalized MM CDR [11] adds a digital offset to the phase-error accumulator to move the locking point to the unequalized position (locking point before applying DFE). The added offset is programmable to trade off between a larger pre-cursor amplitude (h_{-1}) and a better locking point. The gain of the digital offset depends on the transition density of the data pattern. The weight-adjusting MM CDR [10] adjusts the weights of EARLY and LATE of the phase detector, α and $1-\alpha$, to shift the locking point with constant gain of α. Only one error sampler is used in [10] to reduce hardware overhead and power consumption at the price of reducing the update rate of the phase detector. In the sign-sign CDR case, the phase detector updates at transition from 0 to 1 or from 1 to 0. With only one error sampler, the phase detector updates only when the data remain 1 for two consecutive samples. The offset is adapted to maximize $h_0 - h_{-1}$ using the maximum eye tracking algorithm (MET). Authors in [14] also use the weight-adjusting idea, but add an offset to the reference level of the error sampler and adopt an asymmetric weighting according to the transition types.

Recent published research paper attempts to address the poor locking point issue by applying specific pattern filtering according to different transitions in the multilevel system[5, 13]. Obtaining more information from the multiple transitions and the collected histogram of consecutive samples could help determine a better weighting for the phase detector. However, these methods require an estimate of the pulse response and the pre-cursor/ post-cursor $h[k]$

and adjust the CDR locking position based on the estimate. Therefore, the locking position of the CDR will be highly dependent on the accuracy of the pulse response estimate.

The major drawback of Mueller-Muller CDR, generalized to combination of pulse-response coefficient, is that the quality of the locking point depends on the shape of the pulse response. For Type A timing function in Equation. 3.9, the locking point will be nonideal if the pulse response waveform is asymmetry at the main peak of the pulse. Furthermore, phase wandering could occur if there are multiple clock phases that can satisfy $f(\tau) = 0$. That is, it could possibly lose lock if the locking point is not unique.

dLev Maximization CDR

To solve the drawbacks mentioned above of the Mueller-Muller CDR, the dLev maximization CDR proposed in [6] finds the unique locking point utilizing the 1UI integrated pulse response resulting from the integration-and-reset frontend in the datapath. The merit of integration-and-reset frontend compared to the continuous topologies is its superior energy efficiency. The integration-and-reset frontend serves as the 1UI integrator to the original pulse response. If the received signal is filtered by an integration-and-reset frontend, the frontend output will be maximized when the integration window perfectly overlaps with the incoming pulse, which translates to a single locking point, even when the input pulses have wide flat regions. Take the extreme case for an example: if the input pulse response is an ideal 1UI square pulse, the Type A Mueller-Muller CDR could lock anywhere within the pulse. By passing the pulse to the 1UI integrator, the pulse response becomes the triangle waveform with the unique maximum point.

To lock the unique maximum point after the integration and reset front-end, the data level ($dLev$) used in the equalizer adaption can serve as the indicator of the sampled value of the integrated pulse response. In the setting in [6], $dLev$ tracks h_0 and can be the estimate of pulse response sampling at a certain phase. Instead of maximizing the integrated pulse response, the $dLev$ that serves the same purpose as $VREF$ in the weight-adjusting Mueller-Muller PD [10] is the value to maximize in the CDR loop.

Gradient descent is used to find the maximum locking point with a small dithering step $\pm\Delta$ added to the recovered clock. With the known dithering polarity, we can extract the slope of the current $dLev$ versus phase with the error sampler output. Similarly to [10], only one $dLev$ ($VREF$) is used by filtering the $+1$ symbol specifically to reduce the number of the error sampler.

To implement dLev maximization CDR, an additional dithering generator is needed, and the dithering added on top of the recovered clock will increase the jitter and reduce the horizontal eye opening. Another consideration is that the CDR loop has to be slower than the dLev loop in order to get the correct error sample with converged dLev value.

3.3 Proposed Baud-rate Hybrid CDR algorithm

The baud-rate CDR algorithms mentioned in Section 3.10 either have dependency on the shape of the pulse response or add extra dithering jitter in the steady state. To address these issues and maintain power efficiency, the hybrid CDR algorithm is proposed with the following characteristics.

- Baud-rate operation for power efficiency
- Unique locking point robust to pulse shape variation
- Remove extra dithering jitter when locked
- Minimized hardware overhead

The CDR algorithm described in Section. 3.10 has common considerations, that is, trying to reuse the existing information and hardware in the datapath to reduce hardware overhead. Mueller-Muller CDR [3][11][10] and data level maximization CDR [6] are all incorporated with DFE and error sampler for equalization adaptation. In this work, we have feedforward MLSE described in section 3.1 replace the DFE as MLSE breaks the speed bottleneck of the feedback loop and has the superior error statistics. The main disadvantage of MLSE is its high complexity. The single-window strategy simplifies the implementation of the MLSE algorithm but is still more complicated than the loop-unrolled DFE. More specifically, additional summer and slicer is required to generate $mlse_{in}$ in Figure. 3.12 which is not required in DFE. It is desirable to reuse the extra $mlse_{in}$ signal in the CDR algorithm to help extract clock information. Using both the complexity on the datapath and the clockpath, MLSE would be an even more attractive option. To further enhance power efficiency, the integrate-and-reset latch similar to [6] is used in the datapath.

$mlse_{in}$ Update

The MLSE decoder uses the comparison results between the current and previous data sample, which is annotated as $mlsein$. We want to find out if it is possible to utilize this information to find the locking position for the CDR. The integrated pulse response is restricted to one-tap post-cursor pulse response, which is reasonable for the XSR channel.

To enable baud-rate CDR recovery with integrating front end, a pattern filtering technique for baud-rate CDR recovery was proposed in [24, 23]. By filtering certain patterns, the transition information can be used to provide phase information for the symmetrical pulse response. In this work, we need to deal with the asymmetry pulse response due to the one-tap post-cursor. Another constraint is that we want to reuse the complexity (comparison between two consecutive inputs) that originated from the MLSE equalizer. To see whether we can actually reuse the comparison between two consecutive inputs, different combinations of data sequences are swept and analyzed by the transition diagram. There are four possible

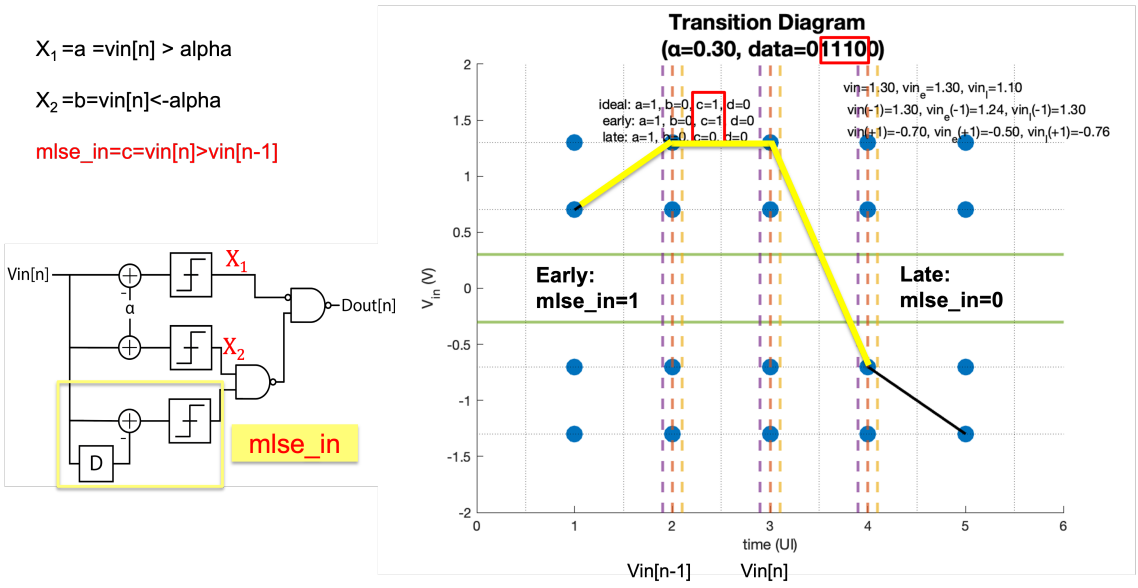


Figure 3.12: Transition diagram analysis for 011100 pattern.

levels in the transition diagram due to the one-tap post-cursor assumption for the pulse response. In Figure. 3.12, we found that the early and late information can be identified with the data pattern consisting of 011100 where $mlse_{in}$ is negative when the clock is late and positive when the clock is early. All other three data sequences were checked in Figure. 3.13 and confirm that the $mlse_{in}$ signal can identify EARLY and LATE clock with pattern $D[n-2:n+1] = 1110$.

The above finding can be formulated to the phase detector update equation below.

```
if D[n-2:n+1] == 1110:
   PD_out = sgn(Vin[n] - Vin[n-1])
```

To determine the locking point of the update equation, we can analyze $mlse_{in}$ with pattern filtering and set the expectation value to 0 and find the locking condition.

$$\begin{aligned} mlse_{in}[n] &= v_{in}[n] - v_{in}[n-1] \\ &= \Sigma_{k=-\infty}^{\infty} D[n-k](h[k] - h[k-1]) \\ &= h[2] - 2h[-1] + h[-2]\Sigma_{k\neq -1,0,1,2}^{\infty} D[n-k](h[k] - h[k-1]). \end{aligned} \tag{3.11}$$

Taking the expectation value of $mlse_{in}$ with pattern filtering $D[n-2:n+1] = 1110$ and assuming that the data sequence is random and independent, we have

$$\begin{aligned} E[mlse_{in}[n]] &= h[2] - 2h[-1] + h[-2] \\ &= 0. \end{aligned} \tag{3.12}$$

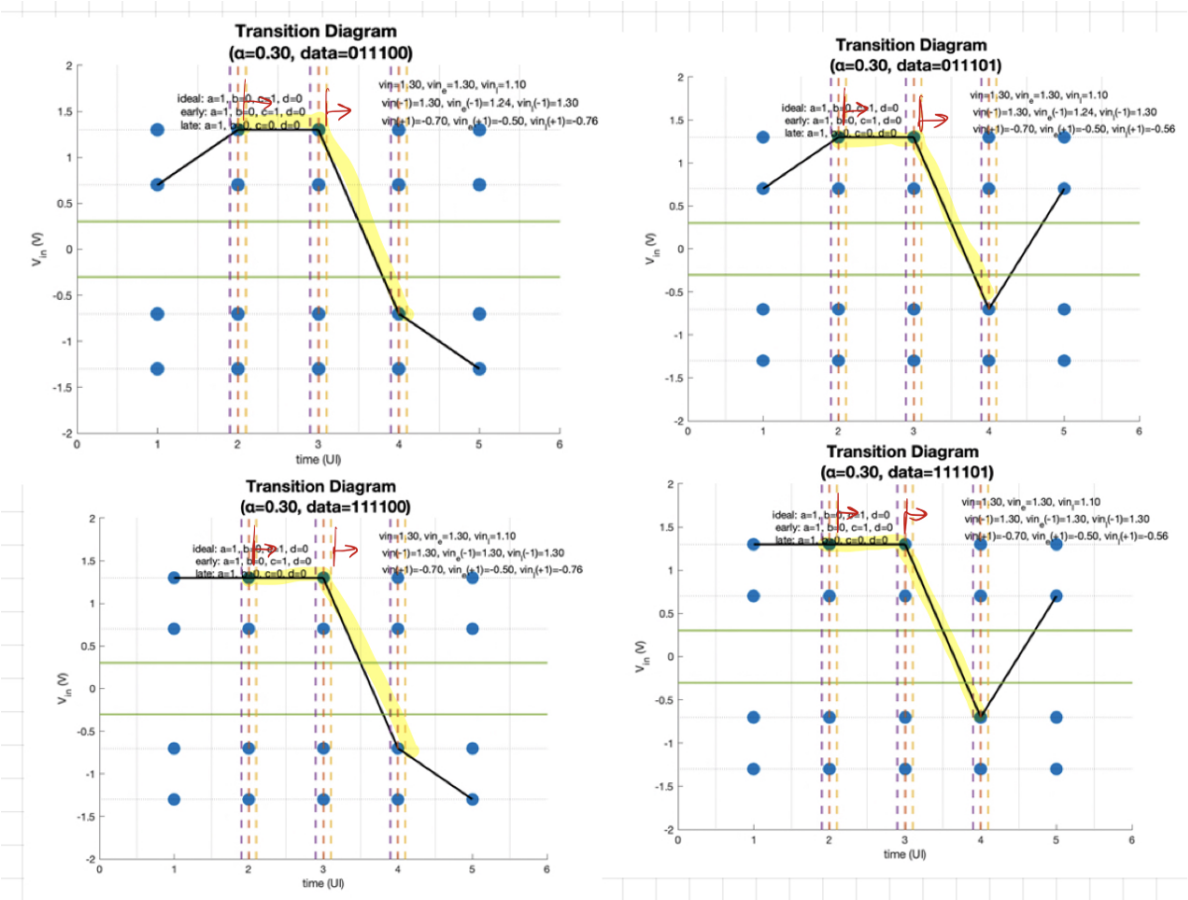


Figure 3.13: Pattern filtering of 1110 for mlse_in-update.

Taking $PD_{out} = mlse_{in}[n]$, we can simplify the locking condition from Equation. 3.12 as the following:

$$h[-1] = \frac{1}{2}(h[2] + h[-2]). \tag{3.13}$$

Note that Equation. 3.13 is the subset of the generalized Mueller-Muller CDR in Equation 3.8 and thus it is a variant of the Mueller-Muller CDR. The advantage of this particular implementation of Mueller-Muller CDR is that it is suitable for the asymmetric pulse response unlike the Type A Mueller-Muller CDR which requires a symmetric pulse response for $h[1] = h[-1]$ to lock to a better phase. In addition, it requires almost no hardware overhead to perform the CDR algorithm, since it uses directly the $mlse_{in}$ signal generated from the MLSE decoder. However, since it still falls into Mueller-Muller CDR and depends heavily on the pulse response coefficient, it could potentially lose lock in some nonideal pulse response.

Combined with dLev maximization

Pattern filtering $mlse_{in}$ from MLSE decoder output can give us the Mueller-Muller style with a better pulse response coefficient relationship than the conventional symmetric locking position. But it could possibly loose locking with the certain pulse response.

To solve this problem, we consider combining the dLev maximization with the $mlse_{in}$ pattern filtering to find the more robust locking point. We choose to maximize the difference between cursor and the first pre-cursor to get a large eye opening. Instead of h_0, dLev is set to adapt to $h_0 - h_{-1}$. Therefore, we need to filter the pattern $D[n : n+1] = 10$ to search the data level of $h_0 - h_{-1}$. Note that $e[n] = sgn(v_{in}[n] - dLev)$.

- dLev Maximization with Dithering ($\delta[n] = \pm\Delta$)

```
if D[n:n+1] == 10:
   PD_out = sgn(δ[n])e[n]
```

Instead of using extra dithering, we reused the previous phase detector output to estimate the slope. The dithering direction would just simply be the previous phase detector output (up or down) for the gradient descent algorithm to work. The relationship of dithering ($\pm\Delta$)and previous phase detector output ($PD_{out,d}$) is illustrated in Figure. 3.14 and Table 3.4. This can be verified by the following equations. Assume $\phi[n]$ is the sampling phase and α is the step size.

$$\phi[n+1] - \phi[n] = \alpha(\phi[n] - \phi[n-1])e[n] \tag{3.14}$$

Since the sampling clock is updated with the phase detector output, we have $\phi[n+1] = PD_{out} + \phi[n]$ and $\phi[n] = PD_{out,d} + \phi[n-1]$. With this information and setting the step size $\alpha = 1$ for simplicity, we can rewrite Equation. 3.14 as

$$PD_{out} = PD_{out,d}e[n]. \tag{3.15}$$

- dLev Maximization with Previous phase detector output ($PD_{out,d}$)

```
if D[n:n+1] == 10:
   PD_out = PD_out,d e[n]
```

We can modify the update equations with two pattern filterings with the following equations. Note that $mlse_{in} = sgn(v_{in}[n] - v_{in}[n-1])$. Compared to [5, 13], the hybrid algorithm does not require an accurate estimate of the pulse response due to the incorporation of the maximization loop.

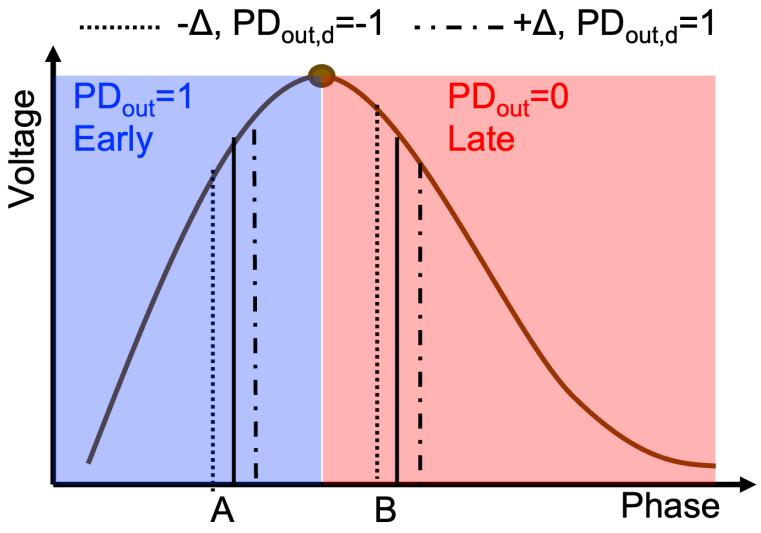


Figure 3.14: dLev maximization illustration (adapted from Figure. 4.5(a) in [26]).

Phase	dithering/ $PD_{out,d}$	$v_{in}[n]$	$dLev$	$e[n]$	PD_{out}
A	$-\Delta$/-1	$V(A-\Delta)$	V(A)	-1	1(Early)
A	$+\Delta$/1	$V(A+\Delta)$	V(A)	1	1(Early)
B	$-\Delta$/-1	$V(B-\Delta)$	V(B)	1	-1(Late)
B	$+\Delta$/1	$V(B+\Delta)$	V(B)	-1	-1(Late)

Table 3.4: Truth table of dLev maximization (adapted from Figure. 4.5(b) in [26].

- Hybrid: $mlse_{in}$ update + dLev maximization with $PD_{out,d}$

```
if D[n:n+1] == 10:
    if D[n-2:n-1]=11:
        PD_out =sgn(mlse_in+PD_out,d e[n])
    else:
        PD_out = PD_out,d e[n].
```

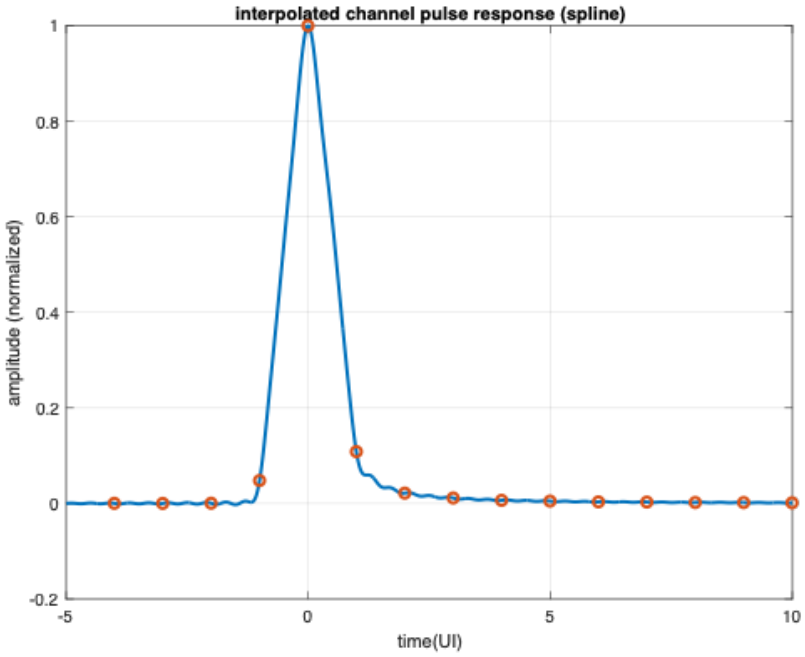


Figure 3.15: Integrated loopback pulse response.

Locking Point

The proposed hybrid CDR algorithm combined Mueller-Muller-based $mlse_{in}$ pattern filter and dLev10 maximization is analyzed with the loopabck channel in 160GHz TRX tapeout with 16nm technology (Sept 2022). The overall integrated pulse response is shown in Figure. 3.15. The red dots represent the samples (channel coefficient h_i) when sampled at the peak of the pulse response.

If we sample the pulse response at different phases over the 1UI window, we can get a different set of channel coefficients. The unique value of the timing function specified for each CDR algorithm is determined by the set of channel coefficients corresponding to a certain sampling phase. Figure. 3.16 shows the timing function curves of different CDR algorithms in the possible sampling phase of 1UI. The locking point can be found on the curves depending on the type of CDR algorithm. For Mueller-Muller CDR, the locking point is the clock phase that forces the timing function to be zero. For dLev maximization, the locking phase corresponds to the maximum value of the timing function. The CDR type, timing function, and corresponding locking point are summarized in Table. 3.5. The locking point in the table refers to the peak in the pulse response. The locking point for $f(\tau) = h_0$ should be 0 UI if there is no resolution issue.

This method can find the locking point when the timing function and the locking condition of the timing function are straightforward. For the hybrid algorithm that involved

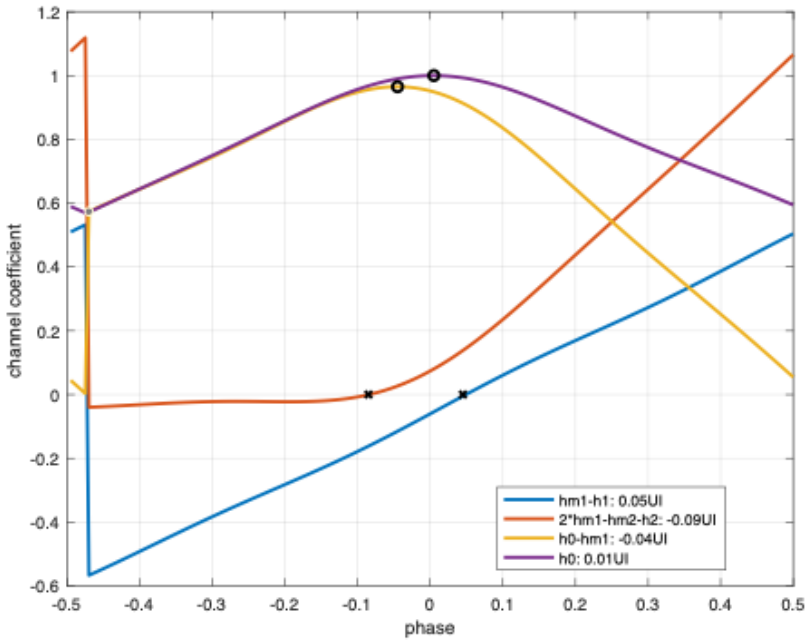


Figure 3.16: Sweeping of timing function.

Color	CDR type	Timing function	Locking point
blue	Mueller-Muller	$h_{-1} - h_1$	0.05UI
red	Mueller-Muller	$2h_{-1} - h_{-2} - h_2$	-0.09UI
yellow	dLev Maximization	$h_0 - h_{-1}$	-0.04UI
purple	dLev Maximization	h_0	0.01UI

Table 3.5: Table of sweeping of timing function.

both Mueller-Muller based CDR and dLev Maximization CDR, the locking point cannot be analyzed directly with this method. Moreover, it can only predict the averaged steady-state locking point but not the jitter performance when reaching the steady state.

Chapter 4

Statistical Analysis of the CDR Algorithm

4.1 Overview

There are two separate methodologies for characterizing the CDR systems found in the literature [42]. In the circuits community, IC designers tend to linearize the phase detector and treat the whole loop as a linear control system. Markov chain analysis is used intensively in communication systems. The pseudo-linear analysis [37, 14, 28] treats the bang-bang phase detector as a 1-bit quantizer and approximates it as gain elements for random and deterministic input and an additive Gaussian noise source. Linearization enables frequency domain analysis and takes loop delay into account. It is useful to predict jitter generation, JTRAN, and JTOL. However, since gain and noise are modeled with variance of Gaussian distribution, the analysis could be inaccurate with an unbounded jitter distribution [43]. Although the transition density is considered in the linearization, it cannot capture the full information of pattern filtering, especially when there are conditional terms such as the hybrid algorithm in Session 3.3 where the order of the bit stream matters. Markov chain analysis describes every possible phase as one state with the transition probability determined by noise and the residual ISI distribution. It is intuitive to apply filtering to the transition probability for pattern filtering. The steady-state probability indicates the locking point and the distribution of the jitter generated by CDR itself. It is useful to compare the locking point and jitter generation of CDR algorithms over different channels. Although the CDR loop latency is not modeled in Markov chain analysis, the jitter caused by the loop latency can be ignored when the phase step is small enough so that input jitter is dominated. In this work, we analyze the CDR with Markov chain, given that the state-based framework is suitable for pattern filtering and multiple conditions in hybrid algorithm.

One drawback of regular Markov chain analysis is that it only depends on the current phase position to predict the next state, while in our hybrid algorithm, the output of the previous phase detector ($PD_{out,d}$) and the bit stream ($D[n-2:n+1]$) are used to decide

the next state. To model hybrid CDR more accurately, we propose using the multiple-variable Markov chain analysis, which is inspired by known statistical analysis [21, 49], as an extension to the regular Markov chain analysis [27, 1] used in the literature. In our analysis for the hybrid algorithm, we need to include the factors that correlate with each other in the transition matrices. By exploring relationships among variables (e.g., $PD_{out,d}$, data sequences), we can develop better models for the output phase sequences and hence better prediction.

In this chapter, we apply Markov chain analysis to Mueller-Muller based CDR implemented by pattern filtering with MLSE decoder input, the data level maximization CDR, and hybrid CDR proposed in Session 3.3.The extension of statistical eye analysis is proposed to determine the optimal locking point window for a certain pulse response.

Both regular Markov chain and the multiple-variable Markov chain analysis applied to three CDR algorithms were verified by comparing with the steady-state histogram from the ground truth time-domain simulation. The proposed hybrid CDR algorithm is shown to be more robust over different channels than Mueller-Muller CDR and dLev maximization CDR in the end of the chapter.

Markov Chain Analysis

Markov chain analysis is widely used to analyze the CDR system in the communication community [42]. Each possible sampling phase is modeled as a state in a Markov chain shown in Figure. 4.1. With a phase detector output value listed in Table 4.1, the transition of state ϕ_i can go either **up** with probability $P_{i,i+1}$, **dn** with probability $P_{i,i-1}$, and **hold** with probability $P_{i,i}$. To derive the transition probability of the Markov chain in each state, we need the information of statistics of the input data and ISI along with the pattern filtering condition on the update equations.

PD_{out}	sampling time	direction	pattern filtering
0	-	hold	False
1	Early	up	True
-1	Late	dn	True

Table 4.1: Sample space of PD_{out}.

ISI Distribution

To understand the probability that the phase detector goes up or down, we need to first obtain the ISI distribution of the pulse response. Assuming that the transmitted data are uncoded, random, and independent, the received signal is a sum of transmitted data (random

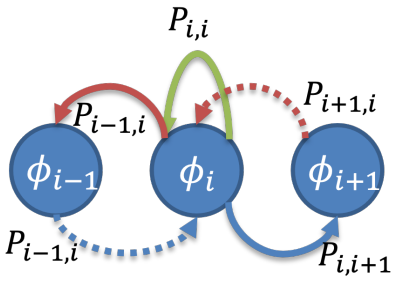


Figure 4.1: Single-variable Markov chain.

variables) weighted by the ISI tap of the pulse response as expressed in Equation. 3.1. The ISI distribution, which is the probability distribution of the resulting received signal, can be calculated by convolving each tap [42]. Assuming the data sequence is random and independent of each other, each data has probability of 0.5 to be either 0 or 1. if we convolute 2 ISI taps h_1 and h_{-1}, we will have four possible values for four combinations of D [n+1] and Dn [n-1]. Each of the combination will have an even probability, i.e. 0.25 in the y-axis. The operation with a pulse response of 200Gbps PAM4 TX in [47] is shown in Figure. 4.2.

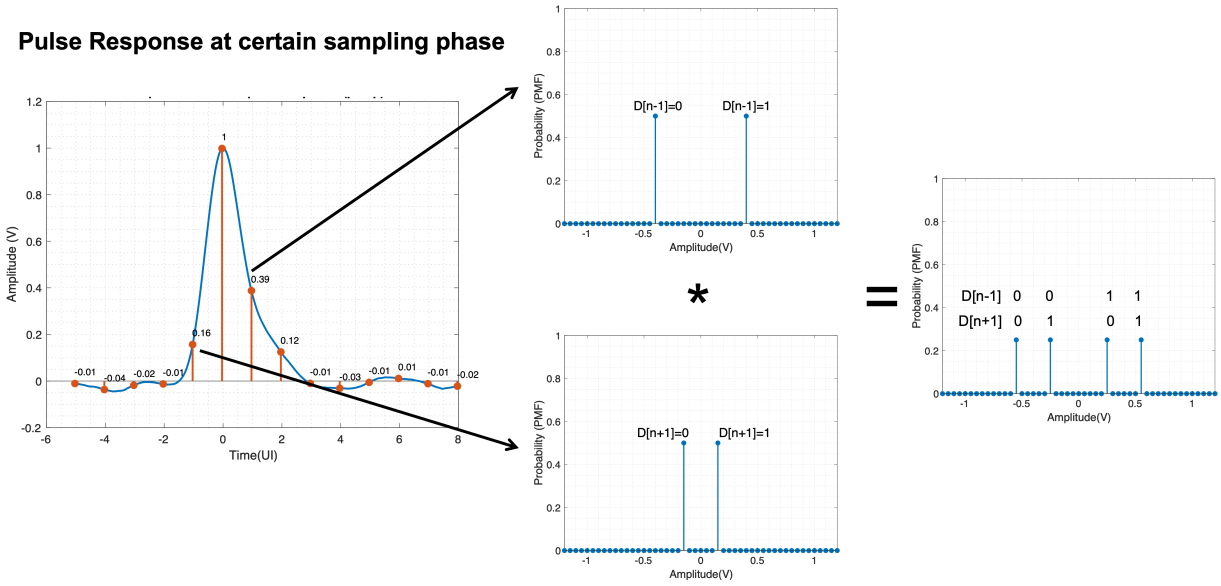


Figure 4.2: ISI distribution.

When calculating the ISI distribution of $V_{in}[n-k]$, we convolve all the taps other than $h[k]$. For example, for the ISI distribution of $V_{in}[n]$, all taps except $h[0]$ are convolved. An example of the ISI distribution of $V_{in}[n]$ (Equation. 4.1) and $V_{in}[n-1]$ (Equation. 4.2) is shown in Figure. 4.3. Pattern filtering can be done by fixing the ISI tap to the chosen

pattern, which will cause the offset shift. Note that the center of Figure. 4.3 is not 0 but offset to 1.5 due to pattern filtering $D[n-2:n+1] = 1110$.

$$\begin{aligned} V_{in}[n] &= \sum_{k=-\infty}^{\infty} D[n-k]h[k] \\ &= \sum_{k\neq -1,0,1,2}^{\infty} D[n-k]h[k] + \sum_{k=-1}^{2} D[n-k]h[k] \\ &= \sum_{k\neq -1,0,1,2}^{\infty} D[n-k]h[k] + \textbf{pattern offset}. \end{aligned} \tag{4.1}$$

$$\begin{aligned} V_{in}[n-1] &= \sum_{k=-\infty}^{\infty} D[n-k]h[k-1] \\ &= \sum_{k\neq -1,0,1,2}^{\infty} D[n-k]h[k-1] + \Sigma_{k=-1}^{2} D[n-k]h[k-1] \\ &= \sum_{k\neq -1,0,1,2}^{\infty} D[n-k]h[k-1] + \textbf{pattern offset}. \end{aligned} \tag{4.2}$$

We can get the probability of $ISI > 0$ and $ISI < 0$ by partitioning the histogram into two parts with **ISI amplitude = 0** and sum the probability in the left part and in the right part, respectively. If we apply a similar operation to the ISI distribution based on the update equation $PD_{out} = sgn(V_{in}[n] - V_{in}[n-1])$, we can get the probability of going up and down by integrating the sum of the histogram to the right and left of the **amplitude = 0**. The example of getting ISI distribution of $MLSE_{in}$ is shown in Figure. 4.4.

