A 28Gbps Serializer & Deserializer For High Speed IO Links

by

Mudit Awasthi

A thesis submitted in partial fulfillment for the
degree of Master of Technology

under supervision of
Dr. Anuj Grover
Department of Electronics and Communication Engineering
Indraprastha Institute of Information Technology, Delhi

June 2019

A 28Gbps Serializer & Deserializer For High Speed IO Links

by

Mudit Awasthi

A thesis submitted in partial fulfillment for the
degree of Master of Technology

to

Indraprastha Institute of Information Technology, Delhi

June 2019

Certificate

This is to certify that the thesis titled **"A 28Gbps Serializer & Deserializer For High Speed IO Links"** submitted by **Mudit Awasthi** (Roll Number - MT17100) for the partial fulfillment for the degree of **Master of Technology** in **VLSI & Embedded Systems** of the requirements is an original work carried out by him under my supervision. In my opinion, thesis has reached the standards fulfilling the requirements of the regulations relating to the degree. The results contained in the thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

Date: _____

Dr. Anuj Grover
Associate Professor
Department of Electronics and Communication Engineering
Indraprastha Institute of Information Technology, Delhi
New Delhi 110020

Mr. Bhavin Odeddara
Sr. Technologist
ASIC DEV Engineering
Western Digital (SanDisk) India Device Design center Pvt Ltd
Bengaluru 560103

Abstract

Technology scaling and continuous increase in data rate have been the driving forces leading the innovations in high-speed interfaces. With the growing need for ultra-low power and high-speed data rate signaling, integrated systems-on-chip have become mainstream critical components in the modern computing system. Traditional parallel links have been used in circuits for a long time, where the skew between clocks and data lanes in the link becomes difficult to control with the faster data rate. The alternate solution is to go with faster serial links (reduced pin count and area). Traditional parallel links like PCI got replaced by high-speed serial IO standards like PCI-e, SATA, USB, TBT, DP, HDMI, M-PHY which serve multiple applications like the processor to processor or processor to peripheral communication. Serializer and Deserializer are the main blocks of High-Speed IO links. Serializer at the transmitter side converts the parallel data stream into a serial data stream while Deserializer at the receiver side converts back serial data stream into a parallel data stream. This allows for higher data rate, less number of interconnect pins, area, and cross-talk compare to parallel data transmission.

This work presents the design and implementation of 28 Gbps Serializer and Deserializer for High-speed IO links. An overview of high-speed IO links with a brief discerption of all the blocks of the system is presented. The fundamentals to design high speed and low power Serializer and Deserializer have been discussed along with the different architecture of Flip-flop, Serializer, and Deserializer. The proposed circuit is implemented in TSMC 16 nm CMOS technology. Simulation outcome has shown that Serializer and Deserializer can support 28 Gbps data rate with maximum power consumption 8.22 mW (Serializer) and 3.20 MW (Deserializer) with a power efficiency of 0.29 pJ/bit (Serializer) and 0.12 pJ/bit (Deserializer), making these circuits quite useful for High-Speed IO links. Simulation results show that the proposed design is robust against PVT variations.

Acknowledgements

I would like to thank my supervisors Dr. Anuj Grover and Mr. Bhavin Odedara for their guidance and corrections made in the scope of this work.

This work is dedicated to my parents Mr. Pradeep Kumar Awasthi and Mrs. Madhu Bala Awasthi for their teachings and values passed on to me. I thank their understanding even in those moments in which my immaturity prevailed. To my brother, Mohit Awasthi for his support and friendship throughout these years.

I would like to thank all the team members of the ASIC Analog team at SanDisk for their help on obtaining this works final results. Special thanks to Nukamreddy Venkata Subba Reddy for the enthusiastic discussions. Performing this work among these outstanding professionals contributed a lot to my evolution as a professional and a human being.

I would also like to thank my friends Sakshar Pathak and Avinash Pandit for their suggestions and support.

Contents

List of Figures

List of Tables

Dedicated to my beloved parents...

Chapter 1

Introduction

1.1 Motivation

The advancement of semiconductor processing technology has facilitated processors to compute huge amount of data. To fully utilize this effect IO link should also scale up in terms of bandwidth with minimal effect on pin count, area and power. Traditional parallel links has been used in circuits for long time, where the skew between clocks and data lanes in the link become difficult to control with faster data rate. The alternate solution is to go with faster serial links (reduced pin count and area). Traditional parallel links like PCI got replaced by high speed serial IO standards like PCI-e, SATA, USB, TBT, DP, HDMI, M-PHY which serve multiple application like processor to processor or processor to peripheral communication. Serial IO communication comes with set of challenges namely channel loss, ISI, crosstalk, added complexity in the receiver to facilitate clock recovery from data stream Fig.1.1[2] .

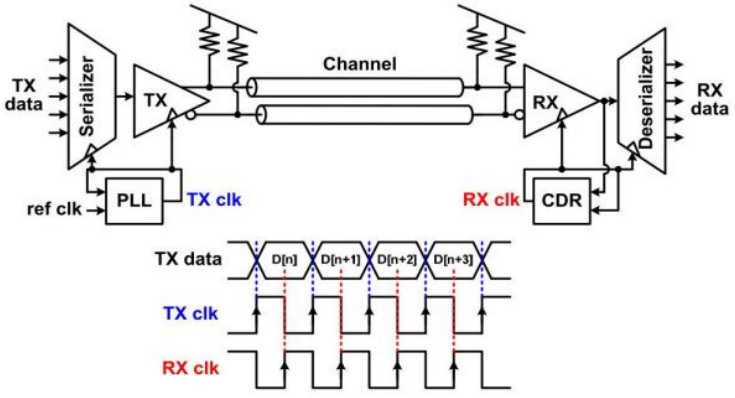

FIGURE 1.1: High Speed IO Link

Figure 1.1 shows the major components of a standard high-speed serial link (SerDes). Due to the limit on the number of IO pins in chips packages and printed circuit board, a transmitter serialize the incoming parallel data and the receiver deserializes high-speed data into parallel data. A Serializer and Deserializer are to be designed to work with high-speed serial link (SERDES) to achieve the goal of high-speed serial data transmission.

1.2 Objectives

The objective of this thesis is to design a Serializer and Deserializer for high-speed IO link systems. The proposed Serializer and Deserializer are to be designed in 16nm FinFET technology. Serializer should convert low speed 40 bit parallel data into 28Gbps serial data and Deserializer should convert high speed 28Gbps serial data into 40 bit parallel data with power consumption less than(< 10mw) in all the process corner. The proposed circuit should work on 0.8V supporting all process, voltage, and temperature variations.

1.3 Structure of the Work

- **Chapter 2: Architecture of High Speed SerDes**
  This chapter describes the basic fundamental concept and architecture of High-speed SerDes.
- **Chapter 3: Different Architecture of PISO and SIPO**
  This chapter covers the basic topologies of Serializer (PISO) and Deserializer (SIPO).
- **Chapter 4: High Speed and low power digital circuits**
  This chapter covers different logics to design high-speed low-power custom analog blocks or digital circuits and also different topologies of Flip-flops.
- **Chapter 5: Design of Serializer and Deserializer**
  This chapter discusses the proposed design and implementation of Serializer (PISO) and Deserializer (SIPO). All the different blocks would be discussed along with internally generated load signals.
- **Results and Discussion**
  This chapter shows the results of the proposed Serializer and Deserializer.

- **Conclusions**
  This chapter concludes all the findings and contributions of this work and discusses possible future work.

Chapter 2

Architecture of High Speed SerDes

This chapter covers the basic fundamental concept and architecture of high-speed SerDes. Section 2.1 describes and compares the two basic data transmission methods, parallel and serial data communications. Section 2.2 discusses a brief review and basic features of high-speed SerDes. Section 2.3 majorly focuses on transmitter blocks. Then, section 2.4 discuss receiver blocks.

2.1 Data Transmission

Data transmission is the transfer of data between two chips on the same board or inter circuit board. Nowadays there are two common methods of data transmission: parallel communication and serial communication. As technology advanced and data rate increased to multi gigabits range, serial communication has become a preferred method compare to parallel communication, due to less number of pin count, area and power.

2.1.1 Parallel Transmission

Parallel communication comprises multiple data lines that can carry multiple data bits simultaneously. The data from the transmitting port will be sent to the corresponding receiving port. Therefore for n-bit parallel communication, 2n ports and n wires are needed as shown in Fig.2.1[3] .

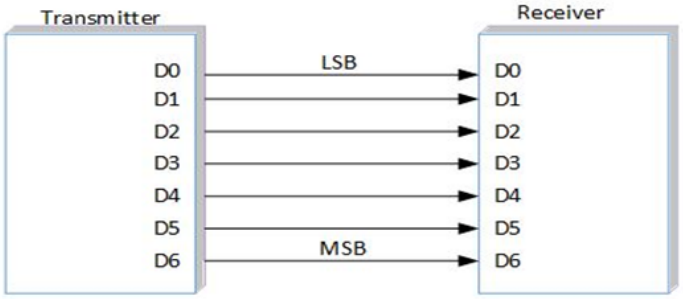

FIGURE 2.1: Parallel Communication Interface

Parallel communication is easy to implement but it occupies lot of area. Data skew is common in parallel communication, which is caused by ISI (Inter-symbol Interference), noise and output impedance mismatch. Because of these effects arrival time of different data can vary according to specifications (PCI-e, SATA, USB, TBT, DP, HDMI, and M-PHY). Cross talk is when one channel create an undesired effect on another channels due to conductors and impedance of channel. Parallel communication suffers from the cross talk because of multiple channels as shown in above figure 2.1.

2.1.2 Serial Communication

Serial Communication sends data one bit at a time over a single channel. Serial communication needs fewer transmitting and receiving port and channel as shown in figure 2.2[3] . Parallel data first converted to serial data using Serializer. Therefore, it requires an extra circuit design.

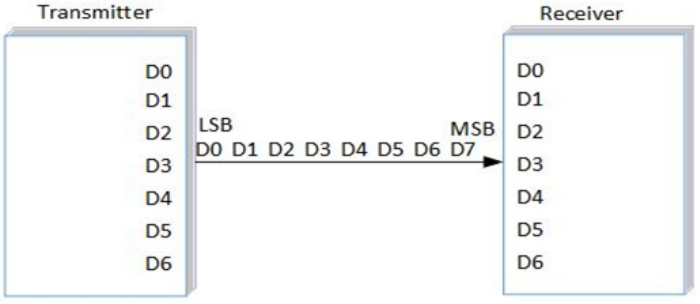

FIGURE 2.2: Serial Communication Interface

Serial communication most commonly use in chip to chip data transfer because of its advantage over parallel communication. A fewer number of pins are required in serial communication therefore, the cost can be very low compared to parallel communication. It has no issue of data skew and cross talk because of only one channel to transmit data.

2.2 High Speed SerDes

A device called SerDes (Serializer/Deserializer) allows transfer data serially and it is playing a very important role in high-speed applications. It is capable of converting parallel data to serial and vice versa. A simplified model is shown in figure 2.3[4] .

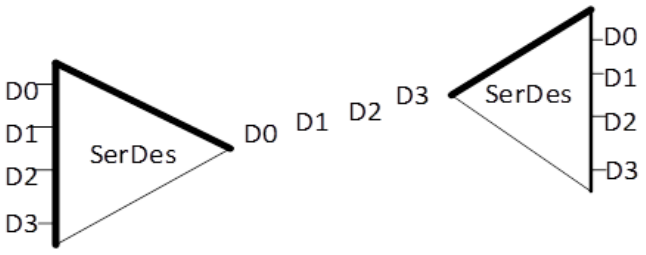

FIGURE 2.3: SerDes Link

A basic block diagram of high-speed SerDes is showing in figure 2.4. Firstly n bit parallel data will convert to serial data through Serializer. An equalizer is used to ensure good signal integrity of serialized data, which will then be driven into a differential channel. On the receiver side, serial data is received by the receiver which is fed in CDR (clock data recovery circuit) which generates the clock, after the recovery and equalization circuitries, data is finally proposed to Deserializer which will convert back parallel data into serial data.

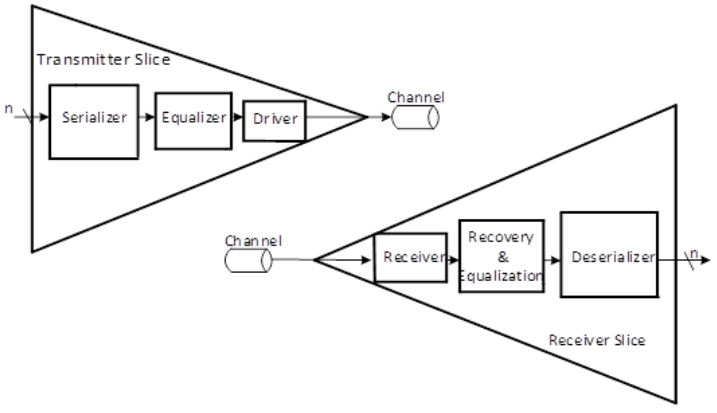

FIGURE 2.4: High-Speed IO Link Block Diagram

2.3 Transmitter

The purpose of the transmitter to send high-speed data stream through a channel to a receiver by delivering high-speed voltage swing at the pins of the transmitter. Figure

2.5[4] shows a typical block diagram of the transmitter block. Four major functions are performed in the transmitter, parallel to serial conversion, clock generation, feed-forward equalization, and line driving. The transmitter input is n bit parallel data stream, following serializer block which will convert n bit low-speed parallel data into high-speed serial data. PLL will generate a multi-phase clock based on an external reference clock. FFE in a transmitter is used to cancel ISI (Inter-symbol interference) effect. The driver is used to match the output impedance of the transmitter to channel (Transmission line) impedance.

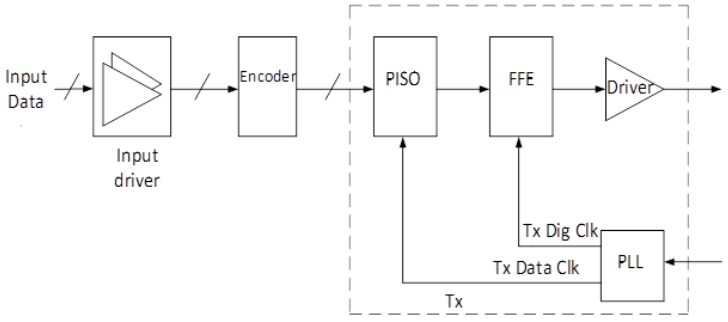

FIGURE 2.5: Block Diagram of Transmitter

2.3.1 Phase Locked Loop (PLL)

A phase lock loop (PLL) is working as a timing or clock generator in high-speed link System, it generates high-frequency clock from the low-frequency reference clock. A PLL is a circuit synchronizing an output signal (generated by an oscillator) with a reference or input signal in frequency as well as in phase. In the synchronized often called locked state. Under the locked condition, the phase error between the oscillator output signal and the reference signal is zero, or it remains constant. If the phase error increases, a control mechanism acts on the oscillator in such a way that the phase error again reduced to a minimum. Because of such a control system, the phase of the output signal locked to the phase of the reference signal. This is the reason why it is referred to as a phase-locked loop. The main components of the PLL are Phase Detectors (PD), Charge Pump (CP), Loop Filter (LPF) and Voltage Controlled Oscillator (VCO) as shown in the figure 2.6[5].

2.3.2 Serializer

Serializer operation perform the parallel-to-serial conversion as shown in the figure 2.7[6].A simplified schematic of a 2:1 Serializer is presented here as an example.

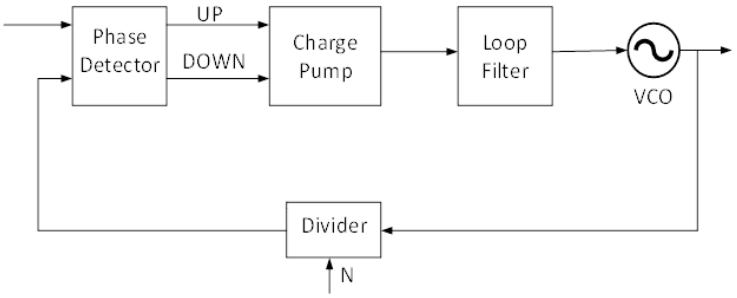

FIGURE 2.6: Block Diagram of PLL

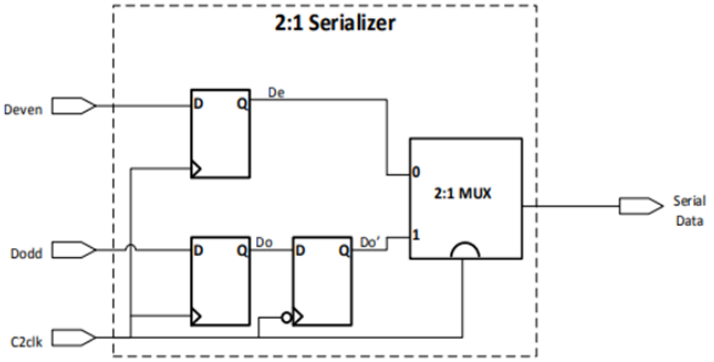

FIGURE 2.7: 2 to 1 Serializer Block Diagram

Assuming the two bits of parallel data, D_{even} and D_{odd} are time-aligned into serializer and are synchronized to the half-rate C_2 clock signal. The parallel D_{even} and D_{odd} signals are captured by the first two D- latches, which create the D_e and D_o outputs on the rising edge of the C_2 clock signal. The D_o' signal is generated by resampling the D_o signal on the falling edge of the C_2 clock signal. The select input of the 2:1 MUX is controlled by the C_2 clock signal, so that when the clock is low De input signal is selected, and when clock is high D_o' is selected.

