Mixed Signal Serializer/Deserializer (SerDes) Design using Sky130 and eSim

Ayush Gupta
SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India

Abstract— **This work is aimed at presenting the design of a Serializer/deserializer mixed signal circuit for SerDes Transceiver based chip applications. It features the conversion 8-bit parallel data to 10-bit serial form via an 8B10B encoding scheme. This 10-bit data is then transmitted to a deserializer to get it back in the parallel form. The implementation focuses on using the Google SkyWater 130nm PDK to design and implement the deserializer circuit. On the other hand, serialization part is implemented in digital design using Verilog HDL. This mixed signal interface with digital logic helps in establishing a serialization-deserialization circuit in both domains for handling the continuous signals similar to their digital counterparts.**

Keywords— Serializer, Deserializer, Sky130, 8B10B Encoder, SerDes

Theory

Modern day computing has seen that utilization of multi-channel circuits is increasing which are being used in parallel to transfer more data at the same time because it is being witnessed that more information is required with less data transmission time. An approach to address these issues at high data speed has been discussed and designed ahead focusing on the Serializer/Deserializer (SerDes) blocks. The SerDes device allows the transmission of parallel data with certain length of bits in a bus with fewer bits often single bit (serializer), sending through a channel to a receiver that performs the inverse operation converting the incoming stream from serial to parallel format (deserializer).

Deserializer

The deserializer block is designed via the transistor level design method. It utilizes different approach to conventional deserializer wherein two types of flip flops (FF) are used: a positive edge triggered flip flop and a negative edge triggered flip flop. These flip flops are implemented using clock overlap insensitive clocked CMOS (C2MOS) registers. This deserializer block is used here to convert the incoming serial bit stream into an encoded parallel data form.

The proposed Deserializer is designed such that it samples input on both edges of clock i.e. positive as well as negative edge. The advantage of this technique of input sampling is that a lower clock—half the original rate—is distributed for the same functional throughput. The deserialization part is done using the standard cell library of Sky130 PDK and KiCAD-NgSpice. The figure below shows the overall block diagram of the Serializer-Deserializer interface.

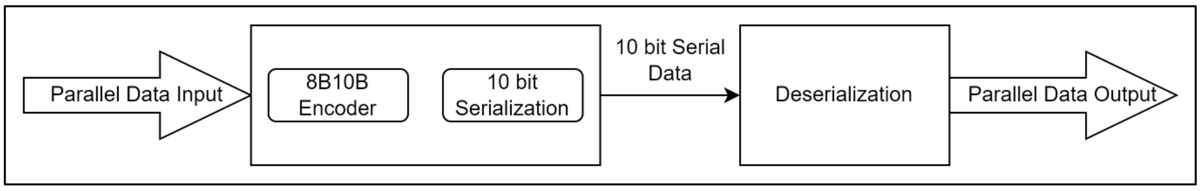


Fig. 1. Block Diagram of SerDes-Interface

Edge Triggered D Flip Flop

Shown ahead are the C^2MOS (Clocked CMOS) logic D flip-flops mentioned previously considered for the design of deserializer.

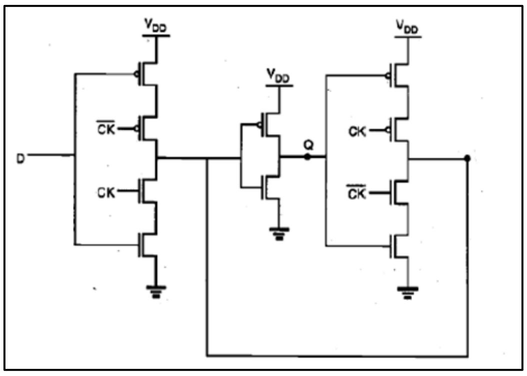


Fig. 2. Positive Edge Triggered DFF

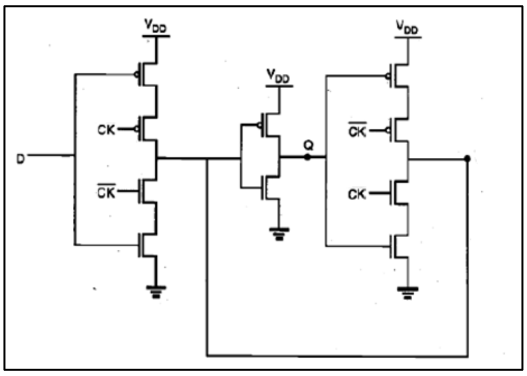


Fig. 3. Negative Edge Triggered DFF

8B10B Encoder

The Encoder is a combinational module used to generate an encoded 10-bit output. It generates the same bit amount of 1's and 0's in order to get a balanced serial data stream.

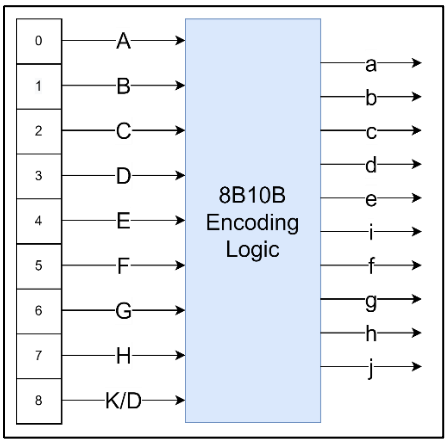


Fig. 4. 8B10B Encoder Block Diagram

Fig. 2 above shows the encoder architecture. The 8B10B logic is divided into two main sub-logic blocks: 5B/6B and 3B/4B. The first one has as an input the less significant bits of the parallel data. Those bits are converted into six bits balanced according to the disparity. The decoder bits are ordered and labeled in the following order: "abcdei". The 3B/4B Encoder has as an input the three most significant bits of the parallel data. Those bits are converted into four bits and balanced according to the disparity. The decoder bits are ordered and labeled in the following order: “fghj”. Encoder then merges all the encoded bits to create a stream in the order: abcdeifgj”. The D/K# disparity indicator bit is used to generate control symbol.

