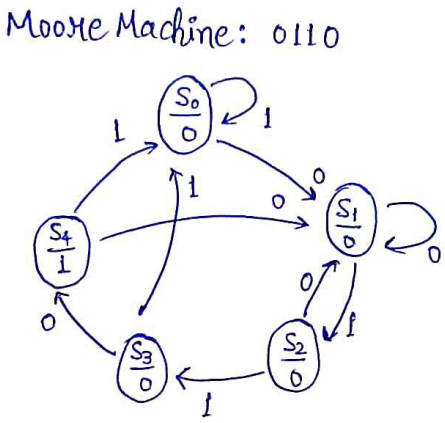
# Digital Electronics Verilog Lab 5

**Submitted By:**
Saurabh Kumar
SC ID: SC22B146
Course: DIGITAL ELECTRONICS AND VLSI DESIGN LAB (AV232)

## LAB SHEET

**Question No. 1** Design a state machine for detecting sequence "0110" using Moore and Mealy state machine
**Sol: Moore Machine:**

**State Diagram:**

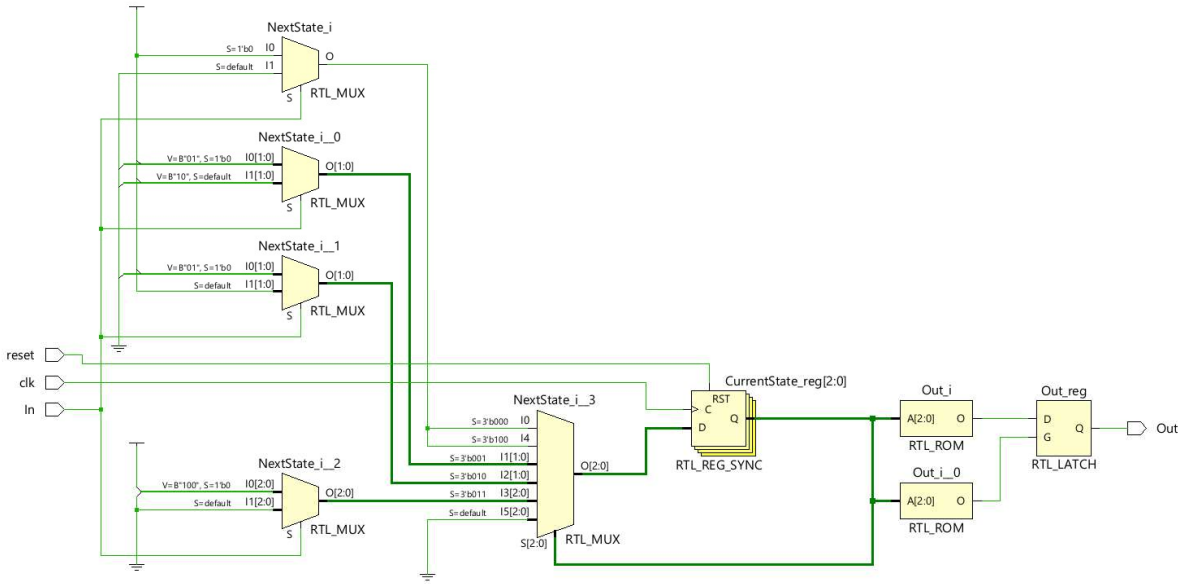



**Code:**

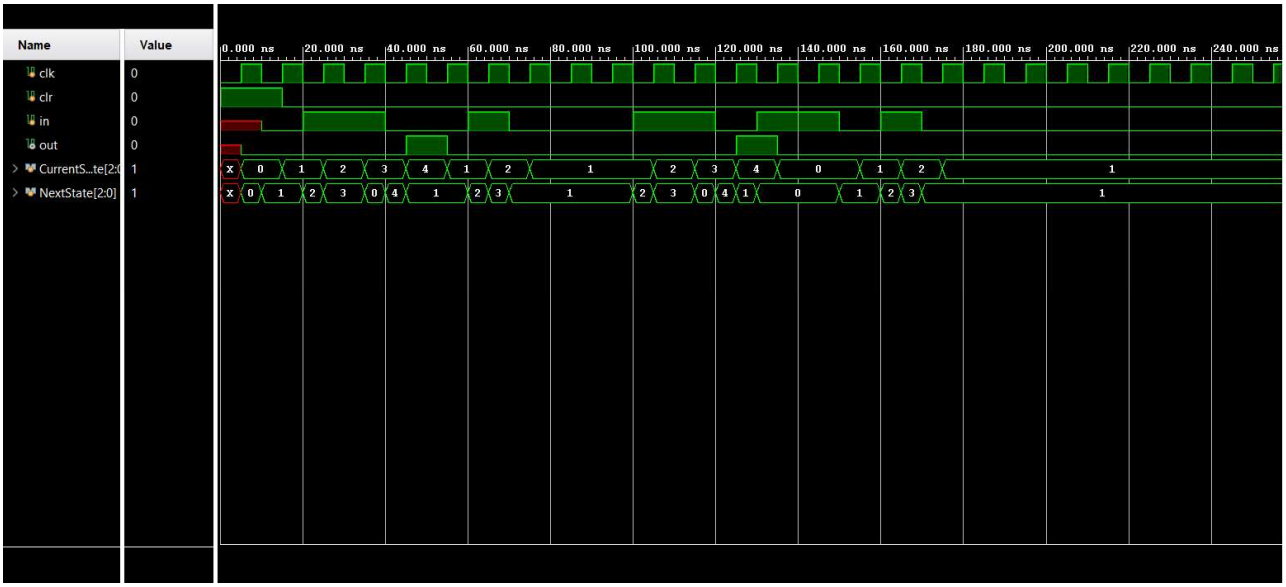
```
module detect_0110_Moore(clk,reset,In,Out);
   input clk, reset, In;
   output reg Out;

parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100;
reg[2:0] CurrentState,NextState;

always@(posedge clk)
begin
if(reset)
   CurrentState = S0;
else
   CurrentState = NextState;
end

always@(CurrentState, In)
begin
case(CurrentState)
S0:begin
   if(!In) NextState = S1;
   else NextState = S0;
```

```
   end
S1:begin
   if(!In) NextState = S1;
   else NextState = S2;
   end
S2:begin
   if(!In) NextState = S1;
   else NextState = S3;
   end
S3:begin
   if(!In) NextState = S4;
   else NextState = S0;
   end
S4:begin
   if(!In) NextState = S1;
   else NextState = S0;
   end
default:NextState = S0;
endcase
end

always@(CurrentState)
begin
case(CurrentState)
S0: Out = 0;
S1: Out = 0;
S2: Out = 0;
S3: Out = 0;
S4: Out = 1;
endcase
end

endmodule
```