2.3.3 Transmitter Equalization (FFE)

FFE in transmitter is used to cancel Inter-Symbol Interference effect. For low loss channel generally 2-tap FFE (one main cursor and one post cursor) is sufficient. For higher loss channels, 3-tap FFE (one pre cursor, one main cursor and one post cursor) is often required. PCIe specifications required 2-tap FFE for Gen1 and Gen2 and 3-tap FFE for Gen3 A 3-tap FFE is typically expressed as: Where c_i is the tap coefficient and a_i is the data stream. Data a_{n+1} is commonly referred as pre cursor, a_n main cursor and a_{n-1} post cursor. Therefore, the tap coefficients associated with them are called pre-cursor tap coefficient (c_{-1}), main cursor tap coefficient (c_0) and post-cursor tap

coefficient (c_{+1}), respectively. Data stream can either +1 or -1, whereas c_0 is usually a positive number, c_{-1} and c_{+1} have opposite sign of c_0 , and $|c_0|$ + $|c_{-1}|$ +$|c_{+1}|$ = maximum swing. Since a_i is either +1 or -1, there are only eight possible TX output levels, as Illustrated in Figure 2.8[7].

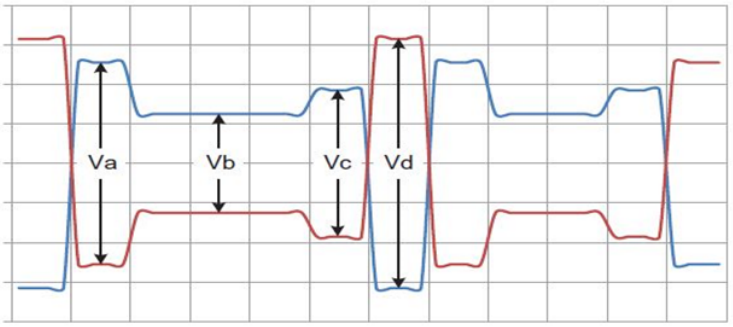

FIGURE 2.8: Transmitter Output Levels

Where Vd is referred as high voltage swing which occurs when the data pattern is (-1, +1, -1) or (1,-1, 1). Vb is DC pattern or low frequency pattern which will occur when data pattern is (-1,-1,-1) or (1, 1, 1). Va is post cursor amplitude which will occur when data pattern is (-1, 1, 1) or (1, -1, -1) and Vc is pre cursor amplitude which will occur when data pattern is (-1, -1, 1) or (1, 1, -1). PCIe specification defines:

2.3.4 Driver

The transmitter must generate accurate voltage swing on the channel according to the specification of protocols. Either current or voltage mode driver is used to generate accurate voltage swing and impedance matching, as shown in fig 2.9. Current mode drivers typically steer current close 20mA to generate voltage swing of 500 mV. Driver output impedance is matched through a resistor which is in parallel with the high-speed current switch. Current mode driver mostly used in high voltage swing, it uses Norton equivalent parallel termination, which is easy to control output impedance. Whereas a voltage mode driver uses the Thevenin-equivalent series termination. Voltage mode driver to times current use to generate the same voltage swing, therefore it is less power consuming to current mode driver but it is difficult to generate impedance matching with voltage mode driver[8].

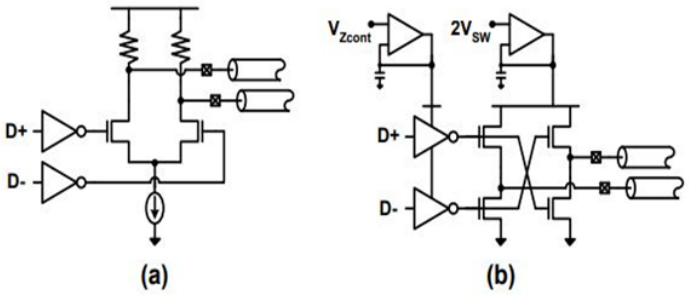

FIGURE 2.9: (a) Current Mode Driver (b) Voltage Mode Driver

2.4 Receiver

The main function of receiver is to perform equalization and recover data and clock. Fig 2.10[3]

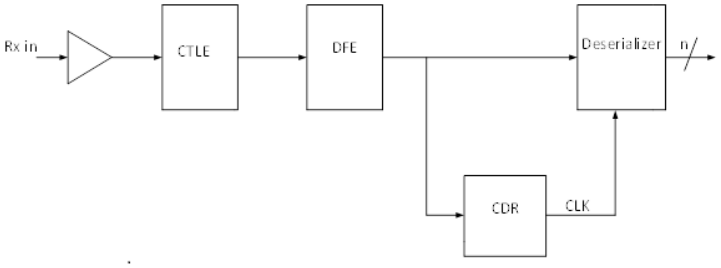

FIGURE 2.10: Receiver Block Diagram

2.4.1 Continues Time Linear Equalizer (CTLE)

The CTLE is Continues Time Linear Equalization. Channel is acting as a low pass filter. This means that the magnitude of the low frequencies component of a signal will stay the same but the high frequencies will be lessened. If we try to send a high-frequency signal, it will be hugely attenuated and it is impossible to detect it on the receiver side. The principle of CTLE is to provide a high pass filter at the receiver to turn down low frequencies and amplify high frequencies.

2.4.2 Decision Feedback Equalization (DFE)

The forward filter generally implemented to cancel pre-cursor ISI. DFE can remove only post-cursor ISI. So practical DFE consists of a feed-forward filter that can cancel

precursor ISI and provide some eye-opening to a slicer. The slicer decides each symbol and sends this symbol data to the feedback filter[1].

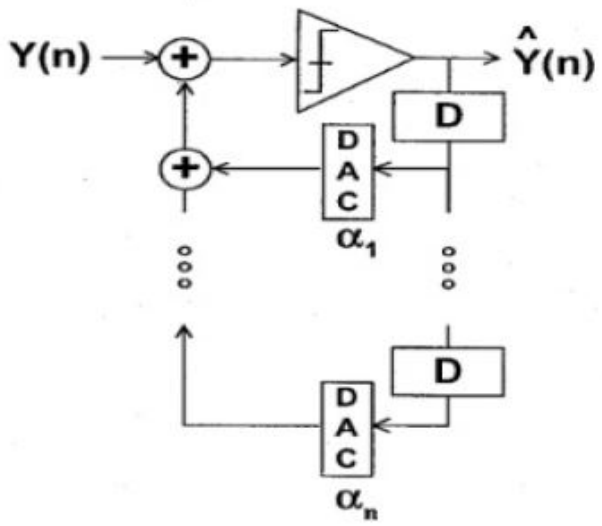

FIGURE 2.11: Decision Feedback Equalizer[1]

Decision Feedback Equalizer removes post-cursor ISI, without increasing noise, to completely open the eye and reduce jitter. Least-mean-square (LMS) and minimum mean square error (MMSE) algorithms are frequently used to obtain decision feedback equalizer tap coefficients. The number of taps is determined keeping in mind implementation issues and is generally a trade off between area, performance and power consumption and implementation possibility.

2.4.3 Clock and Data Recovery (CDR)

CDR circuits in High-Speed serial IO are similar to PLL. The reference signal in PLL is periodic, but input data in the CDR circuit is random. For that reason, the phase detector and PFD which are used in PLL do not work for the CDR circuit at all. Problem is that they get the wrong impression about the missing edges as frequency errors. So PDs and PFDs play an important role in the CDR circuit. In high-speed IO, the phase detector is mostly logic circuits[9].

2.4.4 Deserializer

Deserializer operation is to perform the serial-to-parallel conversion as shown in the figure 2.13[6]. A simplified schematic of 1:2 is presented here as an example.

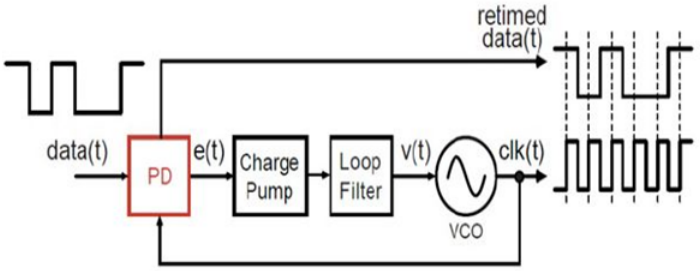

FIGURE 2.12: Clock and Data Recovery Block Diagram

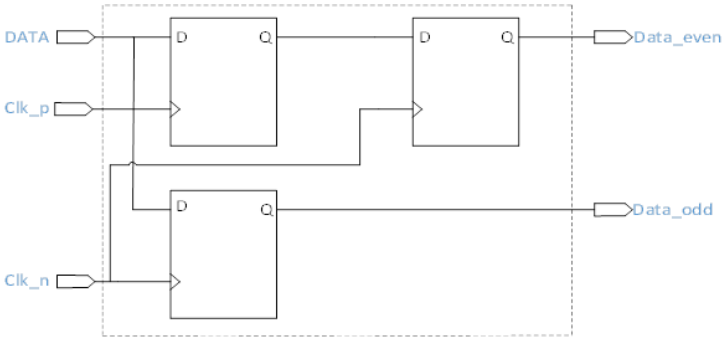

FIGURE 2.13: 1 to 2 Deserializer Block Diagram

First serial data will send to two D flip flops, which will capture the data both at rising and falling edge of the clock. By this we have two different even and odd data streams, but even data comes early. So to retime even data with odd data it will be further delay by falling edge of the clock. So we have two parallel data streams.

2.5 Summary

Detail description of basic fundamental concepts and architecture of SerDes has been given in this chapter. Further, basic data transmission methods have been compared. We have also discussed all the blocks of the transmitter and receiver.

Chapter 3

Different Architecture of PISO and SIPO

As discussed in previous chapter Serializer and Deserializer converts parallel data to serial data and serial data to parallel data respectively. This chapter covers different architectures of PISO (Serializer) and SIPO (Deserializer).

3.1 Architecture of PISO

Power, Area and Speed are main specifications of High Speed serial communication. Depending upon the requirement of these specifications, there are two architecture of PISO.
i. Single Phase Clock based PISO
ii. Multi-Phase Clock based PISO

3.1.1 Single Phase Clock based PISO

Figure 3.1[10] shows the block diagram of a Single-phase clock based PISO. PISO serializes 8 bit parallel data into serial data. The first 8 bit parallel data will convert into 4 bit parallel data using the clock div by 8 after that 4 bit parallel data will convert into 2 bit parallel data using clock div by 4, finally, 2 bit parallel data will convert into serial data using clock div by 2. Here each 2:1 conversion consists of two D flip-flops following by a mux. So for converting 8 bit parallel data into serial data, fourteen D flip flop and seven mux are used but only a single-phase clock is used so it will not be very power-hungry.

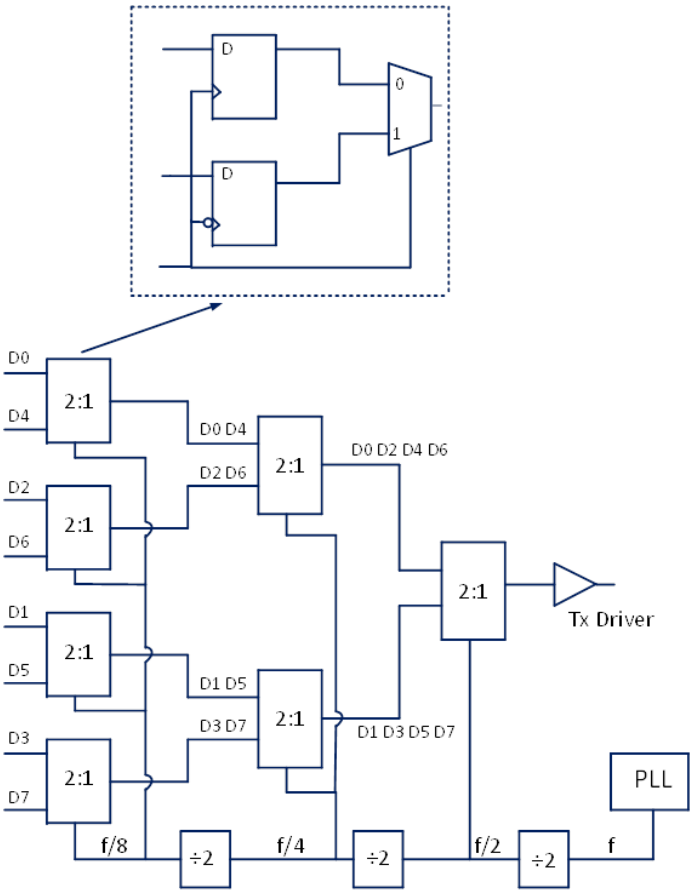

FIGURE 3.1: Single Phase Clock based PISO

3.1.2 Multi Phase Clock based PISO

Figure 3.2[11] shows the block diagram of a Multi-phase clock based PISO. PISO serializes 8 bit parallel data into serial data. The first 8 bit parallel data will convert into 4 bit parallel data using the clock div by 8 after that 4 bit parallel data will convert into serial data by using a multi-phase clock. Here first divider block will generate 0, 90, 180, 270 phase clocks, first LSB bit D0, D1, D2 and D3 will serialize with respective phase clock, then MSB bits will be serialized. Here each 2:1 conversion consists of two D flip-flops following by a mux. So for converting 8 bit parallel data into serial data, eight D flip flop and four mux are used, so it will save the area compare to single-phase clock base PISO but to generate different clock phases it will be more power-hungry.

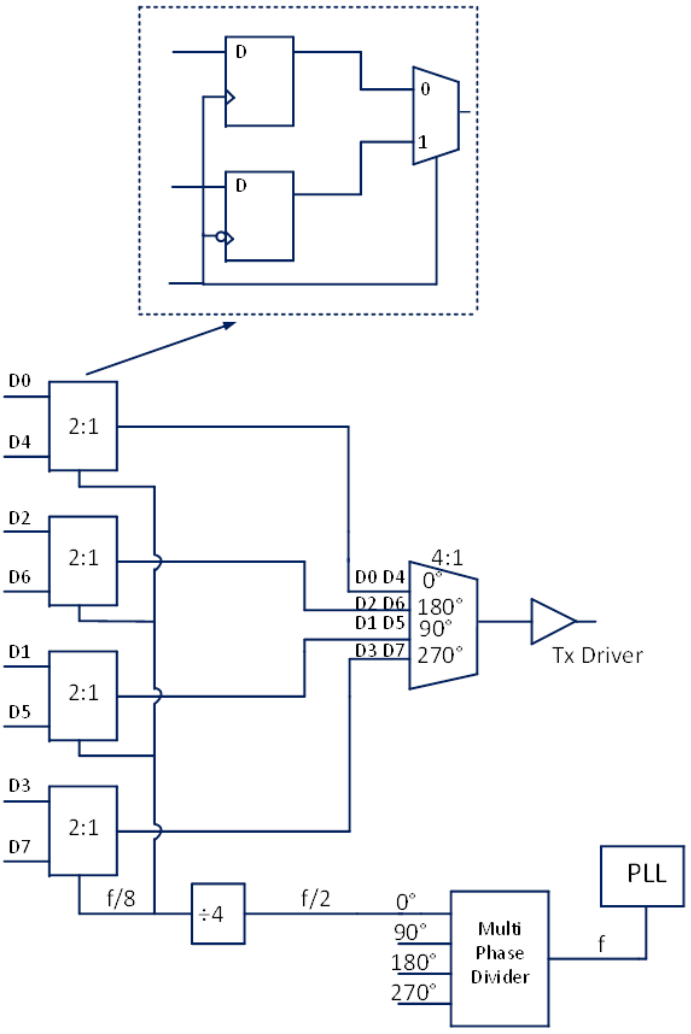

FIGURE 3.2: Multi Phase Clock based PISO

3.1.3 Comparison between Single and Multi Phase Clock based PISO

As previously discussed, there are two type of architecture of PISO. One is Single Phase clock based PISO and second is Multi-phase clock based PISO. Each of them has their own challenges in designing. Some of them are listed below:

Single Phase Clock based PISO

- Single Phase Clock based PISO has a greater number of D flip-flops and MUX, so these components will increase the overall area of the design.
- Single Phase clock based PISO will operate at higher clock frequency compare to other architecture, so it will increase the complexity of PLL to generate high frequency of clock.

- Single Phase Clock based PISO is simple binary stage architecture where in each stage you need divide by 2 clocks, so the number of dividers is more in this architecture.

Multi Phase Clock based PISO

- Multi-Phase clock based PISO has a smaller number of D flip-flops and MUX compare to single phase clock-based architecture, so it will reduce the overall area of the design.
- In this architecture, a divider is needed which generates different phases of clock so it will increase the complexity of overall architecture.
- Multi-Phase clock based PISO is more power consuming compare to single phase clock-based architecture.

3.2 Architecture of SIPO

As discussed in previous chapter, in receiver side, a SIPO block is needed to reduce the bit rate of back end circuit to perform the further signal processing. It is straight forward that SIPO has tree type structure. There are two Architectures of SIPO.