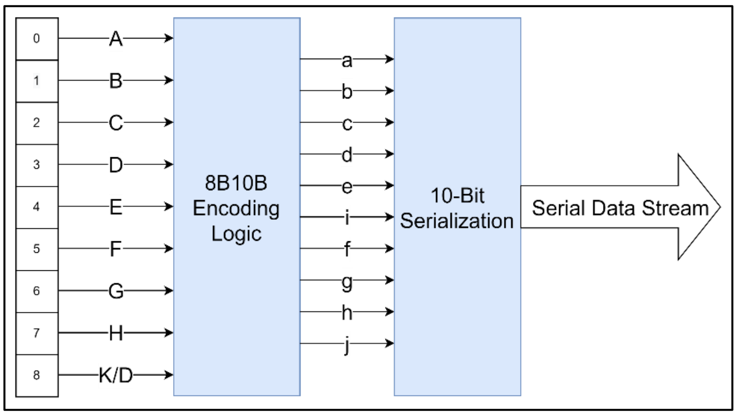
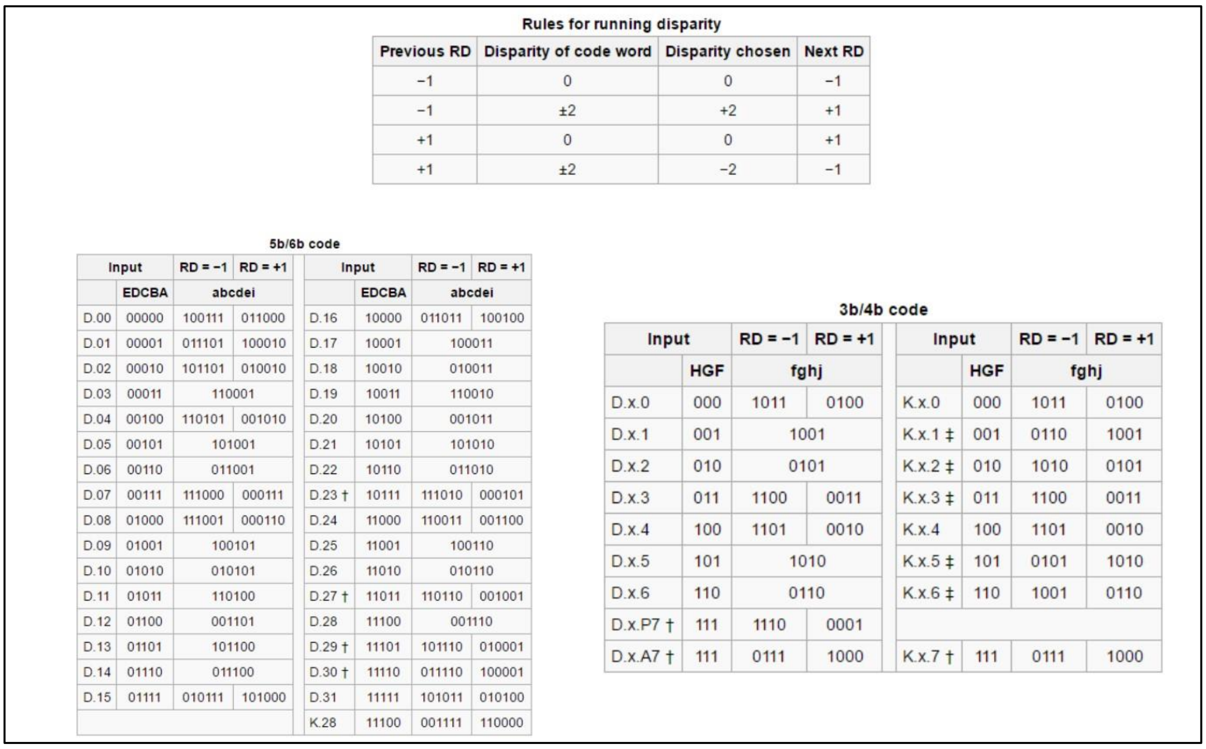


Fig. 5. Serializer Block Diagram

The Digital Serialization block succeeding the 8B10B encoder converts the 10-bit parallel data into a serial data stream.

Encoding Scheme Truth Table



Rules for running disparity

Previous RD	Disparity of code word	Disparity chosen	Next RD
−1	0	0	−1
−1	±2	+2	+1
+1	0	0	+1
+1	±2	−2	−1

5b/6b code

Input		RD = −1	RD = +1	Input		RD = −1	RD = +1
	EDCBA	abcdei			EDCBA	abcdei	
D.00	00000	100111	011000	D.16	10000	011011	100100
D.01	00001	011101	100010	D.17	10001	100011	
D.02	00010	101101	010010	D.18	10010	010011	
D.03	00011	110001		D.19	10011	110010	
D.04	00100	110101	001010	D.20	10100	001011	
D.05	00101	101001		D.21	10101	101010	
D.06	00110	011001		D.22	10110	011010	
D.07	00111	111000	000111	D.23 †	10111	111010	000101
D.08	01000	111001	000110	D.24	11000	110011	001100
D.09	01001	100101		D.25	11001	100110	
D.10	01010	010101		D.26	11010	010110	
D.11	01011	110100		D.27 †	11011	110110	001001
D.12	01100	001101		D.28	11100	001110	
D.13	01101	101100		D.29 †	11101	101110	010001
D.14	01110	011100		D.30 †	11110	011110	100001
D.15	01111	010111	101000	D.31	11111	101011	010100
				K.28	11100	001111	110000

3b/4b code

Input		RD = −1	RD = +1	Input		RD = −1	RD = +1
	HGF	fghj			HGF	fghj	
D.x.0	000	1011	0100	K.x.0	000	1011	0100
D.x.1	001	1001		K.x.1 ‡	001	0110	1001
D.x.2	010	0101		K.x.2 ‡	010	1010	0101
D.x.3	011	1100	0011	K.x.3 ‡	011	1100	0011
D.x.4	100	1101	0010	K.x.4	100	1101	0010
D.x.5	101	1010		K.x.5 ‡	101	0101	1010
D.x.6	110	0110		K.x.6 ‡	110	1001	0110
D.x.P7 †	111	1110	0001				
D.x.A7 †	111	0111	1000	K.x.7 †	111	0111	1000

Fig. 6. Truth Table for 3B4B and 5B6B Encoding Scheme

The table above is showing how the output stream bits will be translated if the Run Disparity is positive or negative. The table 3B/4B explains how the output stream bits will be translated in the case that the Run Disparity is positive or negative. Both tables ensure the uniqueness of the special bit sequence depending on the disparity bit and the K-bit [3]. According to the Figure 10. The rules for Running Disparity (RD) are: if the previous RD is a -1 and the disparity of stream bits is 0, this means that the RD bit does not change. If the disparity of stream bits is +2 and the previous RD is -1, the RD became in +1 value. If the previous RD is a +1 and the disparity of stream bits is +2 that means that the RD bit does not change, but if the disparity of stream bits is positive, the RD became in -1 value.

