Abstract

An integrated circuit designed for high-speed data serialization and transmission in the CML signal domain is presented. The chosen fabrication process is TSMC's 65nm CMOS technology node, selected per CMC guidelines for its balance of performance and scalability. Control logic in the pre-serializer stages is implemented alongside the corresponding clocking subdomain circuitry, both utilizing standard (foundry-defined) transistors and resistor models. The high-speed stages generate the FIR taps for equalization by delaying the data path by a unit interval for the subsequent stages. A 32-bit parallel data stream at user-defined rates of 250 or 500 Mbps can be transmitted at 8 or 16 Gbps—at 8 Gbps through a channel modeled by Peters (Intel), and at 16 Gbps using Matoglu's (Amphenol) model. Both models are included in the IEEE 802.3 Ethernet Working Group family of electrical link channels. Significant insertion loss at these data rates indicated the need for equalization, which is performed with two- and three-tap FFE on target output. Functional testing is performed using Cadence Design Tools, which also serves as the primary environment for performance analysis. Granular test results for both subcomponent and integration testing are included in this report.

Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS

AFE	Analog Front End
CLK	Clock (signal)
CMC	Canadian Microelectronics Corporation
CML	Current Mode Logic
CMOS	Complementary Metal-Oxide-Semiconductor
COEN	Computer Engineering Department, Concordia University
DAC	Digital-to-AnalogConverter
dB	Decibels
DCC	Duty Cycle Correction
DDC	Digital Down Converter
DJ	Deterministic Jitter
EDA	Electronic Design Automation
ELSEE	Ethical, Legal, Societal, Environmental and Economical
EQ	Equalizer
ESD	Electrostatic Discharge
FCC	Federal Communications Commission
FF	Fast nMOS, fast pMOS Process Corner
FFE	Feed-Forward Equalizer
FS	Fast nMOS, slow pMOS Process Corner
FIR	Finite Impulse Response
Gbps	Gigabits per second
HDL	Hardware Description Language
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual Property
ISI	Inter-Symbol Interference
LIDO	Logic-in, differential-out
MUX	Multiplexer
nMOS	n-channel Metal-Oxide Semiconductor
NRZ	Non-Return to Zero
PAM4	4-level Pulse Amplitude Modulation
PCB	Printed Circuit Board
pMOS	p-channel Metal-Oxide Semiconductor
POR	Plan of Record
PLL	Phase Lock Loop
PRBS	Pseudorandom Binary Sequence
PVT	Process, Voltage, and Temperature
RX	Receiver
SERDES	Serializer/Deserializer
SF	Slow nMOS, fast pMOS Process Corner
SS	Slow nMOS, slow pMOS Process Corner
TSMC	Taiwan Semiconductor Manufacturing Company
TT	Typical nMOS, typical pMOS Process Corner
TX	Transmitter
UI	Unit Interval
USD	United States Dollar
VML	VoltageModeLogic
Vppd	Peak-to-PeakDifferentialVoltage
VNA	Vector Network Analyzer

1 INTRODUCTION

1.1 Project Framework

The predictions of Moore's Law are well-known, though the implications at the time it was formalized decades ago are far different than those faced today. As transistor technology nodes push feature lengths deeper into the sub-10 nm range, physical and chemical laws become increasingly critical factors when implementing real devices. Net thermal processes cause many common issues at this scale, for example, as well as limits on device switching rates. These limits not only constrain the rates achievable for any singular local signals but also affect the performance of transmitting and receiving devices used for sending such signals over a variety of distances.

For these reasons, modern analog IC design tends to grapple more with the challenges inherent in moving large amounts of data – at least more so than trying to increase the actual number of transistors on a chip. A variety of industrial and consumer services like 5G technology, generative AI engines, cryptocurrency, and high-throughput streaming services are hotly anticipated if not widely available. Providing such services reliably requires staggering rates of data transfer, underscoring the need to efficiently and reliably create, package, and send signals at increasing speeds - whether the distance is a few microns between components on a board or across the proverbial oceans.

1.2 Objectives

Within this framework, the group has sought to develop a high-speed analog wireline transmitter capable of serializing several instances of local data generated using more easily attainable clock frequencies, before sending the resultant stream at the corresponding higher overall data rate. Due to the prohibitive costs of microchip fabrication, the scope of the project will be realized without actual physical delivery. Thus, the group approaches this project with the mindset of developing a proprietary IP block (the transmitter) to consign for sale at a major fabrication plant (see Section 1.3).

1.3 Outline of Technical Specifications

Following technical guidelines from the project technical supervisor, the group embarked on the transmitter design with a broad range constraints to be observed:

1. Serializer: A half-rate serializer with programmable modes for different data rates;
2. FIR Filter: A 3-tap FIR filter with pre-cursor, main cursor, and post-cursor control to minimize ISI and optimize signal integrity;
3. TX Driver: A voltage-mode or CML TX driver with full-scale launch amplitude control, ensuring robust performance across PVT;
4. Duty Cycle Correction: A closed-loop DCC system capable of correcting duty cycle distortions in the clock, with a correction range of ±5% of the clock period.

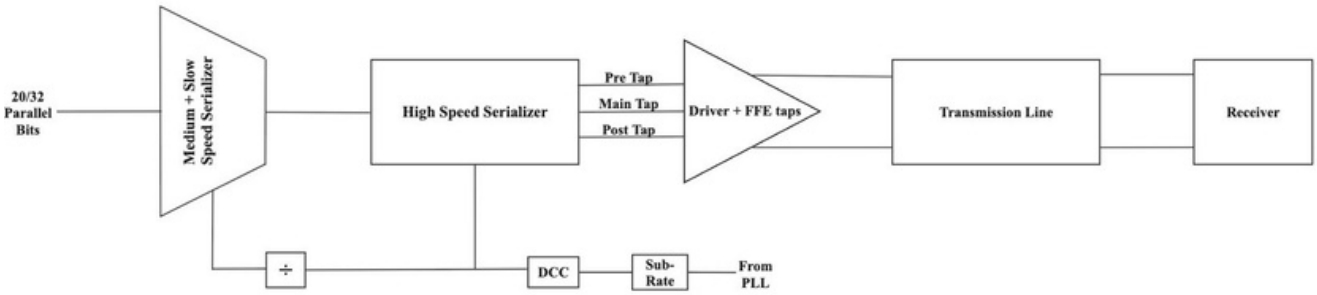


Figure 1: High-level block diagram of the proposed transmitter

To better embody the role of an IP vendor, the technical supervisor functions as the “client” for this product when presenting the set of specifications. Table 1 outlines the key technical requirements specified and define what could comprise a fully defined and implemented transmitter. Over the course of the project, the group realized all but a handful of requirements due to specific constraints such as technical limitations, design trade-offs, or unforeseen challenges encountered during development. A detailed explanation of these circumstances, along with the rationale behind the decisions made, will be provided in their respective sections.

Table 1: Implemented Specifications for the Transmit Path Subsystem

#	Requirement Operating Temperature	Min	Typical	Max	Units	Implemented?
1	Operating Voltage Input Signal Amplitude	-40	25	125	◦ C	Note A
2	Driver FIR-tap Step Granularity Driver	0.95	1.0	1.10	V	Complete
3	Pre-/Post-CursorDe-Emphasis Range Driver	1000	-	-	mVppd	Complete
4	Output Impedance Data Rate (Config 1) DCC	-	-	25	mVppd	Complete
5	Correction Range DDC Maximum Actuator	-	8	-	dB	Complete
6	Step Size DCC Referred Offset, Random	90	100	110	Ω	Complete
7	Input Data Rate (Config 2) Data Rate (Config	8	16	-	Gbps	Complete
8	3) Digital Interface Bus Size Operating	-	Tclk	-	s	Note B
9	Frequency Transistor Finger Width	-	Tclk	1.03 T_{clk}	s	Note B
10		-	-	0.25	mV*	Note B
11		4	8	-	Gbps	Complete
12		5	5	-	Gbps	Note C
13		20	-	32	bits	Note D
14		125	-	500	MHz	Complete
15		-	5	-	µm	Complete

Note A : Late-stage issues with the medium-stage 4-to-1 MUX design and with production of layout cells negated previous test results and prevented the group from obtaining final, useful PVT test results. The issue is resolved but has not been implemented. Details and considerations are further discussed in Sections 3 and 4.

Note B: The complications described in Note A extend to the proposed DCC module, the implemented version of which was insufficient.

Notes C & D : Implementation of the programmable bus size interface was not successful, and a 20-bit bus remains unrealized. See Section 3.1.

1.4 Design Approach: Theoretical Considerations

1.4.1 Top-levelFunctionality

The underlying design philosophy described above seeks to balance the difficulty in generating and maintaining high-frequency data streams with the corresponding emphasis placed on devices involved in the signal path. That is to say: as little of the data path should be exposed to high-speed signals as possible, such that deleterious effects observed at these frequencies are mitigated. Such a philosophy encourages an asymmetric top-level fanout in which the high-speed clock and data circuitry is minimized and more highly optimized than the low-speed modules; the latter trades its relative ease of implementation and maintenance for more of the upfront pre-

serialization work, while the former is given less functional burden to encourage a cleaner high-speed response.

These decisions extend to the equalizer and driver modules, where maintaining signal integrity at high final transmission rates is critical. These components are located at the far end of the transmitter, making them prone to timing issues due to the very high operating frequencies. Additionally, these timing issues may arise from the circuit not having enough gain or swing. This can be problematic, as it may prevent the equalizer's decision circuit from functioning properly. Consequently, our transmitter may not be able to fully launch an equalized bit pattern into the lossy channel.

All circuit components were built, tested, and underwent high-level integration simulations, largely using the Cadence Design EDA tools. The selected software includes the Virtuoso Schematic Editor and Layout Editor (for construction and design), as well as the ADE Product Suite (for testing and verification). Furthermore, following CMC guidelines and with permission from TSMC, we used their 65 nm nodes as the basis for building our transmitter.

1.4.2 EqualizationofTransmissionChannel

After pre-serializing, the second major function of the transmitter is equalization of candidate transmission channels at the data rates presented in Section 1.3. Equalization is essential for improving the quality and reliability of communication systems by mitigating the effects of distortions, noise, and ISI that degrade the transmitted signal within the channel. ISI is an especially important metric that is referenced often in such systems: it occurs when the response due to a given symbol at any point is affected by other symbols in the sequence, making it difficult for the receiver to accurately distinguish between consecutive symbols. Therefore, it is an essential parameter to measure the performance of a channel if it has insufficient bandwidth, mismatched impedance, or dispersive characteristics.

As shown in Figure 2, the selected channels exhibit increasing losses at higher frequencies, with the channel from William Peters at Intel [1] on the left and the channel from Erdem Matoglu at Amphenol [3] on the right. At the Nyquist frequency of the targeted data rates, losses exceed -20 dB, indicating the need for equalization at 8 Gbps and higher. Since the channel on the left supports frequencies only up to 15 GHz, the channel on the right will be used for the 16 Gbps test case.

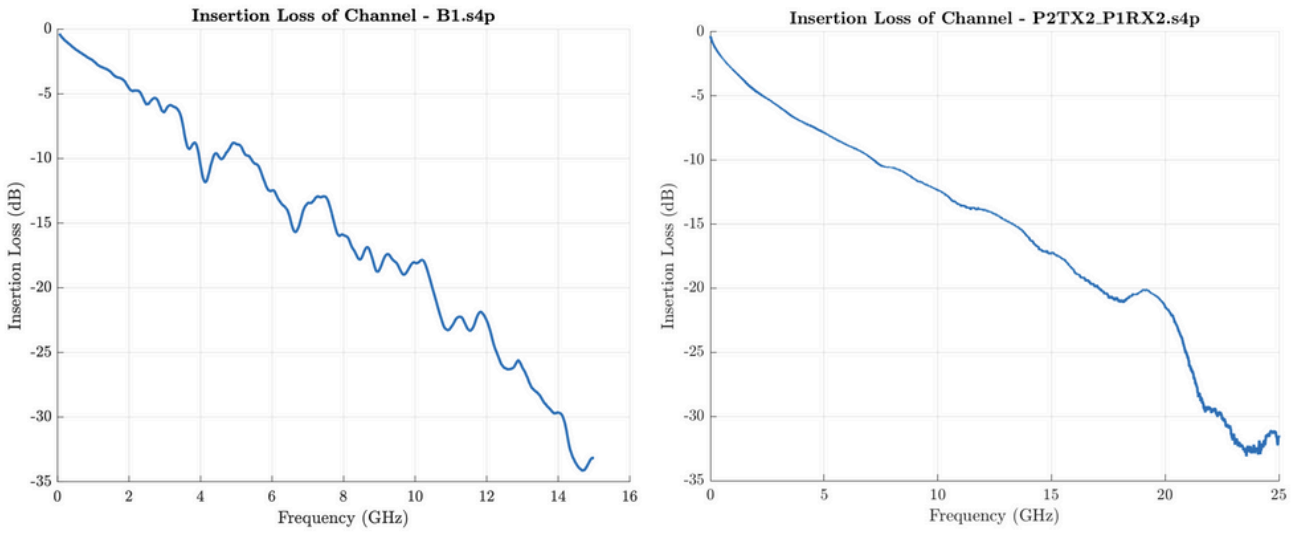


Figure 2: Insertion Loss Plots of the B1 (left) and P2TX2 (right) channels

In a communication system, an eye diagram is a visual representation of the performance of the channel is created by overlaying multiple signal waveforms from consecutive symbol periods (or bits) on top of each other. In the case of our transmitter, data transmission should be carried out using the NRZ signaling. Hence, the binary data is represented by two distinct voltage levels without returning to a baseline between bits. The key metrics for analyzing such diagram are the VEO and the intrinsic jitter. The former is the distance between the minimum 1 value (or maximum 0 value) and the decision threshold; the overall VEO of the transmitter under design is the differential peak-to-peak voltage between these values. The latter refers to the variation in the transition of bits as they move from high to low within the eye (i.e., between bit periods).

This is further exemplified when examining the eye diagrams of both tested channels. In Figure 3, at a data rate of 8 Gbps, the VEO is seen to be small but still observable – the eye remains "open". However, in Figure 4, the channel exhibits prohibitive levels of ISI, making it impossible to observe a VEO – the eye is "closed". This underscores the importance of channel equalization to ensure proper distinction between the bits during transitions.

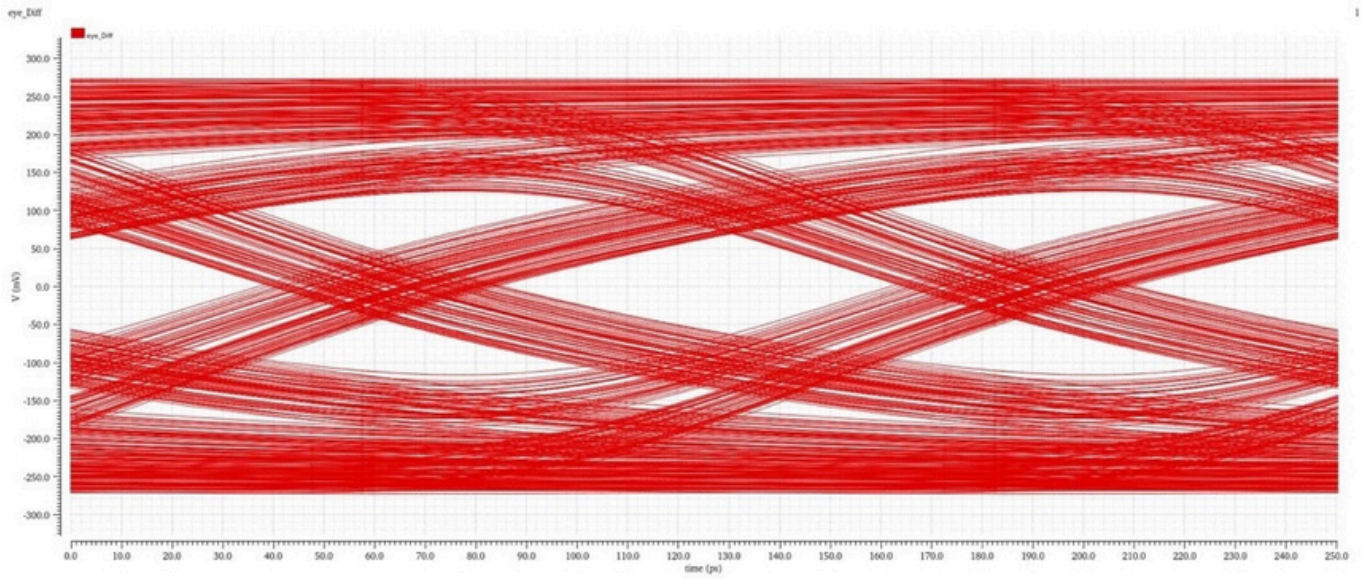


Figure 3: Far End Eye Diagram for B1 channel model at 8 Gbps

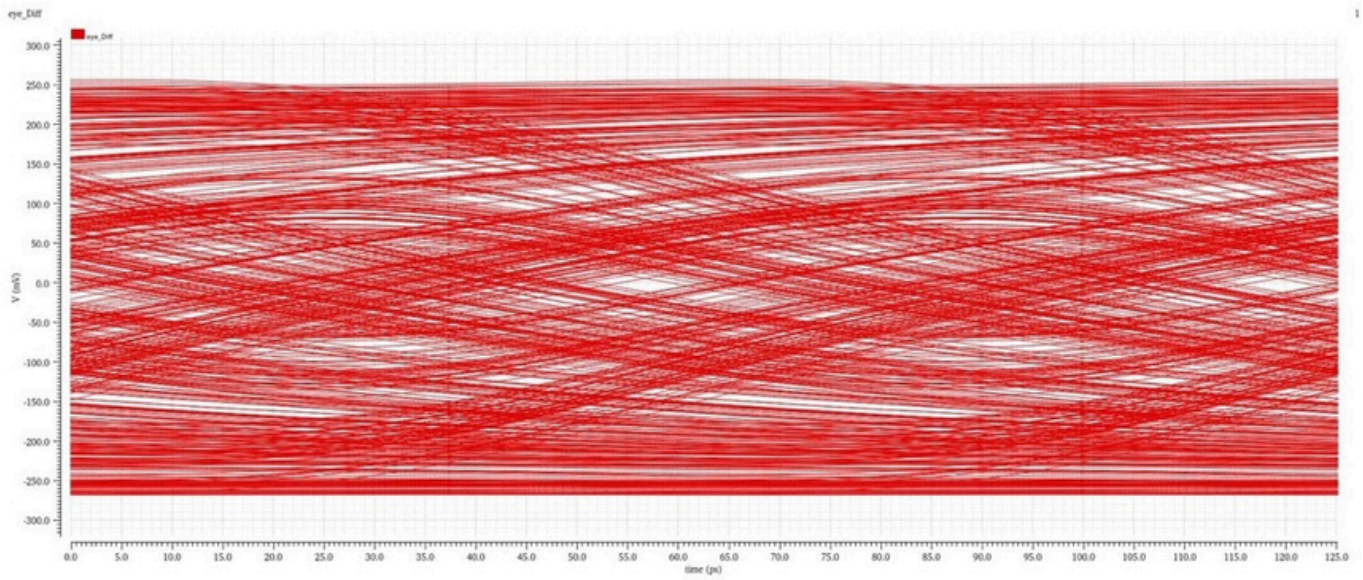


Figure 4: Far End Eye Diagram for P2TX2 channel model at 16 Gbps

1.4.3 SignalDomain:CurrentModevs.VoltageModeLogic

In the current landscape of SERDES designs, two signal domains are typically used to design high-speed wireline links. This section will lay out the difference between them along with a brief overview of fundamental circuits.

Current-mode Logic (CML)

In a conventional CML circuit (see Figure 5), the differential pair operates by steering current between the two branches based on the input voltage difference. The shared current source at the tail ensures that the total current remains constant such that when one transistor conducts more, the other conducts less. This behavior enables fast switching and stable operation at high frequencies.

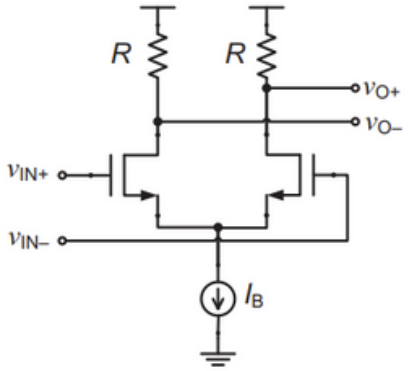


Figure 5: Differential pair, showcasing an example of a CML-domain circuit [2]
The load resistances at the drain terminals convert the current variations into voltage swings, which are then used as logic levels. The advantage of CML circuits is their high-speed operation due to reduced voltage swings and constant tail current, which minimizes power supply noise. Moreover, the value of the bias current and common mode voltage can be easily calculated by selecting a target voltage swing. For the transmitter under design, this is specified as 500 mVppd, which provides:

$$I_{bias} = \frac{V_{swing_{pp}}}{2R} \quad (1)$$

$$V_{cm} = VDD - \frac{I_{bias}}{2R} \quad (2)$$

The interrelation between these design parameters is clearly a multifaceted one, and so optimizing transmitter performance means their balance must be carefully considered. The bias current directly affects the voltage swing of the differential pair in conjunction with the load resistance. While increasing bias current improves swing, excessive current can degrade signal integrity if the load resistance is too low. Moreover, the gain of the differential pair is found via:

$$A_V = g_m R \quad (3)$$

where g_m is the transconductance, which depends on the transistor width and bias current. To maintain a gain of at least 2 V/V, both the transistor width and load resistance must be carefully sized. Increasing either of these parameters brings associated trade-offs: higher load resistance

increases gain but slows the transient response, while a wider set of transistors provides a higher g*m* (increasing available current) but also increases capacitance (slowing the device's switching speed).

Excessively large resistance and capacitance can negatively impact signal setup time in non-negligible ways. A high load resistance limits bandwidth, while increasing the capacitance delays transitions and affects overall performance. Therefore, achieving the optimal balance between gain, speed, and signal integrity in CML requires careful consideration of these design trade-offs.

Voltage-mode Logic (VML; CMOS)

In recent years there has been a resurgence of inverter-based circuitry in SERDES designs due to its efficiency in high-speed applications. Inverters are VML circuits – that is, they operate in the voltage domain; see Figure 6.

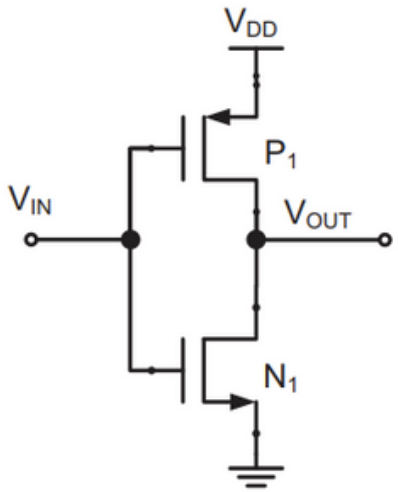


Figure 6: CMOS inverter, showcasing an example of a VML-domain circuit [2]

Unlike CML, which operates by steering currents and keeping transistors in saturation, VML circuits rely on a voltage source for switching between full-rail signal levels. Transistors primarily operate in the triode (linear) region, functioning as voltage-controlled resistors rather than active current sources. This design choice helps reduce power consumption and improve switching speed. Moreover, as stated, an inverter in VML is not a differential circuit but a full-rail circuit; this introduces some potential issues such as:

- Slower switching compared to CML, due to larger voltage swings;
- Higher capacitive loading, which tends to increase propagation delay;
- Increased rise/fall times, limiting the maximum frequency;
- Higher susceptibility to supply noise, as the signaling is no longer differential.

While these are considerable challenges to surmount when designing SERDES circuits, VML's ease of programmability through the use of slices (multiple copies of the same circuit) is an advantage. VML circuitry is nonetheless harder to tune than CML because it is more difficult to impedance match, and VML circuitry is more sensitive to sizing and signal integrity because of the large voltage swings.

Design Approach: CML

The group ultimately decided on CML domain signaling for the overall transmitter design. Since the high-speed serializer outlined in the design specifications (see Section 1.3) is intended to operate with fully differential inputs and outputs, it was more beneficial to avoid using CMOS circuits in the main signal path to eliminate the need for additional circuitry for conversion between single-ended and differential signals. The trade-off between power use and signal integrity (especially switching speed) was also deemed acceptable. The exceptions are the LIDO circuit used for front-end signal domain conversion (as it is not technically a differential circuit; see Section 3.1), and the 10 ps clock buffer used in the high-speed serialization stage (as the target signal is itself full rail; see Section 3.2).

