PES University
DEPARTMENT OF ELECTRONICS COMMUNICATION AND ENGINEERING
(Session: Aug - Dec 2020)

COMPUTER ORGANIZATION AND DIGITAL DESIGN LABORATORY (UE19EC207)

EXPERIMENT-2

AIM: Simulation and FPGA Implementation of Basic logic gates & Universal gates using Xilinx Vivado tool

PROCEDURE:
Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (**Xilinx Artix-7 FPGA: XC7A35T-1CPG236C**)
Step2: Simulate the design using Vivado Simulator (Test Bench Module).
Step3: Synthesize the design and observe the Schematic.
Step4: Implement the design.
Step5: Write Design Constraint (XDC) file to constrain the pin locations.
Step6: Generate the bitstream.
Step7: Configure the FPGA using the generated bitstream and verify the functionality in hardware.

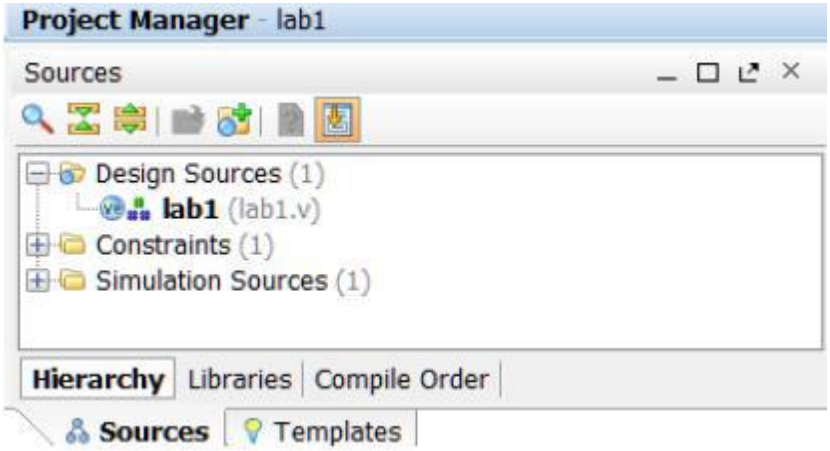
Bit-wise Operators take two single or multiple operands on either side of the operator and return a single bit result. The only exception is the **NOT** operator, which negates the single operand that follows. Verilog does not have the equivalent of **NAND** or **NOR** operator, their funstion is implemented by negating the **AND** and **OR** operators. **DSD LAB**

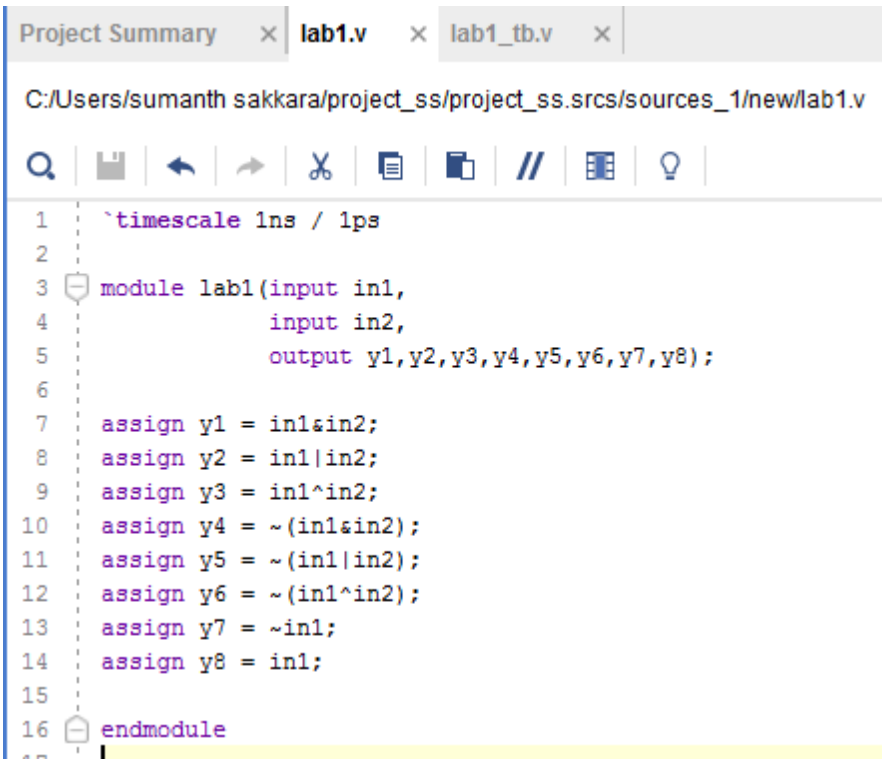
Operator	Name
~	Bitwise negation
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~&	Bitwise NAND
~|	Bitwise NOR
~^ or ^~	Equivalence (Bitwise NOT XOR)

DESIGN SOURCE:

Create a Design Verilog Source file which implements all the gates and set as top module:

Opening the source file:CODD LAB





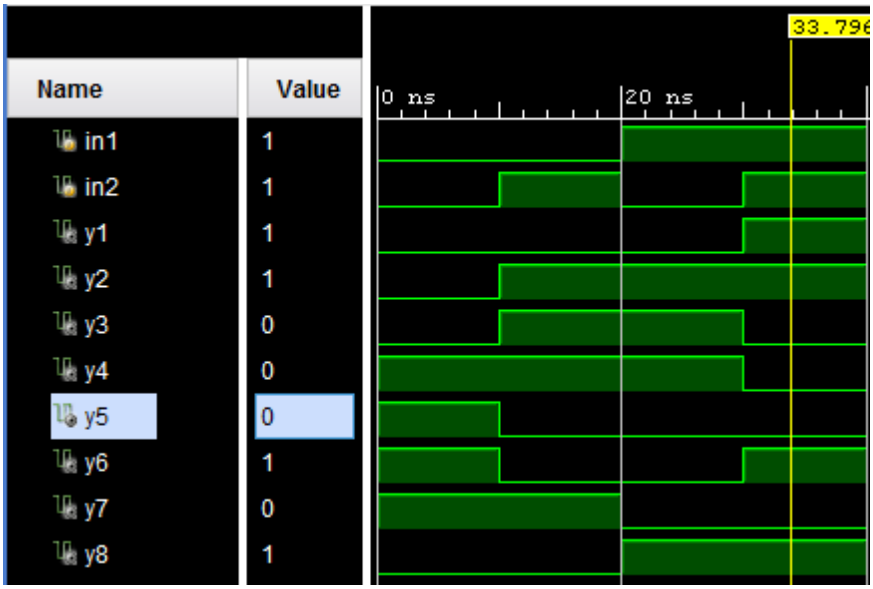

```
`timescale 1ns / 1ps

module lab1(input in1,
            input in2,
            output y1,y2,y3,y4,y5,y6,y7,y8);

assign y1 = in1&in2;
assign y2 = in1|in2;
assign y3 = in1^in2;
assign y4 = ~(in1&in2);
assign y5 = ~(in1|in2);
assign y6 = ~(in1^in2);
assign y7 = ~in1;
assign y8 = in1;

endmodule
```

SIMULATION SOURCE:

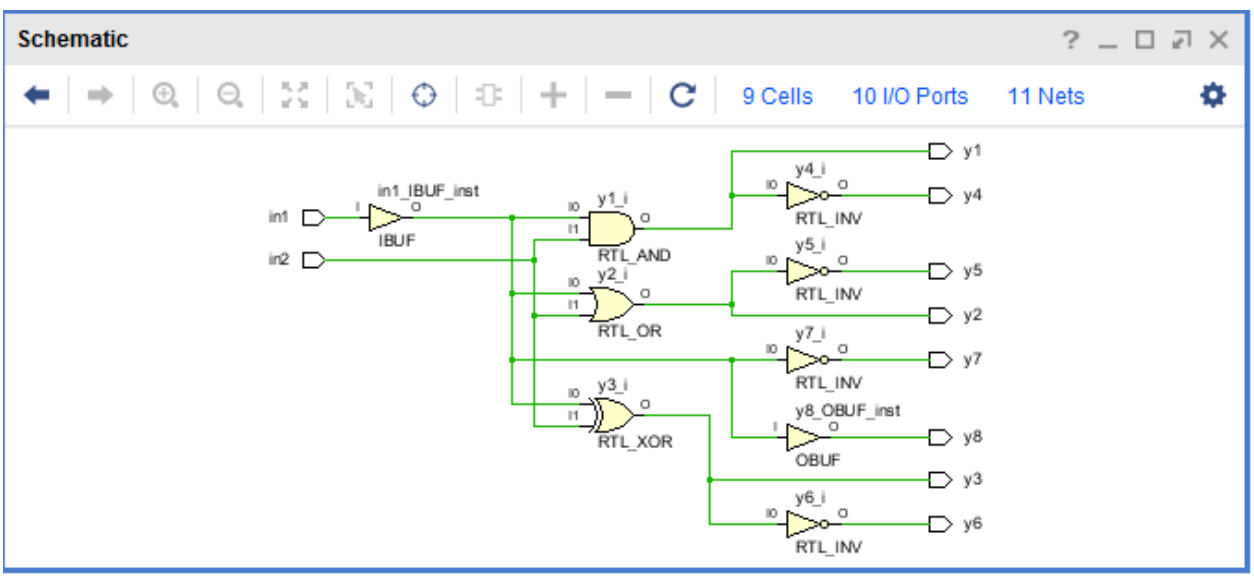
Write a Test Bench Verilog Module and add the test stimulus and verify the waveform **CODD LAB**

```
module lab1_tb ;

        // Inputs
        reg in1;
        reg in2;
        // Outputs
        wire  y1,y2,y3,y4,y5,y6,y7,y8;
        // Instantiate the Unit Under Test (UUT)
        lab1 uut (in1,in2,y1,y2,y3,y4,y5,y6,y7,y8 );
        initial
        begin
         $monitor("%t in1=%b in2=%b y1=%b y2=%b y3=%b y4=%b y5=%b y6=%b y7=%b y8=%b",
         $time,in1,in2,y1,y2,y3,y4,y5,y6,y7,y8);
            // Initialize Inputs
            in1 = 0;
            in2 = 0;
            #10;
            in1=0;
            in2=1;
            #10;
            in1=1;
            in2=0;
            #10;
            in1=1;
            in2=1;
            #10;
            $display("Lab1 TEST done");
            //$finish;
        end

endmodule
```

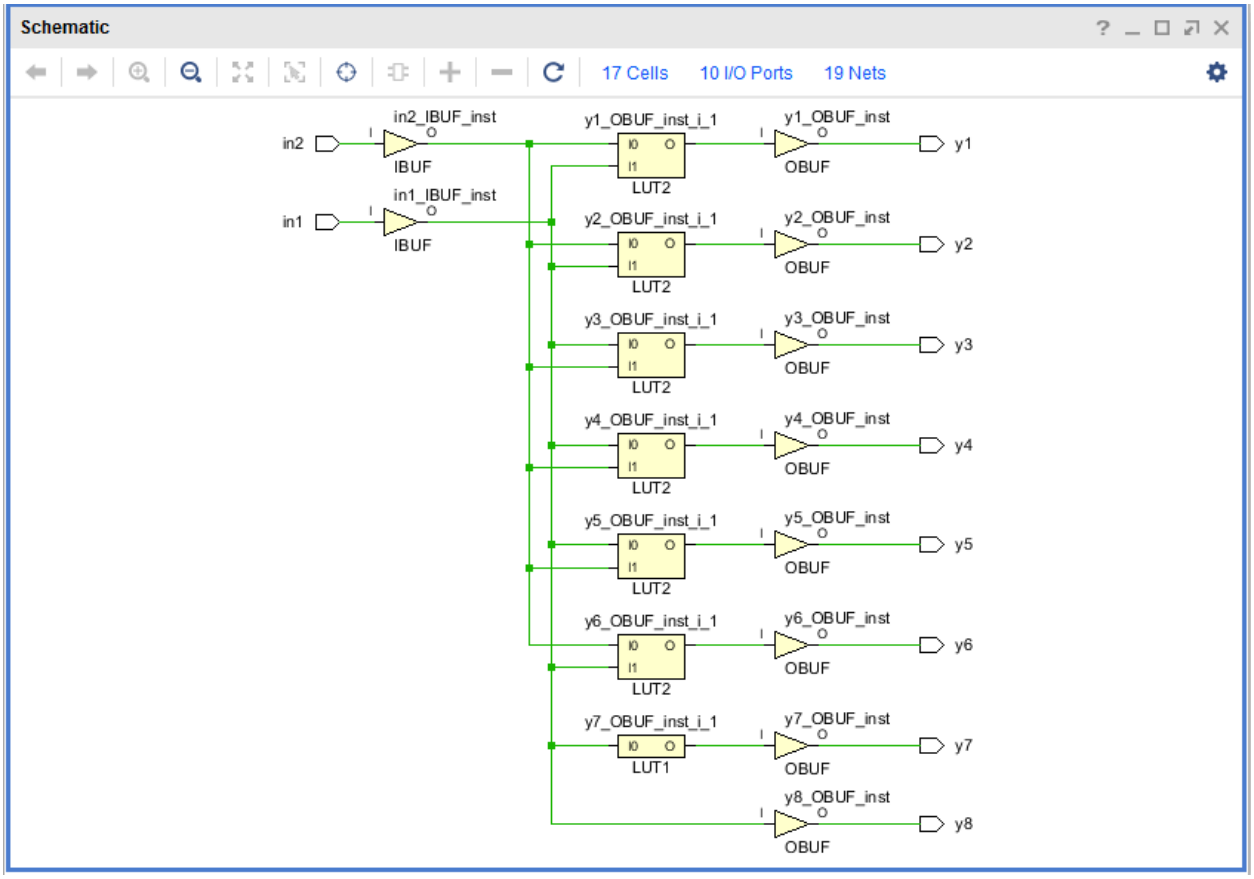
SIMULATION WAVEFORMS: (verify the functional table)
RTL SCHEMATIC: CODD LAB



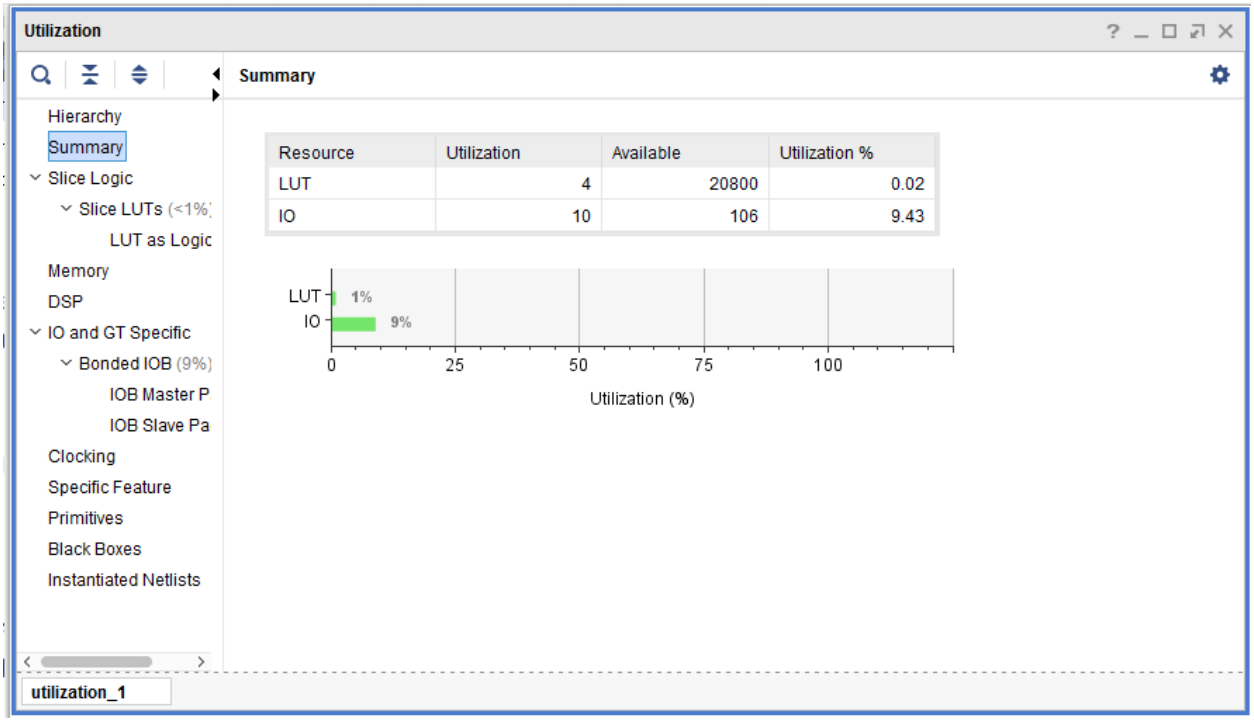
RTL SCHEMATIC:



SYNTHESIS SCHEMATIC: CODD LAB



REPORT UTILIZATION:
XDC CONSTRAINTS SOURCE FILE CODD LAB



XDC CONSTRAINTS SOURCE FILE

ss_cst.xdc

C:/Users/sumanth sakkara/project_ss/project_ss.srcs/constrs_1/new/ss_cst.xdc

```
set_property PACKAGE_PIN R2 [get_ports in1]
set_property PACKAGE_PIN T1 [get_ports in2]
set_property PACKAGE_PIN L1 [get_ports y1]
set_property PACKAGE_PIN P1 [get_ports y2]
set_property PACKAGE_PIN N3 [get_ports y3]
set_property PACKAGE_PIN P3 [get_ports y4]
set_property PACKAGE_PIN W3 [get_ports y6]
set_property PACKAGE_PIN U3 [get_ports y5]
set_property PACKAGE_PIN V3 [get_ports y7]
set_property PACKAGE_PIN V13 [get_ports y8]
set_property IOSTANDARD LVCMOS33 [get_ports y7]
set_property IOSTANDARD LVCMOS33 [get_ports y8]
set_property IOSTANDARD LVCMOS33 [get_ports y6]
set_property IOSTANDARD LVCMOS33 [get_ports y5]
set_property IOSTANDARD LVCMOS33 [get_ports y4]
set_property IOSTANDARD LVCMOS33 [get_ports y3]
set_property IOSTANDARD LVCMOS33 [get_ports y2]
set_property IOSTANDARD LVCMOS33 [get_ports y1]
set_property IOSTANDARD LVCMOS33 [get_ports in2]
set_property IOSTANDARD LVCMOS33 [get_ports in1]
```

Answer the following:

Define synthesis

Define Simulation

Abbreviate FPGA.

EXPERIMENT – 3

AIM:

Simulation and FPGA Implementation of full adder using structural, dataflow and behavioural modelling in Xilinx Vivado tool

PROCEDURE:

Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (Xilinx Artix-7 FPGA: XC7A35T-1CPG236C)

Step2: Simulate the design using Vivado Simulator (Test Bench Module).

Step3: Synthesize the design and observe the Schematic.

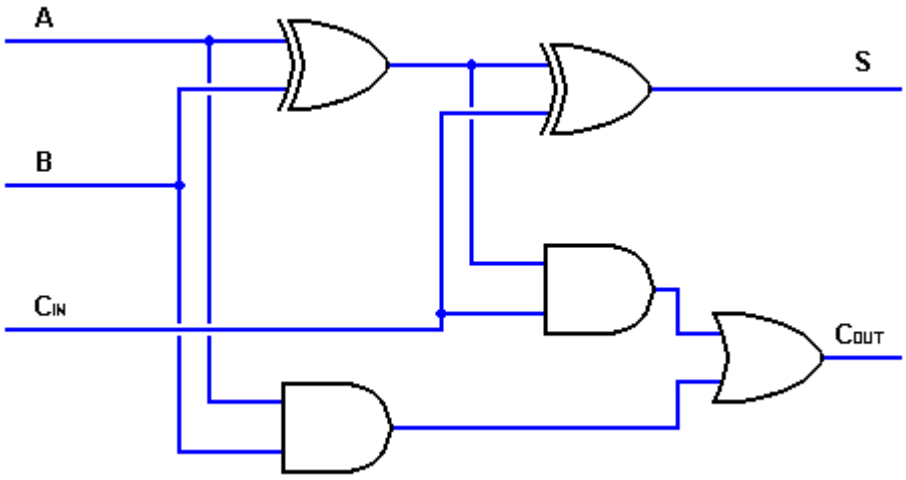
Step4: Implement the design.

Step5: Write Design Constraint (XDC) file to constrain the pin locations.

Step6: Generate the bitstream.

Step7: Configure the FPGA using the generated bitstream and verify the functionality in hardware.