4.2 Mueller-Muller $mlse_{in}$ update Analysis

For the Mueller-Muller algorithm, we force $f(\tau) = 0$ for the locking condition, which leads to the update equation.

$$mlse_{in} = sgn(v_{in}[n] - v_{in}[n-1]). \tag{4.3}$$

Calculate the transition probability shown in the Figure. 4.1 , we find the ISI distribution of both $V_{in}[n]$ and $V_{in}[n-1]$ and take the difference of both to get $x = v_{in}[n] - v_{in}[n-1]$ shown in Figure 4.4a. In ISI distribution x, the summation of the PMF below 0 is the probability that the phase detector will go down (P_{dn}) when the summation of the PMF above 0 is the probability that the phase detector will go up (P_{up}). For example, Figure 4.4a is the ISI distribution when the clock phase is at a phase count of 150. The probability of going up and down can be annotated as two dots at $x = 150$ in Figure. 4.4b. The phase resolution is 0.002 UI with 500 phase counts in total. At the locking point phase = ϕ_{lock}, we have $P_{up} = P_{dn}$, which is the crossing point in Figure. 4.4b.

The process of obtaining the transition probability from the ISI distribution is simplified in 4.5 with the pulse response extracted from the loop-back channel used in the 160Gbps

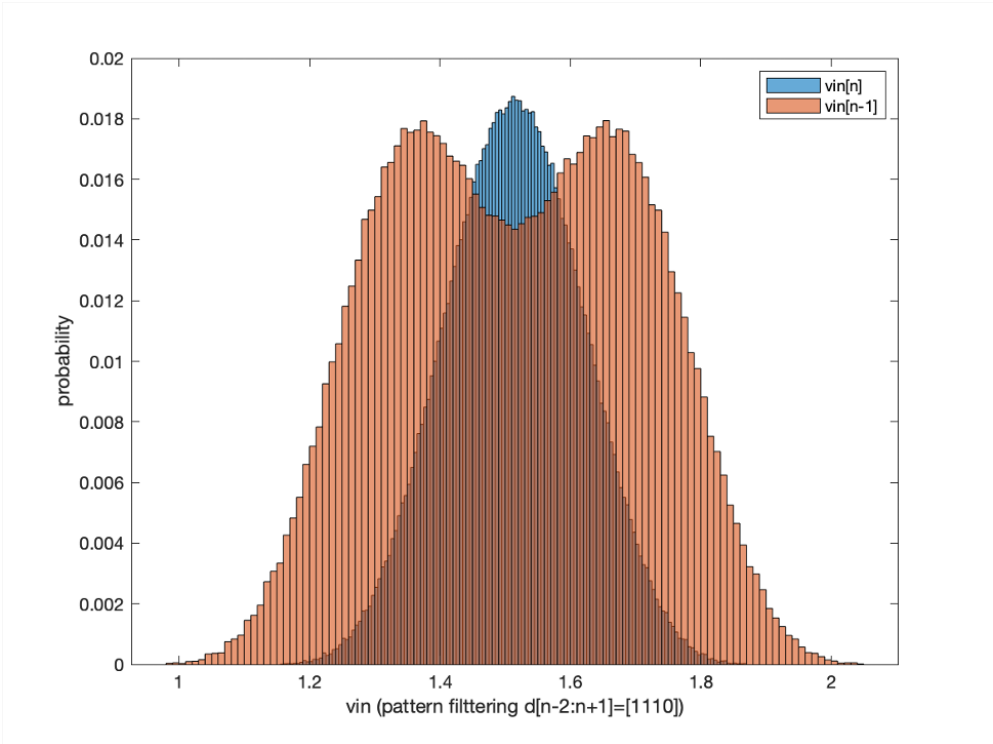
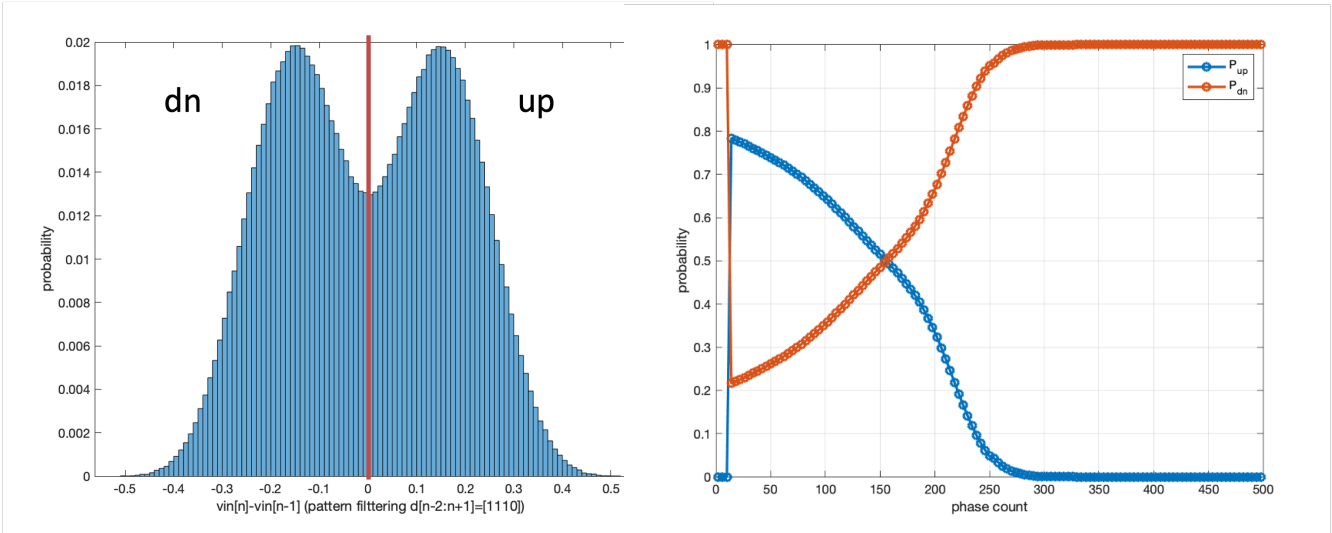


Figure 4.3: ISI distribution of $v_{in}[n]$ and $v_{in}[n-1]$.



(a) ISI distribution of at phase count of 150.

(b) Probability of up and down over 1 UI.

Figure 4.4: MLSE ISI probability ($v_{in}[n] - v_{in}[n-1]$).

TRX tapeout[31]. If there is only an up-and-down decision in the phase detector, then the steady-state probability of the CDR output phase can be calculated directly by finding the eigen vector of the transition probability matrix from the transition probability calculated directly from the ISI distribution. However, there is another **hold** state which occurs when the data pattern $D[n-2:n+1] \neq [1110]$. With pattern filtering, the hold state is considered and the new transition probability is calculated from the up-and-down transition probability with the data pattern condition. The steady-state probability will be the eigen vector of the new transition probability matrix. The locking point is the phase count that gives the maximum point of the steady-state probability and should be the same as the point where $P_{up} = P_{dn}$ in the transition probability plot. The process of the above operation is summarized in Figure. 4.6.

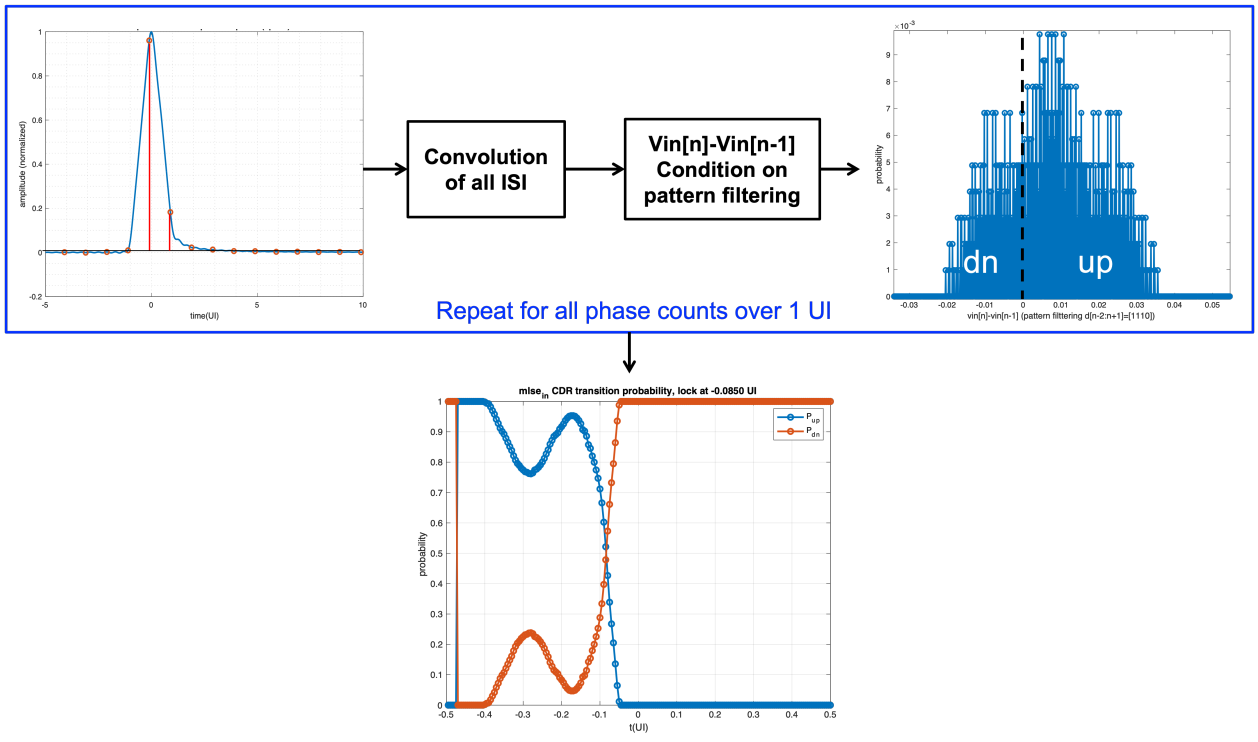


Figure 4.5: Transition probability of $mlse_{in}$ with loop-back channel [31].

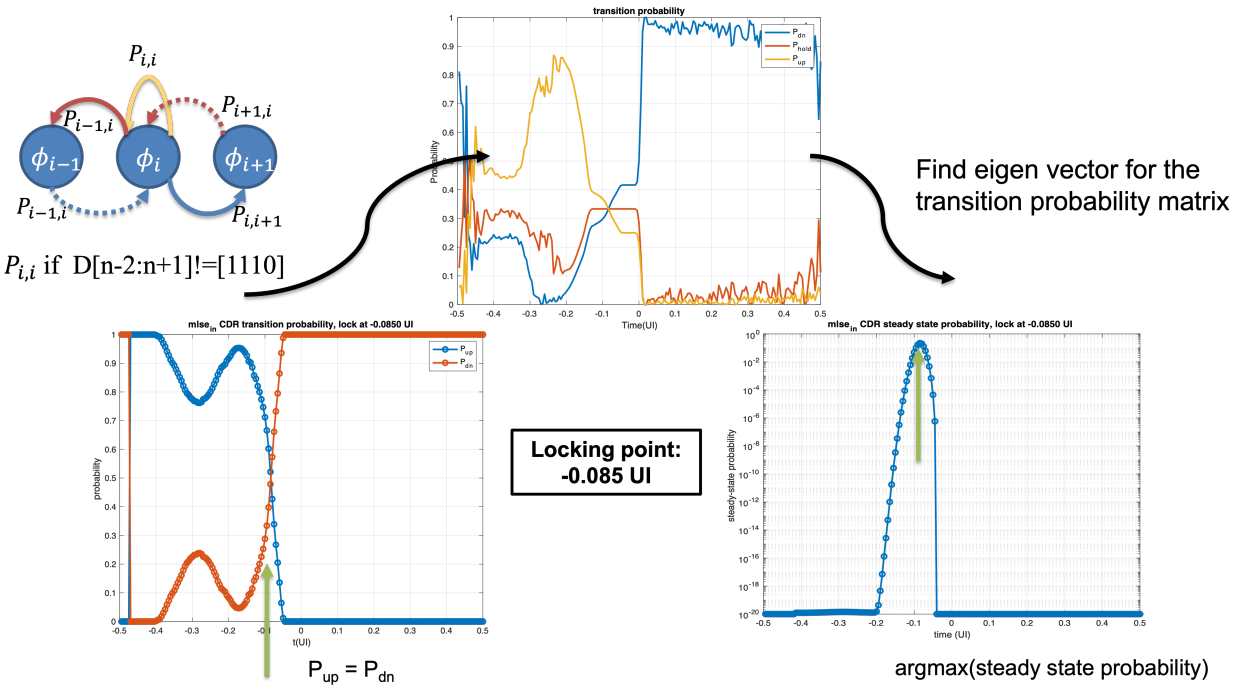


Figure 4.6: Steady-state probability with pattern filtering.

4.3 Data level (dLev) Maximization Analysis

The data level (dLev) is specified as the average received analog signal with certain data pattern filtering and is usually used for the adaptation of equalization. Usually in the link with the equalizers working properly, the dLev would converge to the chosen combination of the main cursor and ISI, for example h_0 depending on the application. This can be achieved by pattern filtering the incoming signal and adapting to the reference level. For example, only adapt with the error signal when $D[n] = 1$ to obtain the level h_0. In some channel, we would like to maximize the main cursor h_0 to get a low BER. In this case, we can maximize the data level with pattern filtering $D[n] = 1$ instead.

Since the main cursor and ISI vary with the sampling phase, $dLev(\phi)$ represents the converged dLev when the clock is in phase ϕ. In the dLev maximization described in the previous chapter, additional dithering ($dith$) is inserted to determine whether the clock phase is early or late compared to the maximum point of h_0. Define

$$\delta dLev(\phi) = dLev(\phi + dith) - dLev(\phi). \tag{4.4}$$

We are then trying to describe the dLev maximization phase detector operation with the equations below, which help us to develop the ISI distribution in order to obtain the transition probability of the Markov chain.

If $dith = \Delta$, then if $\delta dLev(\phi) > 0$, the clock is early and the output of the phase detector should go up. On the other hand, if $\Delta dLev(\phi) < 0$, the clock is late and the phase detector output should go down.

If $dith = -\Delta$, then the results reverse in the above condition: if $\delta dLev(\phi) > 0$, the clock is early and the output of the phase detector should go down. On the other hand, if $\delta dLev(\phi) < 0$, the clock is late and the phase detector should go up.

Based on the observation, we can derive the probability that the phase detector going up at phase ϕ as follows.

$$\begin{aligned} &P_{up}(\phi) \\ &= P(dith = \Delta)\delta dLev|_{dith=\Delta} - P(dith = -\Delta)\delta dLev|_{dith=-\Delta} \\ &= P(dith = \Delta)((dLev(\phi + \Delta) - dLev(\phi))) + P(dith = -\Delta)(dLev(\phi - \Delta) - dLev(\phi)). \end{aligned} \tag{4.5}$$

If we define

$$\begin{aligned} dLev_{slope}(\phi) &= \delta dLev|_{dith=\Delta} \\ &= dLev(\phi + \Delta) - dLev(\phi) \end{aligned} \tag{4.6}$$

then

$$\begin{aligned} \delta dLev|_{dith=-\Delta} &= dLev(\phi - \Delta) - dLev(\phi) \\ &= -(dLev((\phi - \Delta) + \Delta) - dLev(\phi - \Delta)) \\ &= -dLev_{slope}(\phi - \Delta). \end{aligned} \tag{4.7}$$

$$
\begin{aligned}
&P_{up}(\phi) \\
&= P(dith = \Delta)((dLev(\phi + \Delta) - dLev(\phi))) - P(dith = -\Delta)(dLev(\phi - \Delta) - dLev(\phi)) \\
&= P(dith = \Delta)dLev_{slope}(\phi) + P(dith = -\Delta)dLev_{slope}(\phi - \Delta) \\
&= 0.5(dLev_{slope}(\phi) + dLev_{slope}(\phi - \Delta)).
\end{aligned} \tag{4.8}
$$

From the above derivation, we can calculate $P_{up}(\phi)$ by obtaining the ISI distribution of $dLev_{slope}(\phi)$ and average it with the shifted version of itself.

In the receiver with one-tap MLSE, two $dLev$ values are used for the adaptation in the datapath, which are $dLev_{01}$ and $dLev_{11}$. $dLev_{xy}$ stands for the averaged sample at $d[n-1:n] = xy$. Ideally, $dLev_{xy}$ will converge to $yh[0] + xh[1]$ where $h[0]$ is the main cursor and $h[1]$ is the first post-cursor. $dLev_{01}$ and $dLev_{11}$ are obtained with pattern filtering of $d[n-1:n] = 01$ and $d[n-1:n] = 11$, respectively. The results are shown in Figure 4.7(channel of 200Gbps PAM4 TX).

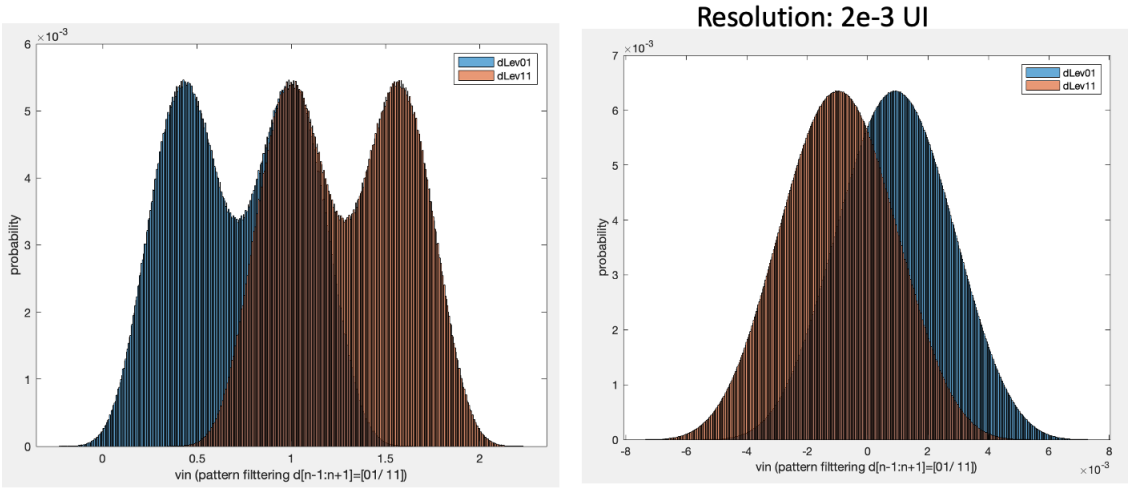


Figure 4.7: ISI distribution of dLev01/ dLev11 at different sample phase count.

4.4 Hybrid Algorithm Analysis

The proposed CDR algorithm takes advantage of both the Mueller-Muller algorithm and dLev maximization to provide the well-defined and optimal locking position across a group of realistic pulse waveforms without adding additional dithering. One challenge that needs to be addressed is the exact locking position and the jitter distribution of this hybrid method.

Since the Mueller-Muller $mlse\_in$-update is pattern filtering at $D[n-2:n+1]1110$ and the 1-tap MLSE is designed to for the first post-cursor, the remaining first pre-cursor could be used to maximize the margin. Therefore, $dLev_{10}$ is chosen to maximize for the largest eye opening.

To solve this problem, we need to obtain the transition probability of both Mueller-Muller in session 4.2 and dLev maximization of $dLev_{10}$ in session 4.3 individually. This information is the basis for the analysis of the hybrid algorithm. With the information of individual Mueller-Muller $mlse\_in$-update and the dLev maximization, we now proceed to combine both with the Markov chain analysis.

Single-Variable Markov chain

We first try to tackle this problem with the single-variable Markov chain described in session 4.1. The update equations for the hybrid algorithm that was derived in session 3.3 are rewritten below. $e[n]$ is the output of the error slicer that compares the sampled value $v_{in}[n]$ with $dLev_{10}$.

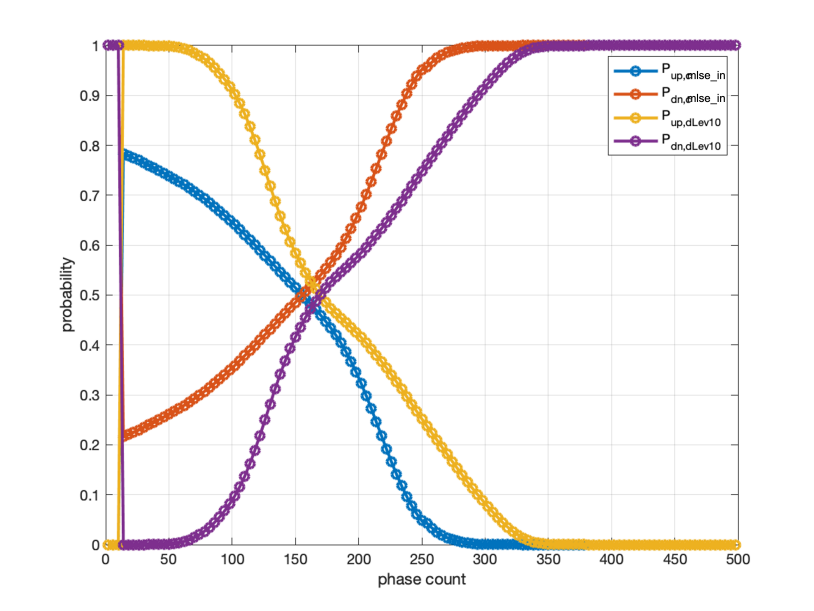


Figure 4.8: Individual $mlse_{in}$-update and dLev10 transition probability.

```
if D[n:n+1] == 10:
   if D[n-2:n-1] == 11:
      PD_out = sgn(sgn(Vin[n] - Vin[n-1]) + PD_out,d e[n])
   else:
      PD_out = PD_out,d e[n]
```

Based on the value of $PD_{out,d}$, we can tabulate the other components as shown below.

$PD_{out,d}$	$P(PD_{out,d})$	$e[n]$	$sgn(PD_{out,d}e[n])$
0	$P_{i,i}$	0	0
1	$\frac{\pi_{i-1}P_{i-1,i}}{\pi_i}$	$dLev(\phi_i) - dLev(\phi_{i-1}) = \Delta dLev(\phi_i)$	$sgn(\Delta dLev(\phi_i))$
-1	$\frac{\pi_{i+1}P_{i+1,i}}{\pi_i}$	$dLev(\phi_i) - dLev(\phi_{i+1}) = -\Delta dLev(\phi_{i+1})$	$sgn(\Delta dLev(\phi_{i+1}))$

Table 4.2: Sample space of PD_{out}.

Here, we define

$$\Delta dLev(\phi_i) = dLev(\phi_i) - dLev(\phi_{i-1}). \tag{4.9}$$

Assume $c = mlse\_in = v_{in}[n] - v_{in}[n-1]$ and $e[n]$ are the output of the comparator, which will only be +1 or -1.

$$\begin{aligned} P_{up,i} &= P_{i,i+1} \\ &= P(PD_{out} > 0) \\ &= P(c > 0, PD_{out,d}e[n] \geq 0) \\ &= P(c > 0)P(PD_{out,d}e[n] \geq 0) \\ &= P(c > 0)(P(PD_{out,d}e[n] = 0) + P(PD_{out,d}e[n] > 0)) \\ &= P(c > 0)(P(PD_{out,d} = 0) + (\sum_{x=\pm 1} P(PD_{out,d} = x))P(xe[n] > 0)) \end{aligned} \tag{4.10}$$

From the equations derived in A.1, we can rewrite the results into a set of equations that describe the transition probability in the state i of the Markov chain. Here, $\vec{\pi}$ is the steady-state distribution of the Markov chain.

$$\pi_i = P(\phi_n = i) \tag{4.11}$$

$$\begin{aligned} P_{i,i+1} &= P(PD_{out} > 0) \\ &= \frac{P_{c,up}(\phi_i)}{\pi_i}[\pi_i P_{i,i} + \pi_{i-1}P_{i-1,i}P_{dLev,up}(\phi_i) + \pi_{i+1}P_{i+1,i}P_{dLev,up}(\phi_{i+1})] \end{aligned} \tag{4.12}$$

$$\begin{aligned} P_{i,i-1} &= P(PD_{out} < 0) \\ &= \frac{P_{c,dn}(\phi_i)}{\pi_i}[\pi_i P_{i,i} + \pi_{i-1}P_{i-1,i}P_{dLev,dn}(\phi_i) + \pi_{i+1}P_{i+1,i}P_{dLev,dn}(\phi_{i+1})] \end{aligned} \tag{4.13}$$

$$\begin{aligned} P_{i,i} &= P(PD_{out} = 0) \\ &= \frac{P_{c,up}(\phi_i)}{\pi_i}[\pi_i P_{i,i} + \pi_{i-1} P_{i-1,i} P_{dLev,dn}(\phi_i) + \pi_{i+1} P_{i+1,i} P_{dLev,dn}(\phi_{i+1})] \\ &+ \frac{P_{c,dn}(\phi_i)}{\pi_i}[\pi_i P_{i,i} + \pi_{i-1} P_{i-1,i} P_{dLev,up}(\phi_i) + \pi_{i+1} P_{i+1,i} P_{dLev,up}(\phi_{i+1})] \end{aligned} \tag{4.14}$$

Assuming that we obtain the information about the Mueller-Muller update and the dLev update, the unknown variables are π_{i-1}, π_i, π_{i+1}, $P_{i,i-1}$, $P_{i,i}$, $P_{i,i+1}$, $P_{i-1,i}$, $P_{i+1,i}$, which are more than the equations we have in the set.

If we multiply both sides of the equations with π, we can now treat $\pi_i P_{i,i}$, $\pi_i P_{i,i+1}$, $\pi_i P_{i,i-1}$, $\pi_{i-1} P_{i-1,i}$, $\pi_{i+1} P_{i+1,i}$ as unknown variables in the set of equations.

$$\pi_i P_{i,i+1} = P_{c,up}(\phi_i)[\pi_i P_{i,i} + \pi_{i-1} P_{i-1,i} P_{dLev,up}(\phi_i) + \pi_{i+1} P_{i+1,i} P_{dLev,up}(\phi_{i+1})] \tag{4.15}$$

$$\pi_i P_{i,i-1} = P_{c,dn}(\phi_i)[\pi_i P_{i,i} + \pi_{i-1} P_{i-1,i} P_{dLev,dn}(\phi_i) + \pi_{i+1} P_{i+1,i} P_{dLev,dn}(\phi_{i+1})] \tag{4.16}$$

$$\begin{aligned} \pi_i P_{i,i} &= P_{c,up}(\phi_i)[\pi_i P_{i,i} + \pi_{i-1} P_{i-1,i} P_{dLev,dn}(\phi_i) + \pi_{i+1} P_{i+1,i} P_{dLev,dn}(\phi_{i+1})] \\ &+ P_{c,dn}(\phi_i)[\pi_i P_{i,i} + \pi_{i-1} P_{i-1,i} P_{dLev,up}(\phi_i) + \pi_{i+1} P_{i+1,i} P_{dLev,up}(\phi_{i+1})] \end{aligned} \tag{4.17}$$

For an K-state Markov chain, we have $\pi_1, \pi_2, \ldots, \pi_K$, and $P_{1,K}, P_{1,1}, P_{1,2}, P_{2,1}, P_{2,2}, P_{2,3}, \ldots, P_{K,K-1}, P_{K,K}, P_{K,1}$. That implies that we will have $4K$ unknown variables in total. However, if we slightly convert the equations, we will have $\pi_1 P_{1,K}, \pi_1 P_{1,1}, \pi_1 P_{1,2}, \pi_2 P_{2,1}, \pi_2 P_{2,2}, \pi_2 P_{2,3}, \ldots, \pi_K P_{K,K-1}, \pi_K P_{K,K}, \pi_K P_{K,1}$, where we only have $3K$ unknown variables to solve in $3K$ equations.

The transition probability can be solved by rearranging in the form of matrix P and using the steady-state Markov chain property $\pi P = \pi$ which is discussed in the next section.

Transition Matrix

To solve the linear equations of $\pi_x P_{x,y}$ above, we can arrange the equations in homogeneous form and then translate them into matrix form. We can solve the unknown variables $\pi_x P_{x,y}$ by solving the null space of the matrix. Then use the characteristic of the transition probability $\sum_{j=1,\ldots,K} \pi_i P_{i,j} = \pi_i$ to get π_i. With π_i being solved, $P_{i,j}$ can be simply solved by $\frac{\pi_i P_{i,j}}{P_{i,j}}$.

Rearrange Equation 4.15, Equation 4.16, and equation 4.17 in homogeneous form and then we can rewrite the equations in matrix form as follows.

$$A\vec{x} = \vec{0} \tag{4.18}$$

.

We know that $\vec{x}$ is in the null space and all elements in $\vec{x}$ have to be non-negative since π_x and $P_{x,y}$ are probability and thus non-negative. $\vec{x}$ would be the null space of A but only the one with positive sign.

After $\vec{x}$ is solved, we can calculate the steady-state probability at state i by summing all $\pi_i P_{i,j}$. Since $\sum_j P_{i,j} = 1$, we have

$$\sum_j \pi_i P_{i,j} = \pi_i \sum_j P_{i,j} = \pi_i \tag{4.19}$$

With $\pi_i P_{i,j}$ and π_i solved in the previous steps, we can derive the transition probability $P_{i,j}$ simply by

$$\frac{\pi_i P_{i,j}}{\pi_i} = P_{i,j} \tag{4.20}$$

An example of a 3-state Markov chain is used to illustrate the steps in A.2. The steps to solve the steady-state probability and the transition state probability are summarized in Figure 4.9.

To evaluate the accuracy of the derived using a single-variable Markov chain, we compare the steady-state probability with the histogram obtained from the time-domain simulation steady state. Two pulse responses are used for the evaluation: the ideal linearized pulse response (Figure. 4.10a) and real-channel pulse response (Figure. 4.11a). The histogram represents the results from time-domain simulation while the piecewise linear curve represents the statistical derivation. Three CDR algorithms are compared for the locking position and jitter distribution: **c** for Mueller-Muller based **mlse_in** algorithm, **dLev** for dLev maximization algorithm, and **hybrid** for hybrid algorithm. In the simple ideal channel, the time domain matches the statistical analysis in all algorithms (Figure. 4.10b). However, in the real channel, the result in the time domain of the hybrid algorithm has a different locking position and jitter distribution than in the statistical analysis (Figure. 4.11b).