2 INITIAL DESIGN CONSIDERATIONS

2.1 Pre-serializer

2.1.1 Functional overview

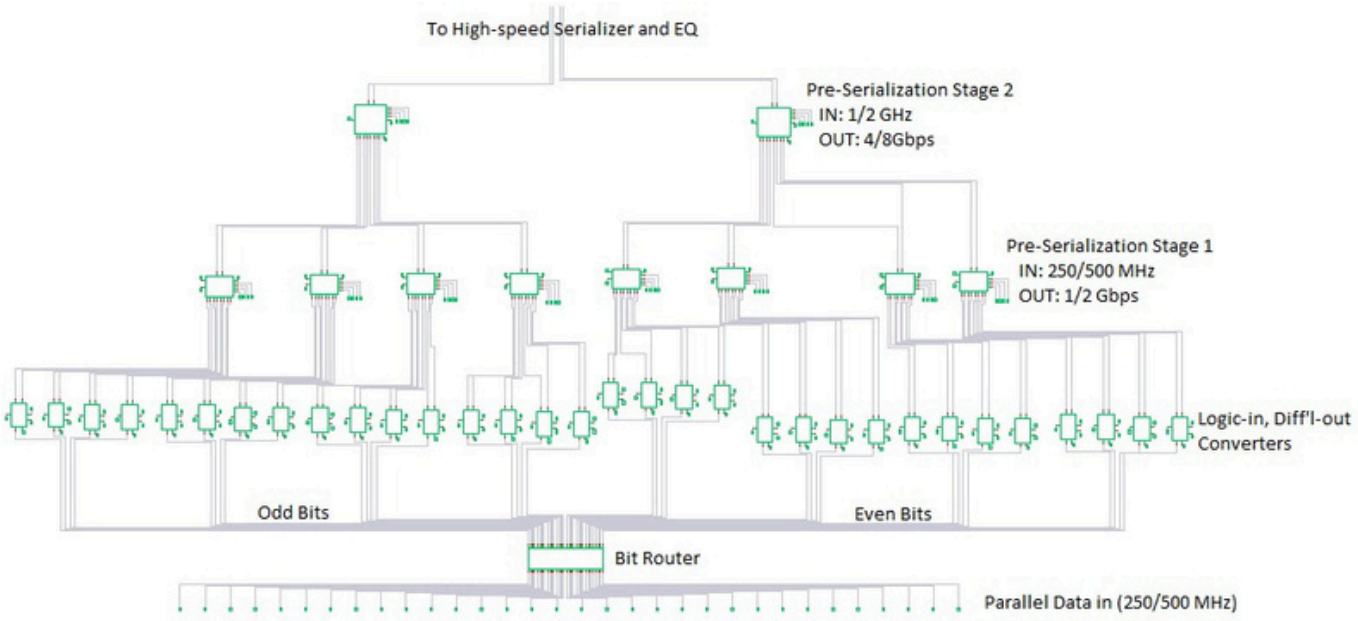


Figure 7: High-level schematic diagram of pre-serializer module

Initial serialization of data is carried out by a dedicated pre-serializer module (Figure 7) before passing the penultimate bit sub-streams to the high-speed stages. Decoupling the high-speed and pre-serialization stages mitigates unwanted effects arising from the clock network driving the subcomponents in each. That is to say: with the philosophy of minimizing the amount of circuitry facing the high-speed clock, the module should be simpler and more lightweight at the fastest substage operating speeds, while any required shaping or processing should be performed at low speeds.

2.1.2 DesignMethodology

Four key functional sub-requirements were identified for a successful pre-serializer:

1. Parallel bits should be curated if necessary; it should not be assumed that MSB-to-LSB order is optimal for operation.
2. Pre-serialization substages should be symmetrical to ensure a balanced, consistent, reliable flow of bits through the module; this helps mitigate possible edge cases and prevents extraneous correction circuitry.
3. Operation should assume non-ideal sources and thus the design phase should anticipate circuitry to handle a voltage supply grid and current mirror network.
4. Likewise,dataandclocksignalsshouldbeassumednon-ideal;suitabledelayelements,duty cycle correction, or pre-filters might be required.

Considering the design philosophy outlined in Section 2.1.1, the datapath will see more circuitry in the slower stages that are designed to convert signal domains and perform smaller serialization

steps. Buffers and related delay elements are also included, as fine-tuning of signals will likely be required during high level integration of lower-level subcomponents designed for different operating speeds.

In the same vein, clocking subdomain circuitry and current mirror network branches are also expected to be denser during slower stages. This is due a) to the predicted higher number of subcomponents required to handle a larger number of substreams and b) to provide the balance of clock complexity and data path exposure mentioned above.

2.1.3 ArchitecturalOverview

The datapath will make extensive use of multiplexing. In a naï¨ve analysis, an N-to-1 MUX is capable of serializing N parallel data streams. If each can produce an output of data rate R, then the MUX can output a maximum data rate of NxR if appropriately designed. Applying such pre-serializing steps sequentially should allow a parallel stream of single bits to be fully serialized via a number of intermediately sized bitstreams (Figure 8). A desirable CML MUX template has an overall load resistance driven by N branches of current via differential pairs and relevant control circuitry.

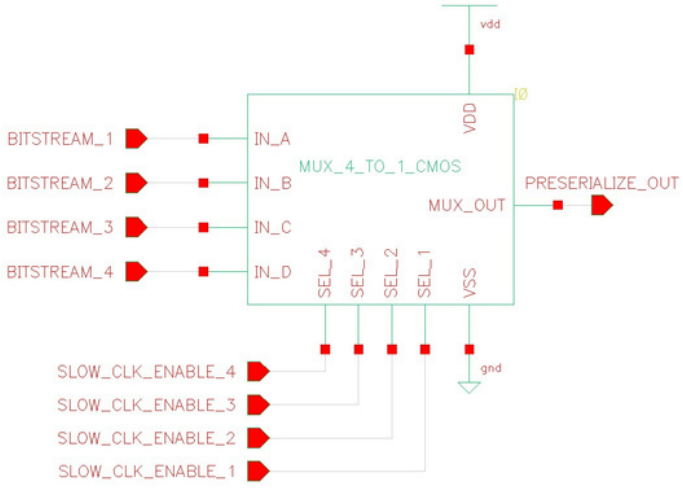


Figure 8: Instance of a pre-serializer stage implemented as 4-to-1 multiplexer

The datapath therefore relies on the MUX operating not via static enable signals but with a set of clocks, tuned to control the sequential extraction of bits from the N input signals. Planning the type of clock signals used, and how they are delivered to the MUX, also involves planning how the input data streams arrive at the MUX and propagate to the output. The clock rate should also be programmable, as the overall transmitter must support different data rates, and so additional control (assumed at this stage to be similarly comprised of e.g., MUXes and buffers) in this domain will also be necessary.

N.B.: Lower-level considerations of differential pairs, including representative calculations, are discussed in Section 1.4; details of subcomponent blueprinting and proposed operation follow below in Section 3.1.

2.2 High-speed Serializer

The transmitter's high-speed stage consists of three classes of subcomponents designed to interface the preserialized data with the channel: latches, pre-drivers, and a final rank of MUXes

(Figure 9). This module includes all circuitry used to condition the data and prepare additional time-shifted copies before passing control to the equalizer taps and output driver.

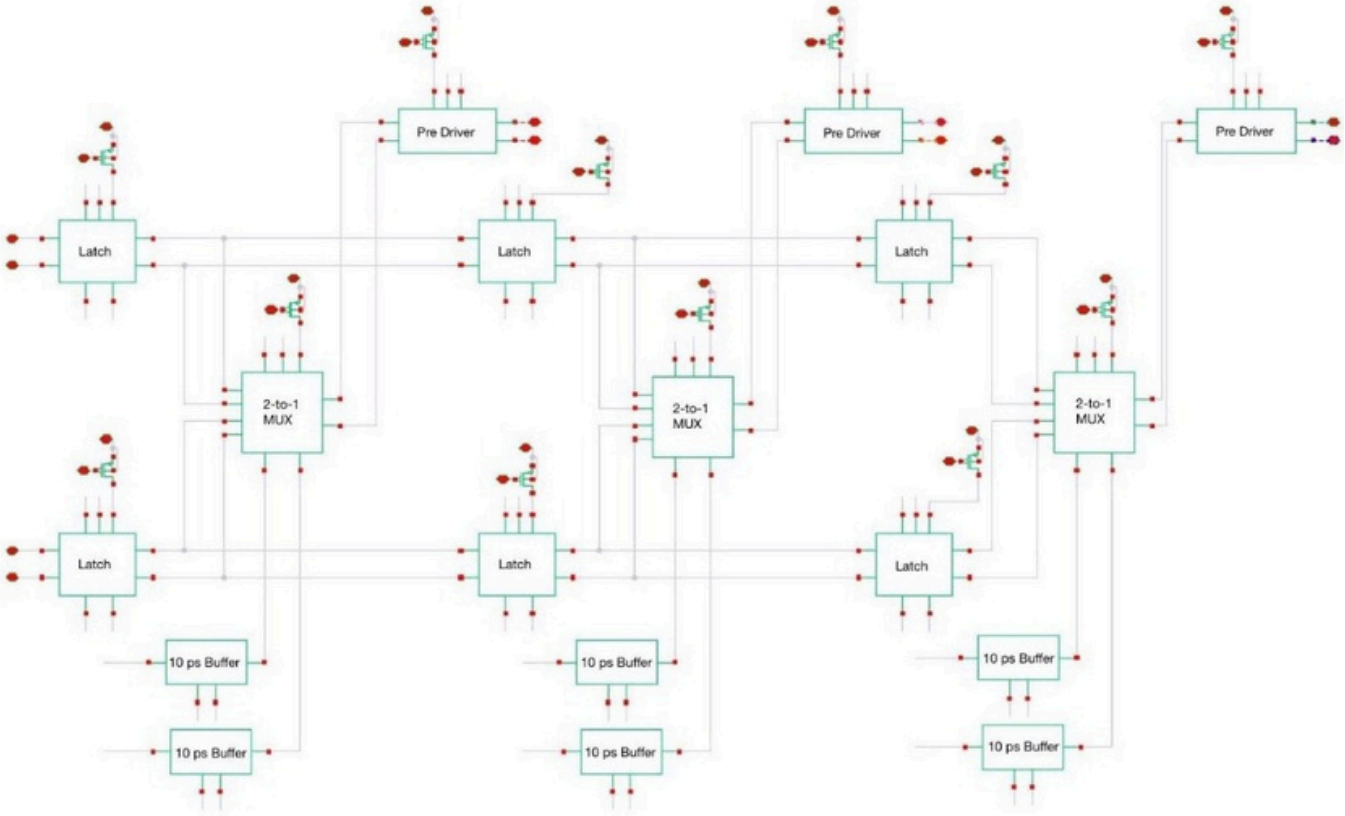


Figure 9: Top Level Schematic View of the High-Speed Serializer

2.2.1 FunctionOverview

As mentioned, the high-speed serializer provides the final stage of the serialization path after the preserialization. Optimization of this stage is crucial because it generates a one UI delay for the given input, allowing it to be equalized in the final stage of the transmitter path; see Section 3.3 for details. It should be noted that this module operates at half the clock rate. Therefore, for a 16 Gbps output the latch operates on an 8 GHz clock – hence the “half-rate serializer” design specification.

Figure 10 demonstrates the functionality of both the latch and 2-to-1 MUX. When the clock signal is LOW the odd latch is in its opaque state and is holding its previous data, which is then passed to the output (denoted as ‘Odd’). Simultaneously, the even latch is in its transparent state, allowing the even data during that period to pass through to the MUX input (denoted as ‘Even’).

Conversely, when the clock signal is HIGH, the odd latch becomes transparent, allowing the odd data during that period to pass through to the MUX input (denoted as ‘Odd’), while the even latch becomes opaque and holds its previous data, which is then passed to the output (denoted as ‘Even’).

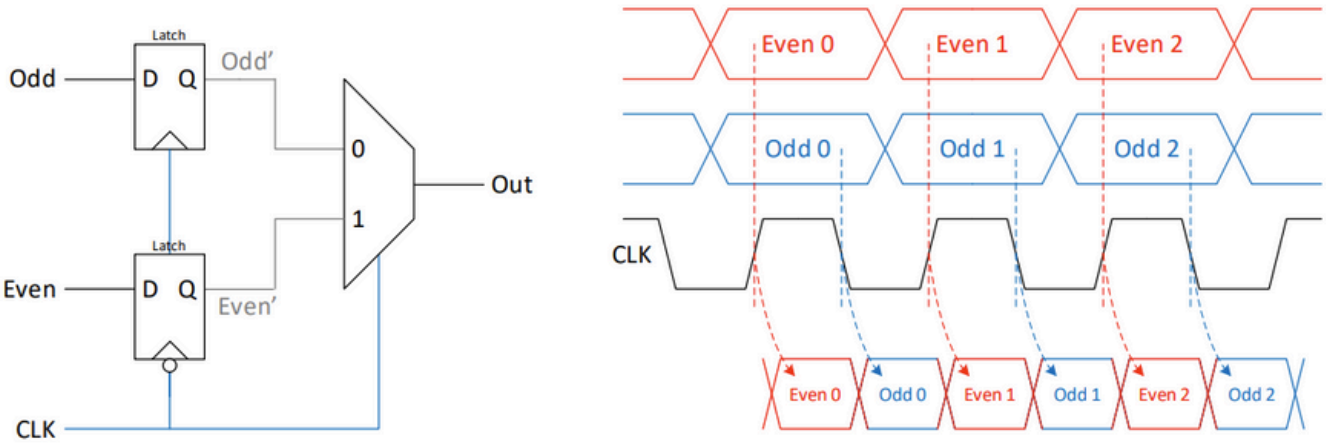


Figure 10: Timing Plot of Functionality of the High-Speed Latch [5]

Cascading the latches as shown in Figure 10 creates a one-UI delay at each stage of the output of the MUX. Copies of the delayed data can enable FIR EQ generation since the delayed data are separate into pre, main, and post cursor signals; details are presented in Section 2.3.

2.2.2 DesignMethodology

CML latch

The latch is an essential component for FIR EQ generation in that it creates the desired UI delay. Moreover, sufficient gain must be realized during regeneration to ensure the correct bit is output to the next stage. Therefore, a level-sensitive CML latch shall be designed with a gain of approximately 2.0 V/V and an output swing of 500 mVppd.

The main parameters affecting the swing are the bias current and the load resistance. Hence, the latch is designed to provide an optimal balance between sufficient output voltage swing and high-speed operation. By carefully selecting the bias current, the voltage swing remains large enough to maintain signal integrity and adequate noise margins while avoiding excessive power consumption and minimizing voltage headroom issues. Similarly, the load resistance is chosen to complement the bias current, optimizing the RC time constant to achieve fast transition times and meet the required setup and hold time constraints for reliable data latching. A candidate high-speed latch would thus provide enough gain during regeneration while allowing for adequate hold and setup time at the highest operating frequencies.

CML 2-to-1 MUX

The 2-to-1 fully differential MUX is the second subcomponent in the high speed serializer. Taking the outputs of two latches as its inputs, the MUX will selectively see the odd or even bits depending on the phase of the clock. Ideally this device should affect neither the gain nor the setup and hold times of the latch: the purpose of the MUX is only to combine the outputs of the even and odd latches into a single serialized data stream.

Key performance metrics for the 2-to-MUX include:

- The ability to switch between inputs very quickly, with minimal propagation delay, to keep up with the high data rates;
- Maintaining the swing of the latch to maintain signal integrity;
- Ensuring the output can suitably drive the load capacitance of the following stage.

CMOS Buffer During implementation of the MUX, the group encountered timing issues when testing at higher
frequencies cause by setup time violations, the details of which are explored in Section 3.2. This experimentation resulted in a shortlist of critical design considerations for the CMOS buffer:

Buffer Placement - the buffer was placed at the input of the MUX for the clock signal rather than at the input of the latches for the data stream. This decision was driven by the need to address setup time violations: placing the buffer at the clock input ensures that the clock signal is delayed, allowing the data signal to meet the setup time requirements. Conversely, placing the buffer at the data input would have addressed hold time violations, which were not the primary concern here.

Buffer Type Selection - The choice between a CMOS buffer and a CML buffer was influenced by the nature of the signal being buffered. Since the signal in question was a full-rail clock signal, a CMOS buffer was deemed more appropriate due to the nature of CMOS circuitry operating at full-rail. CML buffers, while offering advantages in terms of speed and noise immunity, require a constant current to drive the differential pairs, leading to higher power consumption. In contrast, CMOS buffers, which are based on inverter designs, consume significantly less power, making them more suitable for this application where power efficiency is a priority.

CML Buffer (Pre-Driver)

The main role of the CML buffer is to act as a pre-driver and increase the current drive capability. The output driver typically has a lower resistance compared to the previous stages of the high-speed serializer, and so a candidate pre-driver should support this high electrical effort between the decision circuitry and the launch and FFE drivers. By providing sufficient drive strength, the CML pre-driver ensures that the transitions remain fast and clean, preserving signal integrity as it passes to the output stage. Additionally, the pre-driver isolates the sensitive core logic and decision circuits from the large capacitive loads and potential noise sources associated with the output driver. This isolation helps minimize jitter and reduces the risk of data-dependent switching noise propagating back into the serializer core. Finally, the pre-driver allows for fine control over the output swing and the impedance presented to the final driver stage, ensuring that the signal is properly conditioned for high-speed transmission across the channel. Key performance metrics include:

- Biasing of the transistor with enough current so it maintains the specified voltage swing;
- Minimizing of RC time constants

2.3 Driver & FIR Taps

The final (channel-facing) stage of the high-speed transmitter is shown in Figure 11:

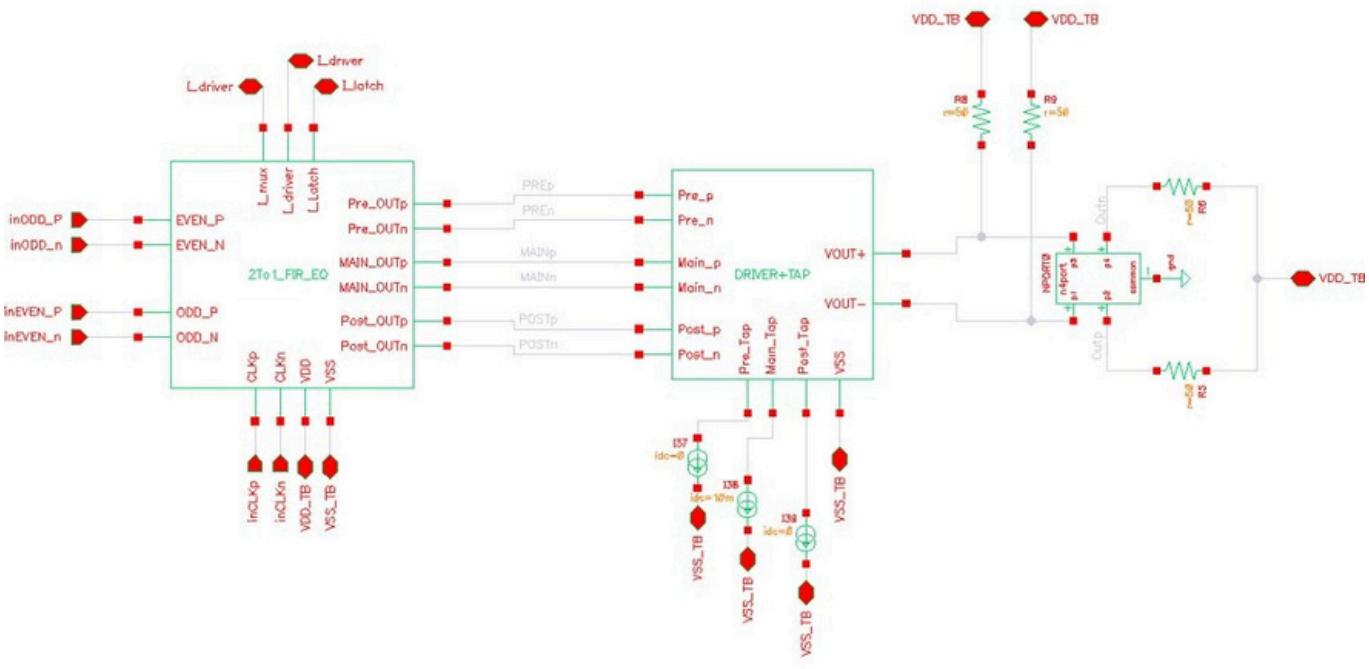


Figure 11: Output Stage Connected to the Transmission Line Schematic

2.3.1 FunctionalOverview

As described above, delayed copies of the serialized data are divided into three data streams. Since these delayed serialization data streams represent different phases of the same signal, they can serve to correct for pre, main and post cursor ISI and provide an equalized overall signal at the transmitter end of the channel.

2.3.2 DesignMethodology

Output Driver

As mentioned in Section 2.2.2.4, the output (launch) driver is designed with a load that matches the characteristic impedance of the transmission channel – specifically, 50 Ω. Impedance matching is essential for minimizing signal reflections and maintaining signal integrity across the high-speed link by preventing signal degradation due to standing waves or echoes, which would otherwise lead to increased jitter and data errors at the receiver.

One common method to increase the gain of a stage is to either widen the transistors or increase the load resistance. However, in the case of the output driver, increasing the load resistance is not an option, as the 50 Ω termination is required for impedance matching. To maintain the target gain of approximately 2 V/V and achieve the necessary peak-to-peak voltage swing across the low-value load resistor, a higher bias current is needed. This requires using larger transistors to accommodate the increased current drive capability. However, increasing transistor widths also introduces greater parasitic capacitances e.g., at the gate/drain and the drain/bulk. These parasitics degrade the speed of the output driver by extending the rise and fall times, which negatively impacts the signal's eye diagram and overall timing accuracy. Additionally, the increased capacitive loading at the driver's output makes it more challenging for the core logic to directly drive the output stage without performance degradation. Hence, the inclusion of the pre-driver, as discussed in the previous section, becomes critical. The pre-driver acts as an intermediate stage that buffers the high-speed logic or serializer output from the heavy loading effects of the output driver.

Key performance metrics include:

- Output impedance matching to the channel of 50 Ω;
- An appropriate bias current to provide a swing of 500mVppd with gain of approximately 2.0 V/V, according to Section 1.4;
- Balanced transistor width so as to not overly increase parasitic capacitances, while increasing gain to avoid limiting bandwidth and transition times.

FFE taps

The design of the FFE taps is very similar to the launch driver design in terms of topology and operation (Figure 12). However, a key difference is that the FFE taps do not include a load resistor to their outputs, consisting instead of bare differential pair structures and relying on current steering to provide the necessary equalization signals. These differential pairs are designed with the same transistor sizing described in Section 2.3.2.1, ensuring consistent current drive capability and matching characteristics. By omitting the load resistor, the FFE taps contribute additional controlled current directly into the summing node of the output driver, effectively shaping the transmitted signal to compensate for channel loss and ISI. This approach allows for fine-tuning of pre-emphasis or de-emphasis levels in the transmitted signal without introducing additional impedance discontinuities or reflections in the output path.

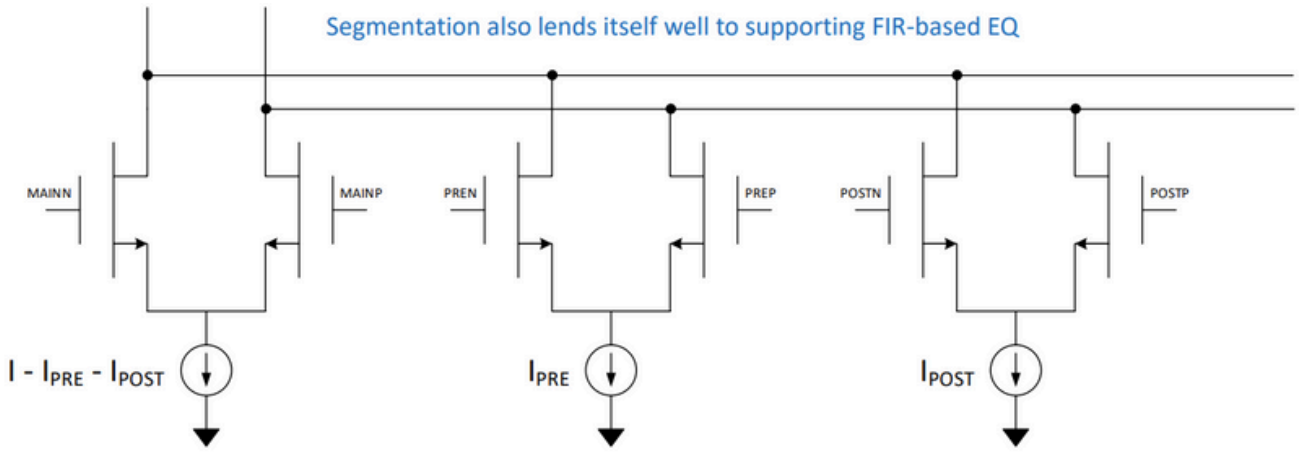


Figure 12: Circuit Schematic of the Equalization taps [5]

2.3.3 TheoreticalTaps

As seen in Figure 12, each output of the MUXes (Section 2.3.1) for the FIR EQ generation circuit serves as the input to the drivers. The taps have a value of α (FIR tap weight) that will be calculated by MATLAB using a least square algorithm that uses parameters of a channel by using s-parameters. The code is inspired by Professor Parlermo code from his course ECEN 720 [4]. It launches a pulse to find the optimal values from a random sequence of bits. The main tap value was determined to be 20 mA from the calculations in Section 1.4 To find the values of the pre- and post-cursor taps, a pulse can be launched into the transmitter to obtain a response similar to Figure 13. This step is essential because the tap values can be determined by recognizing that the peak corresponds to the pulse at the transmitter output. By dividing the peak by the pre- or post-cursor, the value of α can be determined, and the tap values by multiplying α by 20 mA. To maintain a total launch current of 20 mA, the main tap is reduced by the sum of the pre- and post-cursor tap values.

