

Experiment 4

Title of the Experiment: To design Encoder and Decoder using Verilog code and compare with their respective truth tables.

Objective/Motivation: In this lab, a 8x3 encoder and 3x8 decoder are designed. The objective will be to test these designs on Xilinx simulation tool. The tests will be performed for all the possible combinations of inputs to verify their functionality. Moreover, the knowledge gained will be used to design complex designs.

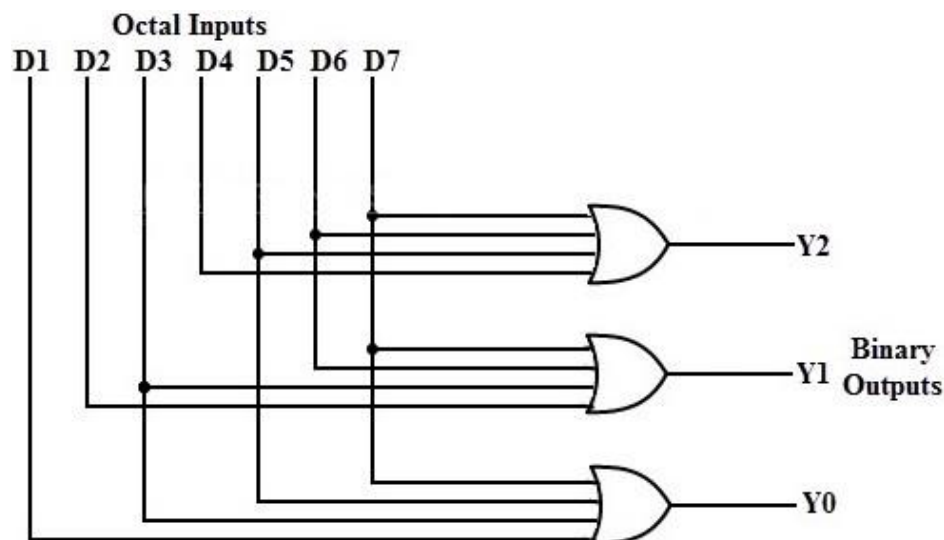
Equipment required:

1. Zybo board - Zynq XC7Z010
2. Xilinx Vivado Design Suite 2016

Logic diagram(s):

Encoder: Logic

Diagram

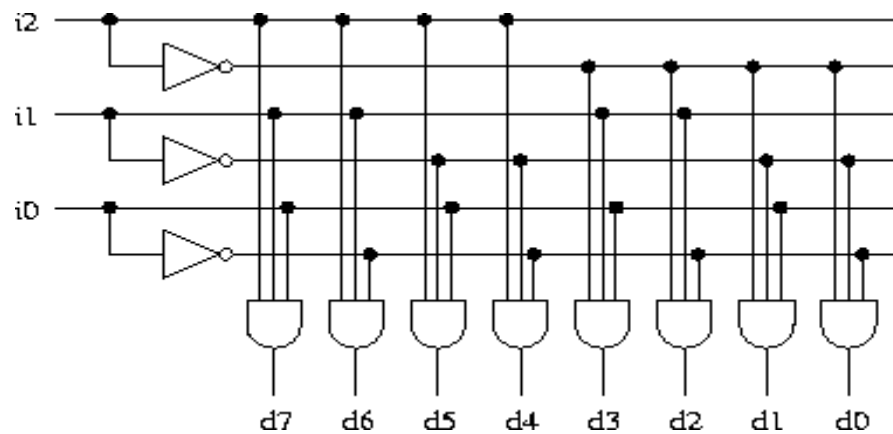


Truth Table

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	A ₂	A ₁	A ₀
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Decoder:

Logic Diagram




Truth Table

Ip2	Ip1	Ip0	Op7	Op6	Op5	Op4	Op3	Op2	Op1	Op0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Source code(s):

Encoder:

```
design.sv   
1 //23bce8268  
2 module encoder83(a0, a1, a2, a3, a4, a5, a6, a7, b0, b1, b2); // Declaration of  
  a module named encoder83 with inputs a0, a1, a2, a3, a4, a5, a6, a7, and outputs  
  b0, b1, b2.  
3 input a0, a1, a2, a3, a4, a5, a6, a7; // Declaration of inputs  
4 output b0, b1, b2; // Declaration of outputs  
5 assign b2= a4| a5| a6| a7; // assigns the value of b2 as the bitwise OR of  
  inputs a4, a5, a6, and a7.  
6 assign b1= a2| a3| a6| a7;  
7 assign b0= a1| a3| a5| a7;  
8 endmodule  
9
```