FULL ADDER:



TRUTH TABLE:

A	B	Cin	SUM (S)	CARRY (Cout)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

BOOLEAN EXPRESSION:

$$S = A'B'Cin + A'BCin' + AB'Cin' + ABCin$$

$$= Cin\,(A'B' + AB) + Cin'\,(A'B + AB')$$

$$= Cin\,(A'B + AB')' + Cin'\,(A'B + AB')$$

$$\boxed{S = A \oplus B \oplus Cin}$$

$$\boxed{Cout = AB + ACin + BCin}$$

FULL ADDER DESIGN USING STRUCTURAL MODELLING:

Project Summary | full_adder_structural_design.v

:/dsplab/project_1_full_adder/project_1_full_adder.srcs/sources_1/new/full_ad

```
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////


module full_adder_structural_design(
    input A,
    input B,
    input Cin,
    output S,
    output Cout
    );
 wire a1, a2, a3;
       xor u1(a1,A,B);
       and u2(a2,A,B);
       and u3(a3,a1,Cin);
       or u4(Cout,a2,a3);
       xor u5(S,a1,Cin);
endmodule

```

FULL ADDER DESIGN USING BEHAVIOURAL MODELLING:

Project Summary | full_adder_behavioural_design.v

er_behavioural/project_1_full_adder_behavioural.srcs/sources_1/new/full_adder_behavioural_design.v

```
module full_adder_behavioural_design (input wire A, B, Cin,
 output reg S,
 output reg Cout);

 always @(A or B or Cin)
  begin

   case ({A,B,Cin})
     3'b000: begin S=0; Cout=0; end
     3'b001: begin S=1; Cout=0; end
     3'b010: begin S=1; Cout=0; end
     3'b011: begin S=0; Cout=1; end
     3'b100: begin S=1; Cout=0; end
     3'b101: begin S=0; Cout=1; end
     3'b110: begin S=0; Cout=1; end
     3'b111: begin S=1; Cout=1; end
   endcase

  end

endmodule
```

FULL ADDER DESIGN USING DATAFLOW MODELLING:

Project Summary | full_adder_dataflow_design.v

me/dsplab/full_adder_dataflow/full_adder_dataflow.srcs/sources_1/new/full_adder_dataflow_design.v

```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////


module full_adder_dataflow_design(
    input a,
    input b,
    input cin,
    output sum,
    output carry
    );
   assign x=a ^ b;
   assign sum=x^cin;
   assign y=x & cin;
   assign z=a & b;
   assign carry= y | z;
endmodule
```

TESTBENCH CODE FOR FULL ADDER:

Project Summary | full_adder_behavioural_design.v | full_adder_tb.v *

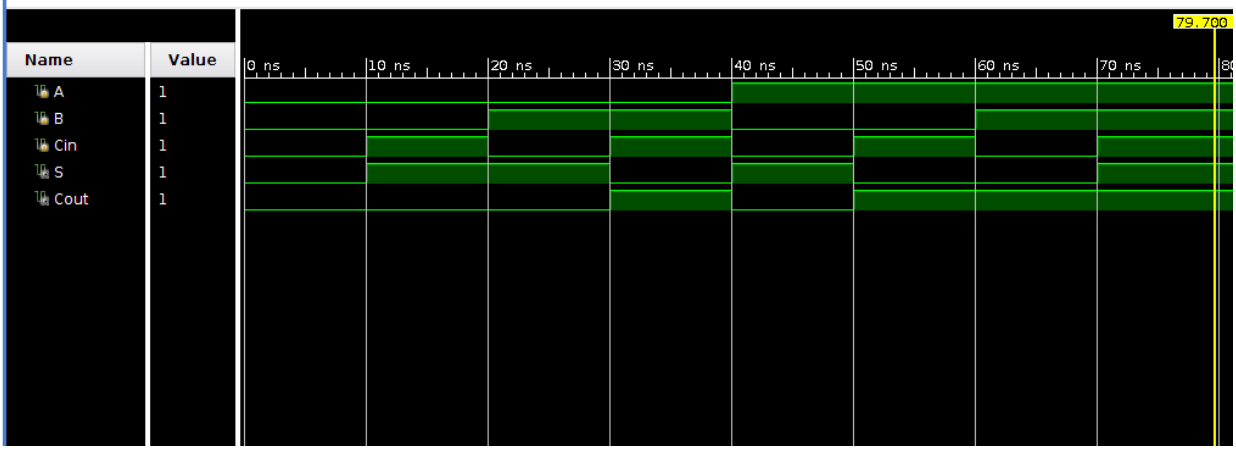
/home/dsplab/project_1_full_adder_behavioural/project_1_full_adder_behavioural.srcs/sim_1/new/full_adder_tb.v

```
module full_adder_tb( );
 reg A,B,Cin;
 wire S,Cout;

 full_adder_behavioural_design uut(A,B,Cin,S,Cout);
 initial begin
   A = 0;
   B = 0;
   Cin = 0;
   #10;
   A = 0;
   B = 0;
   Cin = 1;
   #10;
   A = 0;
   B = 1;
   Cin = 0;
   #10;
   A = 0;
   B = 1;
   Cin = 1;
   #10;
   A = 1;
   B = 0;
   Cin = 0;
   #10;
   A = 1;
   B = 0;
   Cin = 1;
   #10;
   A = 1;
   B = 1;
   Cin = 0;
   #10;
   A = 1;
   B = 1;
   Cin = 1;
   #10;
  end
endmodule
```

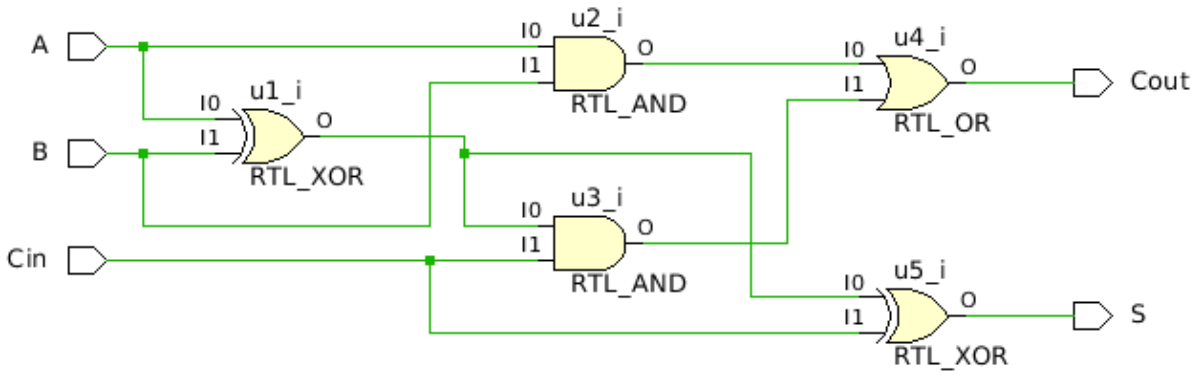
SIMULATION RESULT:

SIMULATION - Behavioral Simulation - Functional - sim_1 - full_adder_structural_tb

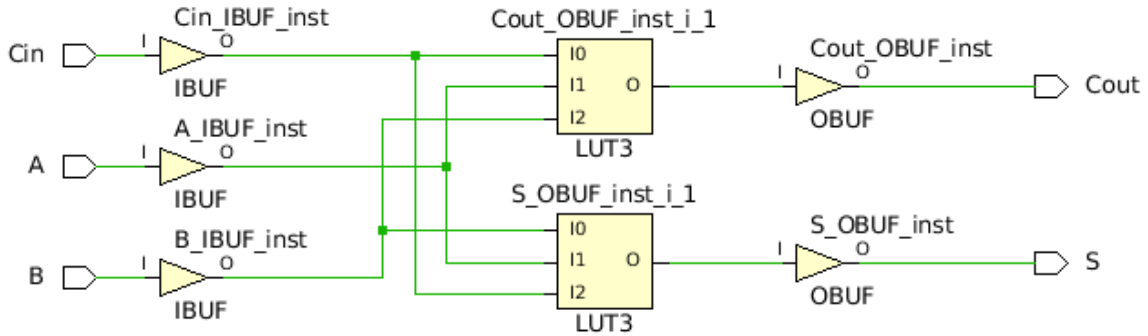
full_adder_structural_design.v | full_adder_structural_tb.v | Untitled 4



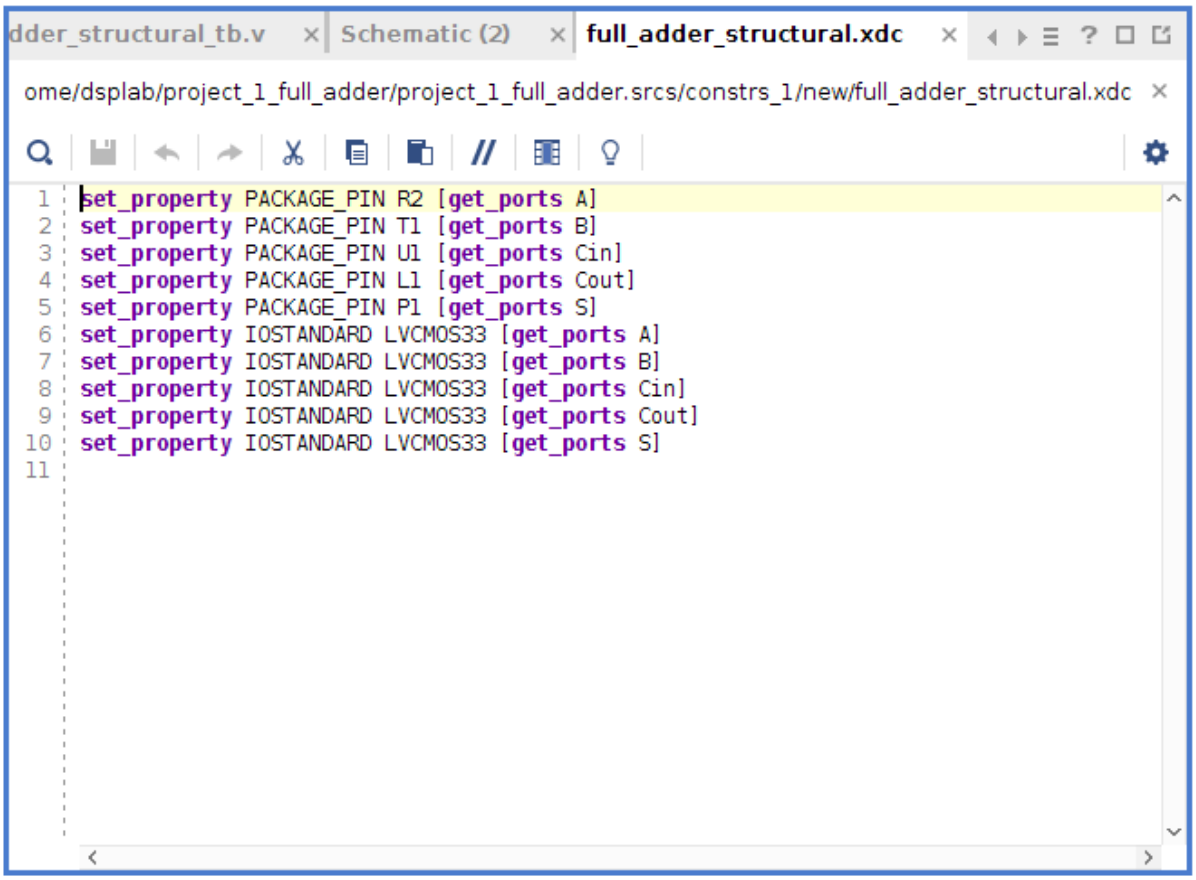
RTL SCHEMATIC:



POST SYNTHESIS SCHEMATIC



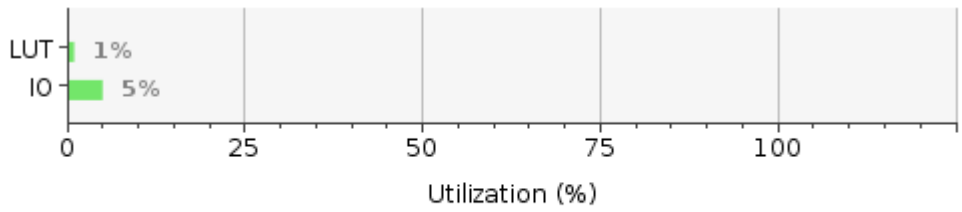
XDC FILE:




```
set_property PACKAGE_PIN R2 [get_ports A]
set_property PACKAGE_PIN T1 [get_ports B]
set_property PACKAGE_PIN U1 [get_ports Cin]
set_property PACKAGE_PIN L1 [get_ports Cout]
set_property PACKAGE_PIN P1 [get_ports S]
set_property IOSTANDARD LVCMOS33 [get_ports A]
set_property IOSTANDARD LVCMOS33 [get_ports B]
set_property IOSTANDARD LVCMOS33 [get_ports Cin]
set_property IOSTANDARD LVCMOS33 [get_ports Cout]
set_property IOSTANDARD LVCMOS33 [get_ports S]
```

UTILIZATION REPORT:

Resource	Utilization	Available	Utilization %
LUT	1	20800	0.00
IO	5	106	4.72



EXPERIMENT – 4

AIM:

Simulation and FPGA Implementation of 1- bit and 4- bit comparator in Xilinx Vivado tool

PROCEDURE:

Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (Xilinx Artix-7 FPGA: XC7A35T-1CPG236C)

Step2: Simulate the design using Vivado Simulator (Test Bench Module).

Step3: Synthesize the design and observe the Schematic.

Step4: Implement the design.

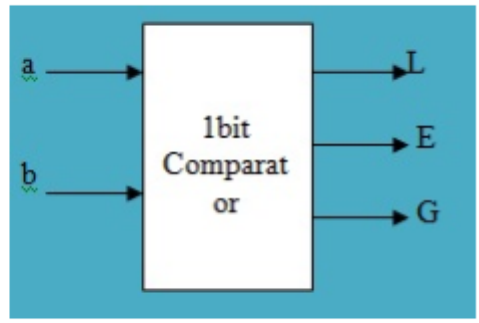
Step5: Write Design Constraint (XDC) file to constrain the pin locations.

Step6: Generate the bitstream.

Step7: Configure the FPGA using the generated bitstream and verify the functionality in hardware.

a) 1 BIT COMPARATOR:

a	b	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0



1- BIT COMPARATOR DESIGN USING DATAFLOW MODELLING:

Project Summary | comparator_1bit_dataflow_design.v

comparator_1bit/project_1_comparator_1bit.srcs/sources_1/new/comparator_1bit_dataflow_design.v

```
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////


module comparator_1bit_dataflow_design(
    input a,
    input b,
    output L,
    output E,
    output G
    );

    assign E= a ~^ b;
    assign L= (~a) & b;
    assign G = a & (~b);
endmodule
```

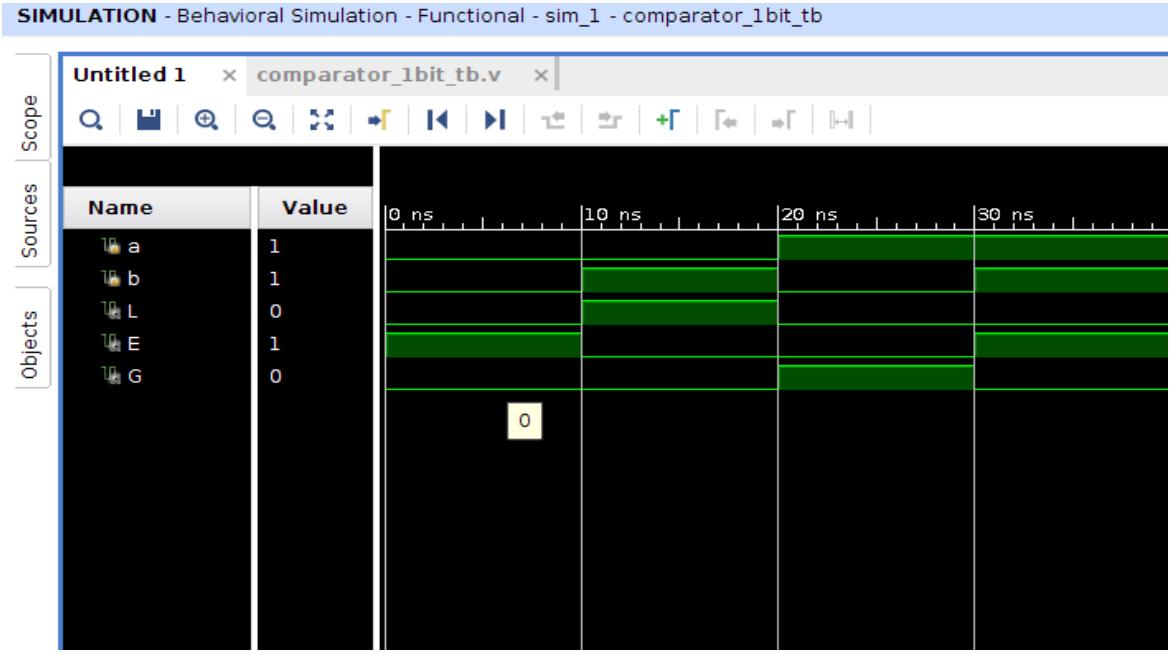
TESTBENCH FOR 1- BIT COMPARATOR:

Project Summary | comparator_1bit_tb.v

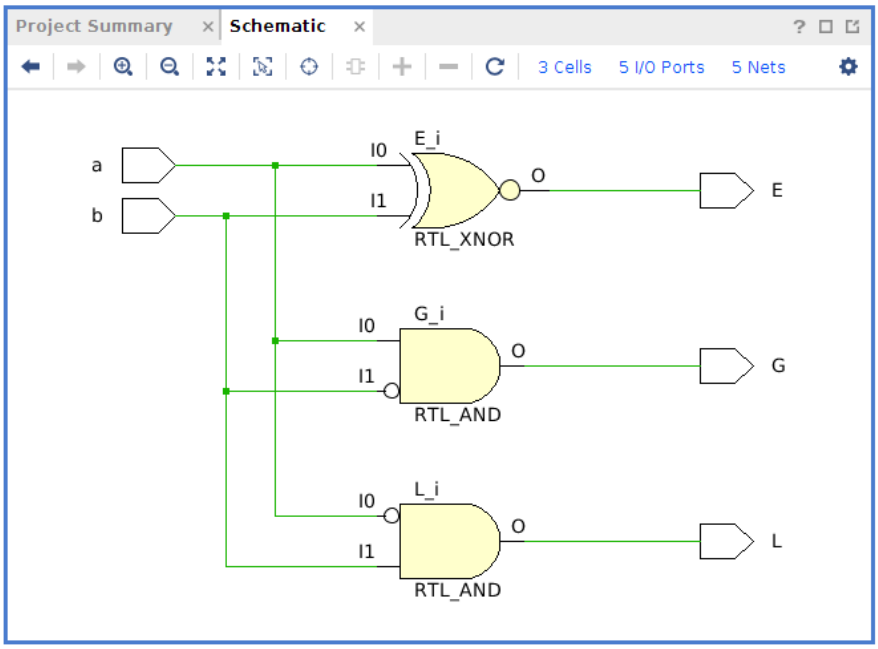
/dsplab/project_1_comparator_1bit/project_1_comparator_1bit.srcs/sim_1/new/comparator_1bit_tb.v

```
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////


module comparator_1bit_tb();
reg a,b;
wire L,E,G;
comparator_1bit_dataflow_design  uut(a,b,L,E,G) ;
 initial
 begin
            a=1'b0;
            b=1'b0;

     #10    a=1'b0;
            b=1'b1;
     #10    a=1'b1;
            b=1'b0;
     #10    a=1'b1;
            b=1'b1;
      #40   $finish;
 end
endmodule
```

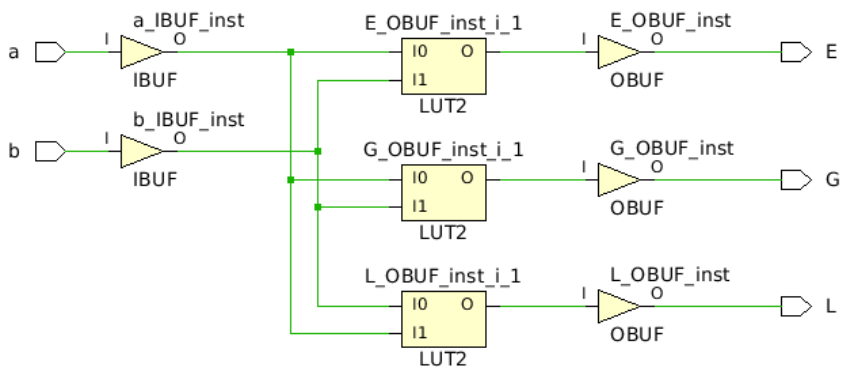
SIMULATION RESULT:



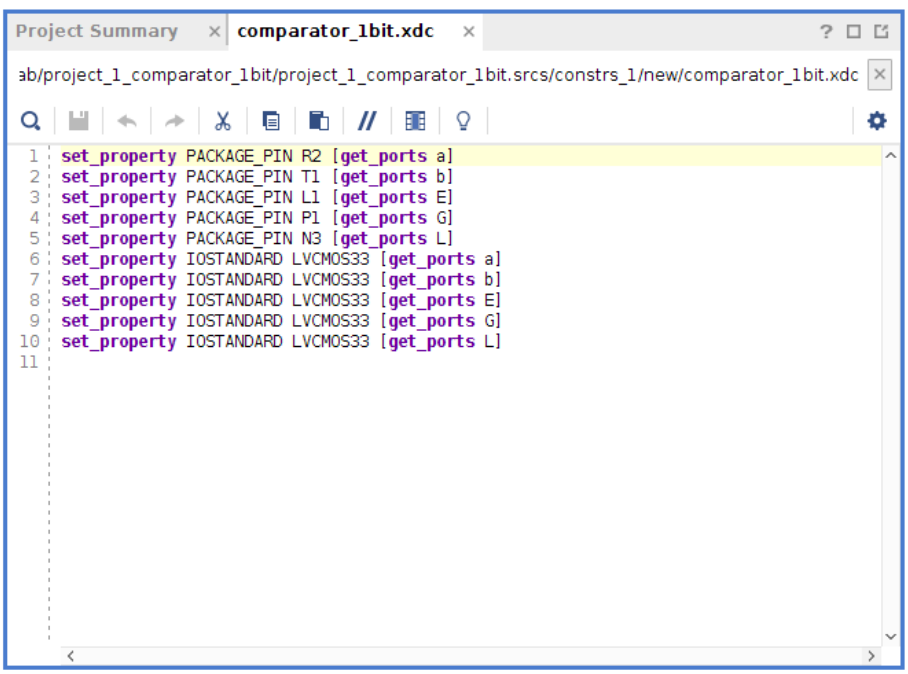
RTL SCHEMATIC:



POST SYNTHESIS SCHEMATIC:



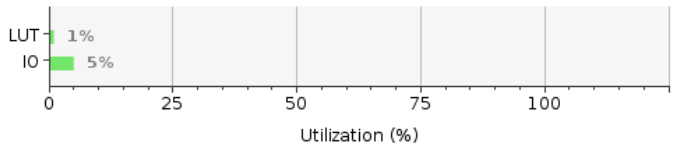
XDC FILE:




```
set_property PACKAGE_PIN R2 [get_ports a]
set_property PACKAGE_PIN T1 [get_ports b]
set_property PACKAGE_PIN L1 [get_ports E]
set_property PACKAGE_PIN P1 [get_ports G]
set_property PACKAGE_PIN N3 [get_ports L]
set_property IOSTANDARD LVCMOS33 [get_ports a]
set_property IOSTANDARD LVCMOS33 [get_ports b]
set_property IOSTANDARD LVCMOS33 [get_ports E]
set_property IOSTANDARD LVCMOS33 [get_ports G]
set_property IOSTANDARD LVCMOS33 [get_ports L]
```

UTILIZATION REPORT

Resource	Utilization	Available	Utilization %
LUT	2	20800	0.01
IO	5	106	4.72



b) 4 BIT COMPARATOR

4-BIT COMPARATOR DESIGN USING BEHAVIOURAL MODELLING

Project Summary | comparator_4bit.v

/home/dsplab/project_1_comparatot_4bit/project_1_comparatot_4bit.srcs/sources_1/new/comparator_4bit.v

```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////


module comparator_4bit(
    input [3:0] a,
    input [3:0] b,
    output reg eq,
    output reg lt,
    output reg gt
    );
    always @(a,b)
    begin
     if (a==b)
     begin
      eq = 1'b1;
      lt = 1'b0;
      gt = 1'b0;
     end
     else if (a>b)
     begin
      eq = 1'b0;
      lt = 1'b0;
      gt = 1'b1;
     end
     else
     begin
      eq = 1'b0;
      lt = 1'b1;
      gt = 1'b0;
     end
    end

endmodule
```

TESTBENCH FOR 4-BIT COMPARATOR:

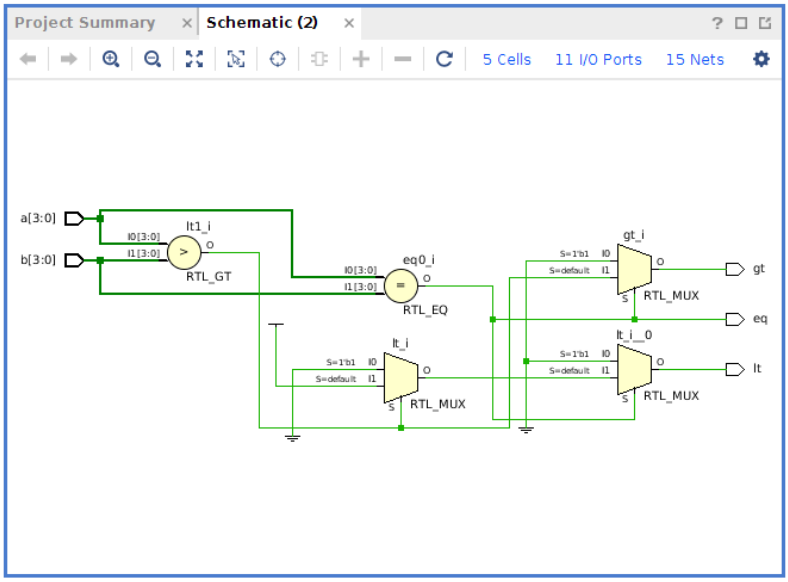
Project Summary | comparator_4bit.v | comparator_4bit_tb.v

/home/dsplab/project_1_comparatot_4bit/project_1_comparatot_4bit.srcs/sim_1/new/comparator_4bit_tb.v

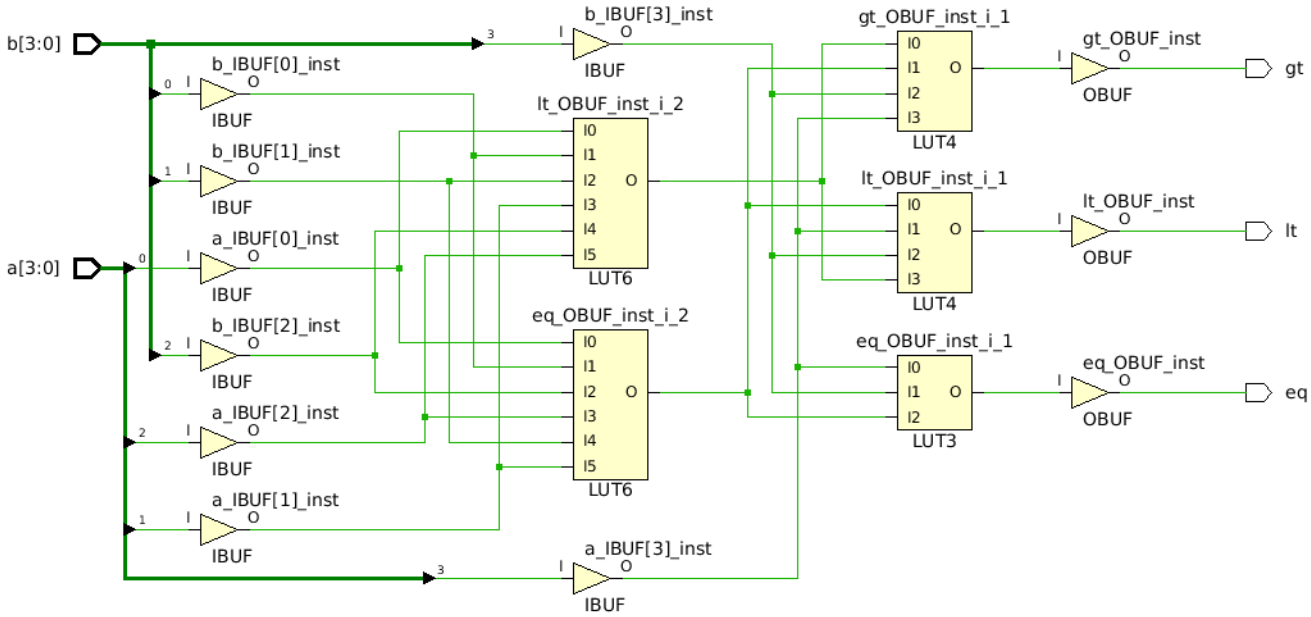
```
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module comparator_4bit_tb(

    );
    reg [3:0] a,b;
      wire eq,lt,gt;

      comparator_4bit DUT (a,b,eq,lt,gt);

      initial
      begin
       a = 4'b1100;
       b = 4'b1100;
       #10;

       a = 4'b0100;
       b = 4'b1100;
       #10;

       a = 4'b1111;
       b = 4'b1100;
       #10;

       a = 4'b0000;
       b = 4'b0000;
       #10;
       $stop;
      end
endmodule
```

SIMULATION RESULT:

RTL SCHEMATIC:



POST SYNTHESIS SCHEMATIC:



XDC FILE:

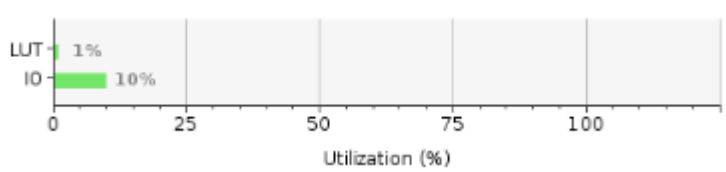
Schematic (2) | comparator_4bit_tb.v | comparator_4bit.xdc

ab/project_1_comparatot_4bit/project_1_comparatot_4bit.srcs/constrs_1/new/comparator_4bit.xdc

```
set_property PACKAGE_PIN R2 [get_ports {a[3]}]
set_property PACKAGE_PIN T1 [get_ports {a[2]}]
set_property PACKAGE_PIN U1 [get_ports {a[1]}]
set_property PACKAGE_PIN W2 [get_ports {a[0]}]
set_property PACKAGE_PIN R3 [get_ports {b[3]}]
set_property PACKAGE_PIN T2 [get_ports {b[2]}]
set_property PACKAGE_PIN T3 [get_ports {b[1]}]
set_property PACKAGE_PIN V2 [get_ports {b[0]}]
set_property PACKAGE_PIN L1 [get_ports eq]
set_property PACKAGE_PIN P1 [get_ports gt]
set_property PACKAGE_PIN N3 [get_ports lt]
set_property IOSTANDARD LVCMOS33 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports eq]
set_property IOSTANDARD LVCMOS33 [get_ports gt]
set_property IOSTANDARD LVCMOS33 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports lt]
```

UTILIZATION REPORT

Resource	Utilization	Available	Utilization %
LUT	4	20800	0.02
IO	11	106	10.38



EXPERIMENT – 5

AIM:

Simulation and FPGA Implementation of 4:1 Multiplexer and 1:4 De-Multiplexer using Prameter construct for variable data bus width in Xilinx Vivado tool.

PROCEDURE:

Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (Xilinx Artix-7 FPGA: XC7A35T-1CPG236C)

Step2: Simulate the design using Vivado Simulator (Test Bench Module).

Step3: Synthesize the design and observe the Schematic.

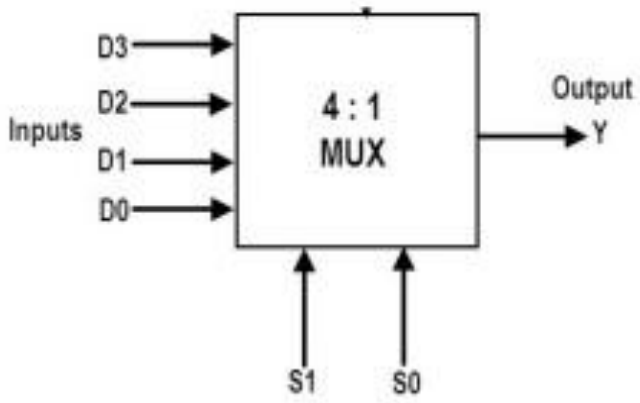
Step4: Implement the design.

Step5: Write Design Constraint (XDC) file to constrain the pin locations.

Step6: Generate the bitstream.

Step7: Configure the FPGA using the generated bitstream and verify the functionality in hardware.

4:1 MULTIPLEXER:



A 4-to-1 multiplexer consists four data input lines as D0 to D3, two select lines as S0 and S1 and a single output line Y. The select lines S1 and S2 select one of the four input lines to connect the output line. The particular input combination on select lines selects one of input (D0 through D3) to the output.

Truth Table:

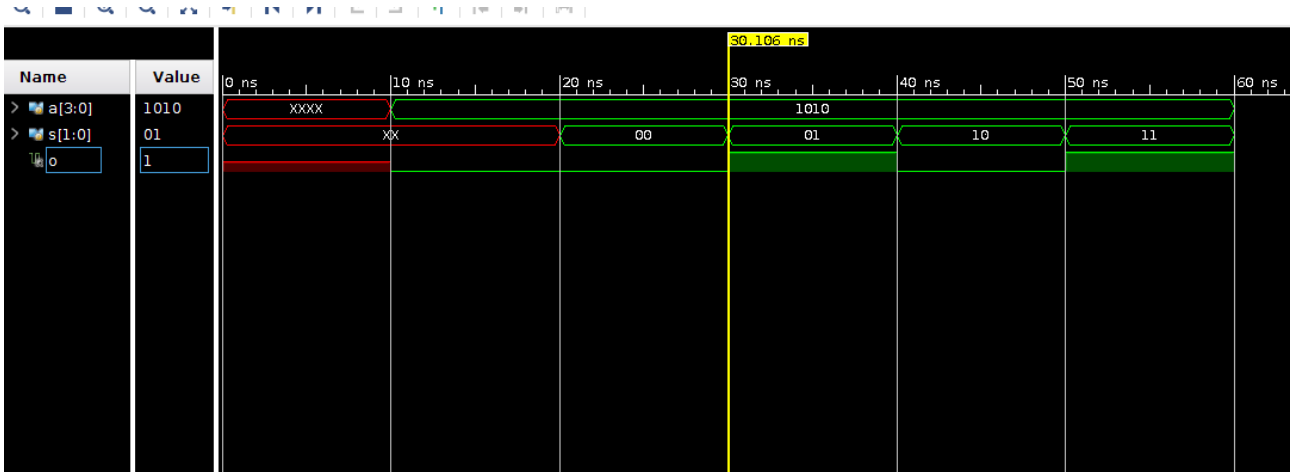
Select Data Inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

```
module mux4bit(a, s, o);

input [3:0] a;

input [1:0] s;

output o;

reg o;

always @(a or s)

begin

case (s)

2'b00:o=a[0];

2'b01:o=a[1];

2'b10:o=a[2];

2'b11:o=a[3];

default:o=0;

endcase

end

endmodule
```

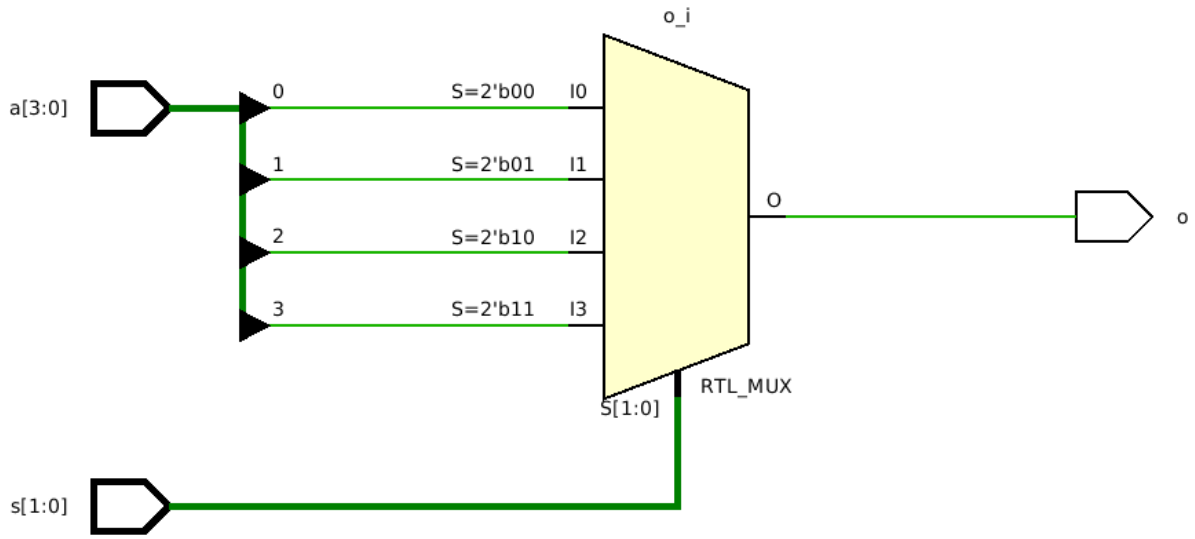
TESTBENCH FOR 4:1 MUX:

```
////////////////////////////////////////////////////////////////////////////////

module muxt_b;

reg [3:0] a;

reg [1:0] s;

wire o;

mux4bit uut (.a(a),   .s(s),.o(o));

initial begin

#10 a=4'b1010;

#10 s=2'b00;

#10 s=2'b01;

#10 s=2'b10;

#10 s=2'b11;

#10 $stop;

end

endmodule
```

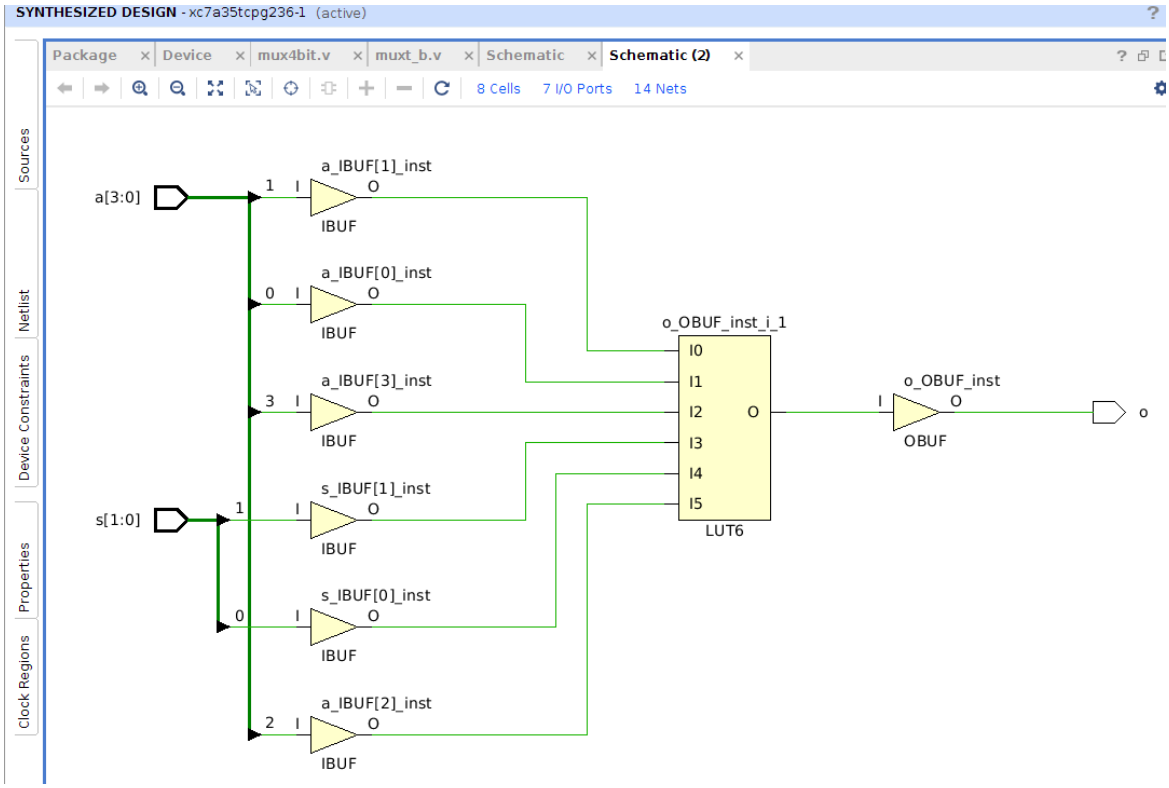
SIMULATION RESULT:



RTL SCHEMATIC:



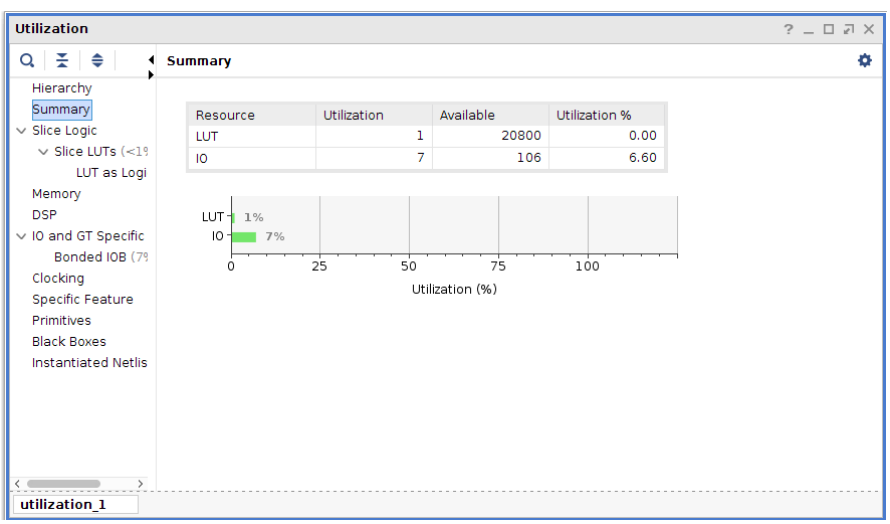
POST SYNTHESIS SCHEMATIC:



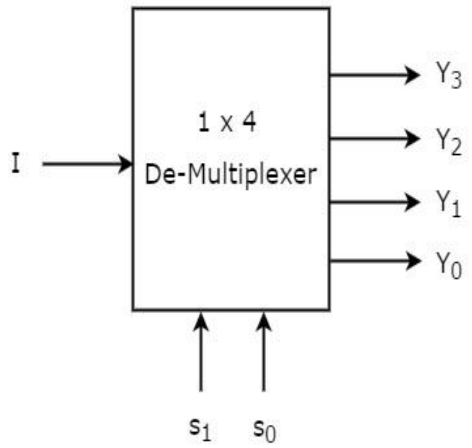
XDC FILE:

```
set_property IOSTANDARD LVCMOS33 [get_ports {a[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[0]}]
set_property PACKAGE_PIN R2 [get_ports {a[0]}]
set_property PACKAGE_PIN T1 [get_ports {a[1]}]
set_property PACKAGE_PIN U1 [get_ports {a[2]}]
set_property PACKAGE_PIN W2 [get_ports {a[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {s[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {s[0]}]
set_property PACKAGE_PIN V16 [get_ports {s[0]}]
set_property PACKAGE_PIN V17 [get_ports {s[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports o]
set_property PACKAGE_PIN L1 [get_ports o]

set_property IOSTANDARD LVCMOS33 [get_ports {d[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {d[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {d[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {d[3]}]
set_property PACKAGE_PIN L1 [get_ports {d[0]}]
set_property PACKAGE_PIN P1 [get_ports {d[1]}]
set_property PACKAGE_PIN N3 [get_ports {d[2]}]
set_property PACKAGE_PIN P3 [get_ports {d[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports a]
set_property PACKAGE_PIN R2 [get_ports a]
```

UTILIZATION REPORT:



1:4 DEMUX



A Demultiplexer is a data distributor read as DEMUX. It is quite opposite to multiplexer or MUX. It is a process of taking information from one input and transmitting over one of many outputs.

The 1 to 4 demultiplexer consists of one input, four outputs, and two control lines to make selections.

TRUTH TABLE:

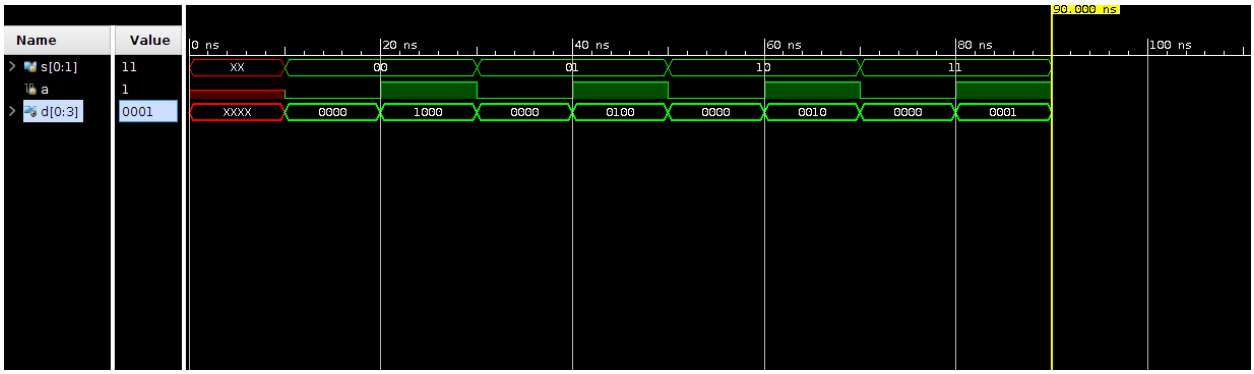
Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

```
module demux(s,a,d);
input [1:0] s;
input a;
output [3:0] d;
reg [3:0] d;
always @(a or s)
begin
if(a==1)
begin
case (s)
2'b00:d=4'b1000;
2'b01:d=4'b0100;
2'b10:d=4'b0010;
2'b11:d=4'b0001;
default:d=0;
endcase
end else if(a==0)
d=0;
end
```

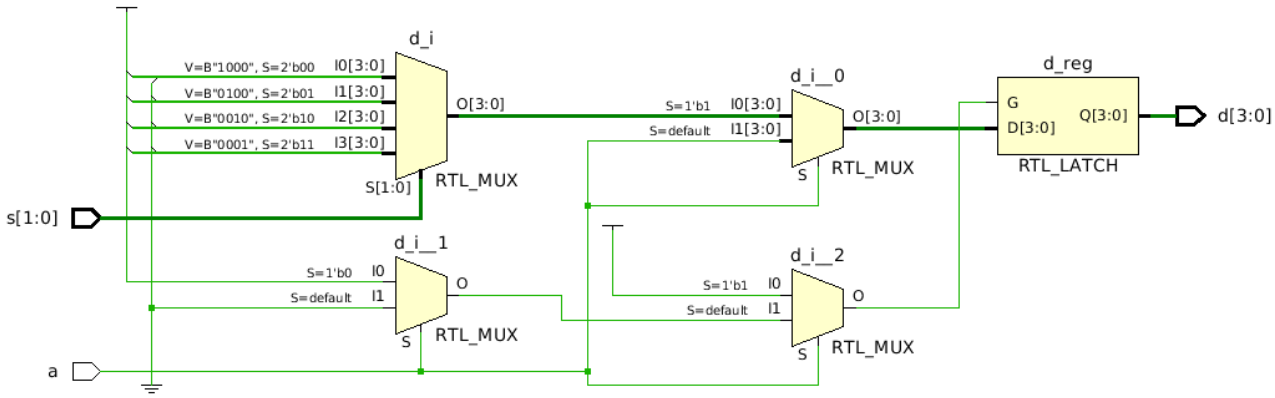
TESTBENCH FOR 1:4 DEMUX:

```
module demuxt_b;

reg [0:1] s;

reg a;

wire [0:3] d;

demux uut (.s(s),.a(a),.d(d)  );

initial begin

#10 s=2'b00;a=1'b0;

#10 s=2'b00;a=1'b1;

#10 s=2'b01;a=1'b0;

#10 s=2'b01;a=1'b1;

#10 s=2'b10;a=1'b0;

#10 s=2'b10;a=1'b1;

#10 s=2'b11;a=1'b0;

#10 s=2'b11;a=1'b1;

#10 $stop;

end

endmodule
```

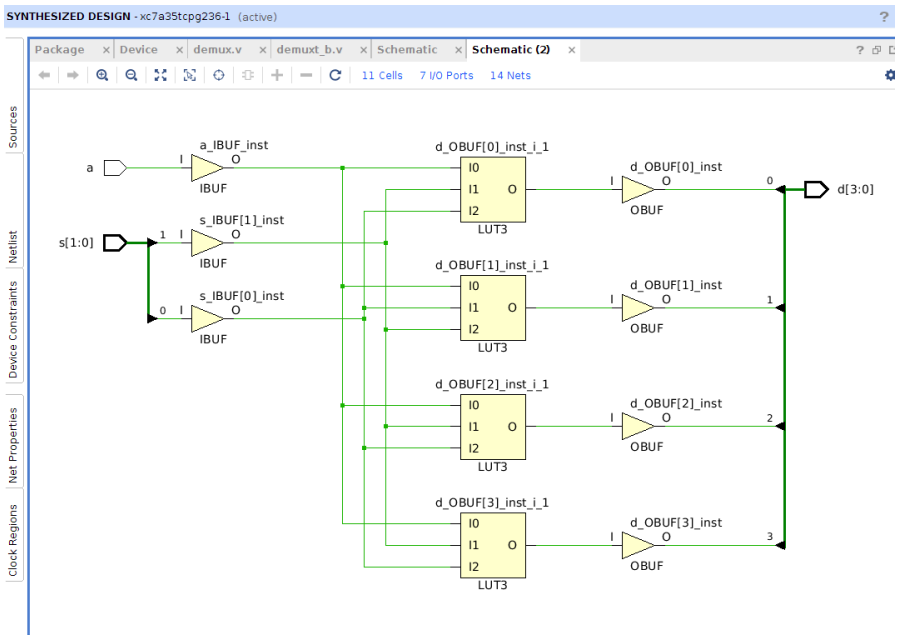
SIMULATION RESULT:



RTL SCHEMATIC:



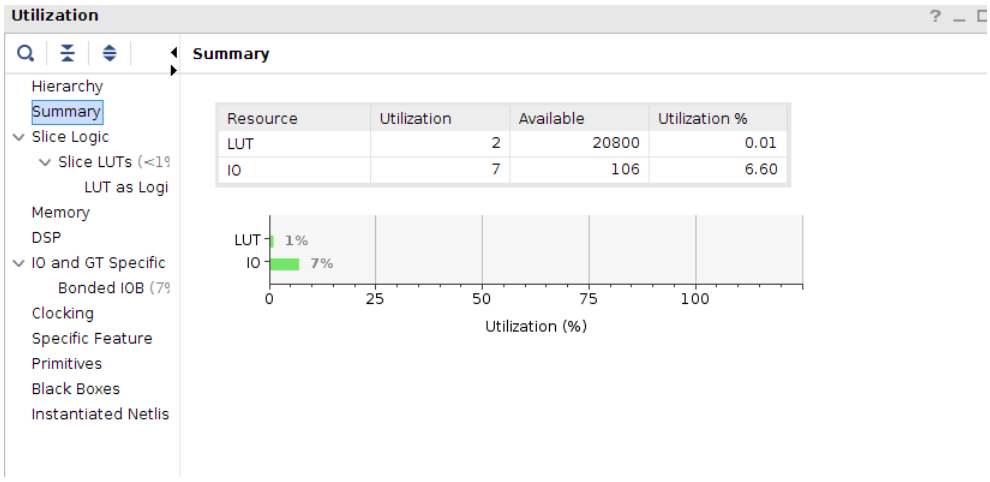
POST SYNTHESIS SCHEMATIC:



XDC FILE:

Similar to 4:1 MUX, observe the XDC file for 1:4 DEMUX

UTILIZATION REPORT:



EXPERIMENT-6

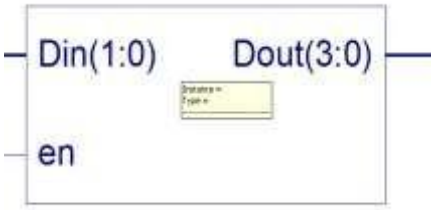
AIM: Simulation and FPGA Implementation of ENCODERS AND DECODERS using Xilinx Vivado tool

PROCEDURE:

Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (**Xilinx Artix-7 FPGA: XC7A35T-1CPG236C**)
Step2: Simulate the design using Vivado Simulator (Test Bench Module).
Step3: Synthesize the design and observe the Schematic.
Step4: Implement the design.
Step5: Write Design Constraint (XDC) file to constrain the pin locations.
Step6: Generate the bitstream.

Step7: Configure the FPGA using the generated bitstream and verify the functionality in hardware.

1. 2 TO 4 DECODER



RTL SCHEMATIC

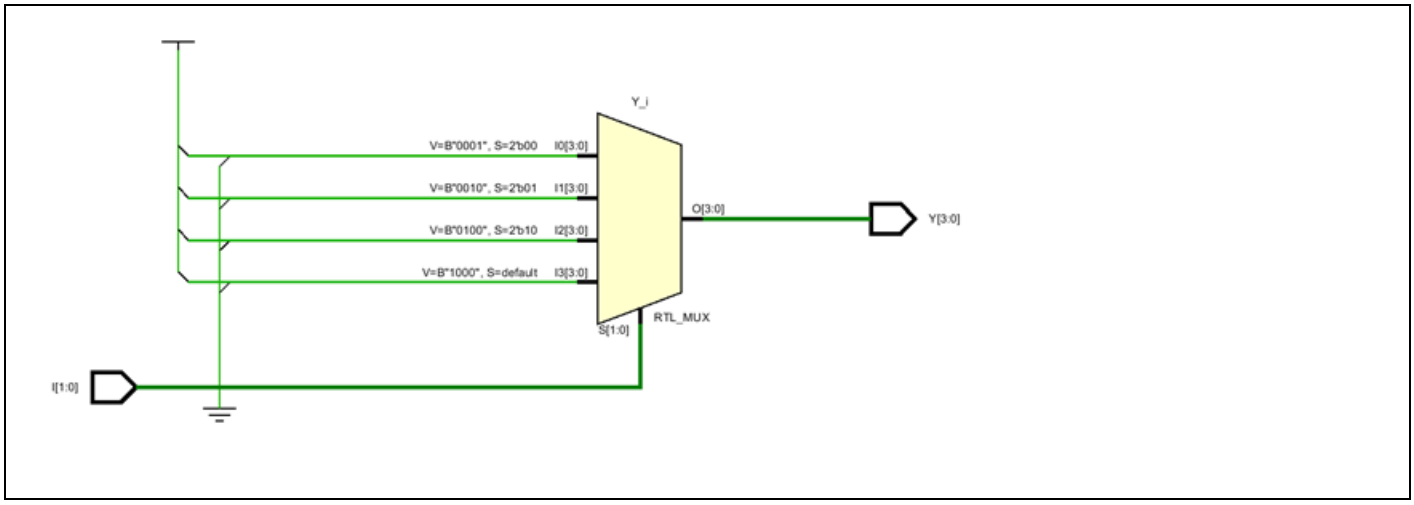
Truth Table

EN	Din(1)	Din(0)	Dout(3)	Dout(2)	Dout(1)	Dout(0)
1	x	x	0	0	0	0
0	0	0	0	0	0	1
0	0	1	0	0	1	0
0	1	0	0	1	0	0
0	1	1	1	0	0	0

VERILOG CODE

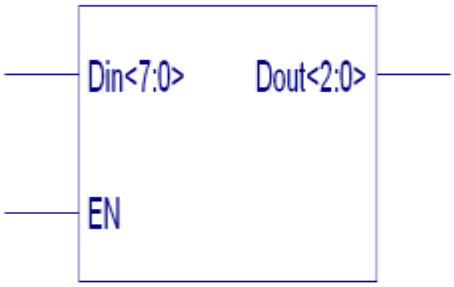
```
module dec2_4 (en,Din,Dout);
input  en;
input [ 1 : 0 ] Din;
output [ 3 : 0 ] Dout;
reg [ 3 : 0 ] Dout;
always@(en,Din)

Begin

if(en == 1)          //Active high enable
Begin
Dout = 4'b0000;     // Initializing Dout to 0000
End

Else
Begin
case (Din)
2'b00:Dout = 4'b0001;
2'b01:Dout = 4'b0010;
2'b10:Dout = 4'b0100;
2'b11:Dout = 4'b1000;
Endcase
End
End
Endmodule
```

RTL schematic



SIMULATION WAVEFORM

2. 8 TO 3 ENCODER WITHOUT PRIORITY



RTL Schematic

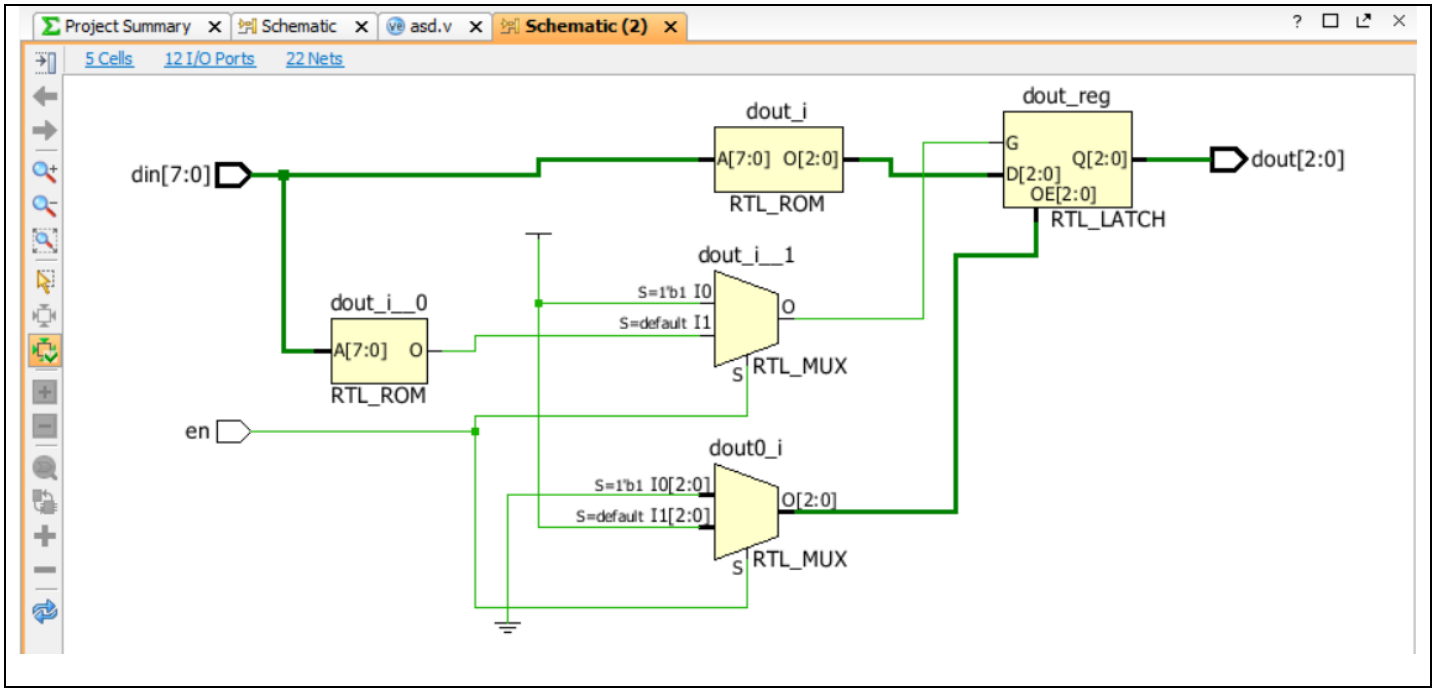
Truth Table

INPUTS									OUTPUTS		
en	**Din(0)**	**Din(1)**	**Din(2)**	**Din(3)**	**Din(4)**	**Din(5)**	**Din(6)**	**Din(7)**	**Dout(0)**	**Dout(1)**	**Dout(3)**
1	x	x	x	x	x	X	x	X	z	z	z
0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0