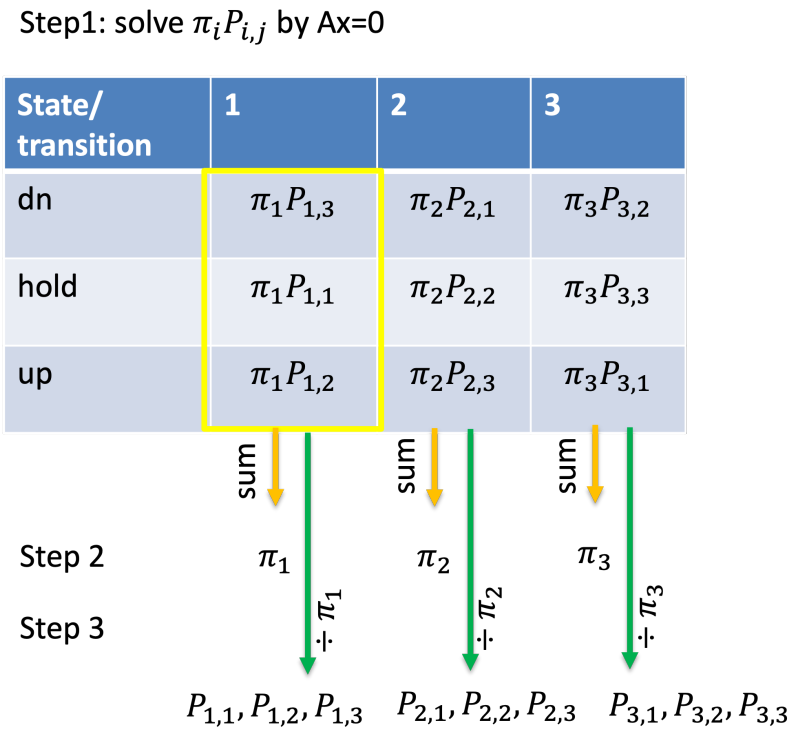
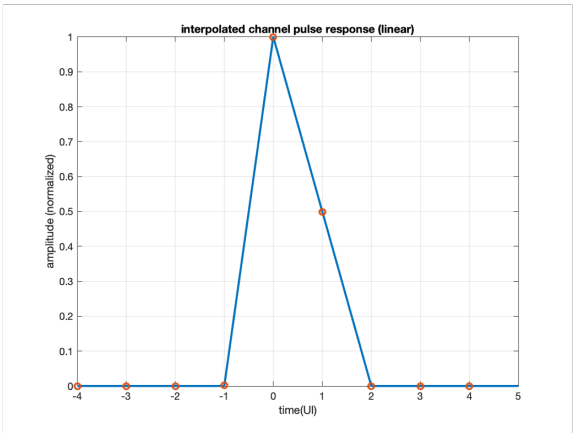
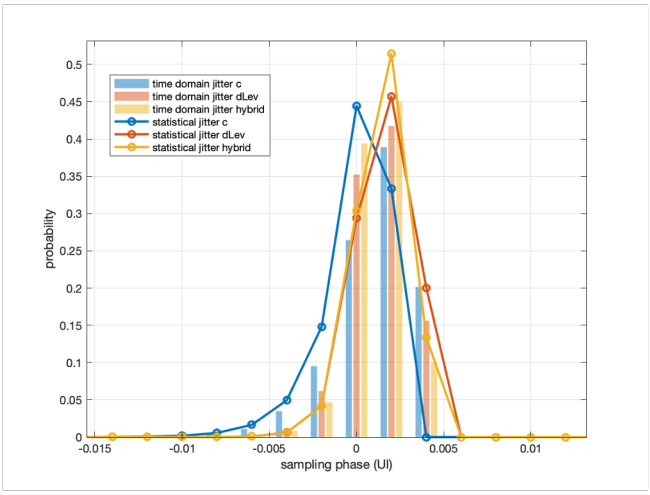


Figure 4.9: Steps of solving steady state and transition probability.

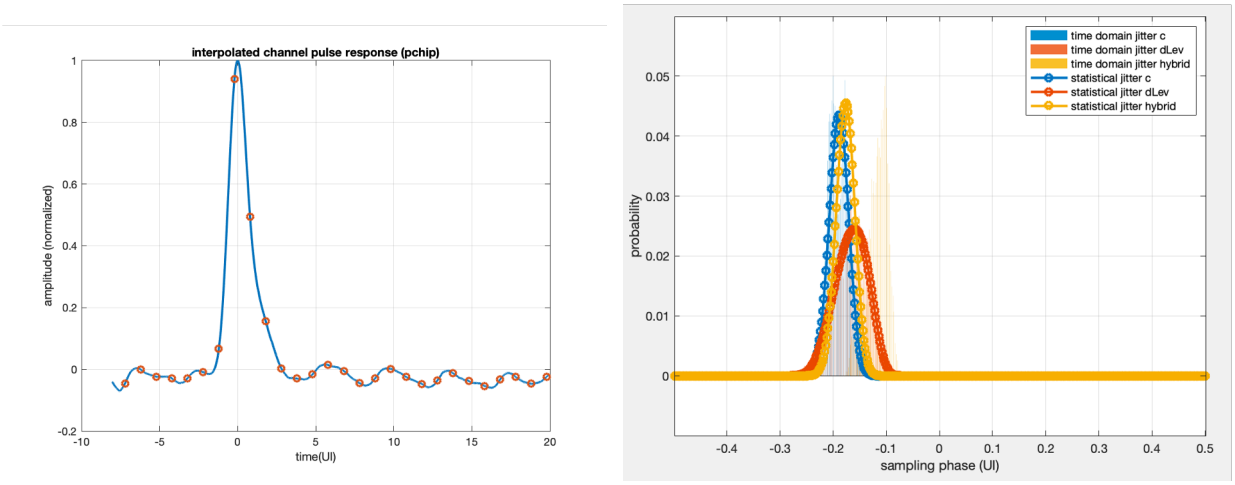


(a) Pulse response.



(b) Steady-state probability.

Figure 4.10: Time-domain simulation vs. statistical analysis with ideal channel.



(a) Pulse response

(b) Steady-state probability

Figure 4.11: Time-domain simulation vs. statistical analysis with real channel.

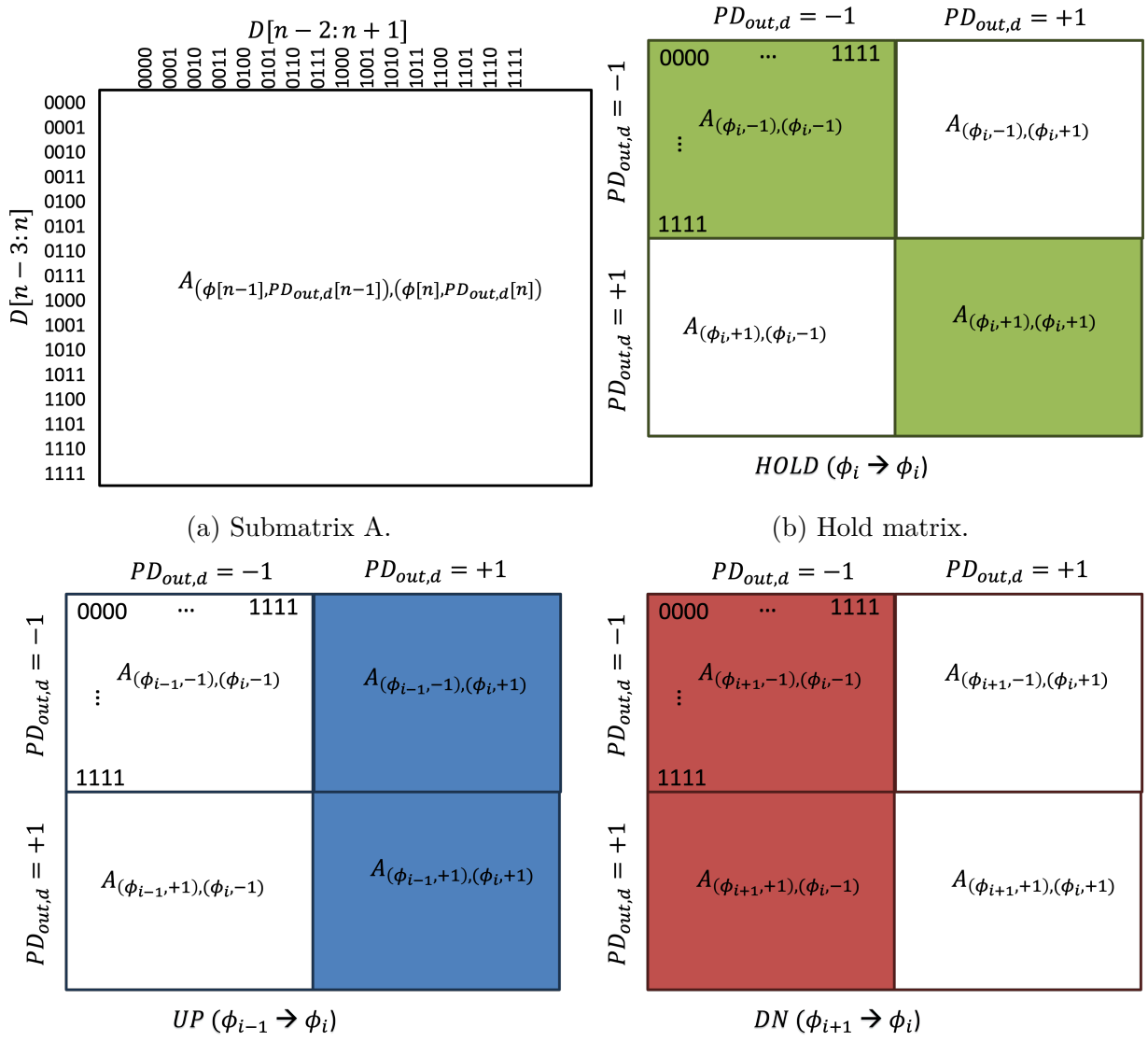
Multiple-variable Markov chain

To resolve the discrepancy of the time-domain simulation and the single-variable Markov chain analysis, the multiple variable Markov chain is being proposed to include more information in each state. We consider three variables in the Markov chain: phase (ϕ_i), previous phase detector output ($PD_{out,d}$), and pattern filtering (sequence of D). The matrix of multiple variables of the Markov chain is illustrated in Figure 4.12, where the submatrix A includes all the possible transitions between the filtered data sequence.

Assume that there are m possible phase detector outputs, k phase count, and the length of the filtered data sequence is p. The size of the submatrix A will be $2^p \times 2^p$. We can integrate submatrix A to include the phase detector output, the resulting matrix size at each phase will be $k^2 \times 2^p \times 2^p$. Finally, with the phase counts considered, the size of the flattened transition matrix would be $m^2 \times k^2 \times 2^p \times 2^p$ in total.

If we filter $D[n-3:n]$, where n is the current time step, then the submatrix A is a 16 by 16 matrix. In our algorithm, $PD_{out,d}$ records the last meaningful output of the phase detector, which means the last time we update with the filtered pattern. Therefore, if the pattern is not found, then both phase and $PD_{out,d}$ will remain the previous recorded output, which should be either **up(+1)** or **dn(-1)**. For each phase (ϕ_i), the previous phase detector output ($PD_{out,d}$) can be $+1$ or -1. In this case, we have $4k^2$ submatrices. The size of the matrix will be $4k^2 \times 16^2 = 1024k^2$.

Figure. 4.12 shows the construction of the matrix. $A_{(\phi[n-1],PD_{out,d}[n-1]),(\phi[n],PD_{out,d}[n])}$ in Figure. 4.12a is the submatrix that denotes the transition that occurs at time step n. The green submatrices ($A_{(\phi_j,x),(\phi_j,x)}$) that lie in the diagonal positions of the integrated matrix (Figure. 4.13) are the transition matrix for the *hold* decision (Figure. 4.12b). The blue submatrices $A_{(\phi_{i-1},+1),(\phi_i,-1)}$ and the red submatrices $A_{(\phi_{i+1},x),(\phi_i,x)}$ are the transition matrices for the decision *up* and *dn*, respectively (Figure. 4.12c and Figure. 4.12d. The other submatrices in the integrated matrices are impossible transitions, and thus the entry will be zero. Although the size of the matrix seems large, it is sparse and thus realistic to be solved.



(a) Submatrix A.

(b) Hold matrix.

(c) Up matrix.

(d) Dn matrix.

Figure 4.12: Construction elements of the multiple variable Markov chain transition matrix.

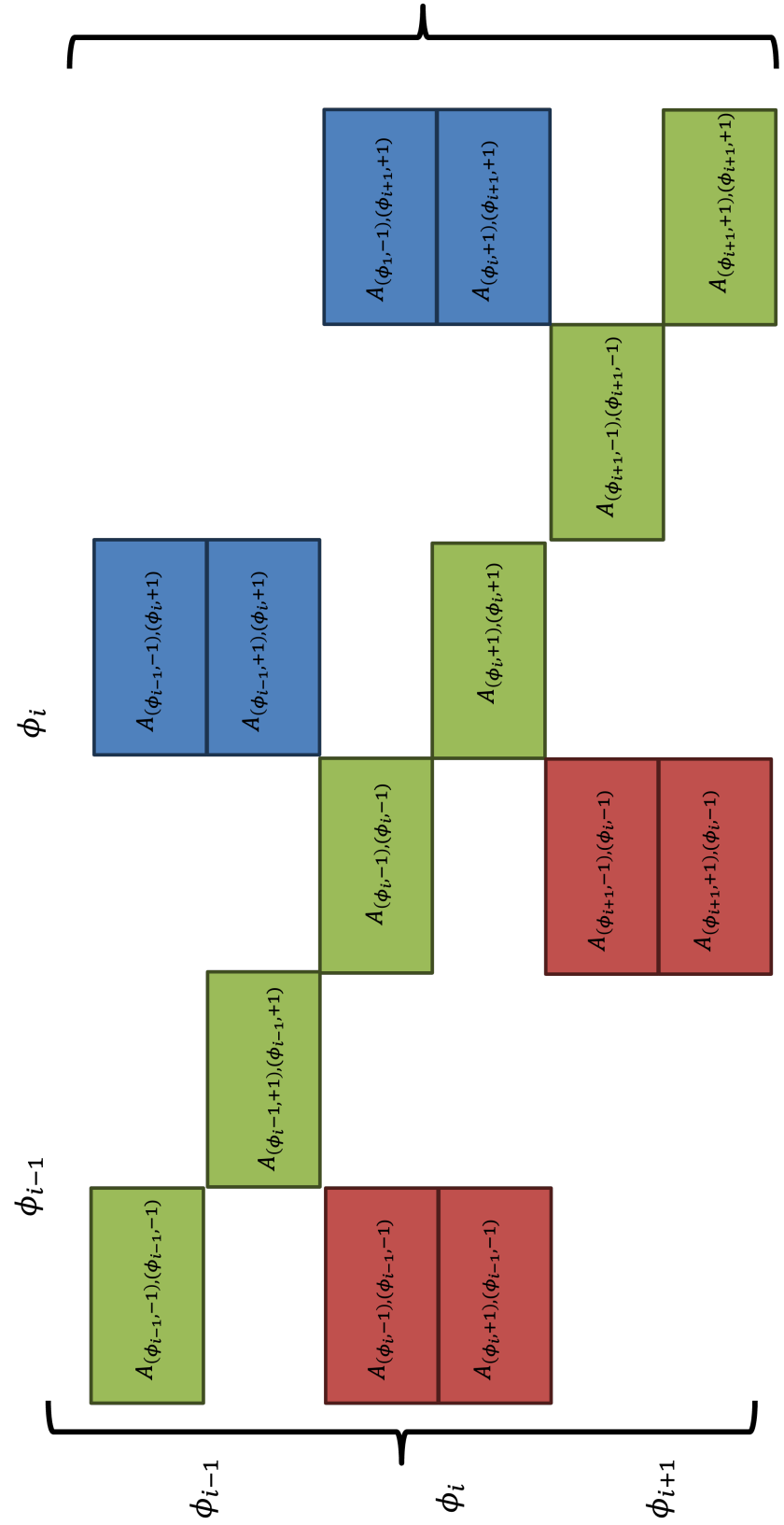


Figure 4.13: Multiple Variable Markov chain transition matrix.

Transition Matrix

The transition matrix in Figure. 4.13 can be rewritten as Equation. 4.21. Since this matrix itself includes hold state and pattern filtering, we can easily find steady-state probability by finding the eigen vector for the transition probability matrix.

$$P = \begin{bmatrix} A_{(\phi_0,-1),(\phi_0,-1)} & A_{(\phi_0,-1),(\phi_0,+1)} & \cdots\cdots & A_{(\phi_0,-1),(\phi_n,-1)} & A_{(\phi_0,-1),(\phi_n,+1)} \\ A_{(\phi_0,+1),(\phi_0,-1)} & A_{(\phi_0,+1),(\phi_0,+1)} & \cdots\cdots & A_{(\phi_0,+1),(\phi_n,-1)} & A_{(\phi_0,+1),(\phi_n,+1)} \\ \vdots & \vdots & & & \vdots \\ \cdots & \cdots & A_{(\phi_a,x),(\phi_b,y)} & \cdots & \cdots \\ \vdots & \vdots & & \vdots & \vdots \\ A_{(\phi_n,+1),(\phi_0,-1)} & A_{(\phi_n,+1),(\phi_0,+1)} & \cdots\cdots & A_{(\phi_n,+1),(\phi_n,-1)} & A_{(\phi_n,+1),(\phi_n,+1)} \end{bmatrix} \tag{4.21}$$

Hold Submatrices When $D[n:n+1] \neq 10$, the phase detector $PD_{out} = 0$ and the update logic will not be triggered. This means that both $PD_{out,d}$ and the phase count $\phi_n = \phi_{n+1}$ will not change. Since we assume that the data sequence is $i.i.d$, we know that $D_{in}[n+1]$ can be either 0 or 1 with a probability of 0.5.

If state n ($S_n =< (i_0, i_1, i_2, i_3), \phi_i, PD_d >$, then state $n+1 S_{n+1}$ the probability mass function (PMF) can be derived as

$$P_{S_{n+1}}(s) = \begin{cases} 0.5, & s =< (i_1, i_2, i_3, 0), \phi_i, PD_d > \\ 0.5, & s =< (i_1, i_2, i_3, 1), \phi_i, PD_d > \\ 0, & otherwise \end{cases}$$

The submatrix of the hold decision is shown in Equation. 4.22. Note that it is not a complete transition matrix, thus the summation of each row is not always one. In fact, all

entries in the row corresponding to the data pattern $D[n:n+1]=10$ are zero.

$$A_{(\phi_i,x),(\phi_i,x)}=\frac{1}{2}\begin{bmatrix}
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1
\end{bmatrix} \tag{4.22}$$

Up and Dn Submatrices Based on the update equation, we only update the phase detector output when $D[n:n+1]=10$. Mueller-Muller-based $MLSE_{in}/c$ update only happens at $D[n-2:n+1]=1110$ while dLev update happens at $D[n-2:n+1]=0010/0110/1010$.

To find the transition probability, we consider two cases where $PD_{out,d}[n]=+1/-1$. The value of $PD_{out,d}[n]$ will affect the dLev maximization part but not the Mueller-Muller-based $MLSE_{in}/c$ update.

For $D[n-2:n+1]=0010/0110/1010$, we only consider dLev maximization. Assuming $PD_{out,d}[n]=PD_d\in[+1,-1]$, i.e. $S_n=<(i_0i_1i_2i_3),\phi_i,PD_d>$, then

$$P_{S_{n+1}}(s)=\begin{cases}
0.5P(\Delta dLev(i_0i_1i_2i_3 0),\phi(PD_d)<0), & s=<(i_1i_2i_3 0),\phi_{i-1},-1>\\
0.5P(\Delta dLev(i_0i_1i_2i_3 1),\phi(PD_d)<0), & s=<(i_1i_2i_3 1),\phi_{i-1},-1>\\
0.5P(\Delta dLev(i_0i_1i_2i_3 0),\phi(PD_d)>0), & s=<(i_1i_2i_3 0),\phi_{i+1},+1>\\
0.5P(\Delta dLev(i_0i_1i_2i_3 1),\phi(PD_d)>0), & s=<(i_1i_2i_3 1),\phi_{i+1},+1>\\
0, & otherwise
\end{cases} \tag{4.23}$$

From Table. 4.2, we know that

$$\phi(PD_d)=\begin{cases}\phi_{i+1}, & \text{for } PD_d=-1\\ \phi_i, & \text{for } PD_d=+1\end{cases} \tag{4.24}$$

For $D[n-2:n+1]=1110$, we need to take into account the Mueller-Muller-based

$MLSE_{in}/c$ update. Assuming $S_n =< (1110), \phi_i, PD_d >$, we have the following.

$$P_{S_{n+1}}(s) = \begin{cases} 0.5P(sgn(f_{hybrid}(11100, \phi_i, PD_d)) < 0), & s =< (1100), \phi_{i-1}, -1 > \\ 0.5P(sgn(f_{hybrid}(11101, \phi_i, PD_d)) < 0), & s =< (1101), \phi_{i-1}, -1 > \\ 0.5P(sgn(f_{hybrid}(11100, \phi_i, PD_d)) > 0), & s =< (1100), \phi_{i+1}, +1 > \\ 0.5P(sgn(f_{hybrid}(11101, \phi_i, PD_d)) > 0), & s =< (1101), \phi_{i+1}, +1 > \\ 0, & otherwise \end{cases} \tag{4.25}$$

where

$$f_{hybrid}(Data, \phi, PD_{out,d}) = sgn(c(Data, \phi) + sgn\Delta dLev(Data), \phi(PD_{out,d})) \tag{4.26}$$

Up Submatrices The up submatrices consist of the transition probability where the phase detector output is $+1$, which means the next delayed phase detector output $PD_{out,d} = +1$. The blue and red entries in Equation. 4.27 correspond to the probability of the first two states in Equation. 4.23 and Equation. 4.25 respectively.

$$A_{(\phi_{i-1},x),(\phi_i,1)} = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P_{up,0} & P_{up,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & P_{up,2} & P_{up,3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P_{up,4} & P_{up,5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & P_{up,6} & P_{up,7} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.27}$$

where

$$\begin{aligned}
P_{up,0} &= P(PD_{x,00100}(\phi_{i-1}) > 0) \\
P_{up,1} &= P(PD_{x,00101}(\phi_{i-1}) > 0) \\
P_{up,2} &= P(PD_{x,01100}(\phi_{i-1}) > 0) \\
P_{up,3} &= P(PD_{x,01101}(\phi_{i-1}) > 0) \\
P_{up,4} &= P(PD_{x,10100}(\phi_{i-1}) > 0) \\
P_{up,5} &= P(PD_{x,10101}(\phi_{i-1}) > 0) \\
P_{up,6} &= P(PD_{x,11100}(\phi_{i-1}) > 0) \\
P_{up,7} &= P(PD_{x,11101}(\phi_{i-1}) > 0)
\end{aligned} \tag{4.28}$$

Dn Submatrices The up submatrices consist of the transition probability where the phase detector output is -1, which means the next delayed phase detector output $PD_{out,d} = -1$. The blue and red entries in Equation. 4.27 correspond to the probability of the third and fourth states in Equation. 4.23 and Equation. 4.25 respectively.

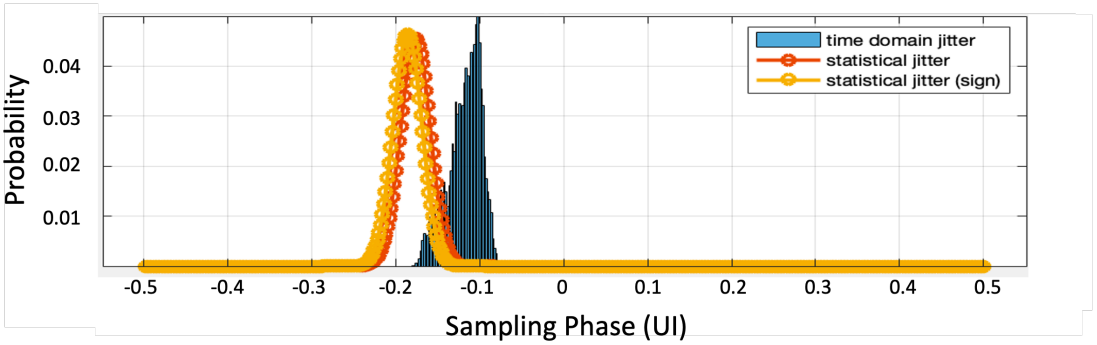
$$A_{(\phi_{i-1},x),(\phi_i,1)} = \frac{1}{2}\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & P_{dn,0} & P_{dn,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & P_{dn,2} & P_{dn,3} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & P_{dn,4} & P_{dn,5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & P_{dn,6} & P_{dn,7} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{4.29}$$

where

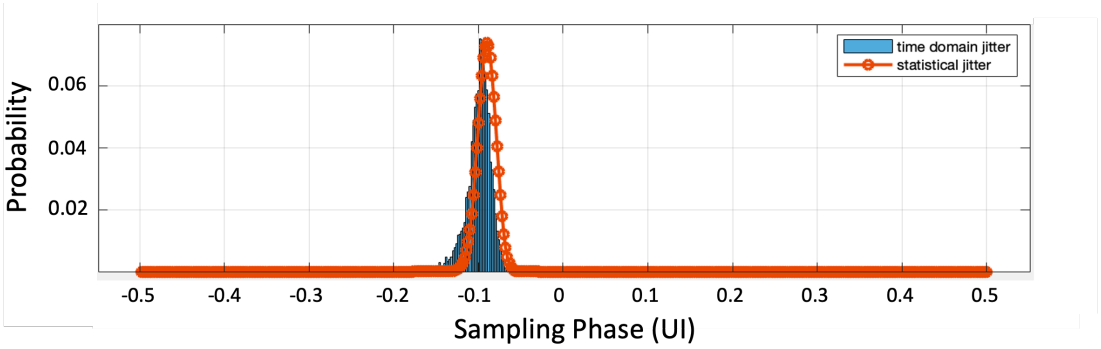
$$\begin{aligned}
P_{dn,0} &= P(PD_{x,00100}(\phi_{i+1}) > 0) \\
P_{dn,1} &= P(PD_{x,00101}(\phi_{i+1}) > 0) \\
P_{dn,2} &= P(PD_{x,01100}(\phi_{i+1}) > 0) \\
P_{dn,3} &= P(PD_{x,01101}(\phi_{i+1}) > 0) \\
P_{dn,4} &= P(PD_{x,10100}(\phi_{i+1}) > 0) \\
P_{dn,5} &= P(PD_{x,10101}(\phi_{i+1}) > 0) \\
P_{dn,6} &= P(PD_{x,11100}(\phi_{i+1}) > 0) \\
P_{dn,7} &= P(PD_{x,11101}(\phi_{i+1}) > 0)
\end{aligned} \tag{4.30}$$

Compare to Single-Variable Markov Chain

The results of statistical analysis are verified through the comparison with histogram from ground truth time-domain simulation. Shown in Figure. 4.14b, the results of the multiple-variable Markov chain match much better than the single-variable counterpart for the hybrid algorithm in the figure. 4.14a. The dotted statistical analysis (dotted curve) agrees well with the time-domain simulation (blue histogram) both in position (locking point) and shape (steady-state jitter distribution) in Figure. 4.14b, which indicates that the accuracy of the statistical results of the multiple-variable Markov chain is enhanced.



(a) Single-variable Markov chain.



(b) Multiple-variable Markov chain.

Figure 4.14: Time-domain simulation vs. statistical analysis with real channel.

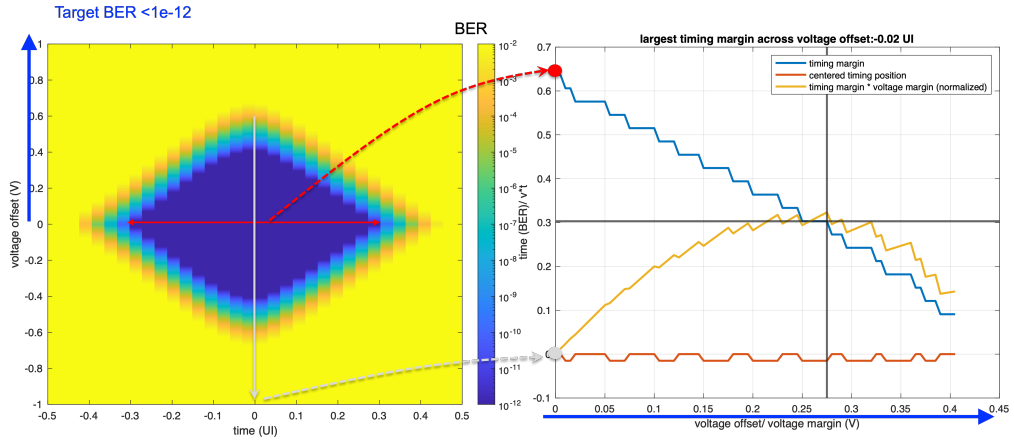
4.5 Performance Matrices

Statistical Eye Analysis

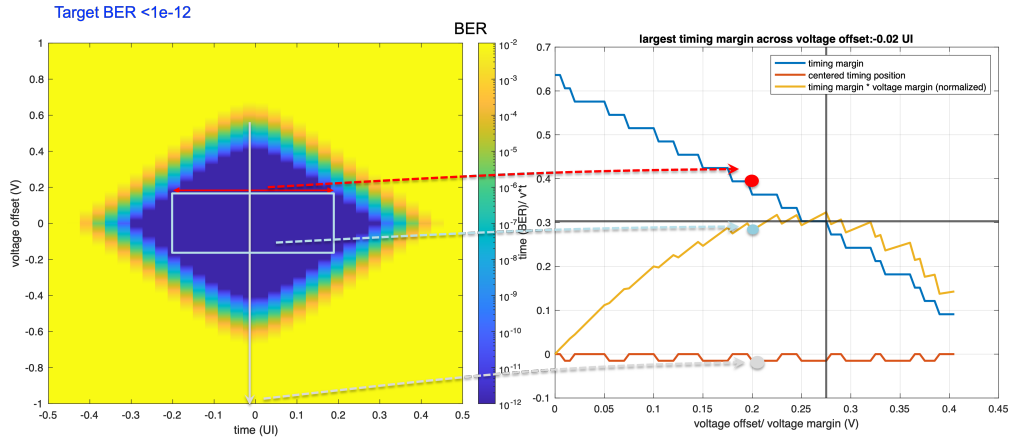
To evaluate the performance of CDR algorithm, we need to first define the performance matrices as a baseline. The locking position and steady-state jitter are two critical considerations of the recovered clock.

We would like to have the steady-state jitter as small as possible. However, it takes more considerations to decide the best locking position of the recovered clock over a known pulse response. The best locking position should give the best performance for a link. Statistical eye analysis [42] is a way to evaluate performance at different sampling phases by understanding the behavior of the signal received throughout the symbol time.

