EC204

Digital System Design Lab

Lab – 4

Utkarsh R Mahajan

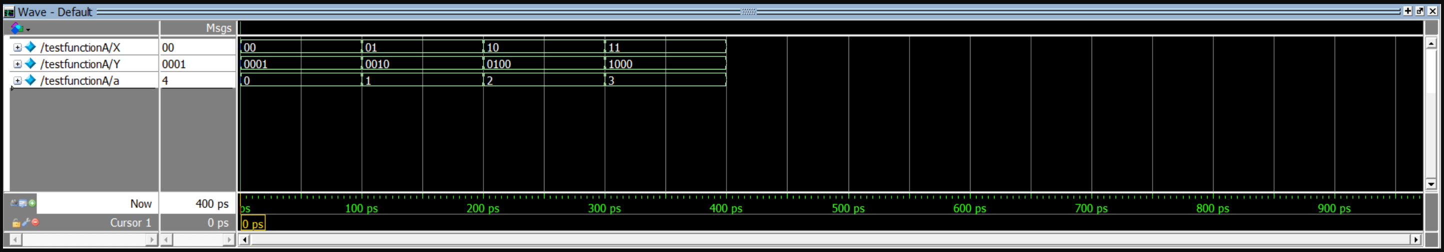
201EC164

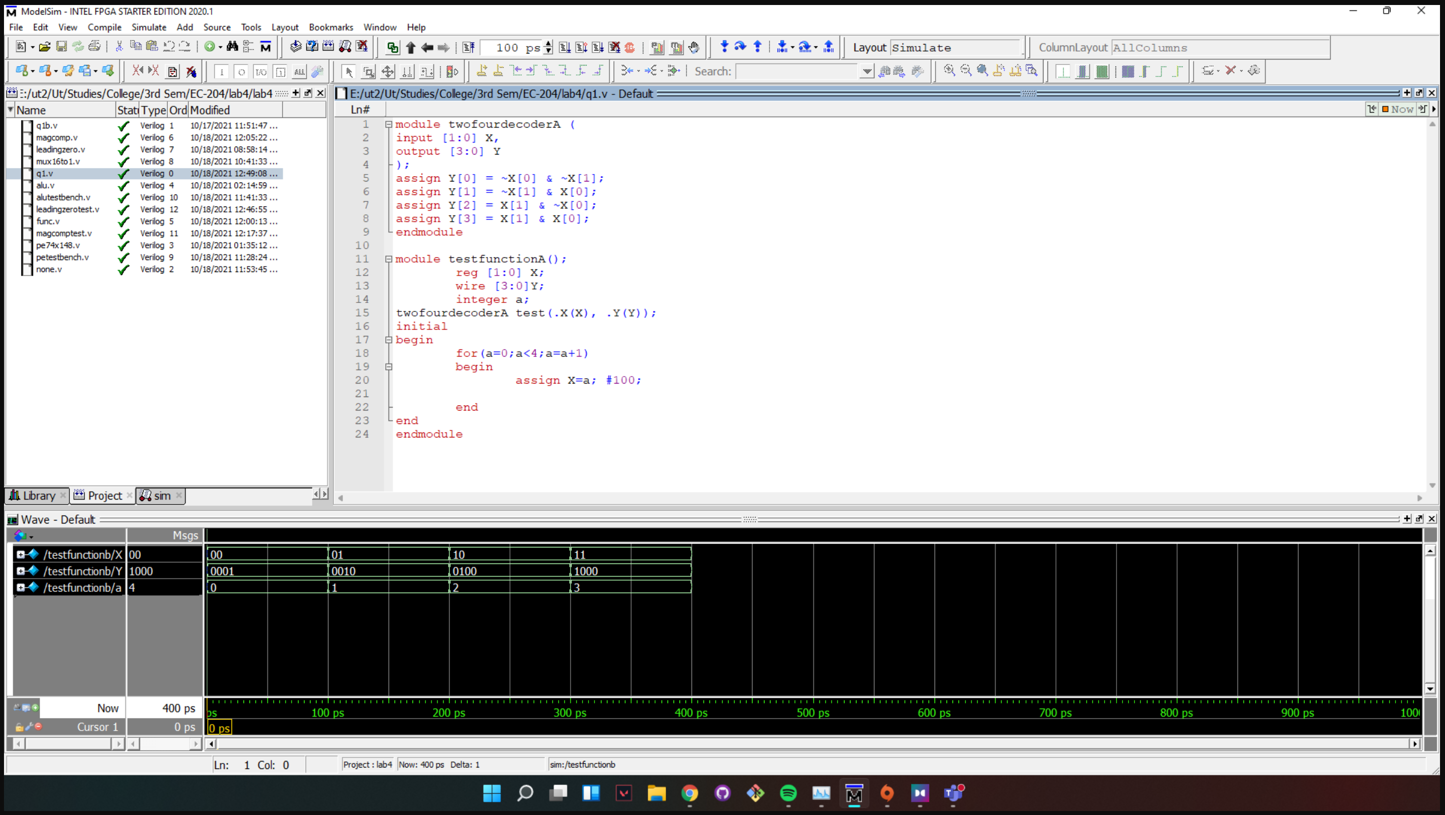
1] 2 to 4 decoder using

(a) concurrent signal assignment statements

```
module twofourdecoderA (
input [1:0] X,
output [3:0] Y
);
assign Y[0] = ~X[0] & ~X[1];
assign Y[1] = ~X[1] & X[0];
assign Y[2] = X[1] & ~X[0];
assign Y[3] = X[1] & X[0];
endmodule

module testfunctionA();
    reg [1:0] X;
    wire [3:0]Y;
    integer a;
twofourdecoderA test(.X(X), .Y(Y));
initial
begin
    for(a=0;a<4;a=a+1)
    begin
        assign X=a; #100;

    end
end
endmodule
```

Wave:





ModelSim - INTEL FPGA STARTER EDITION 2020.1

E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/lab4

Name	Stat	Type	Ord	Modified
q1b.v	✓	Verilog	1	10/17/2021 11:51:47 ...
magcomp.v	✓	Verilog	6	10/18/2021 12:05:22 ...
leadingzero.v	✓	Verilog	7	10/18/2021 08:58:14 ...
mux16to1.v	✓	Verilog	8	10/18/2021 10:41:33 ...
q1.v	✓	Verilog	0	10/18/2021 12:49:08 ...
alu.v	✓	Verilog	4	10/18/2021 02:14:59 ...
alutestbench.v	✓	Verilog	10	10/18/2021 11:41:33 ...
leadingzerotest.v	✓	Verilog	12	10/18/2021 12:46:55 ...
func.v	✓	Verilog	5	10/18/2021 12:00:13 ...
magcomptest.v	✓	Verilog	11	10/18/2021 12:17:37 ...
pe74x148.v	✓	Verilog	3	10/18/2021 01:35:12 ...
petestbench.v	✓	Verilog	9	10/18/2021 11:28:24 ...
none.v	✓	Verilog	2	10/18/2021 11:53:45 ...