Decoder:

```
1 // 23bce8268  
2 module decoder38( i, D); // Defines a module names decoder38 with inputs i and  
  output D  
3 input [2 : 0] i; // Declares a 3-bit input port named i  
4 output [7 :0 ] D; // Declares a 8-bit output port named D  
5 reg [7 : 0 ] D; // Declares a 8-bit register named D inside the module  
6 always @(i) // This is a combinational logic block sensitive to changes in  
  the input i.  
7 begin  
8 case (i) // Starts a case statement based on the input i  
9  
10 3'b000:begin D=8'b00000001;end // When i is 000, sets D to 00000001 (8  
  bits).  
11 3'b001:begin D=8'b00000010;end // When i is 001, sets D to 00000010 (8  
  bits).  
12 3'b010:begin D=8'b00000100;end // When i is 010, sets D to 00000100 (8  
  bits).  
13 3'b011:begin D=8'b00001000;end // When i is 011, sets D to 00001000 (8  
  bits).  
14 3'b100:begin D=8'b00010000;end // When i is 100, sets D to 00010000 (8  
  bits).  
15 3'b101:begin D=8'b00100000;end // When i is 101, sets D to 00100000 (8  
  bits).  
16 3'b110:begin D=8'b01000000;end // When i is 110, sets D to 01000000 (8  
  bits).  
17 3'b111:begin D=8'b10000000;end // When i is 111, sets D to 10000000 (8  
  bits).  
18 endcase  
19 end  
20 endmodule // Ends the Verilog module decoder38  
22
```

Test Bench:

Encoder:

```
testbench.sv
1 //23bce8268
2 module tb_encoder83(); // declaration of testbench module named tb_encoder83.
3   reg a0, a1, a2, a3, a4, a5, a6, a7; // declaration of eight registers a0 to a7
   for the inputs.
4   wire b0, b1, b2, x; // This line declares wires b0, b1, b2, and x for the
   outputs and any additional internal signals.
5   encoder83 uut(a0, a1, a2, a3, a4, a5, a6, a7, b0, b1, b2); // This line
   instantiates an encoder83 module named uut with the declared inputs and outputs
6   initial // This block contains the initial simulation code that sets the inputs,
   dumps the waveform, and monitors the outputs.
7   begin
8     $dumpfile("test.vcd");
9     $dumpvars(0, uut);
10    $monitor("input=%b %b %b %b %b %b %b %b => output= %b %b %b", a7,a6, a5,
        a4, a3, a2, a1, a0, b2, b1, b0); // This line monitors the inputs a7 to a0 and
        outputs b2, b1, b0 and prints them whenever there is a change.
11
12    a7=0; a6=0; a5=0; a4=0; a3=0; a2=0; a1=0; a0=1; #20;
13    a7=0; a6=0; a5=0; a4=0; a3=0; a2=0; a1=1; a0=0; #20;
14    a7=0; a6=0; a5=0; a4=0; a3=0; a2=1; a1=0; a0=0; #20;
15    a7=0; a6=0; a5=0; a4=0; a3=1; a2=0; a1=0; a0=0; #20;
16    a7=0; a6=0; a5=0; a4=1; a3=0; a2=0; a1=0; a0=1; #20;
17    a7=0; a6=0; a5=1; a4=0; a3=0; a2=0; a1=0; a0=0; #20;
18    a7=0; a6=1; a5=0; a4=0; a3=0; a2=0; a1=0; a0=0; #20;
19    a7=1; a6=0; a5=0; a4=0; a3=0; a2=0; a1=0; a0=0; #20; // represent sequences
        of input values with a delay of 20 time units (#20). After each change in input
        values, the simulation waits for 20 time units before moving to the next input
        sequence.
20    end
21  endmodule
22
```

