EEL2020 Digital Design Lab Report

Sem II AY 2023-24

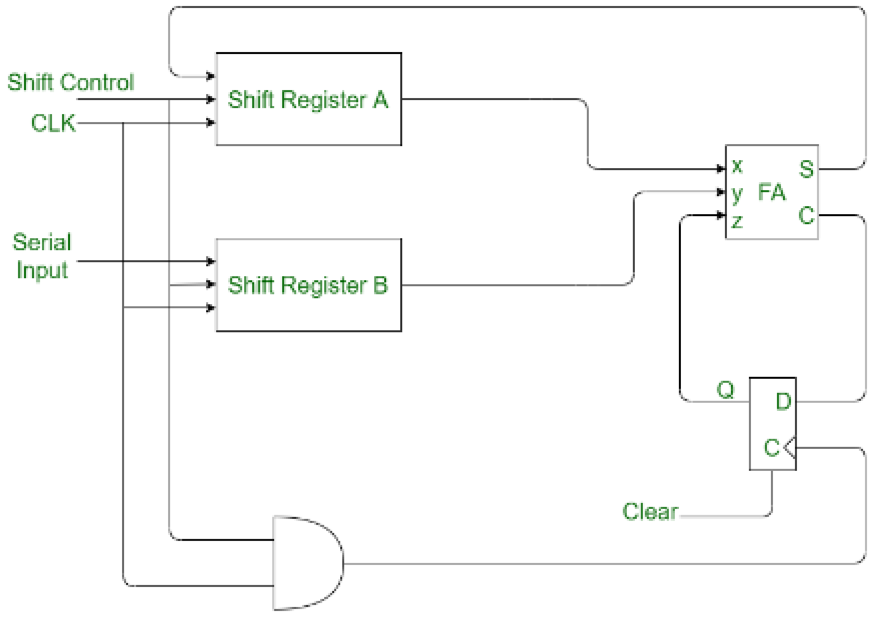
Experiment No.	**:**	**08**
Name	**:**	**Lavangi Parihar**
Roll No.	**:**	**B22EE044**
Partner Name (Roll Number)	**:**	**Mansi Choudhary (B22EE045)**

Objective

To implement a 4-bit serial adder using Verilog on the PYNQ Z2 Development Board using the Vivado Design Suite

Logic Design

A Serial adder performs bit by bit addition. Two shift registers are used to store the binary numbers that are to be added.
A single full adder is used to add one pair of bits at a time along with the carry. The carry output from the full adder is applied to a D flip flop After that output is used as carry for next significant bits. The sum bit from the output of the full adder can be transferred into a third shift register.



Block Diagram of Serial Binary Adder

Source Description

- Design source

```
module serialadder(DATA, loadA, loadB, start, clk, regA, regB, rst, co, cout_f);
input [3:0] DATA;
input loadA, loadB, start, clk, rst;
output reg [3:0] regA, regB;
output co;
output reg cout_f;

reg cin;
reg [27:0] count=0;
wire w1, w2, w3, cout, sum;

// CLK DIV
assign co = count[27];
always@(posedge clk)
count = count + 1;

// 1 bit FA
xor(sum, regA[0], regB[0], cin);
and(w1, regA[0], regB[0]);
and(w2, regA[0], cin);
and(w3, regB[0], cin);
or(cout, w1, w2, w3);

always@(posedge co)
begin
if (rst)
 begin
   regA <= 4'b0000;
   regB <= 4'b0000;
   cin <= 1'b0;
 end
 else if(loadA)
   regA <= DATA;
 else if(loadB)
```

```
  regB <= DATA;
 else if (start)
  begin
    regB <= regB >> 1;
    regA <= regA >> 1;
    regA[3] <= sum;
    cin <= cout;
    cout_f <= cout;
  end
end

endmodule
```

- Constraint file

The XDC file was updated with the following changes:

Ports (from Verilog module)	**PYNQ Component Type**
clk	H16
start	M20
rst	M19
co	N15
regA	R14 P14 N16 M14
DATA	D19 D20 L20 L19

The RPI file was updated with the following changes:

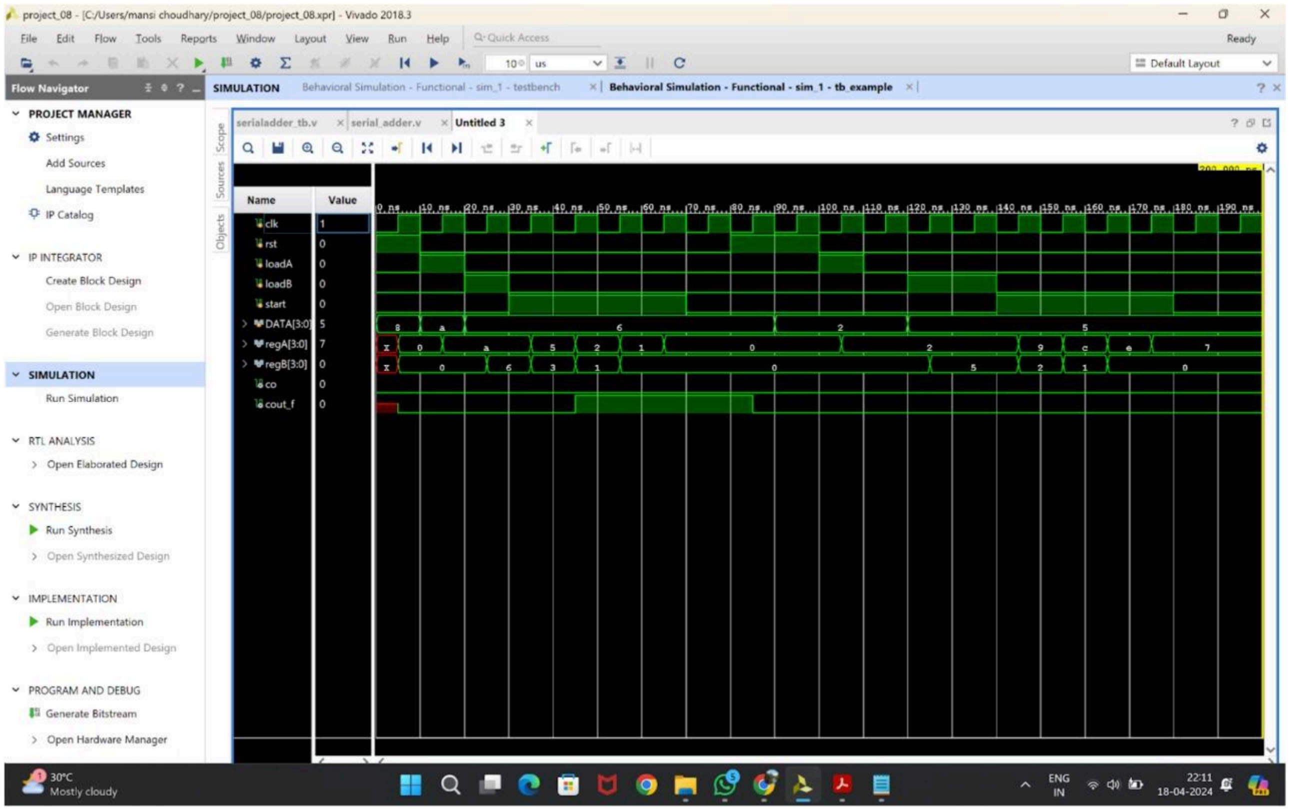
Ports (from Verilog module)	**PRI Component Type**
regB	B20 W8 U8 W6
cout_f	F20
loadA	V6
loadB	Y6

- Simulation source (if any)

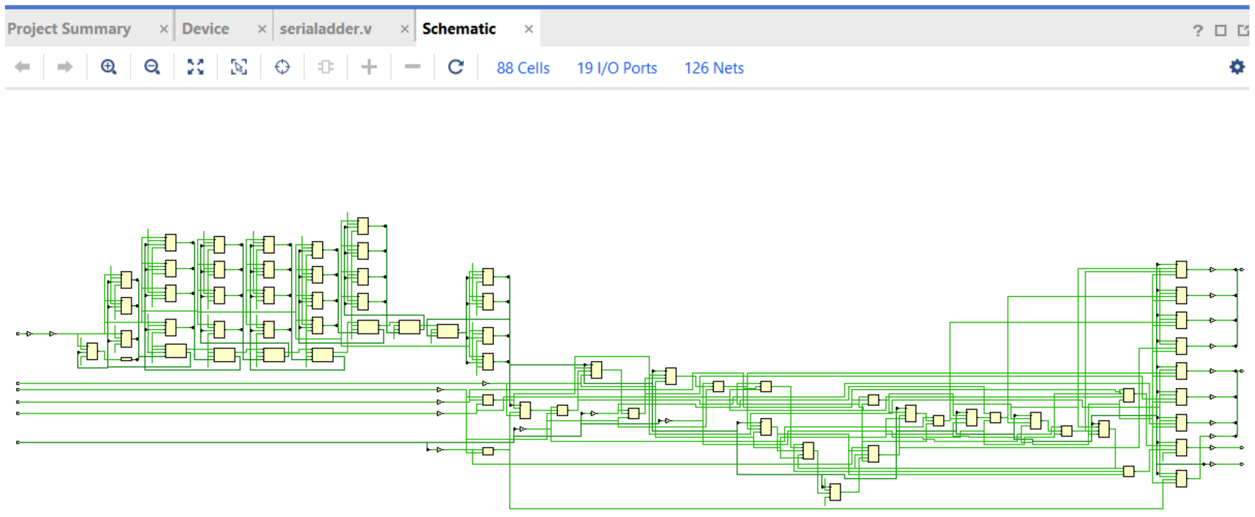
```
module tb_example();
 // Inputs
   reg clk, rst, loadA, loadB, start;
   reg [3:0] DATA;

   // Outputs
   wire [3:0] regA, regB;
   wire co,cout_f;

   // Instantiate the design module
   serialadder dut (
      .clk(clk),
      .rst(rst),
      .loadA(loadA),
      .loadB(loadB),
      .start(start),
      .regA(regA),
      .regB(regB),
      .DATA(DATA),
      .co(co),
      .cout_f(cout_f)
   );

   // Clock generation
always #5 clk = ~clk;
   // Test stimulus
   initial begin
      // Initialize inputs
      clk = 0;
      rst = 1;
      loadA = 0;
      loadB = 0;
      start = 0;
      DATA = 4'b1010; // Example input

 // Apply reset
      #10;
      rst=0;
```

```
    // Load data into register A
    loadA = 1;
    DATA = 4'b1100; // New example input for register A
    #10;
    loadA = 0;


    // Load data into register B
    loadB = 1;
DATA = 4'b0110; // New example input for register B
    #10;
    loadB = 0;


    // Start the process
    start = 1;
    #40;
    start = 0;
    #10;
    rst=1;
    #10;


    rst=1;
    DATA=4'b0010;
    #10;

    rst=0;
    loadA=1;
    #10;
    loadA=0;
    #10;
    DATA=4'b0101;
    loadB=1;
    #20;
    loadB=0;
    start=1;
    #40;
    start=0;
    #20



    // End simulation
    $finish;
    end

endmodule
```

Simulation Results (Timing diagram)



Elaborated Design



PYNQ Working Video

Video Link