E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/q1.v - Default

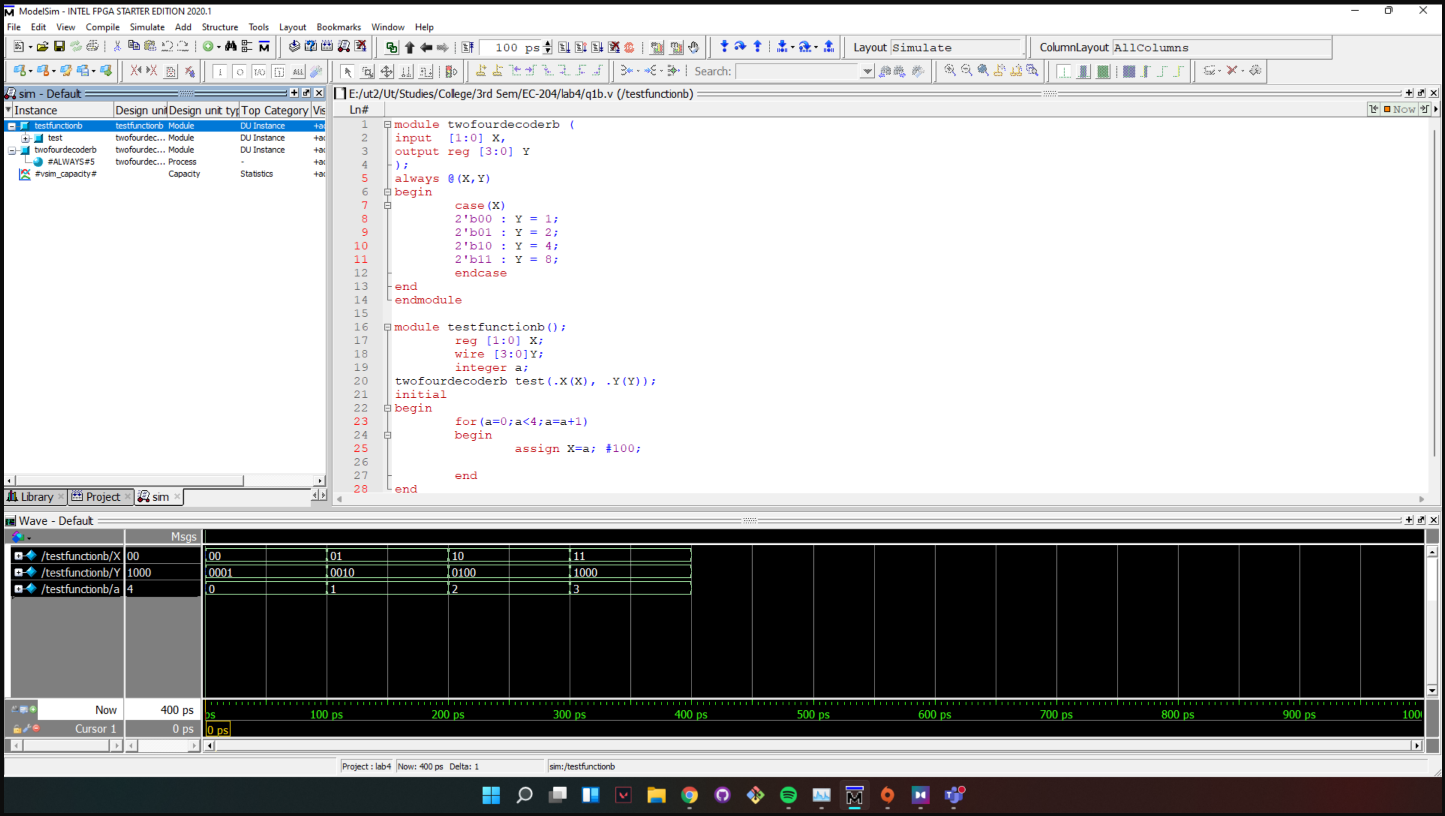
```verilog
module twofourdecoderA (
input [1:0] X,
output [3:0] Y
);
assign Y[0] = ~X[0] & ~X[1];
assign Y[1] = ~X[1] & X[0];
assign Y[2] = X[1] & ~X[0];
assign Y[3] = X[1] & X[0];
endmodule

module testfunctionA();
        reg [1:0] X;
        wire [3:0]Y;
        integer a;
twofourdecoderA test(.X(X), .Y(Y));
initial
begin
        for(a=0;a<4;a=a+1)
        begin
                assign X=a; #100;

        end
end
endmodule
```

Wave - Default

```
                     Msgs
/testfunctionb/X     00     00    | 01    | 10    | 11
/testfunctionb/Y     1000   0001  | 0010  | 0100  | 1000
/testfunctionb/a     4      0     | 1     | 2     | 3

Now       400 ps   ps   100 ps   200 ps   300 ps   400 ps   500 ps   600 ps   700 ps   800 ps   900 ps   100
Cursor 1  0 ps     0 ps
```

Ln: 1 Col: 0 | Project : lab4 | Now: 400 ps Delta: 1 | sim:/testfunctionb

(b) using case statement



ModelSim - INTEL FPGA STARTER EDITION 2020.1

sim - Default

Instance	Design unit	Design unit type	Top Category	Vis
testfunctionb	testfunctionb	Module	DU Instance	+ac
test	twofourdec...	Module	DU Instance	+ac
twofourdecoderb	twofourdec...	Module	DU Instance	+ac
#ALWAYS#5	twofourdec...	Process	-	+ac
#vsim_capacity#		Capacity	Statistics	+ac

E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/q1b.v (/testfunctionb)

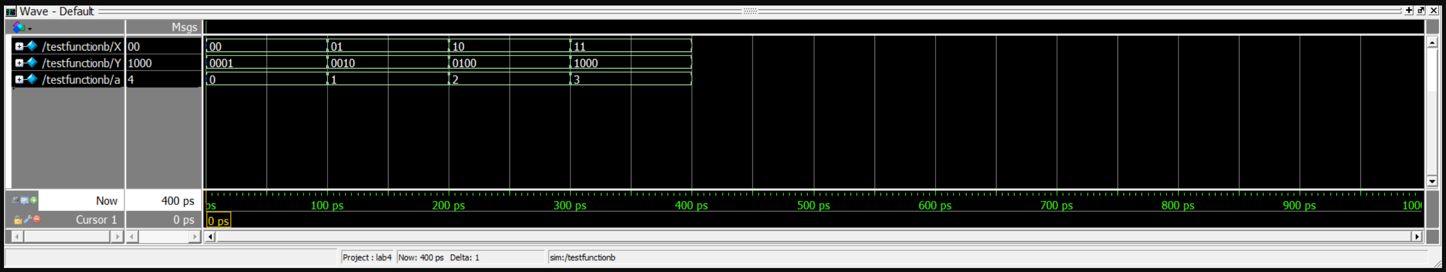
```verilog
module twofourdecoderb (
input [1:0] X,
output reg [3:0] Y
);
always @(X,Y)
begin
        case(X)
        2'b00 : Y = 1;
        2'b01 : Y = 2;
        2'b10 : Y = 4;
        2'b11 : Y = 8;
        endcase
end
endmodule

module testfunctionb();
        reg [1:0] X;
        wire [3:0]Y;
        integer a;
twofourdecoderb test(.X(X), .Y(Y));
initial
begin
        for(a=0;a<4;a=a+1)
        begin
                assign X=a; #100;

        end
end
```