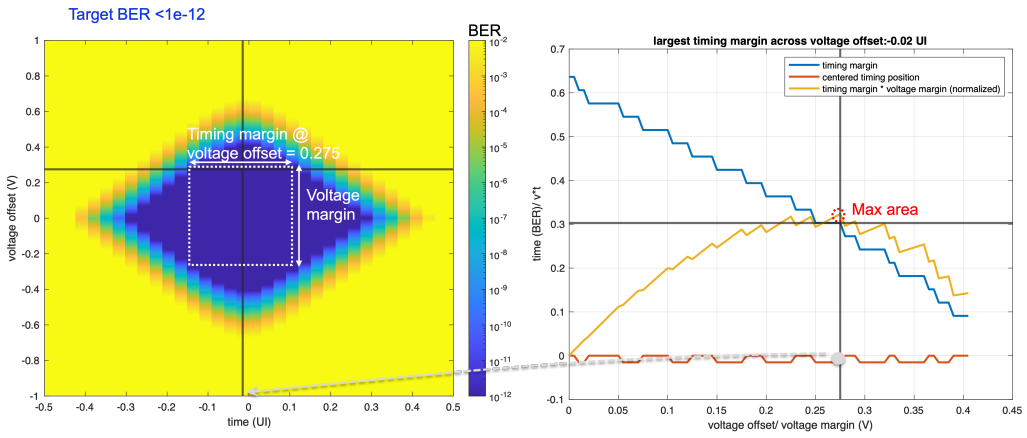
The statistical eye is a set of probability contours. The horizontal axis represents the time domain, and the vertical axis represents the scaling signal amplitude. The colored lines represent the different BER simulations. For the pulse response extracted from the loop-back channel used in the 160Gbps TRX tapeout[31] in Figure. 4.5, we can obtain the statistical eye in Figure. 4.15 and analyze for the optimal locking position. Figure. 4.15a show the timing and voltage margin extracted from the statistical eye. Since the statistical eye is symmetric with respect to the x-axis, we can only consider the timing margin and the voltage margin when *voltage offset* > 0 (blue line). If we look at the voltage offset of 0 and set the target BER to 10^{-12}, we can get the timing margin by the length of the red line. The center of the red line is defined as the center timing position that gives the largest timing margin at 0 voltage offset. As we sweep the voltage offset, we can get the timing margin and optimal sampling time across the voltage offset. For example, if we set the voltage offset to 0.2 UI (Figure. 4.15b, the effective voltage margin is 0.4 UI, which is twice the voltage offset. Here we can find the corresponding timing margin and centered timing position. Furthermore, since the voltage margin is non-zero now, we can find the eye opening area. Both the timing margin and the voltage margin are important performance matrices. Here we define the area as *timing margin* $\times$ *voltage margin*. We weight the timing margin and the voltage margin the same by normalizing them to their own maximum value, respectively. In order to maximize both the voltage margin and the timing margin, we try to find the sampling point that corresponds to the maximal square area. We can also observe that the centered timing position varies a little across the voltage offset. In Figure. 4.15c, the maximum area occurs when the center timing position is -0.02 UI.



(a) Mapping of statistical eye to timing and voltage margin.



(b) voltage offset = 0.2UI.



(c) Maximum area of the opening eye.

Figure 4.15: Statistical eye analysis.

To compare Mueller-Muller $MLSE\_in$, dLev maximization and hybrid algorithm, we find the steady-state phase probability of all algorithms in both time-domain simulation and statistical analysis in Figure. 4.16. The bold dotted line shows the optimal sampling position across different voltage offsets from -0.02UI to 0 UI. We can find that it is on the right side of both MM-base and dLev maximization. And the hybrid algorithm actually fits better in the optimal sampling region. We can estimate the steady-state jitter from the spread of the probability distribution. The wider distribution means the larger steady-state jitter. The hybrid algorithm has the best locking position and moderate jitter distribution by combining Mueller-Muller and dLev maximization, making it a better and more robust candidate for the CDR algorithm.

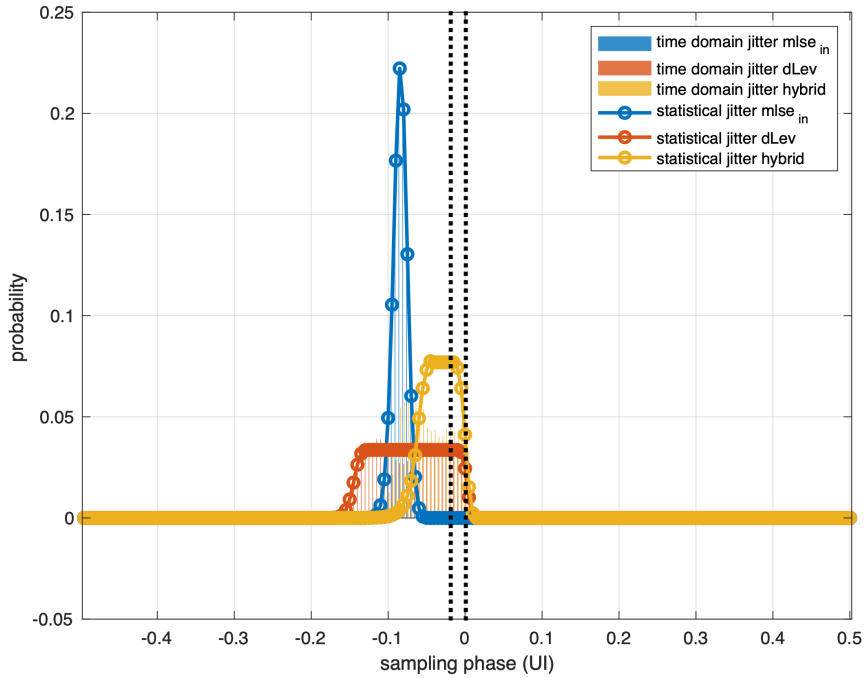


Figure 4.16: Time-domain simulation vs. steady-state probability with 160Gbps loop-back channel. (dotted lines: optimal locking window from statistical eye analysis)

Jitter Tolerance (JTOL)

The jitter tolerance (JTOL) defines the maximum tolerable input jitter that can achieve a given bit error rate (BER).

$$BER = \frac{\text{Number of bit errors}}{\text{Number of transmitted bits}} \tag{4.31}$$

The way to test the JTOL is to add the sinusoidal jitter in the data input of the CDR and gradually increase the amplitude of the jitter until the BER exceeds the given BER at a certain frequency. If we sweep the frequency and repeat the same procedure to find the JTOL at each frequency, we can compare the JTOL curve of the CDR with the JTOL mask specified for a specific application and data rate. If the JTOL curve lies above the JTOL mask, the JTOL is satisfied for the specs. Since we are not linearizing the phase detector as [37, 14, 28] as stated in Session 4.1, JTOL cannot be derived directly from the Markov chain statistical method that we describe in this chapter. We directly apply a time-domain simulation to obtain the JTOL of each CDR algorithm.The JTOLs of different CDR algorithms are shown in Figure. 4.17. All algorithms meet the JTOL specs with the CEI-112G-XSR JTOL mask. The hybrid algorithm has a similar JTOL curve as the dLev maximization algorithm, which is superior to the Mueller-Muller $MLSE\_in$ algorithm.

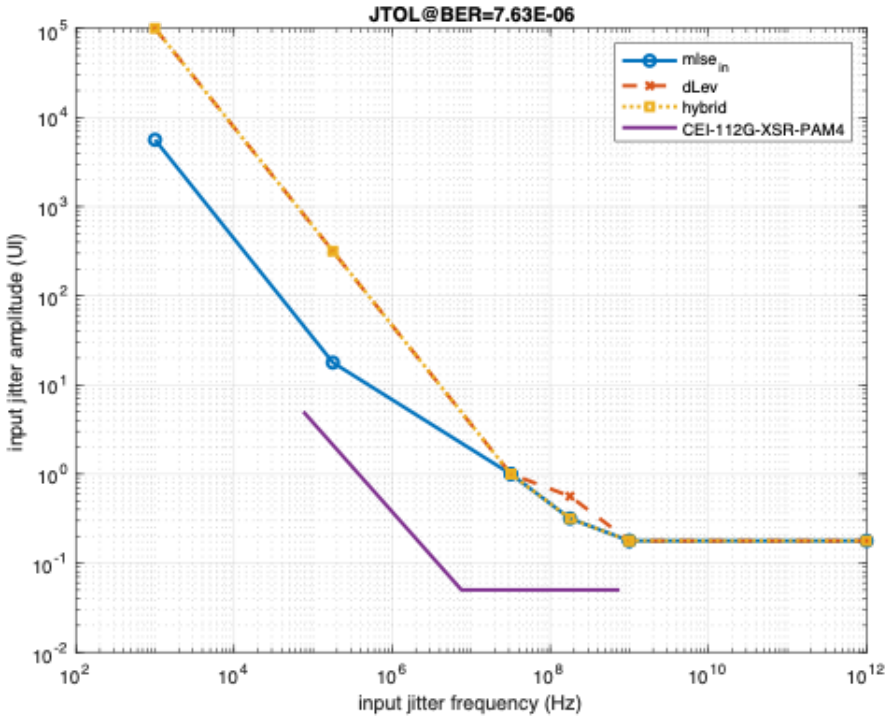


Figure 4.17: JTOL with 160Gbps loopback channel.

4.6 Summary

In this chapter, we apply Markov-chain-based statistical analysis to predict the locking point and steady-state jitter of the CDR algorithm. The multiple-variable Markov chain analysis is proposed to better analyze the CDR algorithm with complicated update equations that include different variables and depend on the data pattern. The statistical results are verified through the ground-truth time-domain simulation. We show that the single-variable Markov chain with filtering is good enough to predict Mueller-Muller CDR and dLev maximization CDR in Session 4.2 and Session 4.3 for two different channels in Figure. 4.10 and Figure. 4.11. We derive and demonstrate that a multiple-variable Markov chain can predict the locking point and the jitter distribution more accurately in Figure. 4.14b. These results conclude the versatile usage of Markov-chain-based statistical analysis for different baud-rate CDR algorithms and the accuracy to predict the locking point and steady-state jitter distribution. Furthermore, we demonstrate that the locking point and steady-state jitter distribution of the hybrid algorithm can be accurately predicted by multiple Markov chain statistical analysis. Although the locking point and steady-state jitter distribution can be predicted accurately, the limitation of the Markov-chain-based analysis lacks the direct derivation for jitter tolerance (JTOL) which we will need to verify through time-domain simulation (Figure. 4.17).

Since JTOL cannot be derived directly through Markov-chain-based statistical analysis, we propose to use the optimal window for the locking point from statistical eye analysis to evaluate the CDR performance in terms of the locking point and the steady-state jitter distribution. Shown in Figure. 4.16, the dotted line shows the optimal locking window and suggests that the proposed hybrid algorithm has the best performance, since the area under the steady-state probability distribution of the hybrid algorithm has greater overlap with the locking window.

Chapter 5

Design of 100Gb/s 1-Tap MLSE Receiver with Baud-rate CDR

5.1 Overview

A 100 Gbps NRZ 1-tap MLSE receiver was designed in a 16 nm FinFET process to implement the hybrid CDR algorithm described and analyzed in the previous chapters.

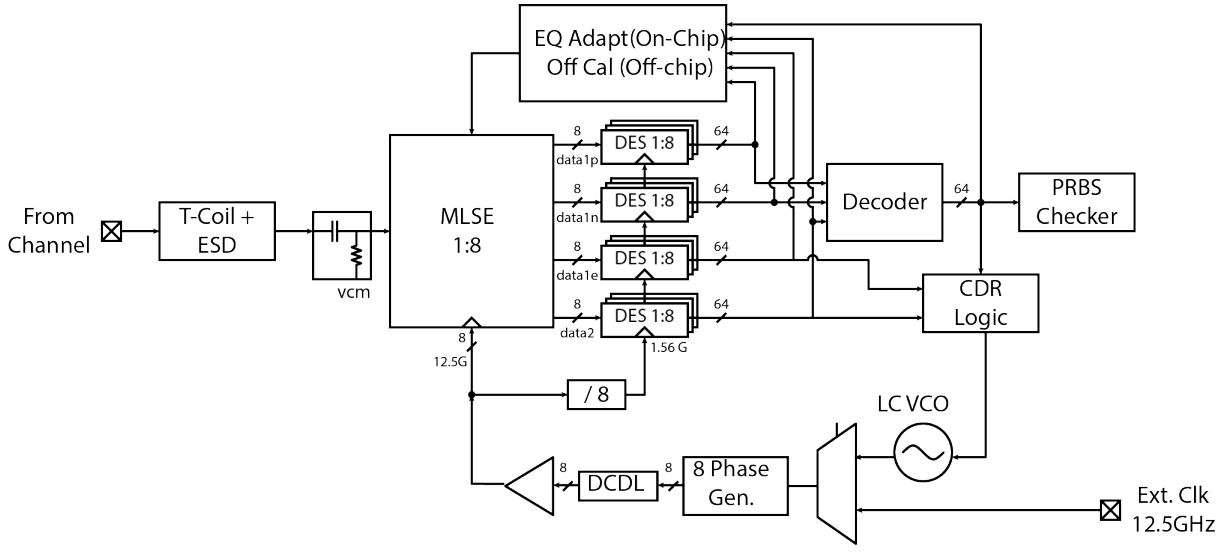


Figure 5.1: Block diagram of the receiver.

The receiver block diagram shown in Figure. 5.1 includes the datapath, the clock path and the digital back-end. The single-window length-2 MLSE design is chosen for its lower power consumption than window length-3 MLSE and error rates comparable to the DFE for $\alpha \leq 0.3$[31]. The receiver topology is based on the 160 Gbps receiver designed by Paul-Kwon

in [31] but is redesigned to optimize for the 100 Gbps data rate and further reduce power and area. CDR loop and the on-chip adaptation loop are added to perform the hybrid CDR algorithm. The T-coil structure designed by Kunmo Kim is used as the passive frontend to improve the bandwidth and signal integrity at the full-bit rate for the datapath and at the 12.5GHz external debugging clock input for the clock path.

The generator-based method using the Berkeley analog generator described in Chapter 1.1 is implemented in the analog design of the receiver. The merit of the generator-based design includes the reduced efforts to port the design to different specs with similar layout styles. This 100Gbps receiver design took advantage of hierarchical design and portability, utilizing the building blocks of the 160Gbps design.

5.2 Datapath

Architecture overview

The overall analog datapath block diagram is shown in Figure. 5.3 compared to its 160Gbps counterpart version in Figure. 5.2.

To save power, current integration techniques [26][31] are used for T&H, 8:16 deserializer, buffer, gain and summer in both designs. For N interleaving, there are NUIs in one sample period. There are three phases: integrate, hold, and reset during one sample period. In the integrate phase, the input voltage is sampled by the current integration. The final integrated value will remain during the hold phase, allowing the next stage to sample the hold voltage. Finally, in the reset phase, the voltage is reset to the default state in preparation for the next sampling. Ideally, we want the integrate phase in one stage to be within the hold phase of the previous stage to obtain the correct and stable input. The reset time should be long enough to prevent residual errors.

To achieve the desired throughput with reduced power, the datapath is implemented in 8 time-interleaved paths instead of 16 time-interleaved paths in the 160Gbps receiver. Deserialization after MLSE remains the factor of 8. This modification makes the operation speed of the slicers and digital back-end higher than that of the 160Gbps receiver. However, the increased speed (12.5GHz for MLSE slices and 1.56GHz for digital backend) is achieveable for the technology node. This change saves half the area and the required slicers and eliminates the 2-to-1 deserialization stage in [31]. The change further simplified the layout and routing. With the number of interleaving halved, the analog clock domain is reduced to one clock domain only, eliminating the need for dividers (Figure. 2.4).

The highpass filter and buffer stage are removed from the datapath for further area and power savings. highpass filter helps set the common mode to the acceptable range for the following 8-16 deserializer in a 160Gbps receiver. With downscaling of speed in the track-and-hold stage, the output common mode falls in the acceptable range for the following integrating latch stage and thus can be removed. The gain of the track-and-hold

and gain/summer increases with the larger integrating window due to lower speed. Therefore, the buffer stage is no longer needed for the budgeted gain.

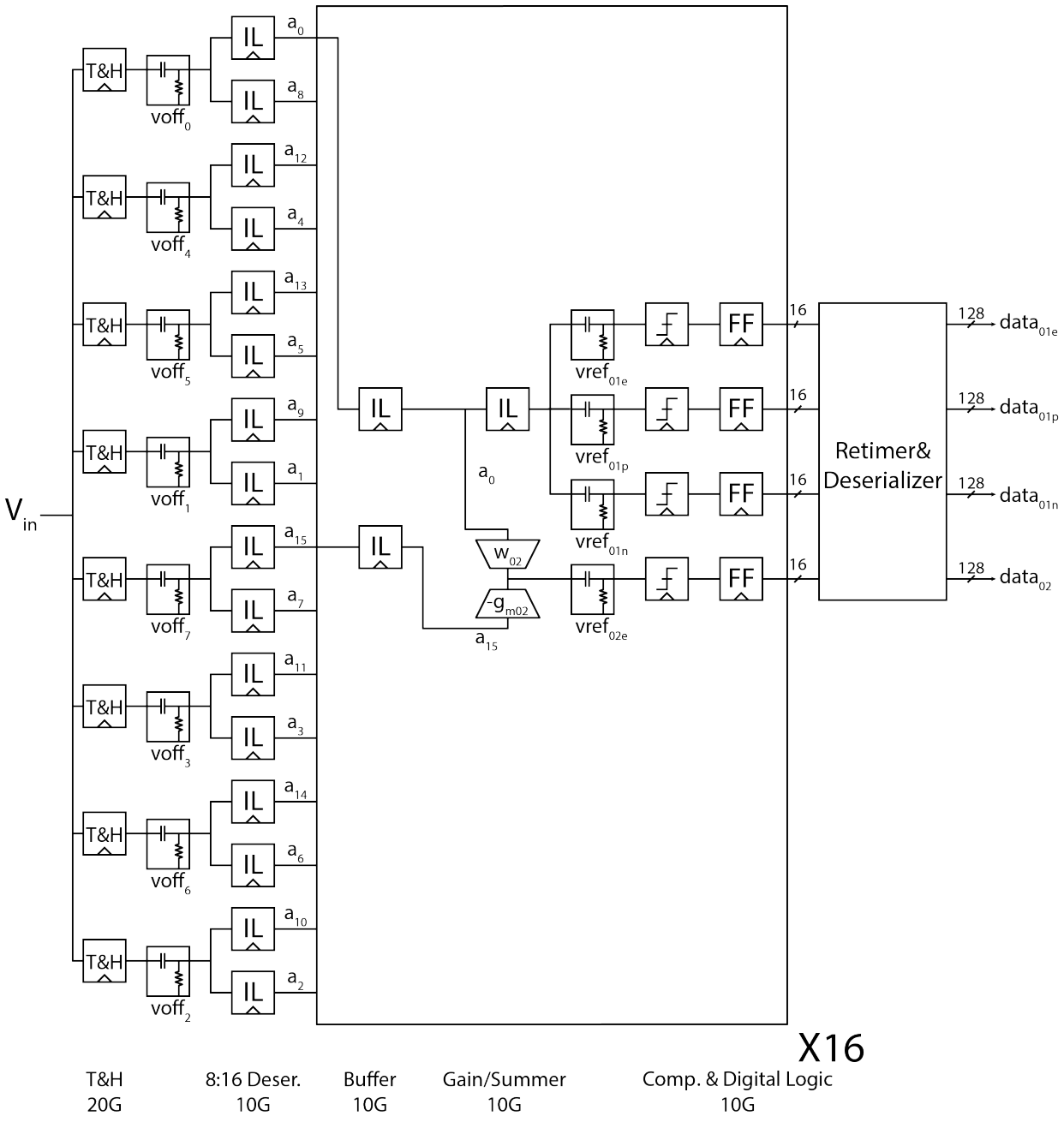


Figure 5.2: 160Gbps receiver[31] analog datapath block diagram.

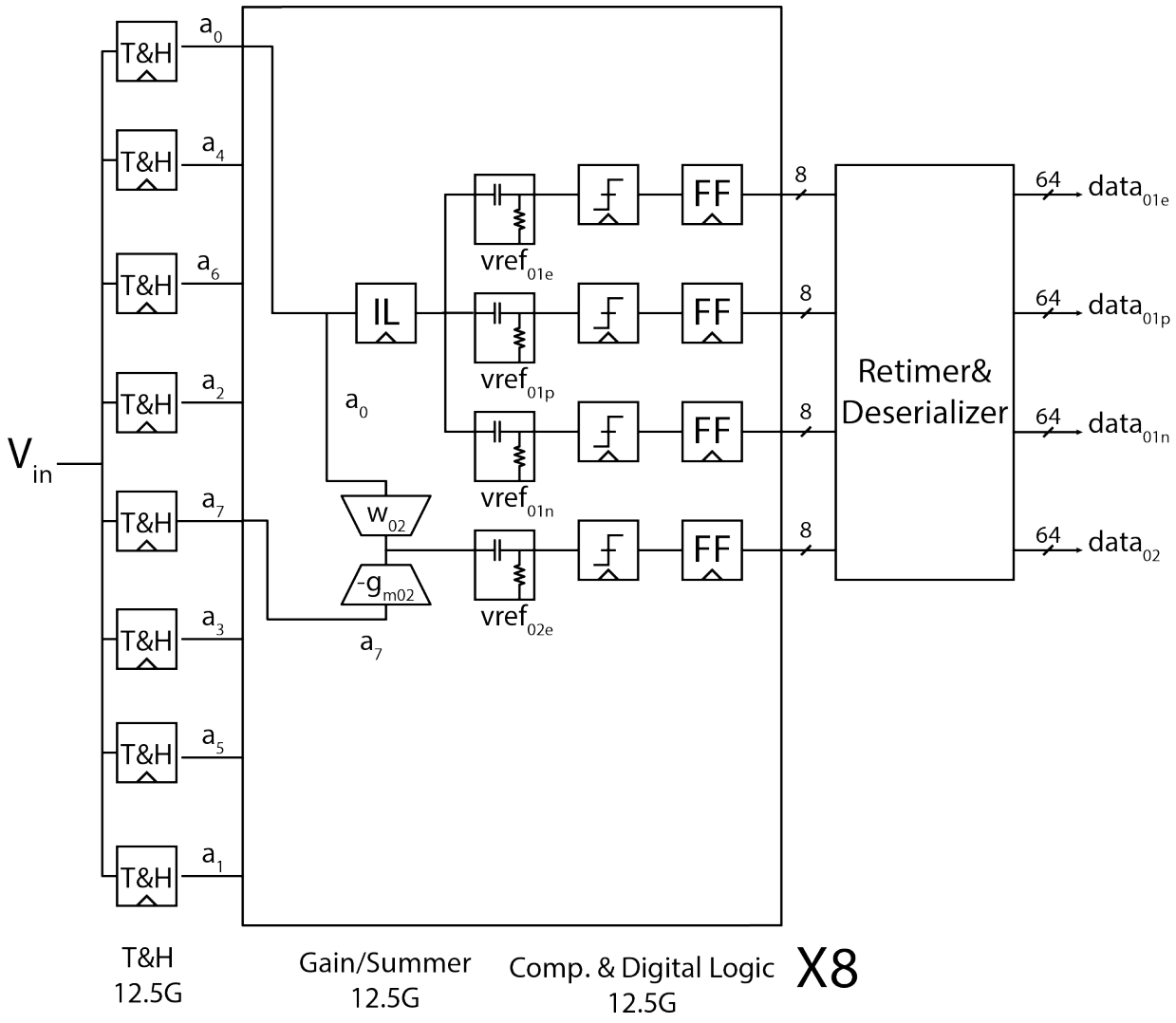


Figure 5.3: 100Gbps receiver analog datapath block diagram.

The resulting layout of the datapath is shown in Figure. 5.4. With the change in architecture, the overall layout reduces 25% in width and 47% in length.

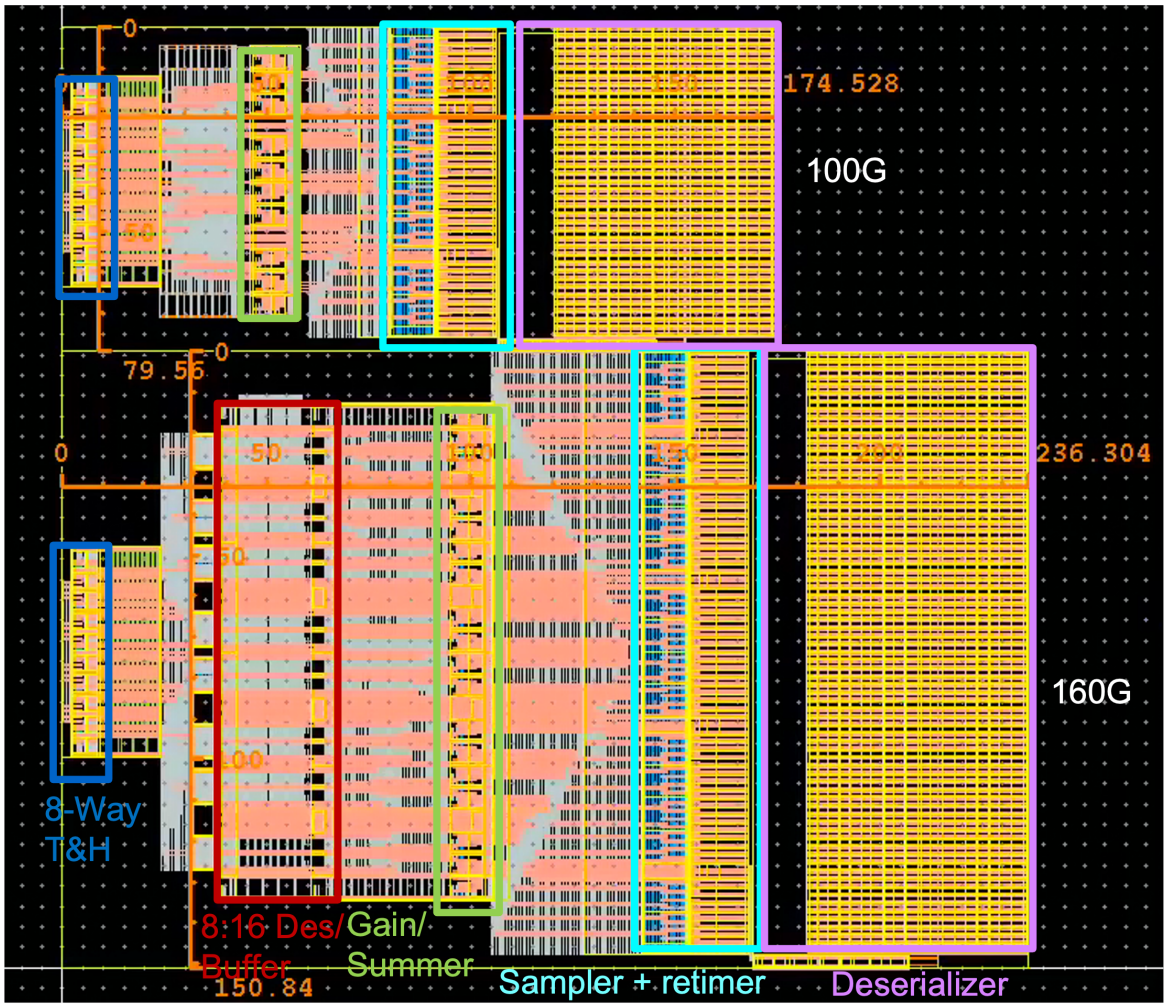


Figure 5.4: 160Gbps vs. 100Gbps MLSE analog datapath layout comparison.

Architeture change from 160Gbps Receiver

Gain and Summer

Both gain and summer functions are implemented with the integrating latches. In 160 Gbps design shown in Figure. 5.6, the summer integrating cycle (Reset/ Integrate/ Hold) is 16UI and Integrate: Hold: Reset is 4UI: 4UI: 8UI. Since the hold phase of the previous buffer stage is also 4UI, the overlapping hold phase between two signals is 3UI leaving 1UI not aligned exactly. The inaccuracy of the 1UI misalignment is compensated for with gain control in the 160Gbps receiver [31]. In the 100 Gbps receiver, the main challenge in integrating latched

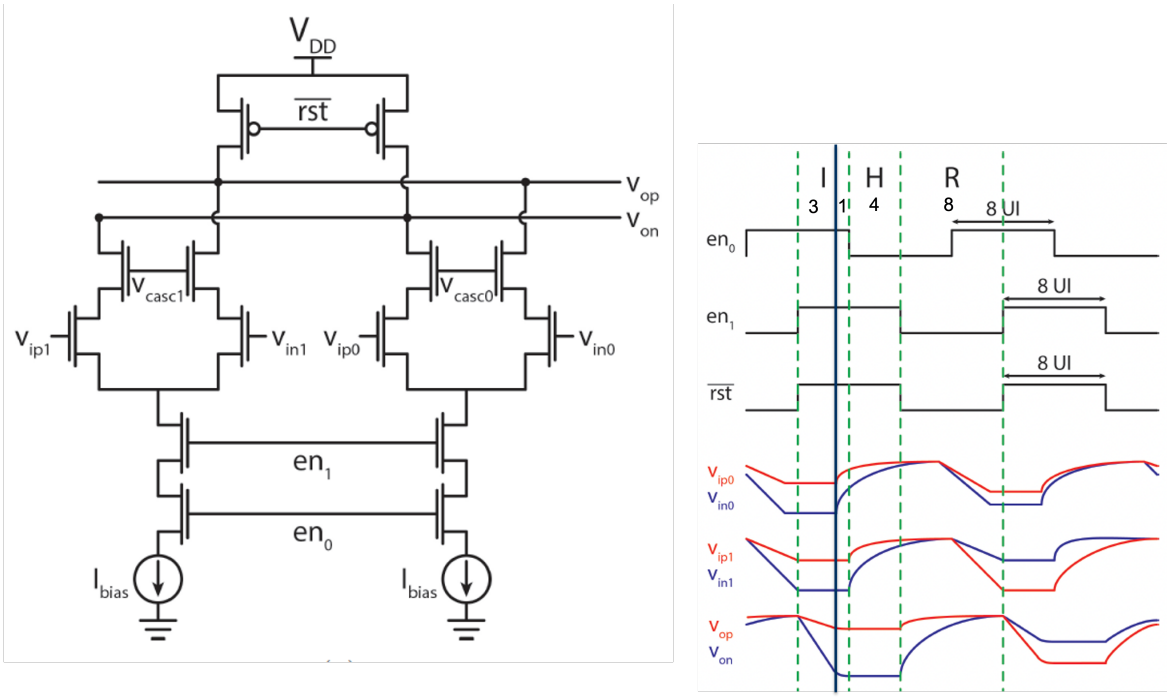
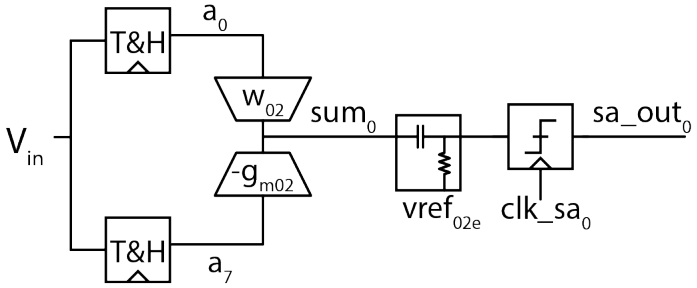


Figure 5.5: Summer schematic and 160 Gbps waveforms.

summer with the removal of 8:16 deserializer is to ensure that the summing operation occurs at the window in which two signals are aligned. There is 1UI delay between the current and previous inputs, which makes timing alignment critical to get the correct summation, especially when the 8:16 deserializer stage is removed. Due to the serialization factor being half of the 160 Gbps receiver, the integration phase was reduced to 2UI instead of 4UI. If we keep the buffer with the 2UI hold phase before the summer, the 1UI misalignment will account for 50% of the summer integrate phase, which turns into a significant inaccuracy. By removing the buffer and directly connecting to the T&H stage, the 3UI hold phase can perfectly cover the 1UI delay and perform the summation without misalignment. The timing

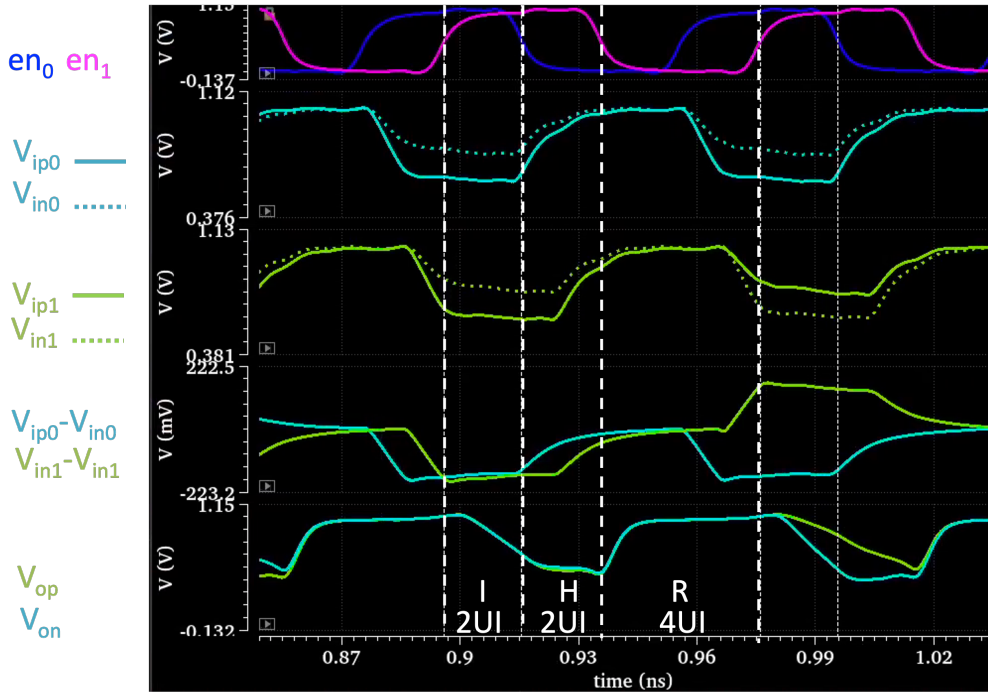
of the summer path including T&H and sampler is revealed in Figure. 5.6b. The simulation waveform is shown in Figure. 5.6c.



