CISC 530 – Computing System Architecture

Rishabh Agarwal

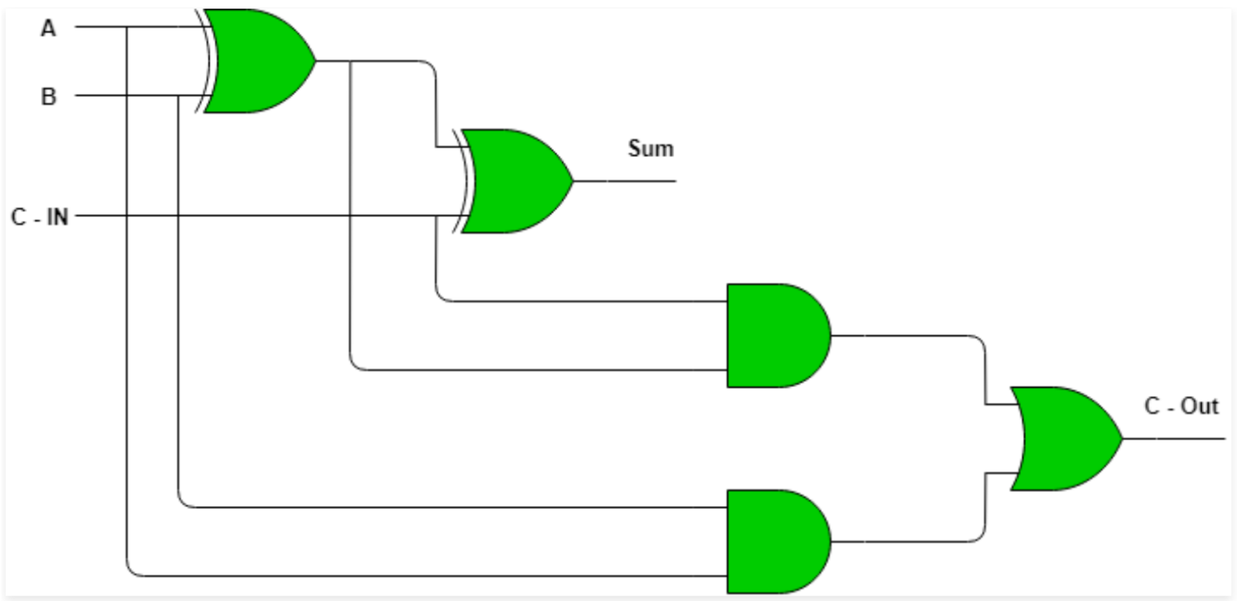
Harrisburg University of Science and Technology

ASSIGNMENT 1

Q1. A n bit full adder without carry out and overflow signals i-e for signed numbers and n-bit full adder with carry out and overflow signals i-e for unsigned numbers.

Ans. When we have N bit full adder with Co (carry out) value as 0, it behave as a N bit full adder without carry out and overflow signal.

Diagram:



Full Adder logic circuit.

Truth Table:

A	B	Cin	So	Co
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

Verilog Code:

```
module fulladder(
    input a,
    input b,
input cin,
    output c0,
  output s0
    );

assign s0=a^b^cin;
assign c0=(a&b)|(a&cin)|(b&cin);
endmodule
```

Below the code has been implemented with n value as 4. If the value of n changes input A and B will change ([n-1:0]A or [n-1:0]B) and wire will be always n-2 i.e wire[n-2:0]C. Stage are always same as the value of n. If the n is 6 we will have 6 stages.

The example below has been implement with n value as 4.

