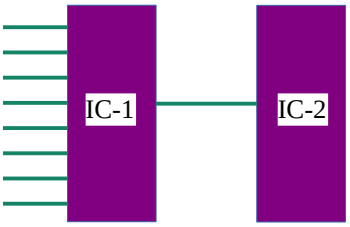
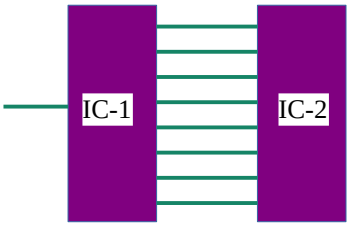
Serializer and Deserializer (SerDes)

What is SerDes?

SerDes is a functional block that Serializes and Deserializes digital data used in high-speed chip-to-chip communication. Modern SoCs for high-performance computing (HPC), artificial intelligence (AI), automotive, mobile, and Internet-of-Things (IoT) applications implement SerDes that can support multiple data rates and standards like PCI Express (PCIe), MIPI, Ethernet, USB, USR/XSR



Serializer



DeSerializer

A SerDes implementation includes parallel-to-serial (serial-to-parallel) data conversion, impedance matching circuitry, and clock data recovery functionality. The primary role of SerDes is to minimize the number of I/O interconnects.

Why do we need SerDes?

Distributed data processing in ICs need high speed data transfer between the ICs. Parallel and serial are the two options to transfer data between chips. Parallel data transfer requires multiple connections between ICs compared to serial data transfer that only needs one pair of connection.

SYNOPSYS®

Parallel: Multiple connections between chips	Serial: Single connection pair
• Consumes more power • Bigger ICs with complex packages • Susceptible to EM interference • Challenging skew balancing requirements • Practically no latency	• Saves power • Fewer pins makes compact IC • Robust EM performance • Clock can be recovered from data • Adds latency

Reference & learn more : https://www.synopsys.com/glossary/what-is-serdes.html#1

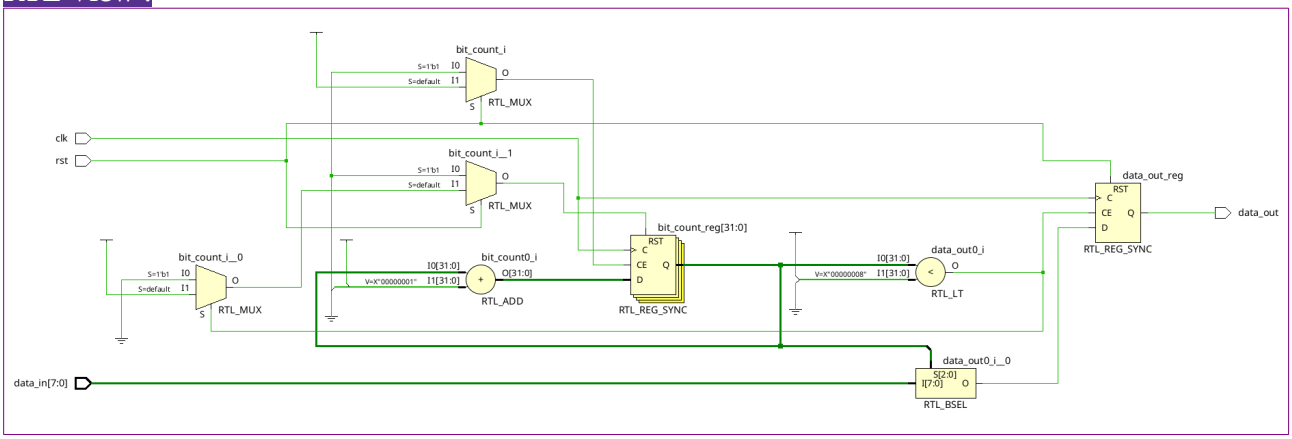
Verilog code for Serializes :

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: NA
// Engineer: Mamunar Rahoman
// Create Date: 11/01/2024 12:44:19 PM
// Design Name:
// Module Name: serializer
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//////////////////////////////////////////////////////////////////////////////////


module serializer(
          input clk,rst,
          input [7:0]data_in,
          output reg data_out
           );
     integer bit_count=0;

always@(posedge clk)
     begin
     if(rst==1'b1) //synchronous reset
          begin
          data_out<=1'b0;
          end
     else
          begin
               if(bit_count<8)
               begin
               data_out<=data_in[bit_count];
               bit_count=bit_count+1;
               end
               else
               begin
               bit_count=0;
               end
          end
     end
endmodule
```

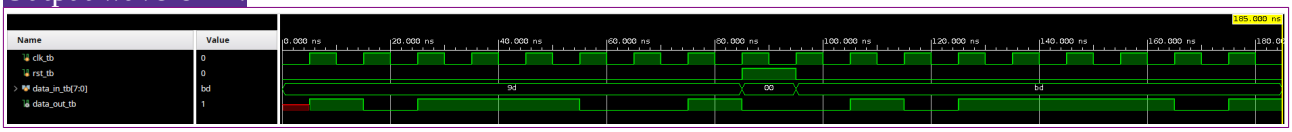
Testbench code for Serializes :

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: NA
// Engineer: Mamunar Rahoman
// Create Date: 11/01/2024 12:45:33 PM
// Design Name:
// Module Name: serializer_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//////////////////////////////////////////////////////////////////////////////////

module serializer_tb;
reg clk_tb;
reg rst_tb;
reg [7:0]data_in_tb;
wire data_out_tb;

serializer dut(
          .clk(clk_tb),
          .rst(rst_tb),
          .data_in(data_in_tb),
          .data_out(data_out_tb)
           );

initial
begin
     clk_tb=1'b0;
     forever #5 clk_tb=~clk_tb;
end

initial
begin
     #0 rst_tb=1'b0;data_in_tb=8'b10011101;
     #85 rst_tb=1'b1;data_in_tb=8'b0;
     #10 rst_tb=1'b0;data_in_tb=8'b10111101;
     #90 $stop;
end
endmodule
```