Proposed Design

1) 8B10B Encoder Digital Block

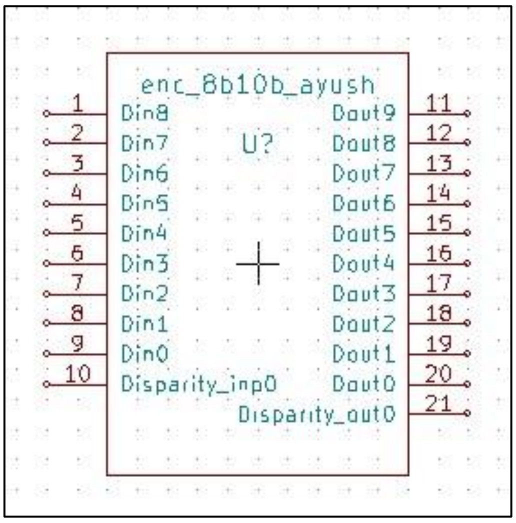


Fig. 7. 8B10B Encoder Symbol

2) 10-bit Serialization Digital Block

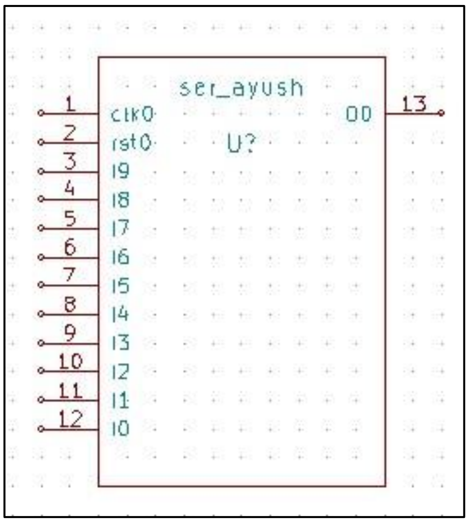


Fig. 8. 10-bit Serialization Block Symbol

3) 8B10B Encoder Testbench Setup

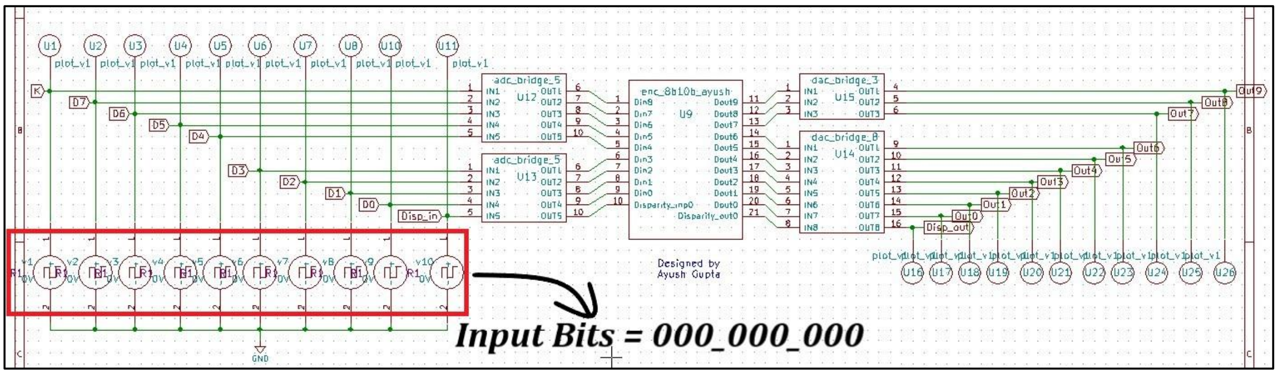


Fig. 9. 8B10B Encoder Testbench

4) Serializer Testbench Setup

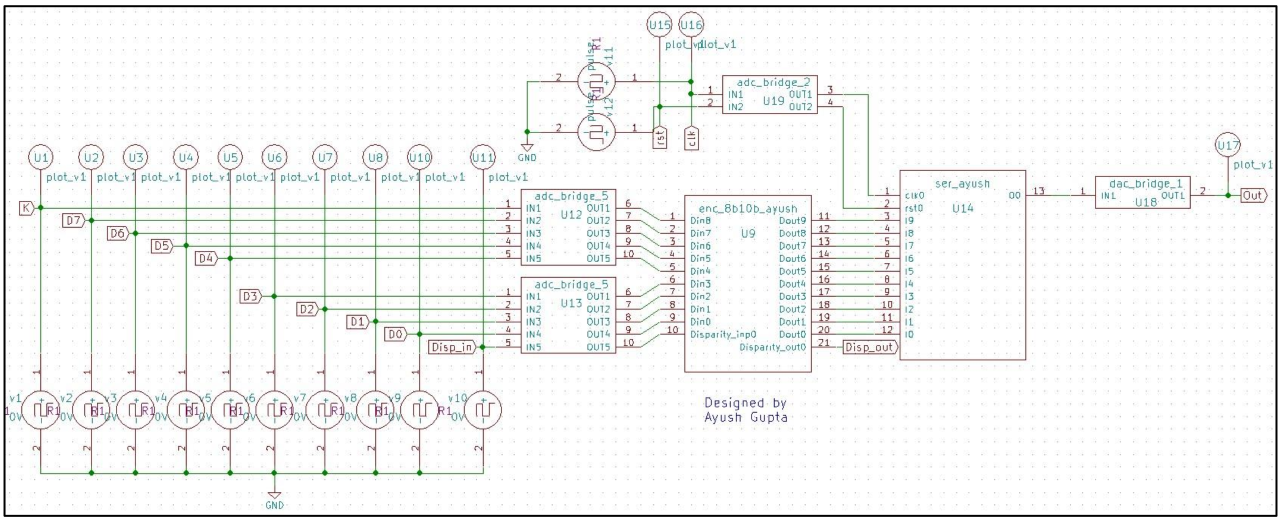