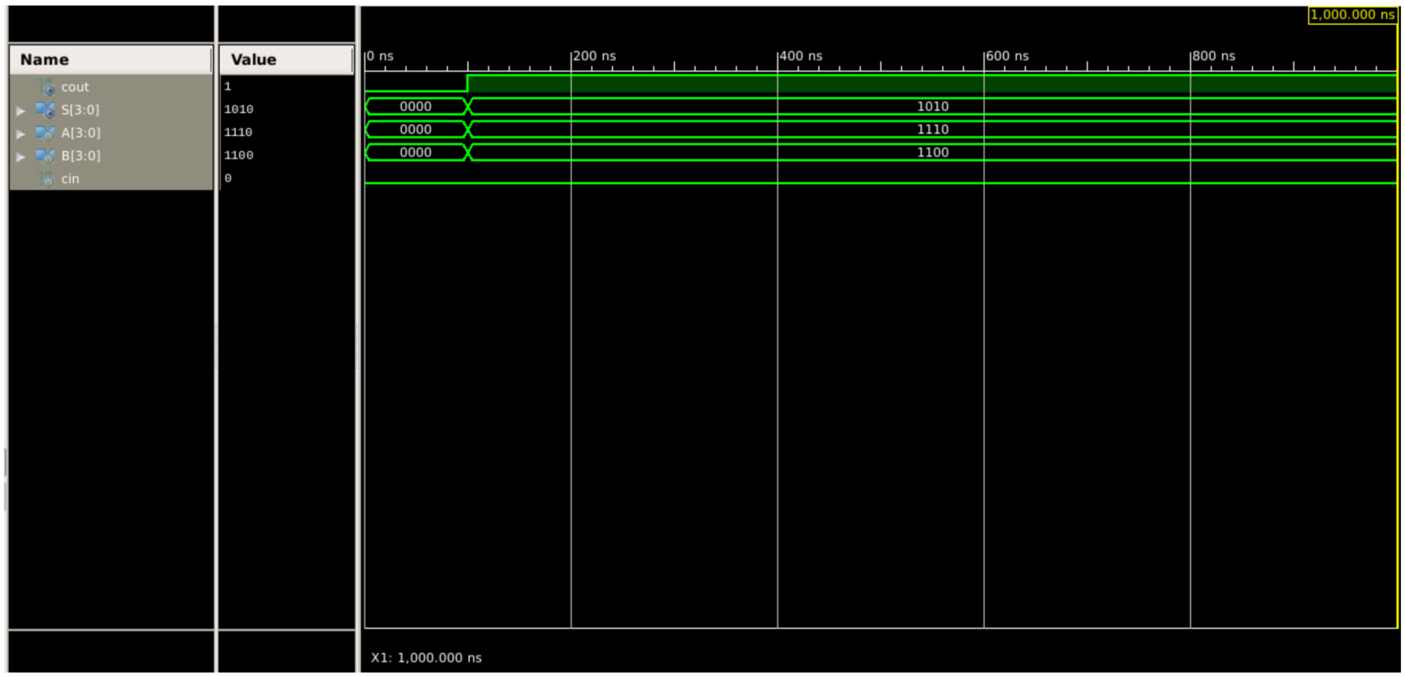
```
module fulladder4bit(
    input [3:0]A,
    input [3:0]B,
input cin,
output cout,
    output [3:0]S
    );
//parameter n = 4; this is used to specify value of n
wire[2:0]C;
//for loop can be also used to make it dynamic
fulladder stage0(A[0],B[0],cin,C[0],S[0]);
fulladder stage1(A[1],B[1],C[0],C[1],S[1]);
fulladder stage2(A[2],B[2],C[1],C[2],S[2]);
fulladder stage3(A[3],B[3],C[2],cout,S[3]);
endmodule
```

Testbench:

```
module fulladder4bittest;

// Inputs
```

```
reg [3:0] A;
reg [3:0] B;
reg cin;
// Outputs
wire cout;
wire [3:0] S;
// Instantiate the Unit Under Test (UUT)
fulladder4bit uut (
.A(A),
.B(B),
.cin(cin),
.cout(cout),
.S(S)
);
initial begin
// Initialize Inputs
A = 0;
B = 0;
cin = 0;
// Wait 100 ns for global reset to finish
#100;
A[0]=0;A[1]=1;A[2]=1;A[3]=1;
B[0]=0;B[1]=0;B[2]=1;B[3]=1;
cin = 0;
// Add stimulus here
end
endmodule
```

Output:



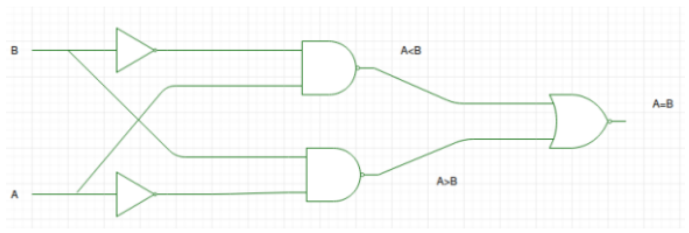
```
cout     1
S[3:0]   1010    0000 | 1010
A[3:0]   1110    0000 | 1110
B[3:0]   1100    0000 | 1100
cin      0
```

Q2. Describe with the help of truth table and logic circuit four bit comparator and design its Verilog code and also explain the importance of this circuit in computer system.

