Experiment - 4: Serial Adder Simulation Using Quartus Prime

NAME: Jayakrishnan Menon

REG NO: 22BEC1205

DATE: 20/01/2025 and 03/02/2025

Aim:
Write a Verilog RTL code for a Serial Adder. Also, perform the simulation of the Full Adder using Quartus Prime and Model Sim. Finally, implement these circuits on the FPGA kit, "5CSXFC6D6F31C6N"

Software Required: Quartus Prime**,** ModelSim

Haraware Required: Altera Cyclone V 5CSXFC6D6F31C6N

Procedure:

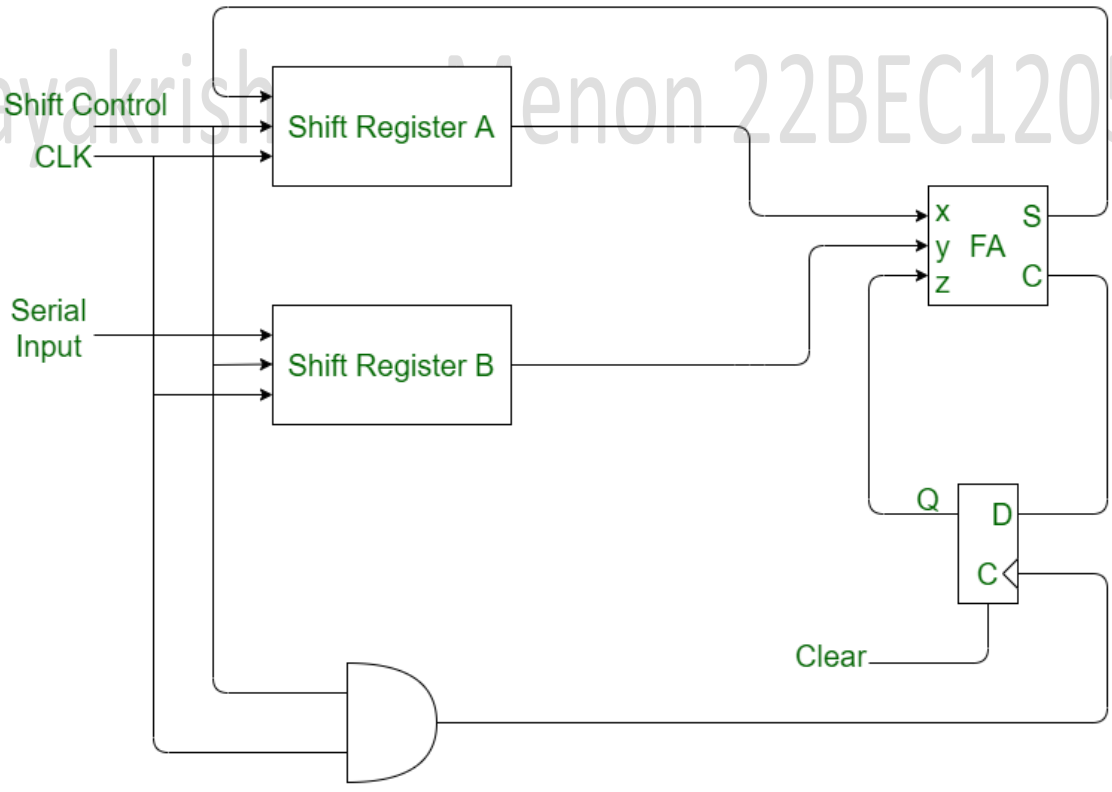
- Open Quartus Prime 21.1.
- Go to File -> New Project wizard -> select Source folder & type file name -> Next.
- Select Empty Project in Project type -> Next.
- Click on Next in Add Files dialog box.
- Select "Cyclone V SX Extended Features" in Device Family.
- In available devices, select the one ending with "31C6" -> Next.
- Select the tool name as "ModelSim" and Format as "VerilogHDL" in simulation.
- Click on Finish. The project is now created.
- In the Task window, select RTL simulation and run, this would open the ModelSim window.
- Simulate the full adder as you would do using ModelSim by forcing the input values.
- In Compilation -> select compile design.
- Go to Assignments tab -> Pin planner -> Give the location for each i/o pin.
- Go to Hardware Setup -> Select USB.
- Change the file to Fulladder.v in the program/configure option and select Start.