(a) block diagram.

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9
a_7	I	H	H	H	R	R	R	R	I	H
a_0	R	I	H	H	H	R	R	R	R	I
sum_0	R	R	I	I	H	H	R	R	R	R
clk_sa_0					↑					
sa_out_0							H	H	H	H

(b) Timing.



(c) 100 Gbps summer waveforms.

Figure 5.6: Summer path and its timing.

Retimer

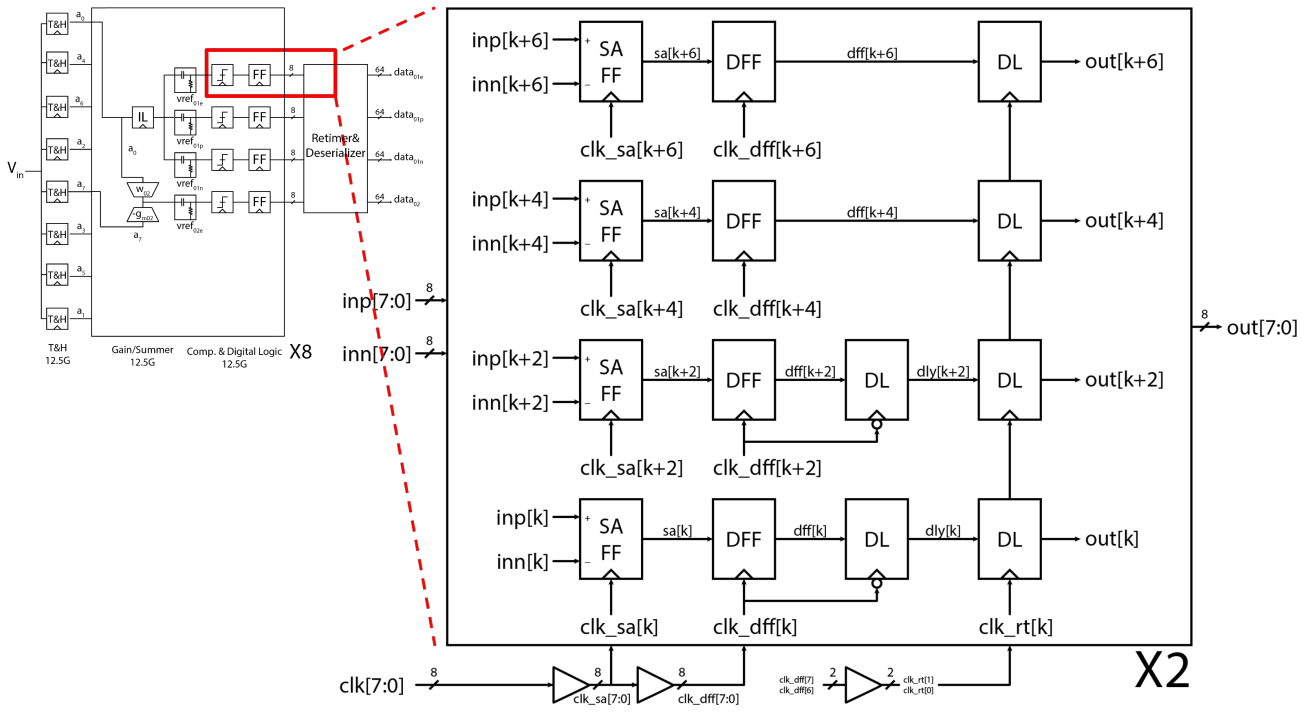


Figure 5.7: Sampler and retimer.

The strong-arm sampler and the retimer are inherited from the 160Gbps receiver designed by Paul-Kwon in [31]. The digitized data after the sampler need to be retimed and deserialized further to 1.56GS/s to process in the digital backend. The detailed description can be found in [31]. Since there is only 8-phase clock instead of 16-phase clock, the coarse retimer is shown in Figure. 5.7 only has two groups of quadrature clocks.

Adaptation

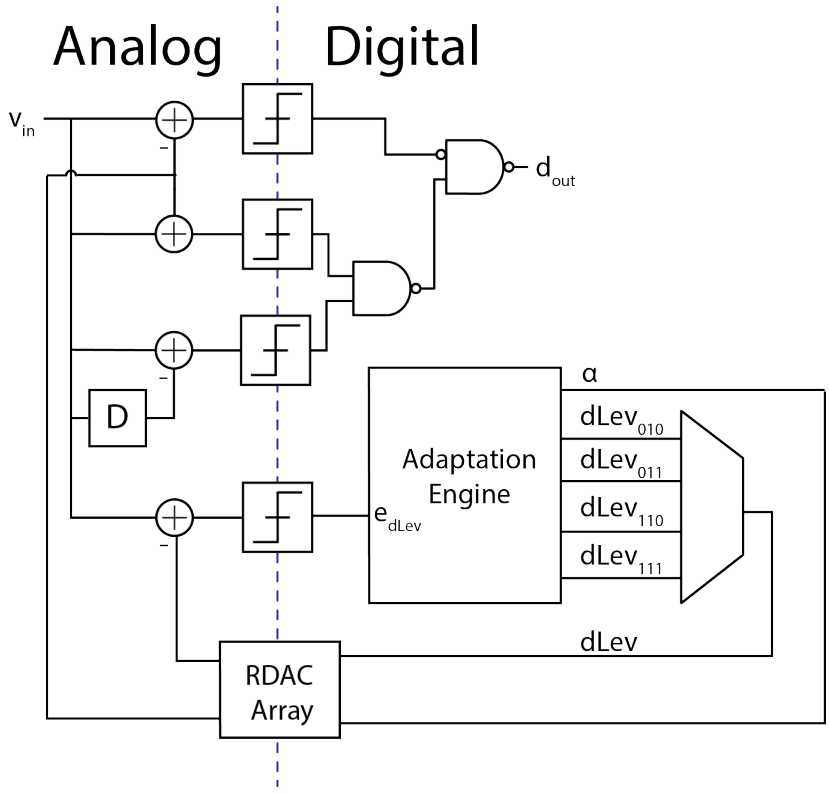


Figure 5.8: Adaptation Loop

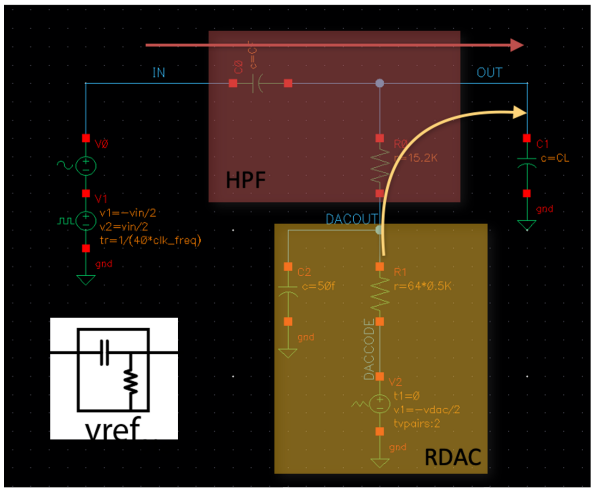
The adaptation engine for α and $dLev$ is implemented in the digital backend. To obtain α for the MLSE decoder, $dLev_{11}$ $(h_0 + h_1)$ and $dLev_{10}$ $(h_0 - h_1)$ are needed. However, $h_0 - h_{-1}$ is the target data level in the CDR hybrid algorithm to maximize eye opening. Therefore, four levels $dLev010, dLev_{011}, dLev_{110}, dLev_{111}$ are adapted by pattern filtering to extract $h_0 - h_{-1}$. The pseudo-code for the adaptation steps is described below.

```
# h0/h1 tap adaptation
# adapt_dlev_*1* -> dlev select
if d[n-1:n+1] == 110 and adapt_dlev110:
    dLev110 += mu_dlev * e_dlev[n]
if d[n-1:n+1] == 111 and adapt_dlev111:
    dlev111 += mu_dlev * e_dlev[n]
if d[n-1:n+1] == 010 and adapt_dlev010:
    dLev010 += mu_dlev * e_dlev[n]
if d[n-1:n+1] == 011 and adapt_dlev011:
    dLev011 += mu_dlev * e_dlev[n]

dlev11 = 0.5 * (dlev110 + dlev 111)
dlev10 = 0.5 * (dlev010 + dlev 011)
α = h1 = 0.5 * (dlev11 − dlev01)
h0-hm1 = 0.5 * (dlev110 + dlev010)
```

In the dLev maximization CDR algorithm, we try to maximize dLev and lock the clock phase at the maximum point. In order to perform the gradient descent algorithm in search of the maximum point, dLev has to be stable before the phase detector uses the error sample to determine whether going up or down. This implies that the dLev loop has to be faster than the CDR loop. Since the bandwidth of the CDR loop is about a few MHz, the bandwidth of the dLev loop has to be at least 100MHz for the dLev maximization to function properly.

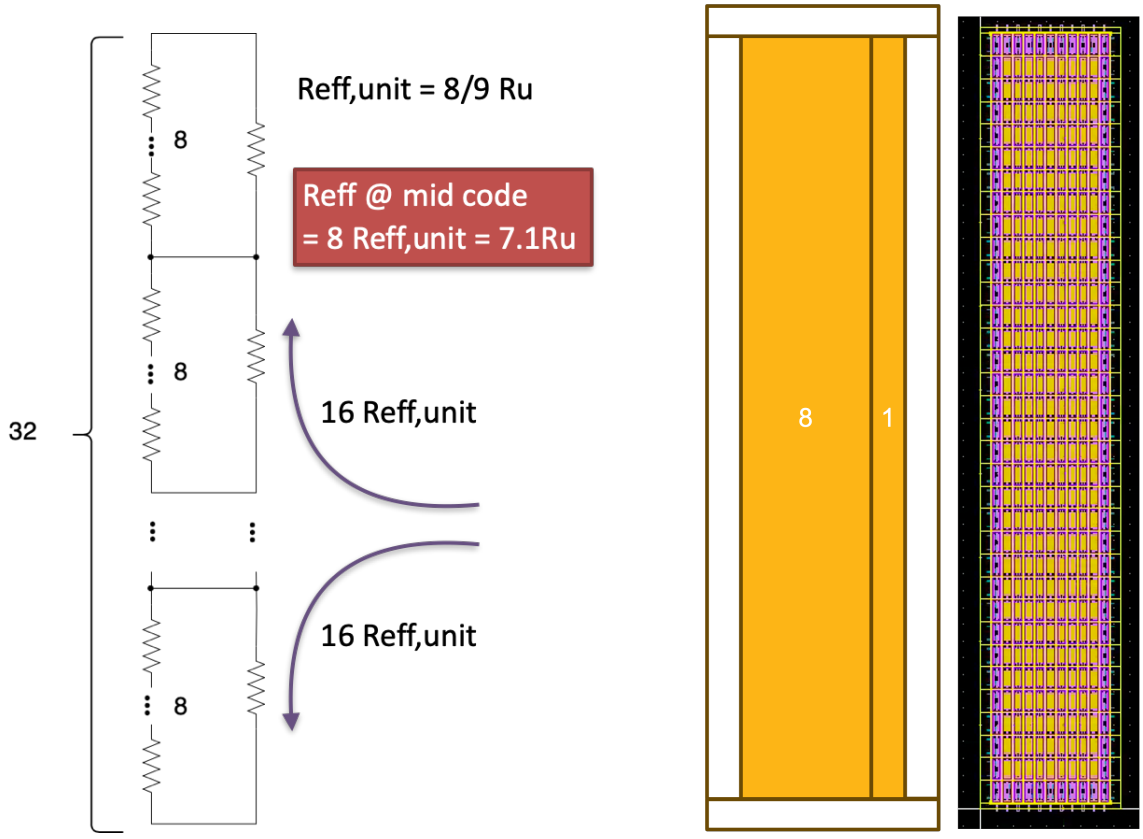
The RDAC array is a bottleneck in achieving a higher dLev loop bandwidth. The 8-bit RDAC array is used to convert the digital output code of adaptation to the analog reference level to compare with the input voltage and then generate the error sample e_{dLev}. The RC model of high-pass filter before the comparator in Figure. 5.3 and RDAC with mid-code configuration is shown in Figure. 5.9. The midcode (128 of 8 bits) is chosen since this is the maximum effective resistance seeing from the high-pass filter. A pole at 47 MHz is introduced in the dLev loop by RDAC with the standard series RDAC topology. For the same unit resistor, the number of resistors grows exponentially with respect to the resolution of the RDAC. The higher the resistance, the lower the frequency of the pole introduced by the RDAC. To reduce resistance at mid-code without compromising the resolution of the RDAC, we can connect a 3-bit RDAC in parallel with the 8-bit RDAC (Figure. 5.10). The effective resistance at mid-code configuration dropped from 64 to 7.1 unit resistance with the corresponding bandwidth enhanced to 170MHz. The bandwidth can be further increased if we connect a lower resolution RDAC e.g., 2-bit to lower the effective unit resistance. However, this implies more power consumption and we have 33 RDACs in total. The sub RDAC is added next to the main RDAC in the layout. The overall layout overhead is 12.5% in width. The minimum change in layout is important to maintain the top-level layout and reduce the rerouting effort and additional parasitics.



(a) High-pass filter RC model.

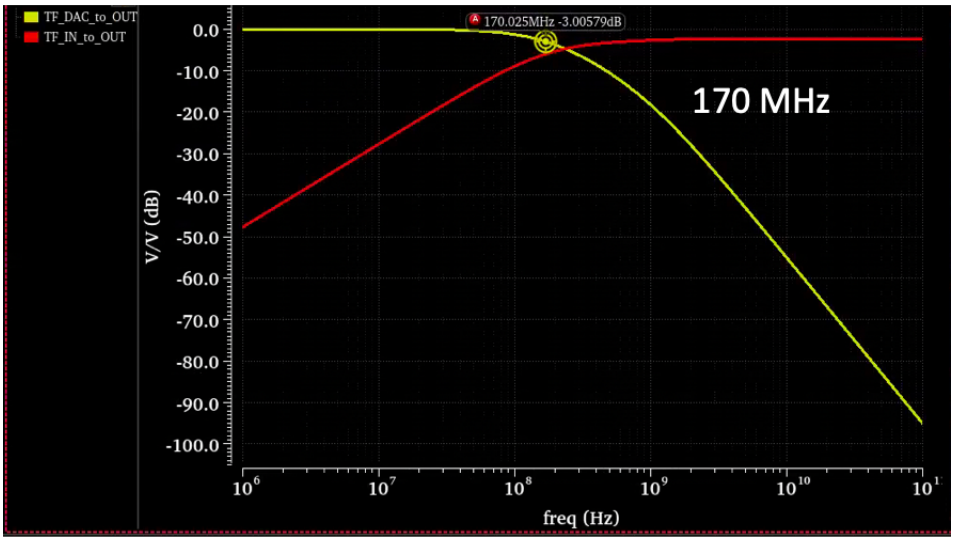
(b) High-pass filter bandwidth.

Figure 5.9: Pole introduced by RDAC.



(a) Parallel RDAC schematic.

(b) Parallel RDAC floorplan and layout.



(c) High-pass filter bandwidth with parallel RDAC.

Figure 5.10: Parallel RDAC.

5.3 Clock path

Voltage-Controlled Oscillator (VCO)

The voltage-controlled oscillator (VCO) is used to generate the clock for CDR operation. The timing information obtained by the phase detector in the CDR logic determines that the clock is early or late than the ideal sampling phase, and the output feeds through the loop filter, which generates the control bits for VCO to adjust the phase by slightly changing the clock frequency.

The VCO is implemented with the LC oscillator, which provides better phase noise than the ring oscillator structure. The inductor area is usually the disadvantage of LC VCO. The required area increases with both inductance and capacitance. However, the area scales down with the operating speed determined in Equation 5.1, making LC VCO a good candidate for high-speed wireline link.

$$\omega = \frac{1}{\sqrt{LC}} \tag{5.1}$$

The frequency varies with both the inductance and the capacitance. In reality, capacitance is the feasible knob to tune the frequency. The tunability is implemented with a cap DAC (CDAC) for coarse tuning and a varactor for fine tuning. Shown in Figure 5.11, the VCO includes the cross-coupled CMOS pair, LC tank, CDAC and varactors. The current enable switch used to turn off the VCO while the external clock is chosen for debugging and octature generator characterizing purpose.

The performance of the VCO depends on the layout and requires thorough verification. The two versions of the layout are shown in Figure. 5.12. The layout order is similar in both layouts: inductor, current mirror, cross-coupling pair, varactor, CDAC, and buffer from right to left. To obtain the better quality factor, the inductor is implmented with the top metal layer, and thus the performance degrades if placed right below the C4 bump. To escape from the C4 bump, the layout should be as compact as possible (Figure. 5.13). The compact layout also reduces the parasitics in the routing. For careful verification, the VCO is separated into two parts: the top layers of metal using EMX to extract the S-parameter file and the lower layers of metal using RC extraction for post-layout simulation. Custom fill is placed in the inductor area to avoid parasitics from random filling for the density design rule.

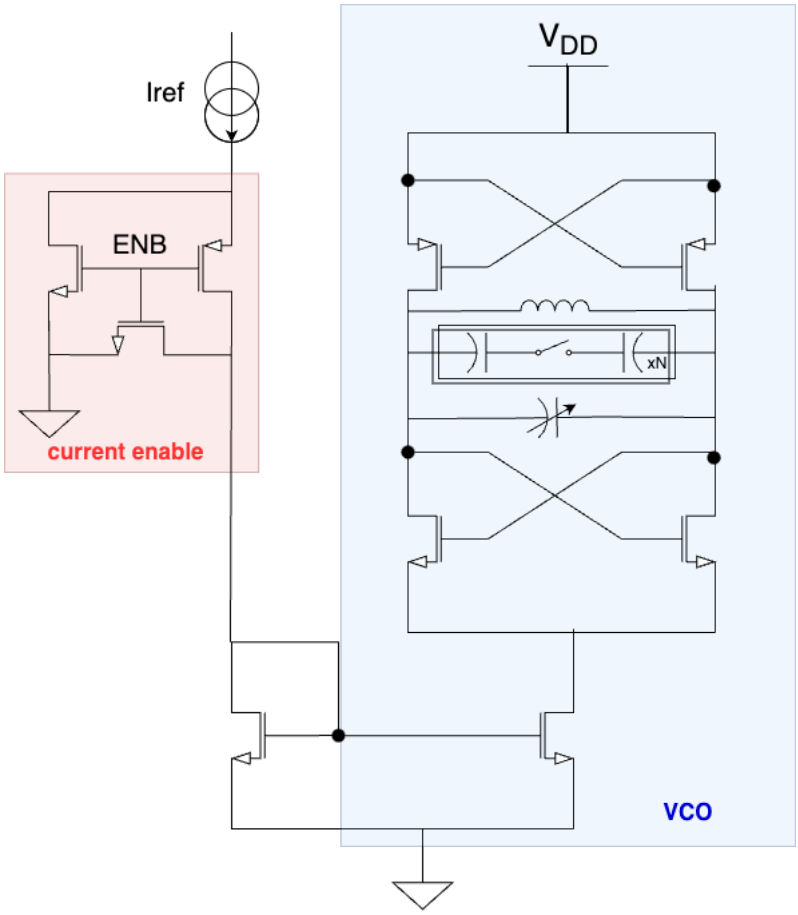


Figure 5.11: Schematic of VCO.

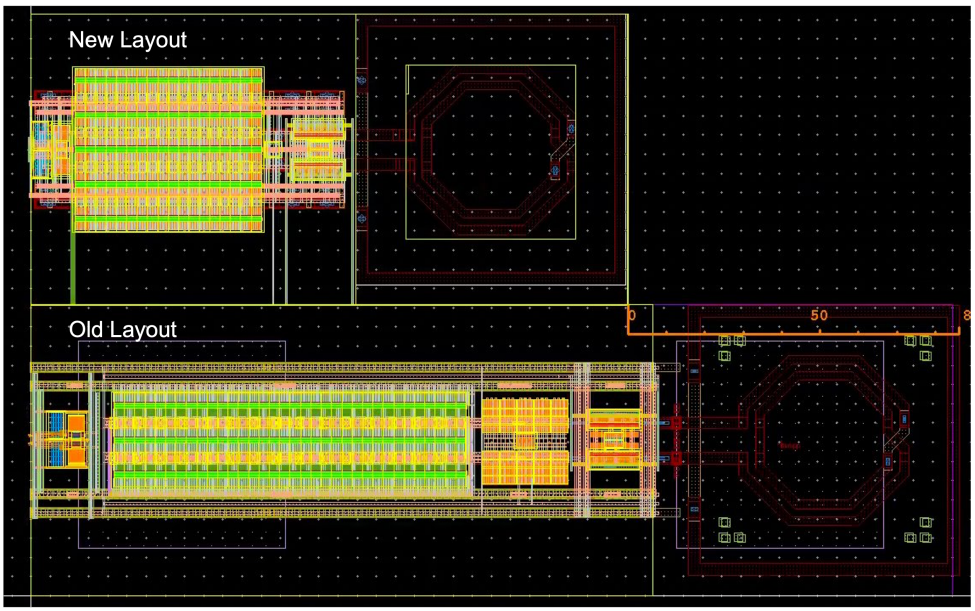


Figure 5.12: Layout of VCO.

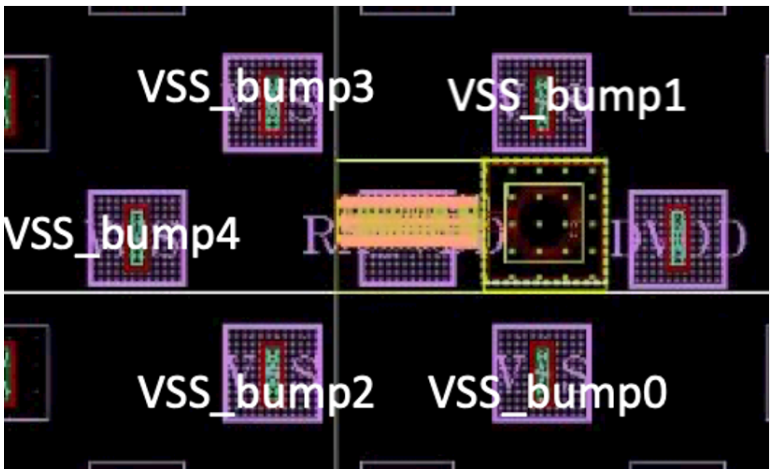


Figure 5.13: VCO placement with C4 bump.

CDR Loop Implementation

The phase detector logic is implemented in the digital domain. The block diagram of the digital part of the CDR loop is shown in Figure. 5.14. PD formatter logic includes the formatter aligning the inputs and the configurable PD logic. There are two CDR locking modes: Mueller-Muller mode and hybrid mode. **data** is the decoded data from the MLSE decoder. **data2** and **data1e** are from the output of the sampler. **dn_d** and **up_d** are the previous PD decision used to estimate the slope dLev as described in Chapter 3.3. The decision of the PD logic then goes through the voter and is filtered by loop filter which consists of integrating and proportional path. The output of the loop filter will be separated into coarse band and fine band to control CDAC and varactor in the VCO, respectively. An override function is added at each stage to ensure visibility and adaptability in testing.

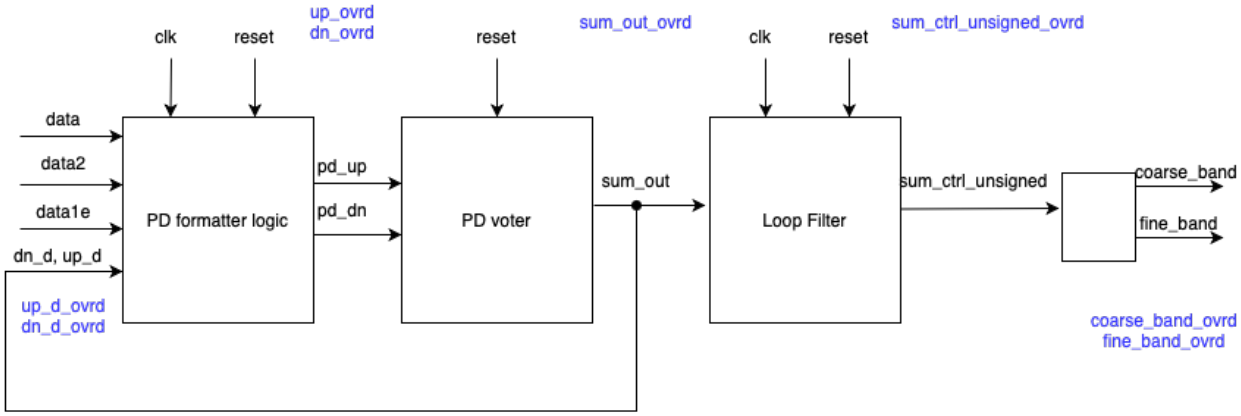


Figure 5.14: Block Diagram of CDR digital backend.

The overall receiver with the complete CDR loop is shown in Figure. 5.15. Datapath and clock path are reusing the 160Gbps structure but optimize for 100Gbps with topology change and block-level retuning. The CDR PD logic and adaptation loop is added for the hybrid CDR.

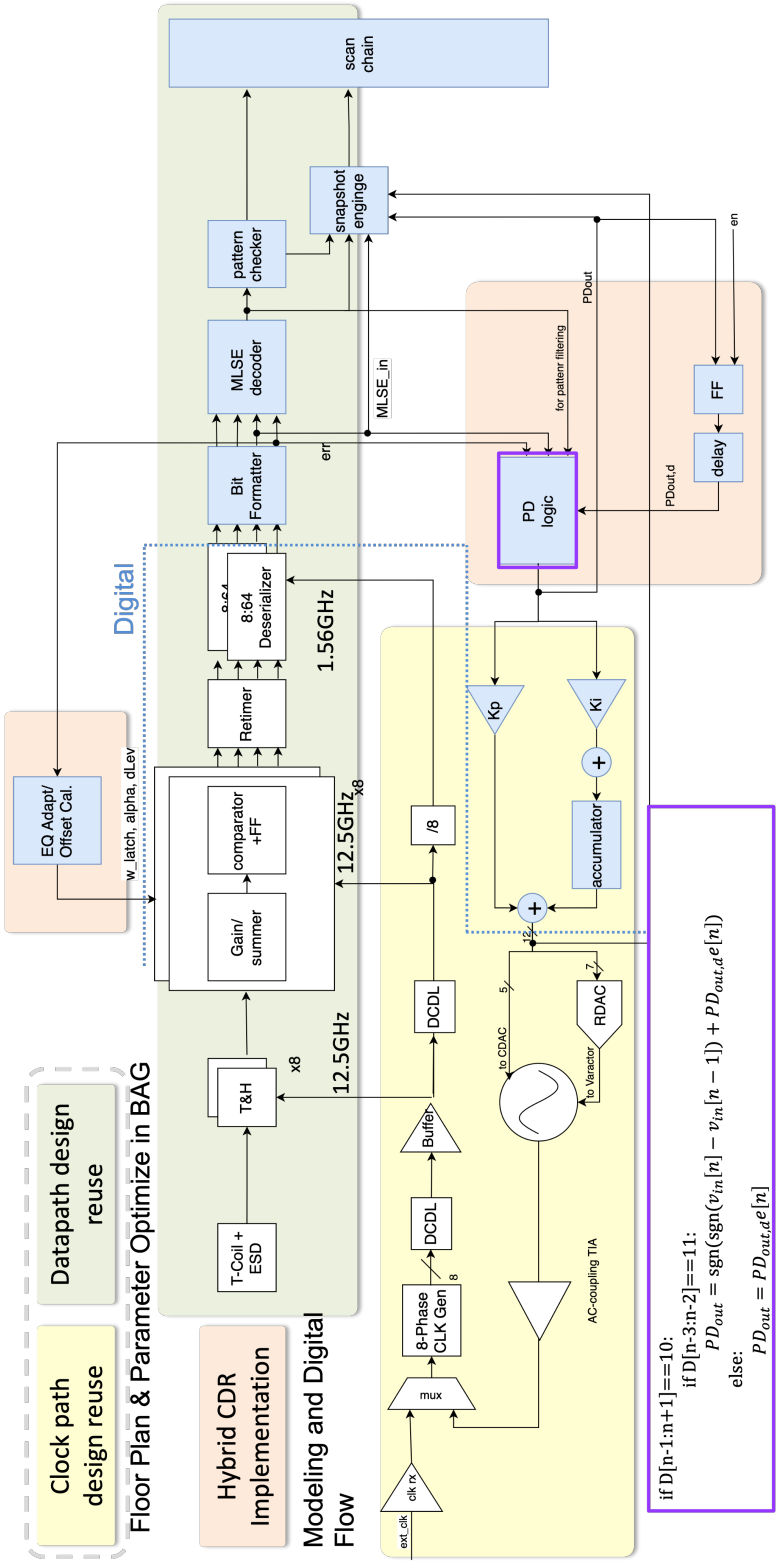


Figure 5.15: Block diagram of CDR loop.

Chapter 6

Integration and Verification of 100 Gb/s 1-Tap MLSE Receiver with Baud-rate CDR

6.1 Integration

The analog part of the chip is designed and laid out by Berkeley Analog Generator (BAG). RTL implementation of digital blocks is generated by Constructing Hardware in a Scala Embedded Lan- guage (Chisel) and Python script with Jinja2 template with configured parameters. Highly Agile Masks Made Effortlessly from RTL (Hammer) are used to perform digital flow including synthesis and place and route, analog and digital interface integration, chiptop level placement including ESD. The overall flow is visualized in Figure. 6.1.

The top level chip integration is shown in Figure. 6.2. The total chip size is 2mm x 2mm. The analog core is placed at the upper left corner of the chip, while the digital backend is placed and routed on the right side. The bottom side consists of the power domain for the clock and the datapath. The customized power grid with decoupling capacitor is modularized and designed to be compatible with the bump size to ease the deployment of the power grid.