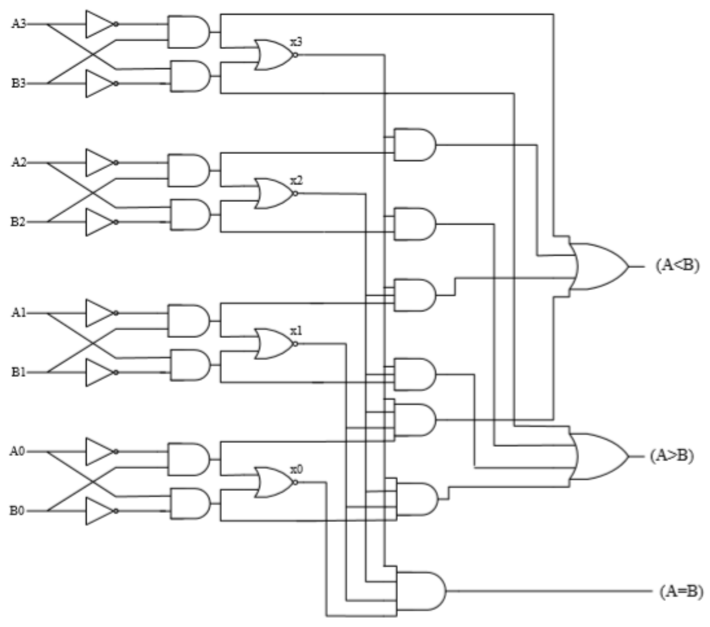
Ans. A comparator is used to compare digital or binary number in order to find out whether one binary number is equal, less than or greater than the other binary number.

Diagram:

Single Bit Comparator



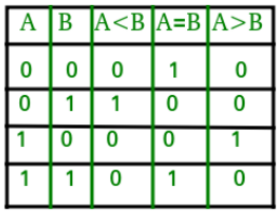
Four Bit Comparator



4-Bit Magnitude Comparator

Truth Table:

1 bit comparator,



A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

4 bit comparator,

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						

Verilog Code:

We will design a single bit comparator and then call that method 4 times using wires.

```
module comparator1bit(
    input A,
    input B,
    output reg AlesserB,
output reg AequalB,
    output reg AlargerB
    );
always@(A,B)
begin
if(A>B)
begin
AlesserB = 0;
AequalB = 0;
AlargerB = 1;
end
else if(A<B)
begin
AlesserB = 1;
AequalB = 0;
AlargerB = 0;
end
else
begin
AlesserB = 0;
AequalB = 1;
AlargerB = 0;
end
end
endmodule
```

```
module comparator4bit(
input [3:0]A,
input [3:0]B,
output reg AlesserB,
output reg AequalB,
output reg AlargerB
    );
always@(A,B)
begin
if(A>B)
begin
AlesserB = 0;
AequalB = 0;
```

```
AlargerB = 1;
end
else if(A<B)
begin
AlesserB = 1;
AequalB = 0;
AlargerB = 0;
end
else
begin
AlesserB = 0;
AequalB = 1;
AlargerB = 0;
end
end
endmodule
```

Testbench:

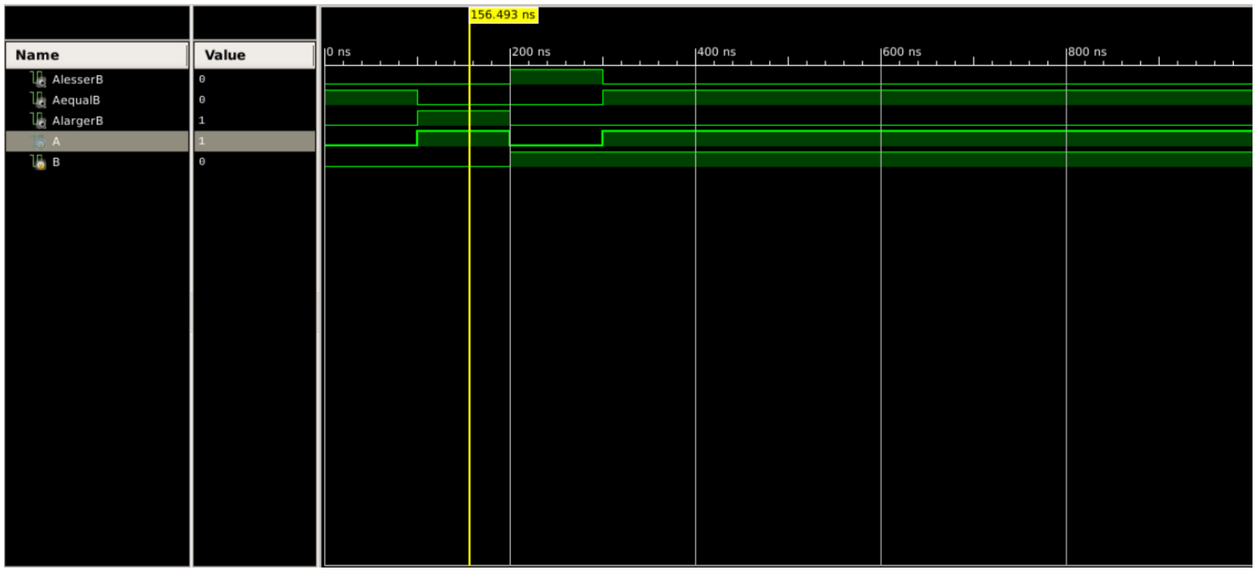
```
module comparator1bittest;
// Inputs
reg A;
reg B;
// Outputs
wire AlesserB;
wire AequalB;
wire AlargerB;
// Instantiate the Unit Under Test (UUT)
comparator1bit uut (
.A(A),
.B(B),
.AlesserB(AlesserB),
.AequalB(AequalB),
.AlargerB(AlargerB)
);
initial begin
// Initialize Inputs
A = 0;
B = 0;
// Wait 100 ns for global reset to finish
#100;
A = 1;
B = 0;
#100;
A = 0;
```

```
B = 1;
#100;
A = 1;
B = 1;
// Add stimulus here
end
endmodule
module comparator4bittest;

        // Inputs
        reg [3:0] A;
        reg [3:0] B;

        // Outputs
        wire AlesserB;
        wire AequalB;
        wire AlargerB;

        // Instantiate the Unit Under Test (UUT)
        comparator4bit uut (
                .A(A),
                .B(B),
                .AlesserB(AlesserB),
                .AequalB(AequalB),
                .AlargerB(AlargerB)
        );

        initial begin
                // Initialize Inputs
                //equal
                A = 0;
                B = 0;

                // Wait 100 ns for global reset to finish
                //greater
                #100;
                A=4'b1111;
                B=4'b0000;

                //less
                #100
                A=4'b0000;
                B=4'b1111;

                //less
```

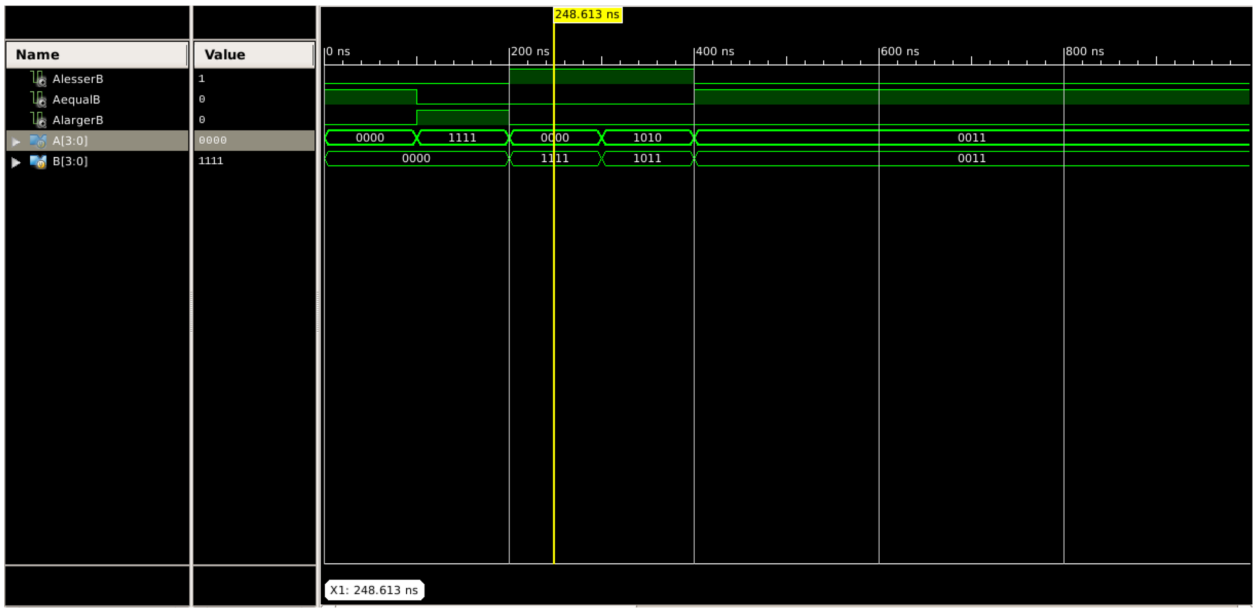
```
                        #100;
                        A=4'b1010;
                        B=4'b1011;

                        //equal
                        #100;
                        A=4'b0011;
                        B=4'b0011;
                        // Add stimulus here

            end

endmodule
```