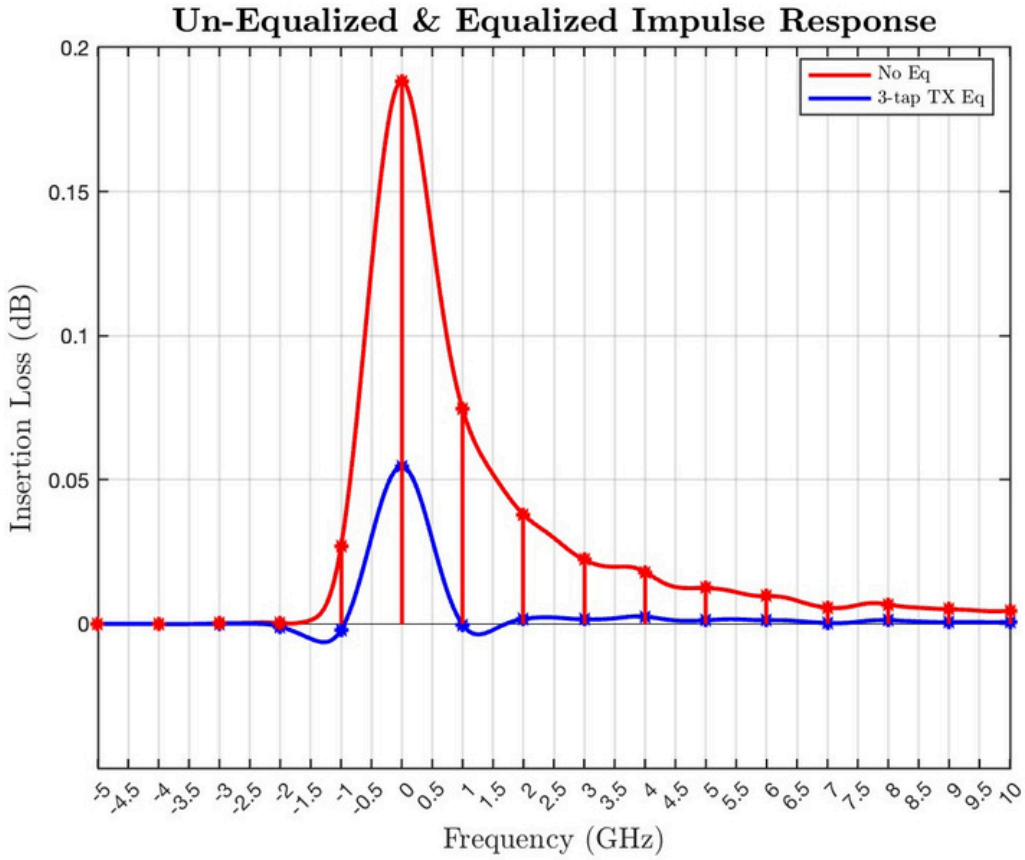


Figure 13: MATLAB Simulation of Equalized (blue) and Un-Equalized (red) Impulse Response at 8 Gbps

3 DESIGN IMPLEMENTATION

3.1 Pre-serializer

3.1.1 Circuit-leveldesignoverview

Signal Domain – The LIDO Converter: The mismatch between input signals (full-rail, NRZ logic) and overall transmitter output (differential signalling) necessitates some point in the datapath where the domain of the signals is converted. Initial attempts to integrate this function into existing e.g., multiplexers proved difficult to tune output responses in the presence of clock signals; a longer timeframe may have permitted further optimization in this sense, but ultimately the decision was made to modularize the conversion operation.

The LIDO converter is a resistively-loaded differential pair (see Figure 5 above) that takes the incoming full rail swing signal as one input and a reference voltage as the other. In this way the device acts as a sort of pseudo-comparator that produces a differential signal of a given swing, polarity, and common-mode voltage for high input, and an identical signal with opposite polarity for low input.

Ranks of Preserialization: Section 2.1 describes the concept of multiplexing parallel data through various preserializing stages before passing the final sub-streams to the high-speed circuitry. With the final serialization defined as a 2-to-1 stage, a 32-bit input bus can be effectively serialized by two stages of 4-to-1 MUXes. The first rank of eight 4-to-1 MUXes creates eight, four-bit sub-streams whose output rates are four times that of the input. The second rank consists of two 4-to-1 MUXes that results in two, sixteen-bit sub-streams at sixteen times the original input rate. These two sub-streams are passed to the high-speed stage, where the 2-to-1 MUX creates the final serial bitstream at the target output rate.

Initial designs targeted e.g. five ranks of 2-to-1 multiplexers, or in any case an increased number of smaller pre-serialization stages rather than a single fast 2-to-1. In the interest of signal integrity and reasonable circuit responses these were ultimately discarded, as theoretical calculations and early testing suggested unfeasibly slow operation overall.

Bit Routing: The incoming parallel data can be conceptualized as a 32-element bit array arranged from most-significant to least-significant bit. Because the final serialization step handles two data sub-streams, it is in fact more useful to initialize the datapath with a bit router that separates the parallel data into odd bits and even bits, which represents alternating bits in the array. By delineating the initial data in this fashion, interleaving the final bitstream by alternating inputs in the 2-to-1 MUX will produce the correct desired serialized signal (see Figure 14).

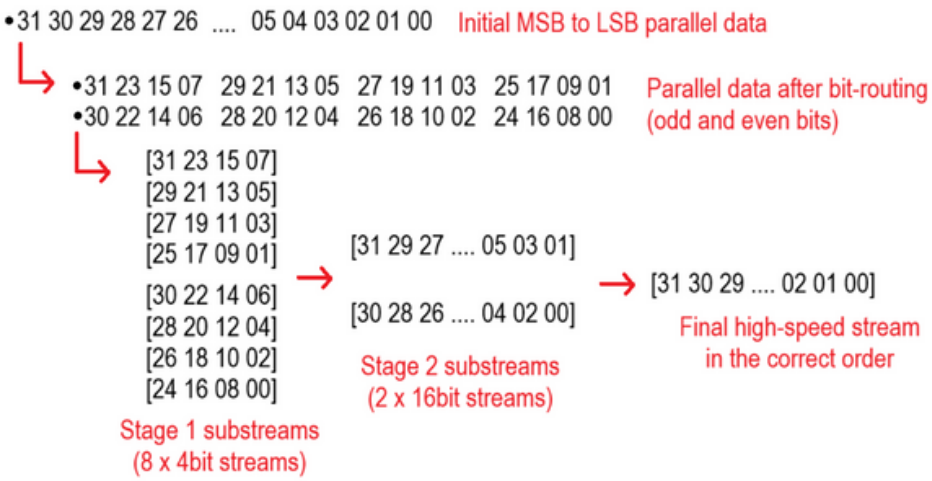


Figure 14: A visualization of the role of a pre-serializer bit-routing scheme

Data Delay Elements: With several clocking subdomains operating simultaneously within the module, the pre-serializer includes various buffers to provide delay to selected data streams. Two primary reasons for this are a) sub-UI tuning of timing constrains to mitigate effects of jitter or to offset inherent propagation delays, and b) fine-tuning of signals during high-level integration of lower-level subcomponents before finalizing parameters. Delay elements consist mainly of CML buffers of certain geometries in series to provide an approximate amount of delay relative to the targeted data stream, with capacitively connected transistors added in shunt to the output to provide finer control. These latter are used sparingly so as not to impact the RC delay characteristics (and thus the integrity) of the output signal too strongly.

3.1.2 ComponentParameters

Table 2 below contains a collection of device parameters used in implementing the pre-serializer.

Table 2: Pre-serializer device components

Subcomponent	Transistor Fingers	Transistor Length (nm)	Resistance (Ω)	Drive Current (μA)	Other Parameters
LIDO Converter	1 (Data) 1 (Bias)	60 (Data) 180 (Bias)	2.5K / 3.0K	100	V_{ref} = 0.5V
Stage 1 MUX	1 (Data) 2 (Clock) 1 (Bias)	60 (Data) 120 (Clock) 60 (Bias)	1.25K	200	Bit Delay = 12% of UI
Stage 1 MUX Delay Element	1 (Data) 1 (Bias) 2 (Cap.)	240 (Data) 120 (Bias) 60 (Cap.)	2.5K	200	60ps during 16 Gbps
Stage 2 MUX	3 (Data) 3 (Clock) 14 (Bias)	60 (Data) 120 (Clock) 60 (Bias)	200	2000	Bit Delay = 16% of UI
Stage 2 MUX Delay Element	1 (Data) 1 (Bias)	240 (Data) 120 (Bias)	1.6K	160	20ps during 16 Gbps

3.1.3 Controlpathdetails

LIDO Converter – Figure 15: A pseudo-differential pair that uses a reference voltage as one of its inputs. A digital (full-rail) voltage signal at the second input produces overall functionality that is similar to a comparator.

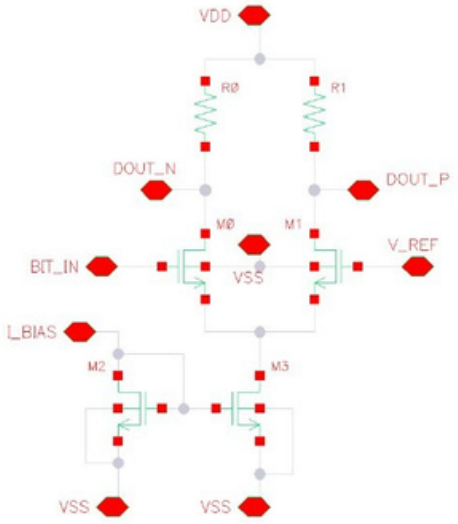


Figure 15: LIDO Converter, schematic view

Multiplexers – Figure 16: The two preserializing stages use 4-to-1 MUXes while creating the various bit sub-streams. Individual select lines are controlled by a set of clocks in quadrature, with pairs of subsequent phase signals driving subsequent inputs (see below, Section 4.1.4).

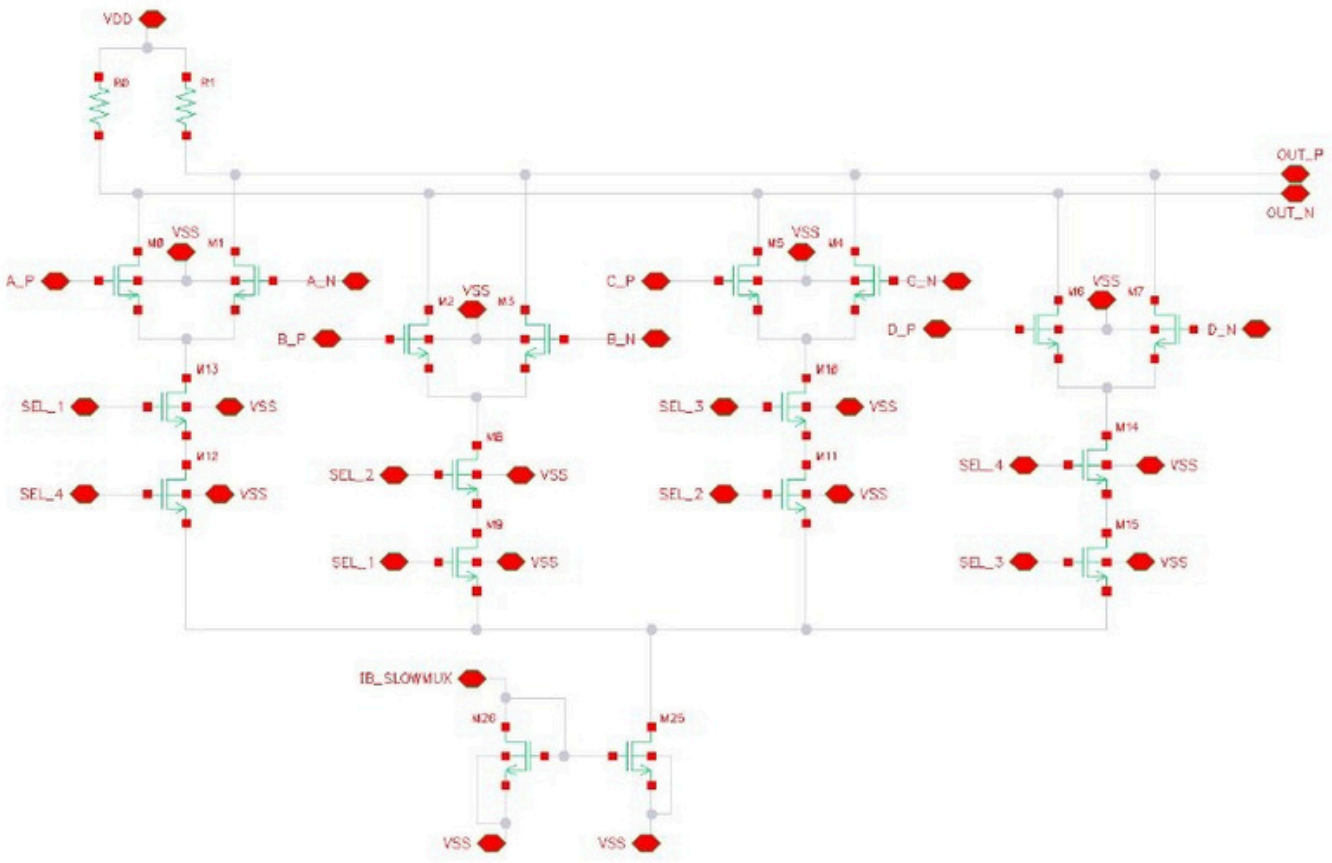


Figure 16: 4-to-1 multiplexer, schematic view

Delay Elements – Figure 17: As discussed, data buffers consist of a two-tiered approach to introducing delay – standard CML buffers to set the approximate delay, and capacitively-connected transistors to adjust the response.

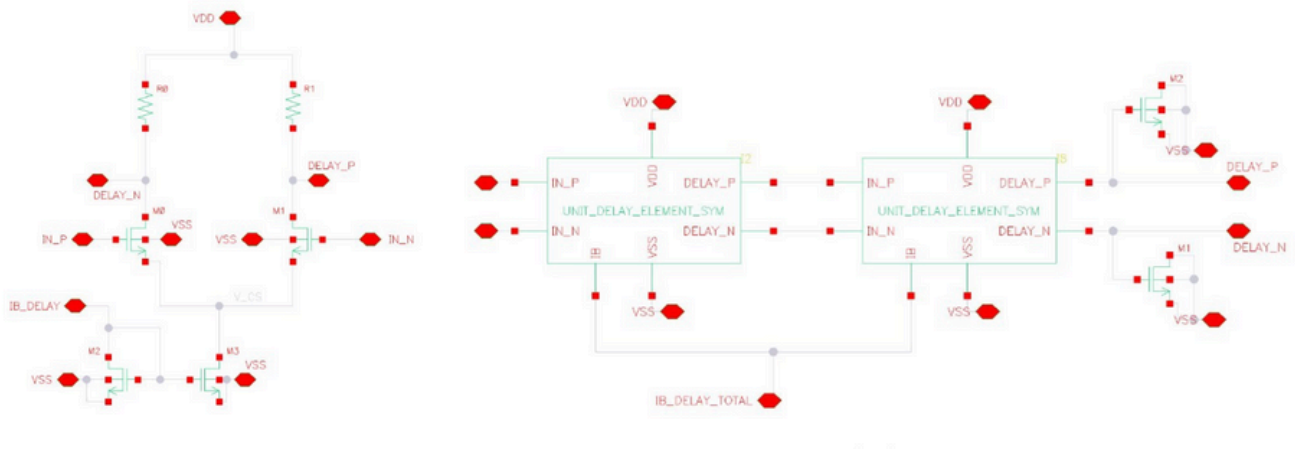


Figure 17: Simple delay element used for 20ps delay (left);

two simple elements in series with added capacitive delay for 60ps delay (right)

Biasing transistors were integrated into the respective subcomponents to allow for high-level adjustment during penultimate integration testing. Further information about the current mirror and clock distribution network can be found below in Section 3.4.

User-programmable Bus Size Module: A specification listed in Section 1 stated that the transmitter should allow for a switchable parallel bus size to allow for 32-bit and 20-bit word length processing. This was one of the earliest challenges tackled by the group but it soon became apparent that the design would rely on a top-level approach; that is, the circuit parameters that allow one transmission scheme to proceed with e.g., suitable VEO and ISI would not necessarily map to the other scheme. The two main approaches to a viable solution included partially or fully modularized transmission datapaths (which seemed very inefficient) and selective shuttering of data substreams to block twelve bits of the original thirty-two (or else fill them with dummy data and discard them).

This latter approach proved the most fruitful in terms of achieved development scope. A second user-programmable MUX-based selection network was prototyped in a similar vein to that of the clocking circuity (see Section 3.4). Functionality would include reset-enabled shift registers for storing outgoing multiplexer data, combined with a counter/accumulator interfaced with the high-speed stage. When the 20-bit mode is selected, the counter becomes active and resets the shift registers every twenty bits. Plans were made to expand the selection network to full capacity, but unforeseen design delays in the principal (32-bit) scheme left further progress unrealized within the allotted timeframe.

3.1.4 SimulationMethodology

Subcomponent design, high-level integration, and all testing and verification procedures were carried out using Cadence Design Tools. Design strategies were taken from research sources, adapted from course materials, and suggested by the co-supervisors. Initial device parameters were calculated using the models and equations described above in Section 1; these preliminary concepts were then built out and iterated upon within the design software.

Simulations and testing were performed primarily using the Cadence Virtuoso ADE Suite. Depending on the module or subcomponent under design, testing included some or all of the following experiments:

- transient (time-domain) analysis of the data path and clock signals, to ensure correct functionality and to tune basic elements;
- bandwidth (frequency-domain) analysis of output signals to gauge the capacity of the circuitry to respond to high-frequency data, as well as the nature of its responsivity;
- operating point analysis to determine currents and voltages at key nodes and branches, which ensures proper operation and helps to verify other experiments;
- parametric analysis, for analyzing design variable sensitivity and for optimizing performance.

Clocking subdomains in the pre-serializer are necessary for the first and second serializing stages, which are driven primarily by 4-to-1 multiplexers. While there are a few methods of selecting consecutive inputs in such a configuration, the chosen technique requires two select (clock) signals per branch to be active at a time (refer to Figure 16) and run the clocks in quadrature. In this way successive clock signals are logically ANDed, providing an overlap of half of their pulse widths. This allows each clock phase to be run at a quarter of the speed of the MUX's output. See Figure 18 for details.

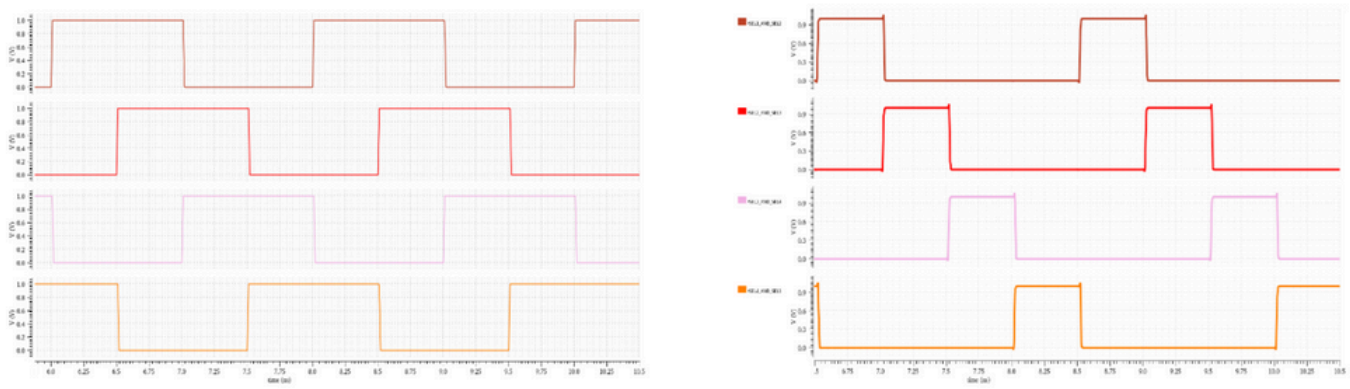


Figure 18: 50% duty cycle quadrature clock signals generated from master clock (left); effective 25% duty cycle MUX enable signals (right)

Representative simulation results for the preserializer module have been selected and are presented in Section 4.

3.2 High-speed Serializer

3.2.1 Top-levelConsiderations

The high-speed serializer is composed of three main types of components (latch, pre-driver/buffer, and multiplexer; see Section 2.2) and generates input for the FIR equalizer. During top-level design (see Table 3), a significant issue arose in the form of jitter at the output of each of the taps' pre-drivers. This jitter was observed during clock transitions, specifically when the clock switched from high to low or low to high. This phenomenon is presented in Figure 19 and indicates that there is a setup time violation from the output of the 2-to-1 MUX.

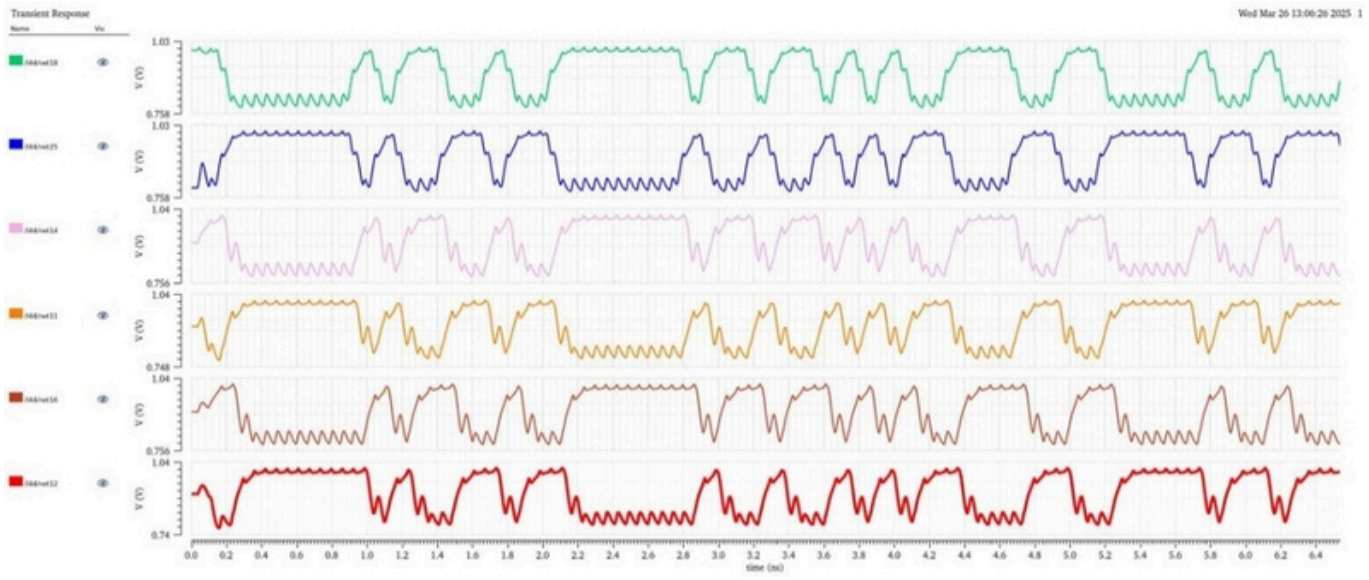


Figure 19: Setup time violations observed at output of the MUX (16 Gbps)

To address this issue, a CMOS buffer was implemented at the input of the MUX for the clock signal. The jitter observed at the outputs of the pre-drivers was primarily attributed to timing violations during clock transitions. In high-speed designs, particularly at data rates of 16 Gbps (with a clock frequency of 8 GHz and a period of 125 ps), even minor timing mismatches can lead to significant performance degradation. The jitter was a direct consequence of setup time violations, which occur when the data signal does not stabilize sufficiently before the clock edge. To mitigate this, a buffer was introduced to delay the clock signal, ensuring that the data signal had adequate time to settle before the clock transition.

Table 3: High-speed Serializer Component Parameters

Component	Transistor Fingers	Transistor Length (nm)	Resistance (Ω)	Drive Current (μA)	Final Bit Delay (ps)
CML Latch	2 (Data) 2 (clock) 1 (Bias)	60 (Data) 60 (Clock) 60 (Bias)	700	650	–
CMOS Buffer	15 (Pull-up) 7 (Pull-Down)	60	–	–	10
Fast MUX	2 (Data) 1 (Clock) 2 (Bias)	180 (Data) 60 (Clock) 60 (Bias)	500	600	–
Pre-Driver	7 (Data) 2 (Bias)	60(Data) 60(Bias)	500	500	–

3.2.2 ResultsoftheImplementedCircuits

CML Latch

Figure 20 below shows the schematic of the CML latch. As shown in Figure 21, the designed latch has a small signal gain of 1.93 V/V during the transparent mode (when it follows the inputs while the clock is high) and a gain of 3.98 V/V during regeneration (when it holds the previous output until the next clock edge). This ensures that the subsequent stages will have enough gain to capture the delayed bit. Moreover, in large signal the latch has enough swing to be able to pass through the minimum voltage decision threshold.

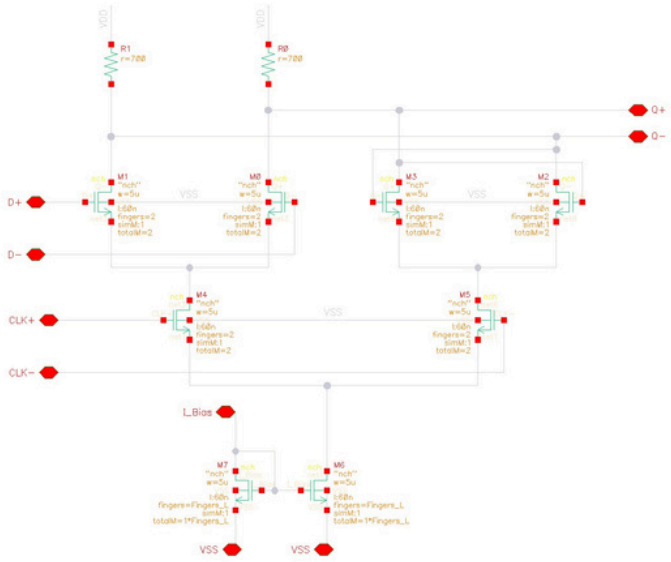


Figure 20: CML latch schematic

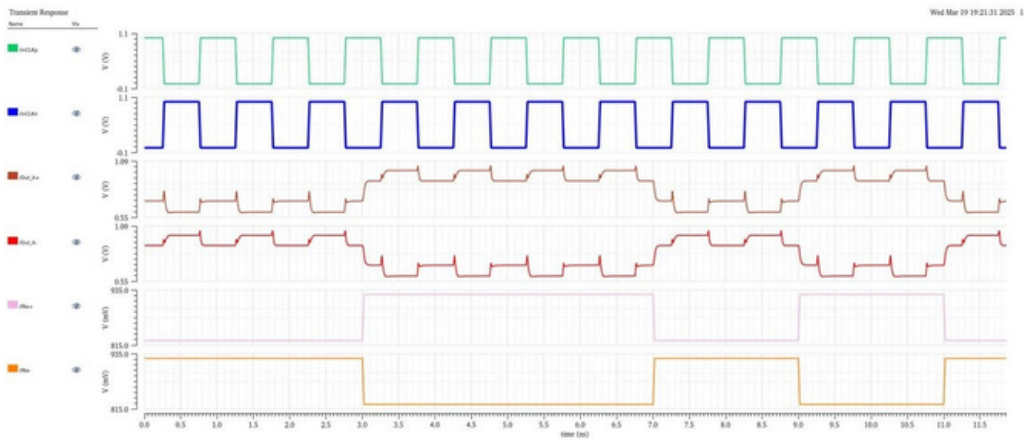


Figure 21: Gain analysis of the CML Level Sensitive Latch

CML 2-to-1 MUX

The primary function of the high-speed MUX is to keep up with the speed of the latch without affecting the output swing. Moreover, it must maintain the integrity of the signal to minimize ISI (and so BER) while transferring it to the next stage.