I. Single Stage Architecture based SIPO

II. Binary Stage Architecture based SIPO

3.2.1 Single Stage Architecture based SIPO

Figure 3.3[6] shows the block diagram of Single Stage Architecture based SIPO. In single-stage architecture, high-speed serial data will directly convert into parallel data. The figure shows a serial data first go into 1:8 DE multiplier (DMUX), after that to synchronized data it will pass through low-speed D flip-flop. Single-stage architecture is easy to implement but it is slow due to large parasitic capacitance at the converging node.

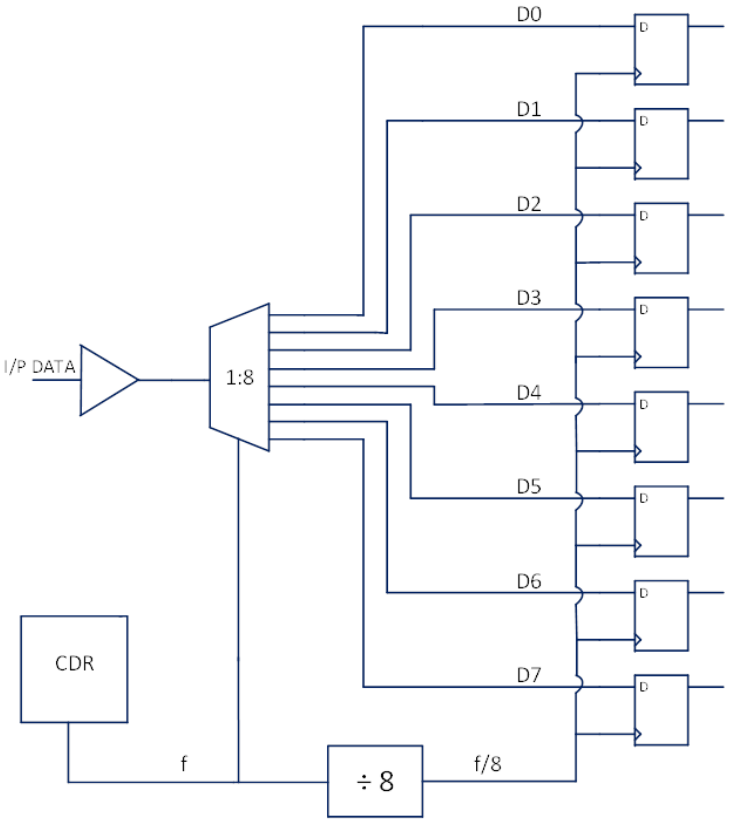

FIGURE 3.3: Single Stage Architecture based SIPO

3.2.2 Binary Stage Architecture based SIPO

Figure 3.4[6] shows the block diagram of Binary Stage architecture based SIPO. In single-stage SIPO, speed is low due to high parasitic capacitance at converging node. To solve this problem binary stage architecture is generally used. Here, First serial data converts into two parallel data streams after that each parallel stream further converts into two parallel data streams. So it follows the pattern 1:2:4:8 to convert serial data in parallel data. Binary Stage architecture is faster in among all architecture due to less parasitic capacitance in each stage but the area will be more in comparison to single stage architecture.

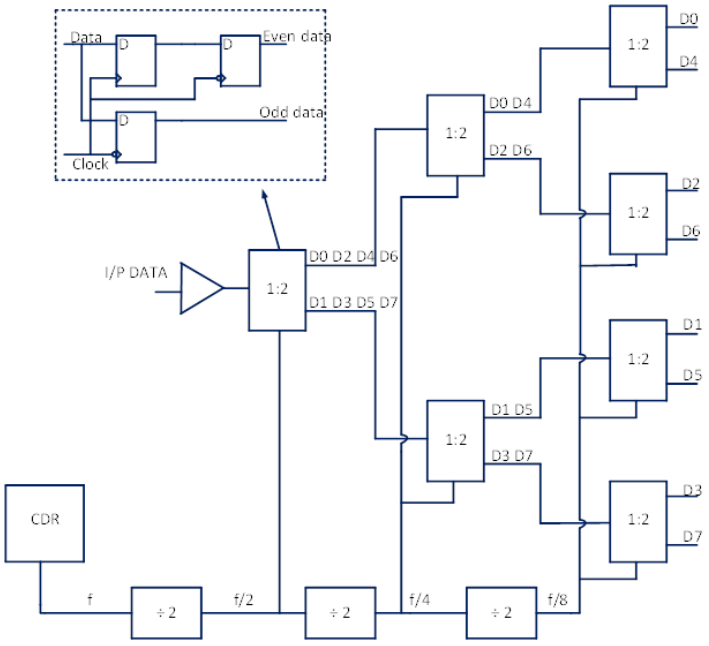

FIGURE 3.4: Binary Stage Architecture based SIPO

3.2.3 Comparison between Single and Binary Stage architecture based SIPO

As previously discussed, there are two type of architecture of SIPO. One is Single Stage Architecture based SIPO and second is Binary Stage Architecture based SIPO. Each of them has their own challenges in designing. Some of them are listed below:

Single stage Architecture based SIPO

- Single Stage Architecture based SIPO is easy to implement but it is slow due to large number of parasitic capacitance at converging node.
- Single Stage Architecture based SIPO has large number of D flip-flop at final stage, so divider will have to generate high driving strength clock so that it will be capable to drive all the flip-flops.
- Single Stage Architecture based SIPO has no interval stages of converting serial data into parallel data, if there is missing of any data in between than it will become difficult to debug that error after manufacturing the chip.

Multi stage Architecture based SIPO

- Binary Stage Architecture based SIPO has a higher number of stages, so overall area is more compare to other architecture.
- It has a higher number of dividers to generate clock for each stage, hence it increases the complexity of the design for synchronizing each stage data from the previous stage data.

3.3 Summary

In this chapter basic topologies of Serializer and Deserializer has been discussed also comparison in different topologies has been given according to specification requirement.

Chapter 4

High Speed & Low Power Digital Circuits

This Chapter covers different logics to design high speed & low-power custom analog blocks or digital circuits. Section 4.2 covers timing and delay definitions of Flip-Flops and latches. Section 4.3 covers different topologies of Flip-Flop.

4.1 Different Logic to design Digital Circuits

In a high speed SerDes transceiver, not only decoder & encoder are digital circuits but also phase detector and frequency dividers are digital ones. Some of these works at very high frequency. Therefore, these circuits should be sufficiently fast and low power consuming. The main factor that limits the speed of a circuit is the capacitive load and parasitic capacitance of the devices. The voltage change on a capacitor is be written as

$$\Delta V = \frac{\Delta Q}{C} = \int_0^{\Delta T} i(t) = \frac{I_o \Delta T}{C} dt$$

Where C is the total load capacitance, ΔT is the time it takes to change voltage across the capacitor with amplitude of ΔV, to make circuit fast ΔT should be very small.

$$\Delta T = \frac{\Delta V C}{I_o}$$

It is quite straight forward how to make circuit faster, to reduce the voltage swing , or to reduce the load capacitance or to increase the charging or discharging current I_o. A large logic family exploit these fundamentals to make circuit faster. There are two basic logic

1) CMOS Logic
2) CML Logic, to make circuit faster and less power consuming.

4.1.1 CMOS Logic

CMOS logic is by the far most commonly used type of logic circuit. Despite of High speed, CMOS logic is still used in some high speed SerDes transceiver. The reason are technology scaling down, reduced power supply voltage and robustness of CMOS logic. CMOS logic has pull-up network and a pull-down network. At any time except transitions, either pull-up network is turn on to pull the output voltage to power supply voltage or pull-down network is on to pull down output voltage to ground. Since both the network cannot be turned on simultaneously except during transitions, Therefore, CMOS logic consumes zero static power. Figure 4.1[12] shows a circuit diagram of rail to rail CMOS inverter.

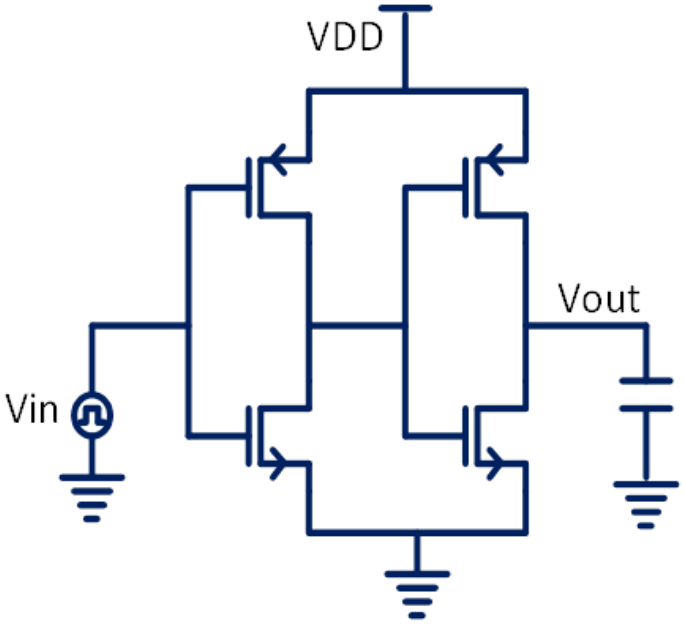

FIGURE 4.1: Rail to Rail CMOS inverter

The average power consumption of any inverter can be estimated by

$$P = V_{dd} I \Delta T f = C_{Total} V_{dd}^2 f$$

Some conclusion can be drawn from the basis of simple analysis. Firstly, CMOS logic has to drive large capacitance i.e parasitic cap, load cap (Pull-up or Pull down), these greatly increases the power consumption according to the equation. CMOS logic consumes more power at high frequencies. CMOS logic is sensitive to common mode noise because it

is not differential. Therefore, high-speed CMOS design favors to current mode logic (CML).

4.1.2 CML Logic

CML logic is based on differential pairs as shown in figure 4.2[12]. We can replace PMOS transistor with resistance. Therefore, it is faster than CMOS logic. It is fully differential, therefore, it has excellent immunity to common mode noise. When the input voltage is sufficiently large one of the branches can be switched off, while the other takes all the tail current I_o, The minimum input voltage can be derived by following equations, The voltage swing is the product of the load current and the tail current. Therefore, it is possible to reduce the voltage swing to improve the speed of circuit.

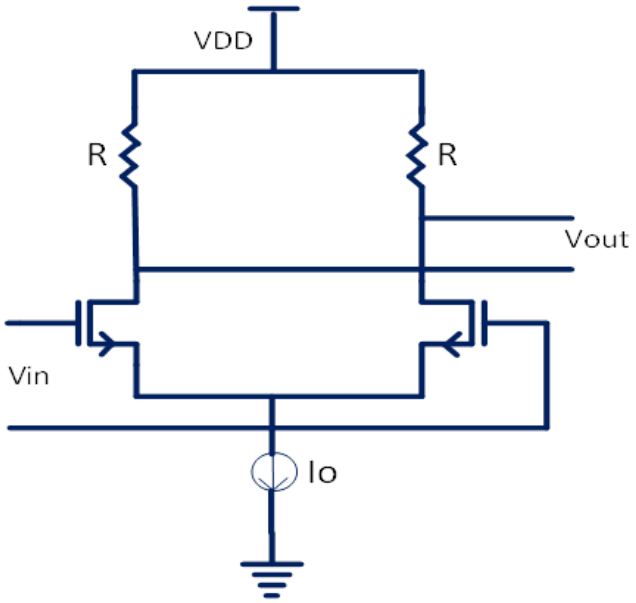

FIGURE 4.2: CML Logic Differential Pair

The power consumption of CML inverter can be simply estimated by Obviously CML logic consumes static power, the power consumption is independent of frequency. Therefore, CML logic is suitable for high frequency applications in terms of speed and power consumption.

4.2 Timing and delay definitions of flip-flops

Building a sequential circuit require memory elements which read a value, save it for some time even if the value has changed. A Boolean logic change its output value as input

change. A memory element has some internal memory and extra circuitry to control it. Internal memory is controlled by clock input, memory element reads input data value according to instructed by clock and stores that in its memory. Memory elements are divided into two major types. Figure 4.3[12][13] shows the difference between positive edges triggered flip-flop and level high latch. As it can be seen from the waveforms latches changes its value when clock is high while flip-flop changes only at the transition of clock from low to high.

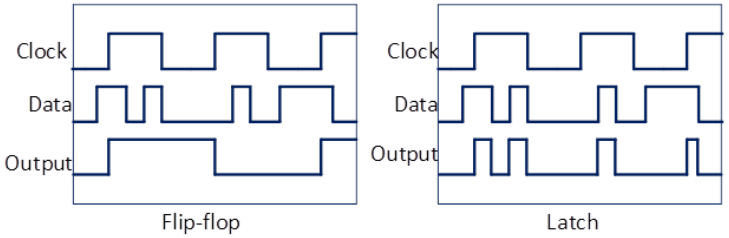

FIGURE 4.3: Timing Diagram of Flip-flop and Latch

The performance of flip-flop is qualified by three important timings and delays: propagation delay (clock-to-output), setup time and hold time. They reflect in the performance of flip-flop.

4.2.1 Propagation delay

Propagation delay (clock-to-output) is the time delay after arrival of clocks active edge the output is considered stable. Clock-to-output equals the time it takes for the output to change after the occurrence of clock. Propagation delay differs for the low-high transition and high-low transitions. So propagation delay of the flip-flop is by definition maximum value of these two delays.

4.2.2 Setup Time

Setup Time is the amount of time the synchronous input (D) must show up, and be stable before the capturing edge of clock. This is so that the data can be stored successfully in the storage device. As setup time may differ for low-high transitions and high-low transitions, setup time is by definition maximum of the values obtained for low-high and high-low transitions.

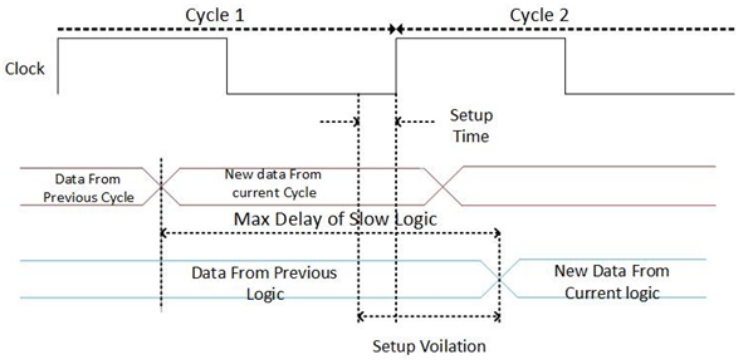

FIGURE 4.4: Timing Definition of Setup Time

4.2.3 Hold Time

Hold Time is the amount of time the synchronous input (D) stays long enough after the capturing edge of clock so that the data can be stored successfully in the storage device. Hold time is by definition maximum of the values obtained for low-high and high-low transitions:

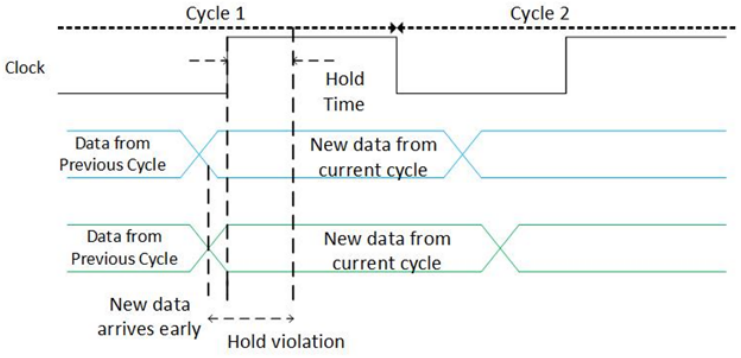

FIGURE 4.5: Timing Definition of Hold Time

The definitions of setup time, hold time and propagation delay are considered as independent variables. However what happens in reality shows that these parameter are not independent to each other. As it shown in figure 4.6[14] propagation delay increases as data arrives later. When data arrival time is very close to clock edge, clock-to-output delay increases drastically as shown in figure and it is possible that flip-flop function will be changed and it will enter in unstable state called meta-stable state. There are several approaches for setup time definition, one of them is Setup time is the time period before clock edge which causes 5% increase in clock-to-output delay.

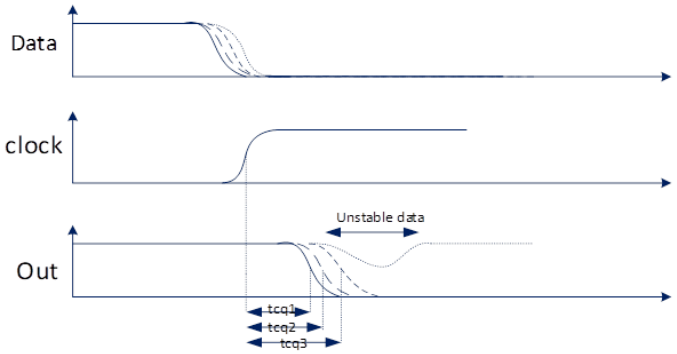

FIGURE 4.6: Real Timing Definition of Propagation Delay

4.3 Flip-flop topologies

There are different topologies of flip-flop depends upon the speed, power and area requirement. In this section some of the topologies will be discussed.