RTL View :



Output waveform :



Verilog code for Deserializes :

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: NA
// Engineer: Mamunar Rahoman
// Create Date: 11/01/2024 01:43:58 PM
// Design Name:
// Module Name: deserializer
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//////////////////////////////////////////////////////////////////////////////////

module deserializer(
                    input clk,rst,
                    input data_in,
                    output reg [7:0]data_out
                   );
    integer bit_count=0;
    reg [7:0]midpoint_out=8'b0;

always@(posedge clk or posedge rst) //asynchronous reset
    begin
        if(rst)
            begin
                data_out<=8'bx;
                midpoint_out<=8'b0;
                bit_count=0;
            end
```

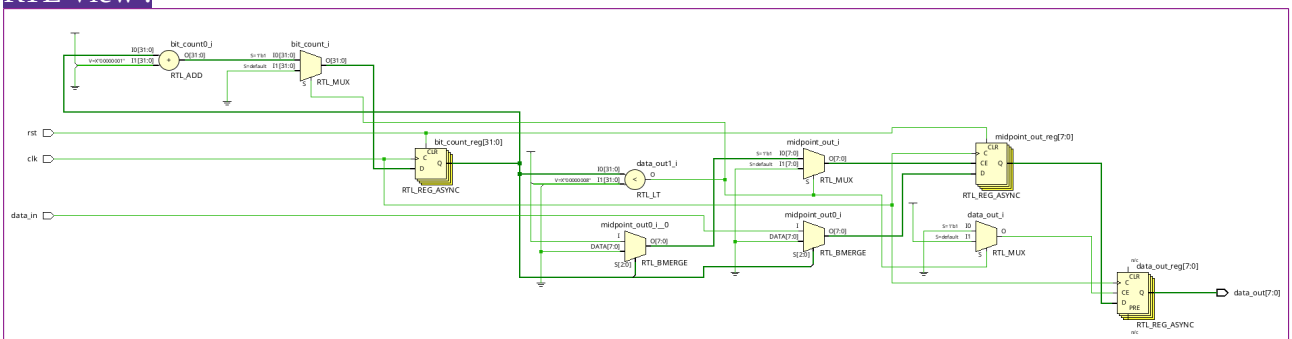
```
        else
            begin
                if(bit_count<8)
                    begin
                        midpoint_out[bit_count]<=data_in;
                        bit_count=bit_count+1;
                    end
                else
                    begin
                        bit_count=0;
                        data_out<=midpoint_out;
                    end
            end
    end
endmodule
```

Testbench code for Serializes :

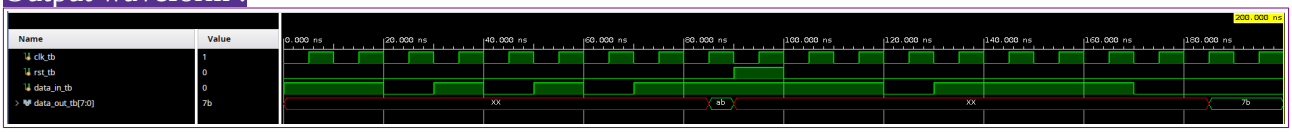
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: NA
// Engineer: Mamunar Rahoman
// Create Date: 11/01/2024 01:44:20 PM
// Design Name:
// Module Name: deserializer_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//////////////////////////////////////////////////////////////////////////////////


module deserializer_tb;
reg clk_tb;
reg rst_tb;
reg data_in_tb;
wire [7:0]data_out_tb;

deserializer dut(
            .clk(clk_tb),
            .rst(rst_tb),
            .data_in(data_in_tb),
            .data_out(data_out_tb)
                );
initial
begin
    clk_tb=1'b0;
    forever #5 clk_tb=~clk_tb;
end
initial
```

```
begin
    #0 rst_tb=0; data_in_tb=1'b1;
    #10 data_in_tb=1'b1;
    #10 data_in_tb=1'b0;
    #10 data_in_tb=1'b1;
    #10 data_in_tb=1'b0;
    #10 data_in_tb=1'b1;
    #10 data_in_tb=1'b0;
    #10 data_in_tb=1'b1;

    #20 rst_tb=1'b1;

    #10 rst_tb=1'b0; data_in_tb=1'b1;
    #10 data_in_tb=1'b1;
    #10 data_in_tb=1'b0;
    #10 data_in_tb=1'b1;
    #10 data_in_tb=1'b1;
    #10 data_in_tb=1'b1;
    #10 data_in_tb=1'b1;
    #10 data_in_tb=1'b0;

    #30 $finish;
end
endmodule
```

RTL View :



Output waveform :



-Prepared by Mamunar Rahoman

Website : https://sites.google.com/view/mamunar/home
LinkedIn : https://www.linkedin.com/in/mamunar-rahoman-b0ba64201/
GitHub : https://github.com/mamunarrahoman