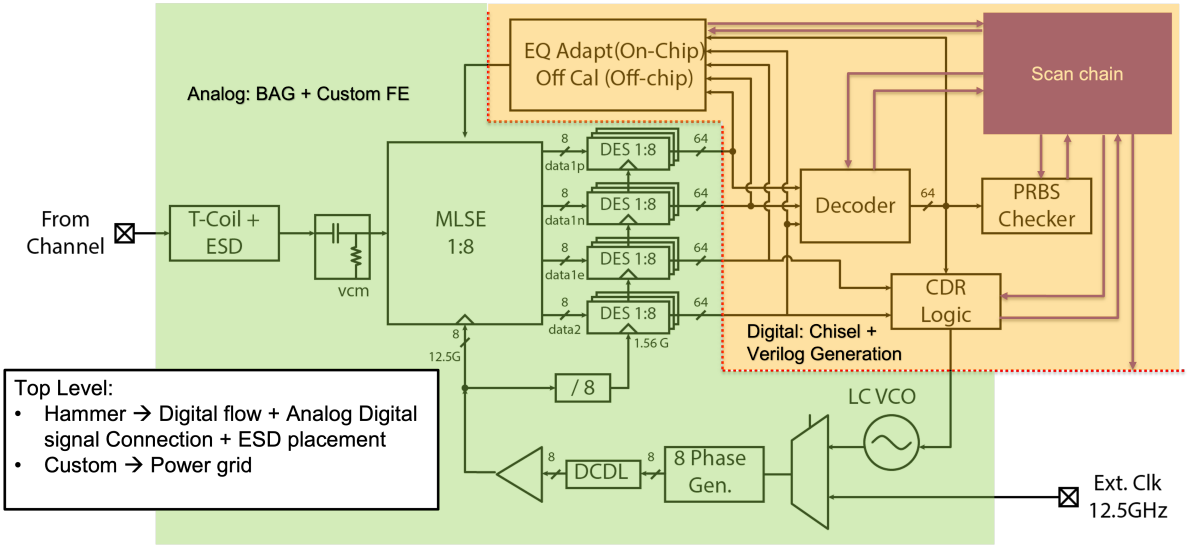


Figure 6.1: Top-level integration flow.

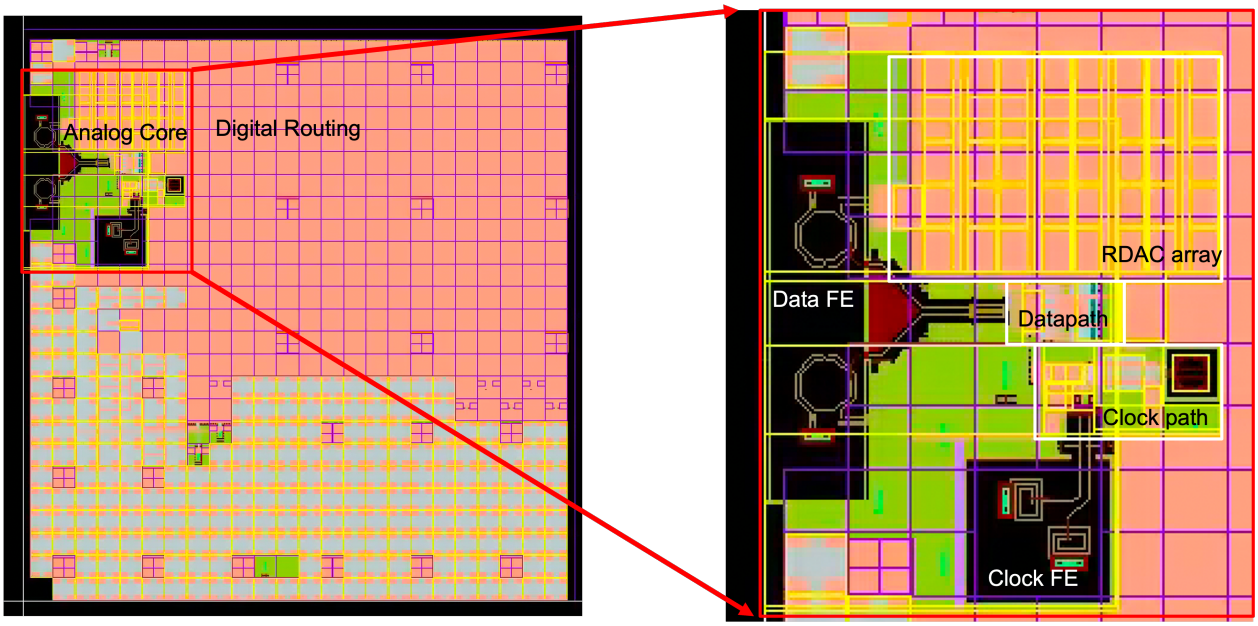


Figure 6.2: Top-level layout integration.

6.2 Verification

Verification of the entire link requires systematic modeling and a careful setup of the test bench. The co-simulation of the analog and digital part of the link system is important to verify the functionality and performance of the link. The verification part is crucial in the CDR system, since it includes feedback in the overall system and requires verification at the top level with the integration of all blocks to ensure its functionality. The verification of the 100Gbps receiver taped out in 16nm process is described in this section.

State-of-the-art

1. Verilog-AMS

   The conventional way to verify mixed signal circuit is by applying the verilog file in the analog environment and performing the time-integrating cosimulation. The accuracy of the simulation increases with the decrease in step size in time. In other words, to get accurate results, the simulation time needs to be long enough.

2. XMODEL

   XMODEL is a commercial extension to the SystemVerilog simulator that expresses analog waveforms in functional expression that can be computed in an event-driven way. There are also some built-in primitives that mimic analog circuit operation in SystemVerilog. The accuracy is decided by how close the functional expression is able to describe the analog waveforms and is independent of the integrating time. Due to the decoupling of accuracy and simulation time, it can achieve a speed-up of up to $10 \sim 100$ times over Verilog-AMS [16].

Event-Driven Model

In our verification system, we apply the similar concept of event-driven modeling in XMODEL. The primitive is not from an existing libraries but was created by us using the top-level model generated by BAG3 netlister. The hierarchical model generation is illustrated in an example (Figure. 6.3) of constructing an 8-way T&H circuits. The leaf cell template of the inverter (XINV) and integrating latch amplifier (XAMP) are written in Jinja2, an expressive and extensible templating engine. The parameter of the behavioral model, such as delay, can be defined in a separate yaml file or passed from the simulation. With an event-driven approach, the value is calculated only when the trigger arrives.

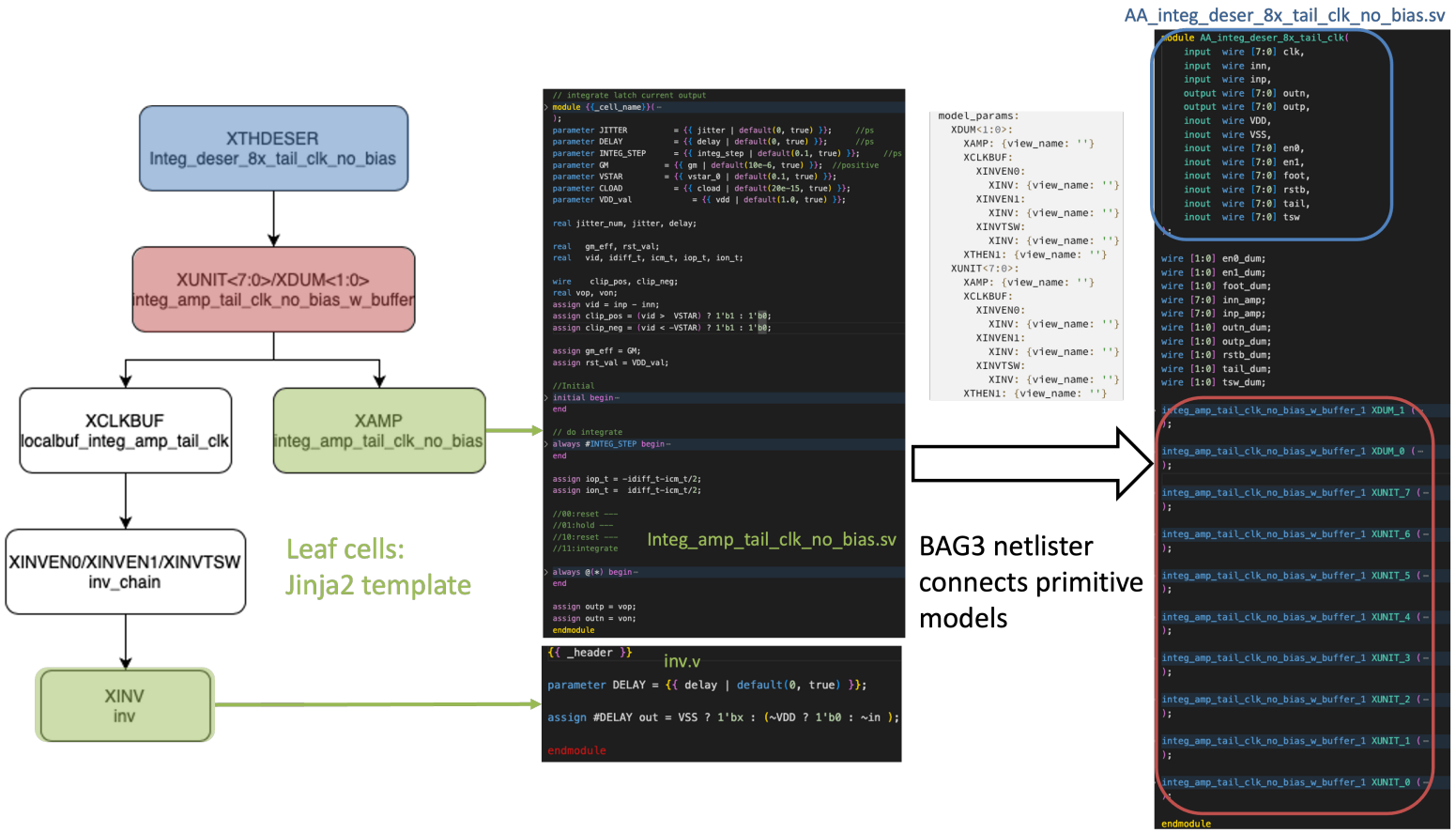


Figure 6.3: Hierarchical behavioral model generation.

Figure. 6.4 shows the block diagram of the testbench of the entire RX system. The testbench is implemented with the SystemVerilog behavioral model for analog blocks and cosimulates with Verilog RTL for digital blocks using the digital flow. The yellow parts are the analog circuit mapping to the SystemVerilog model, and the blue part is the digital Verilog file. The effect of TX and the channel is captured in the SystemVerilog model that takes into account the transfer function of TX and the characteristic of the channel. The model is event-driven and only updates at the sampling point of the receiver clock. The detailed modeling process is described in Appendix. B.

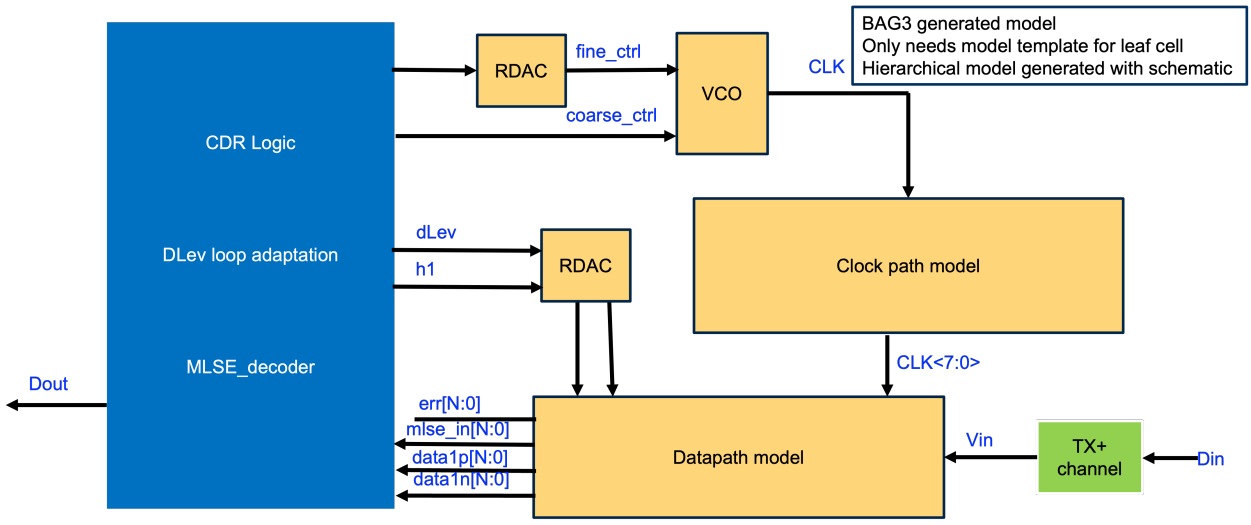


Figure 6.4: Event driven SystemVerilog testbench.

Test Vector Generation

To fully verify the building blocks on the digital side, we established the flow with script and MakeFile to generate the building blocks with different parameters, matching random testvectors, and the testbench includes the building block to test and the testvectors. The simulation will be run automatically and give the results at the end of the flow. This verification flow is significantly important given the complexity of the system.

Verilog Module Visualization

As we start to put all the building blocks together, the complexity of the inputs, outputs, and the connections between the submodules will become difficult to manage manually. To make verification easier and more intuitive, the visualization function is included in the testbench generation process. Figure. 6.5 is an example of visualization results that has input as the blue triangle on the leftmost part and red circles on the rightmost part. The port names of

each instance are marked as a black cross, and the connections are represented by lines with the net name annotated. This can be used to check if the connections are correctly matched and if there are any missing connections.

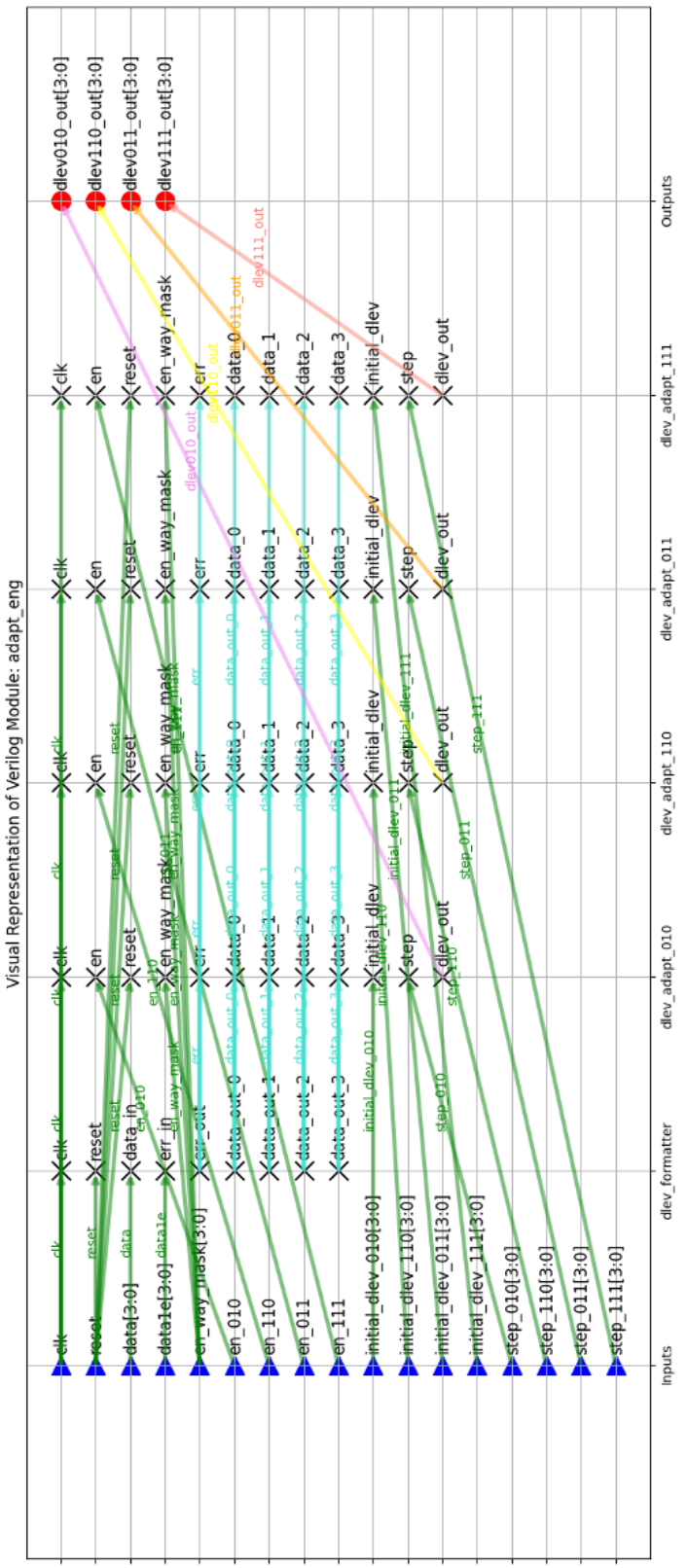


Figure 6.5: Adaptation engine visualization.

6.3 Testing

Test Setup

The test setup for the 100Gbps receiver is shown in Figure. 6.6. Three supply domains to the test chip are generated by LDOs on the PCB board. Reference bias currents are generated by on-board current sources. 100 Gbps input data are transmitted by probing with a GSGSG probe. The 12.5GHz input clock is a testing option for debugging. The input clock to the receiver clock distribution is by default connected to the clock generated by the VCO. 1.56GHz output clock is divided from the clock path and is used to check the operating frequency and if the octature generator is locked. The scan chain delivers the data from the test chip snapshot engine, providing visibility for off-chip offset calibration and debugging purposes. The circuit configuration and overwrite function are set via the scan chain.

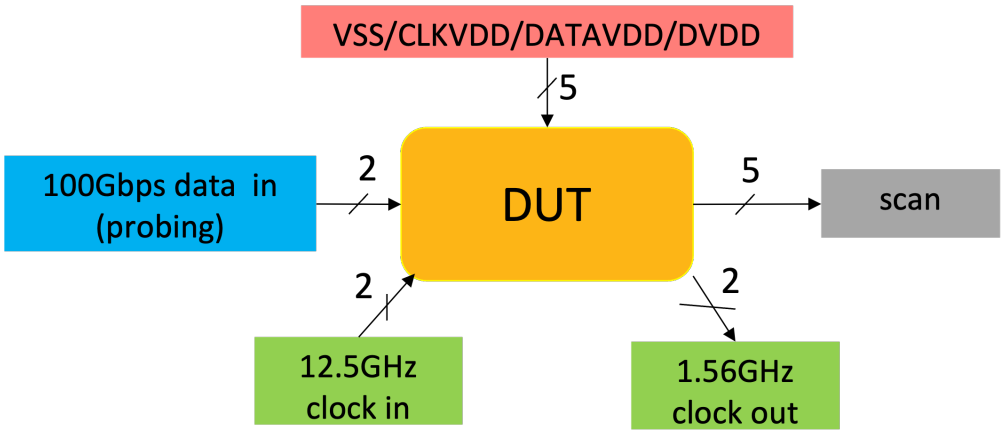
There are three PCB boards in the test setup. The main board includes the DUT chip, LDOs, and current sources. The level shifter board reused from the 160 Gbps transceiver in [18] shifted the signal voltage between the voltage domain on the main board and the FPGA board. FPGA board communicates between the PC and the other boards.

The test chip including 100Gbps receiver with baud rate hybrid CDR is tested with the probed input data from the Keysight M8195A arbitrary waveform generator (AWG). 12.5GHz external input differential clocks are generated by the Agilent E8267D signal generator. The external 3.3V power supply connects to SMA and the on board regulators generate the configurable supply for on-board current source and data/ clock/ digital VDD on chip. The configurable supply and reference current is controlled by the FPGA board. The scan chain is also controlled by the FPGA board to configure the test chip setup and read the data from the chip. The 1.56GHz divided clock generated from the clock path on the chip is connected to a spectrum analyzer for debugging purposes.

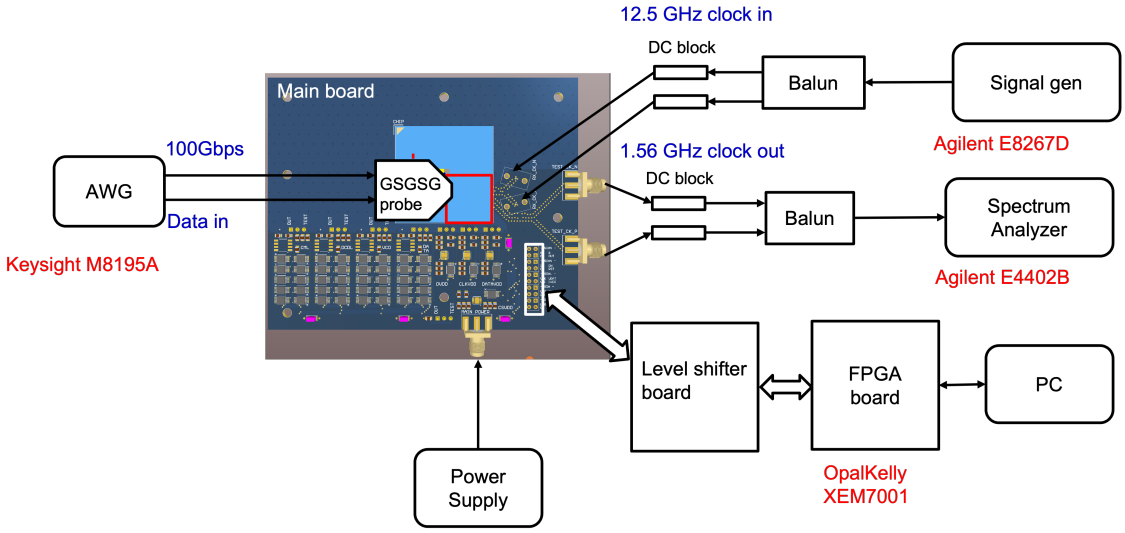
PCB design

The main PCB shown in Figure. 6.8 is designed to support the testing of the 100 Gbps receiver taped out in a 16 nm FinFET process. All signals go right and bottom with the upper left reserved for the probe to move freely from left. The dimension of the main board is important since it needs to be fitted within the probe station whose radius is 4 inches.

To have a clear space for the probe to move freely, the pin heads that sit above the LDOs are designed to protrude on the bottom sides. They are used to connect the supply to LDOs or an external supply. Digital potentiometers are used for both LDOs and current sources for programmability through the FPGA. There are 6 layers in the PCB stack up. Roger material is used for the high-speed stripline clock interface. FR4 material is used in other layers where low-speed signals are located to lower the cost.



(a) Interface of the test chip.



(b) Detailed testing setup with instruments.

Figure 6.6: Testing setup.

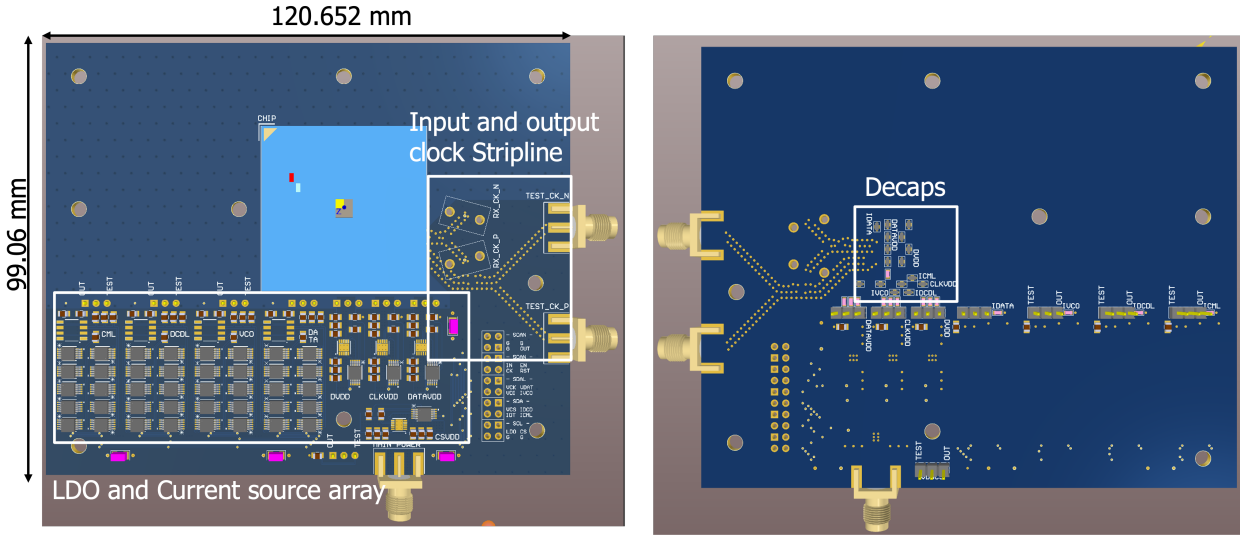


Figure 6.7: Top view and bottom view of PCB.

Stackup Information

	Type	Material	Parameter	Residual Copper	Thickness(mm)
L1	Core	RO4350B	0.5_PT_1OZ 0.254(exclude copper) 0.5_PT_1OZ	100 / 33	0.042 0.254 0.035
L2					
	PP	RO4450F	4450(80)+4450(80)	/	0.1759
L3	Core	S1000-2M	1OZ 0.8(include copper) 1OZ	88 / 38	0.032 0.73 0.032
L4					
	PP	S1000-2MB	1080(64)+106(74)	/	0.1101
L5	Core	S1000-2M	1OZ 0.1(exclude copper) 1OZ	87 / 100	0.032 0.1 0.035
L6					

FinishBoardThickness:1.60(+0.16/-0.16)mm PressBoardThickness:1.54mm

Figure 6.8: 6-layer PCB stack-up.

Measurement Results

Before testing with data input, the clock path is tested and verified with the 1.56GHz output debug clock. VCO and clock distribution are tested and characterized with FPGA, signal generator and spectrum analyzer. The testing setup is shown in Figure. 6.9.

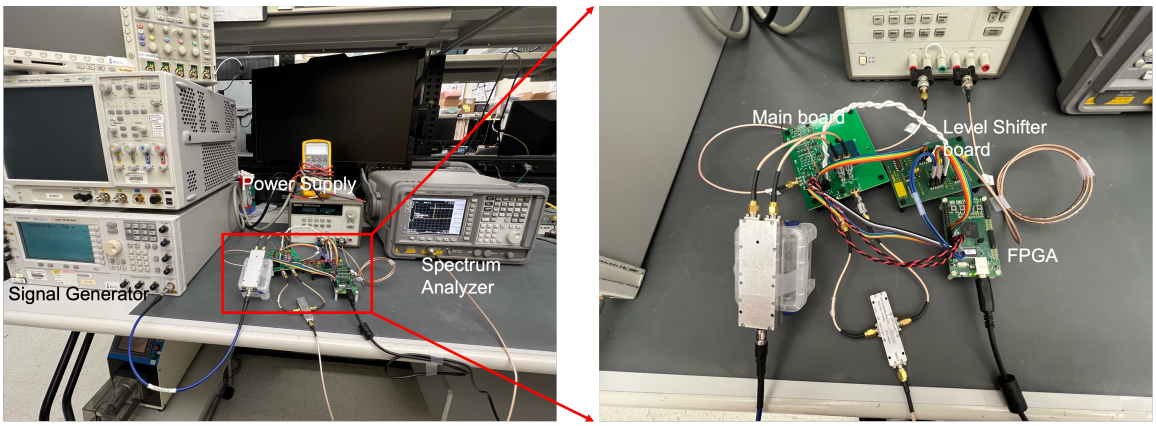


Figure 6.9: Testing setup for clock path.

The VCO free-running frequency range is characterized by choosing the input clock from the VCO clock and disconnecting the external clock source with 50-ohm terminator connected to the k connector. The measurement is taken at a clock VDD of 1.15V. The measured VCO frequency tuning range is close to the post-layout simulation shown in Figure. 6.10 ranging from 11.5GHz to 15.5GHz. The frequency tuning range from the simulation is 3.8GHz and that of the measurement is 4.0GHz. The VCO range is all within 11.3GHz to 15GHz across five tested boards.

There are two sources for the input clocks of the octature generator. The clock source could be generated either by on-chip VCO or a signal generator that passes through the k-connectors to the board. The FPGA configured which mode to use for the clock distribution, and we can observe the divided clock from the clock path on the spectrum analyzer through the SMA connectors on the PCB. The measurement results of the two modes are shown in Figure. 6.11a with the sweep of the supply voltage to the octature generator. The free-running frequency is matched when the supply is less than 0.85V. While the supply is larger than 0.9 V, the frequency is above 10.5GHz and the measurement mode with vco clk connected is affected by the VCO whose freerunning range is 11GHz to 15GHz. It could be that even with the VCO current disabled by switching off the reference current, there is still a leakage current that contributes to the small oscillation of the VCO.