Output:

1 bit Output



4 bit Output

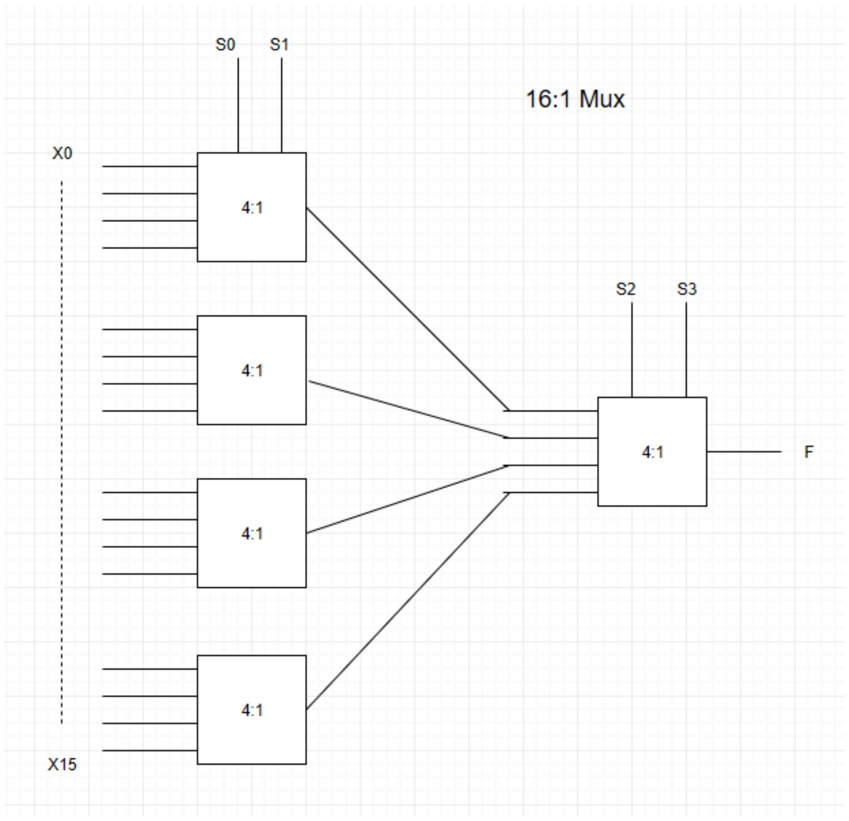


Q3. Describe the importance of multiplexers in computer system (where are they used) and the design the logic circuit and Verilog code for 16:1 multiplexer.

Ans. Multiplexer are used to process multiple input signals to single output signal or "combine into one signal" over a shared medium. Multiplexer are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth. It is being used in telecommunication to reduce the number of electrical connections/wireless channels for transmission of several signals. This helps in making it cost effective.