Fig. 9. Serializer Testbench

5) Positive Edge Triggered D Flip Flop

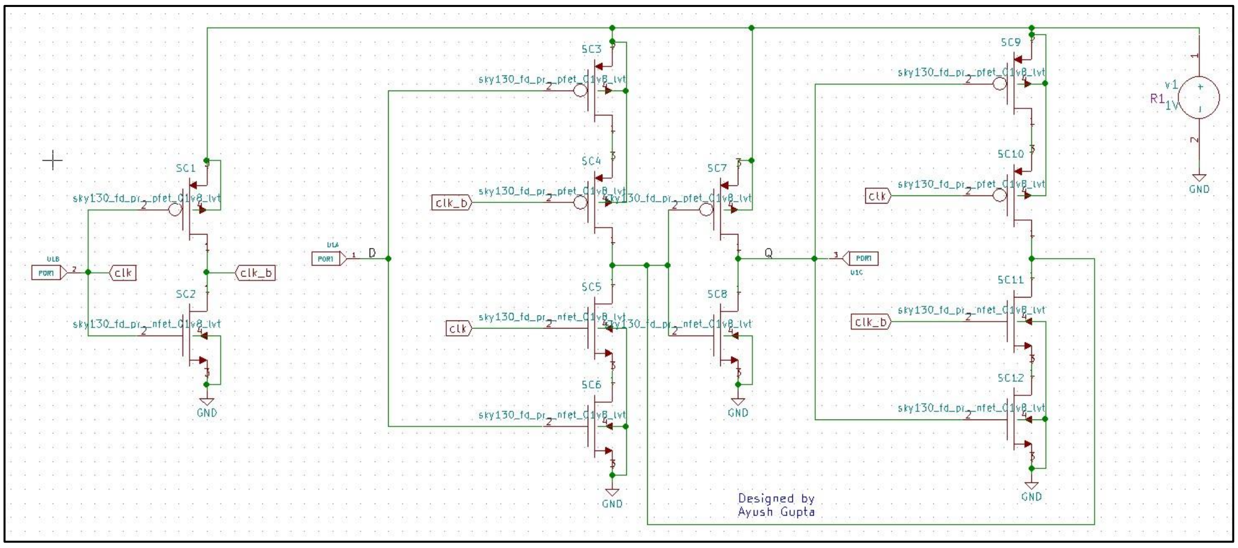


Fig. 10(a). Schematic

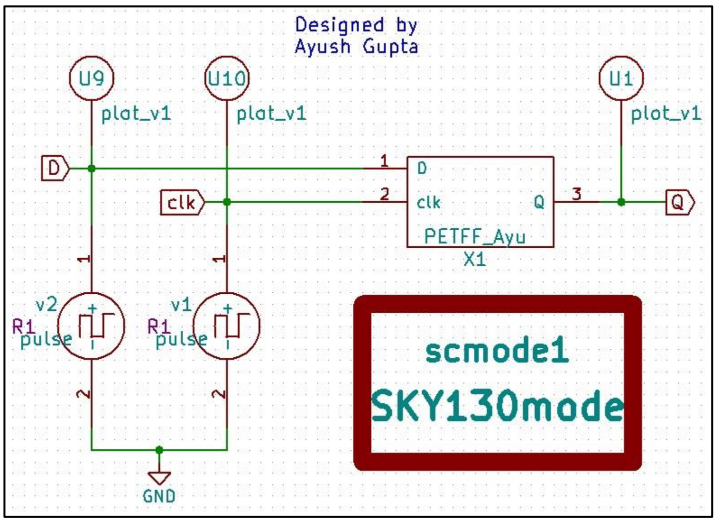


Fig. 10(b). Symbol and Testbench

6) Negative Edge Triggered D Flip Flop

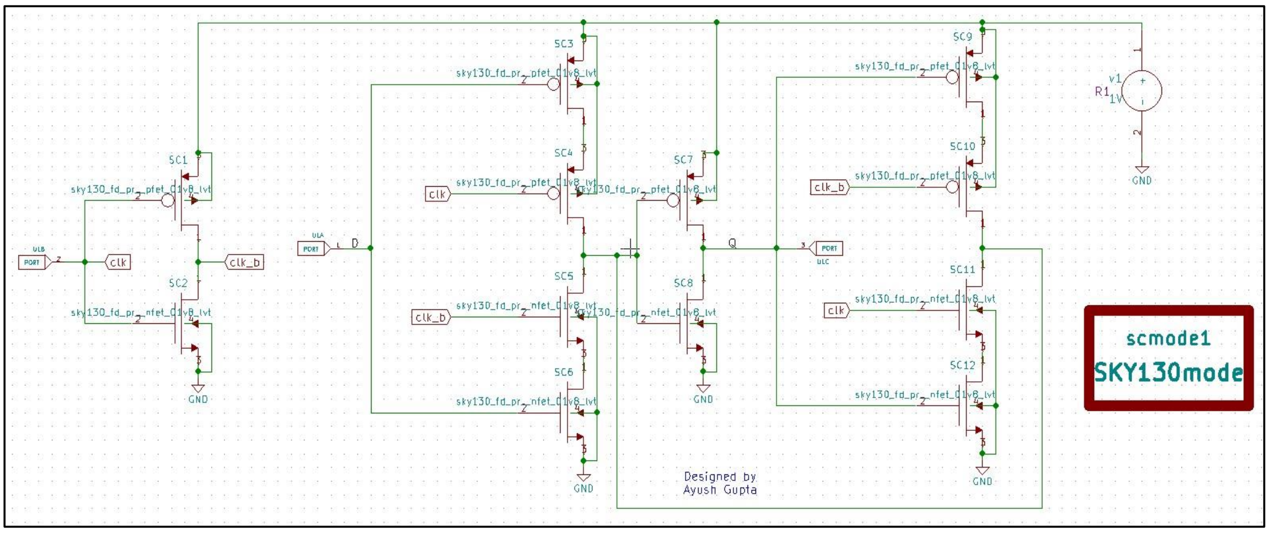


Fig. 11(a). Schematic

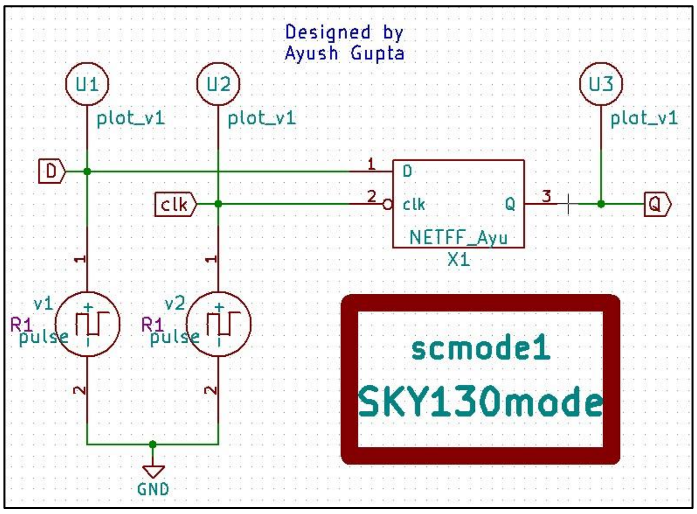


Fig. 11(b). Symbol and Testbench

7) Deserializer

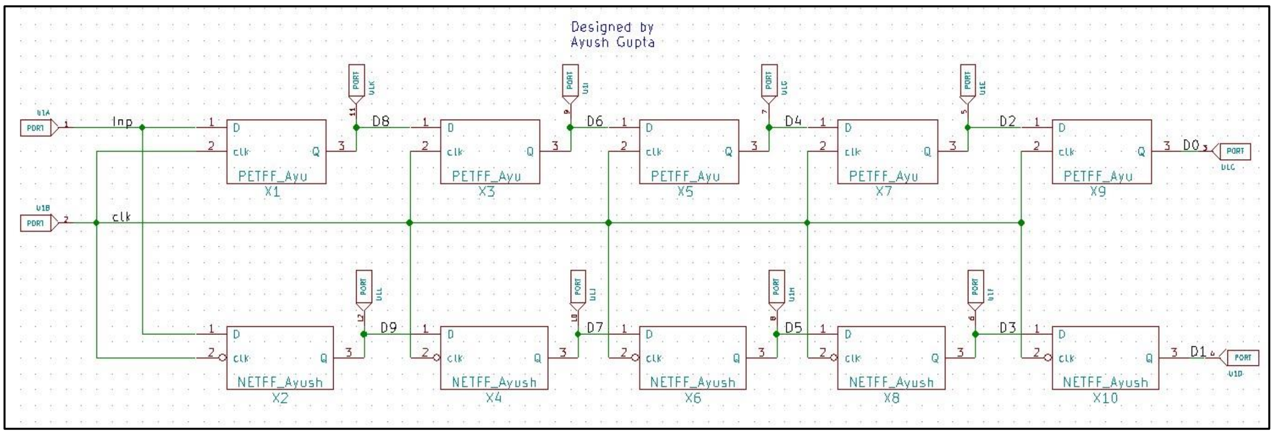