4.3.1 Transmission gate latch based master slave flip-flop

This flip-flop is realized by using two transmission gate based latch operating on complimentary clocks as shown in figure 4.7[12][13][15]. Different variety of transmission gate are available. For example we can remove feedback transmission gate or only nmos based transmission gate can be used. Here when clock is low data will be stored in master latch, as it makes transition from low to high data will pass to the output. If the complimentary clocks have different rise and fall time than there will be glitches at the output.

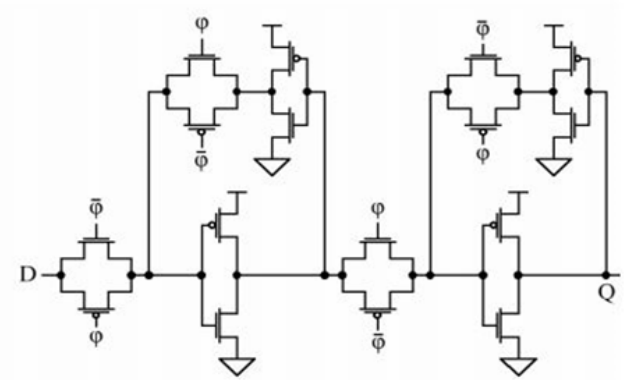

FIGURE 4.7: Transmission gate Flip-flop

4.3.2 C^2MOS master-slave flip-flop

By eliminating the transmission gate followed by inverter, a modified flip-flop is constructed without loss of functionality as shown in figure 4.8 [16]. This structure is called C^2MOS master slave flip-flop because of clocked inverters used in it. Here inverter will changes its value according to the complementary clock value. This structure is fastest in commonly used circuit family because of low input capacitance and no contention during switching. Unlike the transmission gate flip-flop, this structure is insensitive overlap of the clocks.

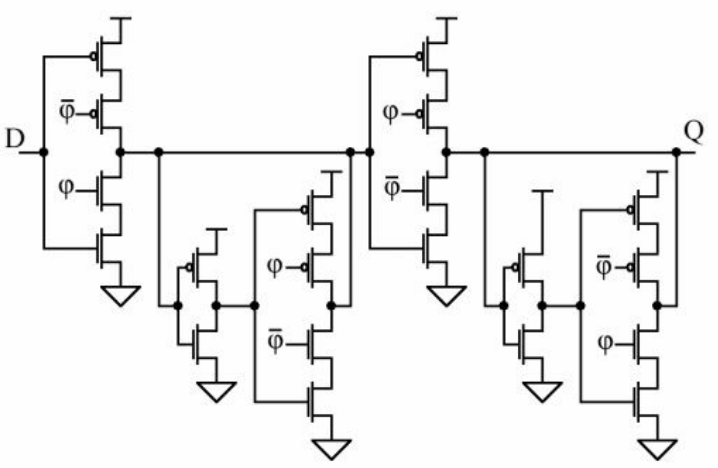

FIGURE 4.8: C^2MOS Flip-flop

4.3.3 TSPC flip-flop

In above topologies complementary clocks are used so there is a problem of clock skew between complementary clocks. Hence to overcome this problem single phase clock based flip-flops are used. True Single Phase Clock (TSPC) flip-flop has advantage of single clock distribution, small area and no clock skew. The basic TSPC latches can be implemented in many ways. Figure 4.9 [17] shows implementation of nine-transistor positive edge triggered flip-flop. Due to single phase clock and less area, these are very high speed but there is also static power consumption due to this at very high speed these are very power hungry.

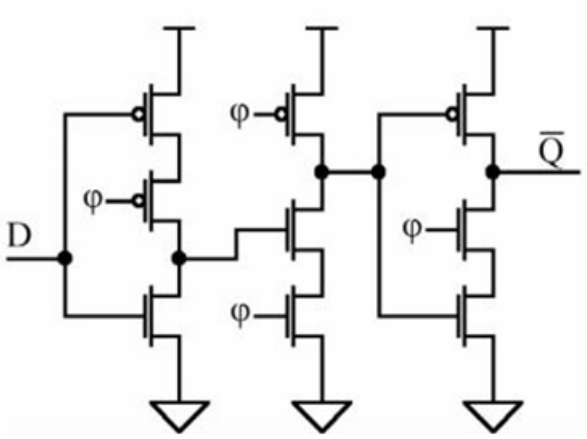

FIGURE 4.9: TSPC flip-flop

4.3.4 Relevance and Comparison of different topologies of Flip-flop for SerDes Design

As previously discussed, there are different topologies of Flip-flop. Each of them has their own pros and cons for SerDes design. Some of them are listed below:

Transmission gate latch-based master slave flip-flop

- Transmission gate latch-based mast slave flip-flop is low speed flip-flop, thats why it will not be used at high speed stage of SerDes.
- It has more requirement of Setup and Hold time, thats why it can be used at low speed stages of SerDes design.
- Since it is used in lower speed stage of SerDes so a great amount of power can be saved using this kind of flip-flops.

C^2MOS master-slave flip-flop

- C^2MOS master-slave flip-flop is moderate speed flip-flop. It will not be used in low and high-speed stages of SerDes design.
- It has better timing requirements compare to transmission gate latch-based master slave flip-flop; do it can used in middle stages of SerDes design.
- It has low input capacitance and no contention during switching. It is insensitive to overlap of the clocks.

TSPC flip-flop

- True Single-phase clock logic (TSPC) flip-flop is high speed flip-flop, thats why it will be used at high speed stage of SerDes design.
- It has advantage of single clock distribution; hence it has no clock skew.
- It has lesser number of transistors so area will be very less.
- It has more static power consumption compare to other topology, since it is used at high speed stage of SerDes design hence overall power consumption of design will be increased.

4.4 Summary

In this chapter different logic to design custom analog or digital circuits have been discussed. Further timing and delay definitions of flip-flop has been given also briefly discussed about different topologies of flip-flops.

Chapter 5

Design of Serializer and Deserializer

This chapter covers the block diagram of proposed PISO (Serializer) and SIPO (Deserializer) along with the detail description of each block. This PISO is designed to convert 40 bit parallel data into 28 Gbps serial data and SIPO is designed to convert 28 Gbps serial data into 40 bit parallel data.

5.1 PISO Architecture

Figure 5.1 shows the top-level architecture of proposed Serializer.. In this design both single and multi-phase architecture are used. Here 40-bit parallel data from PMA will be loaded in serializer block for every rising edge of the clock Tx_par_clk_in. Tx_par_clk_in and sync_en are used to enable the synchronizer block to generate sync_2p and sync_2n signals, which will be used to enable shift registers, dividers. Parallel to serial conversion happens successively 40→20→4→2→1. First stage of PISO is 40→20 which is low speed stage and implemented using single phase clock-based architecture. Second stage 20→4 is implemented using multi-phase clock and shift registers-based architecture. Final stages 4→2→1, which are high speed and most power consuming stages are implemented using Differential D flip-flop which has very low timing requirements and less power-hungry compare to other flip-flop topologies. Detail description of data flow and design of each individual block is further explained in following sections.

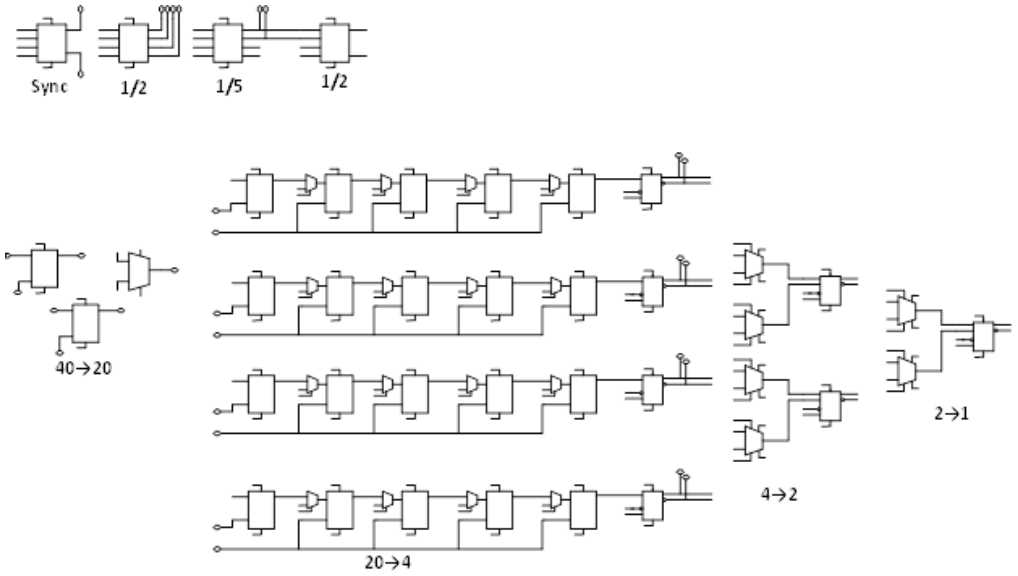

FIGURE 5.1: Top Level Block Diagram of Proposed PISO

5.1.1 PISO 40→20

Figure 5.2 shows the block diagram of first stage of PISO, Here 40 bit parallel data will be serialized into 20 bit serial data. Parallel data launched from PMA is sampled with internally generated clk40p clock, needed for synchronization between data and clock. MSB 20 bits are further sampled and delayed by falling edge clk40n. Synchronized LSB 20 bits passes through mux when clk40n is low and delayed MSB 20 bits passes when clk40n is high. 40 bit parallel data is serialized into 20 bits after first stage of PISO.

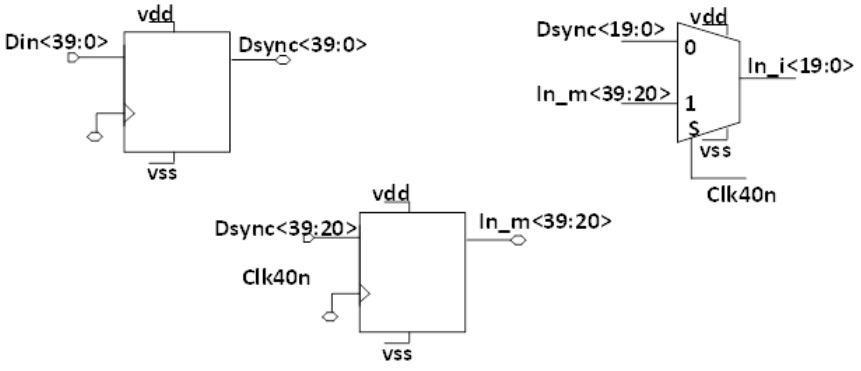

FIGURE 5.2: Block Diagram of PISO 40→20

5.1.2 PISO 20→4

PISO first stage converts 40 bit parallel data into 20 bit data. Figure 5.3 shows the block diagram of second stage of PISO. Here data is loaded on to 20 flip-flops, based on the selection signals which are phase shifted by 90°. Selection signals Sel_*dig*0°, 90°, 180°, and 270° are high for one clock cycle of Clkp_0°,90°, 180°, and 270° respectively and low for four clock cycles. When selection signals are high all the parallel data is loaded on the flops and when it is low data is pushed out of the shift register serially.
The 20→4 stage has two purpose:
1. Analog (from previous bit) or digital data select (with Sel_dig0), Each Mux_flop cells shift its previous cells data or (when sel_dig is up) select new data.
2. After the Mux_flop shift register array we have high speed latch which verify that the data is its appropriate phase. Also high speed latch is used because of higher time margins should be available to avoid post layout variations in setup and hold of flops and latches.

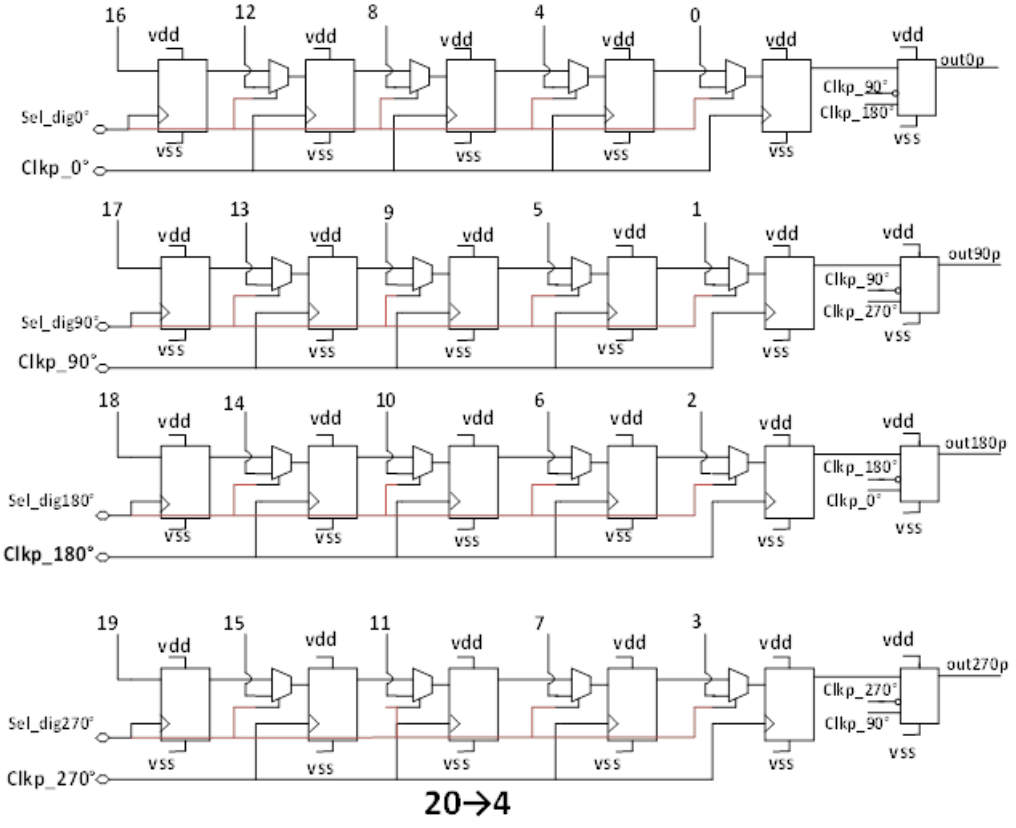

FIGURE 5.3: Block Diagram of PISO 20→4

5.1.3 PISO 4→2→1

Parallel data of 4 bits which are pushed out from shift registers are divided into even and odd signals using multiplexers as shown in figure 5.4 Even and odd bits are synchronized by high speed flip-flop using high speed clock coming from PLL. In whole Transmitter block final stage 2→1 conversion is inside the driver block, clock for 2→1 is coming from PLL in order to reduce the mismatch between the differential clocks and to reduce the final stage jitter.

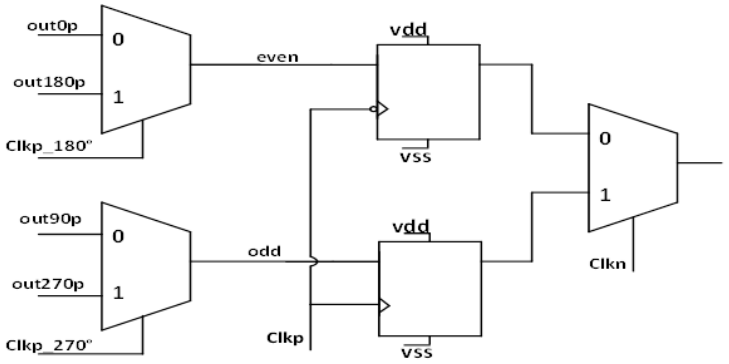

FIGURE 5.4: Block Diagram of PISO 4→2→1

5.1.4 Synchronizer Block

Figure 5.5 shows the block diagram of Synchronizer block. Here Tx_par_clk_in, Sync_en and high speed clocks clkp ,clkn are the inputs. Synchronizer is used to generate Sync_2p and Sync_2n signal, which will work as a reset signals for other blocks.

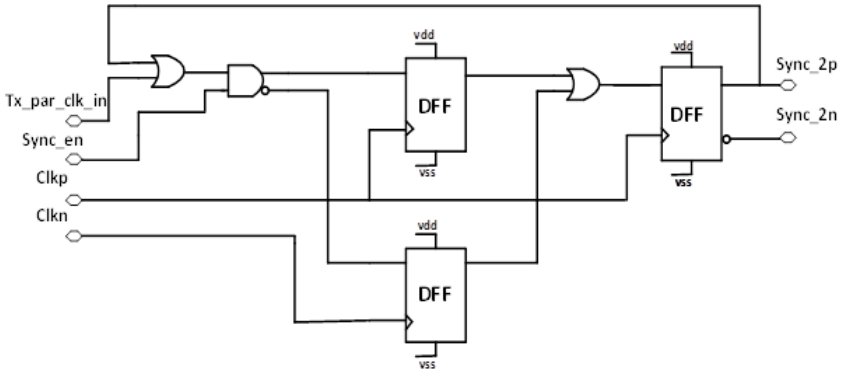

FIGURE 5.5: Block Diagram of Synchronizer

5.1.5 Quadrature phase divide by 2 clock generator

Figure 5.7 shows the proposed design of divide by 2 divider which will generate four phase 0°, 90°, 180°, and 270° of divided clock. Here differential latch based master slave flip-flop topology is used. First enable signal sync2p and sync2n enables block, after that high speed clkp and clkn will be applied. When clock is low master latch will give output and when clock is high slave latch will give output. To stabilize the data differential latch is used.