Wave - Default

```
                     Msgs
/testfunctionb/X     00     00    | 01    | 10    | 11
/testfunctionb/Y     1000   0001  | 0010  | 0100  | 1000
/testfunctionb/a     4      0     | 1     | 2     | 3

Now       400 ps   ps   100 ps   200 ps   300 ps   400 ps   500 ps   600 ps   700 ps   800 ps   900 ps   100
Cursor 1  0 ps     0 ps
```

Project : lab4 | Now: 400 ps Delta: 1 | sim:/testfunctionb

Wave:

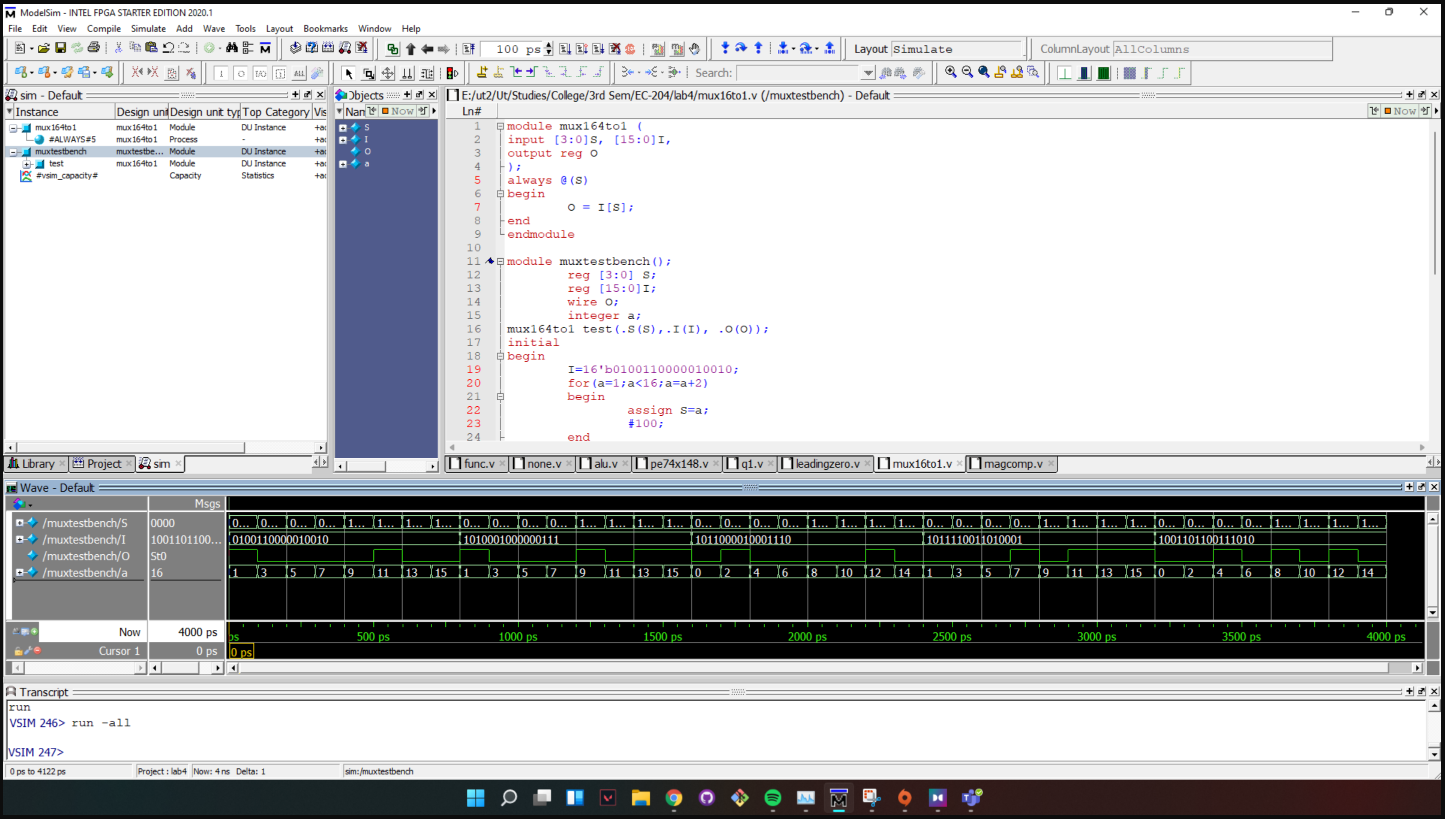


```
module twofourdecoderb (
input  [1:0] X,
output reg [3:0] Y
);
always @(X,Y)
begin
    case(X)
    2'b00 : Y = 1;
    2'b01 : Y = 2;
    2'b10 : Y = 4;
    2'b11 : Y = 8;
    endcase
end
endmodule

module testfunctionb();
    reg [1:0] X;
    wire [3:0]Y;
    integer a;
twofourdecoderb test(.X(X), .Y(Y));
initial
begin
    for(a=0;a<4;a=a+1)
    begin
        assign X=a; #100;

    end
end
endmodule
```

2] 16:1 multiplexer

```verilog
module muxtestbench();
    reg [3:0] S;
    reg [15:0]I;
    wire O;
    integer a;
mux164to1 test(.S(S),.I(I), .O(O));
initial
begin
    I=16'b0100110000010010;
    for(a=1;a<16;a=a+2)
    begin
        assign S=a;
        #100;
    end
    I=16'b1010001000000111;
    for(a=1;a<16;a=a+2)
    begin
        assign S=a;
        #100;
    end
    I=16'b1011000010001110;
    for(a=0;a<16;a=a+2)
    begin
        assign S=a;
        #100;
    end
    I=16'b1011100110100001;
    for(a=1;a<16;a=a+2)
    begin
        assign S=a;
        #100;
    end
    I=16'b1001101100111010;
    for(a=0;a<16;a=a+2)
    begin
        assign S=a;
        #100;
    end
end
endmodule
```

```verilog
module mux164to1 (
input [3:0]S, [15:0]I,
output reg O
);
always @(S)
begin
    O = I[S];
end
endmodule
```

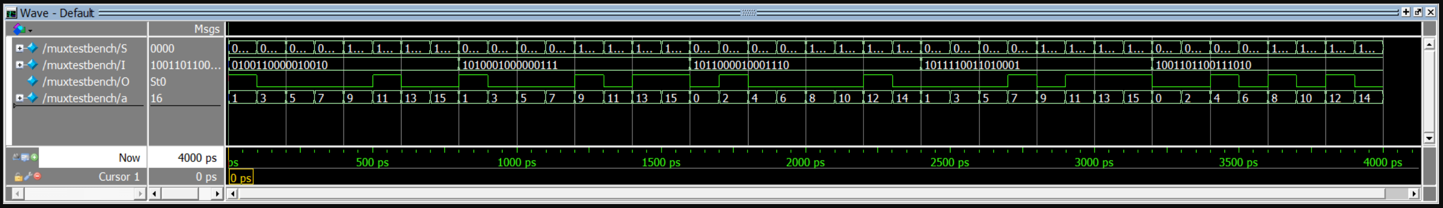


E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/mux16to1.v (/muxtestbench) - Default