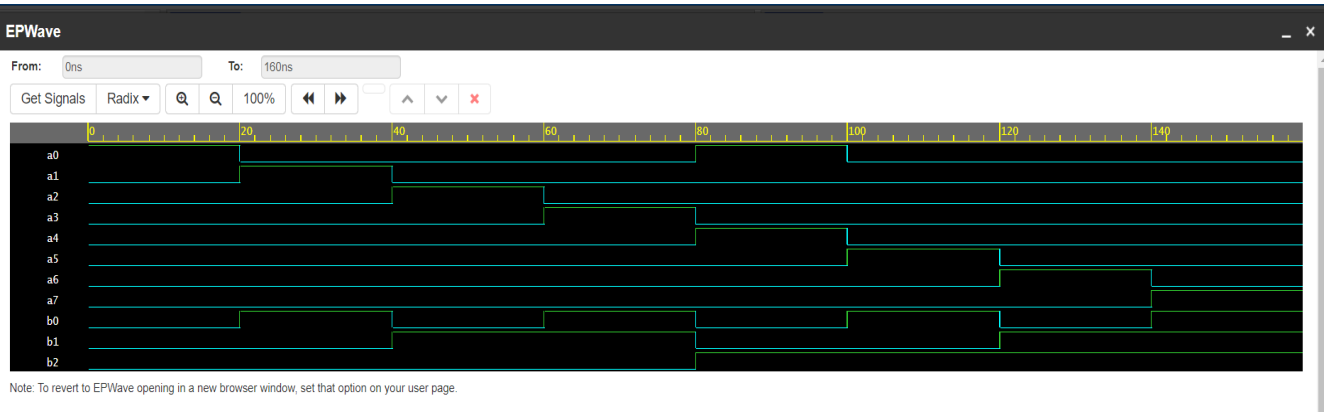
Decoder:

```
1 // 23bce8268
2 module tb(); // Declares a testbench module named "tb".
3   reg [2:0] i; // Declares a 3-bit register i for simulation.
4   wire [7:0] D; // Declares an 8-bit wire D for simulation.
5   decoder38 inst1(.i(i),.D(D)); // Instantiates the decoder38 module with
   connections between i and D.
6   initial begin // Initializes simulation behaviour.
7     $dumpfile("decoder_waveform.vcd"); // Specifies the name of the VCD (Value Change
   Dump) file for waveform viewing.
8     $dumpvars(1); // Dumps variables for waveform viewing
9
10    i=3'b000; #100; // Sets i to 000 and waits for 100 time units.
11    i=3'b001; #100; // do
12    i=3'b010; #100; // do
13    i=3'b011; #100; // do
14    i=3'b100; #100; // do
15    i=3'b101; #100; // do
16    i=3'b110; #100; // do
17    i=3'b111; #100; // do
18    $finish; // Ends the simulation.
19  end
20 endmodule
21
22
```

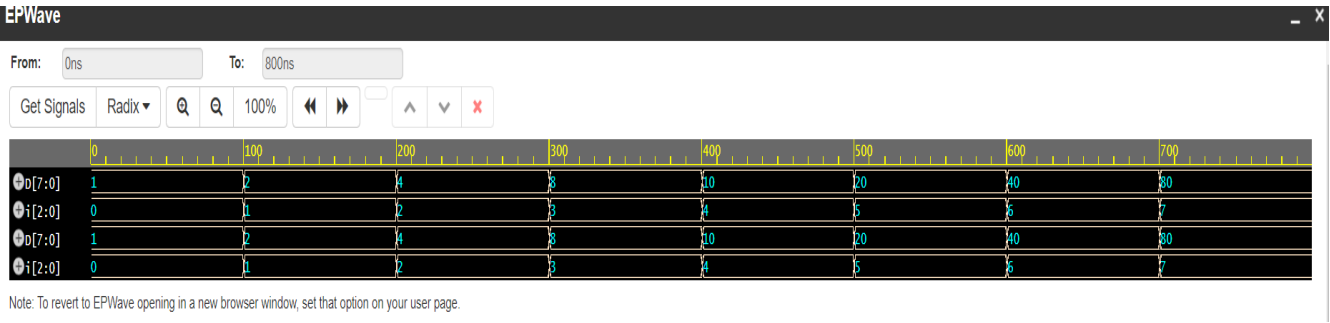
Result:

The simulation waveforms are obtained and verified with the expected waveforms.