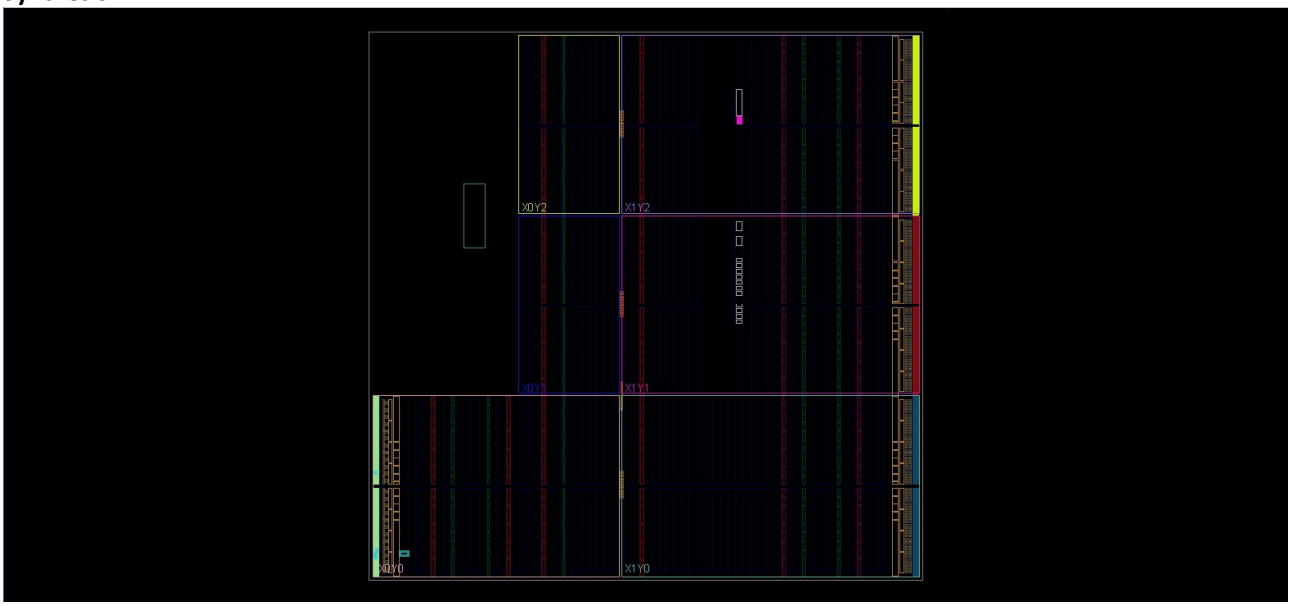
**Schematic:**




**Simulation using TestBench:**
Input Given: 01100100011011010

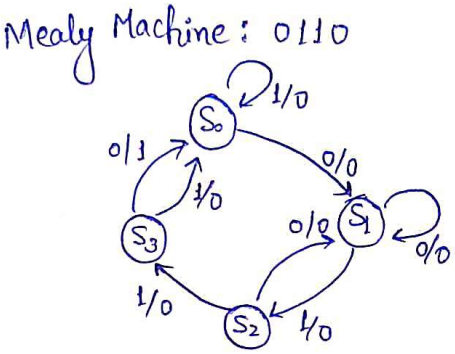



**Synthesis:**




**Mealy Machine:**

**State Diagram:**



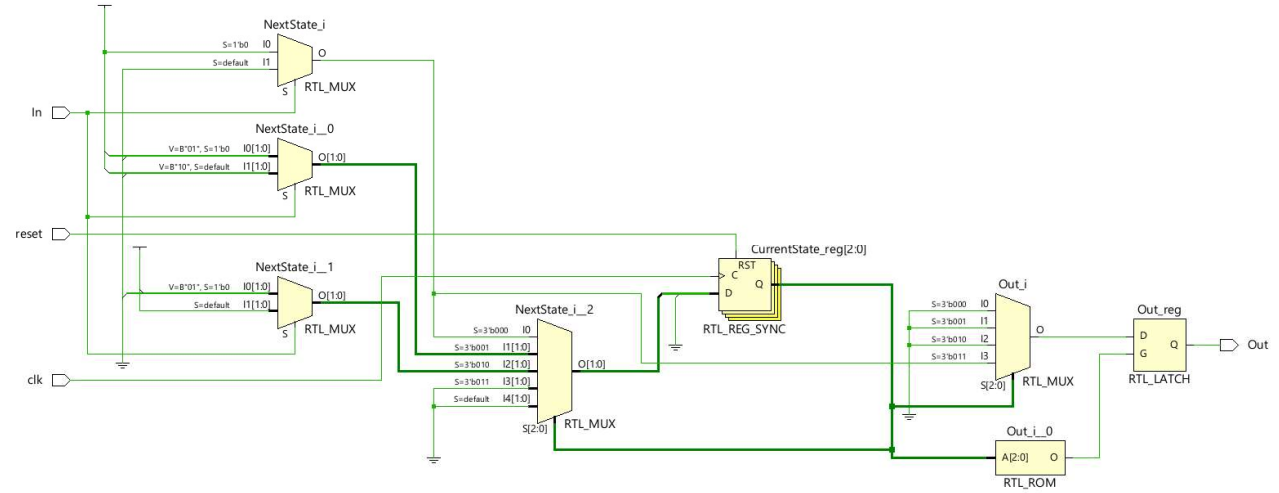

**Code:**

```
module detect_0110_Mealy(clk,reset,In,Out);
   input clk, reset, In;
```

```
    output reg Out;

parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011;
reg[2:0] CurrentState,NextState;

always@(posedge clk)
begin
if(reset)
    CurrentState = S0;
else
    CurrentState = NextState;
end

always@(CurrentState, In)
begin
case(CurrentState)
S0:begin
    if(!In)
        begin
        NextState = S1;
        Out = 0;
        end
    else
        begin
        NextState = S0;
        Out = 0;
        end
    end
S1:begin
    if(!In)
        begin
        NextState = S1;
        Out = 0;
        end
    else
        begin
        NextState = S2;
        Out = 0;
        end
    end
S2:begin
    if(!In)
        begin
        NextState = S1;
        Out = 0;
        end
    else
        begin
        NextState = S3;
        Out = 0;
        end
    end
S3:begin
    if(!In)
        begin
        NextState = S0;
        Out = 1;
        end
    else
        begin
        NextState = S0;
        Out = 0;
```