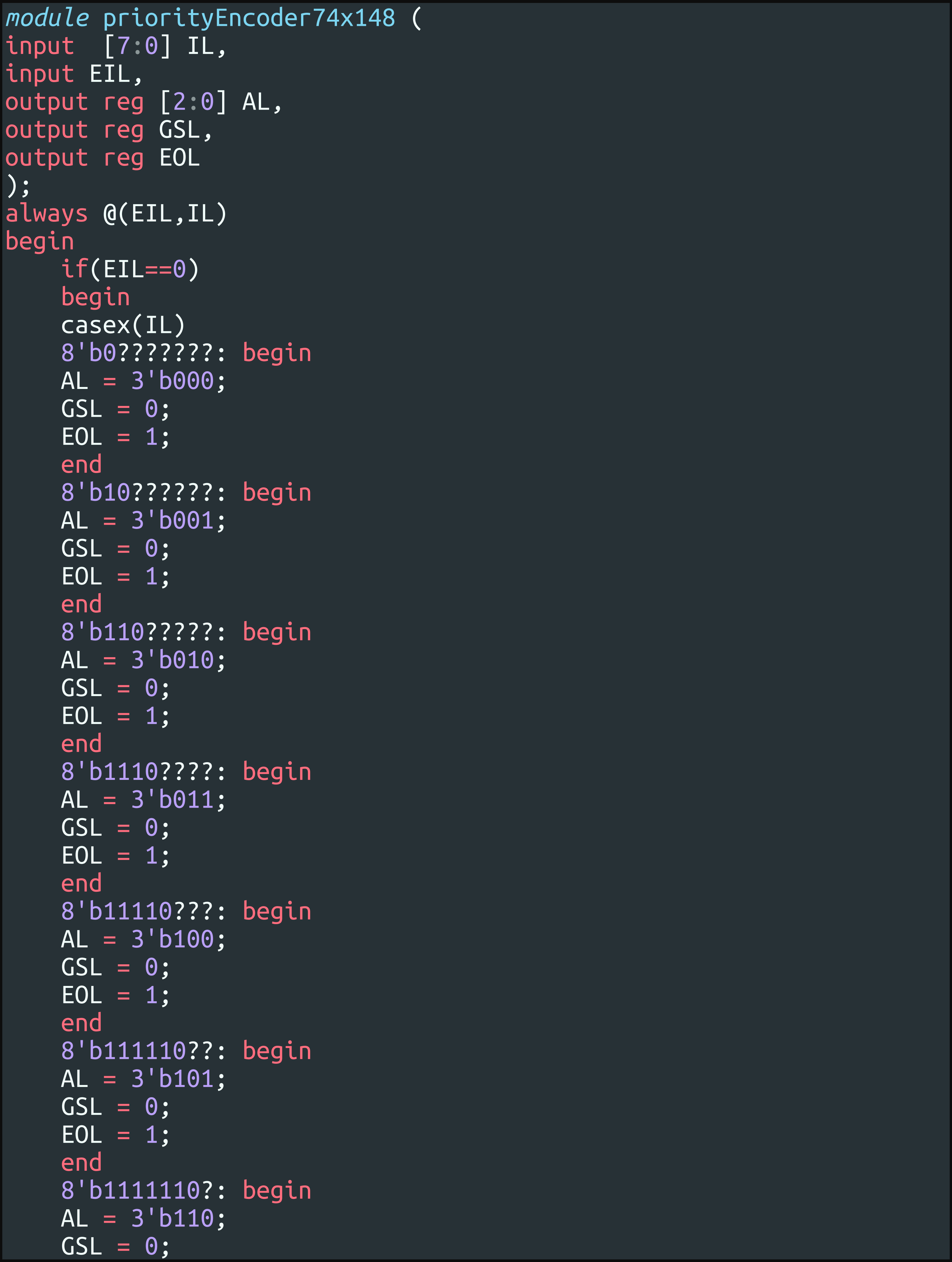
```verilog
module mux164to1 (
input [3:0]S, [15:0]I,
output reg O
);
always @(S)
begin
        O = I[S];
end
endmodule

module muxtestbench();
        reg [3:0] S;
        reg [15:0]I;
        wire O;
        integer a;
mux164to1 test(.S(S),.I(I), .O(O));
initial
begin
        I=16'b0100110000010010;
        for(a=1;a<16;a=a+2)
        begin
                assign S=a;
                #100;
        end
```

```
run
VSIM 246> run -all

VSIM 247>
```

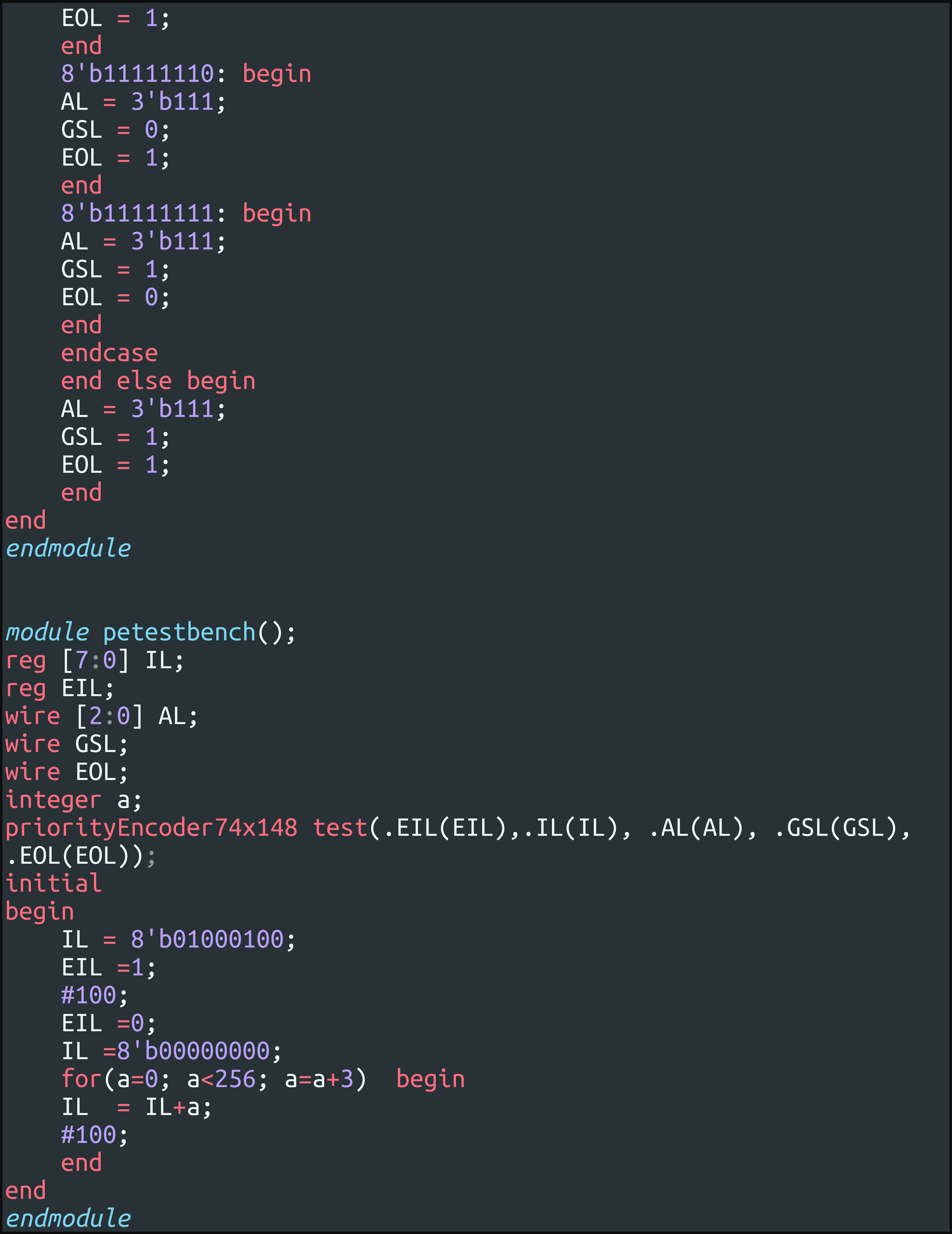
Wave:



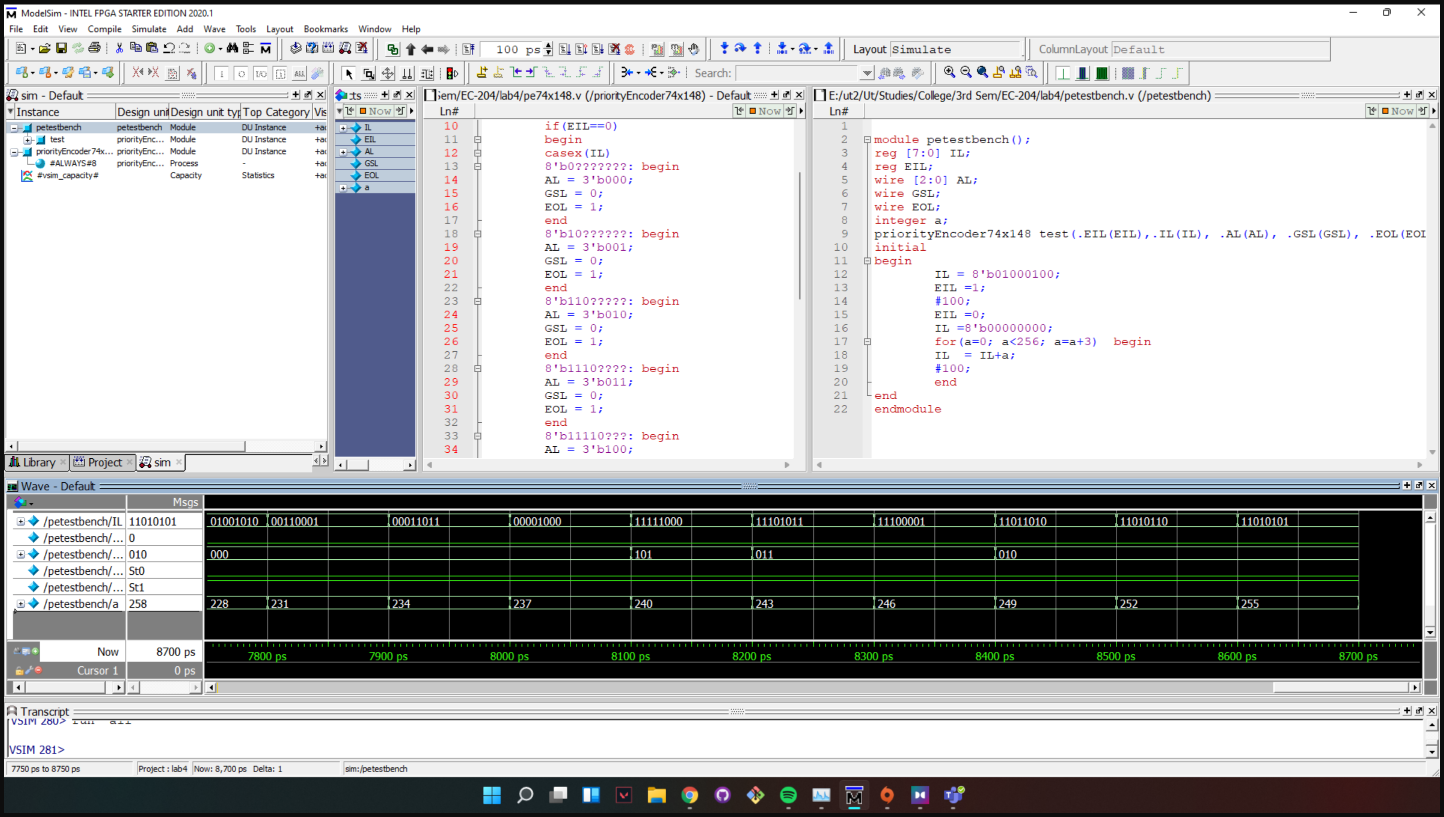
3] Functionality of 74x148 priority encoder



```verilog
module priorityEncoder74x148 (
input  [7:0] IL,
input EIL,
output reg [2:0] AL,
output reg GSL,
output reg EOL
);
always @(EIL,IL)
begin
    if(EIL==0)
    begin
    casex(IL)
    8'b0???????: begin
    AL = 3'b000;
    GSL = 0;
    EOL = 1;
    end
    8'b10??????: begin
    AL = 3'b001;
    GSL = 0;
    EOL = 1;
    end
    8'b110?????: begin
    AL = 3'b010;
    GSL = 0;
    EOL = 1;
    end
    8'b1110????: begin
    AL = 3'b011;
    GSL = 0;
    EOL = 1;
    end
    8'b11110???: begin
    AL = 3'b100;
    GSL = 0;
    EOL = 1;
    end
    8'b111110??: begin
    AL = 3'b101;
    GSL = 0;
    EOL = 1;
    end
    8'b1111110?: begin
    AL = 3'b110;
    GSL = 0;
```



```verilog
    EOL = 1;
    end
    8'b11111110: begin
    AL = 3'b111;
    GSL = 0;
    EOL = 1;
    end
    8'b11111111: begin
    AL = 3'b111;
    GSL = 1;
    EOL = 0;
    end
    endcase
    end else begin
    AL = 3'b111;
    GSL = 1;
    EOL = 1;
    end
end
endmodule


module petestbench();
reg [7:0] IL;
reg EIL;
wire [2:0] AL;
wire GSL;
wire EOL;
integer a;
priorityEncoder74x148 test(.EIL(EIL),.IL(IL), .AL(AL), .GSL(GSL),
.EOL(EOL));
initial
begin
    IL = 8'b01000100;
    EIL =1;
    #100;
    EIL =0;
    IL =8'b00000000;
    for(a=0; a<256; a=a+3)  begin
    IL  = IL+a;
    #100;
    end
end
endmodule
```



sem/EC-204/lab4/pe74x148.v (/priorityEncoder74x148) - Default

```verilog
            if(EIL==0)
            begin
            casex(IL)
            8'b0???????: begin
            AL = 3'b000;
            GSL = 0;
            EOL = 1;
            end
            8'b10??????: begin
            AL = 3'b001;
            GSL = 0;
            EOL = 1;
            end
            8'b110?????: begin
            AL = 3'b010;
            GSL = 0;
            EOL = 1;
            end
            8'b1110????: begin
            AL = 3'b011;
            GSL = 0;
            EOL = 1;
            end
            8'b11110???: begin
            AL = 3'b100;
```