Encoder:



Decoder:



Experiment 5

Title of the Experiment: To design 4:1 mux and 1:4 demux using Verilog code and compare with their respective truth tables.

Objective/Motivation: In this lab, a 4:1 mux and 1:4 demux are designed. The objective will be to test these designs on Xilinx simulation tool. The tests will be performed for all the possible combinations of inputs to verify their functionality. Moreover, the knowledge gained will be used to design complex designs.

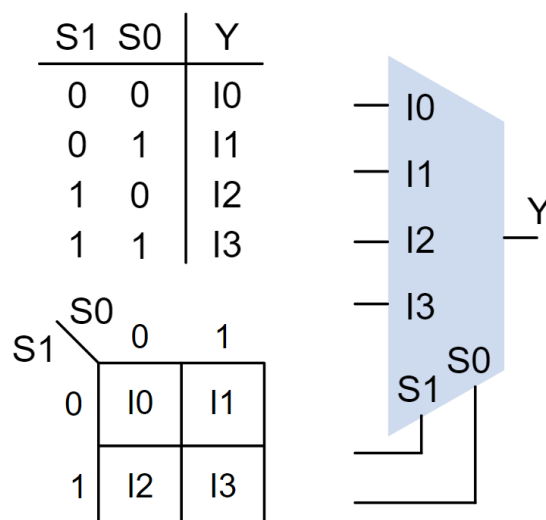
Equipment required:

1. Zybo board - Zynq XC7Z010
2. Xilinx Vivado Design Suite 2016

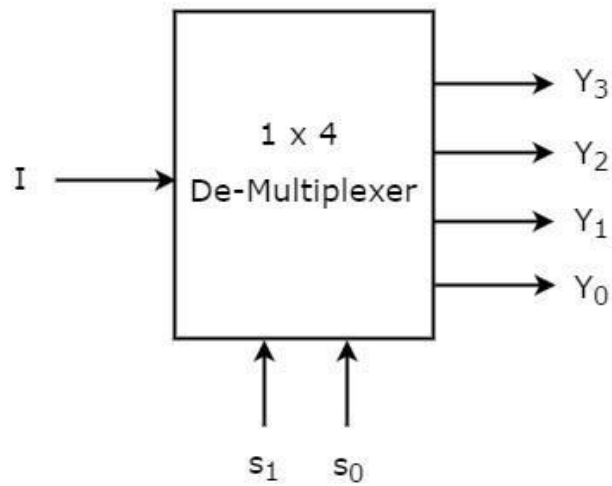
Logic diagram(s):

4:1 Multiplexer:

Truth Table and Logic Diagram



1:4 De-Multiplexer:



Selection Inputs		Outputs			
S ₁	S ₀	Y ₃	Y ₂	Y ₁	Y ₀
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

Source Code:

4:1 Multiplexer:

```
1 //23bce8268
2 module demux1_4 (
3     input I, s0, s1,
4     output y3, y2, y1, y0
5 );
6
7     assign y0 = I & (~s0) & (~s1);
8     assign y1 = I & (s0) & (~s1);
9     assign y2 = I & (~s0) & (s1);
10    assign y3 = I & (s0) & (s1);
11 endmodule
12 |
```

1:4 De-Multiplexer:

```
1 //23bce8268
2 module demux1_4 (
3     input I, s0, s1,
4     output y3, y2, y1, y0
5 );
6
7     assign y0 = I & (~s0) & (~s1);
8     assign y1 = I & (s0) & (~s1);
9     assign y2 = I & (~s0) & (s1);
10    assign y3 = I & (s0) & (s1);
11 endmodule
12
```

Test Bench:

4:1 Mux:

```
1 //23bce8268
2 module test_demux;
3     reg t_I, t_s0, t_s1;
4     wire t_y3, t_y2, t_y1, t_y0;
5
6
7     demux1_4 uut (
8         .I(t_I), .s0(t_s0), .s1(t_s1),
9         .y3(t_y3), .y2(t_y2), .y1(t_y1), .y0(t_y0)
10    );
11
12    initial begin
13        $dumpvars(1, test_demux);
14        t_s0 = 1'b0;
15        t_s1 = 1'b0;
16        t_I = 1'b1;
17        #10
18        t_s0 = 1'b1;
19        t_s1 = 1'b0;
20        t_I = 1'b1;
21        #10
22        t_s0 = 1'b0;
23        t_s1 = 1'b1;
24        t_I = 1'b1;
25        #10
26        t_s0 = 1'b1;
27        t_s1 = 1'b1;
28        t_I = 1'b1;
29        #10
30        $stop;
31    end
32 endmodule
33
```