Figure 22 demonstrates the implemented MUX, which is similar in design to the CML latch. The main difference is the two differential pairs that are switched based on the clocking pair: if the clock is high, it drives the current into the left branch, and if it is low, it drives the current into the right branch. Moreover, we observe a 1 UI delay when connecting the MUX and the latch together. Therefore, the delay required for equalization is achieved (details follow in Section 3.3).

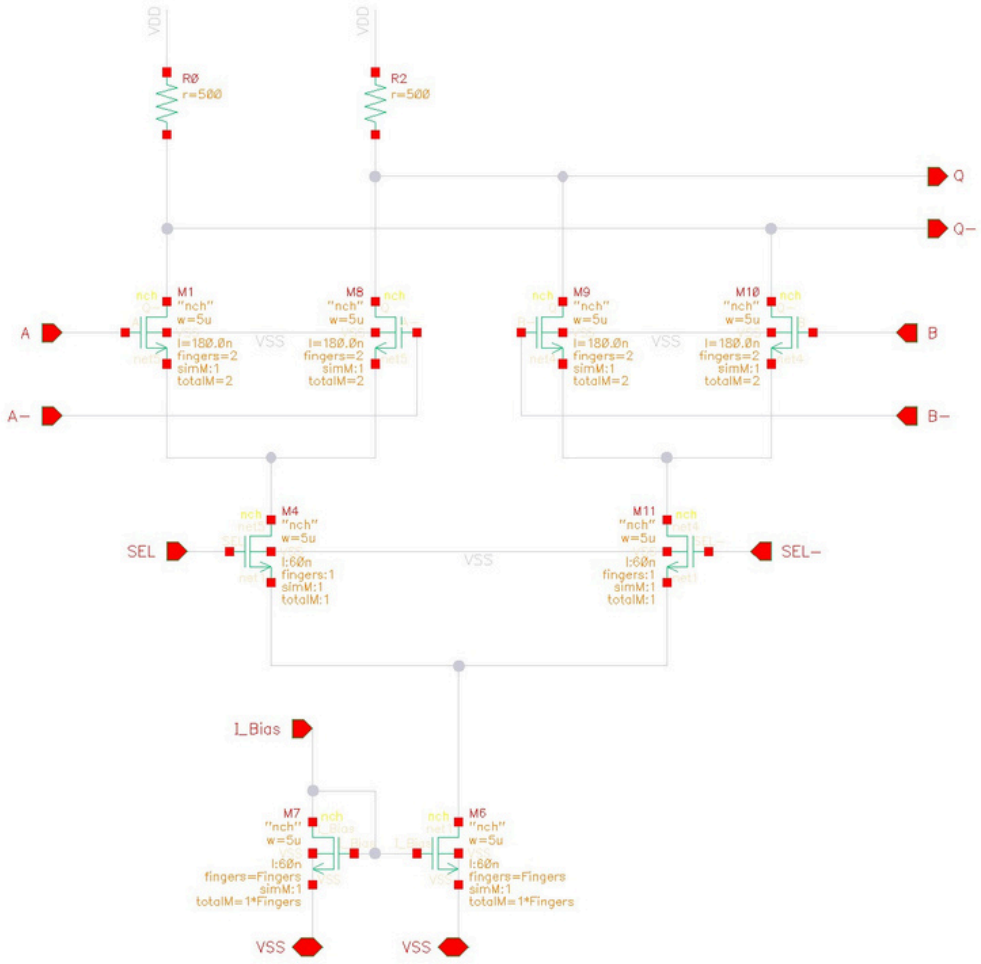


Figure 22: CML 2-to-1 MUX Schematic

Figure 23 shows the 1 UI delay observed at each stage. However, charge injection transients occur when the clock switches, an issue encountered frequently during the design phase of the MUX. In Section 3.2.1, this issue was identified as being caused by a setup time violation. The following section discusses the resolution of this issue with the use of a CMOS buffer.

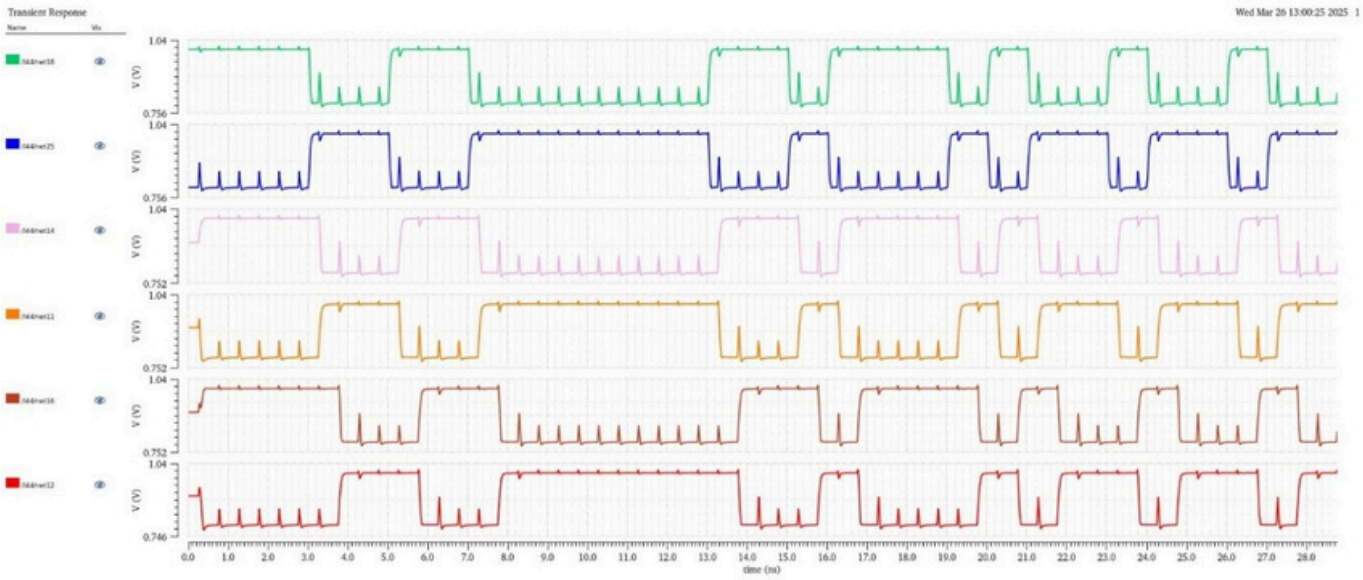


Figure 23: Outputs of the MUX at Low Data Rate to Demonstrate Functionality

CMOS Buffer

The buffer was designed to introduce a delay of 10 ps, which is approximately 10% of the highest frequency (8 GHz) clock period. This delay was chosen to provide sufficient margin to combat setup time violations without excessively impacting the overall timing budget. As shown in Figure 24, the 10 ps delay was initially modeled using an ideal delay block to verify its effectiveness before designing the CMOS buffer.

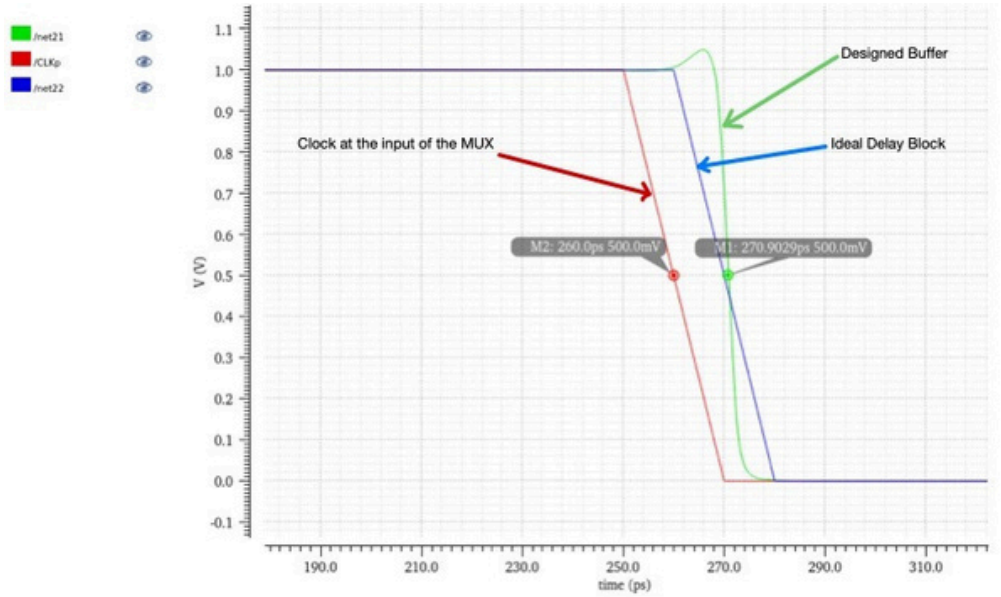


Figure 24: Transient Analysis of the Designed vs Ideal Delay block

Figure 25 shows the implementation of the CMOS buffer, a standard design constructed by combining two inverters. Simulation results (Figure 26) were carried out to compare the performance of the CMOS buffer with an ideal delay block referenced to the system clock; at the decision threshold, the desired 10 ps delay was achieved.

Figure 25: CMOS Buffer Circuit Schematic

The implemented delay allowed the group to solve the issue observed for the bigger spikes in Figure 19 in section 3.2.1. Further improvements downstream will continue to clean up this signal; see Section 3.3.

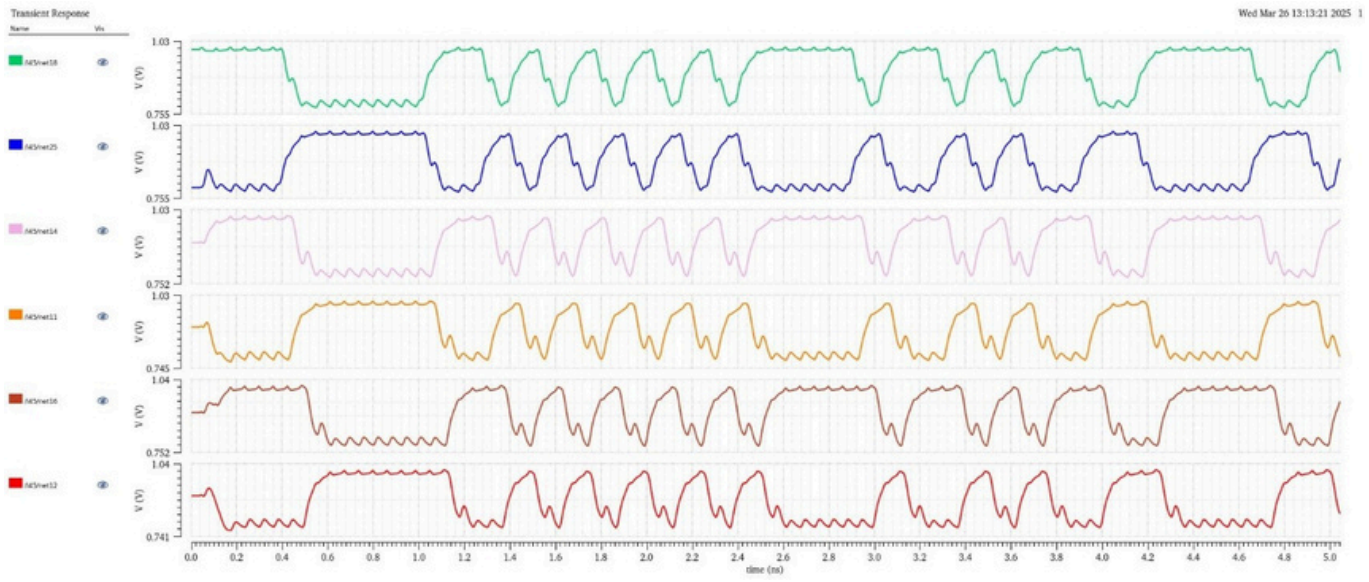


Figure 26: Transient analysis of CMOS buffer

CML Pre-Driver

The CML pre-driver is essentially a buffer that comes before the main drivers. Its circuit schematic (Figure 27) shows that it consists largely of a differential pair with a load sufficient enough to minimize interference with the RC constant.

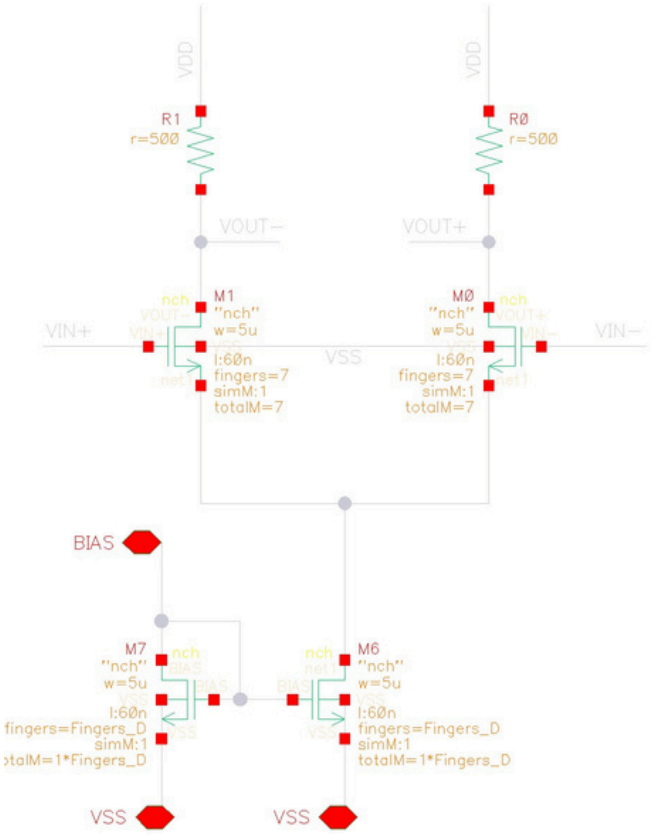


Figure 27: CML Pre-Driver Circuit Schematic

The importance of pre-driver design lies in the ability to alleviate the difference in capacitance between the main drivers and the decision circuits. Hence, the number of fingers used was an in-

termediate value between the 2-to-1 MUX and the main driver. Moreover, it also helps to improve signal integrity (see Figure 28). The output of the pre-driver does not exhibit the undesirable large transients at the edges of the clock.

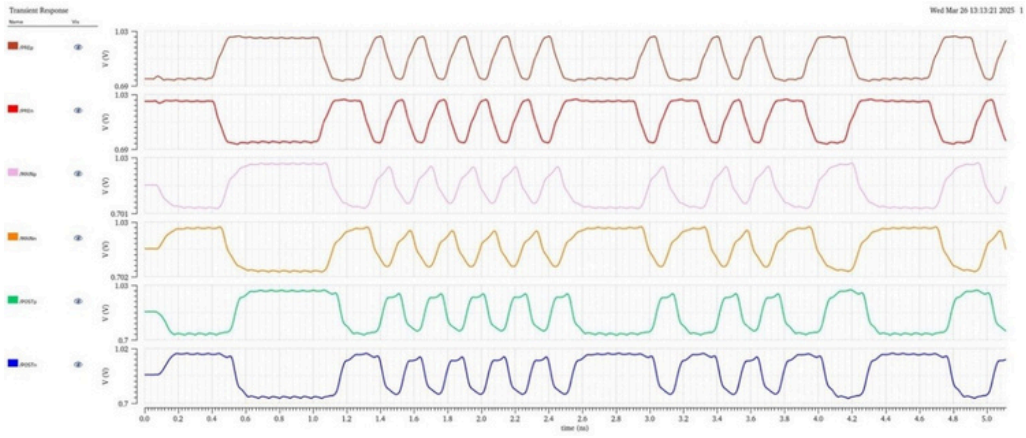


Figure 28: Output of the Pre-Driver with Delay Buffer at 16 Gbps

3.3 Output Driver & FFE Taps

3.3.1 CircuitTopOverview

The output driver and the FFE taps form the final (channel-facing) subcomponent in the transmitter. Circuit design was informed largely by the conventions discussed in Sections 1.4.3.1 and Section 2.3, with the aim of maintaining sufficient signal swing from the high-speed serializer to ensure that the capacitance of the pre-driver remains manageable.

3.3.2 Component Parameters

Table 4: Driver/FFE Component Parameters

Component	Transistor Fingers	Transistor Length (nm)	Resistance (Ω)	Drive Current (mA)
Main Driver	12 (Data)	60 (Data)	50	20
FFETaps	12 (Data)	60 (Data)	–	–

3.3.3 ResultsoftheImplementedCircuit

The design of the main driver and the FFE taps corresponds with the theory in Section 1.4.3.1. Hence, the schematic is quite simple: the main tap (Figure 29) is the main driver, with a load resistance chosen to be 50 Ω (to ensure impedance matching with the channel and receiver, thereby eliminating reflections from the channel). Moreover, the output driver has a larger gain to launch into the channel without loss of gain or swing. As a result, a gain of 4 dB was obtained with a large bandwidth seen in the AC analysis (Figure 30). Furthermore, the FFE taps (Figure 31) have the same sizing as the main driver; however, they have no load resistance, and their output polarity is inverted to effect current subtraction.

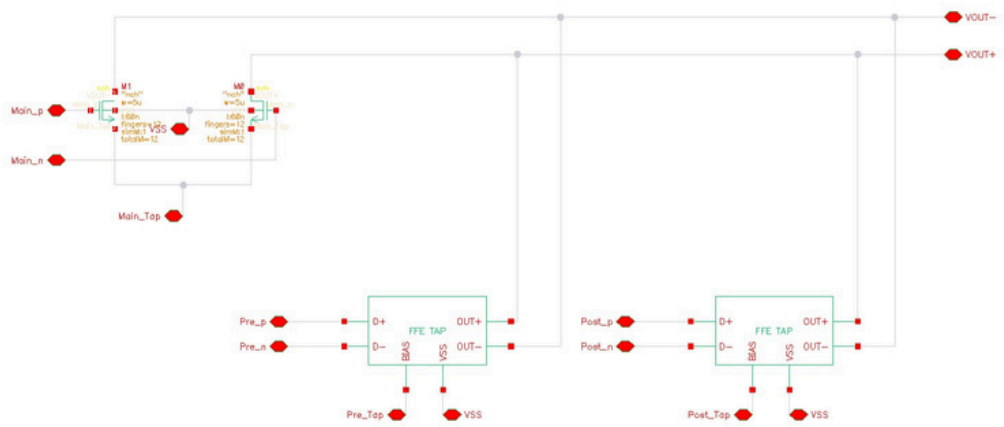


Figure 29: Main/Launch Driver with FFE taps Connected

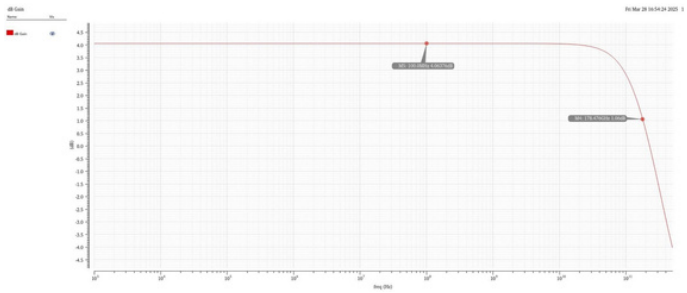


Figure 30: AC Analysis to Obtained the Gain of the Main Driver

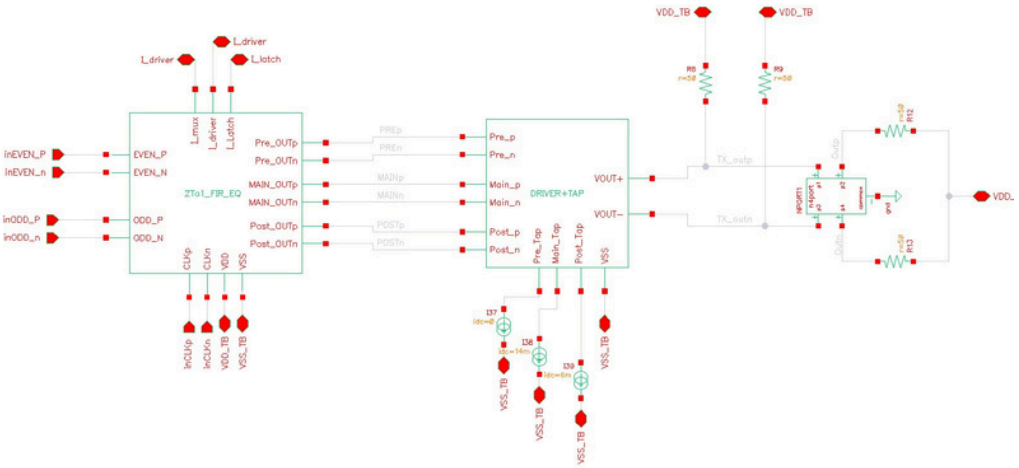


Figure 31: High-Speed Serializer Connected to the Main-Driver & FFE Taps with Transmission Line

3.4 Other Circuitry

Current Mirror Network: Isolation testing demonstrated a need for a variety of bias currents across the different stages of the transmitter. These currents ranged from 100μA to 2mA; details can be found in the respective subcomponent sections above. As simulations moved to higher levels of hierarchy, advanced-stage simulation results included removal of ideal current sources and installation of a simple branched current mirror driven by a single 1mA source. Each branch of

the abstracted current mirror network consisted of a PMOS/NMOS pair for scaling the magnitude of the branch current, while the biasing devices in each subcomponent were terminated with an additional PMOS (referred to as a “header”; see Figure 32). The header serves two purposes: first, it allows each current mirror branch in the network to draw current the same direction (relative to the rail voltages) regardless of topology, simplifying high-level design choices with respect to the supply voltage grid. Secondly, the headers were kept at the second-highest level of hierarchy during penultimate integration testing. This permitted the group to retain an easily accessible control parameter for tuning branch currents during this crucial verification phase.

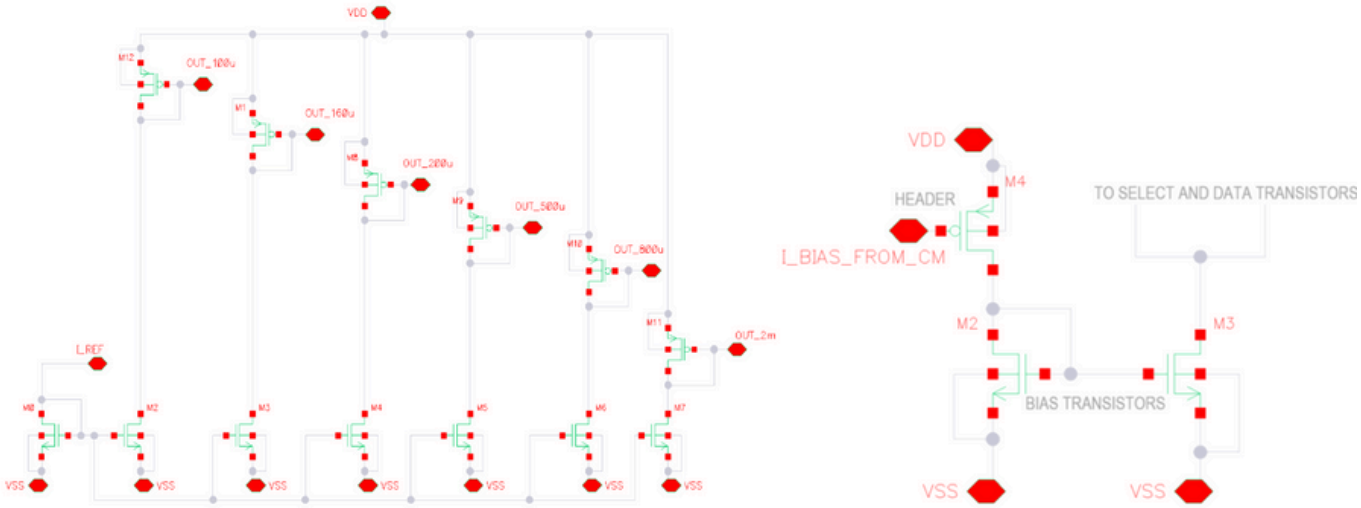


Figure 32: Current network backbone (left); header and bias transistors on example device (right)

Table 5: Current Mirror Transistor Parameters

Device	Branch Current (µA)	Branch NMOS	Branch PMOS	Header PMOS
LIDO Converter	100	5µm/110nm	15µm/180nm	15µm/170nm
Stage 1 MUX	200	10µm/110nm	10µm/120nm	10µm/120nm
Stage 1 MUX Delay Element	200	10µm/110nm	10µm/120nm	10µm/120nm
Stage 2 MUX	2000	10µm/110nm	10µm/120nm	100µm/120nm
Stage 2 MUX Delay Element	160	10µm/110nm	15µm/120nm	20µm/220nm
Stage 3 MUX	50	15µm/110nm	5µm/120nm	10µm/130nm
Stage 3 Latch	0	15µm/110nm	5µm/130nm	15µm/130nm
Stage 3 Pre-Driver	80 0 50 0	15µm/110nm	5µm/120nm	10µm/135nm

Note: Bandgap Reference Source-facing NMOS is 100µm / 120nm.