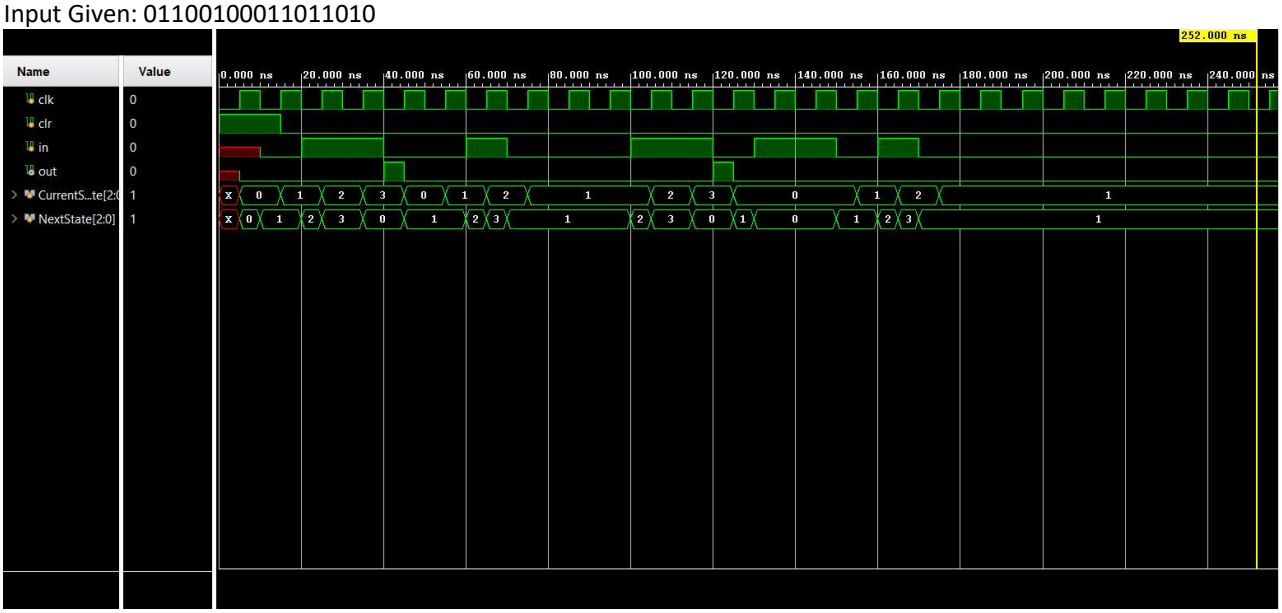
```
      end
   end
default:NextState = S0;
endcase
end

Endmodule

endmodule
```