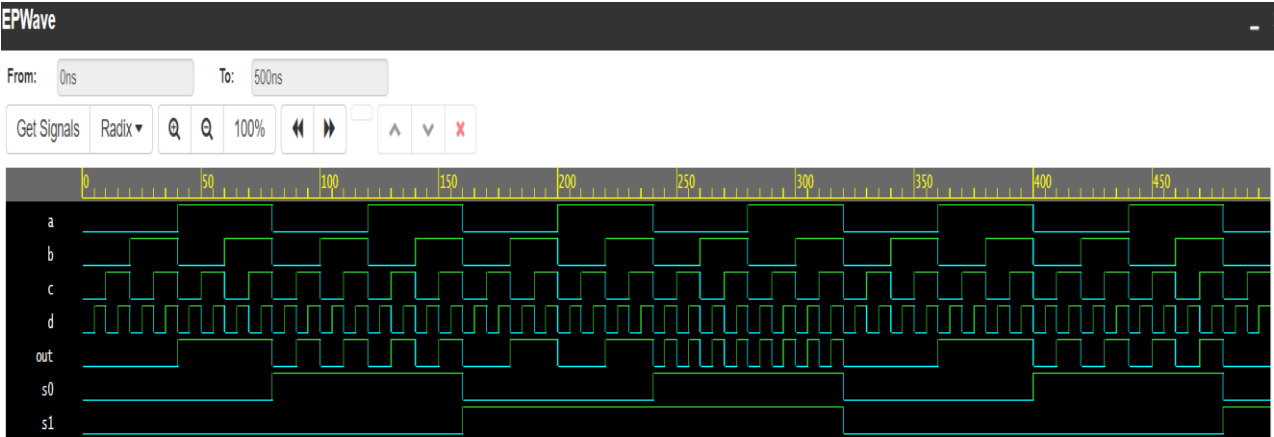

1:4 Demux:

```
1 //23bce8268
2 module test_demux;
3     reg t_I, t_s0, t_s1;
4     wire t_y3, t_y2, t_y1, t_y0;
5
6
7     demux1_4 uut (
8         .I(t_I), .s0(t_s0), .s1(t_s1),
9         .y3(t_y3), .y2(t_y2), .y1(t_y1), .y0(t_y0)
10    );
11
12    initial begin
13        $dumpvars(1, test_demux);
14        t_s0 = 1'b0;
15        t_s1 = 1'b0;
16        t_I = 1'b1;
17        #10
18        t_s0 = 1'b1;
19        t_s1 = 1'b0;
20        t_I = 1'b1;
21        #10
22        t_s0 = 1'b0;
23        t_s1 = 1'b1;
24        t_I = 1'b1;
25        #10
26        t_s0 = 1'b1;
27        t_s1 = 1'b1;
28        t_I = 1'b1;
29        #10
30        $stop;
31    end
32 endmodule
```

Result:

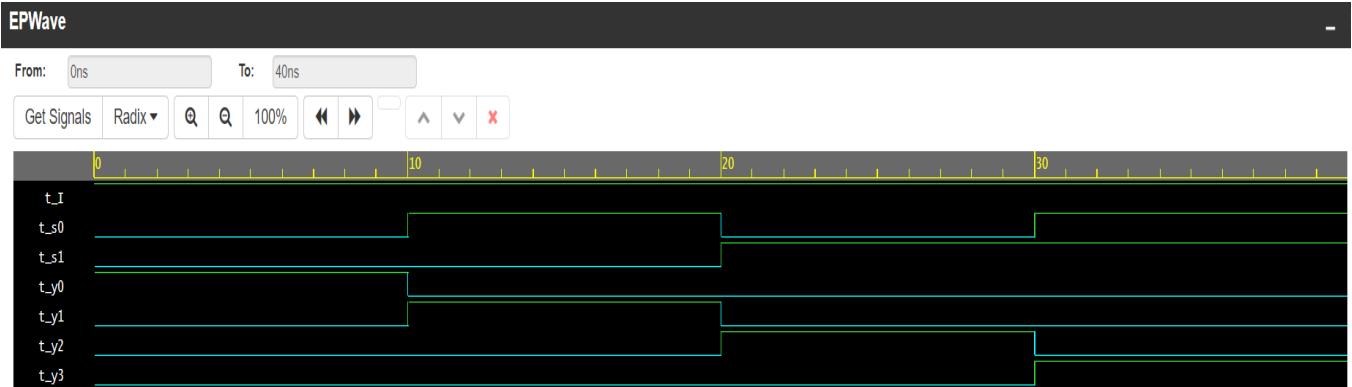
The simulation waveforms are obtained and verified with the expected waveforms.