Fig. 13(a). Schematic

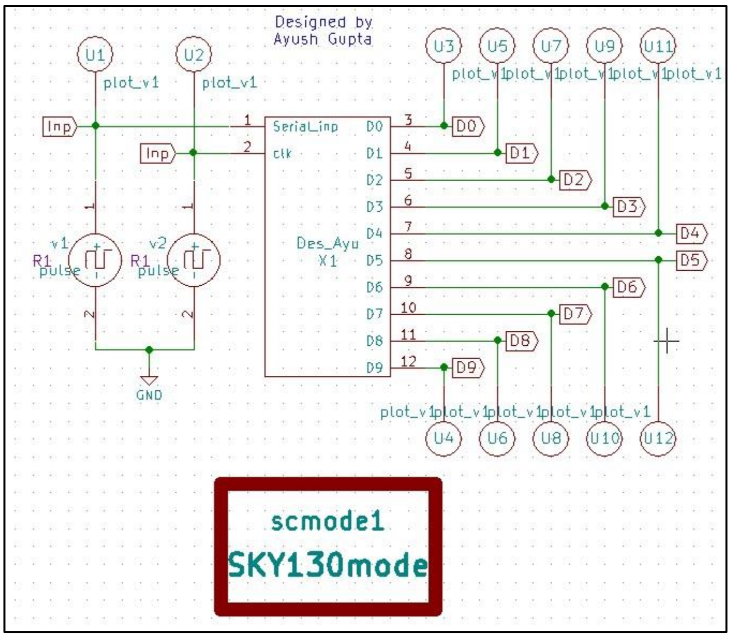


Fig. 13(b). Symbol and Testbench

8) Complete SerDes Setup

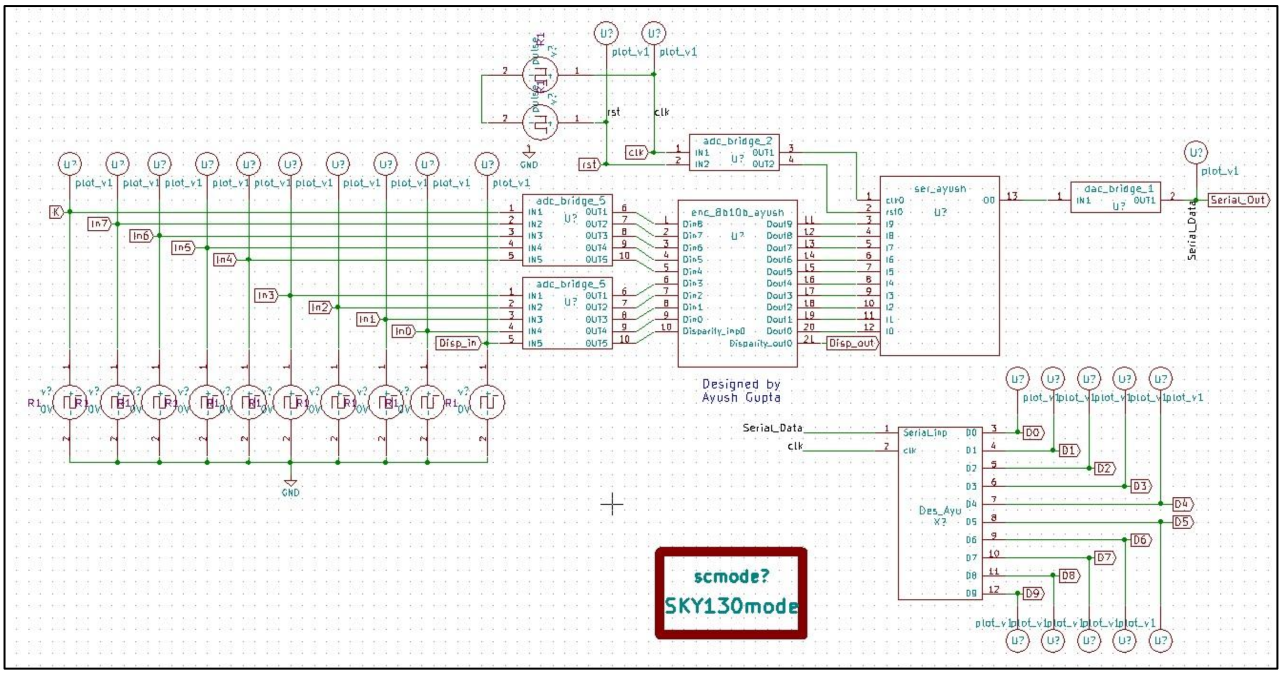
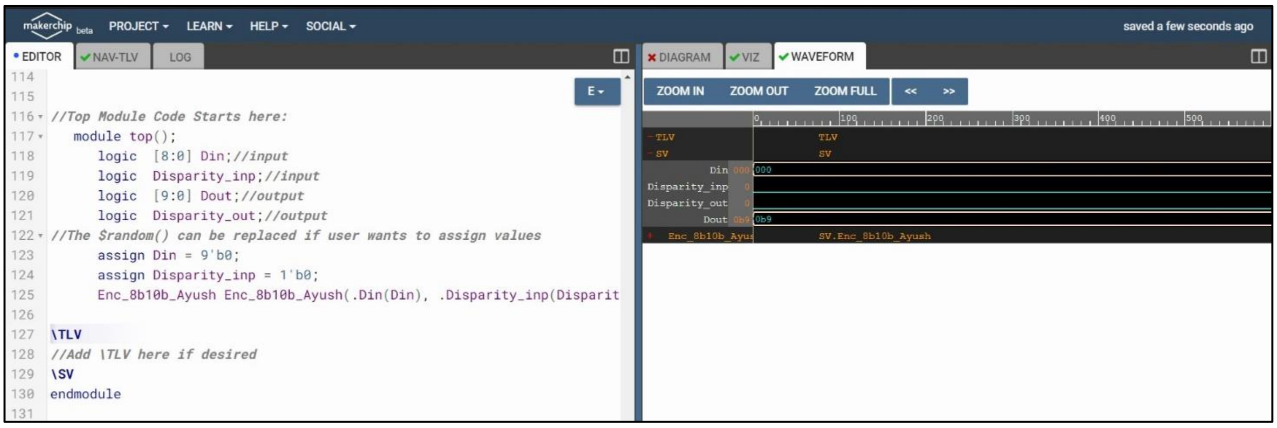


Fig. 14. SerDes Test Setup

Simulation Results

1) 8B10B Encoder

Makerchip IDE Simulation



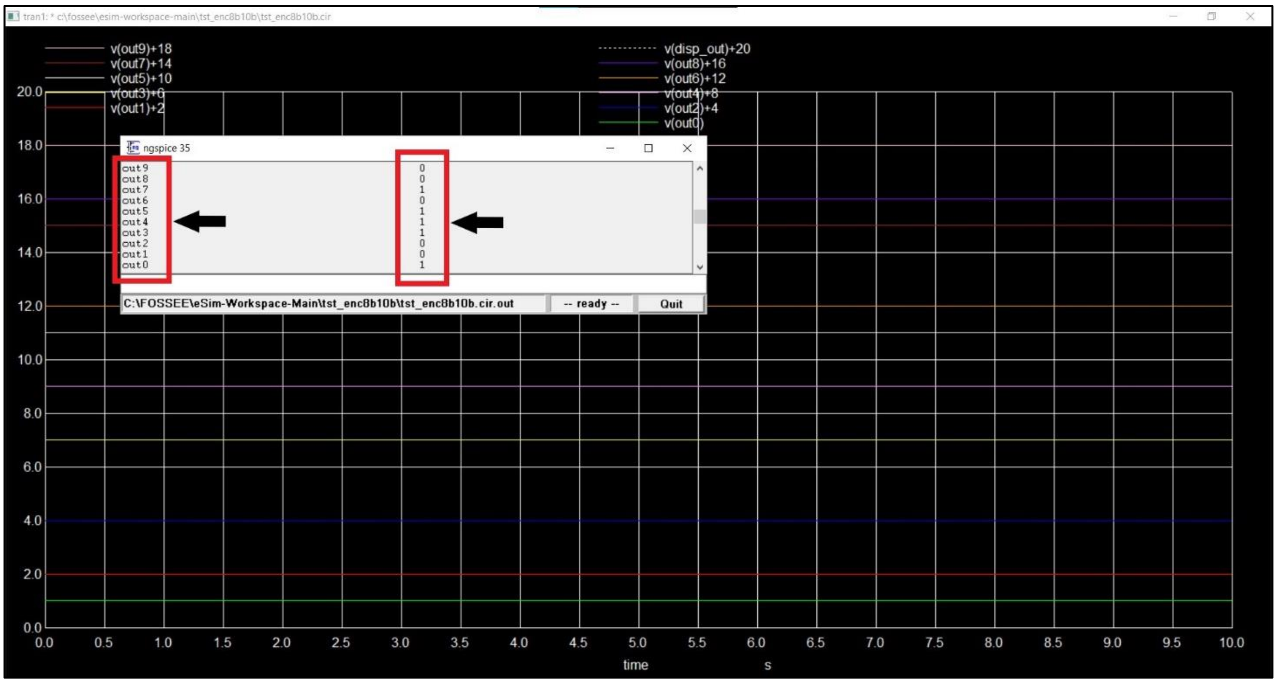
```
//Top Module Code Starts here:
   module top();
      logic  [8:0] Din;//input
      logic  Disparity_inp;//input
      logic  [9:0] Dout;//output
      logic  Disparity_out;//output
//The $random() can be replaced if user wants to assign values
      assign Din = 9'b0;
      assign Disparity_inp = 1'b0;
      Enc_8b10b_Ayush Enc_8b10b_Ayush(.Din(Din), .Disparity_inp(Disparit

\TLV
//Add \TLV here if desired
\SV
endmodule
```

```
Din            000   000
Disparity_inp  0
Disparity_out  0
Dout           0b9   0b9
```