**Schematic:**

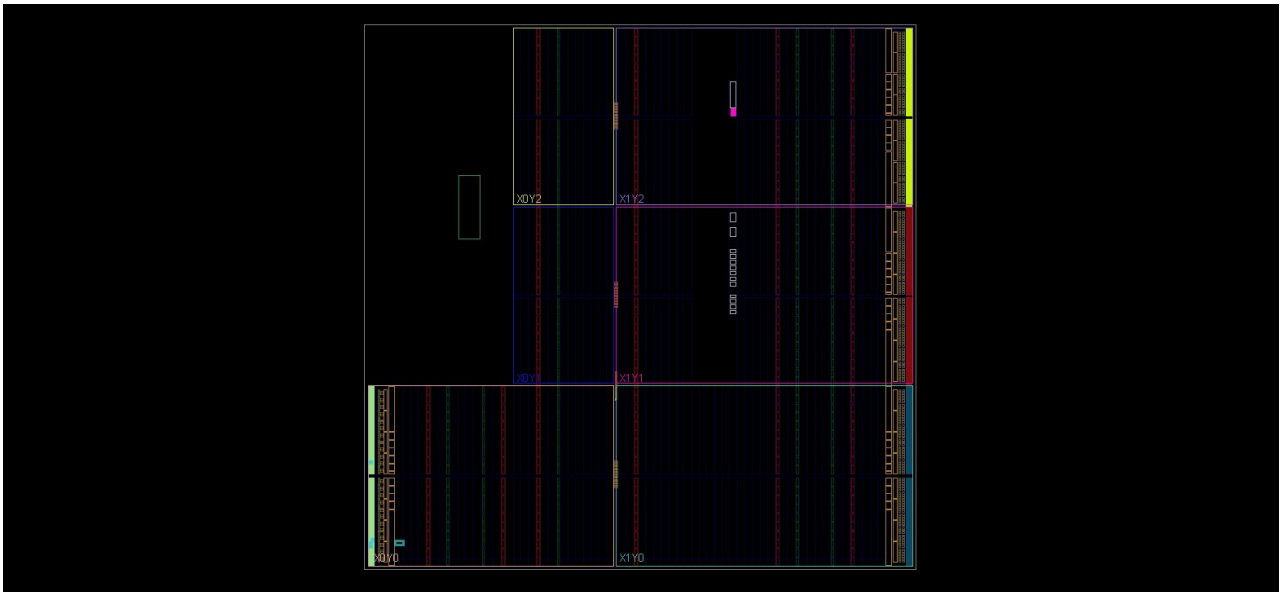



**Simulation using TestBench:**

Input Given: 01100100011011010




**Synthesis:**

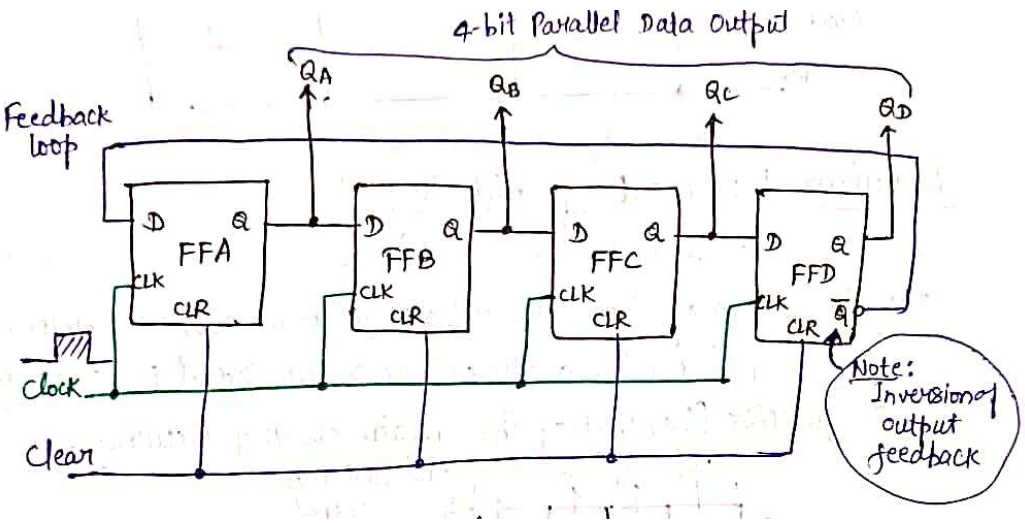



**Question No. 2** Design a 4-bit Johnson counter and implement on FPGA.

**Sol: <u>Implementation using D Flip Flop (DFF Code):</u>**

```
module DFF(clk,reset,D,Q,Qbar);
   input D,clk,reset;
   output reg Q, Qbar;

always@(posedge clk)
begin
   if(reset)
      begin
      Q <= 0;
      Qbar <= 1;
      end
   else
      begin
      Q <= D;
      Qbar <= ~D;
      end

end
endmodule
```