4:1 Mux:



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

1:4 Demux:



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

Experiment- 6

Title of the Experiment: To design 4-bit magnitude comparator using Verilog code and compare with their respective truth tables.

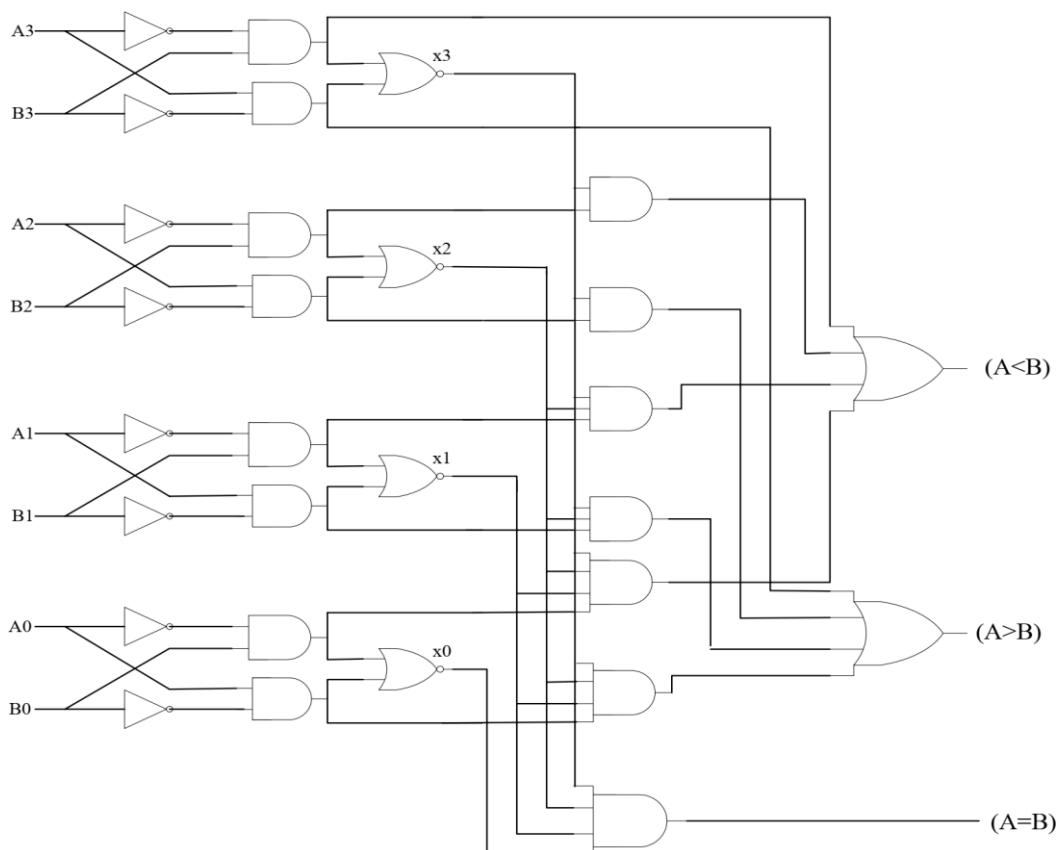
Objective/Motivation: In this lab, a 4-bit magnitude comparator is designed. The objective will be to test this design on Xilinx simulation tool. The tests will be performed for all the possible combinations of inputs to verify its functionality. Moreover, the knowledge gained will be used to design multiple bit comparators.