Theory

Serial binary adder is a combinational logic circuit that performs the addition of two binary numbers in serial form. Serial binary 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.

Shift Register is a group of flip flops used to store multiple bits of data. There are two shift registers used in the serial binary adder. In one shift register augend is stored and in other shift register addend is stored. Full adder is the combinational circuit which takes three inputs and gives two outputs as sum and carry. The circuit adds one pair at a time with the help of it. the carry output from the full adder is applied on the D flip-flop. Further, the output of D flip-flop is used as a carry input for the next pair of significant bits.



Block Diagram of Serial Binary Adder

Following is the procedure of addition using serial binary adder:

- Step-1:
  The two shift registers A and B are used to store the numbers to be added.
- Step-2:
  A single full adder is used too add one pair of bits at a time along with the carry.
- Step-3:
  The contents of the shift registers shift from left to right and their output starting from a and b are fed into a single full adder along with the output of the carry flip-flop upon application of each clock pulse.
- Step-4:
  The sum output of the full adder is fed to the most significant bit of the sum register.
- Step-5:
  The content of sum register is also shifted to right when clock pulse is applied.
- Step-6:
  After applying four clock pulse the addition of two registers (A & B) contents are stored in sum register.

Source code and Outputs:

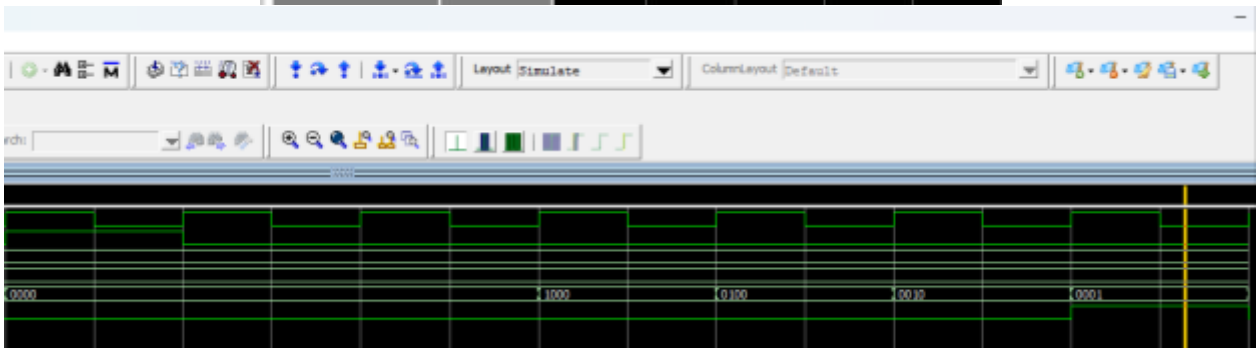
```
module full_adder(a, b, cin, sum, cout);
  input a, b, cin;
  output sum, cout;

  assign {cout, sum} = a + b + cin;

endmodule

module d_flipflop(d, clk, enable, reset, out);
  input d, clk, enable, reset;
  output out;

  reg out;

  always @ (posedge clk or posedge reset) begin
    if (reset)
      out = 0;
    else
      if (enable)
        out = d;
  end
endmodule
```