Din = “000000000”
Dout = “0010111001” (0b9 in Hex)

NgSpice Plot

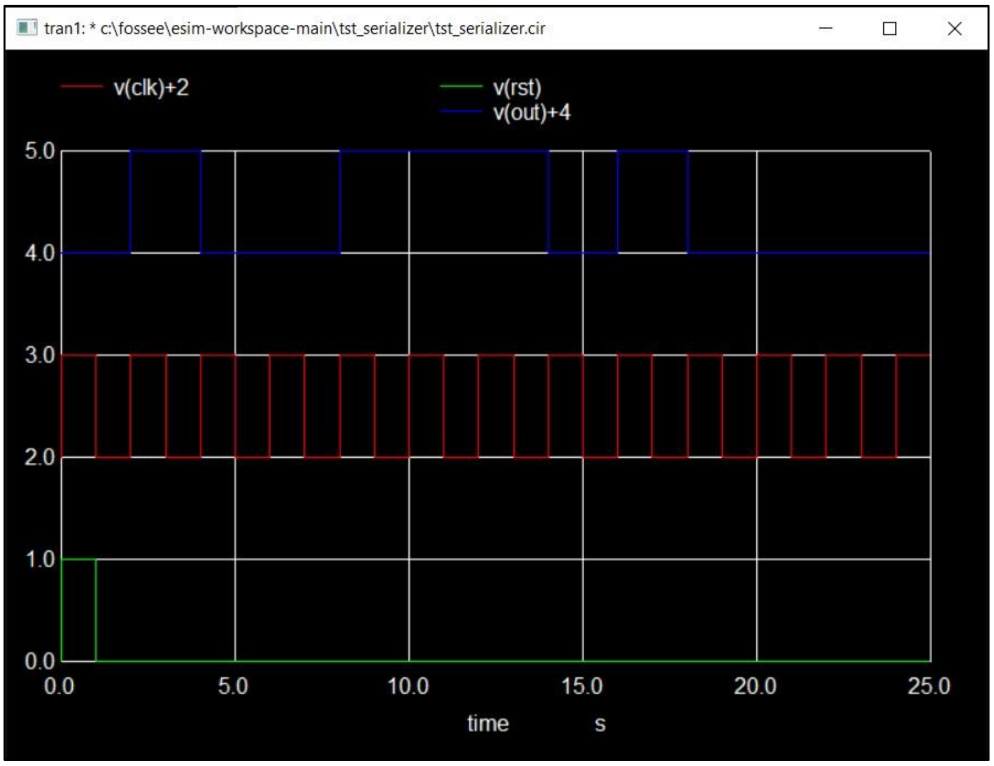


```
out9   0
out8   0
out7   1
out6   0
out5   1
out4   1
out3   1
out2   0
out1   0
out0   1
```

Encoder Output Voltage levels (MSB to LSB) = 0010111001 when Parallel Input = 000000000 (see fig. 9)