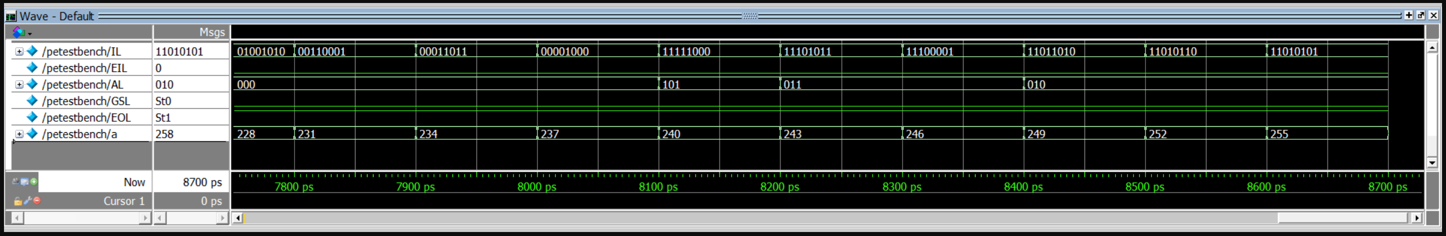
E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/petestbench.v (/petestbench)

```verilog

module petestbench();
reg [7:0] IL;
reg EIL;
wire [2:0] AL;
wire GSL;
wire EOL;
integer a;
priorityEncoder74x148 test(.EIL(EIL),.IL(IL), .AL(AL), .GSL(GSL), .EOL(EOI
initial
begin
        IL = 8'b01000100;
        EIL =1;
        #100;
        EIL =0;
        IL =8'b00000000;
        for(a=0; a<256; a=a+3)  begin
        IL  = IL+a;
        #100;
        end
end
endmodule
```

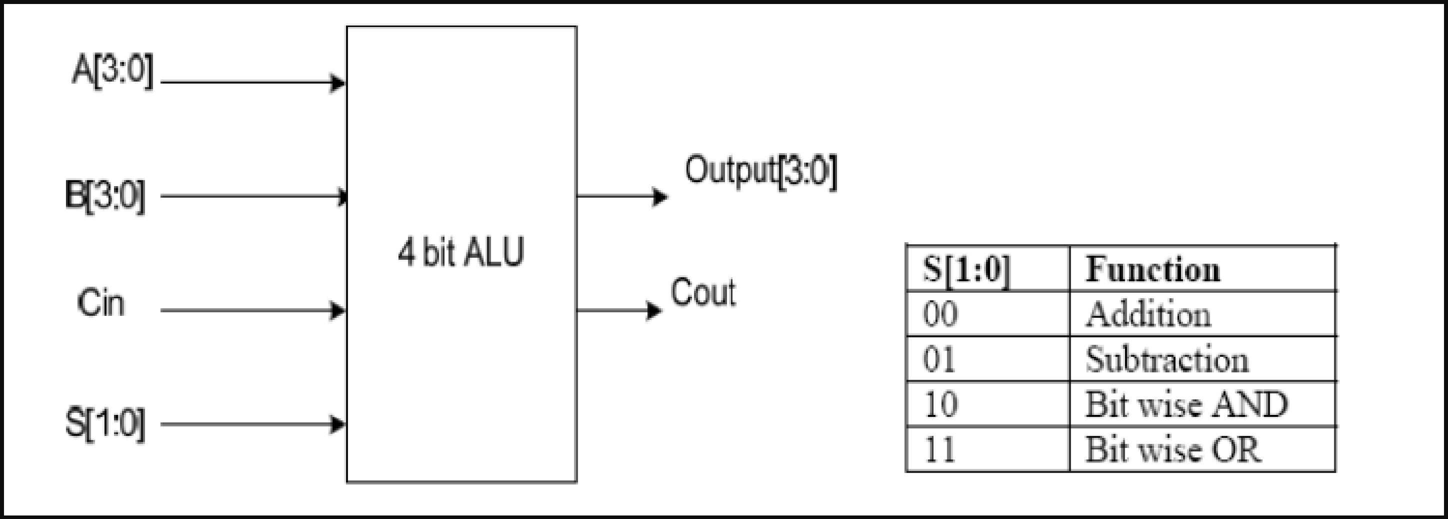
```
VSIM 280> run -all

VSIM 281>
```

```
7750 ps to 8750 ps    Project : lab4    Now: 8,700 ps    Delta: 1    sim:/petestbench
```

Wave:



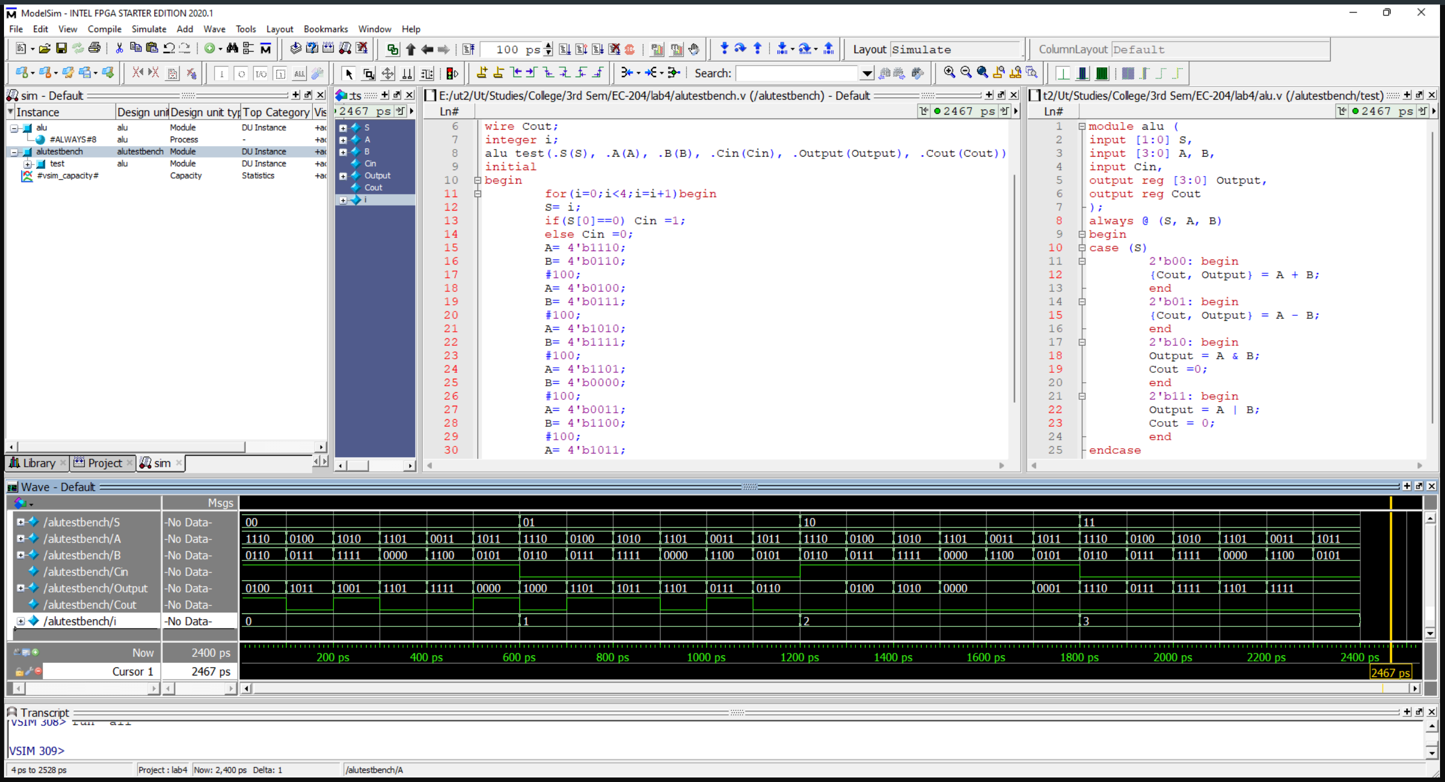
4] ALU design: Design a 4 bit ALU that is capable of performing addition, subtraction, bitwise AND and bit-wise OR instructions on 4 bit operands.