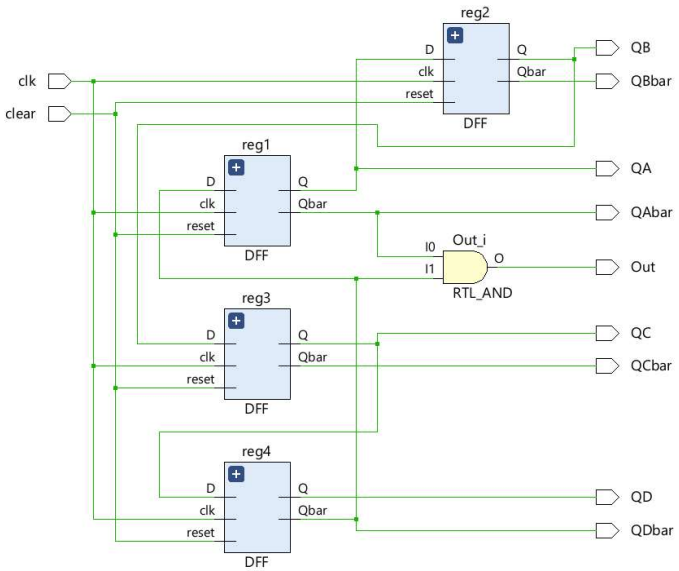
**<u>4-bit Johnson counter</u>**



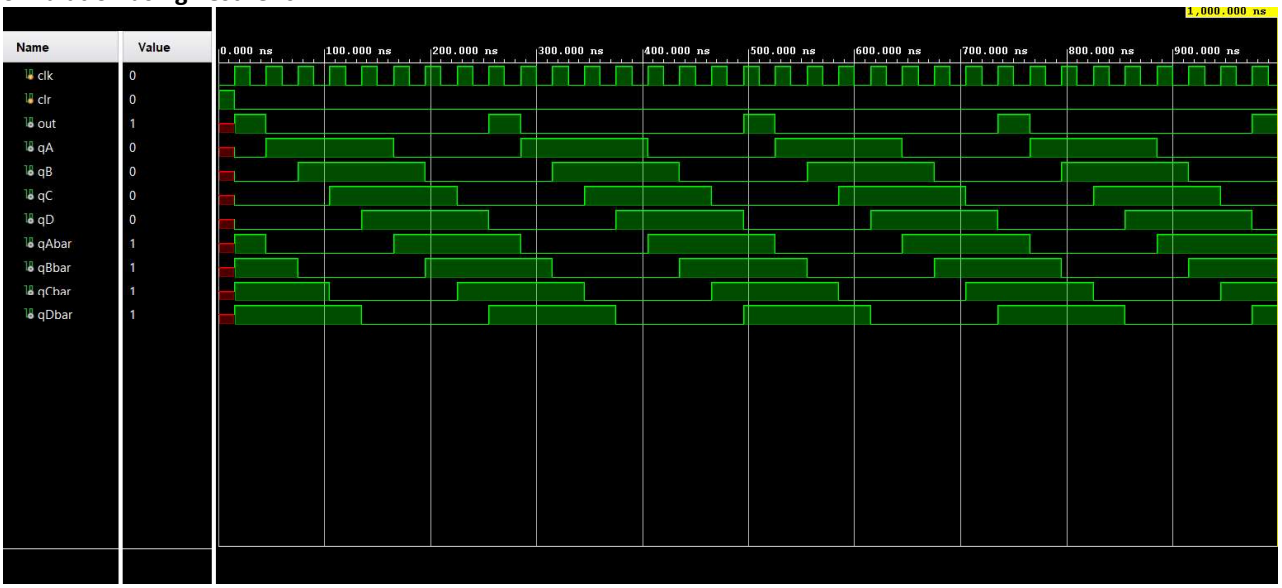

**Code:**

```
module JohnsonCount(clk,clear,Out,QA,QB,QC,QD,QAbar,QBbar,QCbar,QDbar);
   input clk,clear;
   output Out;
   output QA,QB,QC,QD,QAbar,QBbar,QCbar,QDbar;
   wire qa,qb,qc,qd;

   assign QA = qa, QB = qb, QC = qc, QD = qd;
   assign Out = (QAbar & QDbar);
   DFF reg1(clk,clear,QDbar,qa,QAbar);
   DFF reg2(clk,clear,qa,qb,QBbar);
   DFF reg3(clk,clear,qb,qc,QCbar);
   DFF reg4(clk,clear,qc,qd,QDbar);
endmodule
```