<u>**16:1 multiplexer**</u>

Diagram:



Truth Table:

S3	S2	S1	S0	Output
0	0	0	0	X0
0	0	0	1	X1
0	0	1	0	X2
0	0	1	1	X3
0	1	0	0	X4
0	1	0	1	X5
0	1	1	0	X6
0	1	1	1	X7
1	0	0	0	X8
1	0	0	1	X9
1	0	1	0	X10
1	0	1	1	X11
1	1	0	0	X12
1	1	0	1	X13
1	1	1	0	X14
1	1	1	1	X15

Verilog Code:

Just like the diagram here we create a 4:1 mux module and then that module will be called by another module which internally calls it four time.

```
module mux4to1(
   input [0:3]X,
   input [1:0]S,
   output reg f
   );

always@(X,S)
if(S==0)
f=X[0];
else if (S==1)
f=X[1];
else if (S==2)
f=X[2];
else
```

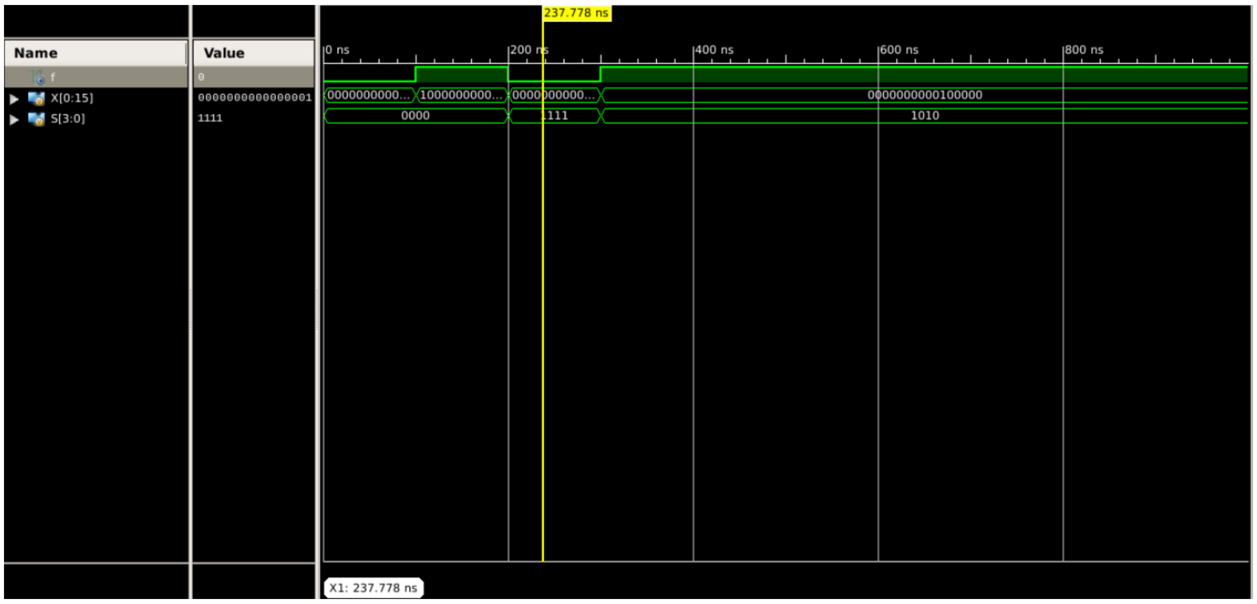
```
f=X[3];

endmodule
module mux16to1(
input [0:15]X,
input [3:0]S,
output f
   );
wire[0:3]M;

mux4to1 Mux1(X[0:3],S[1:0],M[0]);
mux4to1 Mux2(X[4:7],S[1:0],M[1]);
mux4to1 Mux3(X[8:11],S[1:0],M[2]);
mux4to1 Mux4(X[12:16],S[1:0],M[3]);
mux4to1 Mux5(M[0:3],S[3:2],f);

endmodule
```

Testbench:

```
module mux16to1test;
// Inputs
reg [0:15] X;
reg [3:0] S;
// Outputs
wire f;
// Instantiate the Unit Under Test (UUT)
mux16to1 uut (
.X(X),
.S(S),
.f(f)
);
initial begin
// Initialize Inputs
X = 0;
S = 0;

// Wait 100 ns for global reset to finish
//for 0000->x0
#100;
     S[0]=0;S[1]=0;S[2]=0;S[3]=0;
     X[0]=1;
//for 0000->x15
```

```
#100;
   S[0]=1;S[1]=1;S[2]=1;S[3]=1;
   X[0]=0;X[15]=1;
//for 1010->x10
#100;
   S[0]=0;S[1]=1;S[2]=0;S[3]=1;
   X[0]=0;X[15]=0;;X[10]=1;
// Add stimulus here
end
endmodule
```