Clocking Network: Similar to the current network, different parts of the transmitter require different clock subdomains (see Section 3.1.4 for additional details). The master clock domain consists

of two opposite-phase 8 GHz clocks. These are used directly in the half-rate serializer at full output (16 Gbps), and divided by a factor of 2 for slow output (8 Gbps). This is accomplished with a frequency divider, which feeds both clock speeds into a multiplexer used as a selection device (Figure 33). In a similar fashion, progressively slower clock domains are used for driving progressively slower circuit components. Each is selected with the same user-programmable select signal to toggle overall transmission rate.

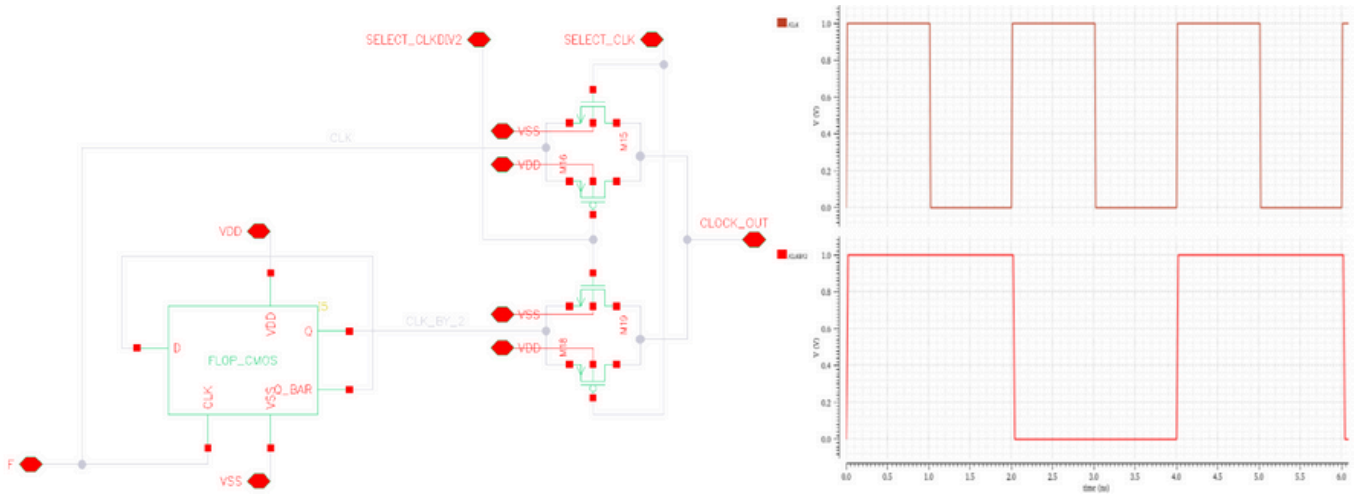


Figure 33: Frequency divider and MUX as selection device (left); original and divided clock signals (right)

While the group was unable to tune the complete network for proper operation before the fi- nal deadline, the experimenters remain confident in the theoretical design and assert that the network could be optimized with a reasonable amount of additional testing.

Further, discussions with the supervisors yielded insight about additional complications that would inevitably arise when implementing a non-ideal clocking network. The single most pressing concern involves an extension of what are labelled “D/A charge injection transients” (see Section 4). These are undesirable artifacts seen in the output of the datapath MUXes that have a chance of occurring every fourth bit, when the edges of the MUX clocks and the data clocks are aligned. While this was handled in the current design with targeted delay elements (Section 3.1.3), the artifacts are expected to extend to all bit lines with the introduction of jitter and nonuniform clock rise/fall times. Quashing the transients systematically will require a more sophisticated delay scheme; one such proposed scheme involves pair-wise delay of data elements such that e.g., in the 4-to-1 MUX, only two elements are held at a time per clock phase while the other two are permitted to setup. This is in contrast to the existing scheme, where each successive
bit is setup and held in a ”rolling” rotation and only the final bit is delayed. Tuning delay elements to effect this alteration should theoretically address many issues that would come about when the transmitter operates with fully non-ideal clocking (master subdomain notwithstanding).

4 TEST RESULTS

4.1 High-level Transmitter

4.1.1 OverallTXDataPathResultforB1ChannelModel(W.Peters,Intel) The overall TX output produces an eye diagram at 8 Gbps with very high BER and a low voltage swing. However, the VEO taps, and intrinsic jitter improved with the addition FFE taps. The issue might be caused by the lower path of the transmitter, which already had poor SNR and low swing. As a result, at the final stage of the TX, the equalizer has to work harder to achieve a better VEO.

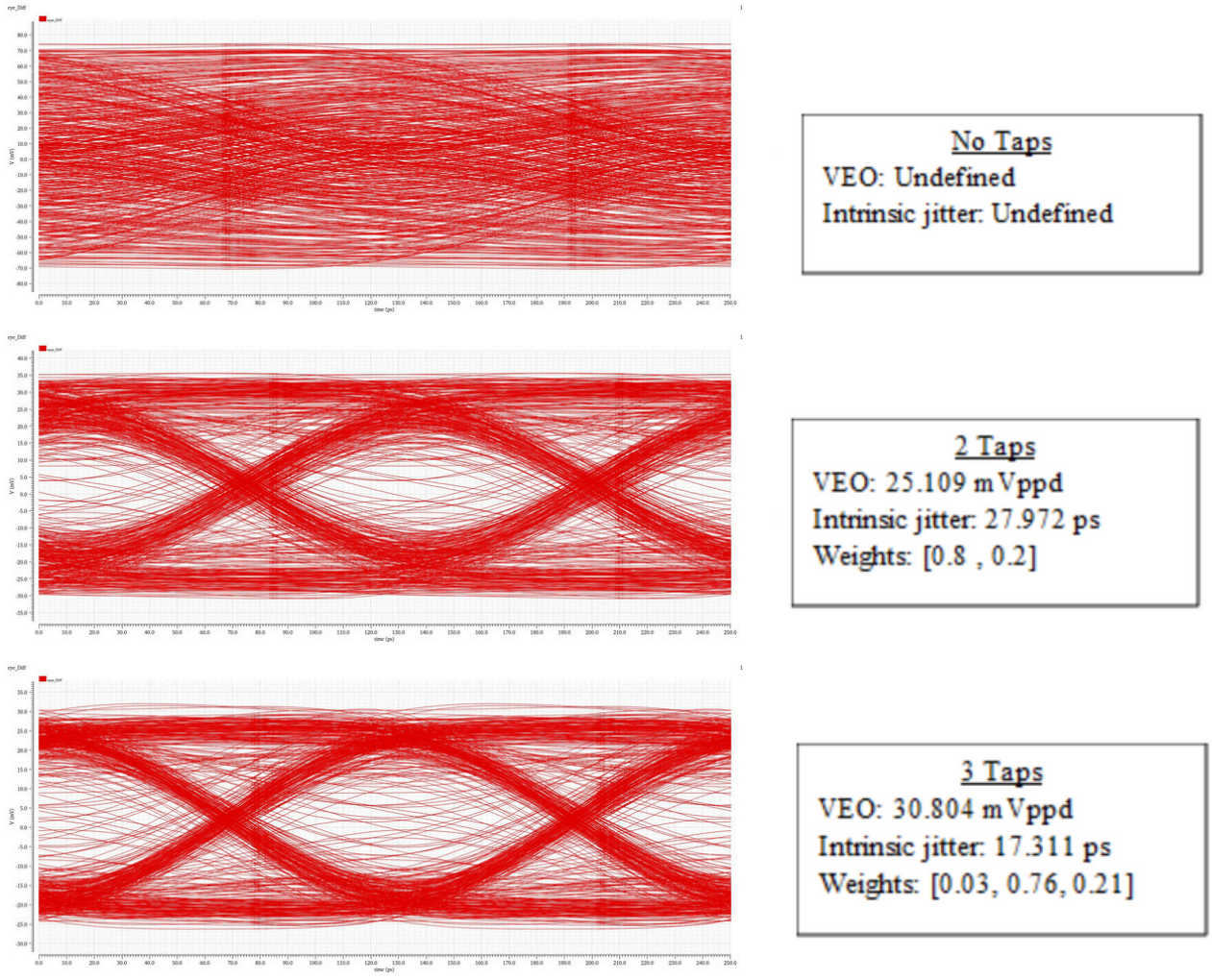


Figure 34: Eye Diagrams of the Overall TX Datapath at 8 Gbps

4.1.2 OverallTXDataPathResultforP2TX2ChannelModel(W.Peters,Intel) At 16 Gbps, the TX performance begins to break down. The BER increases so that even with EQ, the VEO decreases until the eye is almost completely closed. More than 50% of overall swing is also lost. As mentioned previously, the cause is likely related to timing constraints discussed in more detail in Section 4.2.

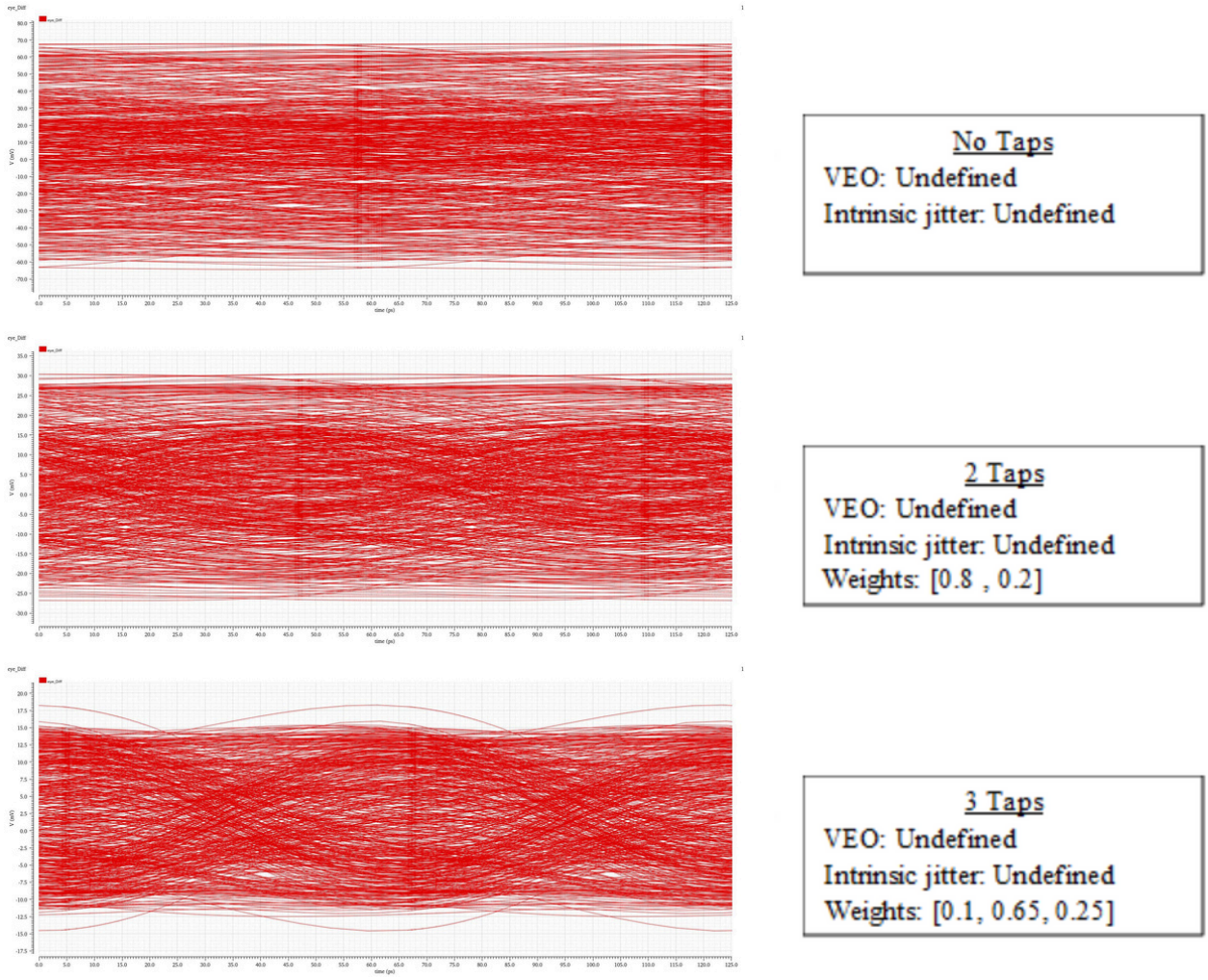


Figure 35: Eye Diagrams of the Overall TX Datapath at 16 Gbps

4.2 Pre-serializer

Stage 1 Multiplexer - Figure 36: "Slow" stage MUX operating at 4x500 MHz in / 2Gbps out or 4x250 MHz in / 1Gbps out.

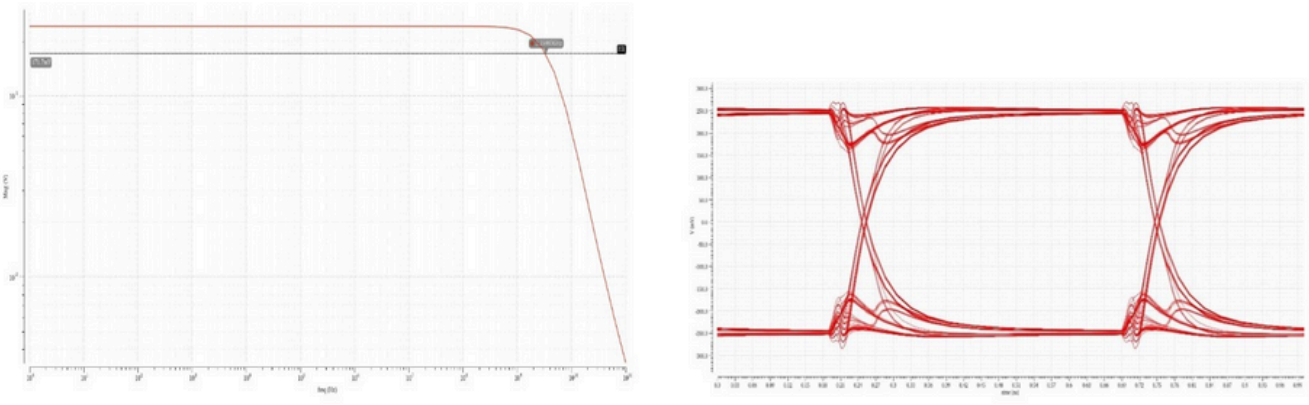


Figure 36: Performance metrics for Stage 1 pre-serializer multiplexer

Left: Frequency response; f−3*dB*=3.4GHz. Right:Eyediagramofoutput

VerticalEye=474mVppd; HorizontalEye=0.49UI; DJ=0.012UI.

Stage 2 Multiplexer - Figure 37: “Medium” stage MUX operating at 4x2 GHz in / 8 Gbps out or 4x1 GHz in / 4 Gbps out.

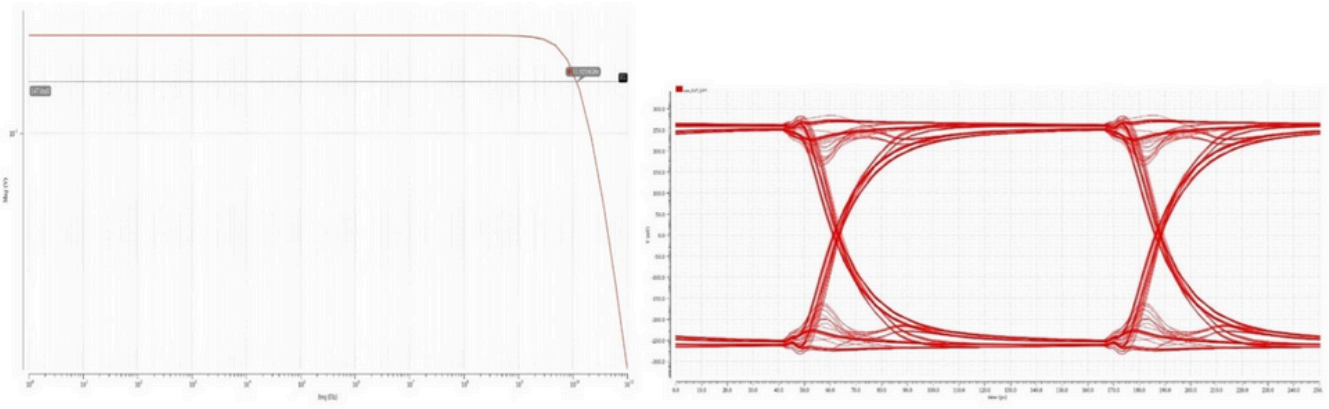


Figure 37: Performance metrics for Stage 2 pre-serializer multiplexer

Left: Frequency response; f−3*dB*=11.9GHz. Right:Eyediagramofoutput

VerticalEye=470mVppd; HorizontalEye=0.484UI; DJ=0.016UI.

Delay Elements - Figure 38: As discussed, buffers were introduced in the datapath to delay data bits set to transition at the same time as the clock. Depending on the input state, omitting these elements sometimes resulted in artifacts in the output signal large enough to cross the decision threshold, implying an increased BER in these transitions. Since this phenomenon happens at the MUX when inputs change from BIT_D to BIT_A, these artifacts were referred to as “D/A charge injection transients”. Attributed to hold time violations, the issue is largely mitigated by allowing the previous data bit to hold its value during the clock transition period through buffering rather than flipping immediately.

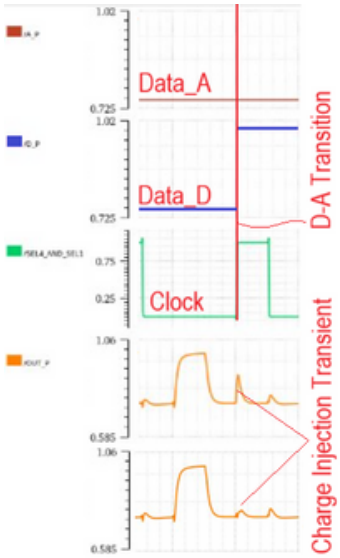


Figure 38: Effect of delay elements on MUX output D/A charge injection transients

High-level Pre-serializer Module: Transient analysis was carried out on the complete pre-serializer module to observe the device’s capability to pass data to the high-speed stage. In the 8 Gbps scheme, the differential output swing was realized within 2% of its target value, though the common-mode voltage level was pulled down by about 2 mV (Figure 39), suggesting current

overdraw in the overall module. This aligns with the design methodology, which uses high resistance in the slow stages to realize the target swing while the fast stages utilize high current, a trend that carries over into the high-speed stage. The result is an eye diagram that, while open, exhibits significant ISI and DJ (Figure 40). Tuning this resistor-current balance with more precision would likely help address these issues.

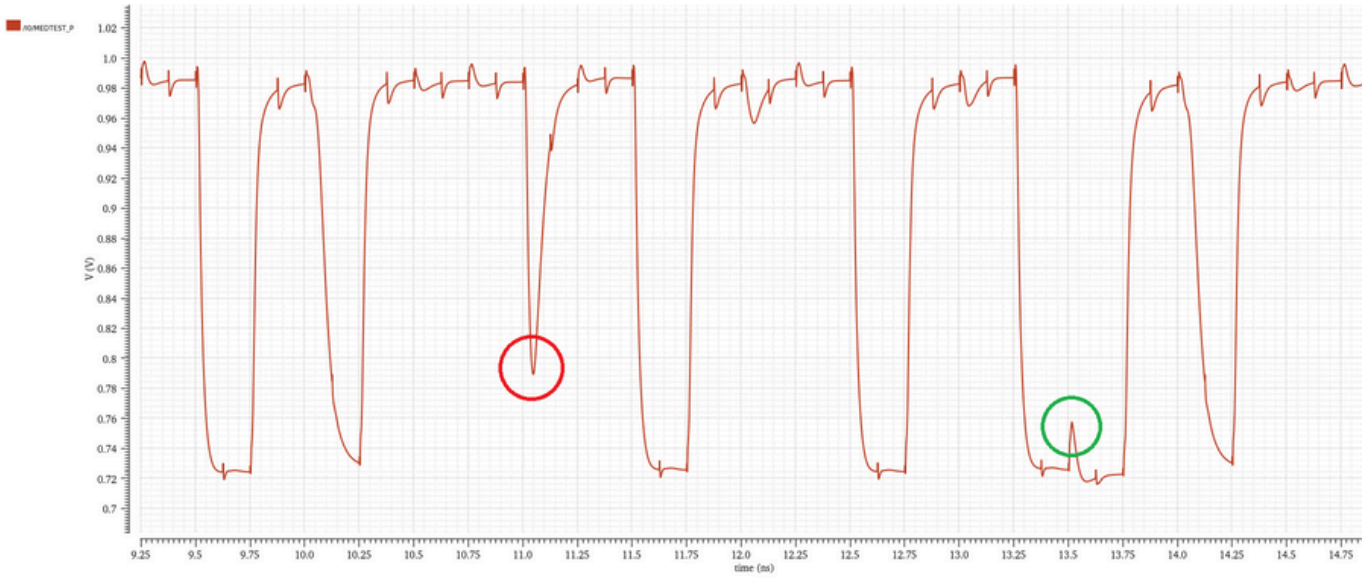


Figure 39: Transient analysis of overall pre-serializer output at 8 Gbps.

Red: New-word charge-injection transient at a major clock transition.

Green: Mid-word transient between clock transitions.

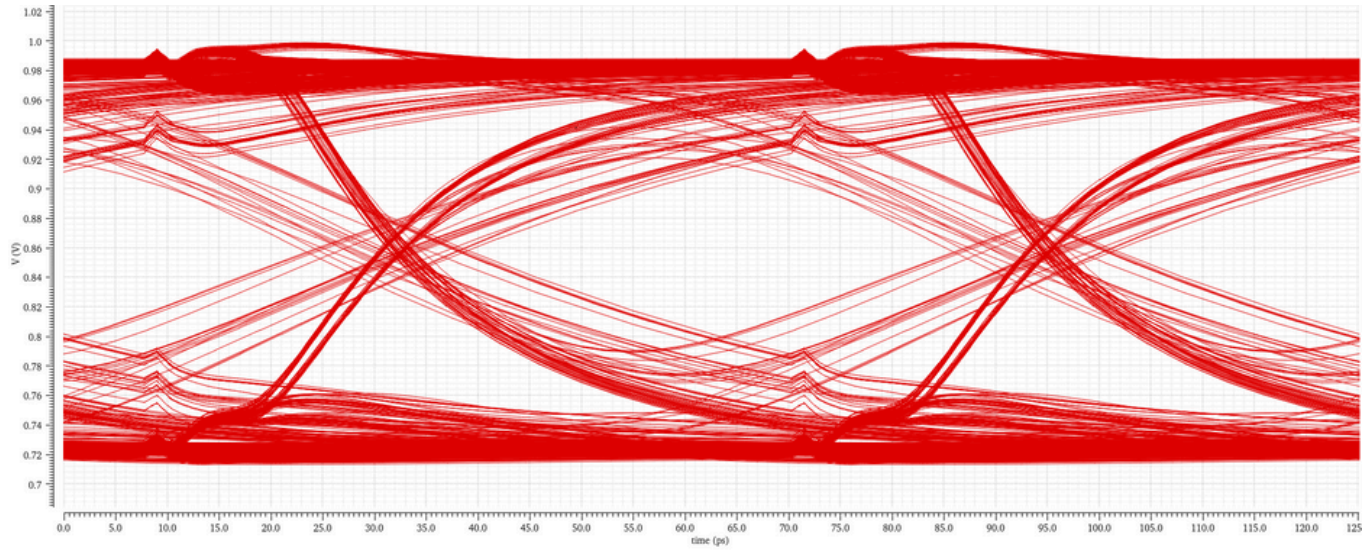


Figure 40: Eye diagram of overall pre-serializer output; 8 Gbps

VerticalEye=232mVppd; HorizontalEye=0.38UI; DJ=0.128UI.

Similar testing was carried out at the max output rate of 16 Gbps. In this case the differential output swing was still realized within 5% of its target value but was in fact larger (not smaller) than the target values. The common-mode voltage level decrease is also slightly less (Figure 41). The reader will notice that the charge injection transients are larger at these higher speeds, and the BER has also visibly increased due e.g., to reduced responsivity during alternating bits. Thus,

the eye diagram shows ballooning ISI and DJ compared to the 8 Gbps case (Figure 42), with transitions that appear increasingly disjointed. Tuning methods similar to those of the 8 Gbps case would likely assist in improving the response at this stage.

Figure 41: Transient analysis of overall pre-serializer output; 16 Gbps

Green: Alternating bits error. Black: New word (D/A-type) charge injection transient.