Equipment required:

1. Zybo board - Zynq XC7Z010
2. Xilinx Vivado Design Suite 2016

Logic diagram(s):

4-bit Comparator



Truth Table

COMPARING INPUTS				OUTPUT		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B
A3 > B3	X	X	X	H	L	L
A3 < B3	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	H	L	L
A3 = B3	A2 < B2	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H
H = High Voltage Level, L = Low Voltage Level, X = Don't Care						

Source Code:

```

1 //23bce8268
2 module comparator_new(
3     input a0, a1, b0, b1,
4     output reg Gt_o, Eq_o, Lt_o
5 ); // definition of module named comparator and declaration of i/ps and o/ps.
6
7     always @* begin // sensitive to all input variations (@*)
8 Gt_o = ~(a1 & ~b1 | ~a1 & b1) & a0 & ~b0 | a1 & ~b1; // this line calculates
whether the binary number a is greater than b and assigns the result to Gt_o. If
a > b then Gt_o is set to true i.e 1 otherwise its false.
9 Eq_o = ~(a0 & ~b0 | ~a0 & b0) & ~(a1 & ~b1 | ~a1 & b1); //this line of code
calculate the equality between 2 pairs of bits and assigns the result to Eq_o. If
both pairs are equal then Eq_o is set to true; otherwise its false
10 Lt_o = ~(a1 & ~b1 | ~a1 & b1) & ~a0 & b0 | ~a1 & b1; // this line calculates
whether the binary number a is less than b and assigns the result to Lt_o. If a <
b then Lt_o is set to true i.e 1 otherwise its false.
11 end
12 endmodule
13
14
15

```

Test Bench:

```

1 //23bce8268
2 module tb_comparator_struct;
3     reg a0, a1, b0, b1;
4     wire Gt_o, Eq_o, Lt_o;
5     comparator_new tb(
6         .a0(a0), .a1(a1),
7         .b0(b0), .b1(b1),
8         .Gt_o(Gt_o), .Eq_o(Eq_o), .Lt_o(Lt_o)
9     );
10
11     initial begin
12         $dumpfile("comp_waveform.vcd");
13         $dumpvars(1);
14     end
15
16     initial begin
17         $display("a1a0=%b%b, b1b0=%b%b,", a0, a1, b1, b0);
18         a1 = 1'b0; a0 = 1'b0; b1 = 1'b0; b0 = 1'b0; #2;
19         a1 = 1'b0; a0 = 1'b0; b1 = 1'b0; b0 = 1'b1; #2;
20         a1 = 1'b0; a0 = 1'b0; b1 = 1'b1; b0 = 1'b0; #2;
21         a1 = 1'b0; a0 = 1'b0; b1 = 1'b1; b0 = 1'b1; #2;
22         a1 = 1'b0; a0 = 1'b1; b1 = 1'b0; b0 = 1'b0; #2;
23         a1 = 1'b0; a0 = 1'b1; b1 = 1'b0; b0 = 1'b1; #2;
24         a1 = 1'b0; a0 = 1'b1; b1 = 1'b1; b0 = 1'b0; #2;
25         a1 = 1'b0; a0 = 1'b1; b1 = 1'b1; b0 = 1'b1; #2;
26         a1 = 1'b1; a0 = 1'b0; b1 = 1'b0; b0 = 1'b0; #2;
27         a1 = 1'b1; a0 = 1'b0; b1 = 1'b0; b0 = 1'b1; #2;
28         a1 = 1'b1; a0 = 1'b0; b1 = 1'b1; b0 = 1'b0; #2;
29         a1 = 1'b1; a0 = 1'b0; b1 = 1'b1; b0 = 1'b1; #2;
30         a1 = 1'b1; a0 = 1'b1; b1 = 1'b0; b0 = 1'b0; #2;
31         a1 = 1'b1; a0 = 1'b1; b1 = 1'b0; b0 = 1'b1; #2;
32         a1 = 1'b1; a0 = 1'b1; b1 = 1'b1; b0 = 1'b0; #2;
33         a1 = 1'b1; a0 = 1'b1; b1 = 1'b1; b0 = 1'b1; #2;
34         $finish;
35     end
36 endmodule
37
38

```

Result:

The simulation waveforms are obtained and verified with the expected waveforms