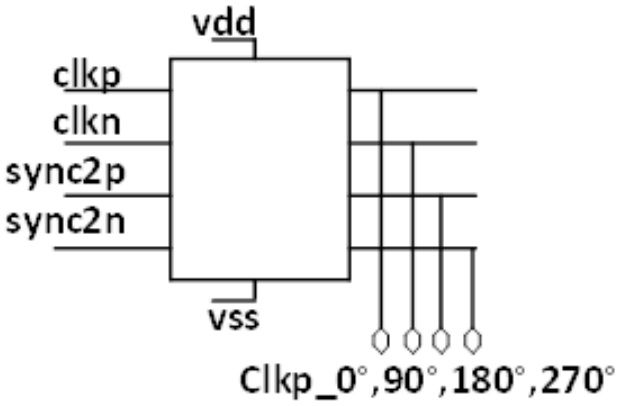

FIGURE 5.6: Block Diagram of Quadrature Phase divide by 2 Divider

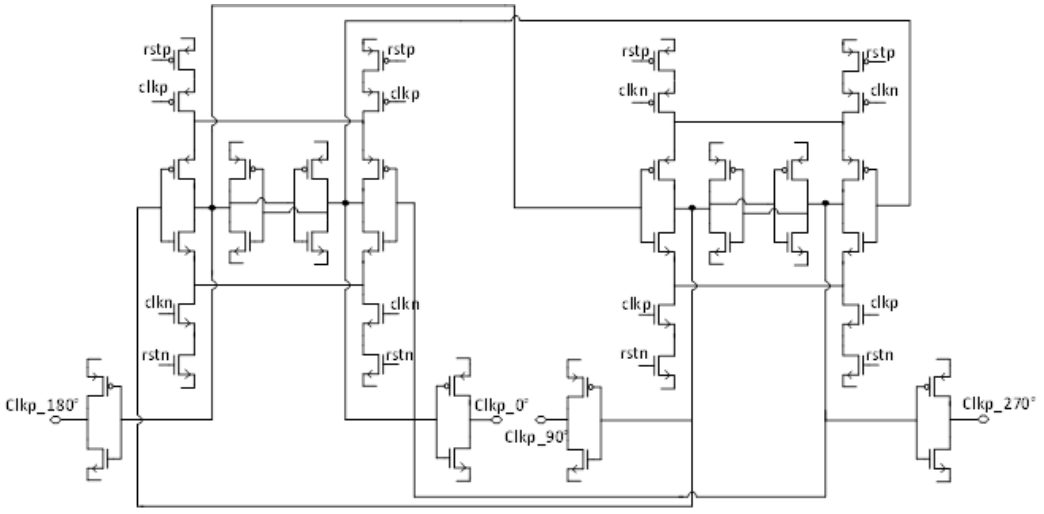

FIGURE 5.7: Schametic of Quadrature Phase divide by 2 Divider

5.1.6 Divide by five clock generator

Figure 5.8 shows the block diagram of divide by five clock generator. Here input clock is coming from quadrature phase divider. This block is used to generate divide by five clock with different duty cycles (50% and 20%). 50% duty cycle clock is further divided to generate divide by 40 clock for first stage, 20% duty cycle clock is used to generate selection signals for the shift registers.

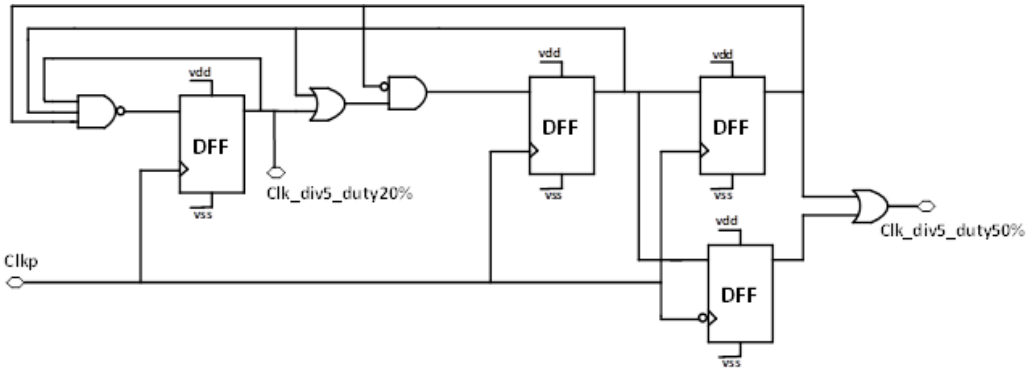

FIGURE 5.8: Divide by 5 clock generator

5.1.7 High Speed D flip-flop

Figure 5.9 shows the design of high speed positive edge triggered D flip-flop. The master and slave latch of D flip-flop are static. Master latch consists M_0-M_9 transistors, Transistors M_0, M_9 M_{10} and M_{19} must be strong enough to pull down and pull up corresponding nodes. Master latch is active when clkp is low, in this phase differential input is latched. In order to stabilize this output two cross coupled CMOS inverters are used to hold the data. When clkp is high slave latch is active, here also cross coupled CMOS inverters are used to hold the data. In order to increase the drive strength and speed of flip-flop strong CMOS inverters are used at the output.

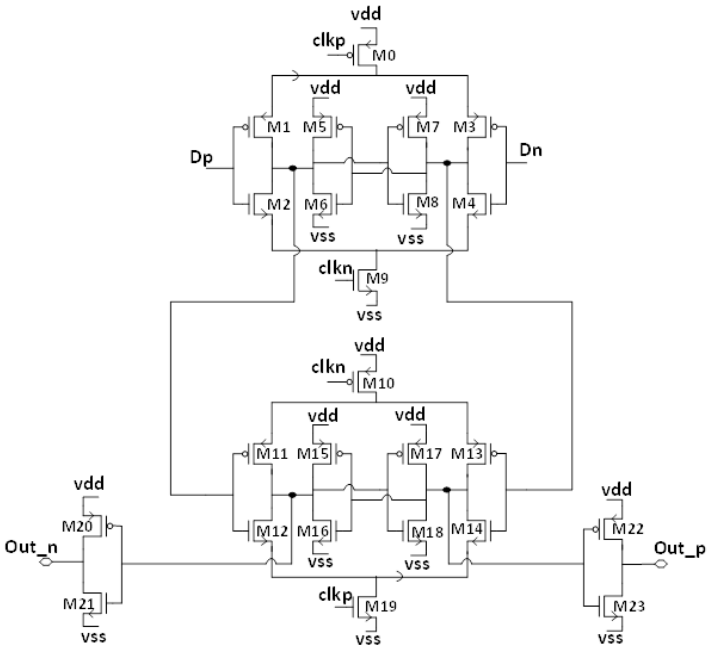

FIGURE 5.9: Schametic of High Speed D Flip-flop

5.2 SIPO Architecture

Figure 5.10 shows the top-level architecture of proposed Deserializer. Binary stage-based architecture has been used to convert high speed serial data into 40-bit parallel data. Here high speed data from CDR will be loaded in deserializer block for every rising edge of the clock Rx_par_clk_in. Rx_par_clk_in and sync_en are used to enable the synchronizer block to generate sync_2p and sync_2n signals, which will be used to enable shift registers, dividers. Serial to parallel conversion happens successively 1→2→4→8→40. Detail description of data flow and design of each individual block is further explained in following sections.

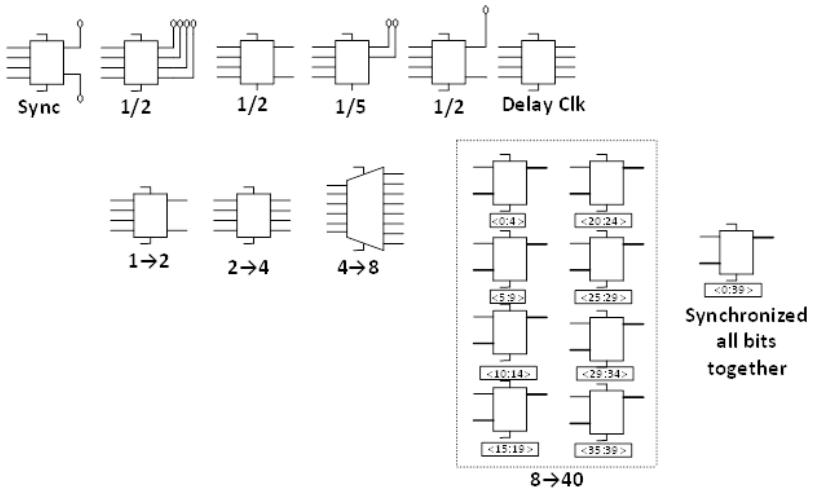

FIGURE 5.10: Top Level Block Diagram of Proposed SIPO

5.2.1 SIPO 1→2→4

Figure 5.11 shows the block diagram of first two stages of SIPO, Here high speed data will be deserialized into four bit serial data. High speed serial data launched from CDR first converted into two bit parallel data by the clock clkp and clkn. To synchronize both even and odd data even data is sampled further by clkn. In second stage of SIPO both even and odd data streams are converted in to four parallel data streams. Both data streams are sampled by the internally generated clock clkp_0° and clkp_180°. Here also for synchronization purpose, data which is sampled by positive edge triggered clock (clkp_0°) is further sampled by negative edge triggered clock (clkp_180°).

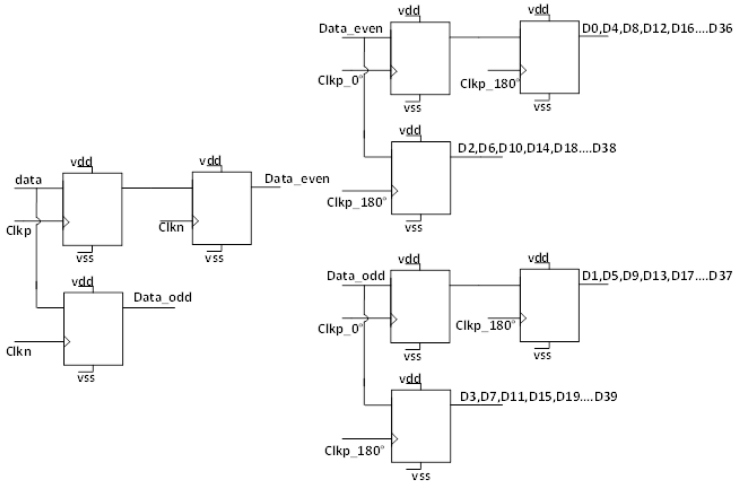

FIGURE 5.11: Block Diagram of SIPO 1→2→4

5.2.2 SIPO 4→8

SIPO first two stages convert high speed data into four parallel data streams, In third stage these streams further converted into eight parallel data streams by the internally generated clocks clkp4 and clkn4 as shown in figure 5.12. Here also for synchronization clkp4 data is further sampled by clkn4.

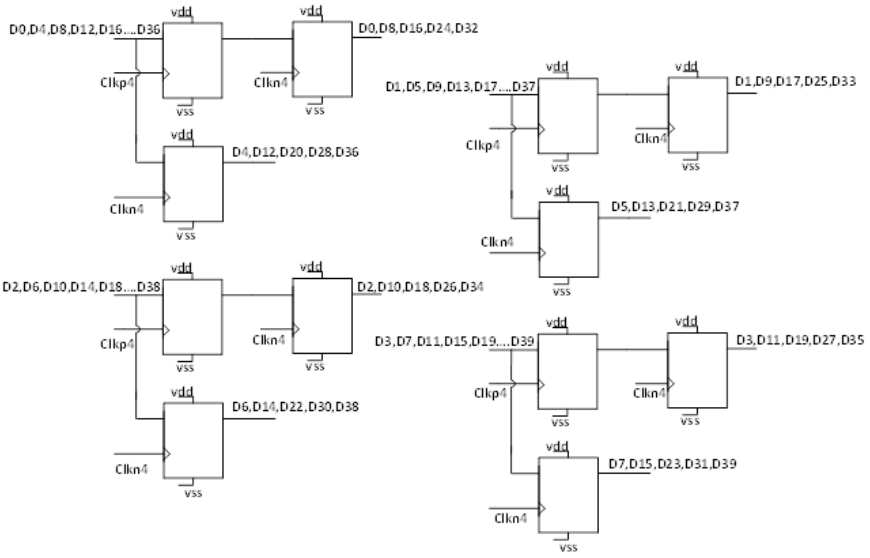

FIGURE 5.12: Block Diagram of SIPO 4→8

5.2.3 SIPO 8→40

Figure 5.13 shows the final stage of SIPO 8→40, Here 8 parallel data stream are converted into 40 bits parallel data. For each stream five flip-flops are used which will sample the data by internally generated clocks clk40p0, clk40p1, clk40p2 clk40p3 and clk40p4. To synchronized all bits together 40 flops are used which will sample the data by the clk_delay.

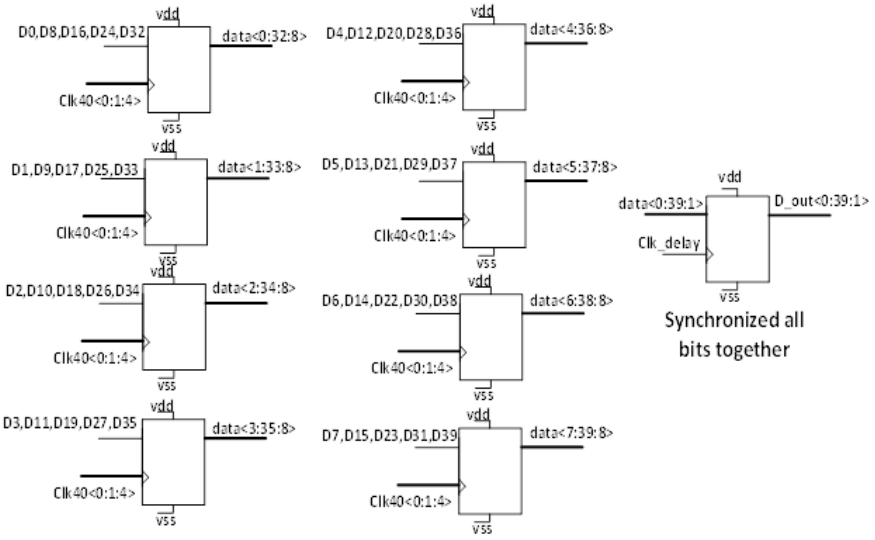

FIGURE 5.13: Block Diagram of SIPO 8→40

5.3 Summary

In this chapter we have discussed about proposed design of Serializer and Deserializer. Further detail description of each and every block has been given. For proposed Serializer single phase and multi-phase both topologies have been used. In Deserializer binary topology has been used, in final stage delay clock has been used for synchronize all parallel bits together.

Chapter 6

Result and Discussion

This chapter presents simulation results of the proposed PISO (Serializer) and SIPO (Deserializer). The schematics are designed in Cadence Virtuoso and TSMC 16nm technology model libraries are used. To verify that the designs are robust to a certain extent of PVT variations, the simulations are performed with supply variations (0.72-0.88V), three process variations (SS, FF, TT) and temperature variation of -40C to 125C. Setup and Hold calculations for all the flops and latches are considered with 10% variations of clock to Q delay.

6.1 Flip-Flops and Latches Timing Results

This section contains the simulation results of high speed flip-flops and latches as discussed in section 5.1. Transient simulations have been performed for the testing of setup and hold timings. Verilog A code has been added to measure the setup and hold violation times in simulations. Figure 6.1 shows the setup measurement and figure 6.2 shows hold measurement wave forms.