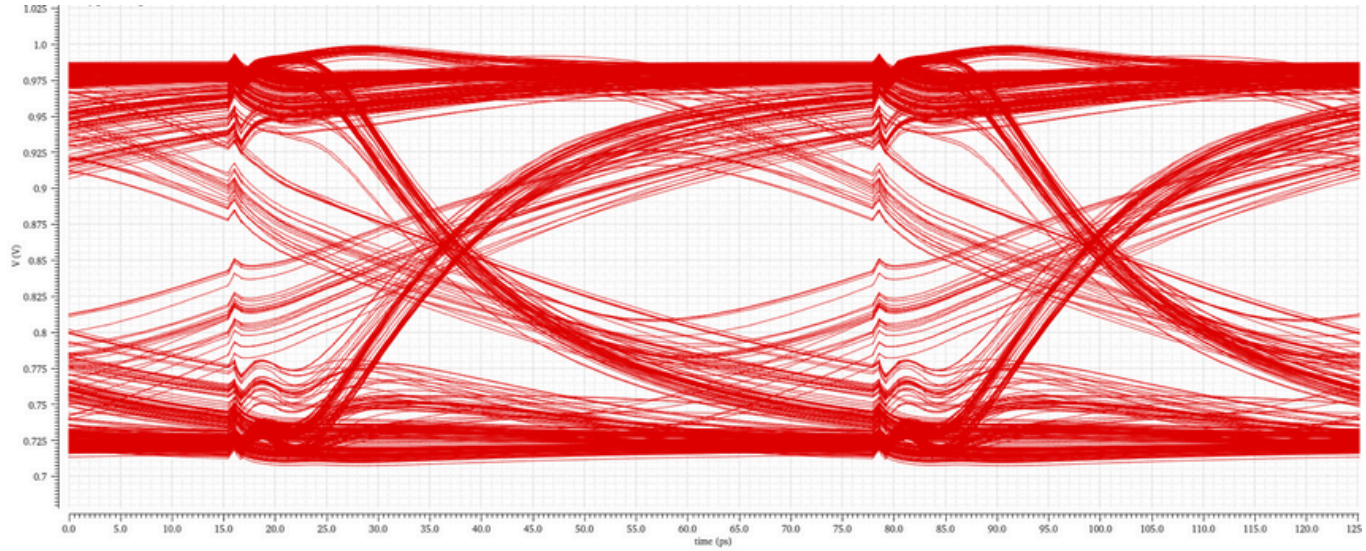


Figure 42: Eye diagram of overall preserializer output; 16 Gbps

VerticalEye=200mVppd; HorizontalEye=0.35UI; DJ=0.130UI.

4.3 High-speed Serializer

Figure 43 shows the eye diagram obtained at the main pre-driver's output for the maximum data rate. Importantly, the assigned swing of 500 mVppd was maintained. Additionally, the transition between constant one and zero, as measured by the intrinsic jitter, is short and operates fast enough at this data rate for the desired operation.

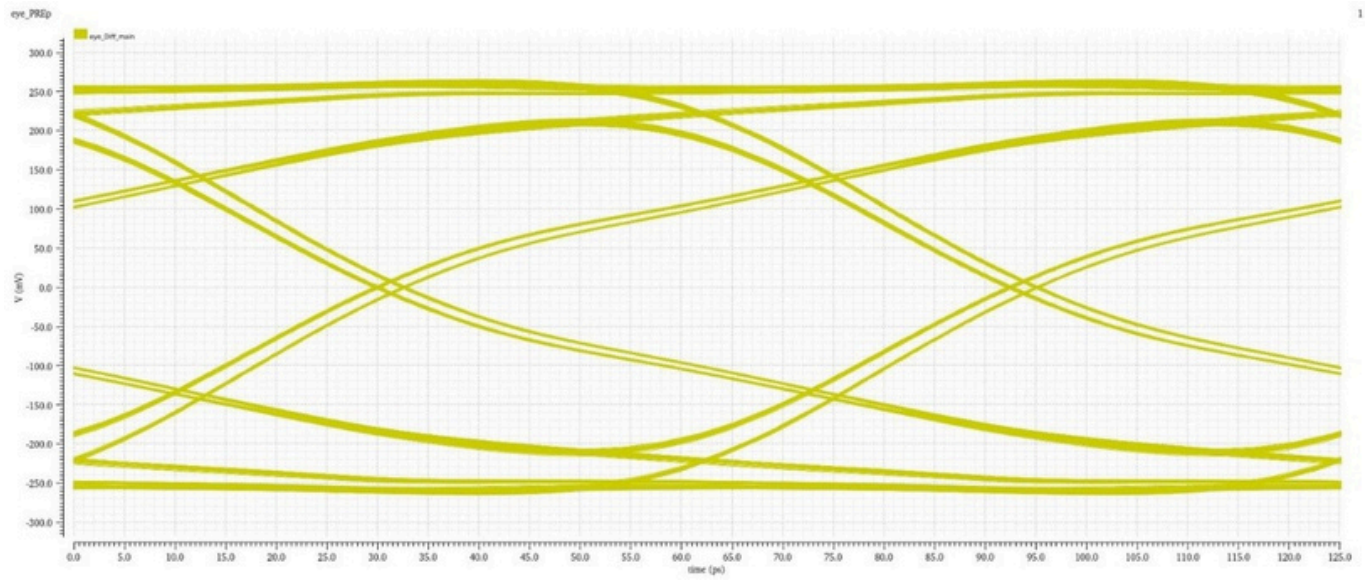


Figure 43: Eye Diagram of the Output of Main Branch at 16 Gbps

4.4 Equalizer and Driver

Figure 44 shows the eye diagram at the output of the channel. It is like that seen in Section 1.1.3, which is expected as the output of the transmitter only amplifies the signal that will be transmitted through the chosen channel. In this case, the swing was increased slightly so that the attenuation at the output of the channel is somewhat mitigated.

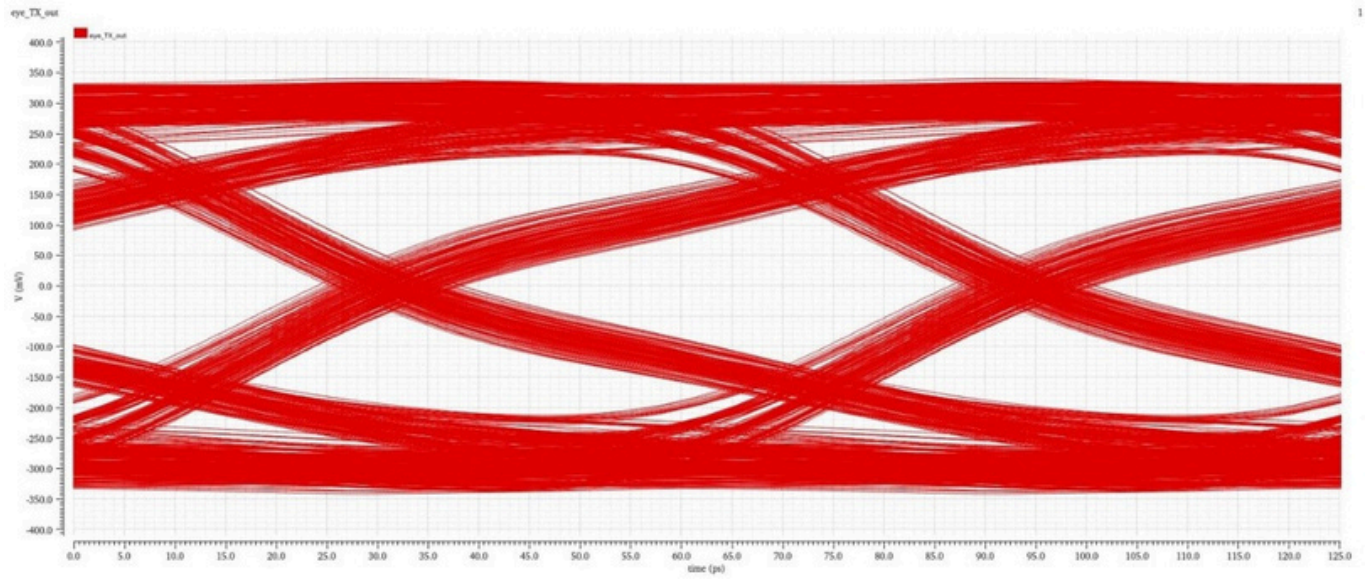


Figure 44: Eye Diagram at the Output of the Transmitter at 16 Gbps

B1 Channel Model (W. Peters, Intel)

To determine the FIR, tap weights, an impulse response (isolated bit) was generated at the target data rate (8 Gbps) in a process similar to that demonstrated in Section 1.6.3. An equalized eye diagram was then obtained at the channel output for 8 Gbps using both two-tap and three-tap configurations (Figure 45). While adding a third tap provided a marginal improvement, it was observed that a two-tap configuration gave sufficient VEO and acceptable intrinsic jitter while using this channel.

Table 6: FIR tap weights with VEO & intrinsic jitter for B1 channel

	Pre-Tap	Main-Tap	Post-Tap	VEO (mVppd)	Intrinsic jitter (ps)
0	0	0	0	117.116	43.192
Taps	0	0.8	0.2	141.685	21.481
2	0.03	0.76	0.21	145.305	16.754

Taps

3

Taps

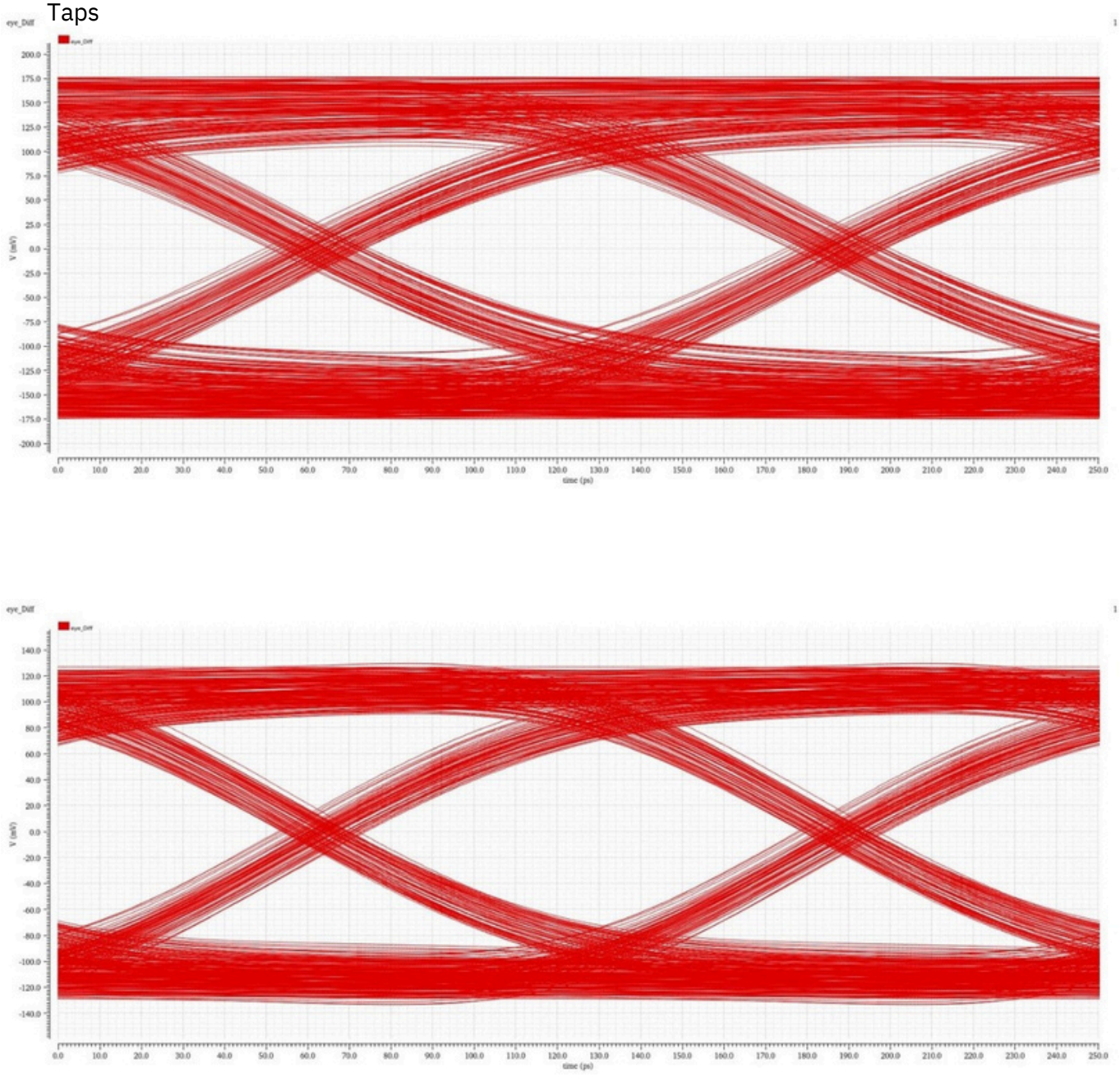


Figure 45: B1 Channel Eye Diagram Results with 2 Taps (top) and 3 Taps (bottom) FIR Equalization

P2TX2 Channel Model (E. Matoglu, Amphenol)

Compared to the previous channel (tested at 8 Gbps), this channel exhibits significantly higher loss at 16 Gbps; recall Figure 4 (Section 1) that showed negligible VEO and oppressive jitter in this instance. Equalization brings about considerable improvement in both metrics (Figure 46). Compared to the B1 channel model, the benefits of using three taps of FFE are much more apparent. Specifically, VEO improved by 26% compared to using only two taps, and intrinsic jitter was reduced by half.

Table 7: FIR tap weights with VEO & intrinsic jitter for P2TX2 Channel

	Pre-Tap	Main-Tap	Post-Tap	VEO (mVppd)	Intrinsic jitter (ps)
0 taps	0	0	0	Undefined	Undefined
2 Taps	0	0.8	0.2	60.449	30.67642
3 Taps	0.1	0.65	0.25	81.685	14.652

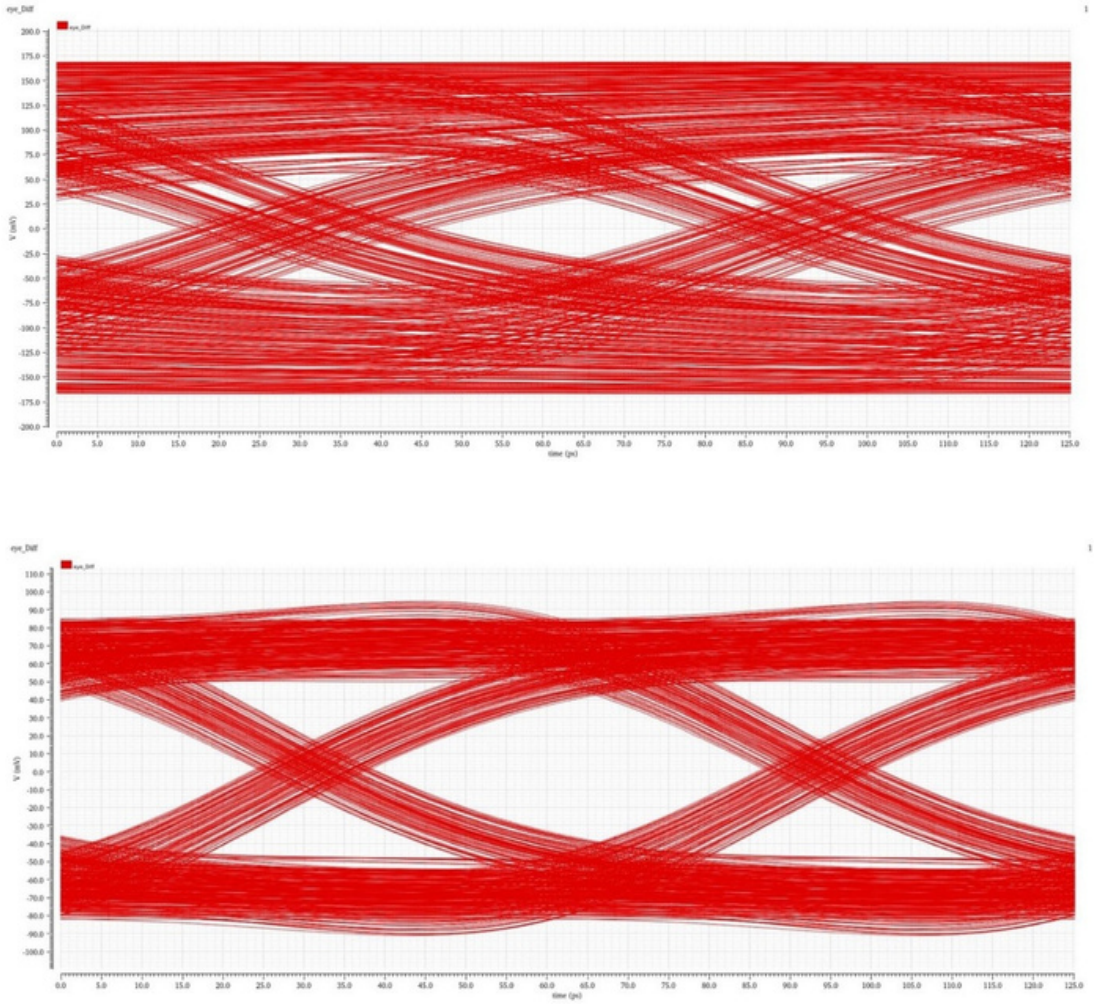


Figure 46: P2TX2 Channel Eye Diagram Results with 2 Taps (top) and 3 Taps (bottom) FIR Equalization

5 PROJECT MANAGEMENT

Effective project management is essential to ensure that the Wireline Transmitter project progresses efficiently, meets its objectives, and delivers high-quality results. A structured management approach is followed to define goals, develop the transmitter in phases, and maintain strong collaboration among team members and supervisors.

5.1 ManagementPhilosophy

- Clear Objectives

  A well-defined and measurable goal ensures that the project stays on track and delivers meaningful outcomes. The objective is to design a high-speed Wireline Transmitter with NRZ signaling, FIR-based equalization, and impedance control to optimize signal integrity. The goal should align with the expectations of stakeholders, including project supervisors and industry standards. Regular evaluations ensure that progress remains aligned with the project scope and technical specifications.

- Agile and Iterative Development

  The project follows an agile and iterative development process, allowing flexibility in design refinements based on testing and feedback. The development is structured into four key phases:

  Phase1: *ConceptandSimulation*–Initialresearch,modeling,andverificationofsystem-level behavior through MATLAB.

  Phase2: *TransmitterCircuitImplementationonCadence*–Schematicdesignandtransistor-level implementation of the transmitter using Cadence tools.

  Phase3: *TestingandValidation*–SimulationofthedesignedcircuitsunderdifferentPVT (Process, Voltage, Temperature) conditions to ensure performance robustness.

  Phase4: *RefinementandFinalDesign*–Optimizationofcircuitparameters,finallayout generation.

- Team Collaboration and supervisor feedback

  Effective collaboration among team members and regular supervisor feedback play a crucial role in the project's success. Tools such as shared file repositories (e.g., Google Drive), weekly meetings, and structured progress reports help streamline communication and ensure that everyone is aligned on the next steps. Supervisor input is integrated into design iterations, ensuring adherence to technical requirements and academic expectations.

- Project Documentation

  Comprehensive project documentation is maintained throughout all phases to ensure clarity and traceability. This includes:

  – Design Specifications

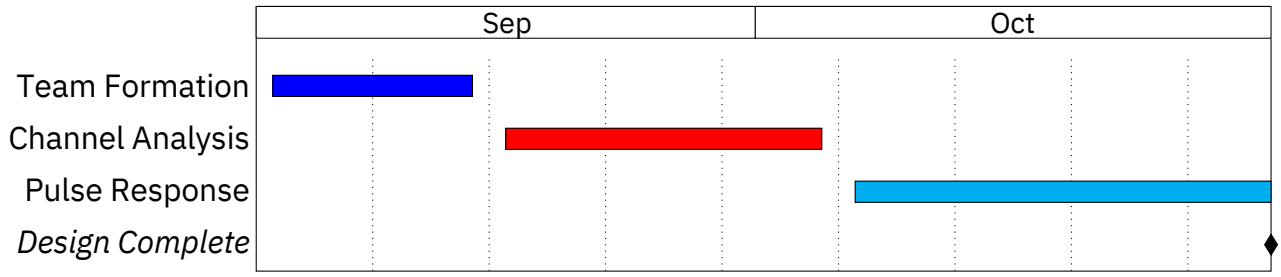
  – Detailed descriptions of circuit components, performance targets, and expected outcomes.

  – Simulation Reports

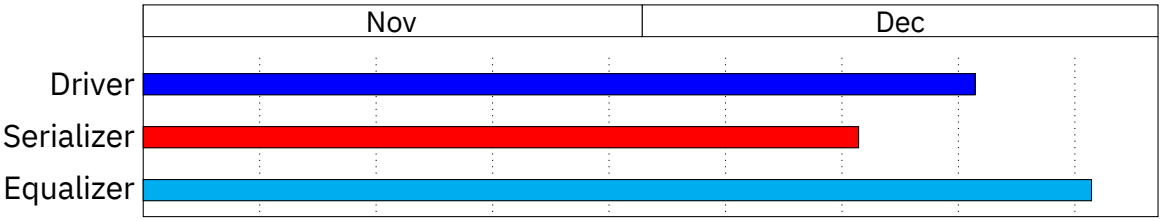
  – Testing results

5.2 GanttChart

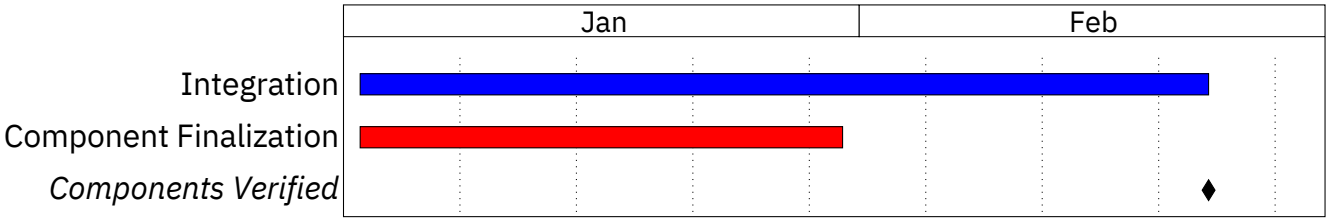
Prelim-Phase: September-October 2024



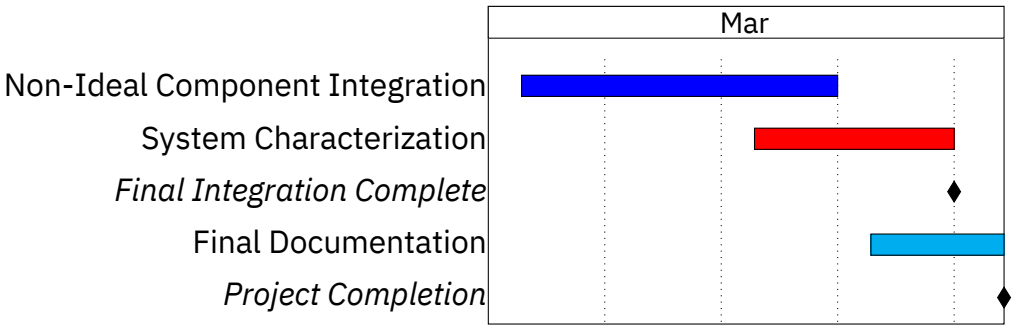
Phase 1: November-December 2024



Phase 2: January-February 2025



Phase 3: March 2025



5.3 ReasonsfordeviationsfromPhase1andPhase2schedules

• *Pre-serializer Timing Issues:* – The 4-to-1 MUX design suffered from hold-time violations and charge injection transients, delaying integration testing by 3 weeks. We resolved this by adding precision delay elements (60ps for Stage 1, 20ps for Stage 2).

• *High-Speed Serializer Violations:* – Setup time violations in the 2-to-1 MUX at 16Gbps required 2 additional weeks for debugging. The solution was implementing a 10ps CMOS buffer in the clock path.

• *FIR Equalization Tuning:* – MATLAB-based tap weight optimization took longer than expected due to channel model complexity, delaying driver testing by 1 week. We automated the tap-weight calculation script to accelerate iterations.

• *PVT Sensitivity:* – Late-stage process variation simulations revealed layout sensitivities, postponing final verification by 2 weeks. We addressed this by increasing transistor finger widths in critical paths.

• *Failure to meet specifications:* – The report outlines certain specifications that were unable to be met. Due to delays in the design process, technical difficulties, setbacks in design iteration, and unforeseen prolonging of certain test batteries, a transmitter with 100% feature-completion remained unrealized.

5.4 Lessons learned from experimentation and how we applied them

Table 8: Key Experimental Lessons and Applications

IssueEncountered	Application/Improvement
Clockskewinserializerstages	Implementedbalancedclocktreewith matched buffers
Excessivejitterat16Gbpsoperation	Optimizedpre-driverbiasingtoreduce rise/fall time asymmetry
Poor vertical eye opening in unequalized mode	Added programmable 3-tap FFE for channeladaptation
Current mirror instability under PVT variations	Redesigned for better tolerance by choosing suitablebandgapreferenceandtuningdistribution network
Charge injection transients in MUX outputs	Added targeted delay elements in data path
Setup time violations in high-speed latch	Introduced CMOS clock buffers for improved timing margin

5.5 Lessons learned from feedback and how we applied them

Table 9: Feedback Implementation Results

FeedbackReceived	ImplementationResult
The assumptions made in the clocking net- workaretooidealistic	Added jitter injection in simulations, re- vealingneedforDCCandmorerobustdata delay scheme
TheFFEtapslackprogrammability	Designed3-tapFIRwithadjustableweights via secondary control bus
Output driver impedance mismatch is caus- ingundesiredarifactsintheoutput	Implemented 50Ω termination calibration networkasstartingpointandexpandedde- sign therefrom
Excessive power is being used in the datap- athandintheCMLbuffers	Optimized transistor sizing while main- tainingspeed;balancedhighercurrentsin high-speed stages with lower resistances, and vice-verse in low-speed stages
Testing and verification coverage is insuffi- cient	Adjusted verification methodology to be more”bottom-up”hierarchical
Late equalization tuning is slowing design progress	Automated MATLAB tap optimization workflowtooffloadthesetasks

6 SALES PITCH

In the current landscape of high data throughput driven by AI and data centers, the need for a high-performance, power-efficient, and scalable transmitter IP block has never been more critical. Our cutting-edge transmitter IP delivers speeds of up to 16 Gbps, ensuring seamless data transmission while maintaining signal integrity in the most demanding environments.