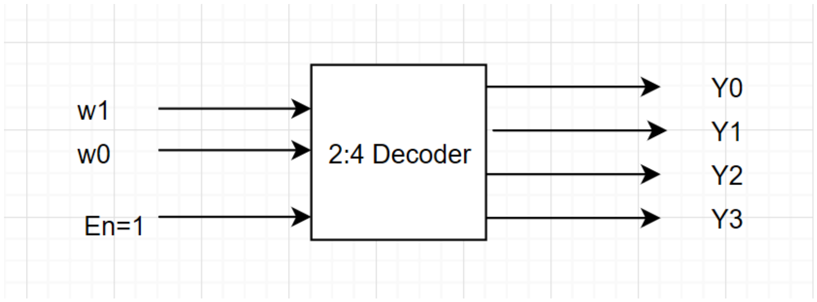
Output:
1 bit comparator output,



Q4. Design the Verilog code for 2 to 4 binary decoder with the help of case statements as well as if -else statements.

2 to 4 Binary Decoder (using case statement)

Diagram:



Truth Table:

En	W1	W0	Y0	Y1	Y2	Y3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Verilog Code:

With case Statement:

```
module dec2to4(
   input [1:0]W,
   input En,
   output reg [0:3]Y
   );
always@(W,En)
case ({En,W})
3'b100: Y=4'b1000;
3'b101: Y=4'b0100;
3'b110: Y=4'b0010;
3'b111: Y=4'b0001;
default: Y=4'b0000;
endcase
endmodule
```

With If -else statement:

```
module dec2to4(
input [1:0]W,
input En,
output reg [0:3]Y
);
always@(W,En)
begin
if(En==1'b1)begin
if(W==2'b00)
Y=4'b1000;
else if(W==2'b01)
Y=4'b0100;
else if(W==2'b10)
Y=4'b0010;
else if(W==2'b11)
Y=4'b0001;
end
else
Y=4'b0000;
end
endmodule
```

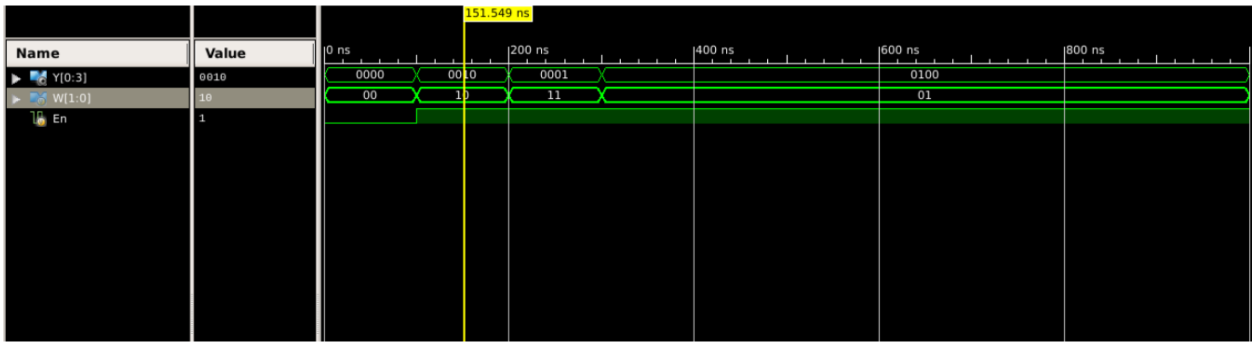
Testbench:

```
module dec2to4test;
// Inputs
reg [1:0] W;
reg En;
// Outputs
wire [0:3] Y;
// Instantiate the Unit Under Test (UUT)
dec2to4 uut (
.W(W),
.En(En),
.Y(Y)
);
initial begin
// Initialize Inputs
W = 0;
En = 0;
```

```
// Wait 100 ns for global reset to finish
#100;
    En=1;
W[0]=0;
W[1]=1;

#100;
    En=1;
W[0]=1;
W[1]=1;
#100;
    En=1;
W[0]=1;
W[1]=0;
// Add stimulus here
end
endmodule
```

Output:

Output for both code with case and if else was same



REFERENCES

https://en.m.wikipedia.org/wiki/Multiplexer

https://www.geeksforgeeks.org/magnitude-comparator/

http://users.encs.concordia.ca/~asim/COEN_6511/Projects/final6511report.pdf

https://www.fpga4student.com/2017/07/n-bit-adder-design-in-verilog.html

https://www.geeksforgeeks.org/full-adder-digital-electronics/