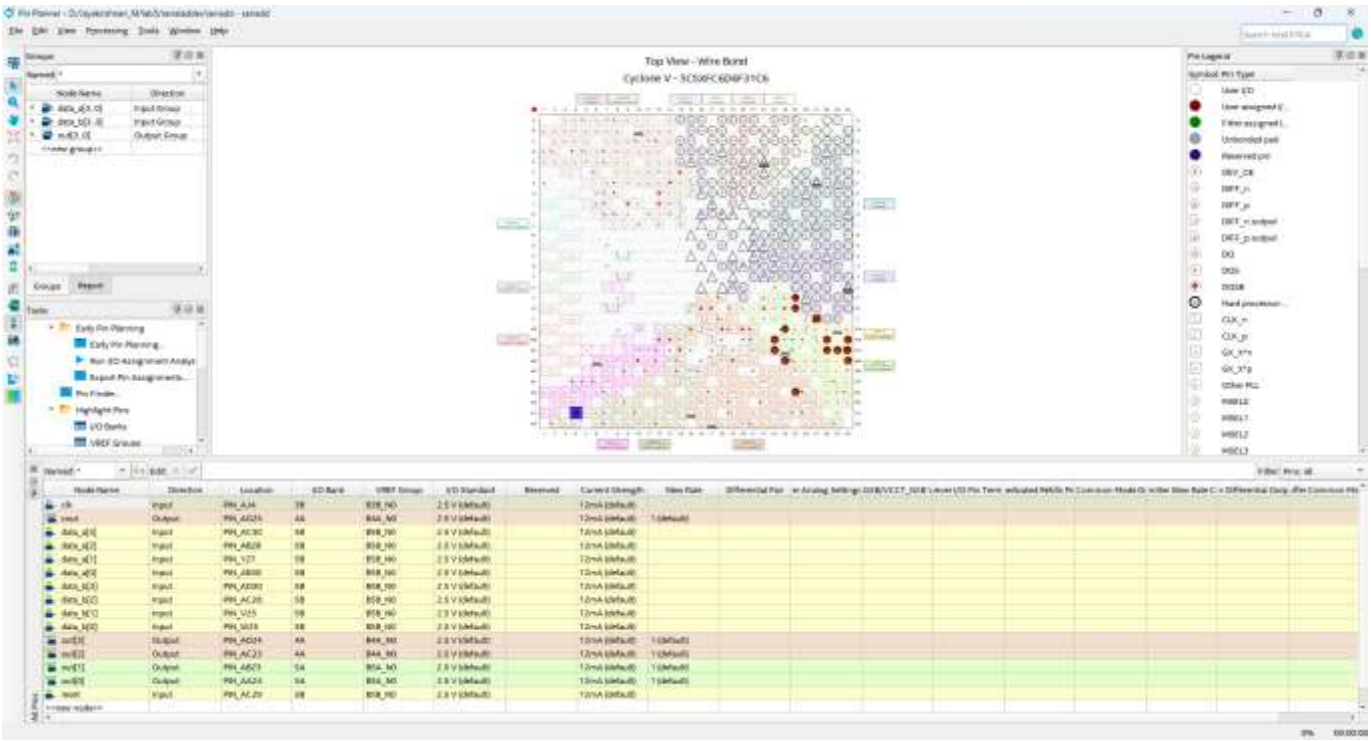
```
module piso(clk, enable, rst, data, out);
  input enable, clk, rst;
  input [3:0] data;
  output out;

  reg out;
  reg [3:0] memory;

  always @ (posedge clk, posedge rst) begin
    if (rst == 1'b1) begin
      out <= 1'b0;
      memory <= data;
    end
    else begin
      if (enable) begin
        out = memory[0];
        memory = memory >> 1'b1;
      end
    end
  end
endmodule

module seriadd(data_a, data_b, clk, reset, out, cout);
  input [3:0] data_a, data_b;
  input clk, reset;
```

```
output cout;
output [3:0] out;

reg [3:0] out;
reg [2:0] count;
reg enable, cout;
wire wire_a, wire_b, cout_temp, cin, sum;

piso piso_a(clk, enable, reset, data_a, wire_a);
piso piso_b(clk, enable, reset, data_b, wire_b);
full_adder adder(wire_a, wire_b, cin, sum, cout_temp);
d_flipflop dff(cout_temp, clk, enable, reset, cin);

always @ (posedge clk or posedge reset) begin
  if (reset) begin
    enable = 1;  out = 4'b0000;
  end
  else begin
        cout = cout_temp;
        out = out >> 1;
        out[3] = sum;
      end
   end

endmodule
```