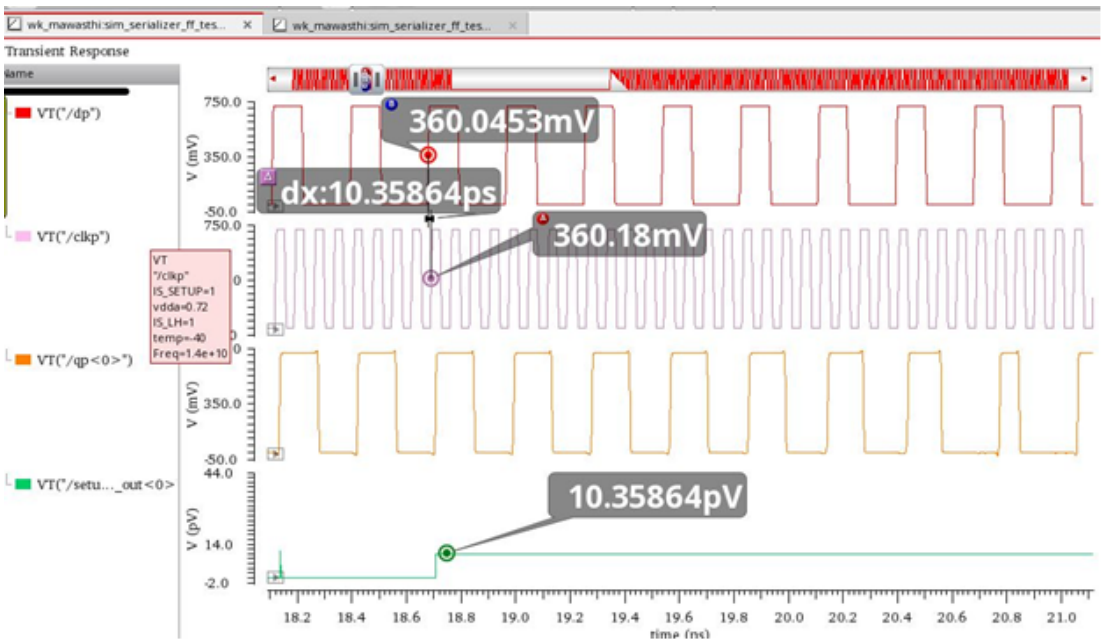

FIGURE 6.1: Setup Time Result Waveform

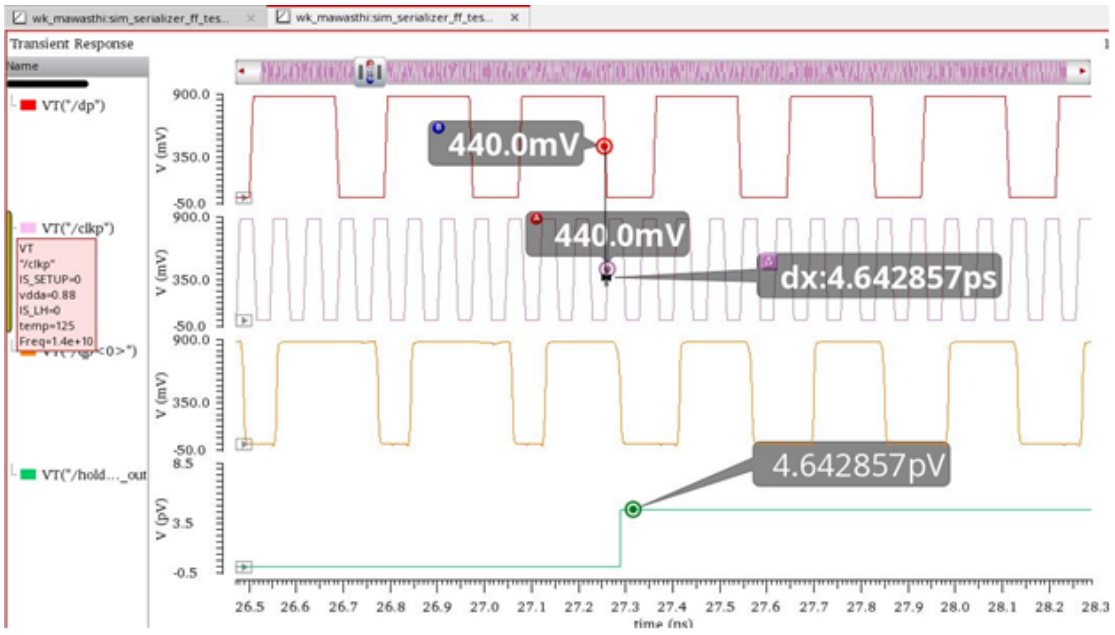

FIGURE 6.2: Hold Time Result Waveform

Tables 6.1 and 6.2 presents the results of High speed flip-flop and latch shown in section 5.1, Rise and fall time for input data and clock is 8ps.

Table 6.1: Setup and Hold Time of Flip-flop across corners

28G	Tcq(ps)	Setup(ps)	Hold(ps)
LV_LT_SS	15.19	15.19	-8.143
NV_NT_TT	12.89	8.36	-6.143
HV_HT_FF	11.47	7.36	-4.64
24G			
LV_LT_SS	15.25	10.5	-8
NV_NT_TT	12.92	7.5	-6
HV_HT_FF	11.54	8.5	-4.5

Table 6.2: Setup and Hold Time of Latch across corners

28G	Tcq(ps)	Setup(ps)	Hold(ps)
LV_LT_SS	15.5	11.25	-8.14
NV_NT_TT	13	9.35	-5.64
HV_HT_FF	11.6	8.85	-4.64
24G			
LV_LT_SS	15.35	11.5	-8
NV_NT_TT	12	9.5	-6
HV_HT_FF	11.6	9	-4.5

6.2 PISO Timing and Results

This section contains the timing wave forms and results of PISO. To simulate the PISO, Verilog A is used to generate 40 bit random data for every tx_par_clk_sig. Detail results of all the stages will be shown in following sections.

6.2.1 Reset Synchronizer and Load Signal Generator

High Speed input clock (clkp) of PISO is 14Ghz,once Synchronizer generate sync2p signal all the divider will start.First quadrature phase divide by 2 divider generates 7 Ghz clockk with the four different phase 0°,90°, 180° and 270°. After that 0° phase clock will generate divide by 5 clock 1.4 Ghz, and load signals. Further divide by 1.4 Ghz clock used to generate divide by 2 clock which is 0.7 Ghz, This clock is the input of first stage of PISO. Figure 6.3 shows the waveform of different internally generated clocks and figure 6.4 shows load signal waveforms.

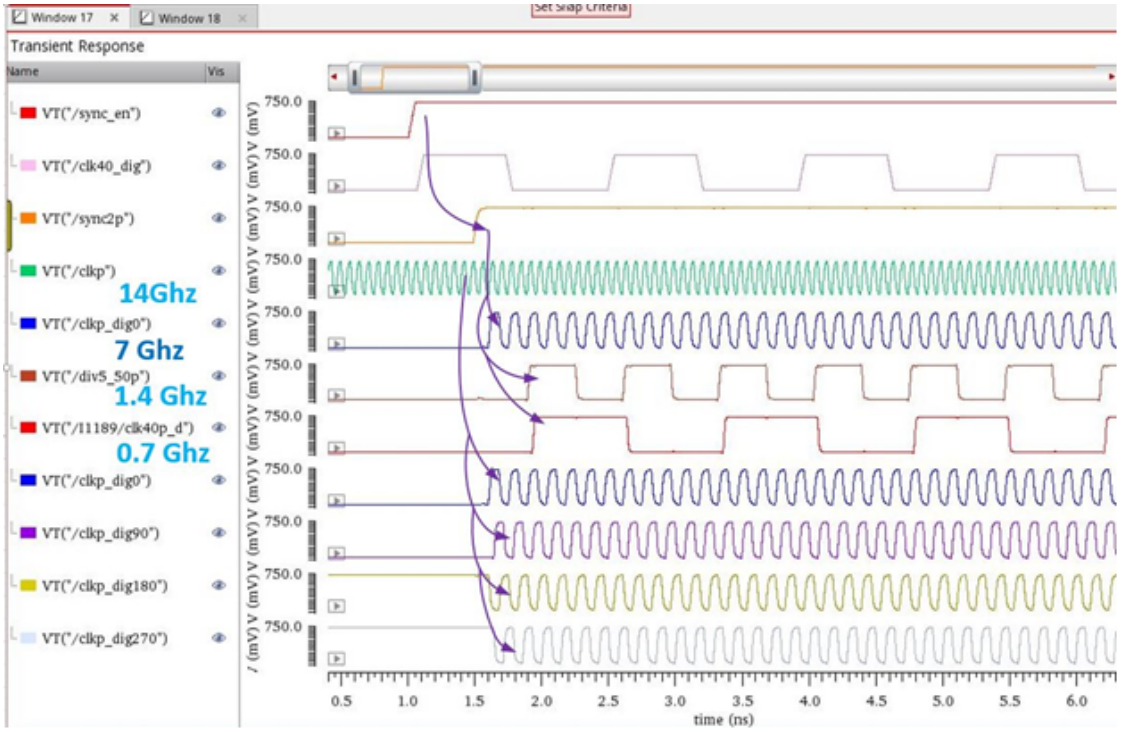

FIGURE 6.3: Wave forms of Different Clocks

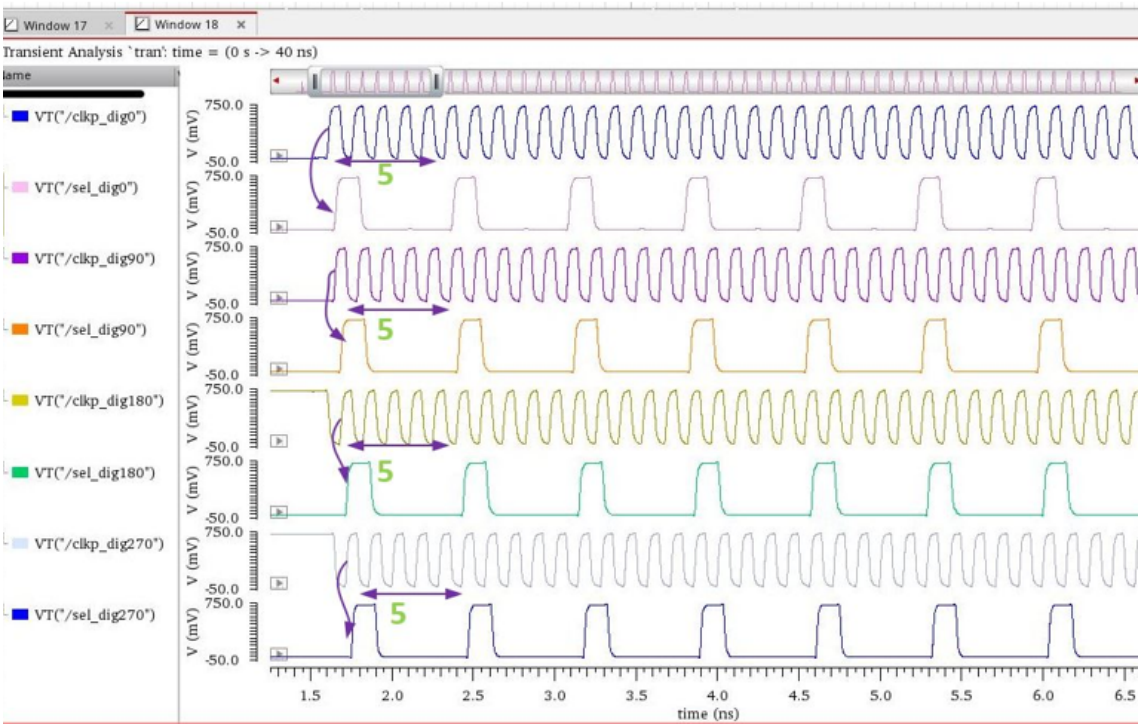

FIGURE 6.4: Wave forms of Load Signals

6.2.2 PISO 40→20

Figure 6.5 shows the waveform of first stage of PISO, here 40 bit parallel data is converted into 20 bit serial data using the clocks Clk40p and Clk40n as discussed in previous section.

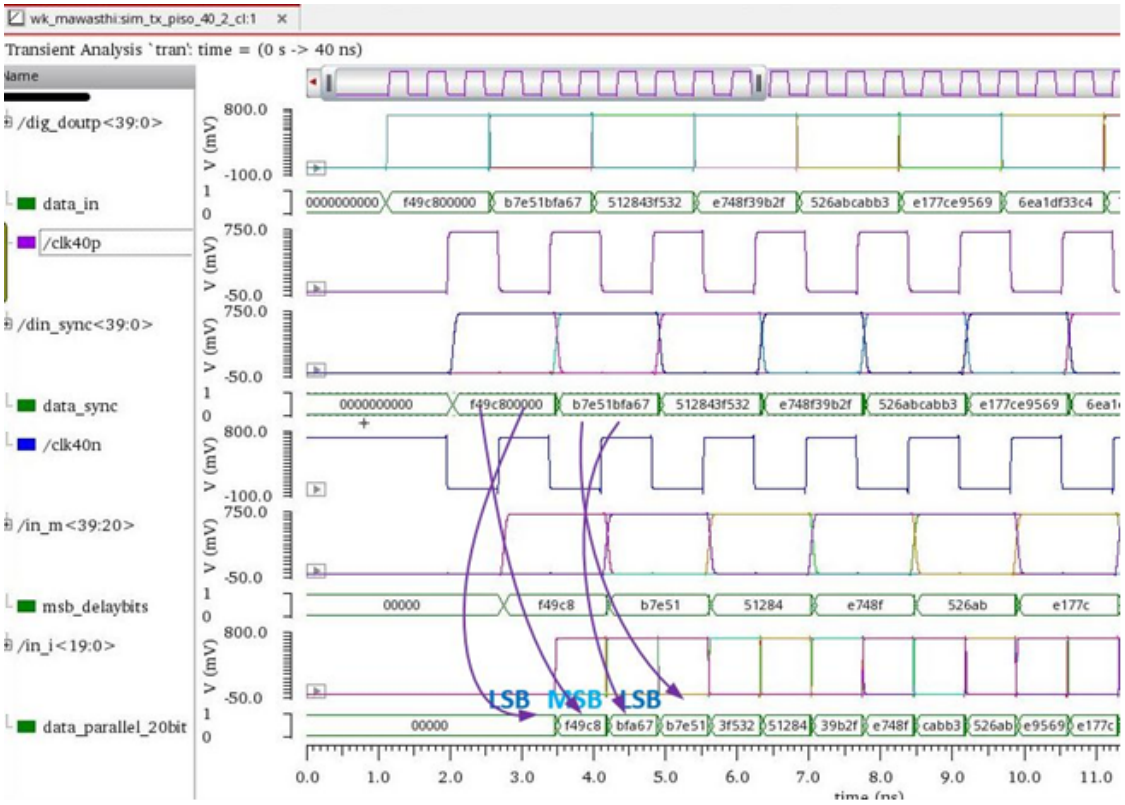

FIGURE 6.5: Wave form of PISO 40→20

6.2.3 PISO 20→4

PISO first stage converts 40 bit parallel data into 20 bits, in second stage 20 bit parallel data converted into 4 bit parallel data. Figure 6.6 shows the waveform of second stage to demonstrate the operation of shift registers.

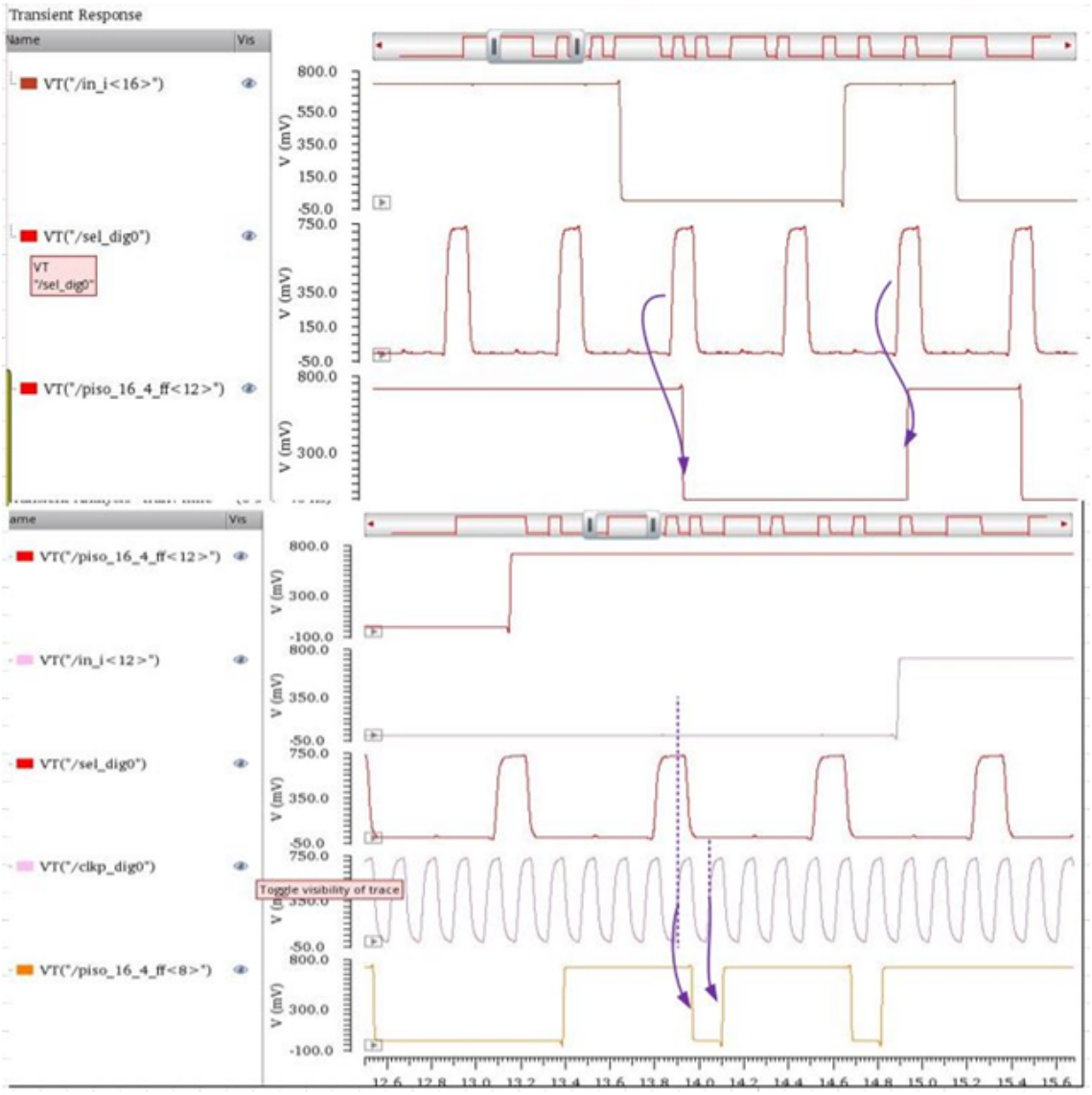

FIGURE 6.6: Wave form PISO 20→4

6.2.4 PISO 4→2→1

Figure 6.7 shows the waveform of 4→2 stage of PISO and figure 6.8 shows the 2→1 stage of PISO. Final serialized data is compared by Verilog A code PISO, for 28 Gbps data rate every bit should have 35.71 ps width.