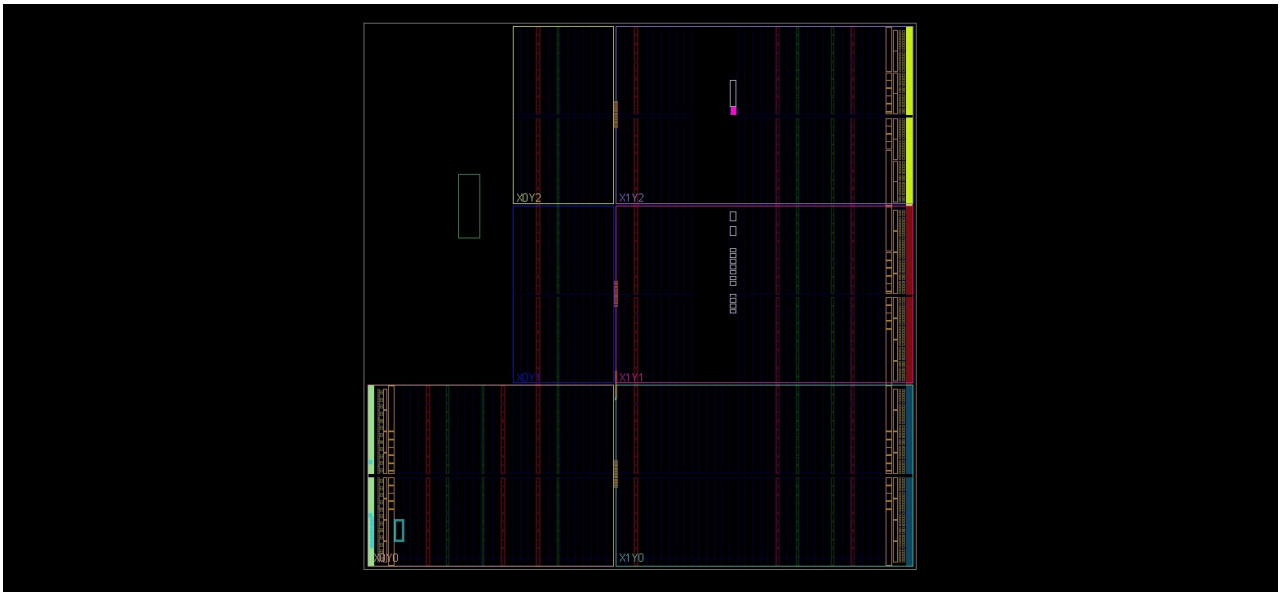
**Schematic:**



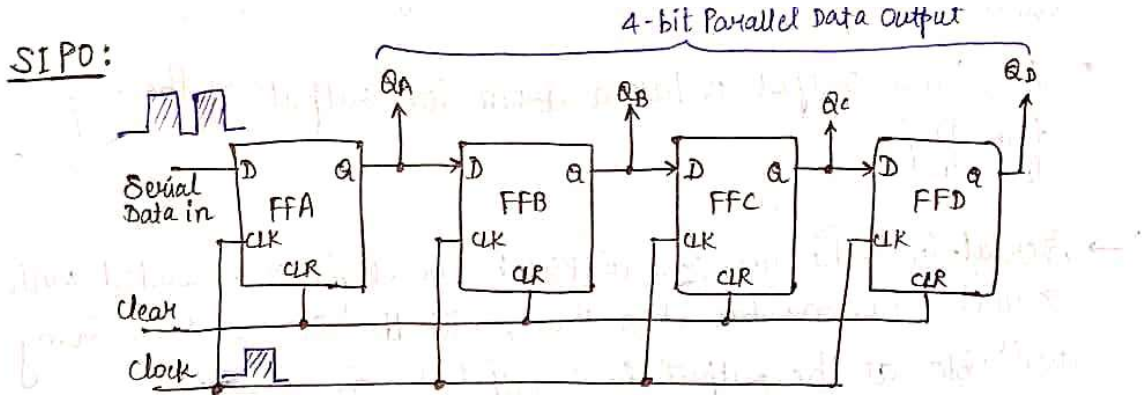

**Simulation using TestBench:**




**Synthesis:**



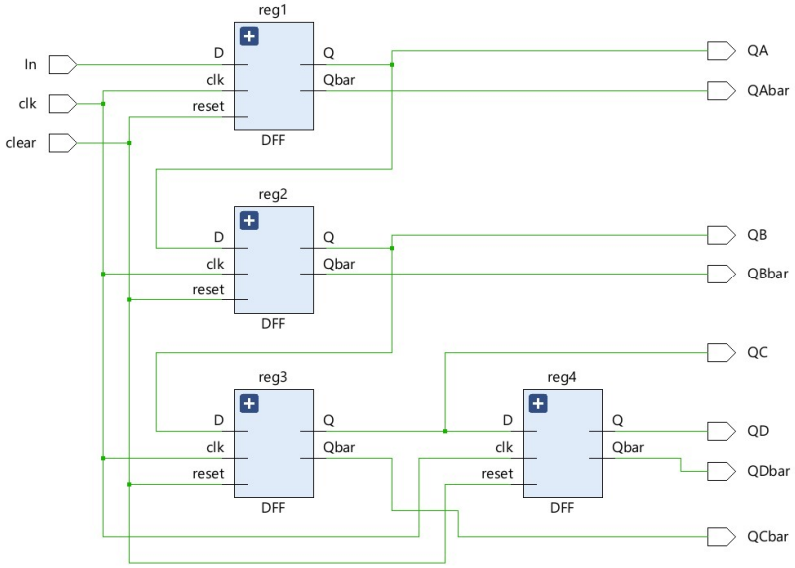
**Question No. 3** Design a 4-bit serial-in-parallel-out shift register and implement on FPGA
**Sol: 4-bit Serial-in parallel-out (SIPO) shift register:**




**Code:**

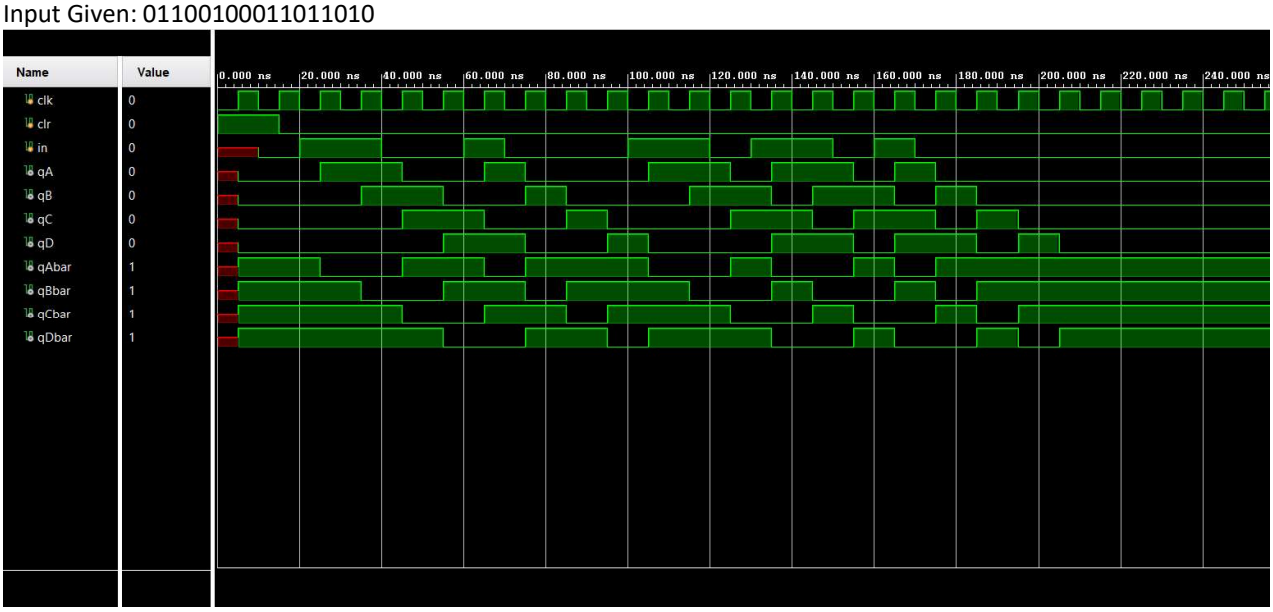
```
module SIPO(clk,clear,In,QA,QB,QC,QD,QAbar,QBbar,QCbar,QDbar);
   input clk,clear,In;
   output QA,QB,QC,QD,QAbar,QBbar,QCbar,QDbar;
   wire qa,qb,qc,qd;

   assign QA = qa, QB = qb, QC = qc, QD = qd;
   DFF reg1(clk,clear,In,qa,QAbar);
   DFF reg2(clk,clear,qa,qb,QBbar);
   DFF reg3(clk,clear,qb,qc,QCbar);
   DFF reg4(clk,clear,qc,qd,QDbar);
endmodule
```

**Schematic:**




**Simulation using TestBench:**

Input Given: 01100100011011010




**Synthesis:**