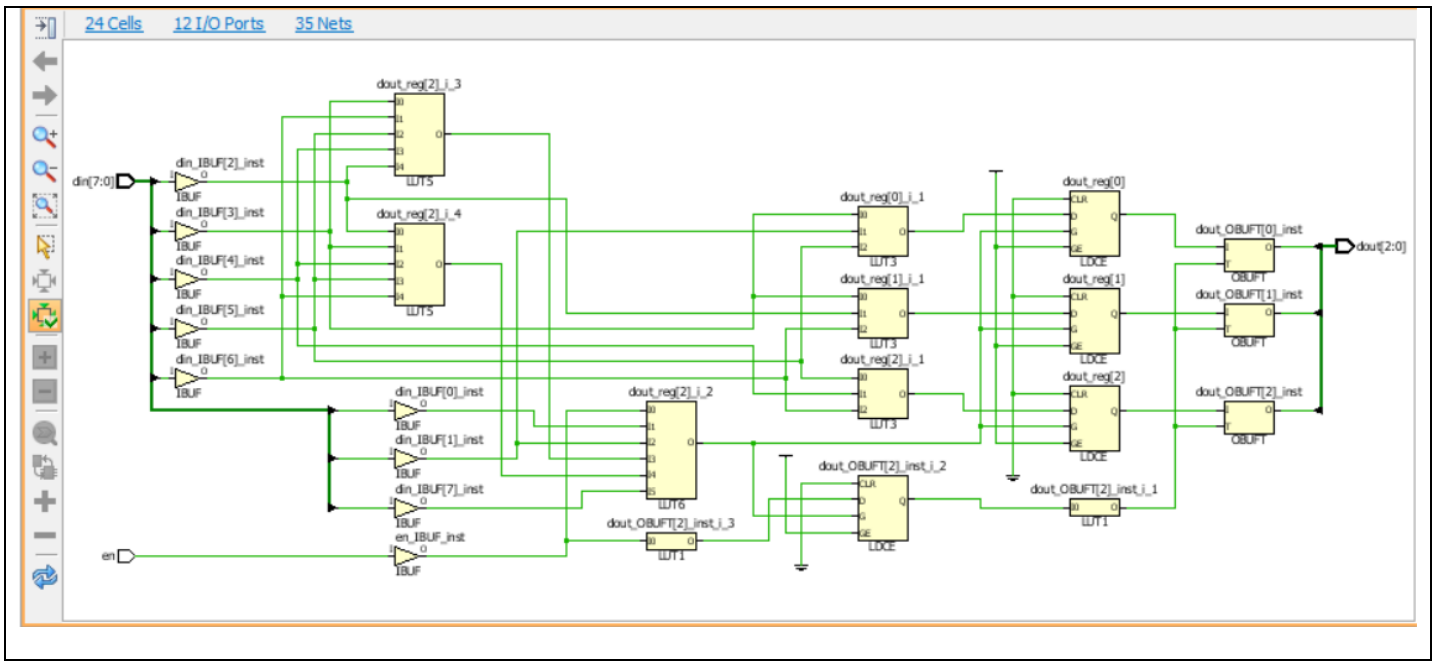
VERILOG CODE

```
module WPencode(en,Din,Dout);
input  en;
input [ 7 : 0 ] Din;
output [ 2 : 0 ] Dout;
reg [ 2 : 0 ] Dout;
always@(en,Din)
Begin
if(en == 1)
Begin
Dout = 3’bZZZ;
End
Else
Begin
Case (Din)
8’b00000001:Dout = 3’b000;
8’b00000010:Dout = 3’b001;
8’b00000100:Dout = 3’b010;
8’b00001000:Dout = 3’b011;
8’b00010000:Dout = 3’b100;
8’b00100000:Dout = 3’b101;
8’b01000000:Dout = 3’b110;
8’b01000000:Dout = 3’b111;
Endcase
End
End
Endmodule
```

RTL schematic



Technological schematic

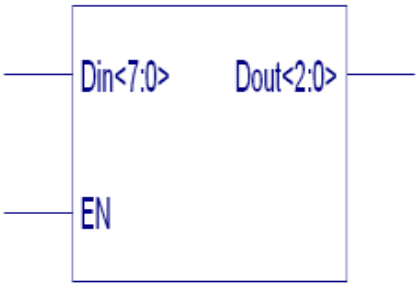


XDC file

```
## Switches
set_property PACKAGE_PIN V17 [get_ports {din[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[0]}]
set_property PACKAGE_PIN V16 [get_ports {din[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[1]}]
set_property PACKAGE_PIN W16 [get_ports {din[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[2]}]
set_property PACKAGE_PIN W17 [get_ports {din[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[3]}]
set_property PACKAGE_PIN W15 [get_ports {din[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[4]}]
set_property PACKAGE_PIN V15 [get_ports {din[5]}]
```

```
        set_property IOSTANDARD LVCMOS33 [get_ports {din[5]}]
set_property PACKAGE_PIN W14 [get_ports {din[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[6]}]
set_property PACKAGE_PIN W13 [get_ports {din[7]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[7]}]
set_property PACKAGE_PIN V2 [get_ports {en}]
        set_property IOSTANDARD LVCMOS33 [get_ports {en}]




## LEDs
set_property PACKAGE_PIN U16 [get_ports {dout[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {dout[0]}]
set_property PACKAGE_PIN E19 [get_ports {dout[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {dout[1]}]
set_property PACKAGE_PIN U19 [get_ports {dout[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {dout[2]}]
```

3. 8 TO 3 ENCODER WITH PRIORITY



RTL Schematic

Truth Table

INPUTS									OUTPUTS		
en	**Din(0)**	**Din(1)**	**Din(2)**	**Din(3)**	**Din(4)**	**Din(5)**	**Din(6)**	**Din(7)**	**Dout(0)**	**Dout(1)**	**Dout(3)**
0	x	x	x	x	x	x	x	x	z	z	z
1	x	x	x	x	x	x	x	1	1	1	1
1	x	x	x	x	x	x	1	0	1	1	0
1	x	x	x	x	x	1	0	0	1	0	1
1	x	x	x	x	1	0	0	0	1	0	0
1	x	x	x	1	0	0	0	0	0	1	1
1	x	x	1	0	0	0	0	0	0	1	0
1	x	1	0	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	0	0	0	0

VERILOG CODE

```
module priori (en,Din,Dout);
input en;
input [ 7 : 0 ] Din;
output [ 2 : 0 ] Dout;
reg [ 2 : 0 ] Dout;
always@(en,Din)
Begin
if(en == 1)   // Active high enable
Begin
Dout = 3'bZZZ; // Initializing Dout to high Impedance
```

```
End
Else
Begin
casex(Din)
8'b00000001 :Dout = 3'b000;
8'b0000001X :Dout = 3'b001;
8'b000001XX :Dout = 3'b010;
```

```
8'b00001XXX :Dout = 3'b011;
8'b0001XXXX :Dout = 3'b100;
8'b001XXXXX :Dout = 3'b101;
8'b01XXXXXX :Dout = 3'b110;
8'b1XXXXXXX :Dout = 3'b111;
Endcase
End
End
Endmodule
```

EXPERIMENT-7:
LATECHES & FLIP FLOPS

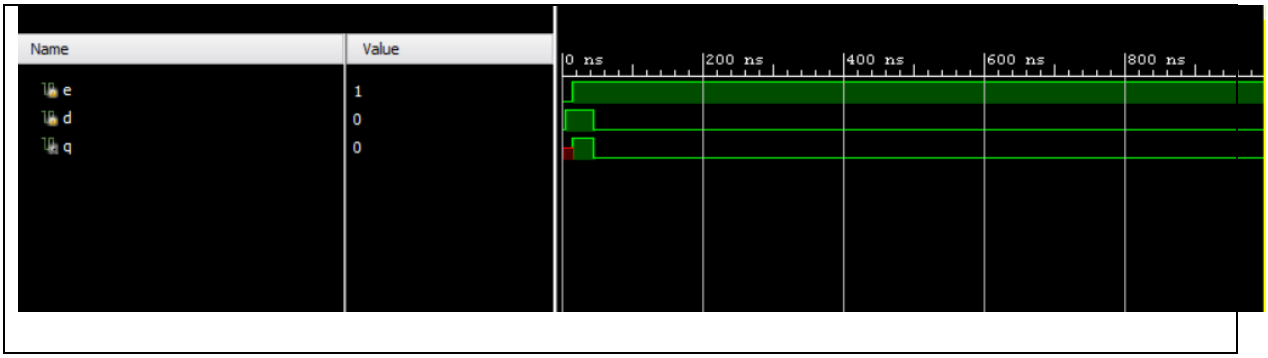
PROCEDURE:
Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (**Xilinx Artix-7 FPGA: XC7A35T-1CPG236C**)
Step2: Simulate the design using Vivado Simulator (Test Bench Module).
Step3: Synthesize the design and observe the Schematic.
Step4: Implement the design.
Step5: Write Design Constraint (XDC) file to constrain the pin locations.
Step6: Generate the bitstream.
Step7: Configure the FPGA using the generated bitstream and verify the functionality in hardware.

Aim: Write a Verilog code for the following:

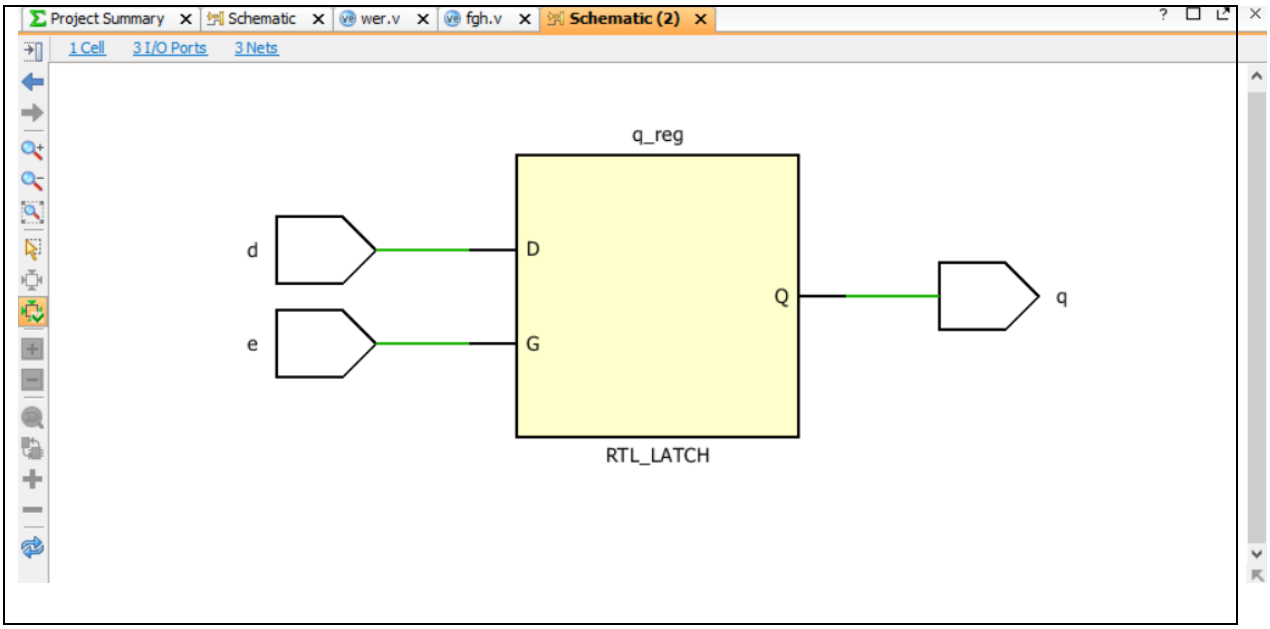
D LATCH

module dlatch(e, d, q);
input e;
input d;
output q;
reg q;
always @(e or d)
begin
if (e)
q<=d;
end
endmodule
	TEST BENCH
	module dlatch_tb;
	reg e;
	reg d;
	wire q;
	dlatch uut (.e(e),.d(d),.q(q));
	initial
	begin
	d = 0;
	e = 0;
	#5 d=1;
	#10 e=1;
	#10 d=1;
	#20 d=0;
	#10 e=1;
	end
	endmodule

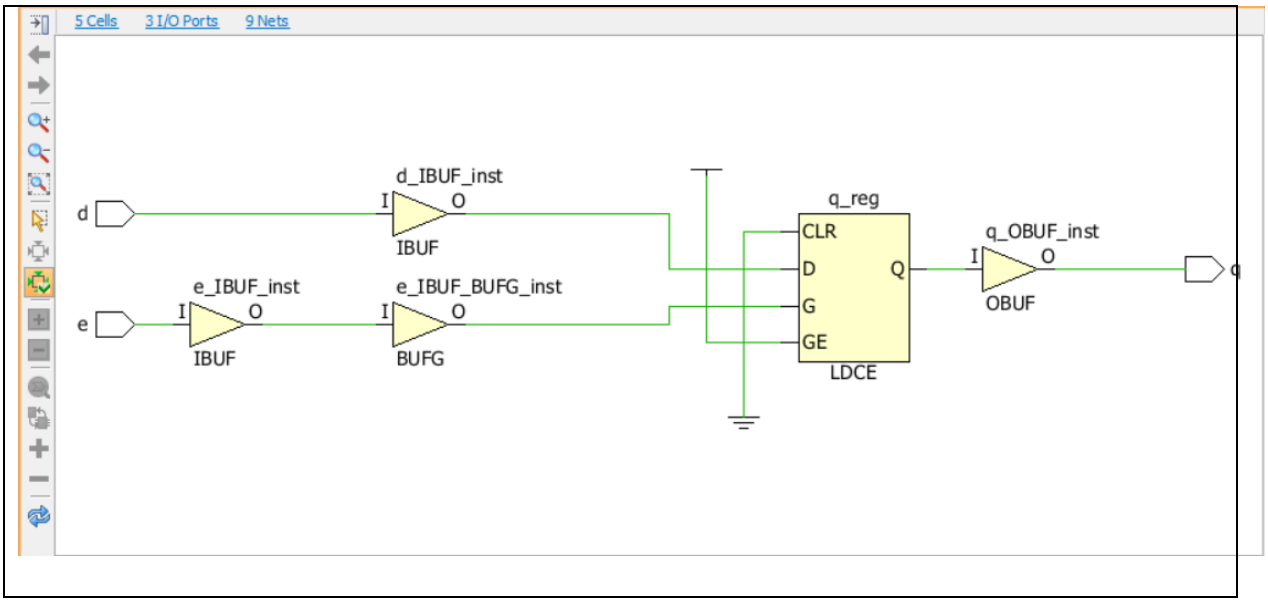
Simulation waveform



RTL schematic



Technological schematic



JK FLIP FLOP :

JK Flip Flop

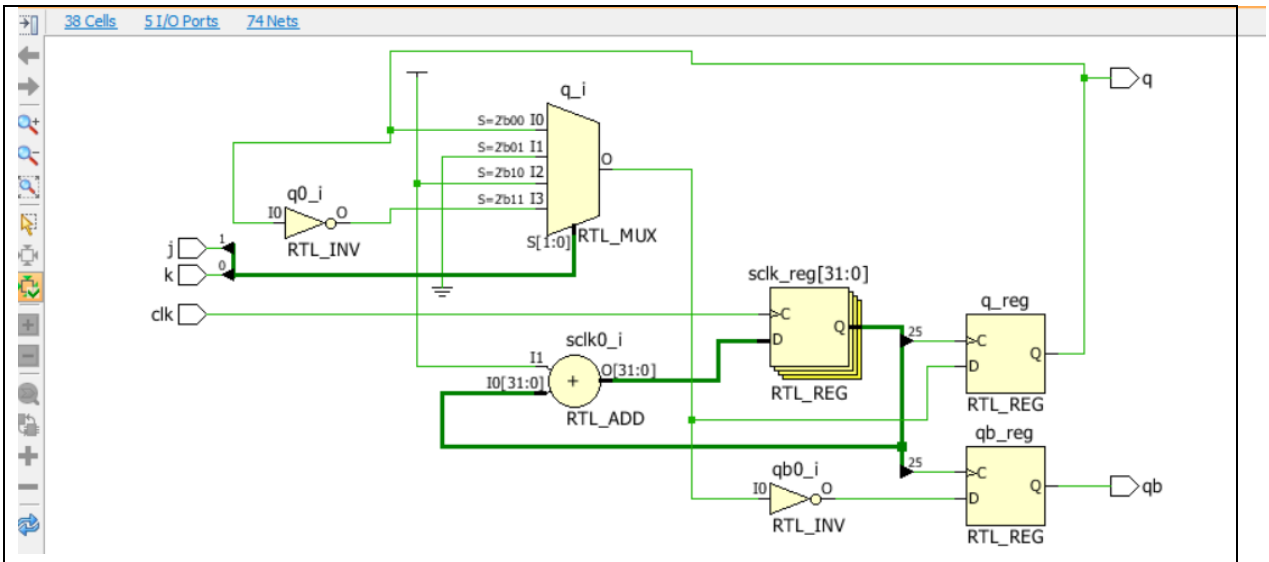
Design

```
module jk_ff ( input j,
               input k,
               input clk,
               output q);

   reg q;

   always @ (posedge clk)
      case ({j,k})
         2'b00 :  q <= q;
         2'b01 :  q <= 1;
         2'b10 :  q <= 0;
         2'b11 :  q <= ~q;
      endcase
endmodule
```

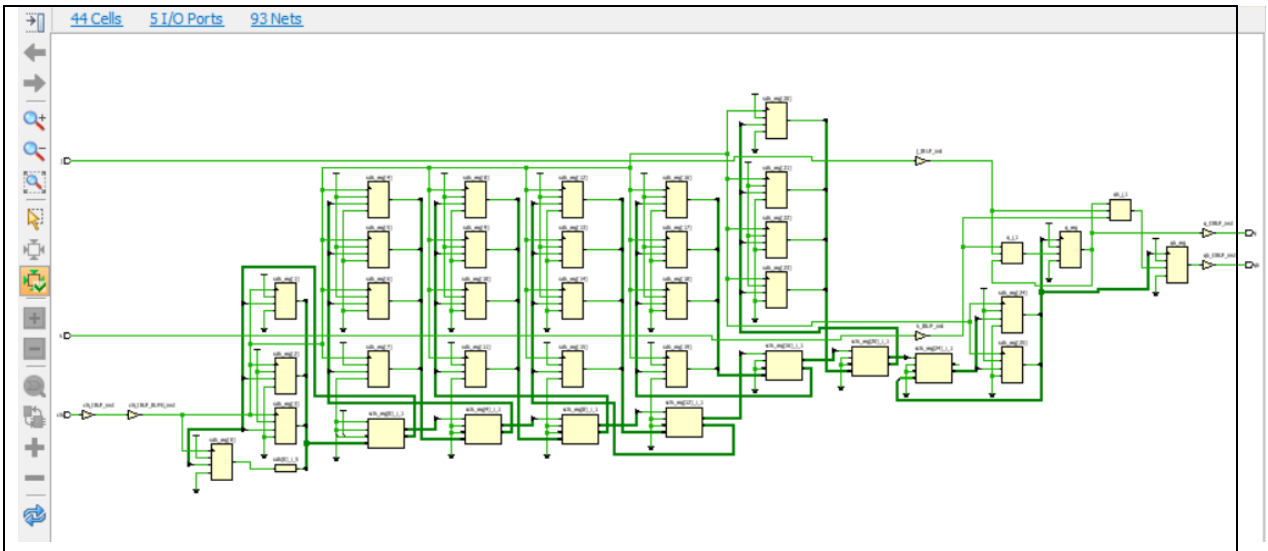
Testbench

```
module tb_jk;
   reg j;
   reg k;
   reg clk;

   always #5 clk = ~clk;

   jk_ff    jk0 ( .j(j),
                  .k(k),
                  .clk(clk),
                  .q(q));

   initial begin
      j <= 0;
      k <= 0;

      #5 j <= 0;
         k <= 1;
      #20 j <= 1;
          k <= 0;
      #20 j <= 1;
          k <= 1;
      #20 $finish;
   end

   initial
      $monitor ("j=%0d k=%0d q=%0d", j, k, q);
endmodule
```

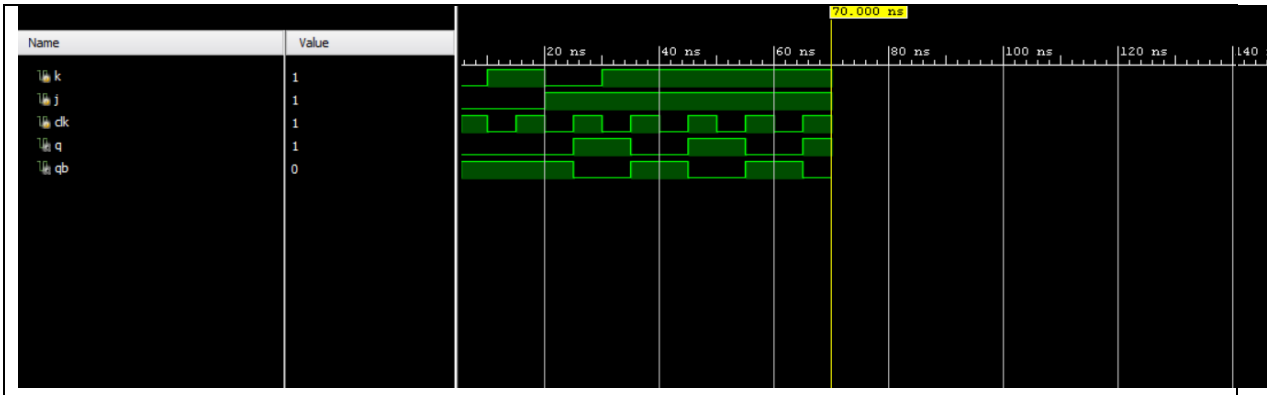
RTL schematic



Technological schematic



Simulation Waveform



Xdc file

```
## Clock signal

set_property PACKAGE_PIN W5 [get_ports clk]

        set_property IOSTANDARD LVCMOS33 [get_ports clk]

        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]


## Switches

set_property PACKAGE_PIN V17 [get_ports {j}]
```

```
        set_property IOSTANDARD LVCMOS33 [get_ports {j}]
set_property PACKAGE_PIN V16 [get_ports {k}]
        set_property IOSTANDARD LVCMOS33 [get_ports {k}]
## LEDs
set_property PACKAGE_PIN U16 [get_ports {q}]
        set_property IOSTANDARD LVCMOS33 [get_ports {q}]
set_property PACKAGE_PIN E19 [get_ports {qb}]
        set_property IOSTANDARD LVCMOS33 [get_ports {qb}]
```

EXPERIMENT-8

AIM: Simulation and FPGA Implementation of SHIFT REGISTERS using Xilinx Vivado tool

PROCEDURE:

Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (**Xilinx Artix-7 FPGA: XC7A35T-1CPG236C**)
Step2: Simulate the design using Vivado Simulator (Test Bench Module).
Step3: Synthesize the design and observe the Schematic.
Step4: Implement the design.
Step5: Write Design Constraint (XDC) file to constrain the pin locations.
Step6: Generate the bitstream.
Step7: Configure the FPGA using the generated bitstream and verify the functionality in hardware.