Designed for next-generation computing, networking, and storage applications, our transmitter IP features advanced equalization techniques, low-latency architecture, and robust jitter performance. Whether you're working with high-speed interconnects, PCIe, Ethernet, or custom SerDes applications, our IP integrates seamlessly into your design and providing exceptional reliability.

The key advantages of our Transmitter IP block are:

1. Seamless Integration — Depending on the application, our designed can be retrofitted into existing designing depending on the user requirement.
2. MultiSupportDataRates—Supportsamaxdatarateof16Gbpswith32-bitparallelwords, with modes for reduced clock speeds and bus sizes.
3. Superior Signal Integrity — Built-in equalization and de-emphasis to mitigate losses over different channel lengths.

In sum, with data bandwidth demands skyrocketing due to AI, cloud computing, and hyperscale data centers, the need for high-speed transmitter is more urgent than ever. Bottlenecks in data movement can cripple performance, impacting everything from real-time analytics to machine learning workloads. Don't let bandwidth limitations freeze your innovation: our 16 Gbps transmitter IP ensures your system is able to support the current and future demands of your business infrastructure and your clients' needs.

7 CONCLUSION

The final transmitter design displayed full-bus width, 8 Gbps operation with reasonable performance metrics. Areas for improvement were identified for improved operation: better matching between serialization stages, a more robust clocking network with properly implemented bit-line delay elements, and more thorough integration testing to ensure low-BER serial output across different operating conditions.

Validation for maximum transmission speed (16 Gbps) showed promising results but suffered too greatly from ISI and jitter on both ends of the channel. While the improvements listed above would also help address this mode of operation indirectly, a more focused approach for optimizing individual subcomponents in the datapath. Both of these schemes would benefit from additional stress testing and careful parametric analysis to ensure the best possible response at the output of each successive stage, before the first instance of channel-facing circuitry.

Reduced-size bus width operation remained an intractable problem at this stage of the design.

Attempts at isolating dedicated circuit paths proved power inefficient, while the addition of specialized counting/reset circuitry was unable to be realized due the late stage of completion in the principal transmission scheme. While the group presented a theoretically sound plan of action to tackle this issue, implementation would require additional time for development.

Likewise, the design philosophy for this transmitter leaves room for a relatively facile layout regimen. With emphasis on symmetrical design and progressively streamlined data/clocking circuitry, the underlying circuitry would respond well to producing layout cells of even the smallest subcomponents. It is especially worth noting that all elements (both implemented and proposed) in the pre-serializer can be built using foundry-provided standard cells. Implementation of a layout version of the transmitter would be time-intensive, of course, although the schematic design protocol has been designed to set up such an endeavour for success.

Finally, the overall circuit proved amenable to non-idealities in the current network, with a fully-implemented on-chip current mirror branching network driven by a reasonable bandgap reference source. Designer-implemented, device-level checks allowed for circuit tuning even at the highest levels of hierarchy. As discussed, the group is confident in its ability to map these successes to a non-ideal clocking network in much the same way. Proof of concept for this task was detailed in terms of methodology and proposed implementation. Potential problems that would arise from removing ideal clocking have been proposed and assessed in terms of their relative impact on signal integrity, especially with respect to the expected change in timing constraints. Changes to the data delay scheme currently implemented were proposed to combat these problems while maintaining control over those issues currently being addressed in the ideal case.

8 REFERENCES

[1] I. . AP. Channel model for ieee 802.3 ap, 2021. Accessed: March 2025.

[2] G. Cowan. Introduction to wireline communication. In *Mixed-Signal CMOS for Wireline Communication: Transistor-Level and System-Level Design Considerations*, pages 1–39. Cambridge University Press, 2024.

[3] E. Matoglu. Contributed channel data for 25gb/s ethernet, 2014. Last updated: November 9, 2014.

[4] S. Palermo. S. palermo - ecen 720: High-speed links circuits and systems. https://people.engr.tamu.edu/spalermo/ecen720.html. Accessed: 2025-03-29.

[5] M. Venditti. Elec 413/6071 – mixed-signal vlsi for wireline communication systems transmit path subsystems, April 2024. Lecture notes.

9 APPENDIX

9.1 AppendixA–ELSEEReport

Overview This project involves designing a high-speed, mixed-signal PISO wireline transmitter capable of
a 16Gbps data rate with a strong emphasis on signal integrity at high frequencies. Using ELSEE methods, ELSEE Phase 1 identified potential concerns arising from the project by considering what impacts the actual fabrication process would entail. In Phase 2, the group's focus was shifted
to potential solutions that could be realized by acting from within the constraints of the project, and separately again by hypothesizing a scenario where any resources or power necessary to effect
change were not obstacles.

The Phase 1 presentation identified several pressing concerns within the five spheres of ELSEE:

1. Environmental: The analysis focused on the used life cycle of electronic products, examining the effects of e-waste on human and environmental health through a case study of global e-graveyards.

2. Economical: Concerns were raised regarding a lack of access to technologies for certain countries and supply chain inequalities among national competitors, with the aim of democratizing these devices and their market representation.

3. Ethical: The analysis looked at regional product life cycle issues at all levels of government, using this lens to examine the e-graveyard case study and the people vs. profits trade-offs made by local and federal governments.

4. Societal: The group acknowledged the contribution to the global widespread overreliance on technology.

5. Legal: The focus was on analyzing the IP and product liability issues that IP vendors face, with special attention to the real-world implications on the e-graveyard case study.

Phase 2 pivoted towards solutions to these issues that were most feasibly addressed in either hypothetical scenario (with and without executive power). The most achievable of these solutions are discussed below.

Considerations and Solutions

Environmental

Regarding human and environmental health, it was noted that while environmental metrics like greenhouse gas emissions and resource use have decreased since 2018 as minimum feature size has decreased, metrics indicating damage to human health have increased. Furthermore, this data does not include the active use and end-of-life phases.

In the active use phase, the right to repair movement struggles with proprietary fasteners and married chip-motherboard consumer electronics, limiting repair options and driving down supply. Meanwhile, global e-waste generation continues to rise, with the UN tracking 62 million tons in 2022. This suggests that tangible impact can be best achieved in the active use and end-of-life phases.

Traditional solutions like OEM replaceable parts, recycling alliances like ERI, and dedicated take-back schemes were considered. However, take-back schemes are challenging for mixed electronics due to increasing recycling costs with more mixed waste. While recycling alliances can help,

there's a cost trade-off as the alliance grows, and throughput saturation in alliance economics has been suggested.

Given these limitations, potential solutions include offering OEM parts (even for non-user-serviceable products) or creating avenues for end-of-life device collection. Implementing end-of-life collection would require management-level investment but offers the best way to handle large waste streams. Allocating plant space for material reclamation could maximize e-waste reduction and potentially create new income streams through new alliances if the on-site recycling program is effective.

Economical

Extended Producer Responsibility (EPR) is considered a critical component of the e-waste reduction strategy, especially in a B2B setting. This involves a collaborative approach with customers (other businesses) through three key elements:

1. Take-Back Program: Establishing streamlined logistics for customers to return end-of-life wireline transmitters for recycling or refurbishment, potentially integrated into existing service agreements.
2. Incentivize Returns: Offering incentives like discounts on future purchases, credits, or loyalty rewards tailored to the specific needs of business customers to encourage participation.
3. Recycling Partnerships: Collaborating with certified e-waste recyclers to provide transparency and accountability to customers through detailed reports on processing returned products, aligning with their sustainability goals.

As a project group, we acknowledge the importance of considering environmental impacts from the perspective of an IP vendor. Economic incentives such as the Green CHIPS Program in New York, created under the CHIPS Act, encourage companies to adopt sustainable measures. These incentives include potential Investment Tax Credits, R&D Tax Credits, Salaries and Wages Tax Credits, Property Taxes Credits, and Discounted Utility service delivery rates for companies meeting certain criteria. Convincing clients to adopt sustainability measures can mitigate greenhouse gas emissions and allow them to benefit from these economic incentives.

On the other hand, in the role of an IP vendor it is crucial to meet clients' specifications while considering market factors such as the high costs associated with manufacturing and design, as well as environmental impacts. While achieving both objectives can be challenging due to limited control over manufacturing processes and costs, the CHIPS Act offers economic incentives for sustainability, as mentioned.

Social

We emphasize the importance of designing for accessibility and compliance across various regions and countries in an increasingly technology-reliant world. Engineering advancements should be inclusive and adhere to global standards, which can be achieved by collaborating with local stakeholders and prioritizing energy efficiency and cost-effectiveness to ensure seamless integration into diverse societal contexts, particularly benefiting communities with limited resources. The potential to bridge the digital divide by making advanced communication technology available to underserved areas, thereby empowering education, healthcare, and economic opportunities and fostering a more equitable society. Inspiration is drawn from global initiatives like the UN Technology Facilitation Mechanism (TFM)'s STI Roadmaps, which supports the transfer of critical technologies to developing countries like Ghana, Kenya, and India to integrate advanced technologies into national development plans.

As graduating engineers, we recognize the responsibility to ensure designs are not only solve technical problems but also serve society by prioritizing public safety, environmental protection, and social equity to build trust, foster innovation, and benefit everyone, especially marginalized communities.

Observations

While the nature of IC fabrication makes it difficult to effect change at the level of an independent IP vendor, large scale change for the better of humanity and the global markets can very well be realized. Such a paradigm shift would require large-scale, industry-wide collaboration across multiple players in the industry due to the effort, time, and resources demanded across each phase of relevant product life cycles.

HIGH-SPEED WIRELINE TRANSMITTER DESIGN

TECHNICAL MANUAL

CADENCE VIRTUOSO BASED DESIGN & SIMULATION

Team 8

Concordia University

March 2025

Features This IP block is a high-speed, low-voltage mixed-signal wireline transmitter based on

TSMC’s

65nm CMOS technology node. The pre-serializer stages are implemented utilizing standard (foundry-defined) transistors and resistor models. A 32-bit parallel data stream at user-defined rates of 250 or 500 Mbps can be transmitted at 8 or 16 Gbps using selected models included in the

IEEE 802.3 Ethernet Working Group family of electrical link channels. Equalization is performed via two- or three-tap FFE.

Module Functions

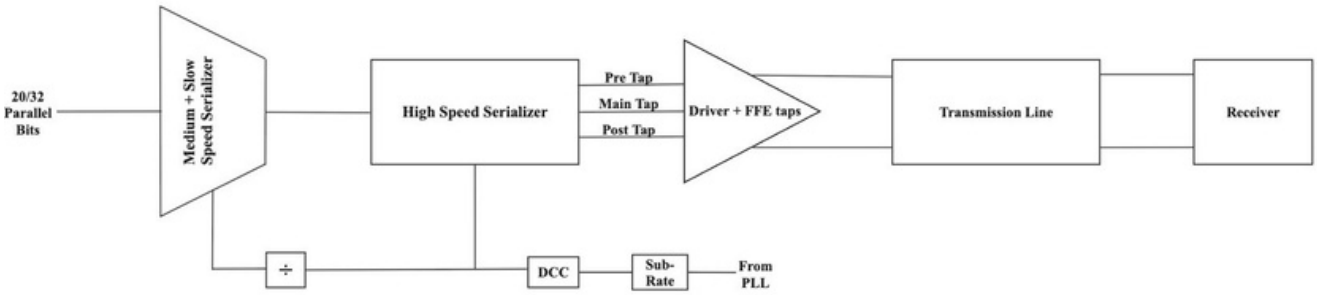


Figure A: High-level block diagram of the transmitter

The pre-serializer (Figure B) implements a multi-stage architecture to gradually serialize par- allel 20/32-bit input data into medium-speed differential streams. Logic-In-Differential-Out (LIDO) converters convert single-ended (digital, full-swing) input bits into odd/even differen- tial pairs operating at 250/500MHz. These signals then feed into the first MUX stage (Figure C) consisting of eight differential 4:1 multiplexer (slow stage) (figure 5.3.0) that combine data streams to 0.5-1 Gbps outputs. Each slow-stage MUX incorporates precisely tuned delay ele- ments (60ps at 16Gbps operation) to maintain bit alignment within 12% of the unit interval (UI). The medium-speed stage further aggregates these streams to 4-8 Gbps using higher-drive mul- tiplexers (2000μA current) with 16% UI delay matching (20ps at 16Gbps). Critical components like the LIDO converter employ 180nm bias transistors (VREF = 0.5V) for stable common-mode operation, while data path devices use minimum 60nm lengths for speed optimization.

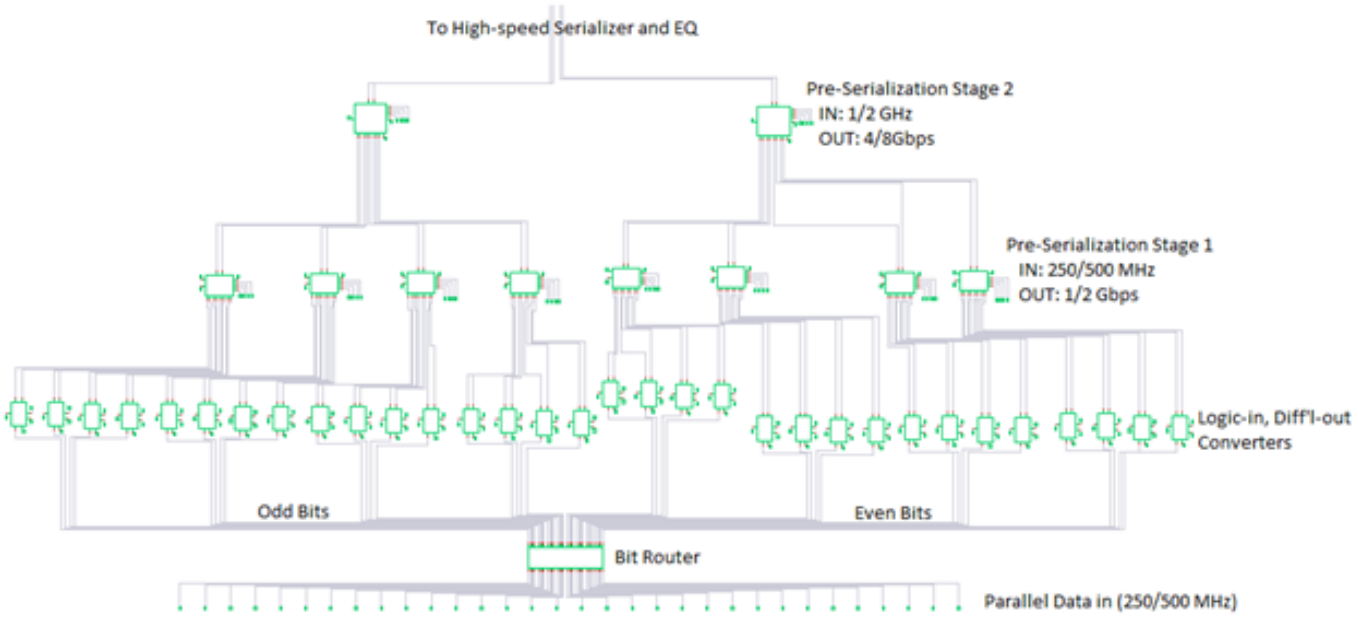


Figure B: Pre-serializer module block diagram

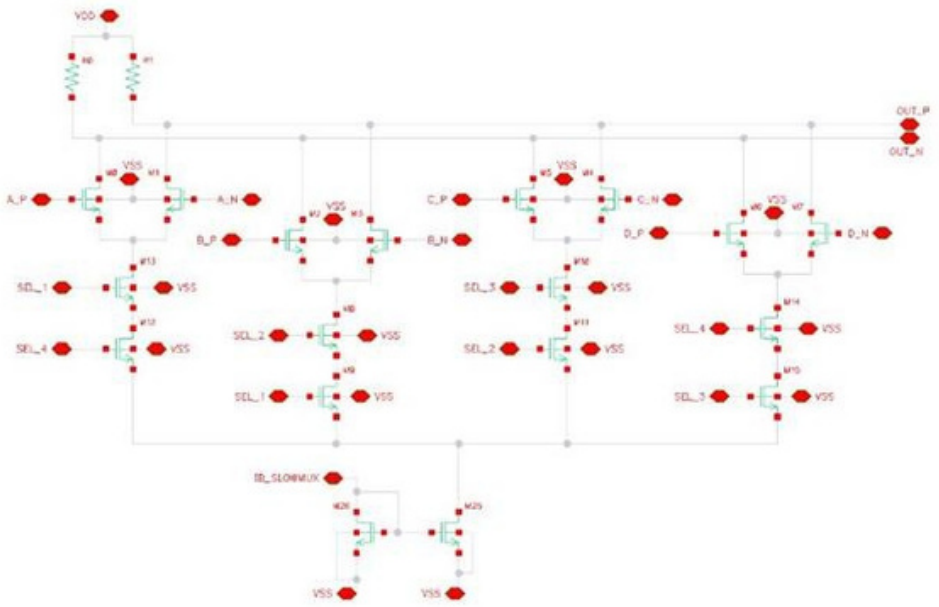


Figure C: 4-to-1 multiplexer, schematic diagram, for both preserializing stages

The high-speed multiplexer (Figure D) completes serialization with a final 2:1 stage that generates the 8Gbps/16Gbps output stream. Implemented as a current-steering differential pair, this stage toggles between positive and negative output branches synchronized to the clock edges. Key features include adjustable slew rate control (via 600µA current sources) and integrated termination to minimize reflections. The output maintains compatibility with the original 20/32-bit parallel input width while achieving full serialization through this cascaded architecture. Delay matching in this final stage is critical, with the transistor widths optimized to balance loading and propagation delay.

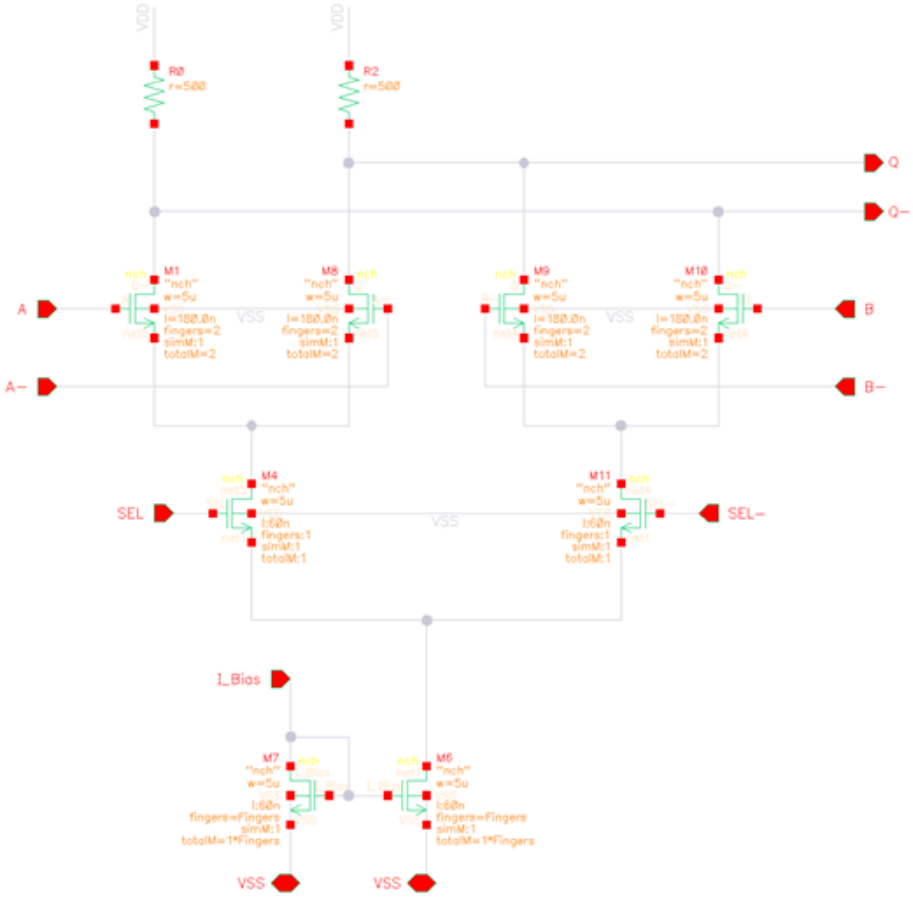


Figure D: 2-to-1 MUX Schematic diagram

The main driver (Figure E) employs a current-mode logic (CML) architecture optimized for high-speed wireline transmission up to 16Gbps. The driver core uses 12-finger NMOS transistors (L=60nm) in the differential switching pairs to achieve the target 20mA drive current while maintaining 50Ω output impedance. The multi-finger layout improves matching and reduces gate resistance, critical for minimizing jitter in the output eye diagram. Each finger width is carefully optimized to balance RC delays and current density limits of the 65nm process.

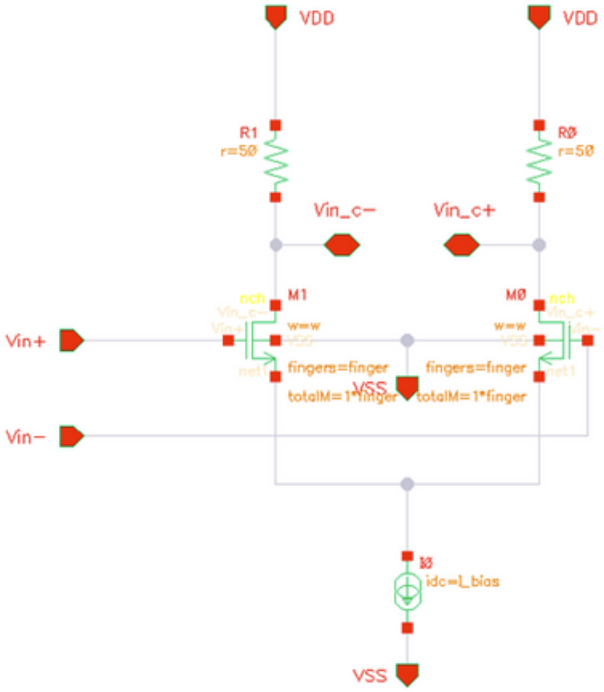


Figure E: Main Driver Schematic

The feed-forward equalizer (FFE) (Figure F) implements three programmable taps (pre-cursor, main, post-cursor) using identical 12-finger, 60nm-length transistors for current DACs. This matching ensures consistent timing characteristics across all taps. The main cursor tap delivers the full 20mA drive current, while pre/post-cursor taps scale proportionally (α ×20mA). The absence of series resistors in the FFE path indicates direct current-steering into the summing node.