2) Serializer

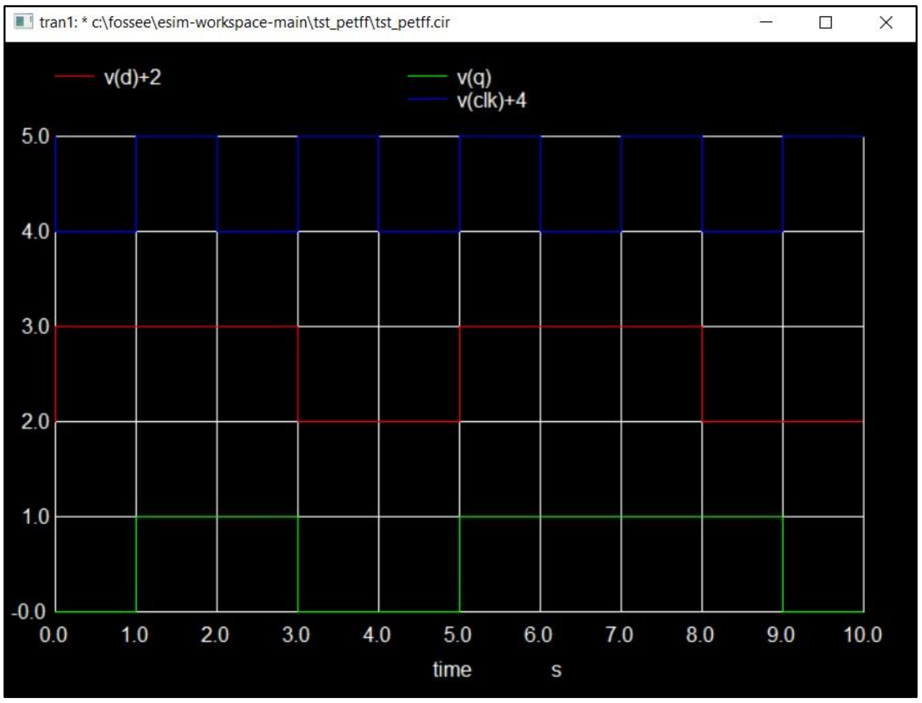
NgSpice Plot



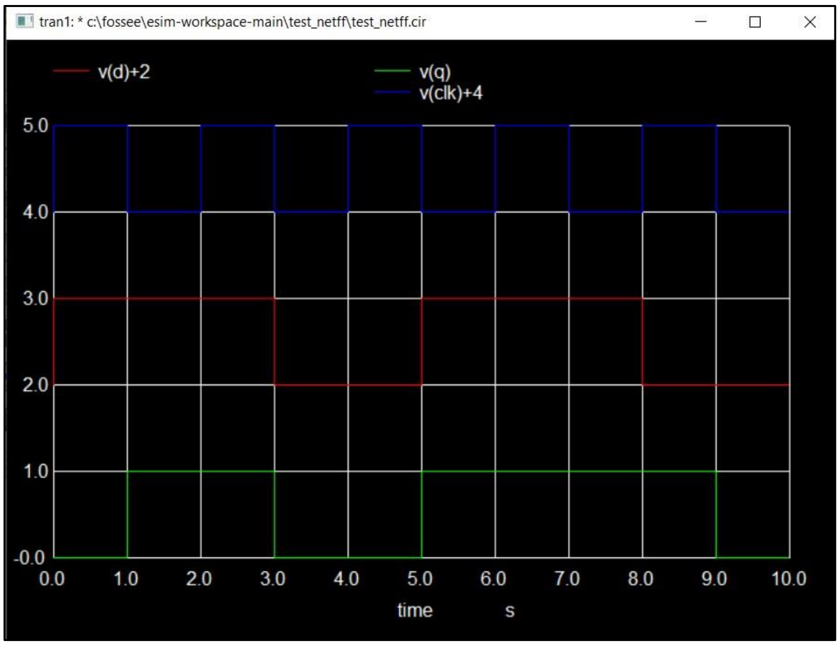
Serial Data Stream (LSB to MSB) = “1001110100”

3) Edge Triggered Flip-flops Test Simulation

NgSpice Plots



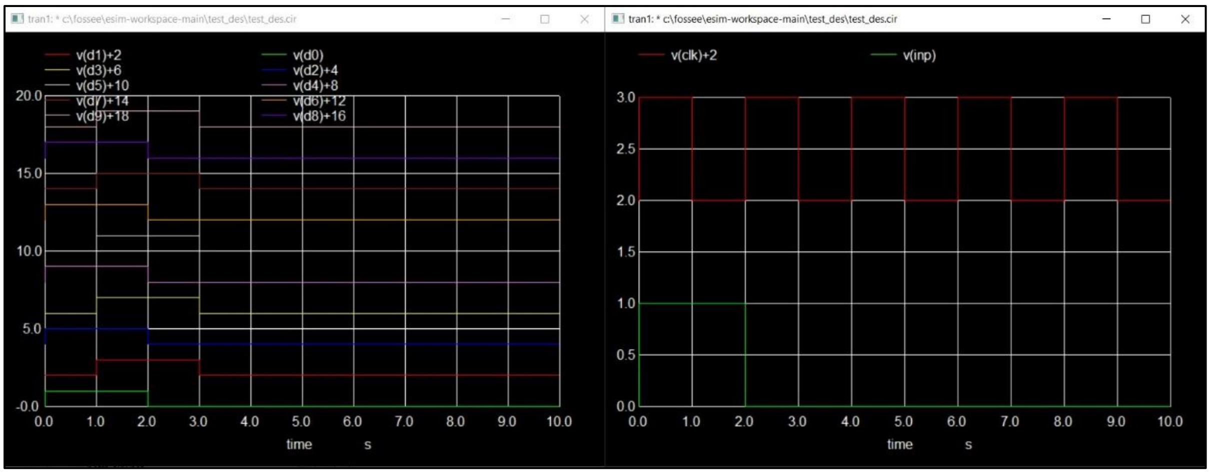
Positive Edge Triggered D Flip Flop Transient Analysis



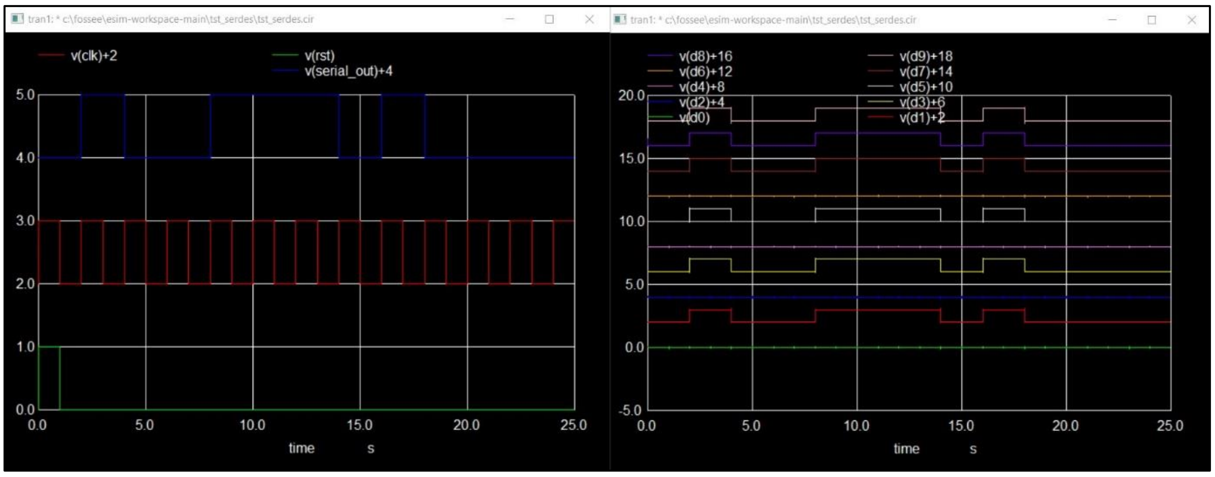
Negative Edge Triggered D Flip Flop Transient Analysis

4) Deserializer

NgSpice Plot



5) Complete SerDes Transient Analysis for Data Stream = 1001110100



Conclusion

Thus, the Mixed Signal Serializer/Deserializer blocks were designed and simulated. There output waveforms were studied using eSim NgSpice and the conversion of parallel encoded data to serial form and back to parallel was observed.

References

[1] Luo, Yifei, "A high speed serializer/deserializer design" (2010). Doctoral Dissertations. 536
[2] Arunthathi.G, Design of CMOS Serializer, Journal of Network Communications and Emerging Technologies (JNCET), Volume 8, Issue 6, June (2018)
[3] Nivedita Jaiswal, Radheshyam Gamad, Design of a New Serializer and Deserializer Architecture for On-Chip SerDes Transceivers, Circuits and Systems, 2015, 6, 81-92