S[1:0]	Function
00	Addition
01	Subtraction
10	Bit wise AND
11	Bit wise OR

```verilog
module alu (
input [1:0] S,
input [3:0] A, B,
input Cin,
output reg [3:0] Output,
output reg Cout
);
always @ (S, A, B)
begin
case (S)
    2'b00: begin
    {Cout, Output} = A + B;
    end
    2'b01: begin
    {Cout, Output} = A - B;
    end
    2'b10: begin
    Output = A & B;
    Cout =0;
    end
    2'b11: begin
    Output = A | B;
    Cout = 0;
    end
endcase
end
endmodule

module alutestbench ();
reg [1:0]S;
reg [3:0] A, B;
reg Cin;
wire [3:0] Output;
wire Cout;
integer i;
alu test(.S(S), .A(A), .B(B), .Cin(Cin), .Output(Output),
.Cout(Cout));
initial
begin
    for(i=0;i<4;i=i+1)begin
    S= i;
    if(S[0]==0) Cin =1;
    else Cin =0;
    A= 4'b1110;
    B= 4'b0110;
```

```
        #100;
        A= 4'b0100;
        B= 4'b0111;
        #100;
        A= 4'b1010;
        B= 4'b1111;
        #100;
        A= 4'b1101;
        B= 4'b0000;
        #100;
        A= 4'b0011;
        B= 4'b1100;
        #100;
        A= 4'b1011;
        B= 4'b0101;
        #100;
        end
end
endmodule
```



E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/alutestbench.v (/alutestbench) - Default

```
wire Cout;
integer i;
alu test(.S(S), .A(A), .B(B), .Cin(Cin), .Output(Output), .Cout(Cout))
initial
begin
        for(i=0;i<4;i=i+1)begin
        S= i;
        if(S[0]==0) Cin =1;
        else Cin =0;
        A= 4'b1110;
        B= 4'b0110;
        #100;
        A= 4'b0100;
        B= 4'b0111;
        #100;
        A= 4'b1010;
        B= 4'b1111;
        #100;
        A= 4'b1101;
        B= 4'b0000;
        #100;
        A= 4'b0011;
        B= 4'b1100;
        #100;
        A= 4'b1011;
```

t2/Ut/Studies/College/3rd Sem/EC-204/lab4/alu.v (/alutestbench/test)

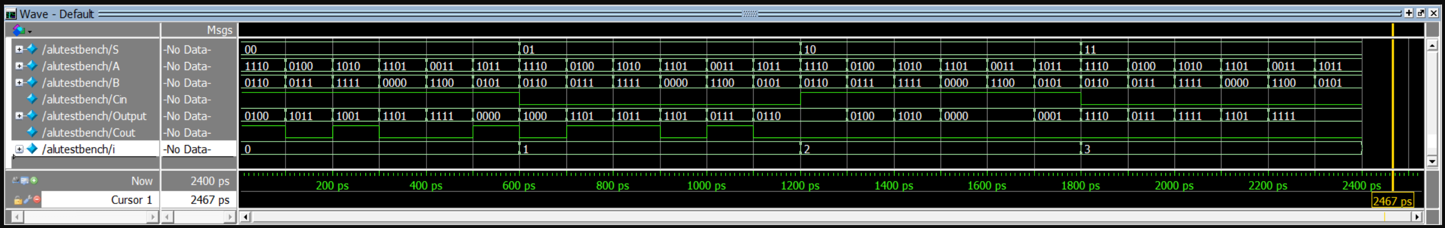
```
module alu (
input [1:0] S,
input [3:0] A, B,
input Cin,
output reg [3:0] Output,
output reg Cout
);
always @ (S, A, B)
begin
case (S)
        2'b00: begin
        {Cout, Output} = A + B;
        end
        2'b01: begin
        {Cout, Output} = A - B;
        end
        2'b10: begin
        Output = A & B;
        Cout =0;
        end
        2'b11: begin
        Output = A | B;
        Cout = 0;
        end
endcase
```

```
/alutestbench/S       -No Data-  00    01    10    11
/alutestbench/A       -No Data-  1110 0100 1010 1101 0011 1011 (repeated for each S)
/alutestbench/B       -No Data-  0110 0111 1111 0000 1100 0101 (repeated for each S)
/alutestbench/Cin     -No Data-
/alutestbench/Output  -No Data-  0100 1011 1001 1101 1111 0000 1000 1101 1011 1101 0111 0110 0100 1010 0000 0001 1110 0111 1111 1101 1111
/alutestbench/Cout    -No Data-
/alutestbench/i       -No Data-  0  1  2  3
Now 2400 ps
Cursor 1  2467 ps
```

```
VSIM 308> run -all

VSIM 309>
```