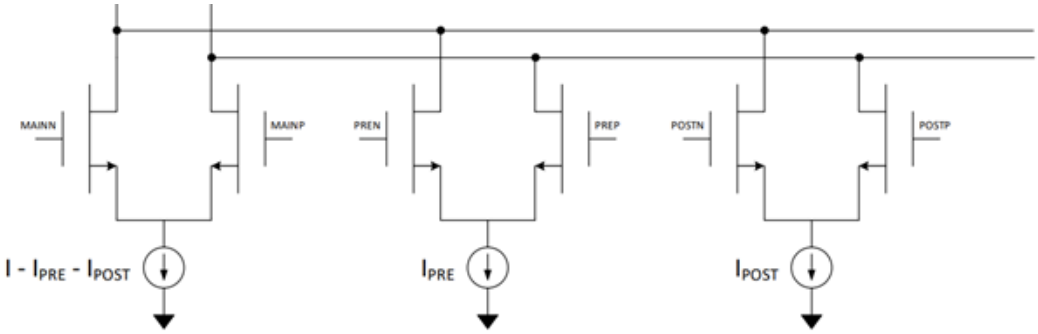


Figure F: Equalization Taps Circuit Schematic

Key Specifications for the Transmit Path Subsystem

#	Requirement	Min	Typical	Max	Units
1	Operating Temperature	-40	25	125	◦ C
2	Operating Voltage	0.95	1.0	1.10	V
3	Input Signal Amplitude	1000	-	-	mVppd
4	Driver FIR-tap Step Granularity	-	-	25	mVppd
5	Driver Pre-/Post-Cursor De-Emphasis Range	-	8	-	dB
6	Driver Output Impedance	90	100	110	Ω
7	Data Rate (Config 1)	8	16	-	Gbps
8	DCC Correction Range	-	T*clk*	-	s
9	DDC Maximum Actuator Step Size	-	T*clk*	1.03 T*clk*	s
10	DCC Referred Offset, Random Input	-	-	0.25	mV*
11	Data Rate (Config 2)	4	8	-	Gbps
12	Data Rate (Config 3)	5	5	-	Gbps
13	Digital Interface Bus Size	20	-	32	bits
14	Operating Frequency	125	-	500	MHz
15	Transistor Finger Width	-	5	-	µm

Serializer device parameters

Subcomponent	Transistor Fingers	Transistor Length (nm)	Resistance (Ω)	Drive Current (μA)	Other Parameters
LIDO Converter	1 (Data) 1 (Bias)	60 (Data) 180 (Bias)	2.5K / 3.0K	100	V_{ref} = 0.5V
Stage 1 MUX	1 (Data) 2 (Clock) 1 (Bias)	60 (Data) 120 (Clock) 60 (Bias)	1.25K	200	Bit Delay = 12% of UI
Stage 1 MUX Delay Element	1 (Data) 1 (Bias) 2 (Cap.)	240 (Data) 120 (Bias) 60 (Cap.)	2.5K	200	60ps during 16 Gbps
Stage 2 MUX	3 (Data) 3 (Clock) 14 (Bias)	60 (Data) 120 (Clock) 60 (Bias)	200	2000	Bit Delay = 16% of UI
Stage 2 MUX Delay Element	1 (Data) 1 (Bias)	240 (Data) 120 (Bias)	1.6K	160	20ps during 16 Gbps
Fast MUX	2 (Data) 1 (Clock) 2 (Bias)	180 (Data) 60 (Clock) 60 (Bias)	500	600	–

Driver and FFE tap device parameters

Component	Transistor Fingers	Transistor Length (nm)	Resistance (Ω)	Drive Current (mA)
Main Driver	12 (Data)	60 (Data)	50	20
FFETaps	12 (Data)	60 (Data)	–	–

External Connections

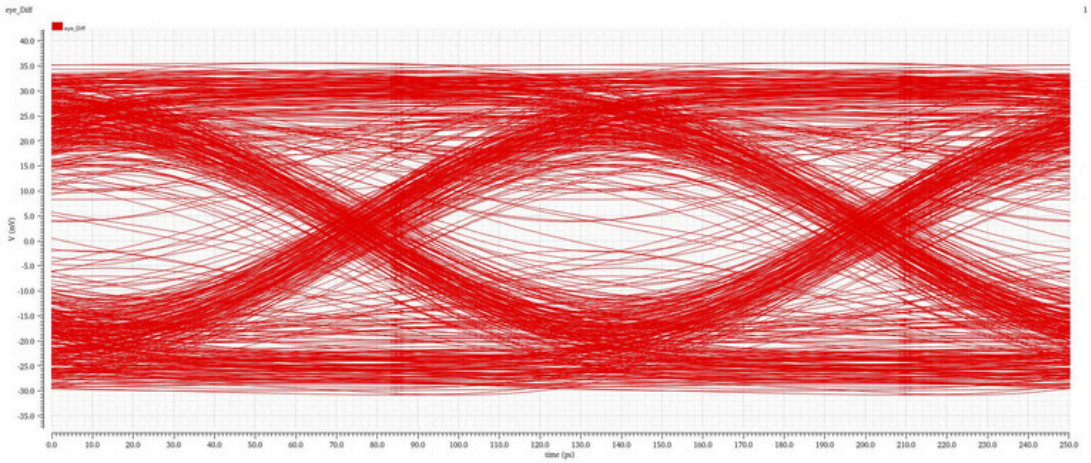
Power Connections

Input Current	1mA
Input Voltage	1.0V
Ground	Zero V

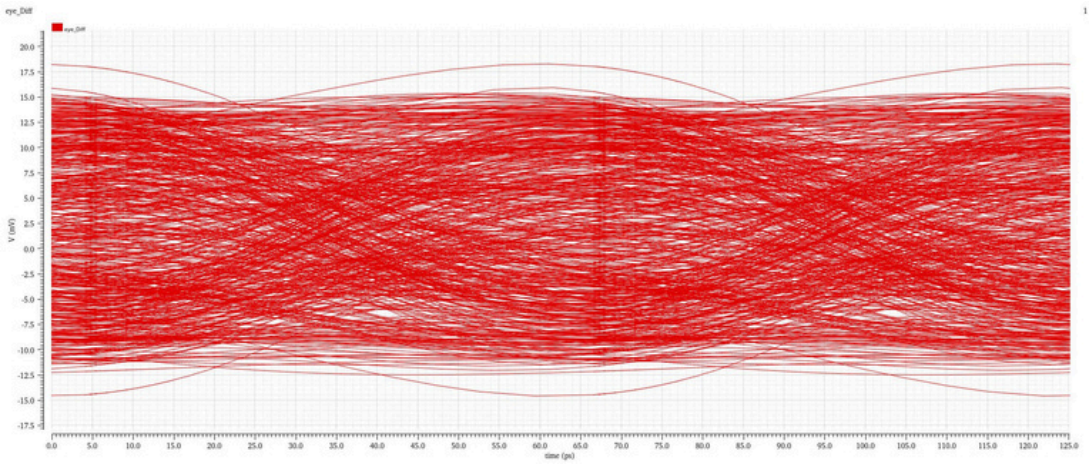
I/O Connections

Input Data	20- or 32-bit, parallel in, 250/500 MHz
Output Data	5 to 16 Gbps serial
Clock	2x complementary 4 or 8 GHz

Response Diagrams



Eye Diagram, 8 Gbps



Eye Diagram, 16 Gbps

9.3 AppendixC–User'sManual

Product Description

Congratulations on your purchase of the Team 8 High-speed Transmitter block! You now have access to a high-speed, low-voltage mixed-signal wireline transmitter based on TSMC's 65nm CMOS technology node. This module can process 20-bit and 32-bit parallel data streams at an input rate of 250 or 500 MHz with a maximum output rate of 16 Gbps. Your new transmitter is compatible with selected IEEE 802.3 Ethernet Working Group electrical link channels and features multitap feed-forward equalization for channel compensation.

Operation Summary

TheTeam8High-speedTransmitterrequiresthefollowinginputstooperate:

- a 1 mA reference current. Connect this to I REF pin.
- a suitably grounded 1V reference voltage network. Connect to VDD and VSS pins.
- complementary 1V/8GHz reference clock pair. Connect these to CLK P and CLK N pins.

Connect your parallel data source to the input bus. Select your desired word size (parallel bus width) and input data rate using the switch console provided.

Troubleshooting

If unexpected issues occur during operation:

- clear the inputs and restart the transmitter
- ensureoperatingtemperaturedoesnotexceed-40to125degreesCelsius,asthiscanseverely impact operation
- ensure that supply voltages and currents are correct, stable, and calibrated correctly
- ensure consistent levels and frequencies in the input data, as aberrations can cause timing clashes and transmission errors

Frequently Asked Questions

What sort of access does my purchase provide? – This transmitter module was created with Cadence Design Tools using TSMC's 65nm process node via CMC Microsystems; all rights reserved for any associated entity or party. Purchasing this block grants limited access for high-level use in any authorized third-party application. By purchasing you agree not to dismantle or reverse engineer any physical components nor dissemble any proprietary technologies covered under your IP licensing agreement. For more information, contact your designated customer service representative.

Where can I use this technology? – Team 8 Transmitter technology is tested for channel lengths up to 5m, supporting a wide range of data transfer applications for short- and medium-length transmission.

Why Team 8 Transmitters? – As leaders of innovation in the field of mixed-signal processing, and with a dedicated team of staff and advisors, Team 8 is proud to bring you the best in transceiver technology for your IP or B2B needs.

9.4 AppendixD–MATLABCode

```
% imp_from_s4p.m
clos   all
e      all;
clear
clc
T_sym=   62.5e-12;          % SymbolRate@ 16  Ghz
%T_sy   =   125e-12; % Symb   Rate @8  Ghz
m       = 200e-12;  % ol       Rate @5  Ghz
%T_sy                Symb
Ts =T_sym/100;      % CppSiminternal time step
m                   ol
nsym_short=500*100e-12/T_sym; %Impulse response

% LoadChannelData (S-parametersfroma file)
FileName= 'P2TX2_P1RX2.s4p';
SingleEndedData read(rfdata.data,  FileName);
Freq = SingleEndedData.Freq;
DifferentialSparams= s2sdd(SingleEndedData.S_Parameters);

% Create the networkobject fromS-parametersand frequency
rfdata.network('Data',  DifferentialSparams, 'Freq',  Freq);

% Extract S-parameters(S11  and S21)  an
thru_S11(1,:)   =   DifferentialSparams(1,1,:);   thru_S21(1,:)   =
   DifferentialSparams(2,1,:);
thru_S11_mag=20*log10(abs(thru_S11));
thru_S21_mag=20*log10(abs(thru_S21));

% Thru ChannelImpulseResponse
H21=thru_S21;
imp= xfr_fn_to_imp(Freq',H21,Ts,T_sym);  % Producesimpulseresponse
imp_short=imp(1:floor(nsym_short*T_sym/Ts));

% Save the importantpoints of  the ImpulseReponse
thru_imp_short=imp_short;
thru_imp=imp;
```

```
time=(1:size(imp,2))*1e-12;
ir =[ thru_imp_short '];
save channel_impulse.mat ir;

% Plot the Insertion Loss of the Channel
f_GHz = Freq / 1e9;
 figure;
hold on;
plot(f_GHz, thru_S21_mag,'LineWidth', 2);
title('\textbf{Insertion Loss of Channel - P2TX2\_P1RX2.s4p}', '
   Interpreter', 'latex', 'FontSize', 24);
xlabel('Frequenc (GHz)', 'Interpreter', 'latex', 'FontSize', 14);
y Loss (dB)', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('Insertion
xlim([0 25]);
grid on;

ax = gca;
ax.TickLabelInterpreter = 'latex';
ax.FontSize = 12;

% -%--%----------------------------------------------
function to Convert Frequency Response to Impulse Response
n ----------------------------------------------
        imp= xfr_fn_to_imp(f, H, Ts, T_sym)

    % Set the number of FFT points
    num_fft_pts = 2^16;

    % Symbol frequency: inverse of symbol time (1/T_sym)
    f_sym = 1 / T_sym;

    % Maximum sampling frequency based on transfer function
    f_sym_max = 2 * max(f);

    if (f_sym > f_sym_max)
```

```
    error('Max input frequency too low for requested symbol rate,
        can''t interpolate!');
end

% Adjust the maximum symbol frequency
f_sym_max = f_sym * floor(f_sym_max / f_sym);

% Magnitude and phase of the frequency response
H = abs(H);
m = angle(H);
H
% Handle the case when the first frequency is 0 to avoid phase
    issues
if f(1) == 0
    % If f(1) is zero, create a symmetric frequency range and make
        the phase zero
    Hm_ = [fliplr(H(2:end-1)) H];
    ds  = [fliplr(-Hp(2:end-1)) Hp];
    fds_ = [fliplr(-f(2:end-1)) f];
    fds = fds_;
    m   = fds;
else fds_
    % If f(1) is not zero, mirror the response around zero
        frequency
    Hm_ = [fliplr(Hm(1:end-1)) Hm];
    ds  = [fliplr(-Hp(1:end-1)) [- 0 Hp];
    Hp_ = fliplr(f(1:end-1)) [- f];
    ds  = fliplr(f(1:end-1)) 0 f];
end fds_
    m
    fds
% Sampling frequency related variables
    p
fmax = 1 / Ts;
df = fmax/2 / num_fft_pts; % Frequency resolution

% Create the interpolated frequency range
```

```
f_ds_interp = -f_sym_max/ 2 + df : df : f_sym_max/ 2;
f_ds_interp_a = -fmax/ 2 + df : df : fmax/ 2;

% Interpolate the magnitude and phase of the transfer function over
    the new frequency grid
Hm_ds_inte  = spline(fds_m  Hm_ds,f_ds_interp);
rp          = ,             unwrap(Hp_ds)f_ds_interp);
Hp_ds_inter   spline(fds_p,
% Shift the interpolated magnitude and phase to center around zero
   frequency
Hm_ds_interp_sh_or = fftshift(Hm_ds_interp);
ig                 = fftshift(Hp_ds_interp);
Hp_ds_interp_sh_ori
% Zero-padding the interpolated frequency response to match FFT
   points
Hm_ds_interp_a(1: (2 * num_fft_pts- size(Hm_ds_interp,2)) / 2) =
    0;
Hm_ds_interp_a((2* num_fft_pts- size(Hm_ds_interp,2)) / 2 + 1 :
   (2 * num_fft_pts+ size(Hm_ds_interp,2)) / 2) = Hm_ds_interp;
Hm_ds_interp_a((2* num_fft_pts+ size(Hm_ds_interp,2)) / 2 + 1 :
   2 * num_fft_pts) = 0;

Hp_ds_interp_a(1: (2 * num_fft_pts- size(Hp_ds_interp, 2)) / 2) =
    0;
Hp_ds_interp_a((2* num_fft_pts- size(Hp_ds_interp, 2)) / 2 + 1 :
   (2 * num_fft_pts+ size(Hp_ds_interp, 2)) / 2) = Hp_ds_interp;
Hp_ds_interp_a((2* num_fft_pts+ size(Hp_ds_interp, 2)) / 2 + 1 :
   2 * num_fft_pts) = 0;

% Shift the zero-padded frequency response
Hm_ds_interp_  = fftshift(Hm_ds_interp_a);
sh             = fftshift(Hp_ds_interp_a);
Hp_ds_interp_s
h
% Combine the magnitude and phase to create the full complex
   frequency response
```

```
    H_ds_interp_sh= Hm_ds_interp_sh* exp(1j * Hp_ds_interp_sh);

    % Impulseresponseis the IFFT of the interpolated frequency
       response
    imp= ifft(H_ds_interp_sh);

    % Extract the real part of the impulseresponse(imaginary part is
       typically small)
    imp_r = real(imp);

    % Symboltime step based on the maximumfrequency
    dt_sym= 1 / fmax;

    % Assign the real part of the impulseresponseas the final result
    imp= imp_r;

    % Refitting the data into the simulator's time step
    dt_time = 0 : dt_sym:dt_sym * (length(imp_r) - 1);
    time = 0 : Ts : dt_time(end);
return
end
```

```
% channel_data.m

clear all
clc
close all

bit_perio   = round(1e12 /  16e9); % This is 8Gb/s with 1ps step time
d           = 1 * (0e-2);  %Used in plotting pulse response
opt_samp
% Load ChannelImpulseResponse
le
load channel_impulse.mat;

tsample= 1e-12;     %timestep (1ps) for the impulsereponse
sample_num= size(ir,   1);
ir_data =   ir(:, 1);
scale_ir  = 0.5;              % (This is Vpp Differential)

sig_ir  = ir_data * scale_ir;

% TX Equalization Function
eq_tap_number= 3;      % Equalization Tap Number
precursor_number= 1; % Number of Pre-Cursor Taps
num_bits= 6;                  % f  Equalization Resolution
                                T
% Calls TX Equalization Function X
[taps,  taps_quan]= tx_eq(sig_ir,  bit_period, eq_tap_number,
   precursor_number,num_bits);

% For Pulse Response
nt =  100;
m(1:20) =  0;  m(21) = 1;  m(22:100) = 0;

% TX FIR Equalization Taps
eq_taps = taps_quan;
m_fir = filter(eq_taps,   1,   m);
```

```
m_dr= reshape(repmat(m_fir,bit_period, 1), 1, bit_period * size(m_fir
   , 2));
m_tx= m_dr;

% GenerateUnequalizedData
eq_taps_noeq= [1];   % No Equalization
m_fir_noeq= filter(eq_taps_noeq, 1, m);
m_dr_noeq= reshape(repmat(m_fir_noeq,bit_period, 1), 1, bit_period *
   size(m_fir_noeq, 2));

% ChannelOutput
data_channel= 0.5 * conv(sig_ir(:, 1), m_dr(1:nt * bit_period));
data_channel_noeq= 0.5 * conv(sig_ir(:, 1), m_dr_noeq(1:nt*
   bit_period));

% Calculate PulseResponse for Equalized Data
[max_data_ch,max_data_ch_idx]= max(abs(data_channel));
pulse_xaxis = ((1:size(data_channel, 1)) - max_data_ch_idx)/
   bit_period;

sample_offset= opt_sample* bit_period;
for i = 1:101
    sample_points(i) = max_data_ch_idx+ sample_offset+ (i - 11) *
       bit_period;
end
sample_values= data_channel(sample_points);
sample_points= (sample_points- max_data_ch_idx)/ bit_period;

% Calculate Pulse Responsefor UnequalizedData
[max_data_ch_noeq,max_data_ch_idx_noeq]= max(abs(data_channel_noeq));
pulse_noeq_xaxis= ((1:size(data_channel_noeq, 1)) -
   max_data_ch_idx_noeq)/ bit_period;

sample_offset_noeq= opt_sample* bit_period;
for i = 1:101
```

```
    sample_points_noeq(i) = max_data_ch_idx_noeq + sample_offset_noeq +
        (i - 11) * bit_period;
end
sample_values_noeq = data_channel_noeq(sample_points_noeq);
sample_points_noeq = (sample_points_noeq - max_data_ch_idx_noeq) /
    bit_period;

% Plotting
figure;
noeq_pulse = plot(pulse_noeq_xaxis, 2 * data_channel_noeq, '-r'); hold
    on;
eq_pulse = plot(pulse_xaxis, data_channel, '-r'); hold on;
eq_points = stem(sample_points, sample_values, '-*r');
noeq_points = stem(sample_points_noeq, 2 * sample_values_noeq, '-*r');
grid on;

% Formatting
set(eq_pulse, 'LineWidth', [2.0]);
set(noeq_points, 'LineWidth', [2.0]);
set(noeq_pulse, 'LineWidth', [2.0]);
set(eq_points, 'LineWidth', [2.0]);
set(eq_pulse, 'Color', [0 0 1]);
set(eq_points, 'Color', [0 0 1]);

AX = gca; % Current axes
set(AX, 'LineWidth', [2.0]);
ax.TickLabelInterpreter = 'latex';
    'XLim', [-10 10],
    'XTick', -5:0.5:10);
    'YTick', 0:0.05:1);
                = 'latex';
ax.FontSize = 12;
xlabel('Frequency (GHz)', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('Insertion Loss (dB)', 'Interpreter', 'latex', 'FontSize', 14);
title('\textbf{Un-Equalized \& Equalized Impulse Response}', '
    Interpreter', 'latex', 'FontSize', 16);
```

```
legend('NoEq', '3-tap TX Eq', 'Location', 'best', 'Interpreter', '
    latex');

disp('Quantized Taps')
disp(taps_quan);

% Equalized Eye Height Calculation
eye_height_equalized = 2*(sample_values(11)-(sum(abs(sample_values
    (1:10)))+sum(abs(sample_values(12:101)))));
disp('Equalized Eye Height (V):');
disp(eye_height_equalized);

% Unequalized Eye Height Calculation
unequalized_eye_height = 2 * (sample_values_noeq(11)- (sum(abs(
    sample_values_noeq(1:10))) + sum(abs(sample_values_noeq(12:101)))));
disp('Unequalized Eye Height (V):');
disp(unequalized_eye_height);

ISI_contribution_equalized = sample_values(11) - (eye_height_equalized
    / 2);
disp('ISI for equalized')
disp(ISI_contribution_equalized);

ISI_contribution_un_equalized = sample_values_noeq(11) - (
    unequalized_eye_height/2);
disp('ISI for un-equalized')
disp(ISI_contribution_un_equalized);

P_noise_eq = ISI_contribution_equalized^2;
P_noise_noeq = ISI_contribution_un_equalized^2;

P_signal_eq = (eye_height_equalized^2)/4;
P_signal_uneq = (unequalized_eye_height^2)/4;

SNR_eq = P_signal_eq/P_noise_eq;
```

```
SNR_eq_dB = 10*log10(SNR_eq);
disp('SNR equalized (dB)');
disp(SNR_eq);

SNR_uneq = P_signal_uneq/P_noise_noeq;
SNR_uneq_dB = 10*log10(SNR_uneq);
disp('SNR un-equalized(dB)');
disp(SNR_uneq_dB);

% Calculate BER for equalized signal
BER_eq = qfunc(sqrt(2 * SNR_eq));
disp('BER for equalized signal:');
disp(BER_eq);

% Calculate BER for unequalized signal
BER_uneq = qfunc(sqrt(2 * SNR_uneq));
disp('BER for unequalized signal:');
disp(BER_uneq);

% Compute Frequency Response of Unequalized Impulse Response
IR_FFT_no_eq = linspace(0, 1/(2*tsample), sample_num/2); % Frequency axis in Hz
IR_FFT_no_eq_Magnitude = fft(sig_ir);    % FFT of the impulse response
                       = abs(IR_FFT_no_eq(1:sample_num/2)); % Magnitude of the FFT

% ------------------------------------------------------
% function that use least square algorithm to obtain FIR taps
%              ------------------------------------------------------

function [taps,taps_quan]=tx_eq(ir_in,bit_period,eq_tap_number,
   precursor_number ,num_bits)

    ir_process =  ir_in;
    precursor_samples = 20;
```

```
% Compute the pulse response by convolving the input IR with a
    rectangular pulse of length 'bit_period'
pulse_response = conv(ir_process, ones(1, bit_period));

% Find the maximum value and its index (time) in the pulse response
[max_pulse_value, max_pulse_time] = max(pulse_response);

% Define the sample times around the pulse peak, considering the
    precursor samples and a range of 30 bit periods
sample_times = [max_pulse_time - 1 * precursor_samples * bit_period
    : bit_period : max_pulse_time + 30 * bit_period];
sample_values = pulse_response(sample_times);

% Construct the H matrix, which will be used to solve for the
    equalizer taps
H(:, 1) = sample_values';
H(size(sample_values, 1) + 1 : size(sample_values, 1) +
    eq_tap_number - 1, 1) = 0;

for i = 2 : eq_tap_number - 1
    H(size(sample_values, 1) + i : size(sample_values, 1) +
    eq_tap_number - 1, i) = sample_values';
end;

H(1 : eq_tap_number - 1, eq_tap_number) = 0;
H(eq_tap_number, eq_tap_number) = size(sample_values, 1) - 1,
    eq_tap_number) = sample_values';

% Construct the desired output vector Ydes for the equalizer,
    initializing precursor and postcursor values
Ydes(1 : precursor_samples - 1, 1) = 0;
Ydes(precursor_samples, precursor_samples + precursor_number - 1, 1) =
    0;
```

```
    Ydes(precursor_samples + precursor_number + 1, 1) = 1;   % Set the
       middlepoint to 1
    Ydes(precursor_samples + precursor_number + 2 : size(H, 1), 1) = 0;
         % Rest of the values are 0

    % Solve for the equalizer taps using the Least Squaressolution: W
       = (H'H)^(-1) H' Ydes
    W= (H' * H) ^ (-1) * H' * Ydes;

    % Scale the equalizer taps to normalizetheir total power
    Itot = 1;
    tap = Itot * W/ sum(abs(W));
    s
    % Separatethe magnitudeand sign of the taps
    taps_abs= abs(taps);
    taps_sign = sign(taps);

    % Quantizethe taps basedon the numberof bits specified
    partition = [1 / (2 * (2 ^ num_bits- 1)) : 1 / (2 ^ num_bits- 1)
       : 1 - 1 / (2 * (2 ^ num_bits- 1))];
    codebook= [0 : 1 / (2 ^ num_bits- 1) : 1];
    [index, abs_taps_quan]= quantiz(abs(taps), partition, codebook);

    % Reapplythe sign to the quantizedtaps to get the final quantized
        tap values
    taps_quan= taps_sign' .* abs_taps_quan;

    % Adjust the middletap value to preserve the sumof the taps
    taps_quan(precursor_number+1) = sign(taps_quan(precursor_number+
        1)) * (1 - (sum(abs(taps_quan))- abs(taps_quan(
       precursor_number+ 1))));

end
```

```
% plot_eye_diagram.m

clear all;
clc
close all

bit_perio  = round(1e12 / 8e9); % This is 10Gb/s with 1ps step time
d          = 1 * (0e-2); %Used in plotting pulse response
opt_samp
% Load ChannelImpulseResponse
le
load channel_impulse.mat;

tsample= 1e-12; % Impluseresponsehas 1ps time step
sample_num= size(ir, 1);
ir_data =  ir(:, 1);
scale_ir = 1; % Vp Differential
              p
sig_ir = ir_data * scale_ir;


% PRBS- Generatrerandominput signal to test our channel
n  = 1e3; %umber of bits
t = rand(1, nt + 1);
m= -1 * sign(m- 0.5).^2 + sign(m- 0.5) + 1;
m
% TX FIR Equalization Taps
Normal_signal= [1];
eq_taps= [-0.0476  0.7460  -0.2063]; % Obtainedfrom
m_fir = filter(eq_taps, 1, m);

m_dr= reshape(repmat(m_fir,bit_period, 1), 1, bit_period * size(m_fir
   , 2));
m_tx= m_dr;

% Pass data throughthe channel
```

```
data_channel= 0.5 * conv(sig_ir(:, 1), m_dr(1:nt * bit_period));

%time_m_dr= (1:size(m_dr, 2)) * 1e-12;
%time_dc= (1:size(data_channel, 1)) * 1e-12;

% Eye Diagramfor NormalSignal
m_fir_normal= filter(Normal_signal, 1, m); % Filter using
    Normal_signal

m_dr_normal= reshape(repmat(m_fir_normal,bit_period, 1), 1,
    bit_period * size(m_fir_normal, 2));
data_channel_normal= 0.5 * conv(sig_ir(:, 1), m_dr_normal(1:nt*
    bit_period));

% EyejDiagramPlot for Equalized
offset;
        = 144;

for i = 55:floor(size(data_channel, 2) / bit_period) - 500
    eye_data(:, j) = 2 * data_channel(floor((bit_period * (i - 1))) +
        offset: ...
                                    floor((bit_period * (i + 1))) +
                                        offset);
    j = j + 1;
end;
time = 0:2 * bit_period;

% Eye DiagramPlot for Un-Equalized
eye_data_normal= []; % Initialize for NormalSignal
for i = 55:floor(size(data_channel_normal,2) / bit_period) - 500
    eye_data_normal(:,j) = 2 * data_channel_normal(floor((bit_period*
        (i - 1))) + offset: ...
                                                    floor((bit_period *
                                                        (i + 1))) +
                                                    offset);
```

```
    j = j + 1;
end;

figure;
plot(time, eye_data);
title('\textbf{Eye Diagramwith Equalization}', 'Interpreter', 'latex',
    'FontSize', 16);
xlabel('UI', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('Voltage (V)', 'Interpreter', 'latex', 'FontSize', 14);
grid on;
ylim([-1 1]);

ax = gca;
ax.TickLabelInterpreter = 'latex';
ax.FontSize = 12;

figure
plot(time, eye_data_normal);
title('\textbf{Eye Diagramwith Equalization}', 'Interpreter', 'latex',
    'FontSize', 16);
xlabel('UI', 'Interpreter', 'latex', 'FontSize', 14);
ylabel('Voltage (V)', 'Interpreter', 'latex', 'FontSize', 14);
grid on;
ylim([-1 1]);

ax = gca;
ax.TickLabelInterpreter = 'latex';
ax.FontSize = 12;
```