DESIGN SOURCE:

To design **(a) SISO (b) SIPO (c) PIPO (d) PISO** using Verilog

(i) SISO

	Testbench

module siso(din ,clk ,reset ,dout);
output dout ;
input din ;
input clk ;
input reset ;
reg [3:0]s;
always @ (posedge clk)
begin
if(reset)
s <= 4'b0;
else
begin
s[3] <= din;
s[2] <= s[3];
s[1] <= s[2];
s[0] <= s[1];
end
end
assign dout = s[0];
endmodule
	module tb_siso();
	wire dout;
	reg din,reset,clk;
	siso dut(din ,clk ,reset ,dout);
	initial begin
	reset = 0; clk = 0; din = 0;
	#5 reset = 1;
	#10 reset =0;
	#10 din = 1;
	#20 din = 0;
	#20 din = 1;
	#20 din = 0;
	#40 $stop;
	end
	always #10 clk = ! clk;
	endmodule

(ii)SIPO

```
module sipo ( din ,clk ,reset ,dout );
output [3:0] dout ;
input din;
input clk;
input reset ;
reg [3:0]s;
always @ (posedge clk)
begin if (reset)
s <= 0;
else begin
s[3] <= din;
s[2] <= s[3];
s[1] <= s[2];
s[0] <= s[1];
end
end
assign dout = s;
endmodule
```

Testbench

```
module tb_sipo;
 reg din,clk,reset;
wire [3:0] dout;
sipo uut(din, clk, reset,dout);

initial
begin
reset=0;
clk=0;
din=0;
#20 reset =1;
#20 reset=0;
#20 din=1;
#20 din=0;
#20 din=1;
#20 din=0;
#20 din=1;
#20 $finish;
end
always #10 clk = ~ clk;
endmodule
```

(iii)PIPO

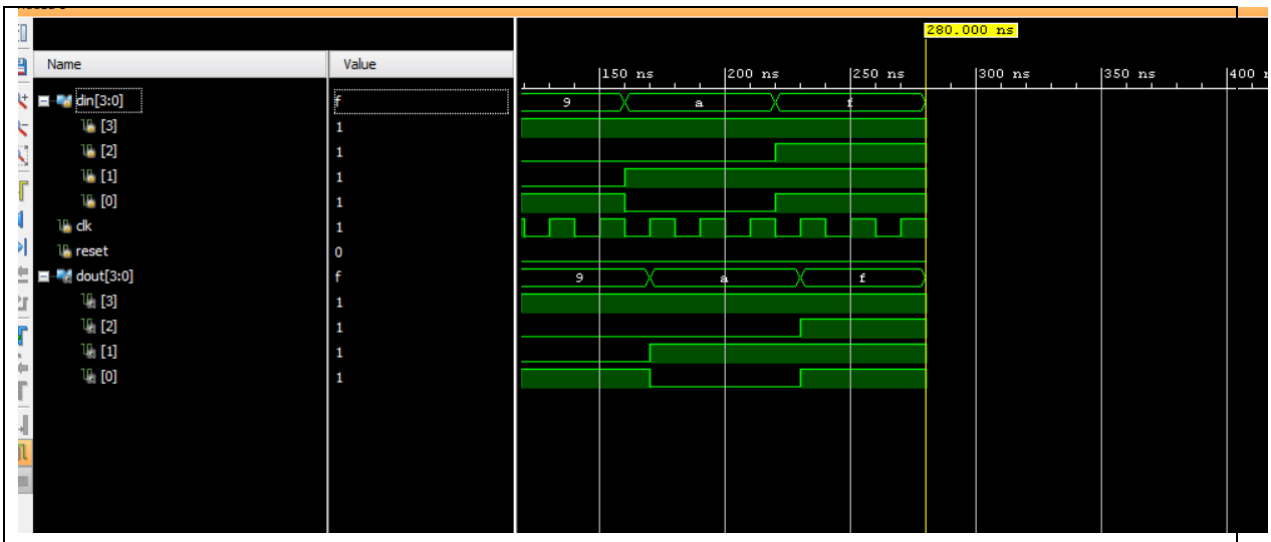
```
module PIPO ( din ,clk ,reset ,dout );
output [3:0] dout ;
reg [3:0] dout ;
input [3:0] din ;
input clk, reset ;

always @ (posedge clk)
begin if (reset)
dout <= 0;
else
dout <= din;
end
endmodule
```

Testbench

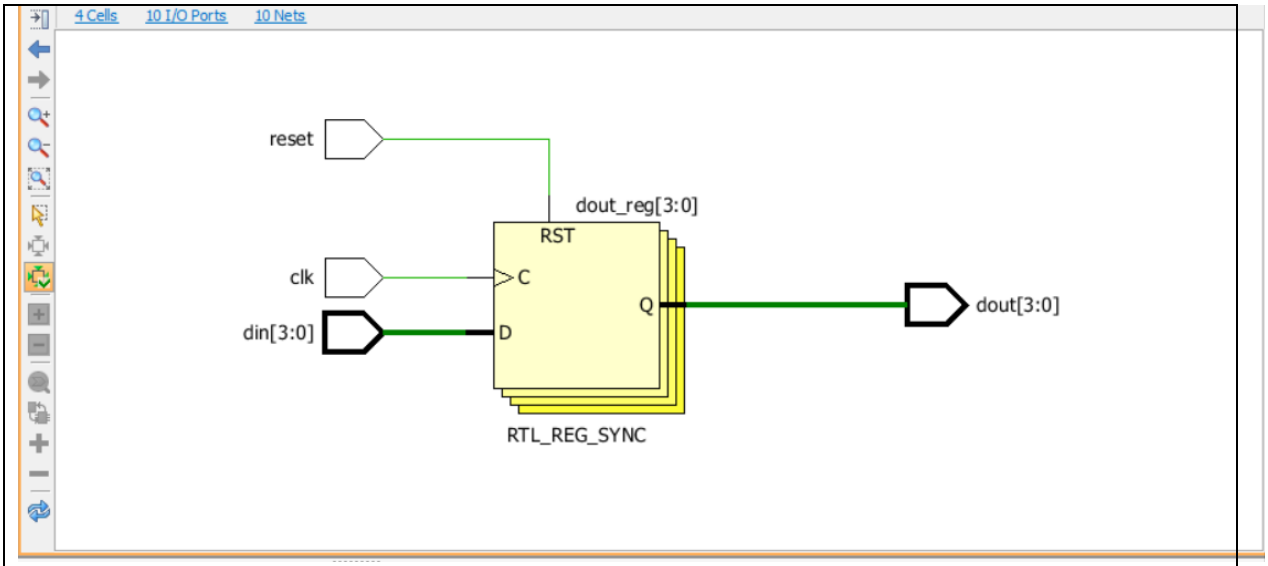
```
module tb_pipo;

 reg [3:0] din;


reg clk,reset;

wire [3:0] dout;

pipo uut(din, clk, reset,dout);


initial

begin

reset=0;

clk=0;

din=4'b0000;

#20 reset =1;

#20 reset=0;
```

```
#60 din=4'b1001;

#60 din=4'b1010;

#60 din=4'b1111;


#60 $finish;

end

always #10 clk = ~ clk;

endmodule
```

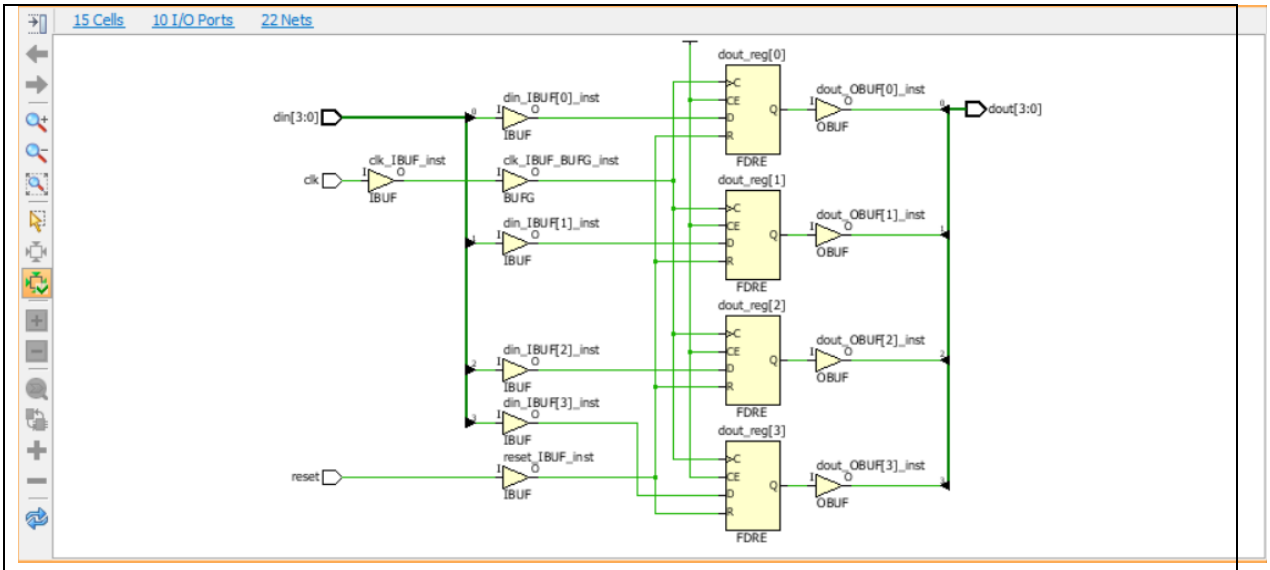
Simulation Waveform



RTL schematic



Technological schematic



XDC file

```
## Clock signal

set_property PACKAGE_PIN W5 [get_ports clk]

        set_property IOSTANDARD LVCMOS33 [get_ports clk]

        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
```

```
## Switches
set_property PACKAGE_PIN V17 [get_ports {din[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[0]}]
set_property PACKAGE_PIN V16 [get_ports {din[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[1]}]
set_property PACKAGE_PIN W16 [get_ports {din[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[2]}]
set_property PACKAGE_PIN W17 [get_ports {din[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {din[3]}]
set_property PACKAGE_PIN W15 [get_ports {reset}]
        set_property IOSTANDARD LVCMOS33 [get_ports {reset}]

## LEDs
set_property PACKAGE_PIN U16 [get_ports {dout[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {dout[0]}]
set_property PACKAGE_PIN E19 [get_ports {dout[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {dout[1]}]
set_property PACKAGE_PIN U19 [get_ports {dout[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {dout[2]}]
set_property PACKAGE_PIN V19 [get_ports {dout[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {dout[3]}]
```

Report Utilization Summary

C:/Users/gnchi/project_8_shift registers/project_8_shift registers.runs/synth_1/pipo_utilization_synth.rpt Read-onl

```
28 --------------
29
30 +-------------------------+------+-------+-----------+-------+
31 |        Site Type        | Used | Fixed | Available | Util% |
32 +-------------------------+------+-------+-----------+-------+
33 | Slice LUTs*             |    0 |     0 |     20800 |  0.00 |
34 |   LUT as Logic          |    0 |     0 |     20800 |  0.00 |
35 |   LUT as Memory         |    0 |     0 |      9600 |  0.00 |
36 | Slice Registers         |    4 |     0 |     41600 | <0.01 |
37 |   Register as Flip Flop |    4 |     0 |     41600 | <0.01 |
38 |   Register as Latch     |    0 |     0 |     41600 |  0.00 |
39 | F7 Muxes                |    0 |     0 |     16300 |  0.00 |
40 | F8 Muxes                |    0 |     0 |      8150 |  0.00 |
41 +-------------------------+------+-------+-----------+-------+
42 * Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_desig
43
```

(iv) PISO

```
module piso ( din ,clk ,reset ,load ,dout );
output dout ;
reg dout ;
input [3:0] din ;
input load,reset,clk ;
reg [3:0]temp;

always @ (posedge clk)
begin if (reset)
temp <= 1;
else if (load)
temp <= din;
else
begin
dout  <= temp[3];
temp <= {temp[2:0],1'b0};
end
end
endmodule
```

Testbench

```
module tb_piso;

reg [3:0] din;
```

```
reg clk,reset,load;

wire dout;

piso uut(din, clk, reset,loaddout);


initial

begin

reset=0;

load=0;

clk=0;

din=4'b0000;

#20 reset =1;

#20 reset=0;

#20 load =1;

#20 load =0;


#60 din=4'b1001;

#60 din=4'b1010;

#60 din=4'b1111;


#60 $finish;

end

always #10 clk = ~ clk;

endmodule
```

EXPERIMENT-9

AIM: Simulation of verilog code for Asynchronous counter and Synchronous counter in Xilinx Vivado tool

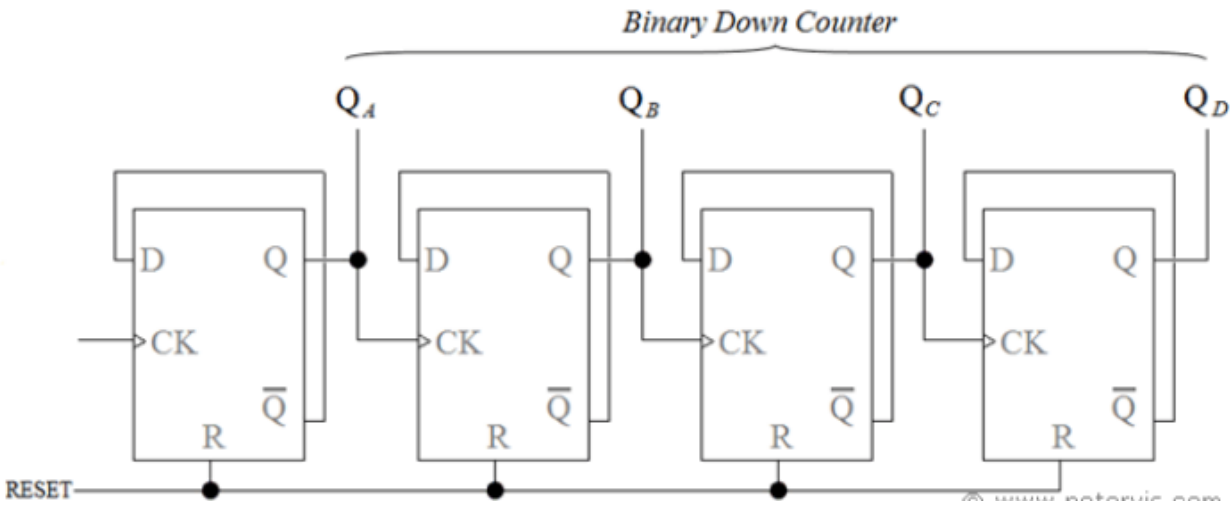
PROCEDURE:

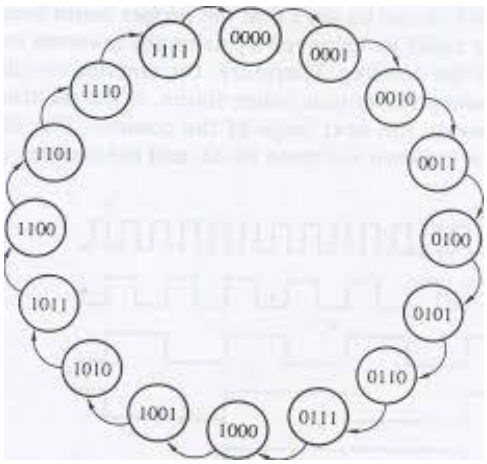
Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (Xilinx Artix-7 FPGA: XC7A35T-1CPG236C)

Step2: Simulate the design using Vivado Simulator (Test Bench Module)

Step3: Synthesize the design and observe the Schematic.

(i) Asynchronous Counter





Verilog Code

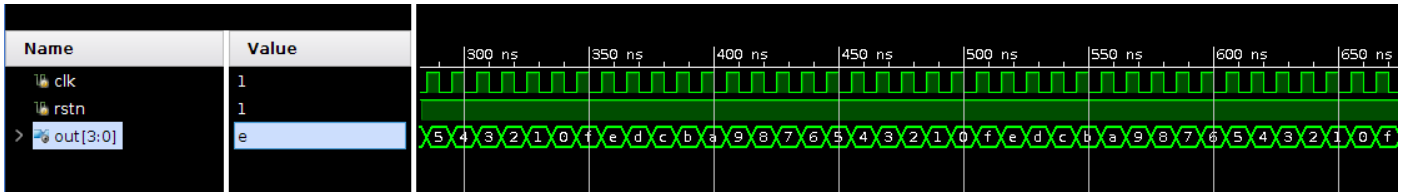
```
module dff (   input d,input clk, input rstn, output reg q,output qn);
   always @ (posedge clk or negedge rstn)
      if (!rstn)
         q <= 0;
      else
         q <= d;
 assign qn = ~q;
endmodule

module ripple ( input clk,
               input rstn,
               output [3:0] out);
   wire  q0,q1,q2,q3;
   wire  qn0,qn1,qn2,qn3;
   dff   dff0 ( .d (qn0), .clk (clk), .rstn (rstn), .q (q0), .qn (qn0));
   dff   dff1 ( .d (qn1), .clk (q0),  .rstn (rstn), .q (q1),  .qn (qn1));
   dff   dff2 ( .d (qn2), .clk (q1),.rstn (rstn),.q (q2),  .qn (qn2));
   dff   dff3 ( .d (qn3), .clk (q2), .rstn (rstn), .q (q3), .qn (qn3));
   assign out = {q3, q2, q1, q0};
endmodule
```

Testbench

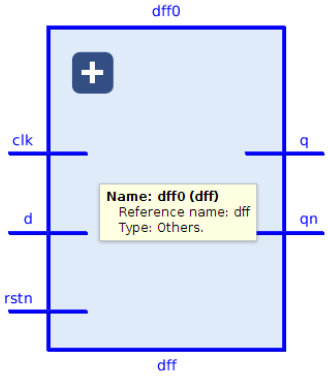
```
module tb_ripple;
   reg clk;
   reg rstn;
   wire [3:0] out;
  ripple r0   (  .clk (clk), .rstn (rstn),.out (out));
initial begin
      rstn = 0;
      clk = 0;
#20 rstn = 1;
end
always #5 clk = ~clk;
```