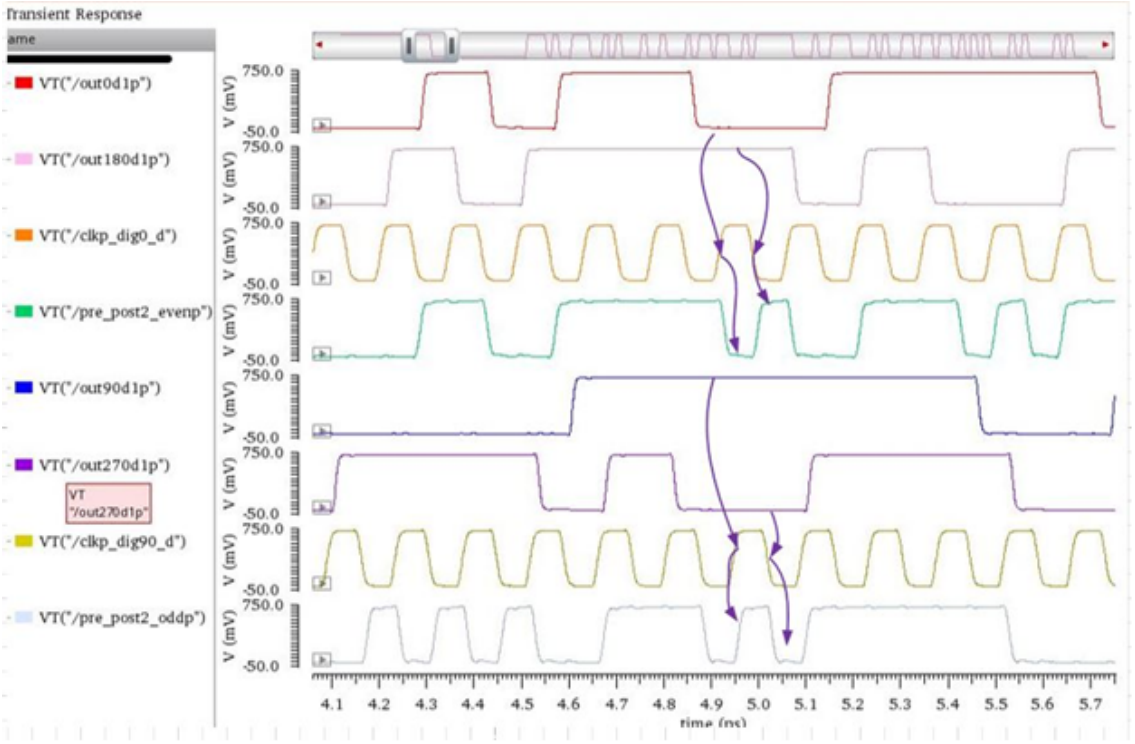

FIGURE 6.7: Wave form PISO 4→2

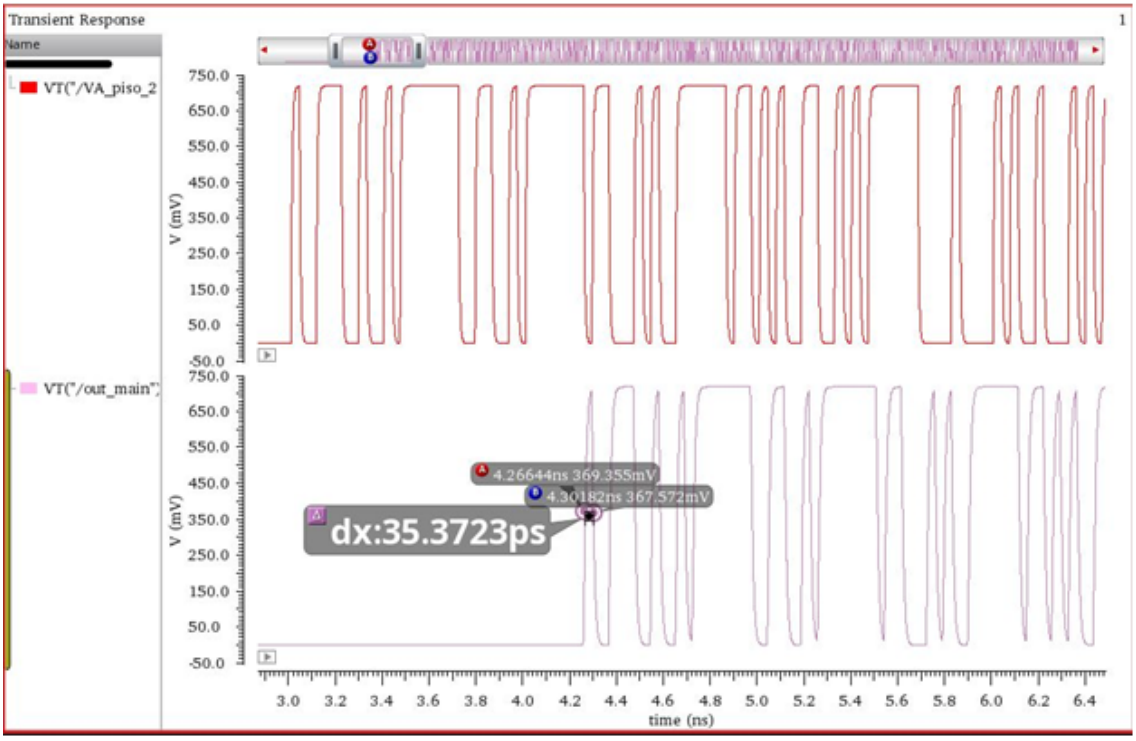

FIGURE 6.8: Wave form PISO 2→1

6.2.5 PISO worst path Timing Margins and Power

Figure 6.9 and 6.10 show the worst path of PISO. Here setup and hold margin should be enough so that after layout data will be able to pass. For calculation of timing margins setup and hold time of flop and latch is subtracted from time delay. For latch, margin should be greater than zero.

Tables 6.3 presents the timing margins results for second last stage and 6.4 presents the final stage timing margins results. Table 6.5 and 6.6 represents power consumption of whole PISO.

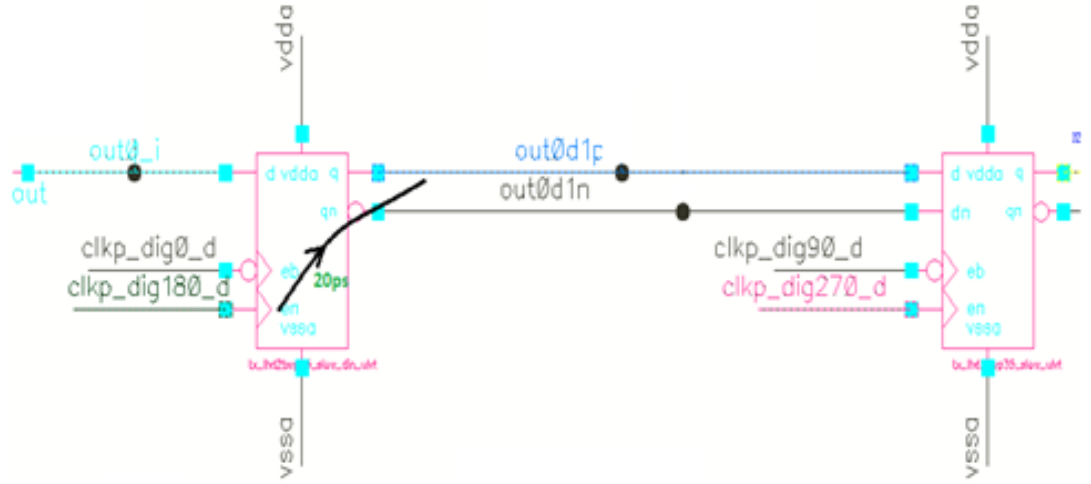

FIGURE 6.9: Second last Worst Path

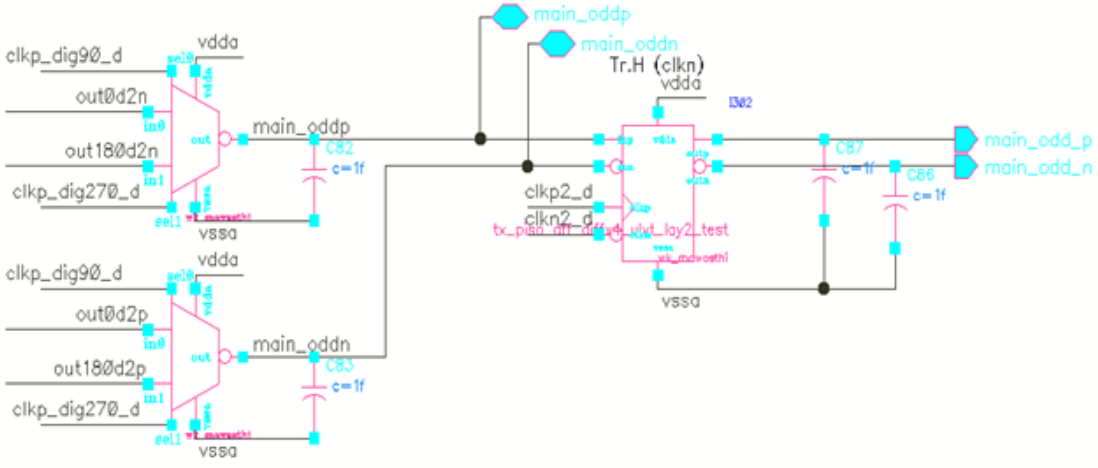

FIGURE 6.10: Final Stage Worst Path

Table 6.3: Timing Margins for Second Last Stage

28G	Setup(ps)	Setup(ps)	Setup(ps)	Hold(ps)	Hold(ps)	Hold(ps)
	Time Delay	Setup(L)	Setup Margin	Hold	Hold(L)	Hold Margin
LV_LT_SS	13.3	11.35	1.98	57.9	-8.14	66.04
NV_NT_TT	18.35	9.35	9	53	-5.64	58.64
HV_HT_FF	21.4	8.85	12.55	50.85	-4.64	55.49
24G						
LV_LT_SS	19.68	11.5	8.3	63	-8	71
NV_NT_TT	24.412	9.5	14.91	59.48	-6	65.48
HV_HT_FF	27.5	9	18.5	56.657	-4.5	61.1

Table 6.4: Timing Margins for Last Stage

28G	Setup(ps)	Setup(ps)	Setup(ps)	Hold(ps)	Hold(ps)	Hold(ps)
	Time Delay	Setup(DFF)	Setup Margin	Hold	Hold(DFF)	Hold Margin
LV_LT_SS	28.15	10.35	17.8	37.08	-8.143	45.22
NV_NT_TT	41.71	8.36	33.35	25	-6.143	31.14
HV_HT_FF	46.72	7.36	39.36	20.97	-4.643	25.6
24G						
LV_LT_SS	42.75	10.5	32.25	32.12	-8	40.12
NV_NT_TT	53.76	7.5	46.26	25.169	-6	31.17
HV_HT_FF	58.51	8.5	50.01	21.18	-4.5	25.68

Table 6.5: Average Power of PISO

	Iavg(mA)	Iavg(mA)	Ipeak(mA)	Ipeak(mA)	Pavg(mW)	Pavg(mW)
	24G	28G	24G	28G	24G	28G
LV_LT_SS	4.17	4.97	11.592	13.326	3	3.57
NV_NT_TT	5.55	6.75	20.91	19.214	4.44	5.25
HV_HT_FF	7.56	9.35	31.761	29.673	6.65	8.22

Table 6.6: Leakage Power of PISO

	Leakage Current(uA)	Leakage Current(uA)	Leakage Power(uW)	Leakage Power(uW)
	24G	28G	24G	28G
LV_LT_SS	8.142	9.467	5.8	6.81
NV_NT_TT	33.62	35.03	26.89	28.02
HV_HT_FF	409.5	411.3	360.36	361.94

6.3 SIPO Timing and Results

This section contains the timing wave forms and results of SIPO. To simulate the SIPO, Verilog A is used to generate high speed random data for every rx_par_clk_sig. Detail results of all the stages will be shown in following sections.

6.3.1 Reset Synchronizer and Load Generator

Figure 6.11 shows the waveform of internally generated divider clocks and delay clocks for last stage of SIPO.

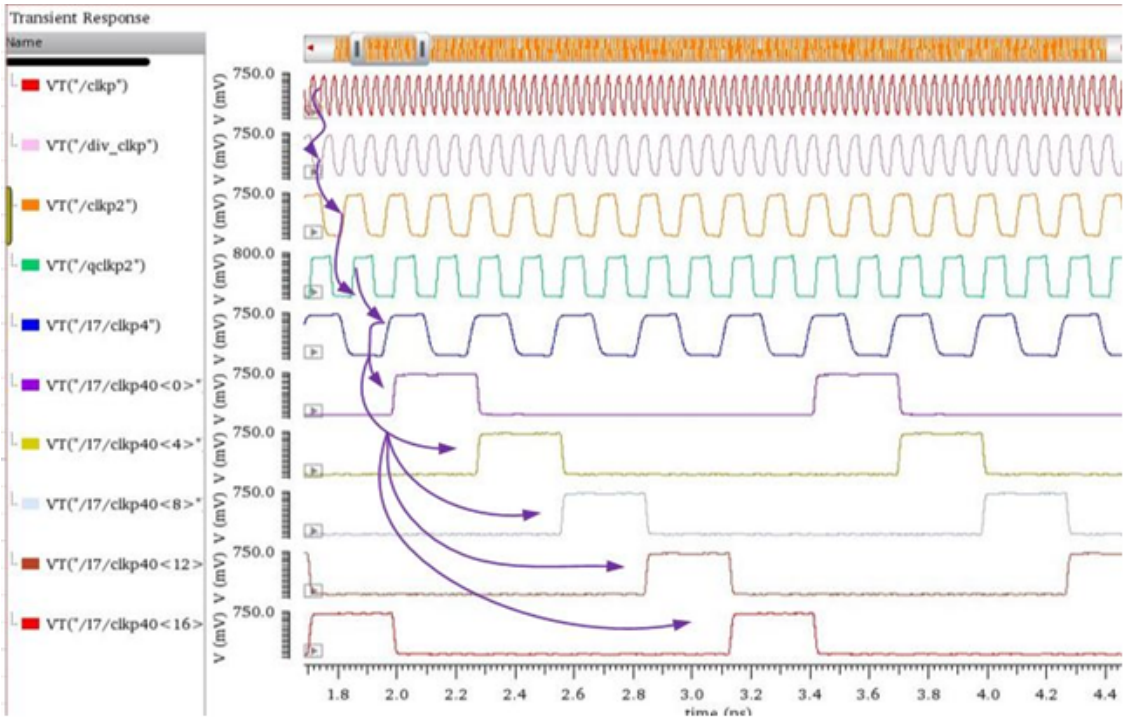

FIGURE 6.11: Wave forms of internally Generated Clocks

6.3.2 SIPO 1→2→4

Figure 6.12 shows the waveform of first two stages of SIPO, here serial data is converted into 2 bit parallel data and then 2 bit parallel data into 4 bit parallel data as discussed in previous section.

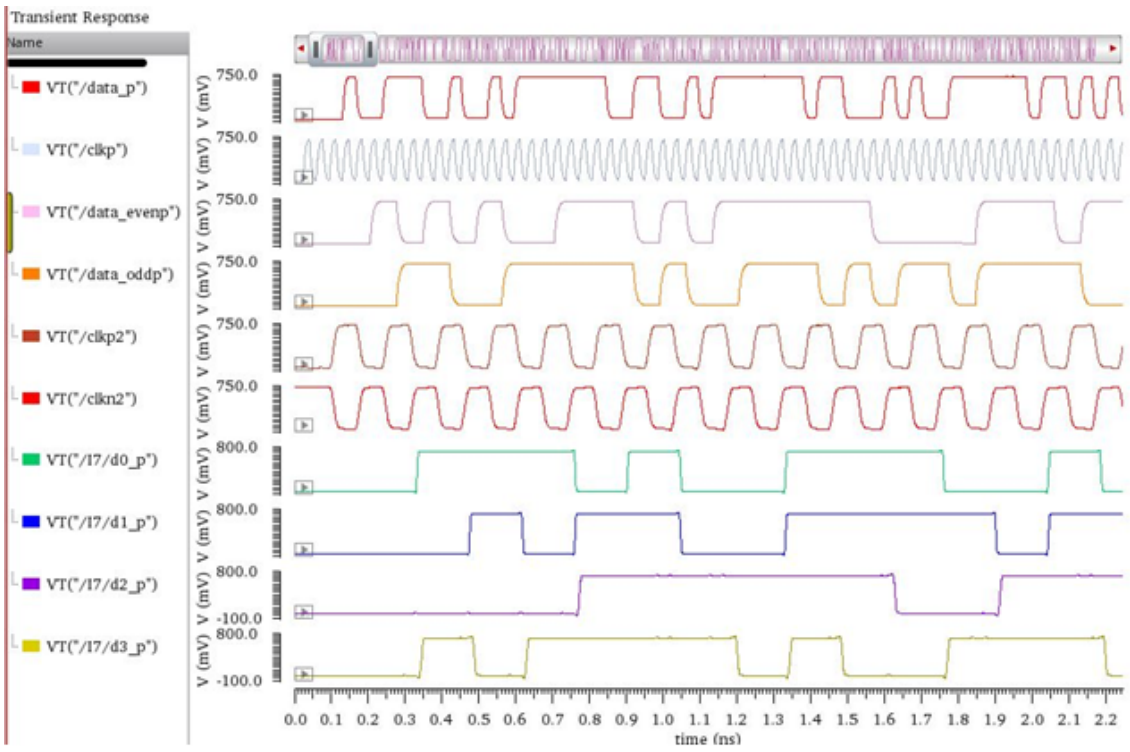

FIGURE 6.12: Wave form of SIPO 1→2→4

6.3.3 SIPO 4→8

Figure 6.13 shows the waveform of third stage of SIPO, here 4 bit parallel data is converted into 8 bit parallel data as discussed in previous section.