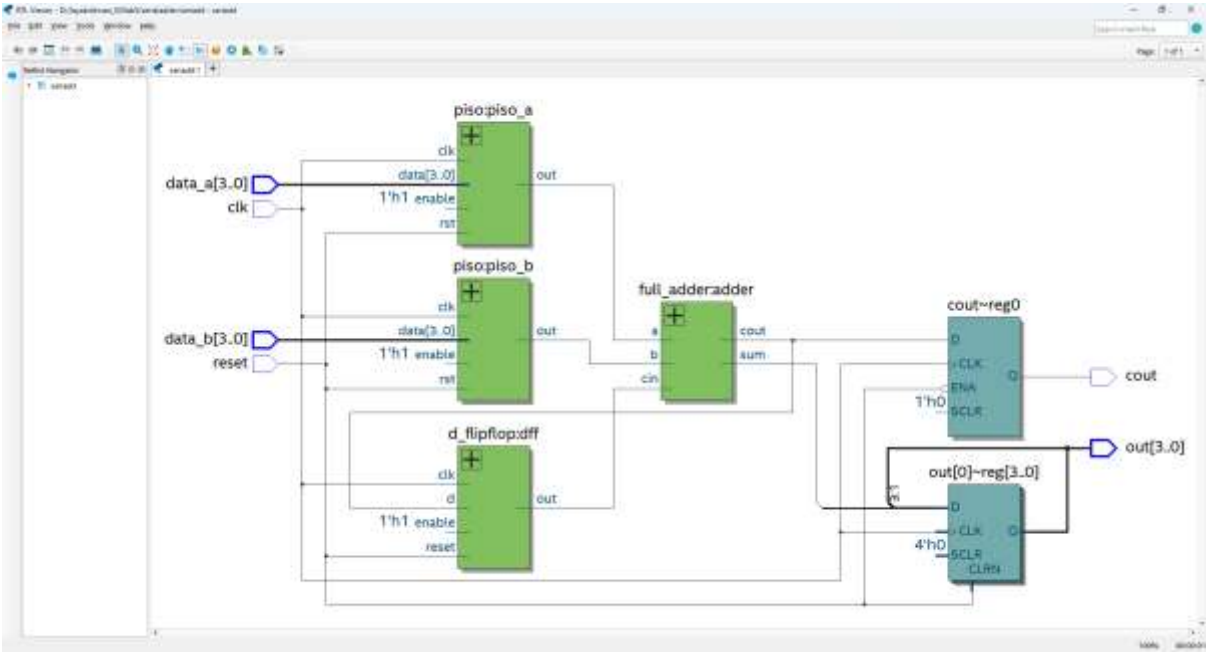

Output:



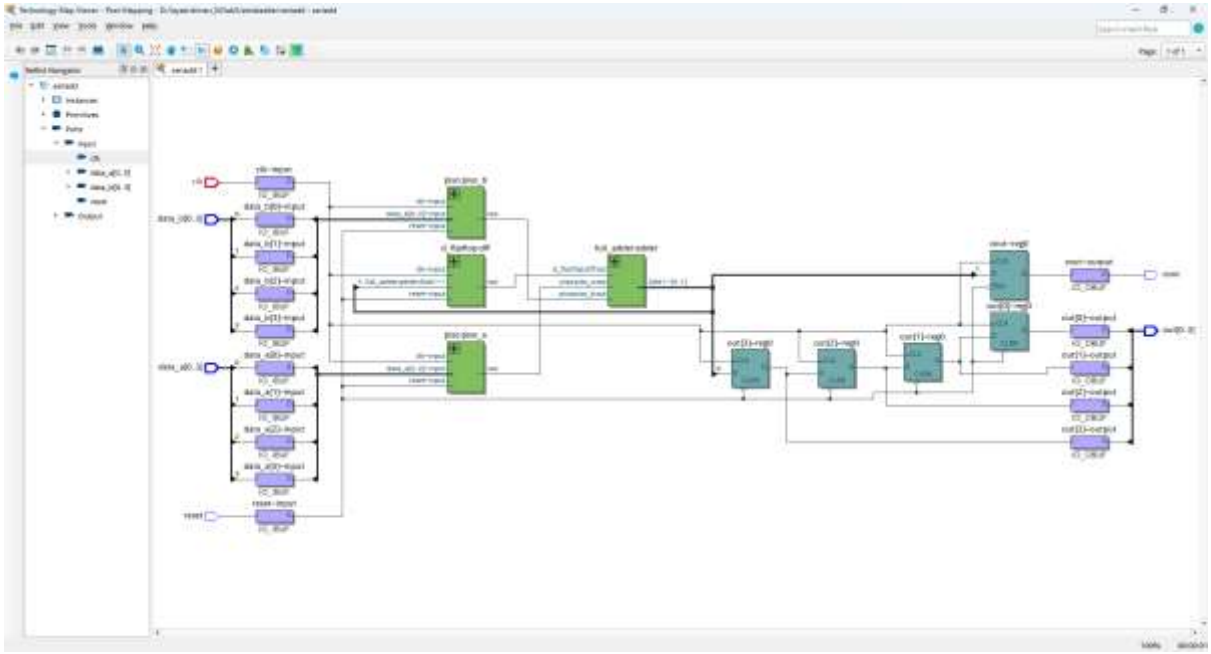
Pin Planning:



RTL Viewer:



Technology Map Viewer:



Hardware Implementation:

Input: a=0111, b=1000

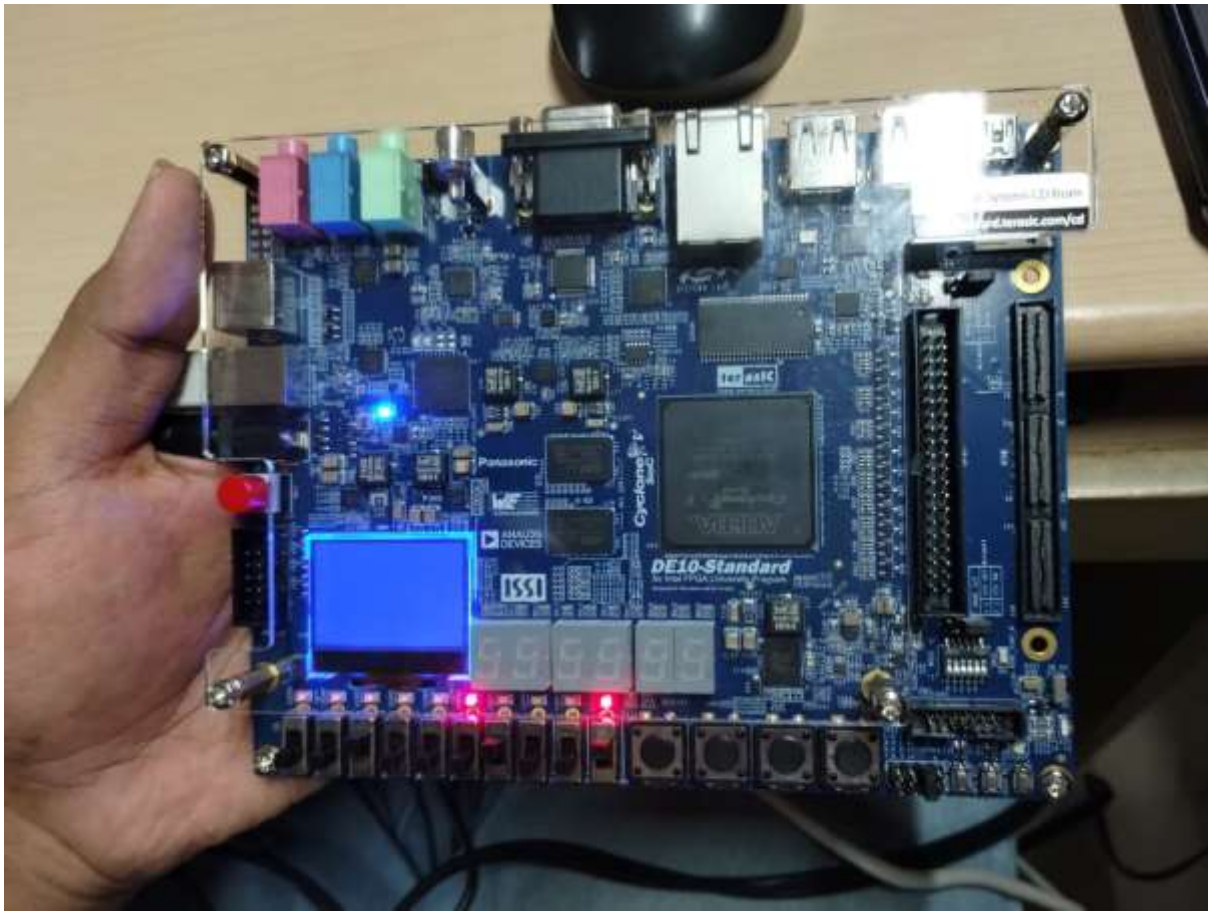
Output: c=1111 (7+8=15)

Input: a=0011, b=0011

Output: c=0110 (3+3=6)

Input: a=1000, b=1001

Output: c=10001 (8+9=17)



Result:

Hence, Verilog RTL code for a Serial Adder, was successfully verified and implemented using Quartus Prime.