The free-running frequency of the octature generator is measured with disconnecting the VCO clock source and choosing an external clock source with scan chain. However, the k connector is not connected to the signal generator, but is connected to a 50 ohm terminator to eliminate noise from the K connector to the injection port of the octature generator. The

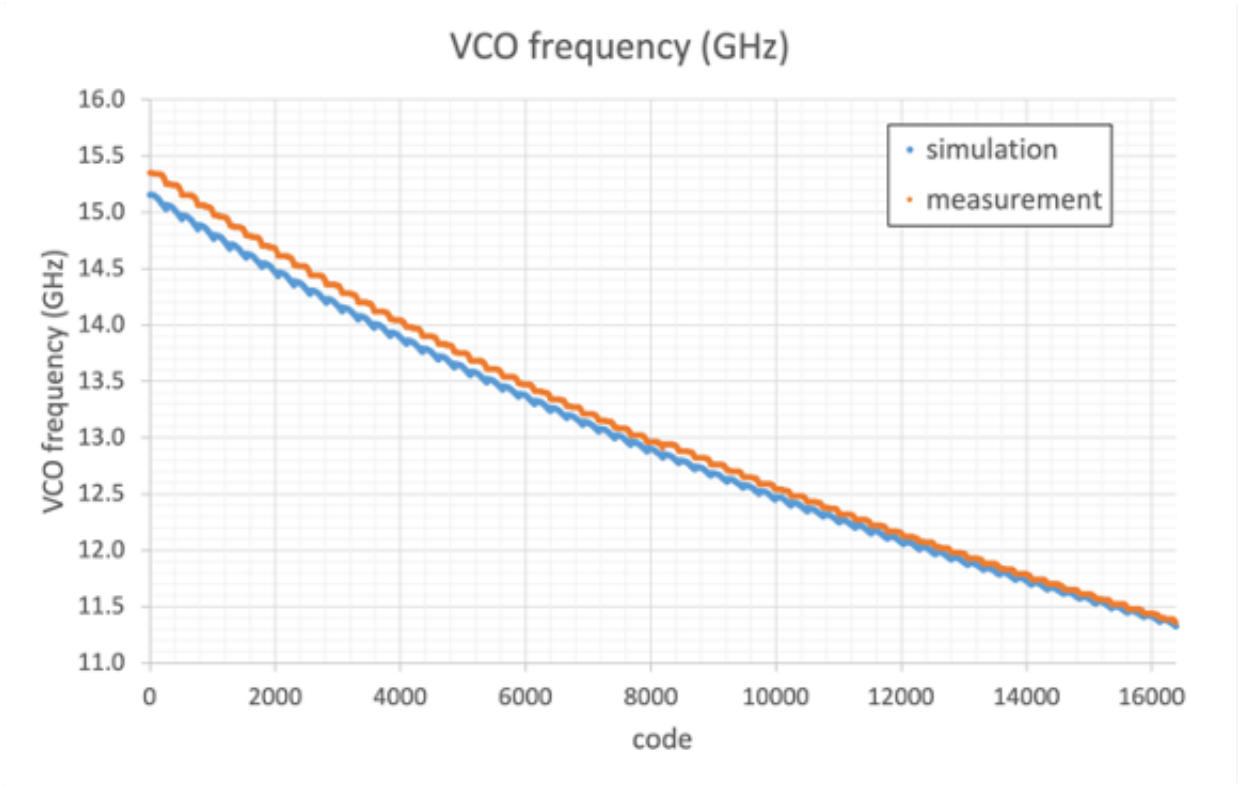
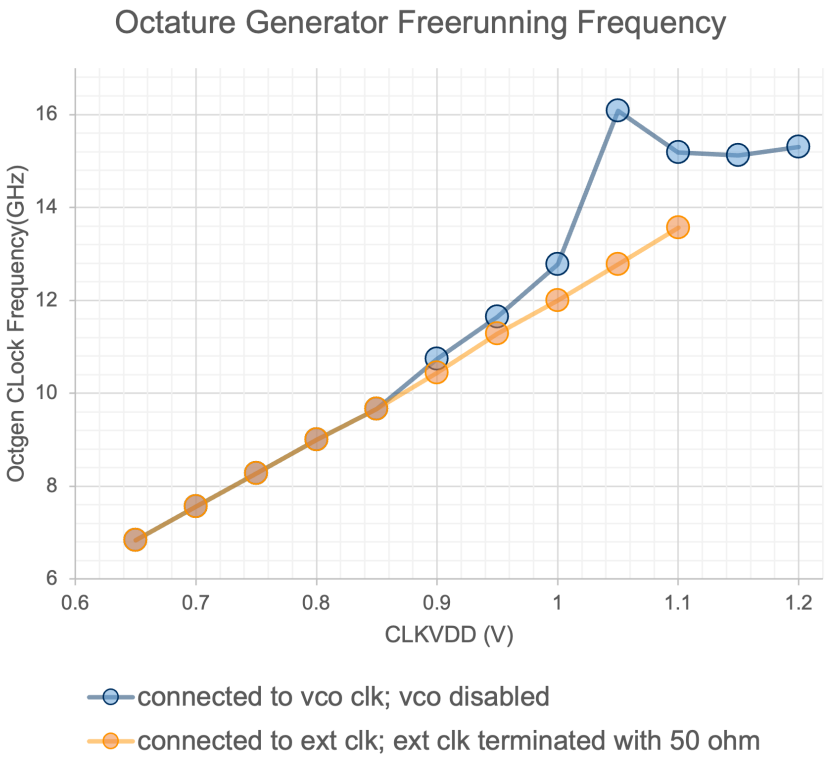


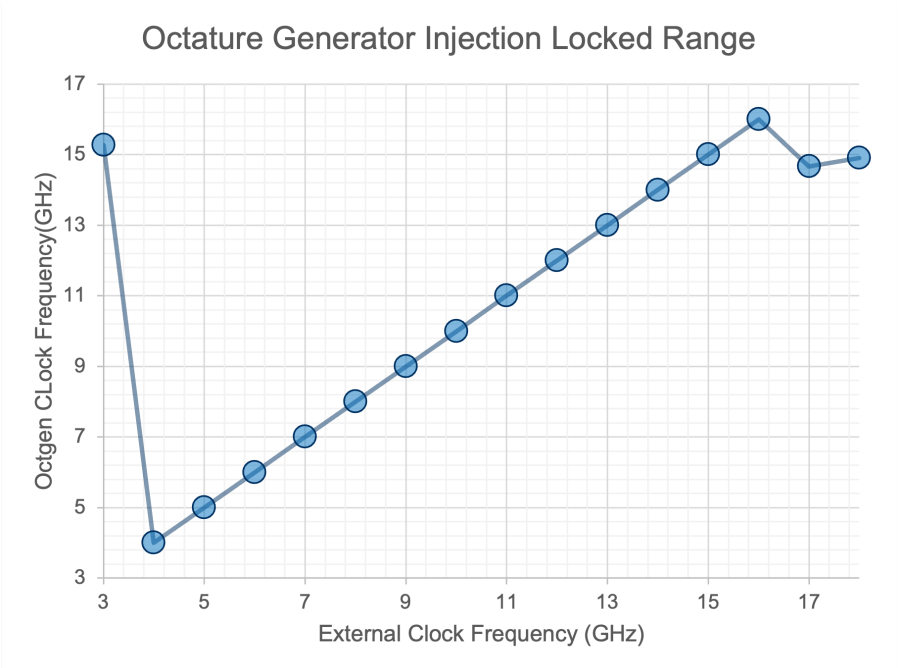
Figure 6.10: VCO measurement results.

injection lock range of the octature generator is shown in Figure 6.11b.It can be locked to the external clock from 4GHz to 16GHz.

Jitter measurement was not performed as there is no high-speed clock (12.5GHz) probed for measurement. We could only obtain the information from the low-speed divided clocks (1.56GHz). This clock is divided from the 12.5GHz clock in datapath and is sent to the digital back end as the digital clock. It goes through three supply domains (CLKVDD/ DATAVDD/ DVDD) and is expected to pick up noise from all supplies and the noisy digital backend. Therefore, the divided clock only provides the frequency information of the clock path, but cannot reflect the jitter of the VCO or octature generator. To fully characterize the VCO and octature generator, we need to probe the clock directly after the octature generator and the VCO. To do this, we need to be careful not to load the high-speed clock path and take good care of the matching of the clock phases.



(a) Free-running frequency of octature generator.



(b) Injection lock range of octature generator.

Figure 6.11: Octature generator measurement.

To test the datapath and the whole system, the test setup has to be done with the probe station to provide the data input through GSGSG probe. For functional testing, M8195A is used to check the half-speed function of the receiver. The whole test setup is shown in Figure 6.12. There are several offset and gain settings that must be set correctly for the datapath to work. To test the sampler, we set the RDAC of the path p and n to the minimum code and the maximum code, respectively, and obtain all 0 and all 1 from the snapshot engine. This ensures that the sampler works properly in extreme cases. However, unfortunately, even if the offset is calibrated and the clock source is synced with the AWG, the output from the snapshot engine does not respond correctly all the time to neither clock pattern nor pulse pattern. The restriction of visibility prevents further debugging. While testing 160 Gbps receiver in [31], we were unable to measure the clock path due to a potential divider failure described in Session 2.3. We were able to measure and verify the clock generation and distribution that we describe and analyze in Chapter 2 through the 100Gbps design. The datapath design proposed in [31] unfortunately was not able to function in testing, and the same issue may be propogated from the 160Gbps design in [31] but not captured because the datapath were unmeasurable at that time.

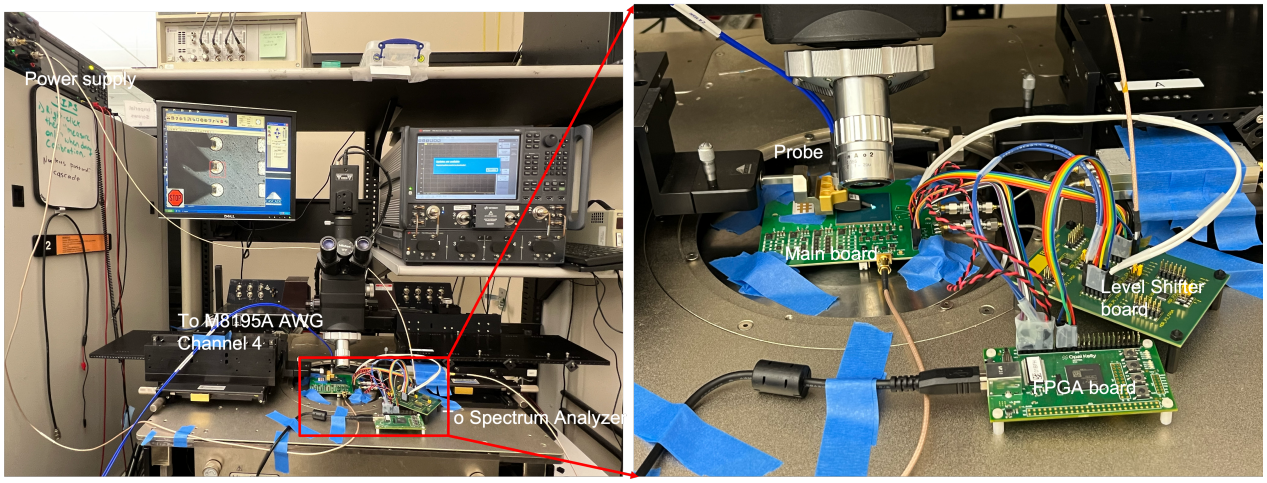


Figure 6.12: Testing setup with probe station.

6.4 Performance

Due to the aforementioned data path testing issue, we would not be able to fully evaluate the performance of the entire receiver. Therefore, the following power and performance are mainly from the simulation results.

The power breakdown of the 100Gbps receiver with hybrid CDR is summarized in Table. 6.1. Power breakdown 160 Gbps RX is listed side by side for comparison. Both the datapath and the clock path are reduced by more than 60% in area and 50% in power. The RDACs are almost 4 times shrinked in size with 3 mW increased. The trade-off of area with power for RDACs is crucial for both speed and area. In fact, with modification of the structures of the RDACs, the total area is reduced by more than 50% with power increasing by less than 1%. The power increase due to RDACs can be easily compensated for by reducing other datapath blocks. Therefore, the overall power is reduced approximately by half.

	160Gbps RX		100 Gbps RX	
Block	Area(mm^2)	Power(mW)	Area(mm^2)	Power(mW)
Datapath (DP)	0.04	115	0.014	55
Clock path (CP)	0.01	200	0.004	80
DP RDACs	0.39	0.6	0.102	3.6
Passive Frontend	0.09	0	0.111	0
Digital Backend (DBE)	0.05	17	0.033	23
VCO	-	-	0.012	9
Total	0.58	332.6	0.275	170.6

Table 6.1: Power breakdown (simulation).

If we take the data rate into consideration, the bandwidth power efficiency is defined in Equation. 6.1 and the results are summarized in Table. 6.2. Overall, the 100Gbps receiver has 20% better bandwidth power efficiency over the 160Gbps link attributed to modification of the receiver stages and layout.

$$\text{Bandwidth Power Efficiency} = \frac{\text{Power}}{\text{Data rate}} \tag{6.1}$$

Although the performance of the proposed 100Gbps receiver with hybrid CDR could not be fully measured due to datapath failure, we were able to measure the VCO operation range and the octature generator free-running and locking range that operate in the desired region.

The post-layout simulation results of 100Gbps RX (this work) vs. 160Gbps RX in [31] are summarized in Table 6.3. Recent publications are listed for reference of the trend. This work attempts to achieve 2x of the Nyquist frequency of other works.

	160Gbps RX	100 Gbps RX
Data rate ($Gbps$)	160	100
Power (mW)	332.6	170.6
Bandwidth Power Efficiency (pJ/bit)	2.08	1.7

Table 6.2: Bandwidth power efficiency (simulation).

Specs	ISSCC 2022 G.Gangasani[4]	ISSCC 2023 S. Park [12]	CICC 2019 M. Erett[9]	160Gbps RX* [31]	This Work*
Technology	5nm	28nm	16nm	16nm	16nm
Data Rate (Gbps)	113	52	56	160	100
Bandwidth (GHz)	28.25	13	28	80	50
Modulation	PAM4	PAM4	NRZ	NRZ	NRZ
RX Equalization	CTLE	CTLE	CTLE	MLSE	MLSE
Channel Loss@ BER	11.5dB @1e-9	7.1dB @1e-12	8dB @1e-15	3dB @1e-12	3dB @1e-12
CDR type	Band-Bang	Baud-rate	Baud-rate	Baud-rate	Baud-rate
RX Energy Eff.(pJ/bit)	-	0.83	-	2.08	1.70
Total Energy Eff.(pJ/bit)	1.55	-	2.25	4.08	-
Area (mm^2/lane)	0.26	0.11**	0.33	0.58**	0.28**
Edge Throughput (Gbps/mm)***	221.6	495.8	97.5	201.1	189.0

*simulated results **RX only *** Assume number of lane $\propto 1/\sqrt{Area}$ and edge length = 1mm

Table 6.3: Performance comparison table.

Chapter 7

Conclusions

7.1 Thesis Summary

The demands for efficient high-speed die-to-die link beyond 100Gbps is growing rapidly with the wide application such as chiplet design. At such a high speed, the channel limited effect is worsening causing ISI that requires equalizers to achieve the target BER. Conventional DFE can equalize post cursors without noise enhancement while the inherent timing constraint of the feedback loop limits the bandwidth. Power is another stringent constraint that makes the design of such an equalizer in high-speed links more challenging. To increase connectivity in constrained dimensions, the link design is required to be as compact as possible. In this work, we focus on die-to-die connection at a data rate of 100+Gbps.

The 1-tap MLSE 100Gbps receiver with hybrid CDR is implemented in 16nm FinFET process with general packaging. The analog circuits are designed and layout with the Berkeley analog generator. The digital backend and the top-level integration is managed through the Hammer flow. The event-driven model and the generation of the test vector are used to speed up the verification of the overall receiver. Although the entire receiver performance could not be evaluated due to the datapath issue, the octature generator and VCO performance is characterized in silicon. The simulated energy efficiency of the 100Gbps receiver is 1.7 pJ/bit.

The key contributions are as follows.

- We proposed two methods, ring-oscillator-based and polyphase-filter-based, to generate octature clocks and techniques to adjust the phase skews linearly with the code. The ring-oscillator-based octature generator coupled two quadrature generators and was implemented at different data rates. The scalability of the free-running frequency with supply and wide injection locked range is demonstrated with the 100Gbps chip taped out in the 16nm FinFET process.

- The hybrid CDR algorithm proposed in the thesis combines the Mueller-Muller algorithm with dLev maximization and is implemented with minimal hardware overhead

for the 1-tap MLSE receiver. The Mueller-Muller-based pattern filtering reuses the hardware complexity from the purely feedforward MLSE in the datapath. The proposed hybrid CDR is shown to be more robust than the state-of-the-art CDR with different example channels in simulation.

- Performance matrices and statistical analysis using multiple-variable Markov chains are proposed to analyze the locking point and jitter performance of CDR algorithms. The results of statistical analysis are verified with ground-truth time-domain simulation.
- The event-driven SystemVerilog behavioral model and test vector generation are developed to verify the mixed-signal receiver system. The analog circuits of the receiver are built with generators and ported from different technology nodes and adopted to different data rates. The generator-based design feature the capability to integrate and verify with digital flow easily.

7.2 Future Works

The hybrid CDR is described with 1-tap MLSE in this work. The usage of the hybrid CDR algorithm can be expanded to cover more taps and work with different equalizers in the future study. Markov-chain-based statistical analysis for the CDR algorithm is a flexible tool for analyzing the performance of a more complicated CDR algorithm. However, the performance is only restricted to the ideal update equation and can be expanded to include nonideality of the actual circuit implementation for more accurate analysis.

Bibliography

[1] A.Payzin. "Analysis of a Digital Bit Synchronizer". In: *IEEE Transactions on Communications* 31.4 (1983), pp. 554–560.

[2] Eric Chang et al. "BAG2: A process-portable framework for generator-based AMS circuit design". In: *Custom Integrated Circuits Conference (CICC)* (2018).

[3] Fulvio Spagna et al. "A 78mW 11.8Gb/s Serial Link Transceiver with Adaptive RX Equalization and Baud-Rate CDR in 32nm CMOS". In: *IEEE International Solid-State Circuits Conference (ISSCC)* (2010), pp. 366–367.

[4] G. Gangasani et al. "A 1.6Tb/s Chiplet over XSR-MCM Channels using 113Gb/s PAM-4 Transceiver with Dynamic Receiver-Driven Adaptation of TX-FFE and Programmable Roaming Taps in 5nm CMOS". In: *2022 IEEE International Solid- State Circuits Conference (ISSCC)* 65 (2022), pp. 122–124.

[5] H. Ju et al. "Design Techniques for 48-Gb/s 2.4-pJ/b PAM-4 Baud-Rate CDR With Stochastic Phase Detector". In: *IEEE Journal of Solid-State Circuits* 57.10 (2022), pp. 3014–3024.

[6] J. Han et al. "Design Techniques for a 60-Gb/s 288-mW NRZ Transceiver With Adaptive Equalization and Baud-Rate Clock and Data Recovery in 65-nm CMOS Technology". In: *IEEE JOURNAL OF SOLID-STATE CIRCUITS* 52.12 (2017), pp. 3474–3485.

[7] J. Kaukovuori et al. "Analysis and Design of Passive Polyphase Filters". In: *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS* 55.10 (2008), pp. 3023–3037.

[8] K.-h. Kim et al. "A 2.6mw 370mhz-to-2.5ghz open-loop quadrature clock generator". In: *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers* (2008), pp. 458–627.

[9] M. Erett et al. "A 2.25pJ/bit Multi-lane Transceiver for Short Reach Intra-package and Inter-package Communication in 16nm FinFET". In: *IEEE Custom Integrated Circuits Conference (CICC)* (2019), pp. 1–8.

[10] Moon-Chul Choi et al. "A 0.1-pJ/b/dB 28-Gb/s Maximum-Eye Tracking, Weight-Adjusting MM CDR and Adaptive DFE with Single Shared Error Sampler". In: *IEEE Symposium on VLSI Circuits* (2020).

[11] Rajeev Dokania et al. "A 5.9pJ/b 10Gb/s serial link with unequalized MM-CDR in 14nm tri-gate CMOS". In: *IEEE International Solid-State Circuits Conference (ISSCC)* (2015), pp. 184–185.

[12] S. Park et al. "A 0.83pJ/b 52Gb/s PAM-4 Baud-Rate CDR with Pattern-Based Phase Detector for Short-Reach Applications". In: *2023 IEEE International Solid- State Circuits Conference (ISSCC)* (2023), pp. 118–120.

[13] S. Park et al. "A 52-Gb/s Low-Power PAM-4 Baud-Rate CDR Using Pattern-Based Phase Detector for Short-Reach Applications". In: *IEEE Journal of Solid-State Circuits* 60.8 (2025), pp. 2794–2806.

[14] S. Roh et al. "A Low-Jitter Phase Detection Technique With Asymmetric Weights in Multi-Level Baud-Rate CDR". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 71.12 (2024), pp. 5861–5872.

[15] AMD. *AMD Launches 5th Gen AMD EPYC CPUs, Maintaining Leadership Performance and Features for the Modern Data Center*. 2024. URL: https://www.amd.com/en/newsroom/press-releases/2024-10-10-amd-launches-5th-gen-amd-epyc-cpus-maintaining-le.html.

[16] Scientific Analog. *XMODEL Introduction*. 2025. URL: https://www.scianalog.com/xmodel/ (visited on 2025).

[17] Apple. *Apple introduces M2 Ultra*. 2023. URL: https://www.apple.com/newsroom/2023/06/apple-introduces-m2-ultra/.

[18] Ayan Biswas. "Design Methodologies and Automated Generation of Ultra High Speed Wireline SerDes Transmitters". In: *Ph.D. dissertation, EECS Dept., Univ. California at Berkeley, Berkeley, CA* (2023).

[19] Broadcom. *"nAUI Channel" data*. 2009. URL: https://www.ieee802.org/3/ba/public/channel.html.

[20] E. Chang and N. Narevsky et al. "BAG: A Process-Portable Framework for Generator-based AMS Circuit Design". In: *IEEE Custom Integrated Circuits Conference (CICC)* (2017).

[21] Wai-Ki Ching and Michael K. Ng. "Multivariate Markov Chains". In: *Markov Chains: Models, Algorithms and Applications*. Boston, MA: Springer US, 2006, pp. 141–169. ISBN: 978-0-387-29337-0. DOI: 10.1007/0-387-29337-X_7. URL: https://doi.org/10.1007/0-387-29337-X_7.

[22] INTERNATIONAL ROADMAP FOR DEVICES and SYSTEMS. *OUTSIDE SYSTEM CONNECTIVITY*. 2024. URL: https://irds.ieee.org/images/files/pdf/2024/2024IRDS_OSC.pdf.

[23] A. Emami-Neyestanak et al. "CMOS transceiver with baud rate clock recovery for optical interconnects". In: *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)* (2004), pp. 410–413.

[24] Azita Emami-Neyestanak. "Design of CMOS Receivers for Parallel Optical Interconnects". In: *Ph.D. dissertation, EE Dept., Stanford University* (2004).

[25] Roberto Piacentini Filho. *What is a Chiplet, and Why Should You Care?* 2024. URL: https://www.keysight.com/blogs/en/tech/sim-des/2024/2/8/what-is-a-chiplet-and-why-should-you-care.

[26] Jaeduk Han. "Design and Automatic Generation of 60Gb/s Wireline Transceivers". In: *Ph.D. dissertation, EECS Dept., Univ. California at Berkeley, Berkeley, CA* (2019).

[27] J.K. Holmes. "Performance of a First-Order Transition Sampling Digital Phase-Locked Loop Using Random-Walk Models". In: *IEEE Transactions on Communications* 20.2 (1972), pp. 119–131.

[28] J. -H. Yoon J. -Y. Lee and H. -M. Bae. "A 10-Gb/s CDR With an Adaptive Optimum Loop-Bandwidth Calibrator for Serial Communication Links". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 61.8 (2014), pp. 2466–2472.

[29] A. Puggelli et al. J. Crossley. "Bag: A designer-oriented integrated framework for the development of ams circuit generators". In: *IEEE/ACM International Conference on Computer-Aided De- sign (ICCAD)* (2013), pp. 74–81.

[30] S. Kundu et al. J. Kim. "A 224Gb/s DAC-Based PAM-4 Transmitter with 8-Tap FFE in 10nm CMOS". In: *IEEE International Solid-State Circuits Conference (ISSCC)* (2021), pp. 126–127.

[31] Paul Kwon. "Feedforward MLSE Equalization for High Speed Serial Links". In: *Ph.D. dissertation, EECS Dept., Univ. California at Berkeley, Berkeley, CA* (2023).

[32] Z. Wang et al. M. Choi. "An Output-Bandwidth-Optimized 200Gb/s PAM-4 100Gb/s NRZ Transmitter with 5-Tap FFE in 28nm CMOS". In: *2021 IEEE International Solid-State Circuits Conference (ISSCC)* (2021), pp. 128–129.

[33] Stephanie Michel. *With PCIe® 6.0 You Have to Move from NRZ to PAM4... But What is PAM4 signaling?* 2023. URL: https://www.keysight.com/blogs/en/inds/2023/05/11/why-pcie-6-moves-from-nrz-to-pam4.

[34] Shahriar Mirabbasi and Laura C Fujino. "Through the Looking Glass—The 2025 Edition: Trends in solid-state circuits from ISSCC". In: *IEEE Solid-State Circuits Magazine* 17.1 (2025), pp. 97–118.

[35] Kurth Mueller and Markus Muller. "Timing Recovery in Digital Synchronous Data Receivers". In: *IEEE Trans. Communications* (1976), pp. 516–531.

[36] P. Onufryk and S. Choudhary. "UCIe: Standard for an Open Chiplet Ecosystem". In: *IEEE Micro* (2025), pp. 16–25.

[37] M. -J. Park and J. Kim. "Pseudo-Linear Analysis of Bang-Bang Controlled Timing Circuits". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 60.6 (2013), pp. 1381–1394.

[38] Behzad Razavi. "The Design of a Millimeter-Wave Frequency Divider". In: *IEEE Solid-state circuits magazine* (Fall 2022).

[39] A. Rezayee and K. Martin. "A coupled two-stage ring oscillator". In: *Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems. MWSCAS 2001 (Cat. No.01CH37257)* 2 (2001), pp. 878–881.

[40] J. C. Rudell. "Frequency translation techniques for high-integra-tion high-selectivity multi-standard wireless communication systems". In: *Ph.D. dissertation, EECS Dept., Univ. California at Berkeley, Berkeley, CA* 3.2 (2000), pp. 207–230.

[41] F. Spagna. "Clock and data recovery systems". In: *2018 IEEE Custom Integrated Circuits Conference (CICC)* (2018), pp. 1–120.

[42] Vladimir Stojanović. "Channel-Limited High-Speed Links: Modeling, Analysis and Design". In: *Ph.D. dissertation, EE Dept., Stanford University* (2004).

[43] Vladimir Stojanović and M. Horowitz. "Modeling and analysis of high-speed links". In: *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference* (2003), pp. 589–594.

[44] Synopsys. *UCIe Universal Chiplet Interconnect Express*. 2025. URL: `https://www.uciexpress.org` (visited on 2025).

[45] Synopsys. *What is UCIe*. 2025. URL: `https://www.synopsys.com/glossary/what-is-ucie.html` (visited on 2025).

[46] King L. Tai. "System-In-Package(SIP): Challenges and Opportunities". In: *Proceedings 2000. Design Automation Conference.* (2000), pp. 191–196.

[47] Zhongkai Wang. "Analog Generators for SerDes Clock Generation and Distribution". In: *Ph.D. dissertation, EECS Dept., Univ. California at Berkeley, Berkeley, CA* (2021).

[48] Z. Wang Y. Zhang and P. R. Kinget. "Analysis of Injection-Locked Ring Oscillators for Quadrature Clock Generation in Wireline or Optical Transceivers". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 69.8 (2022), pp. 3074–3082.

[49] H. Yamamoto and S. Mori. "Performance of a Binary QuantizedAll Digital Phase-Locked Loop with a New Class of Sequential Filter". In: *IEEE Transactions on Communications* 26.1 (1978), pp. 35–45.

[50] M. Choi et al. Z. Wang. "An Automated and Process-Portable Generator for Phase-Locked Loop". In: *2021 58th ACM/IEEE Design Automation Conference (DAC)* (2021), pp. 511–516.

[51] Y. Onizuka Z. Wang Y. Zhang and P. R. Kinget. "Multi-Phase Clock Generation for Phase Interpolation With a Multi-Phase, Injection-Locked Ring Oscillator and a Quadrature DLL". In: *EEE Journal of Solid-State Circuits* 57.6 (2022), pp. 1776–1787.

Appendix A

Single Variable Markov Chain Analysis for Hybrid Algorithm

A.1 Transition Probability Derivation

From Table 4.2, we can rewrite Equation 4.10 as

$$\begin{aligned}
P_{up,i} &= P_{i,i+1} \\
&= P(c>0)[P_{i,i} + \frac{\pi_{i-1}P_{i-1,i}}{\pi_i}P(\Delta dLev(\phi_i)>0) + \frac{\pi_{i+1}P_{i+1,i}}{\pi_i}P(\Delta dLev(\phi_{i+1})>0)] \\
&= \frac{P_{c,up}(\phi_i)}{\pi_i}[\pi_i P_{i,i} + \pi_{i-1}P_{i-1,i}P_{dLev,up}(\phi_i) + \pi_{i+1}P_{i+1,i}P_{dLev,up}(\phi_{i+1})]
\end{aligned} \tag{A.1}$$

By the same logic, $P_{i,i-1}$ and $P_{i,i}$ can be derived as below.

$$\begin{aligned}
P_{dn,i} &= P_{i,i-1} \\
&= P(PD_{out}<0) \\
&= P(c<0, PD_{out,d}e[n]\leq 0) \\
&= P(c<0)P(PD_{out,d}e[n]\leq 0) \\
&= P(c<0)(P(PD_{out,d}e[n]=0) + P(PD_{out,d}e[n]<0)) \\
&= P(c<0)(P(PD_{out,d}=0) + (\sum_{x=\pm 1}P(PD_{out,d}=x))P(xe[n]<0)) \\
&= P(c<0)[P_{i,i} + \frac{\pi_{i-1}P_{i-1,i}}{\pi_i}P(\Delta dLev(\phi_i)<0) + \frac{\pi_{i+1}P_{i+1,i}}{\pi_i}P(\Delta dLev(\phi_{i+1})<0)] \\
&= \frac{P_{c,dn}(\phi_i)}{\pi_i}[\pi_i P_{i,i} + \pi_{i-1}P_{i-1,i}P_{dLev,dn}(\phi_i) + \pi_{i+1}P_{i+1,i}P_{dLev,dn}(\phi_{i+1})]
\end{aligned} \tag{A.2}$$

$$
\begin{aligned}
P_{hold,i} &= P_{i,i} \\
&= P(PD_{out} = 0) \\
&= P(c > 0, PD_{out,d}e[n] < 0) + P(c < 0, PD_{out,d}e[n] > 0) \\
&= P(c > 0)P(PD_{out,d}e[n] < 0) + P(c < 0)P(PD_{out,d}e[n] > 0) \\
&= P(c > 0)(\sum_{x=\pm 1} P(PD_{out,d} = x))P(xe[n] < 0)) \\
&+ P(c < 0)(\sum_{x=\pm 1} P(PD_{out,d} = x))P(xe[n] > 0)) \\
&= P(c > 0)[\frac{\pi_{i-1}P_{i-1,i}}{\pi_i}P(\Delta dLev(\phi_i) < 0) + \frac{\pi_{i+1}P_{i+1,i}}{\pi_i}P(\Delta dLev(\phi_{i+1}) < 0)] \\
&+ P(c < 0)[\frac{\pi_{i-1}P_{i-1,i}}{\pi_i}P(\Delta dLev(\phi_i) > 0) + \frac{\pi_{i+1}P_{i+1,i}}{\pi_i}P(\Delta dLev(\phi_{i+1}) > 0)] \\
&= \frac{P_{c,up}(\phi_i)}{\pi_i}[\pi_i P_{i,i} + \pi_{i-1}P_{i-1,i}P_{dLev,dn}(\phi_i) + \pi_{i+1}P_{i+1,i}P_{dLev,dn}(\phi_{i+1})] \\
&+ \frac{P_{c,dn}(\phi_i)}{\pi_i}[\pi_i P_{i,i} + \pi_{i-1}P_{i-1,i}P_{dLev,up}(\phi_i) + \pi_{i+1}P_{i+1,i}P_{dLev,up}(\phi_{i+1})]
\end{aligned}
\tag{A.3}
$$

A.2 3-state Markov Chain Transition Matrix

To illustrate the steps, let us consider a case of K = 3, where the Markov chain only has three states shown in Figure A.1.

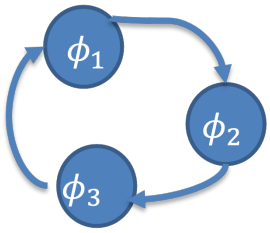


Figure A.1: 3-state Markov chain.