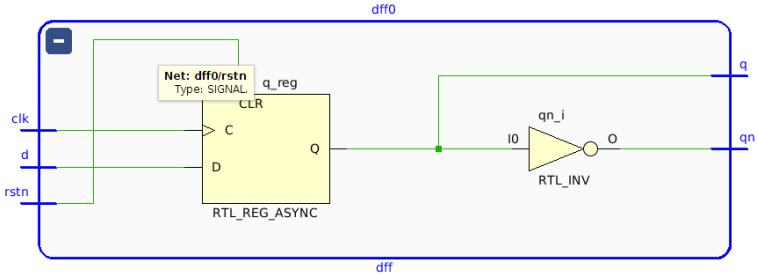
endmodule

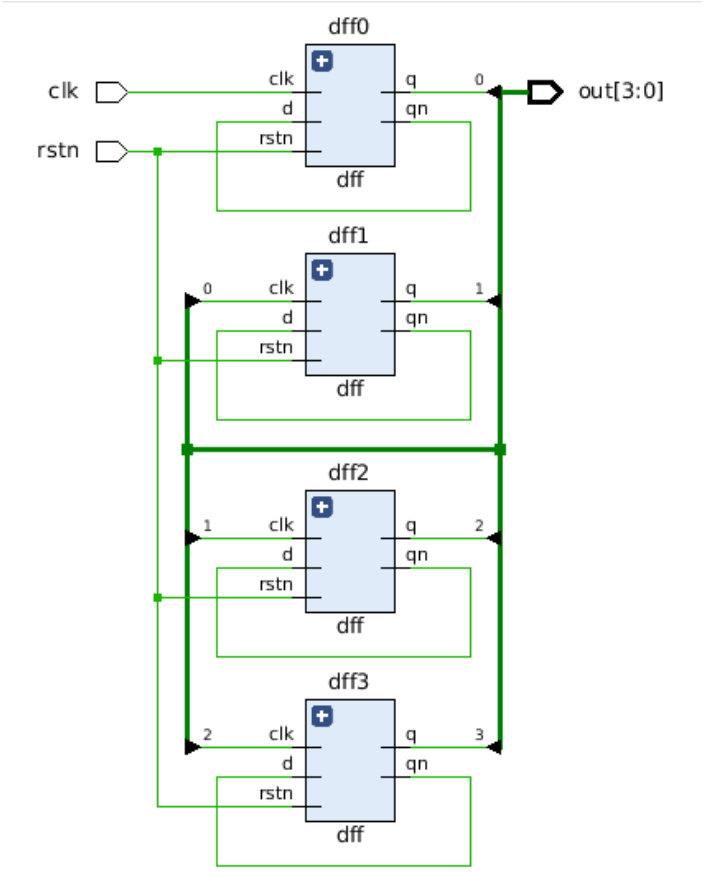
Simulation result:



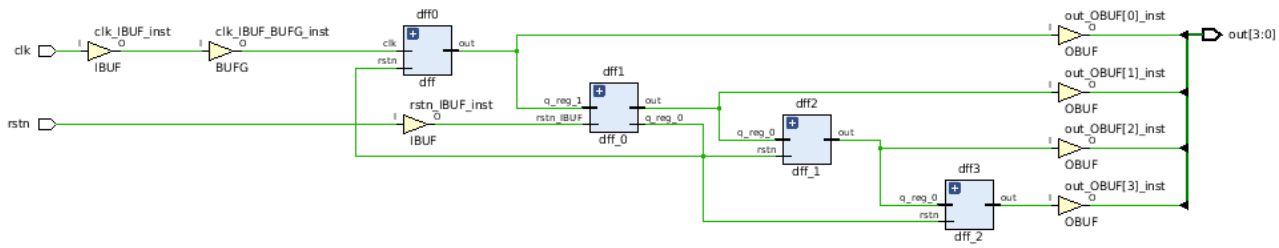
RTL Schematic:





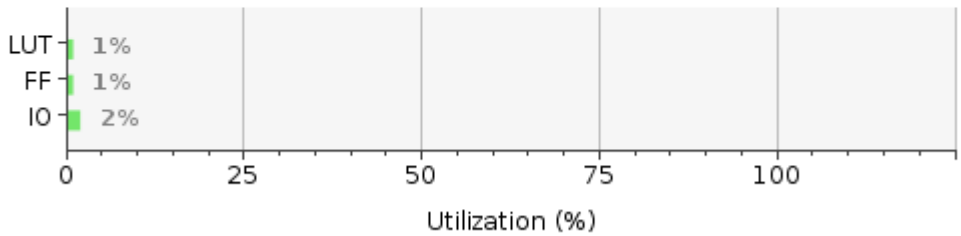


Post synthesis schematic:



Utilization report:

Resource	Utilization	Available	Utilization %
LUT	5	134600	0.00
FF	4	269200	0.00
IO	6	400	1.50



(ii)Synchronous Counter

Verilog code

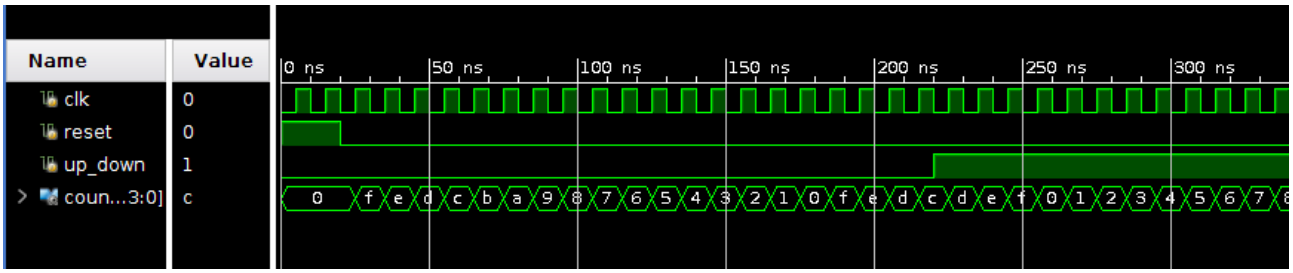
```
module up_down_counter(input clk, reset,up_down, output[3:0]  counter );
reg [3:0] counter_up_down;
always @(posedge clk or posedge reset)
begin
if(reset)
```

```
 counter_up_down <= 4'h0;
else
 if(up_down)
 counter_up_down <= counter_up_down + 4'd1;
else
 counter_up_down <= counter_up_down - 4'd1;
end
assign counter = counter_up_down;
endmodule
```

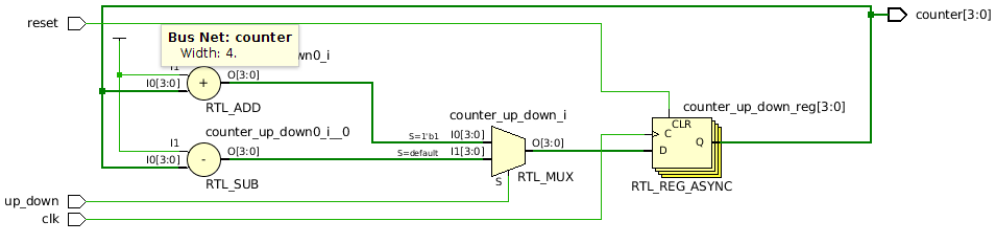
Testbench

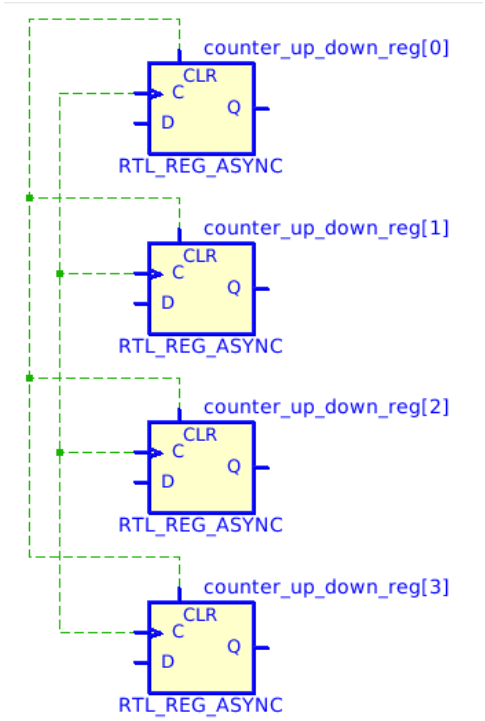
```
module updowncounter_testbench();
reg clk, reset,up_down;
wire [3:0] counter;

up_down_counter dut(clk, reset,up_down, counter);
initial begin
clk=0;
forever #5 clk=~clk;
end
initial begin
reset=1;
up_down=0;
#20;
reset=0;
#200;
up_down=1;
end
endmodule
```

Simulation result:

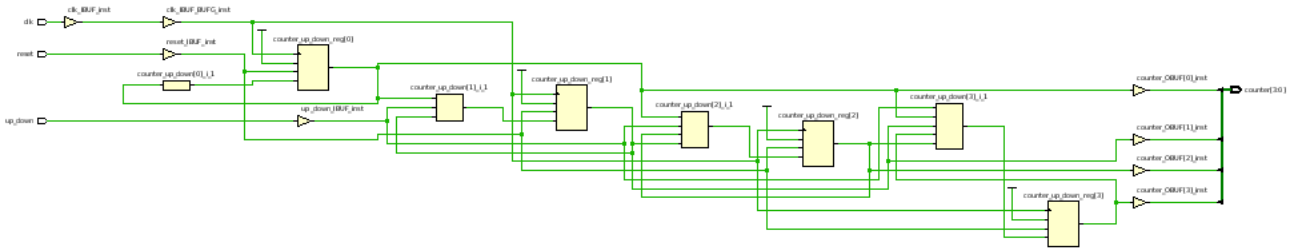


RTL Schematic:



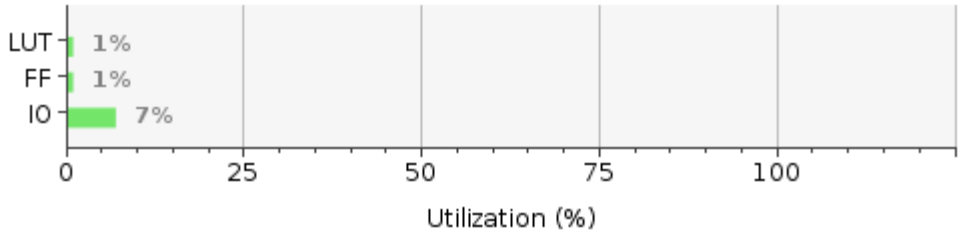


Post synthesis schematic:



Utilization report:

Resource	Utilization	Available	Utilization %
LUT	2	20800	0.01
FF	4	41600	0.01
IO	7	106	6.60



EXPERIMENT-10

AIM: Design and Simulate of Mealy & Moore Machine Sequence Detector in Verilog using Xilinx Vivado tool

PROCEDURE:

Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (Xilinx Artix-7 FPGA: XC7A35T-1CPG236C)

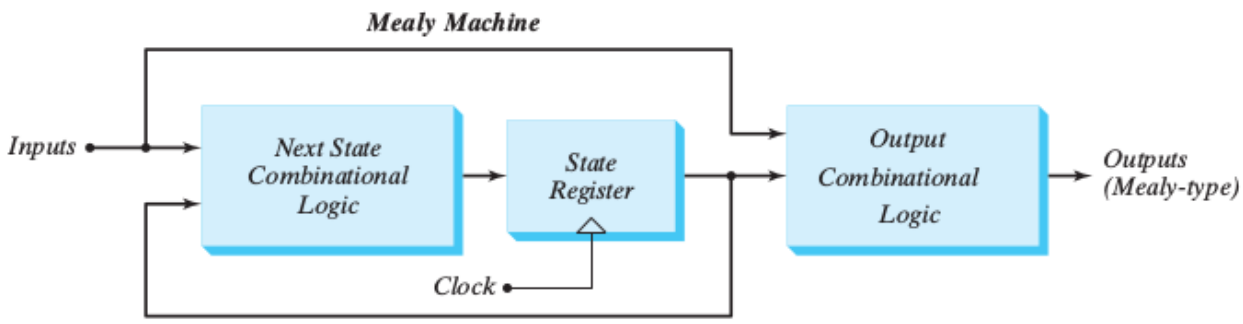
Step2: Simulate the design using Vivado Simulator (Test Bench Module).

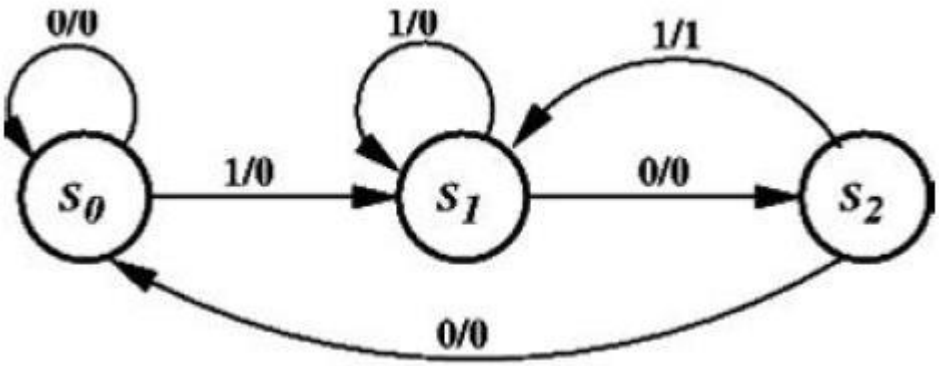
Step3: Synthesize the design and observe the Schematic.

THEORY:

Design a Mealy and Moore system to detect 101 sequence in a given set of input datas.

(i) FSM using Mealy Machine:





Design Source of Mealy Machine:

```
module  sequence_detector_mealy(

input rst,

input clk,

input in_seq,

output reg out_detect

);

reg in_seq_reg;

parameter size =2,s0=2’b00,s1=2’b01,s2=2’b10;

reg [size -1 :0] state, next_state;

always @(posedge clk)

begin

if (rst)

begin

        in_seq_reg <=0;

        state <=0;

end

else begin

        in_seq_reg <= in_seq;

        state<= next_state;
```

```
        end
end
always @ (state or  in_seq_reg or rst)
begin
next_state<=0;
if (rst)
        out_detect<=0;
else begin
        case(state)
        s0 : if(in_seq_reg == 1’b0)
                begin
                        next_state <=s0;
                        out_detect <= 1’b0;
                end
        else
                begin
                        next_state <=s1;
                        out_detect <= 1’b0;
                end
s1 : if(in_seq_reg == 1’b0)
```

```
                begin
                        next_state <=s2;
                        out_detect <= 1'b0;
                end
        else
                begin
                        next_state <=s1;
                        out_detect <= 1'b0;
                end
s2 : if(in_seq_reg == 1'b0)
                begin
                        next_state <=s0;
                        out_detect <= 1'b0;
                end
        else
                begin
                        next_state <=s1;
                        out_detect <= 1'b1;
                end
default : begin
```

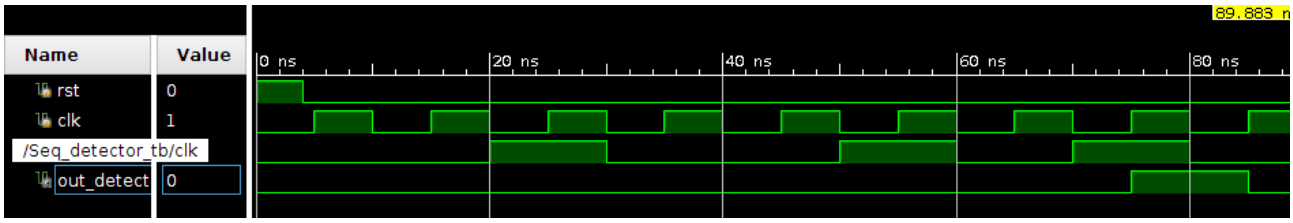
```
                next_state <=s0;

                out_detect <= 1'b0;

        end

endcase

end

end

endmodule
```

Test bench Program:

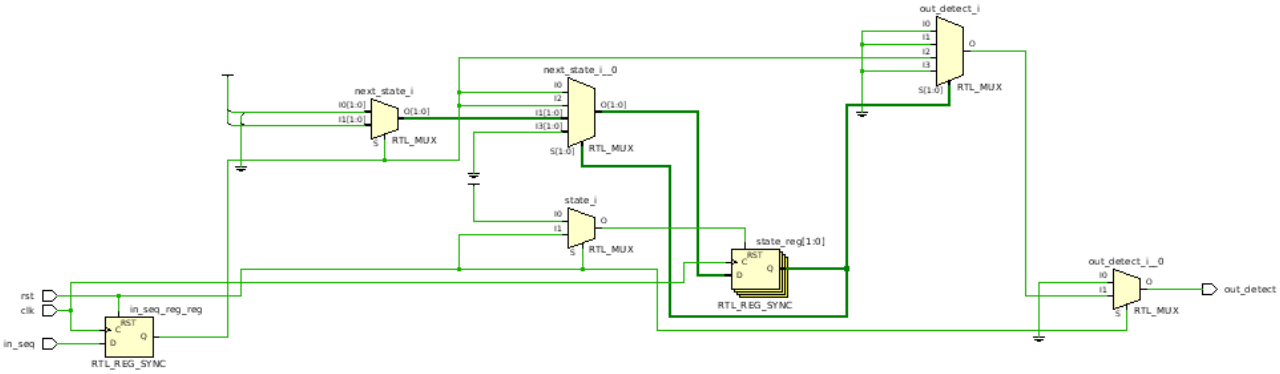
```
module  Seq_detector_tb;

reg rst;

reg clk;

reg in_seq;

wire out_detect;

sequence_detector_mealy uut(

.rst(rst), .clk(clk), .in_seq(in_seq), .out_detect(out_detect));

always #5 clk= ~clk;

initial

begin

$monitor($time, "rst= %b  in_seq=%b out=%b", rst, in_seq,out_detect);

rst=1;
```

```
clk=0;

in_seq =0;

#4 rst=0;

#6 in_seq=0;

#10 in_seq=1;

#10 in_seq=0;

#10 in_seq=0;

#10 in_seq=1;

#10 in_seq=0;

#10 in_seq=1;

#10 in_seq=0;

#10 $finish;

end

endmodule
```

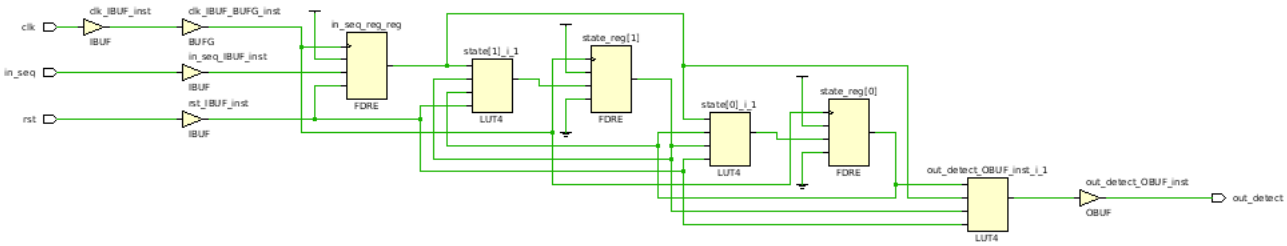
Simulation result:



RTL Schematic:

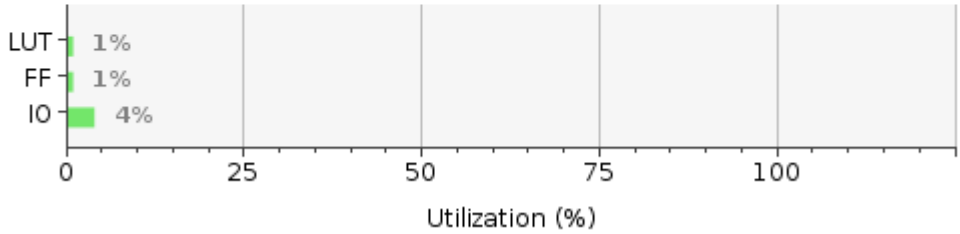


Post synthesis schematic:

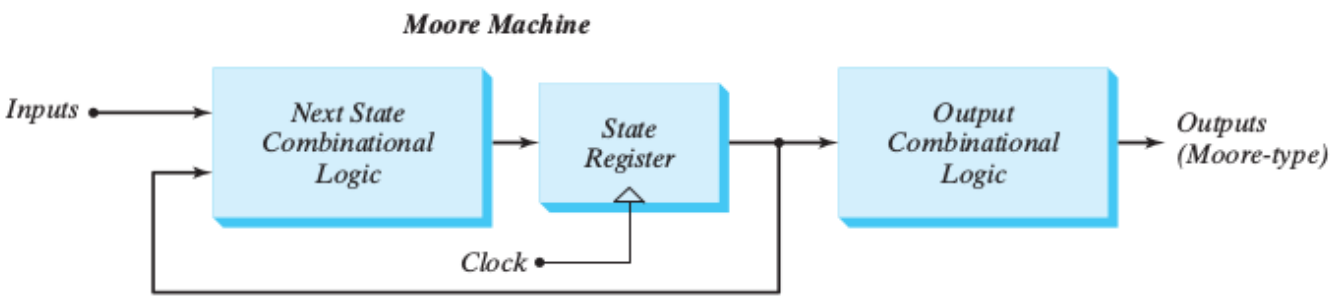


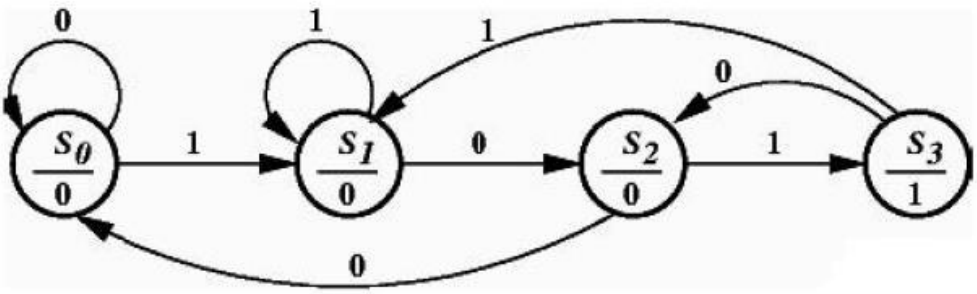
Utilization report:

Resource	Utilization	Available	Utilization %
LUT	2	20800	0.01
FF	3	41600	0.01
IO	4	106	3.77



ii) FSM Using Moore Machine:





Design Source of Moore_Machine:

```
module  sequence_detector_moore(

input rst,

input clk,

input in_seq,

output reg out_detect

);

reg in_seq_reg;

parameter size =4,s0=4'b0001,s1=4'b0010,s2=4'b0100, s3 = 4'b1000;
```

```
reg [size -1 :0] state, next_state;

always @(posedge clk)

begin

if (rst)

begin

        in_seq_reg <=0;

        state <=s0;

end

else begin

        in_seq_reg <= in_seq;

        state<= next_state;

        end

end

always @ (state or  in_seq_reg or rst)

begin



if (rst)

        next_state<=0;



else begin
```

```
        case(state)
        s0 : if(in_seq_reg == 1'b0)
                next_state <=s0;
        else    next_state <= s1;
        s1 : if(in_seq_reg == 1'b0)
                next_state <=s2;
        else    next_state <= s1;
s2 : if(in_seq_reg == 1'b0)
                next_state <=s0;
        else    next_state <= s3;
s1 : if(in_seq_reg == 1'b0)
                next_state <=s1;
        else    next_state <= s2;
default : next_state <=s0;
endcase
end
end
always@(state or rst)
if (rst) out_detect=0;
else
```

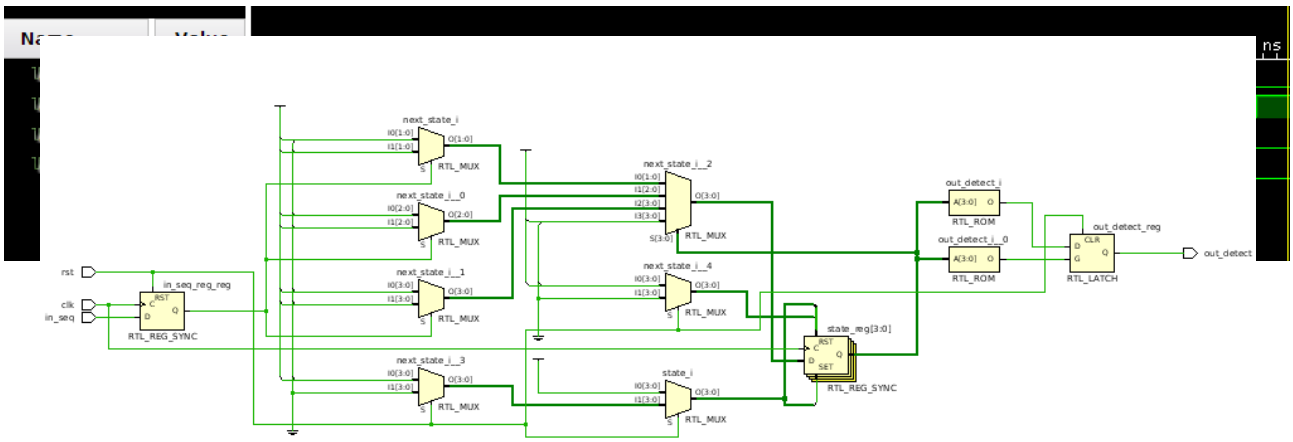
```
case(state)

s0: out_detect=0;

s1: out_detect=0;

s2: out_detect=0;

s3: out_detect=1;

endcase

endmodule
```

Test bench Program:

```
module  Seq_detector_tb;

reg rst;

reg clk;

reg in_seq;

wire out_detect;

sequence_detector_moore uut(

.rst(rst), .clk(clk), .in_seq(in_seq), .out_detect(out_detect));

always #5 clk= ~clk;

initial

begin

$monitor($time, “rst= %b  in_seq=%b out=%b”, rst, in_seq,out_detect);
```

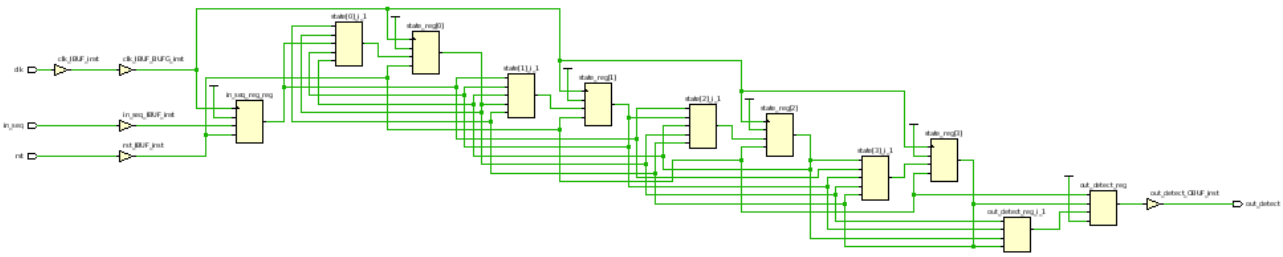
```
rst=1;

clk=0;

in_seq =0;

#4 rst=0;

#6 in_seq=0;

#10 in_seq=1;

#10 in_seq=0;

#10 in_seq=0;

#10 in_seq=1;

#10 in_seq=0;

#10 in_seq=1;

#10 in_seq=0;

#10 $finish;

end

endmodule
```

Simulation result:



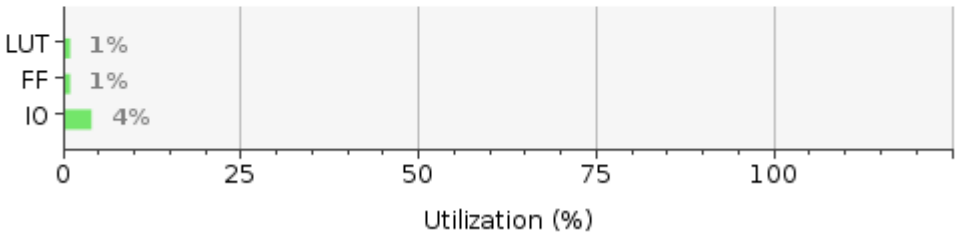
RTL Schematic:

Post synthesis schematic:



Utilization report:

Resource	Utilization	Available	Utilization %
LUT	3	10400	0.03
FF	6	20800	0.03
IO	4	106	3.77



EXPERIMENT – 11

AIM:

Simulation and FPGA Implementation of Arithmetic and Logic Unit(ALU) in Xilinx Vivado tool

PROCEDURE:

Step1: Create a Vivado Project using IDE sourcing verilog HDL model and targeting a specific FPGA device located on the Basys3 (Xilinx Artix-7 FPGA: XC7A35T-1CPG236C)

Step2: Simulate the design using Vivado Simulator (Test Bench Module).

Step3: Synthesize the design and observe the Schematic.

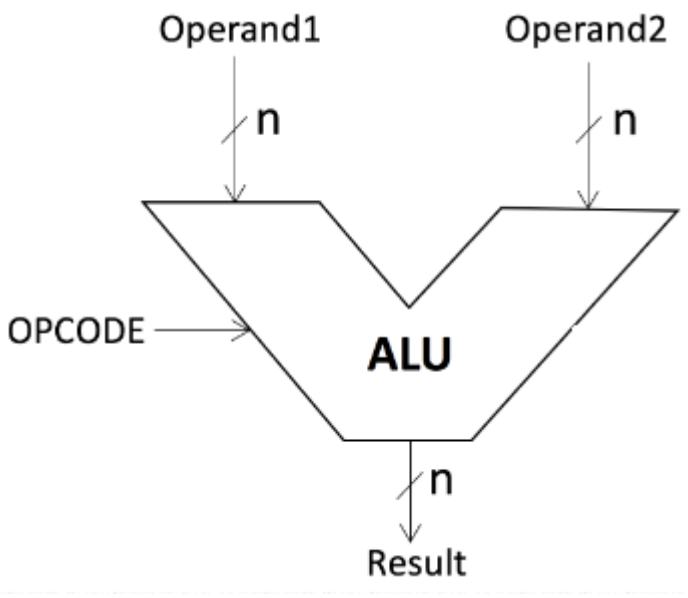
Step4: Implement the design.

Step5: Write Design Constraint (XDC) file to constrain the pin locations.

Step6: Generate the bitstream.

Step7: Configure the FPGA using the generated bitstream and verify the functionality in hardware.

ALU:



ALU DESIGN USING BEHAVIOURAL MODELLING:

PROJECT MANAGER - project_1_alu

Sources

Source File Properties

Project Summary | alu_design.v

/home/dsplab/project_1_alu/project_1_alu.srcs/sources_1/new/alu_design.v

```
// Design Name:
// Module Name: alu_design
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////


module alu_design(
    input [3:0] a,
    input [3:0] b,
    input [2:0] opcode,
    output [7:0] outdata
        );
        reg [7:0]res;

        assign outdata = res;
        always @ (opcode, a , b)
        begin
        case (opcode)
         3'b001: res = a + b;
         3'b010: res = a - b;
         3'b011: res = a * b;
         3'b100: res =~a;
         3'b101: res = a & b;
         3'b110: res = a | b;
         3'b111: res =  (a  ^ b);
         default: res = 4'bx;
         endcase
         end

endmodule
```

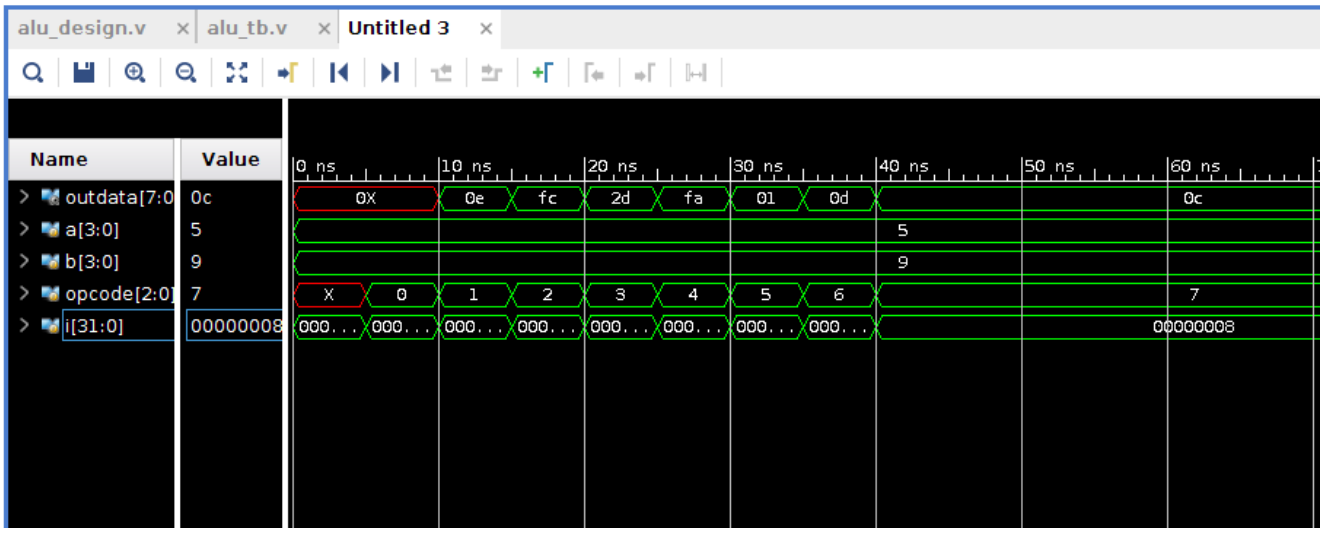
TESTBENCH FOR ALU:

Project Summary | alu_design.v | alu_tb.v

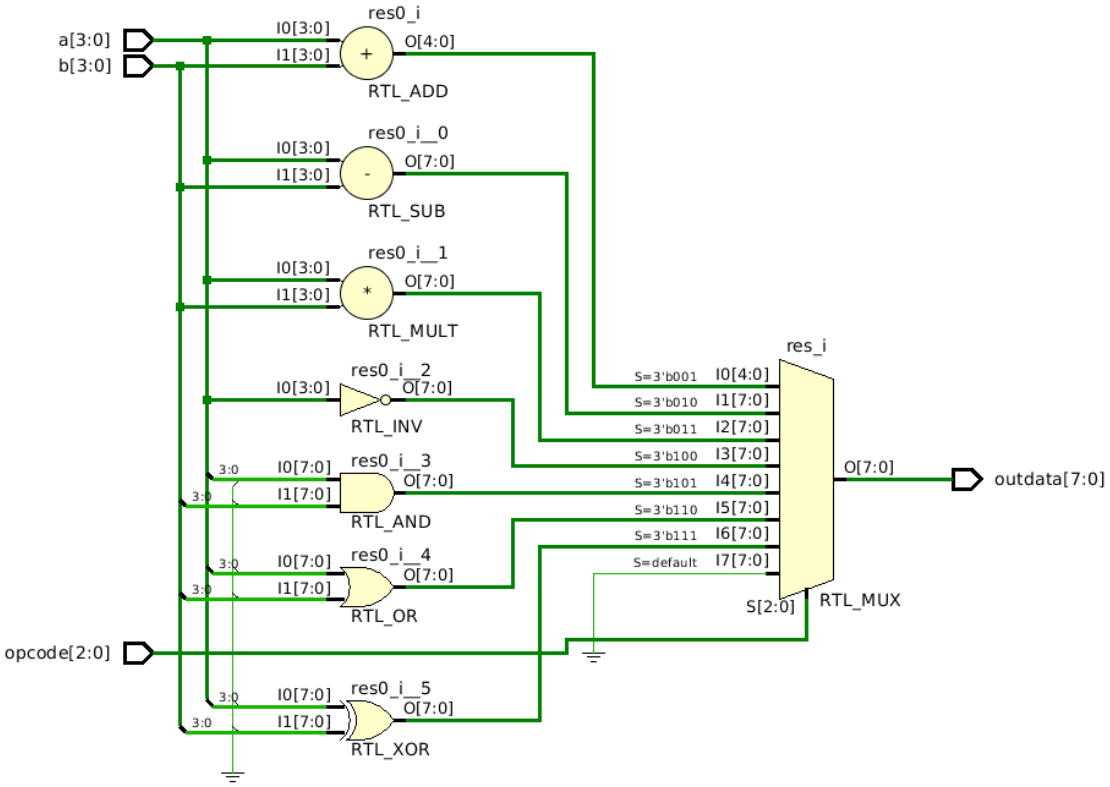
/home/dsplab/project_1_alu/project_1_alu.srcs/sim_1/new/alu_tb.v

```
module alu_tb(  );
wire [7:0]outdata;
reg [3:0]a,b;
reg [2:0]opcode;
integer i;

alu_design dut(a,b,opcode,outdata);

initial begin
a = 4'b0101;
b = 4'b1001;
for(i = 0; i < 8; i = i + 1)
#5 opcode = i;
end

endmodule
```

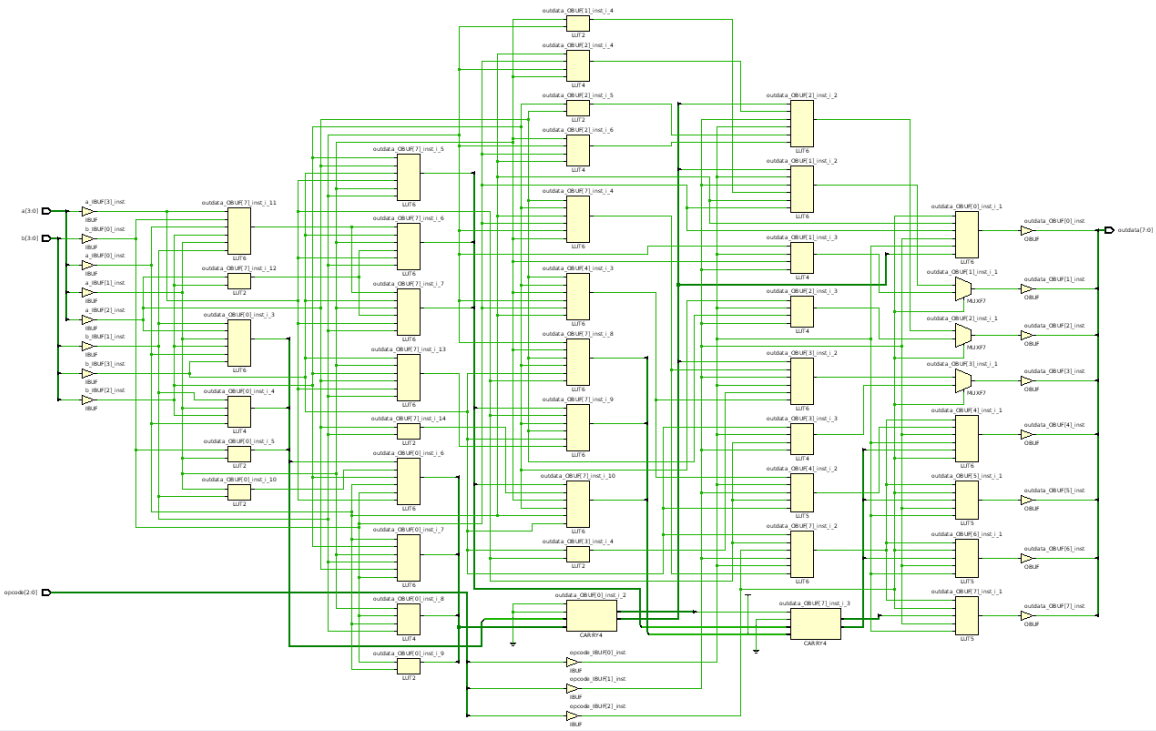
SIMULATION RESULT:



RTL SCHEMATIC:



POST SYNTHESIS SCHEMATIC:



XDC FILE:

Package | Device | alu_design.v | alu_tb.v | alu_xdc.xdc

/home/dsplab/project_1_alu/project_1_alu.srcs/constrs_1/new/alu_xdc.xdc

```
set_property PACKAGE_PIN R2 [get_ports {a[3]}]
set_property PACKAGE_PIN T1 [get_ports {a[2]}]
set_property PACKAGE_PIN U1 [get_ports {a[1]}]
set_property PACKAGE_PIN W2 [get_ports {a[0]}]
set_property PACKAGE_PIN R3 [get_ports {b[3]}]
set_property PACKAGE_PIN T2 [get_ports {b[2]}]
set_property PACKAGE_PIN T3 [get_ports {b[1]}]
set_property PACKAGE_PIN V2 [get_ports {b[0]}]
set_property PACKAGE_PIN W13 [get_ports {opcode[2]}]
set_property PACKAGE_PIN W14 [get_ports {opcode[1]}]
set_property PACKAGE_PIN V15 [get_ports {opcode[0]}]
set_property PACKAGE_PIN L1 [get_ports {outdata[7]}]
set_property PACKAGE_PIN P1 [get_ports {outdata[6]}]
set_property PACKAGE_PIN N3 [get_ports {outdata[5]}]
set_property PACKAGE_PIN P3 [get_ports {outdata[4]}]
set_property PACKAGE_PIN U3 [get_ports {outdata[3]}]
set_property PACKAGE_PIN W3 [get_ports {outdata[2]}]
set_property PACKAGE_PIN V3 [get_ports {outdata[1]}]
set_property PACKAGE_PIN V13 [get_ports {outdata[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {opcode[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {opcode[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {opcode[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {outdata[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {outdata[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {outdata[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {outdata[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {outdata[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {outdata[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {outdata[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {outdata[0]}]
```

UTILIZATION REPORT:

Resource	Utilization	Available	Utilization %
LUT	33	20800	0.16
IO	19	106	17.92