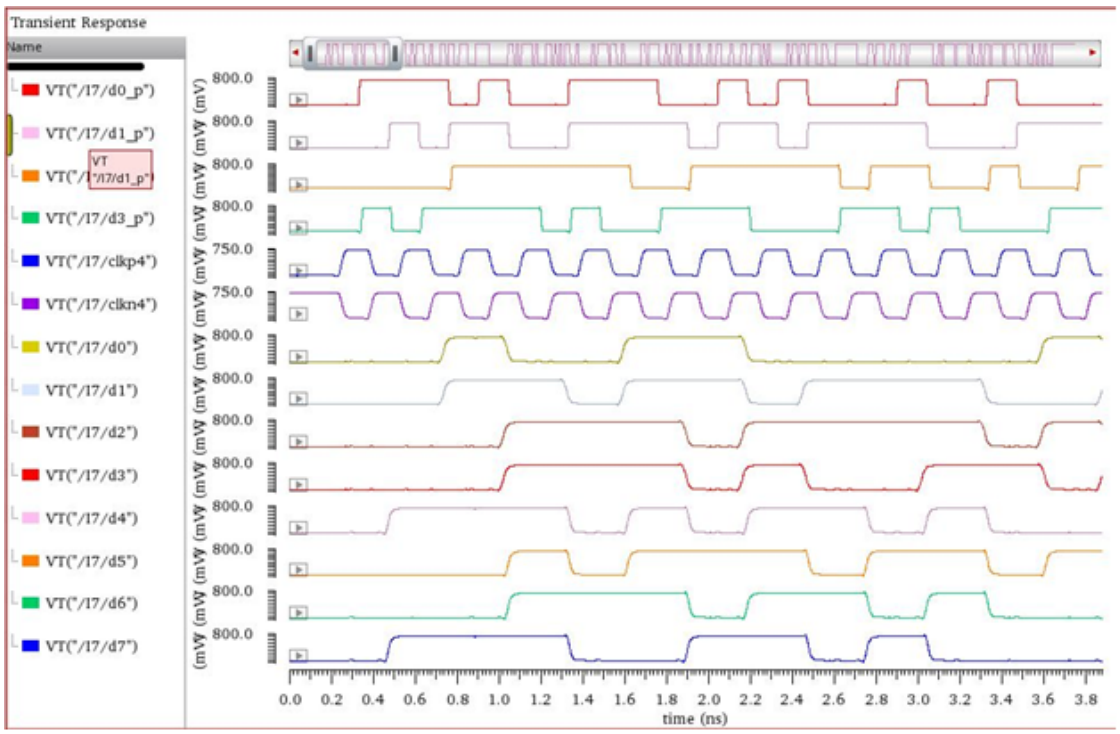

FIGURE 6.13: Wave form of SIPO 4→8

6.3.4 SIPO 8→40

Figure 6.14 and 6.15 shows the waveform of final stage of SIPO, Here 8 bit parallel data is converted into 40 bit parallel data. To synchronized 40 bits clk_delay is used. To check the results, parallel bits are again converted into serial data using Verilog A code block and compared with original serial data.

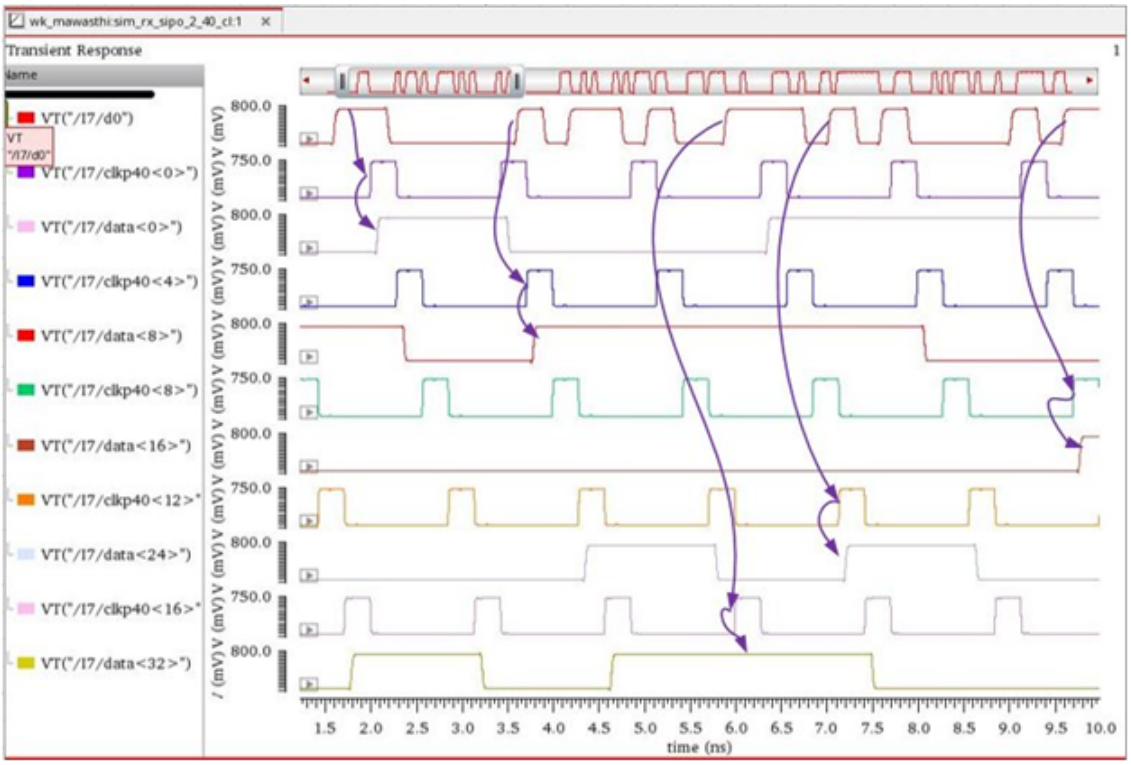

FIGURE 6.14: Wave form of SIPO 8→40

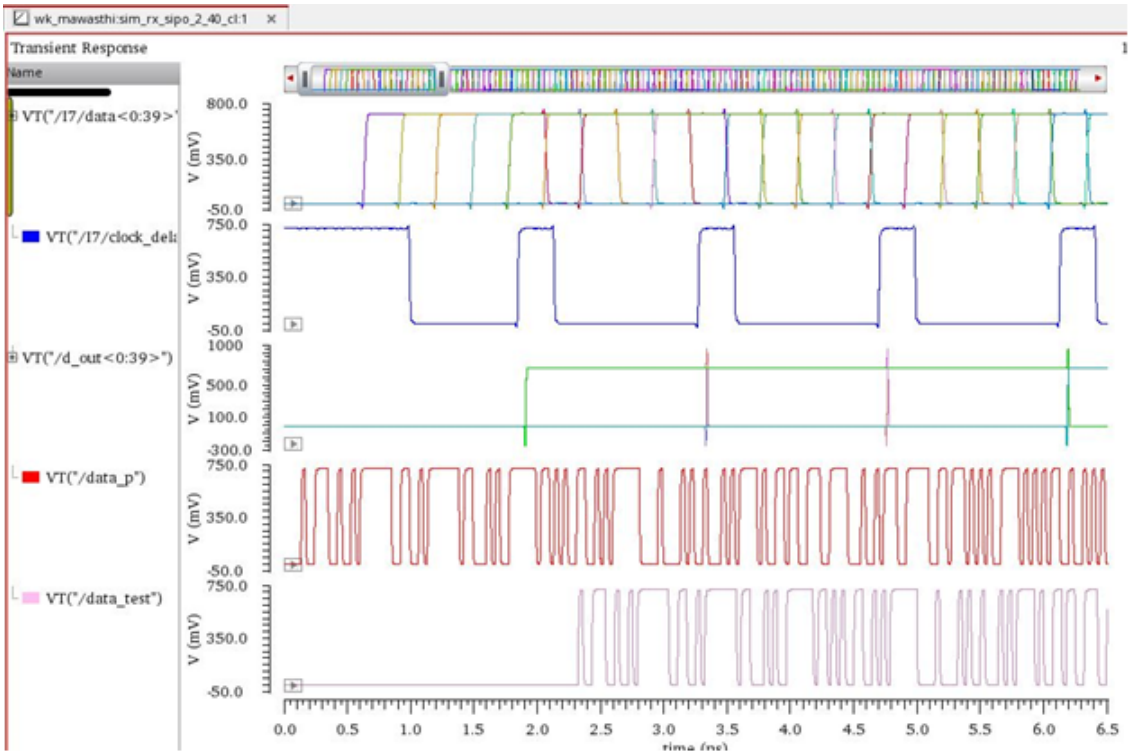

FIGURE 6.15: Final Deserialized Data

6.3.5 SIPO Power

Table 6.7 and 6.8 represents power consumption of whole SIPO

Table 6.7: Average Power of SIPO

	Iavg(mA)	Iavg(mA)	Ipeak(mA)	Ipeak(mA)	Pavg(mW)	Pavg(mW)
	24G	28G	24G	28G	24G	28G
LV_LT_SS	1.85	2.1	4.13	6.3	1.33	1.57
NV_NT_TT	2.43	2.81	6.44	6.59	1.94	2.24
HV_HT_FF	3.24	3.64	9.54	10.24	2.85	3.20

Table 6.8: Leakage Power of SIPO

	Leakage Current(uA)	Leakage Current(uA)	Leakage Power(uW)	Leakage Power(uW)
	24G	28G	24G	28G
LV_LT_SS	80.1	99.56	57.67	71.68
NV_NT_TT	134.6	154.8	107.68	123.8
HV_HT_FF	272.3	299.5	239.64	263.56

6.4 Comparison with State-of-the-Art work

The tables below show the comparison of the performance metrics of the proposed Serializer with the other State-of-the-Art Serializer available. The present work has been power efficient compared to the existing work. Usually for such a high frequency power efficiency is greater than 0.5pJ/bit. It has 0.29pJ/bit of efficiency. The performance has been compared with [11], [18] and [19]. The architecture is used to support for high speed data link i/o ports for various protocol.

Table 6.9: Comparison with previously reported work

	[11]	[18]	[19]	This Work PISO	This Work SIPO
Technology	65nm CMOS	65nm CMOS	65nm CMOS	TSMC 16nm	TSMC 16nm
Supply (v)	1.2	1.2	NA	0.8	0.8
Data Rate (Gbps)	32-48	50-64	16-40	28	28
Bit Conversion	64:1	64:1	4:1	40:1	1:40
Power (mW)	15	34	68.25	8.22	3.20
Power Efficiency (pJ/bit)	0.3125	0.53	1.70	0.29	0.11

6.5 Summary

The simulations result in the chapter are provided to show the proper functioning of a high-speed low power Serializer and Deserializer. A summary of setup and hold time of high-speed flip-flop and latch has been discussed in the table. A further summary of average current, leakage current, peak current, average power, and average leakage power has been given in the table. The proposed architecture has less area, high speed, and low power efficiency. All the results summary has been shown in the table.

Chapter 7

Conclusion

High Speed 28Gbps Serializer and Deserializer are implemented. The thesis addresses design considerations, circuit implementation and performance analysis of Serializer and Deserializer. Firstly, the basic architecture of a high-speed IO link system (SerDes) was discussed. A thorough study of different architecture of PISO (Serializer) and SIPO (Deserializer) was presented. Different logics to design high speed and low power digital circuits were discussed, also timing definition and different topologies of flip-flops were discussed. Secondly, the proposed architecture of the Serializer and Deserializer was discussed. Each block of the Serializer and Deserializer used in this thesis work was discussed in depth. The proposed architecture of Serializer to achieve 28 Gbps speed with low power consumption is mixed with both single-phase and multiphase topology. Here the first stage is a direct implementation of single-phase clock topology, in second stage multiphase based shift register topology is used. The advantage of using this mixed topology has been discussed. To increase the time margin for the final stage of Serializer high-speed latches are used in the second stage. In the final stage to achieve 28 Gbps data rate high-speed flops are used, those flops have very less setup and hold time requirement. The proposed architecture of Deserializer will be able to convert high-speed 28Gbps data into parallel data. Here the first 1248 conversion is in binary topology, in the final 840 conversions, five shift register-based delay clocks are used. Final 40 bit parallel data is synchronized using one delay clock. This Serializer and Deserializer work on TSMC 16nm on a voltage supply of 0.8V supporting process and temperature variations. Serializer is designed to convert 40 bit parallel data into serial data and Deserializer is designed to convert serial data into 40 bit parallel data. In this work Serializer is consuming power less than (< 10mw) and Deserializer is consuming power less than (¡ 4mw). Both PISO and SIPO can support speed up to 28Gbps, PISO can drive the load less than (< 50fF).

7.0.1 Future Work

In the current work, Serializer and Deserializer are supporting data rate up to 28Gbps with power consumption less than ($<$ 10mw) for PISO and less than ($<$ 4mw) for SIPO. This work can further be improved for higher data rate and reduce power consumption. In PISO this work can extended by tapping pre, main and post data. In SIPO, less number of flip-flops can be used so that it will reduce area for given data rate and power.

Bibliography

[1] http://www.ece.tamu.edu/ spalermo/ecen620.html.

[2] http://www.ece.tamu.edu/ spalermo/ecen689/lecture1_ee720_intro.pdf.

[3] Zexian Li. Design of serializer and deserializer operating in 65 nm cmos technology for high-speed serial link (hssl) applications. *Senior Thesis in Electrical Engineering University of Illinois at Urbana-Champaign.*

[4] Rishi Ratan. Design of a phase locked loop based clocking circuit for high speed serial link applications. *University of Illinois at Urbana-Champaign, 2014.*

[5] F. M. Gardner. Charge-pump phase-lock loops. *IEEE Trans.Commun*, COM-28 (11):18491858, 1980.

[6] http://www.ee.iitm.ac.in/videolectures/doku.php?id=ee6322$_2$018 : *start*.

[7] Hanqiao Zhang Steven Krooswyk Jeff Ou. High speed digital design, design of high speed interconnects and signaling. *Elsevier Inc.*, 2015.

[8] http://www.ece.tamu.edu/ spalermo/ecen689/lecture5_ee720$_t$*ermination_txdriver.pdf*.

[9] Razavi. Design of integrated circuits for optical communication. 2003.

[10] H. Tao. 40-43-gb/s) oc-768 16:1 mux/cmu chipset with sfi-5 compliance. *IEEE J. Solid-State Devices*, 38(12):2169–2180, 2003.

[11] M. Chen A. A. Hafez and C. K. Yang. A 32-48 gb/s serializing transmitter using multiphase serialization in 65 nm cmos technology. *IEEE J. Solid-State Circuits*, 50(3): 763–775, 2015.

[12] S. Sidiropulos Horowitz M., Chih-kong Yen Yang. High-speed electrical signaling: Overview and limitations. *IEEE Micro*, 18(1):12–24, 1998.

[13] Eshraghian K. Weste N. H. E. Principles of cmos vlsi design, a systems perspective. second edition, 1994.

[14] Stojanovic V. Oklobdzija V.G. Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems. *IEEE*, 34:536–548, 1999.

[15] Nikolic B. Rabaey J. M., Chandrakasan A. Digital integrated circuits, a design perspective. second edition, 2003.

[16] Brodersen R.W. Markovic D. Nikolic B. Analysis and design of low-energy flip-flops. *International Symposium on Low Power Electronics and Design*, pages 52–55, 2001.

[17] Yuan J. Sevensson C. New single-clock cmos latches and flip-flops with improved speed and power saving. *IEEE Journal of Solid-State Circuits*, 32:62–69, 1997.

[18] M. Chen and C. K. Yang. A 5064 gb/s serializing transmitter with a 4-tap, lc-ladder-filter-based ffe in 65 nm cmos technology. *IEEE J. Solid-State Circuits*, 50(8):1903–19016, 2015.

[19] Amr Elshazly Yan-Yu Huang Hang Song Kai Yu Frank OMahony Jihwan Kim, Ajay Balankutty. A 16-to-40gb/s quarter-rate nrz/pam4 dual-mode transmitter in 14nm cmos. *IEEE International Solid-State Circuits Conference*, pages 60–63, 2015.