To decide on the matrix form, we need to write the equations for Equation 4.15, Equation 4.16, and Equation 4.17 as follows.

$$
\begin{aligned}
\pi_1 P_{1,3} &= P_{c,dn}(\phi_1)[\pi_1 P_{1,1} + \pi_3 P_{3,1} P_{dLev,dn}(\phi_1) + \pi_2 P_{2,1} P_{dLev,dn}(\phi_2)] \\
\pi_1 P_{1,1} &= P_{c,up}(\phi_1)[\pi_1 P_{1,1} + \pi_3 P_{3,1} P_{dLev,dn}(\phi_1) + \pi_2 P_{2,1} P_{dLev,dn}(\phi_2)] \\
&\quad + P_{c,dn}(\phi_1)[\pi_1 P_{1,1} + \pi_3 P_{3,1} P_{dLev,up}(\phi_1) + \pi_2 P_{2,1} P_{dLev,up}(\phi_2)] \\
\pi_1 P_{1,2} &= P_{c,up}(\phi_1)[\pi_1 P_{1,1} + \pi_3 P_{3,1} P_{dLev,up}(\phi_1) + \pi_2 P_{2,1} P_{dLev,up}(\phi_2)] \\
\pi_2 P_{2,1} &= P_{c,dn}(\phi_1)[\pi_2 P_{2,2} + \pi_1 P_{1,2} P_{dLev,dn}(\phi_2) + \pi_3 P_{3,2} P_{dLev,dn}(\phi_3)] \\
\pi_2 P_{2,2} &= P_{c,up}(\phi_2)[\pi_2 P_{2,2} + \pi_1 P_{1,2} P_{dLev,dn}(\phi_2) + \pi_3 P_{3,2} P_{dLev,dn}(\phi_3)] \\
&\quad + P_{c,dn}(\phi_2)[\pi_2 P_{2,2} + \pi_1 P_{1,2} P_{dLev,up}(\phi_2) + \pi_3 P_{3,2} P_{dLev,up}(\phi_3)] \\
\pi_2 P_{2,3} &= P_{c,up}(\phi_2)[\pi_2 P_{2,2} + \pi_1 P_{1,2} P_{dLev,up}(\phi_2) + \pi_3 P_{3,2} P_{dLev,up}(\phi_3)] \\
\pi_3 P_{3,2} &= P_{c,dn}(\phi_3)[\pi_3 P_{3,3} + \pi_2 P_{2,3} P_{dLev,dn}(\phi_3) + \pi_1 P_{1,3} P_{dLev,dn}(\phi_1)] \\
\pi_3 P_{3,3} &= P_{c,up}(\phi_3)[\pi_3 P_{3,3} + \pi_2 P_{2,3} P_{dLev,dn}(\phi_3) + \pi_1 P_{1,3} P_{dLev,dn}(\phi_1)] \\
&\quad + P_{c,dn}(\phi_3)[\pi_3 P_{3,3} + \pi_2 P_{2,3} P_{dLev,up}(\phi_3) + \pi_1 P_{1,3} P_{dLev,up}(\phi_1)] \\
\pi_3 P_{3,1} &= P_{c,up}(\phi_3)[\pi_3 P_{3,3} + \pi_2 P_{2,3} P_{dLev,up}(\phi_3) + \pi_1 P_{1,3} P_{dLev,up}(\phi_1)]
\end{aligned}
\tag{A.4}
$$

Rearrange Equation A.4 in homogeneous form and then we can rewrite the equations in matrix form as follows.

$$A\vec{x} = \vec{0} \tag{A.5}$$

where

(A.6)

$$
A=\begin{bmatrix}
0 & P_{c,u}(\phi_1) & -1 & P_{c,u}(\phi_1)P_{L,u}(\phi_1) & 0 & 0 & 0 & P_{c,u}(\phi_1)P_{L,u}(\phi_3)\\
-1 & P_{c,d}(\phi_1) & 0 & P_{c,d}(\phi_1)P_{L,d}(\phi_1) & 0 & 0 & 0 & P_{c,d}(\phi_1)P_{L,d}(\phi_3)\\
0 & -1 & 0 & 0 & 0 & 0 & 0 & P_{c,u}(\phi_1)P_{L,d}(\phi_3)+P_{c,d}(\phi_1)P_{L,u}(\phi_3)\\
0 & 0 & P_{c,u}(\phi_2)P_{L,u}(\phi_1) & P_{c,u}(\phi_1)P_{L,d}(\phi_1)+P_{c,d}(\phi_1)P_{L,u}(\phi_1) & 0 & 0 & 0 & 0\\
0 & 0 & P_{c,d}(\phi_2)P_{L,d}(\phi_1) & 0 & P_{c,u}(\phi_2) & 0 & 0 & 0\\
0 & 0 & P_{c,d}(\phi_2)P_{L,u}(\phi_1)+P_{c,u}(\phi_2)P_{L,d}(\phi_1) & 0 & P_{c,d}(\phi_2) & -1 & P_{c,u}(\phi_2)P_{L,u}(\phi_2) & 0\\
P_{c,u}(\phi_3)P_{L,u}(\phi_3) & 0 & 0 & 0 & 0 & 0 & P_{c,d}(\phi_2)P_{L,d}(\phi_2) & 0\\
P_{c,d}(\phi_3)P_{L,d}(\phi_3) & 0 & 0 & 0 & 0 & P_{c,u}(\phi_3)P_{L,u}(\phi_2) & P_{c,u}(\phi_2)P_{L,d}(\phi_2)+P_{c,d}(\phi_2)P_{L,u}(\phi_2) & -1\\
P_{c,d}(\phi_3)P_{L,u}(\phi_3)+P_{c,u}(\phi_3)P_{L,d}(\phi_3) & 0 & 0 & 0 & P_{c,d}(\phi_3)P_{L,u}(\phi_2)+P_{c,u}(\phi_3)P_{L,d}(\phi_2) & P_{c,d}(\phi_3) & -1 & 0
\end{bmatrix}
$$

$$\vec{x} = \begin{bmatrix} \pi_1 P_{1,3} & \pi_1 P_{1,1} & \pi_1 P_{1,2} & \pi_2 P_{2,1} & \pi_2 P_{2,2} & \pi_2 P_{2,3} & \pi_3 P_{3,2} & \pi_3 P_{3,3} & \pi_3 P_{3,1} \end{bmatrix}^T \quad \text{(A.7)}$$

$$\vec{0} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad \text{(A.8)}$$

Appendix B

Driver Model and Channel Fitting

1. Overall Modeling Method

   The overall CML driver schematic is shown in Figure B.1, real-type input/output is denoted in red. the additional block, filter_cal, is used to generate the time-domain output waveform combining the driver pole with the channel response corresponding to the ideal square wave outp_m and outn_m.

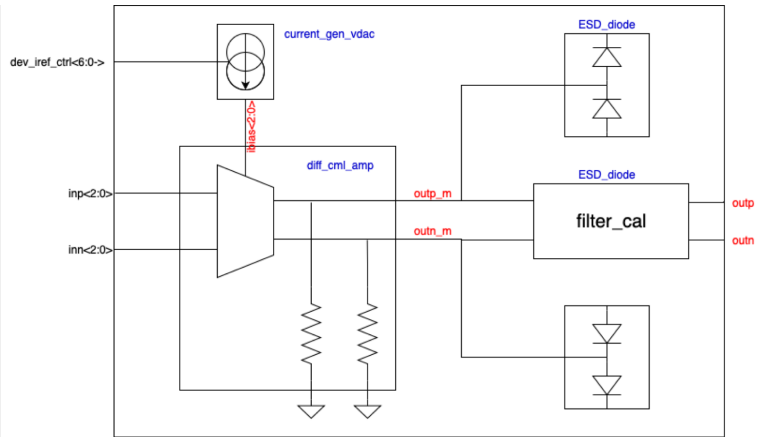


Figure B.1: CML driver schematics.

The overall steps to derive time-domain output waveform form input, driver poles, and channel response are shown in Figure B.2. The square-wave input waveform could be decomposed into superposition of step function with different delay. Each input step experiences driver poles, channel response and received by receiver as output waveform.

a) Channel Fitting In order to implement true-event driven model, the conversion from input to output is done in s-domain. Thus, the channel response should also

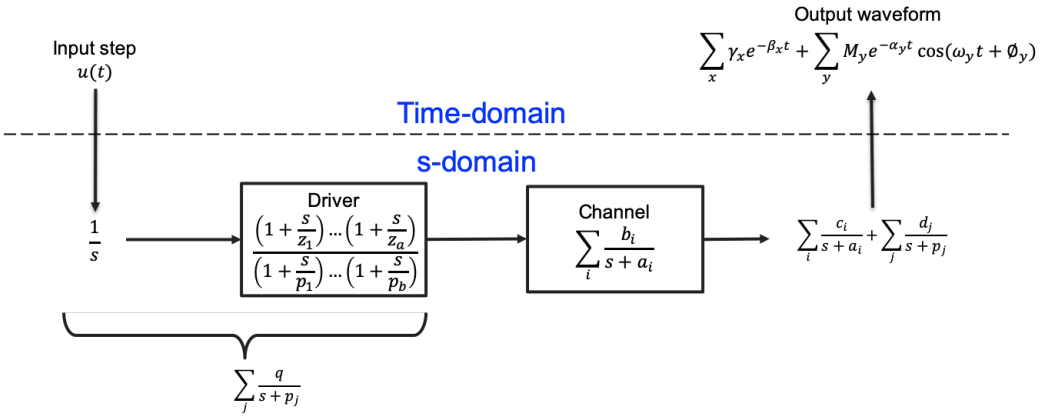
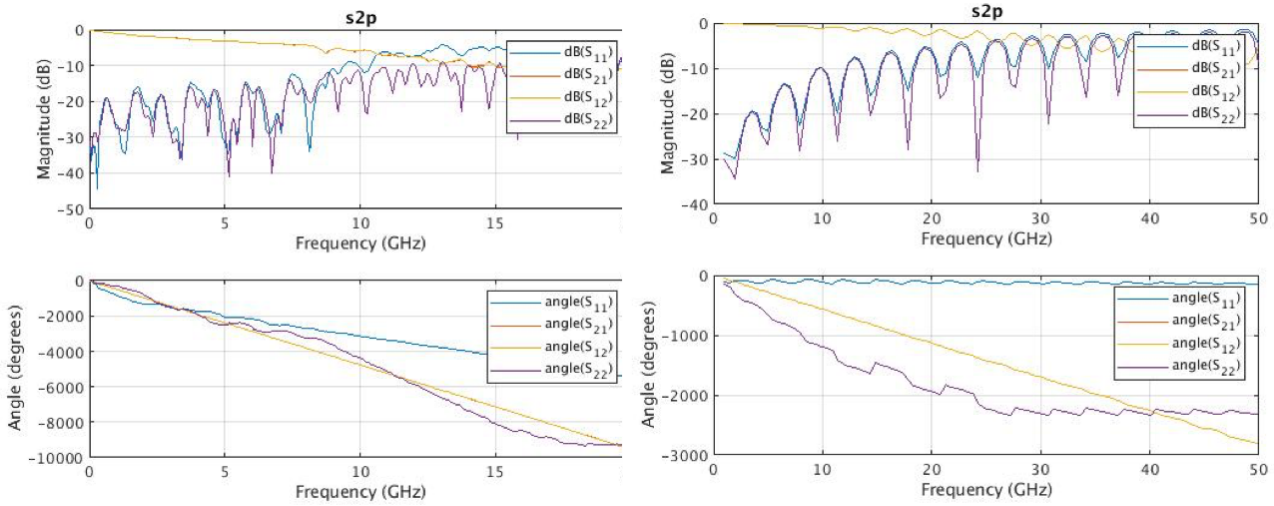


Figure B.2: Overall modeling steps.

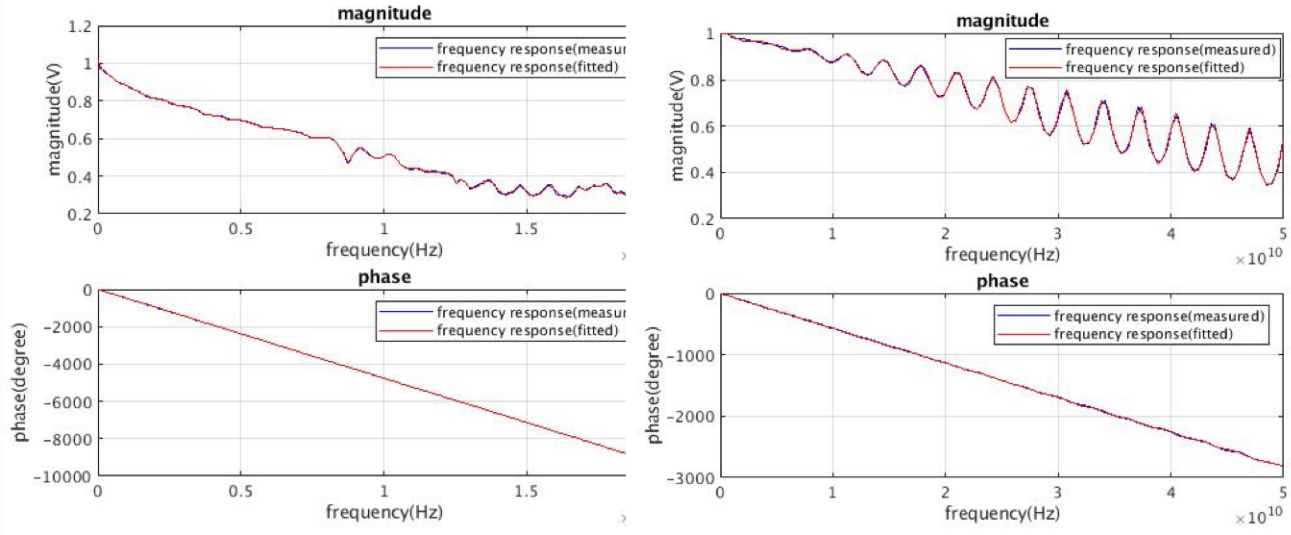
be fitted in s-domain. First, the measured channel data in s4p format is converted to differential s-parameter in s2p format. Assuming properly terminated at both Tx and Rx side, the source and load resistance of the channel are set to be Z_0 of the channel for output voltage to input voltage transfer function derivation. The nAUI channel S-parameter data from IEEE802 website[**IEEE802**] and simulated channel from ADS LindCalc are shown in Figure B.3.

Channel fitting is performed using *rationalfit* in MATLAB RF toolbox. The fitting format is linear combination of rational first-order expression. The first-order expression avoids overflow problem when fitting high order function as in *tfest* function which fits in the format of polynomial numerator divided by polynomial denominator. The fitting results of the transfer function shown in red line in Figure B.4. The nAUI channel is fitted with 65 poles and simulated channel is fitted with 31 poles.



(a) nAUI channel. (b) simulated channel.

Figure B.3: s-parmeter of channels.



(a) nAUI channel. (b) simulated channel.

Figure B.4: Transfer functions derived from s-parameters.

b) S-Domain Calculation

In order to easily apply Inverse Laplace Transform to obtain time-domain output waveform, the output in s-domain is required to be converted in the format of

linear combination of first order or second order system. As mentioned in the previous section, expanding channel fitted results in common denominator does not work because the zero-order coefficient would be too large to express (maximum number in MATLAB is approximately 10^{301}). Therefore, we could not do partial fractional expansion on the overall output after multiplication and reduction to common denominator. The way to avoid this problem is to group the input step and driver function and do partial fractional expansion to get its basis function expression and then use following equation to calculate overall basis function expression after combining with channel response which is already expressed in basis function expression.

$$\begin{aligned} V_{out}(s) &= V_{in}(s) \times H_{driver}(s) \times H_{channel}(s) \\ &= \frac{1}{s} \times \frac{(1+\frac{s}{z_1})...(1+\frac{s}{z_a})}{(1+\frac{s}{p_1})...(1+\frac{s}{p_b})} \times \sum_i \frac{b_i}{s+a_i} \\ &= \sum_j \frac{q_j}{s+p_j} \times \sum_i \frac{b_i}{s+a_i} \end{aligned} \tag{B.1}$$

To handle the multiplication of two basis function expression, we could use the simplified version of equation proposed in [**true_event**] below.

$$\sum_i \frac{b_i}{s+a_i} \times \sum_j \frac{q_j}{s+p_j} = \sum_i \frac{c_i}{s+a_i} + \sum_j \frac{d_j}{s+p_j} \tag{B.2}$$

where

$$c_i = \sum_{x=1}^{j} \frac{b_i q_x}{p_x - a_i}$$

$$d_j = \sum_{x=1}^{i} \frac{b_x q_j}{a_x - p_j}$$

c) Inverse Laplace Transform

Since the s-domain output is expressed in basis function, we could map the individual term into the coefficient of the exponential function or sinusoidal function for real pole and complex pole respectively. The SystemVerilog Direct Programming Interface (DPI) allows us to import functions from C language. The exponential and sinusoidal function could be imported from DPI and used to express the output waveform in time domain.

2. Evaluations of Fitted Channel Response

   To evaluate how well the fitted s-domain expression fits the transfer function derived from the measured s-parameter data, we compared the residual ISI distribution and time-domain step response between the fitted function and the original function derived from measured data. Testbench in Cadence shown in Figure B.5 is also setup for verification in time domain.

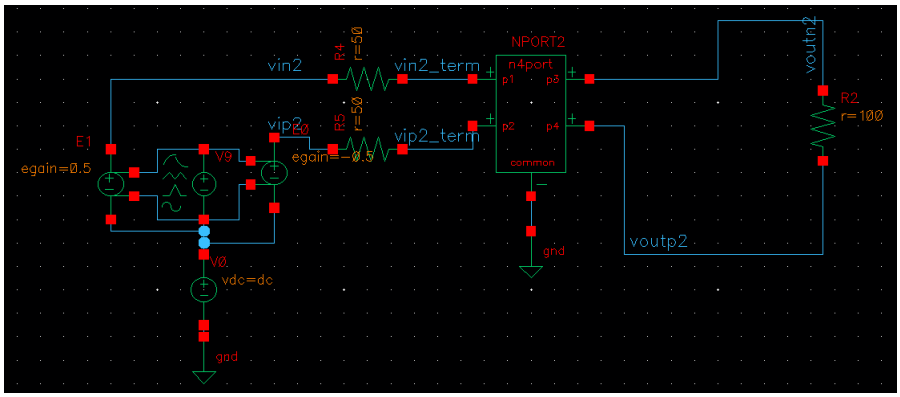
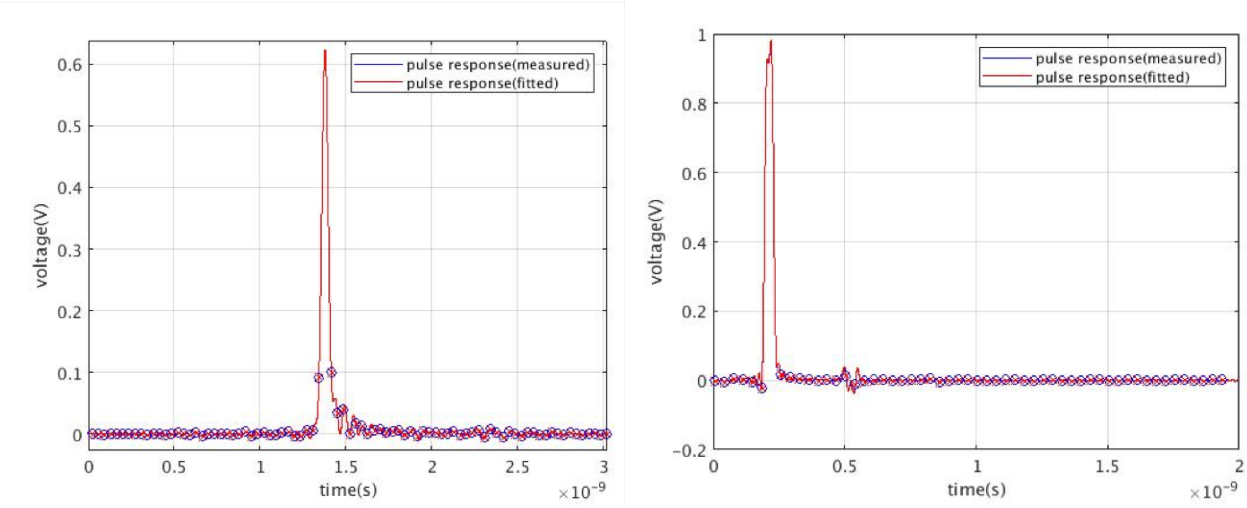


Figure B.5: Testbench for pulse and step response simulation in Cadence.

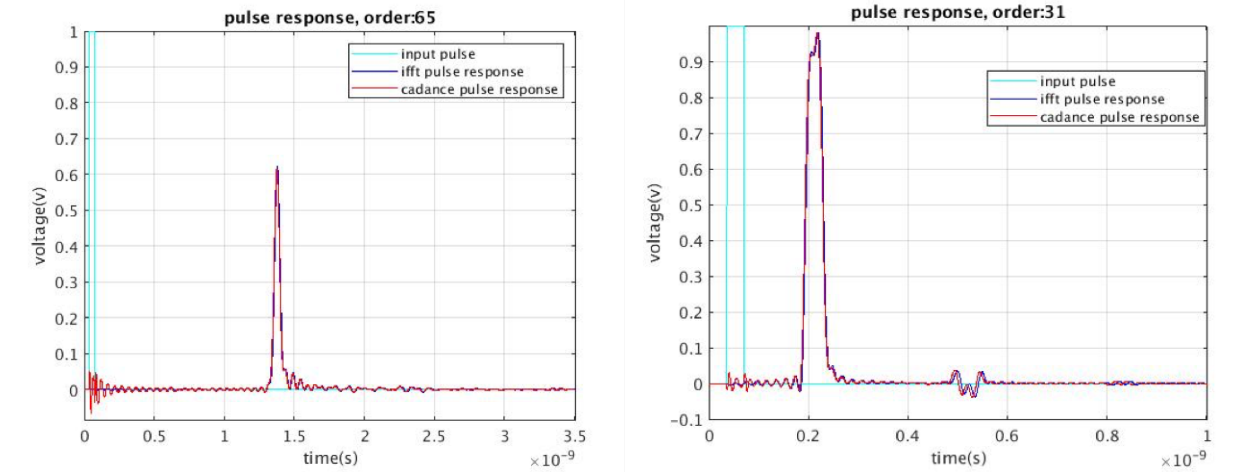
a) Residual ISI Distribution

   The residual ISI distribution is obtained from the pulse response (Figure B.6) at 28GHz, assuming the cursor is sampled at the peak. To obtained the pulse response from the transfer function using ifft function, the transfer function derived from s-parameter need to be extrapolated and interpolated for frequency response at DC and equally spaced frequency. The extrapolation and interpolation is done in polar coordinate, i.e. magnitude and unwrapped phase respectively. Zero-padded between the positive frequency and negative frequency is used for increasing resolution in time-domain. Figure B.7 shows the comparison between pulse response obtained by convolution of ifft impulse response and input pulse and pulse response from cadence simulation. The corresponding ISI distribution by convolution of 20 largest pre-cursors/ post-cursors are shown in Figure B.8, the error is calculated from the root mean square of the difference between measured residual ISI and fitted residual ISI.



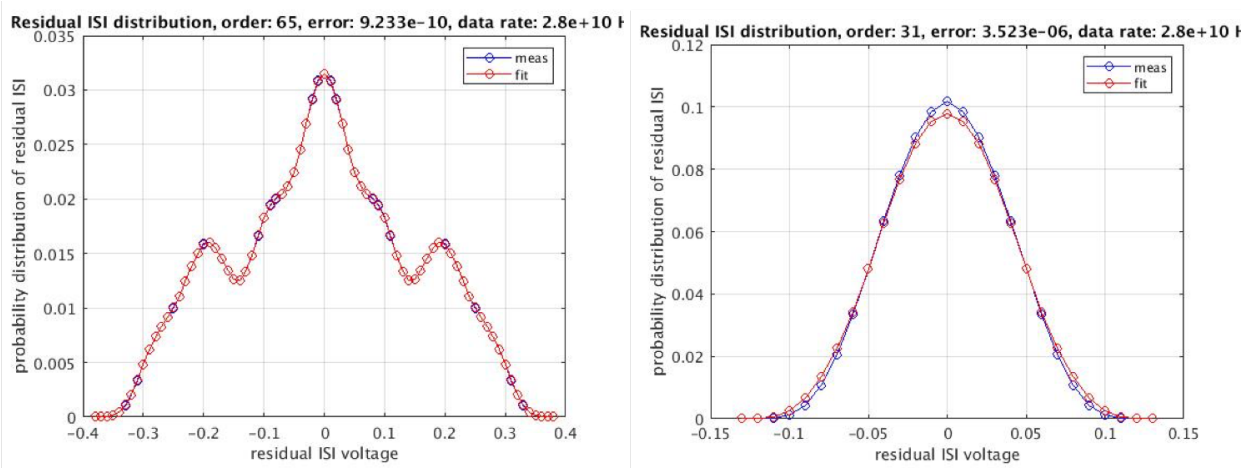
(a) nAUI channel. (b) Simulated channel.

Figure B.6: Pulse response: data rate = 28GHz.



(a) nAUI channel. (b) Simulated channel.

Figure B.7: Comparison of pulse response.

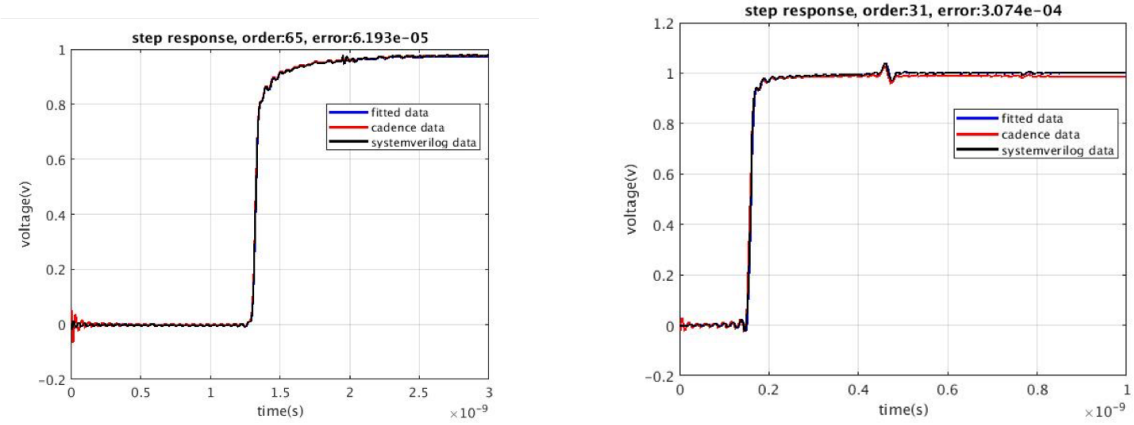


(a) nAUI channel fitted with 65 poles. (b) Simulated channel fitted with 31 poles.

Figure B.8: Residual ISI distribution.

b) Time-Domain Step Response

Time-domain step response obtained from the time-domain calculation in MATLAB (fitted data) , cadence simulation (cadence data), and the fitted time-domain expression in SystemVerilog behavioral model (SystemVerilog data) are compared in Figure B.9. The fitted data in MATLAB overlaps with SystemVerilog data as expected since both are calculated with same time-domain expression, indicating that the function in behavioral model is correct. The error is the root mean square of the fitted data and cadence data.



(a) nAUI channel.

(b) Simulated channel.

Figure B.9: Step response.

3. Behavioral Model Simulation

Based on the time-domain settling coefficient obtained from the above section, the time domain output signal could be calculated by the superposition of multiple step response at different time delay.

$$V_{in}(t) = V_{int0} + \Delta V_0 u(t - t_0) + \Delta V_1 u(t - t_1) + ... + \Delta V_n u(t - t_n) \tag{B.3}$$

where

$$\Delta V_n = V_{in}(t_n^+) - V_{in}(t_n^-) \tag{B.4}$$

Assuming linear time-invariant system, the output could be obtained from the following equation, where $f(t)$ denotes step response.

$$V_{out}(t) = V_{int0} + \Delta V_0 f(t - t_0) + \Delta V_1 f(t - t_1) + ... + \Delta V_n f(t - t_n) \tag{B.5}$$

In practice, when $t - t_n$ is large enough, $f(t - t_n) \approx 1$, therefore, we could lump that term into initial value V_{int}. The number of the superposition waveform should be

chosen based on the settling and the delay of the step response. Figure B.11 shows the case superposing 5 step response, where the 64-bit registers stores and updates the delay time, voltage difference, and initial voltage expressed in real number when the input changes.

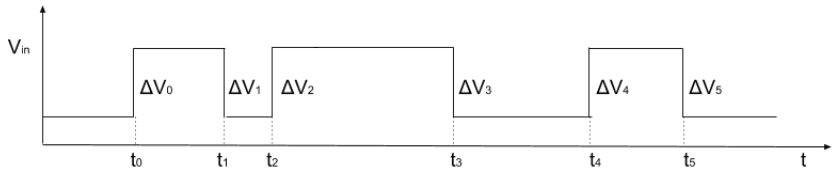


Figure B.10: Timing diagram of input signal.

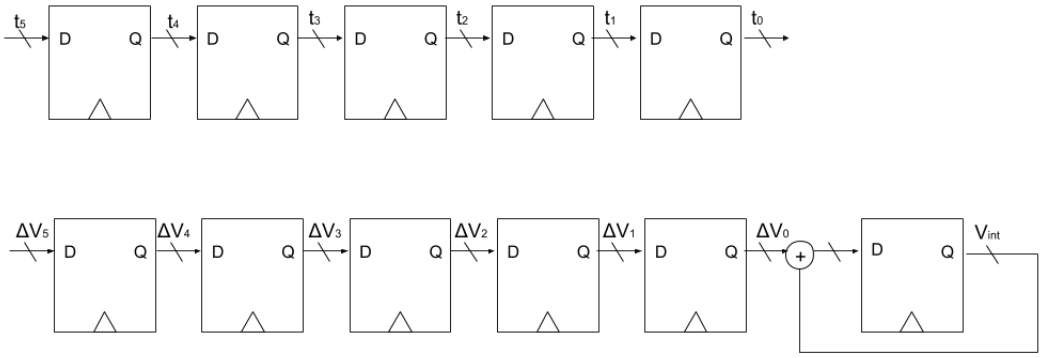
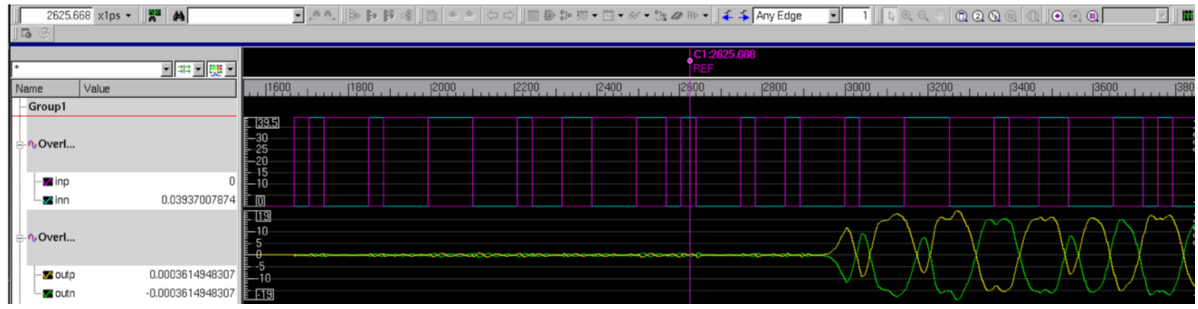
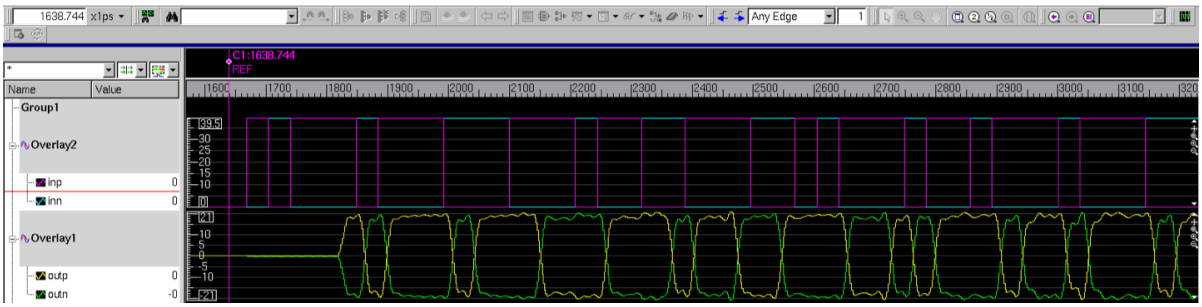


Figure B.11: Block diagram.

The simulation results of the entire Tx datapath model is shown in Figure B.12, where inp and inn indicate the input of the driver whereas outp and outn represent the output of the driver including channel response. The settling function combines one driver pole and the channel response. The overall settling is dominated by the channel response. For driver pole at lower frequency, the settling would be dominated by the driver pole. The time scale of the plot is 1 ps and data rate is 28 GHz. Oversampling is only for checking the pulse shape but not related to the accuracy. The number of superposition waveform are 7 and 36 for simulated channel and nAUI channel respectively. The delay of nAUI channel and simulated channel are 1.4 ns and 0.16 ns respecively, which match the delay in pulse response and step response. The shape and the peak value of the output waveform conform to the pulse response as well.



(a) nAUI channel.



(b) Simulated channel.

Figure B.12: Tx datapath simulation.