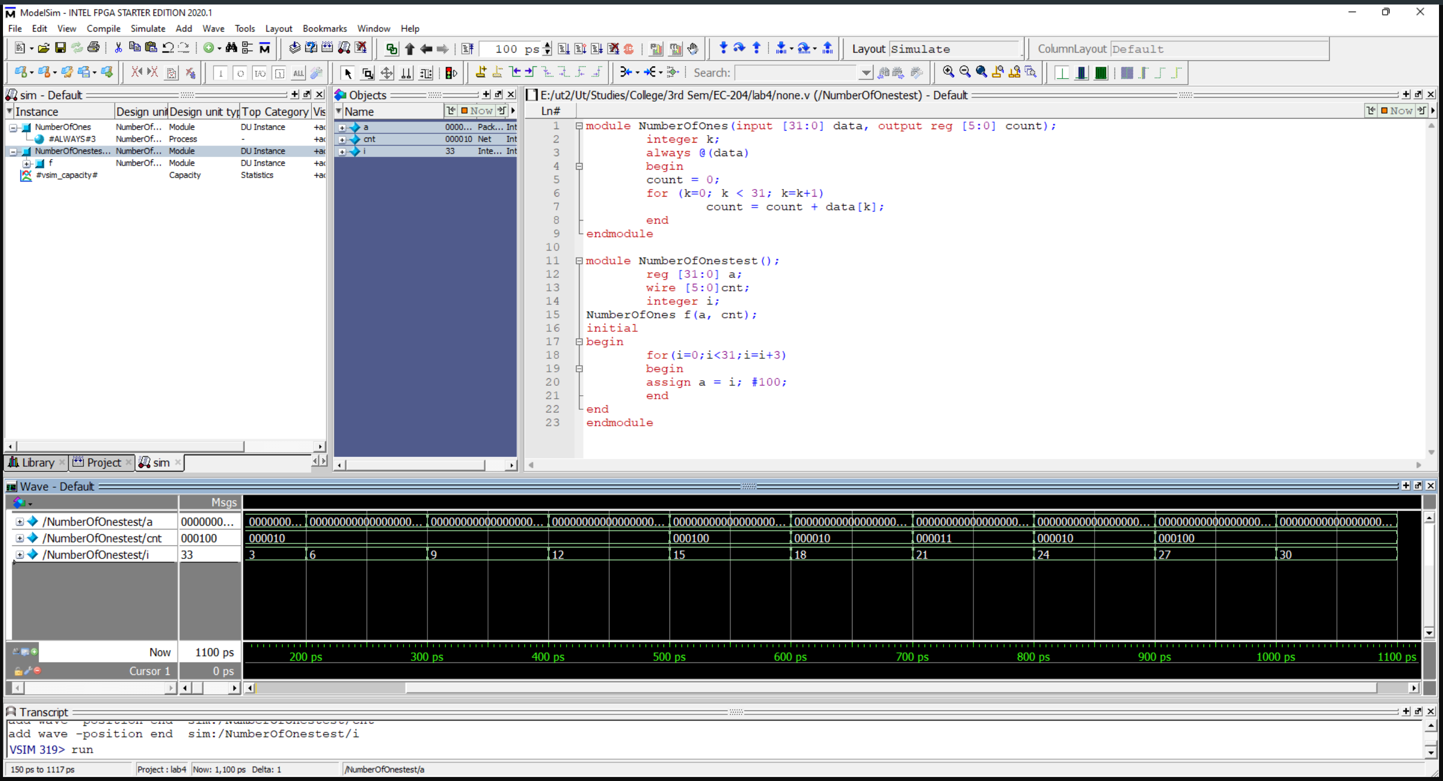
4 ps to 2528 ps   Project : lab4   Now: 2,400 ps Delta: 1   /alutestbench/A

Wave:



5] Count the number of 1s in a 32 bit number

```
module NumberOfOnes(input [31:0] data, output reg [5:0] count);
    integer k;
    always @(data)
    begin
    count = 0;
    for (k=0; k < 31; k=k+1)
        count = count + data[k];
    end
endmodule

module NumberOfOnestest();
    reg [31:0] a;
    wire [5:0]count;
    integer i;
NumberOfOnes f(.data(a), .count(count));
initial
begin
    for(i=0;i<429496729;i=i+42900000)
    begin
    assign a = i; #100;
    end
end
endmodule
```


ModelSim - INTEL FPGA STARTER EDITION 2020.1

sim - Default

Instance	Design unit	Design unit type	Top Category	Vis
NumberOfOnes	NumberOf...	Module	DU Instance	+ac
#ALWAYS#3	NumberOf...	Process	-	+ac
NumberOfOnestes...	NumberOf...	Module	DU Instance	+ac
f	NumberOf...	Module	DU Instance	+ac
#vsim_capacity#		Capacity	Statistics	+ac

Objects

Name	Now		
a	0000...	Pack...	Int
cnt	000010	Net	Int
i	33	Inte...	Int

E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/none.v (/NumberOfOnestest) - Default

```
module NumberOfOnes(input [31:0] data, output reg [5:0] count);
        integer k;
        always @(data)
        begin
        count = 0;
        for (k=0; k < 31; k=k+1)
                count = count + data[k];
        end
endmodule

module NumberOfOnestest();
        reg [31:0] a;
        wire [5:0]cnt;
        integer i;
NumberOfOnes f(a, cnt);
initial
begin
        for(i=0;i<31;i=i+3)
        begin
        assign a = i; #100;
        end
end
endmodule
```

Wave - Default

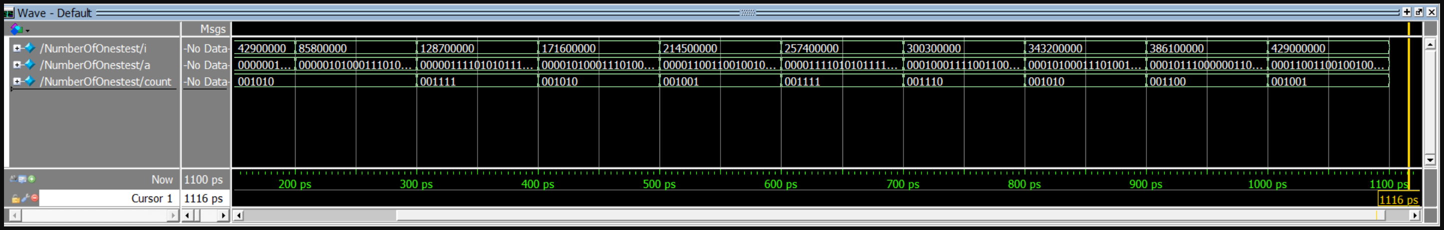
```
                      Msgs
/NumberOfOnestest/a   0000000...  0000000... 0000000000000000... (repeated)
/NumberOfOnestest/cnt 000100      000010                      000100  000010  000011  000010  000100
/NumberOfOnestest/i   33          3  6  9  12  15  18  21  24  27  30
Now       1100 ps     200 ps  300 ps  400 ps  500 ps  600 ps  700 ps  800 ps  900 ps  1000 ps  1100 ps
Cursor 1  0 ps
```

Transcript

```
add wave -position end  sim:/NumberOfOnestest/i
VSIM 319> run
```

150 ps to 1117 ps | Project : lab4 | Now: 1,100 ps Delta: 1 | /NumberOfOnestest/a

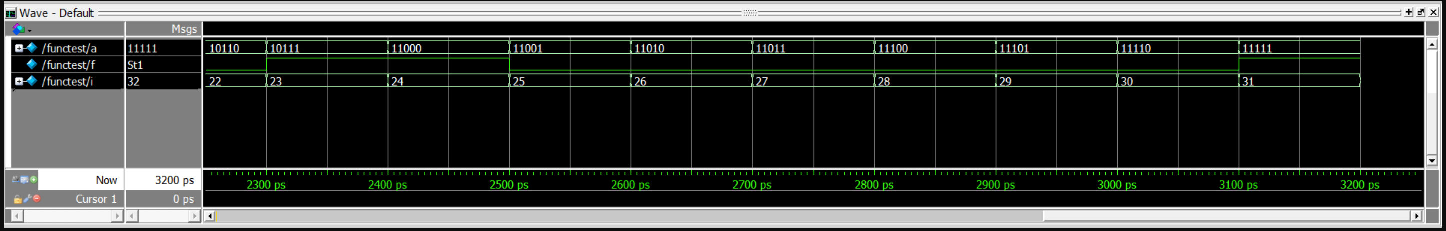
Wave:



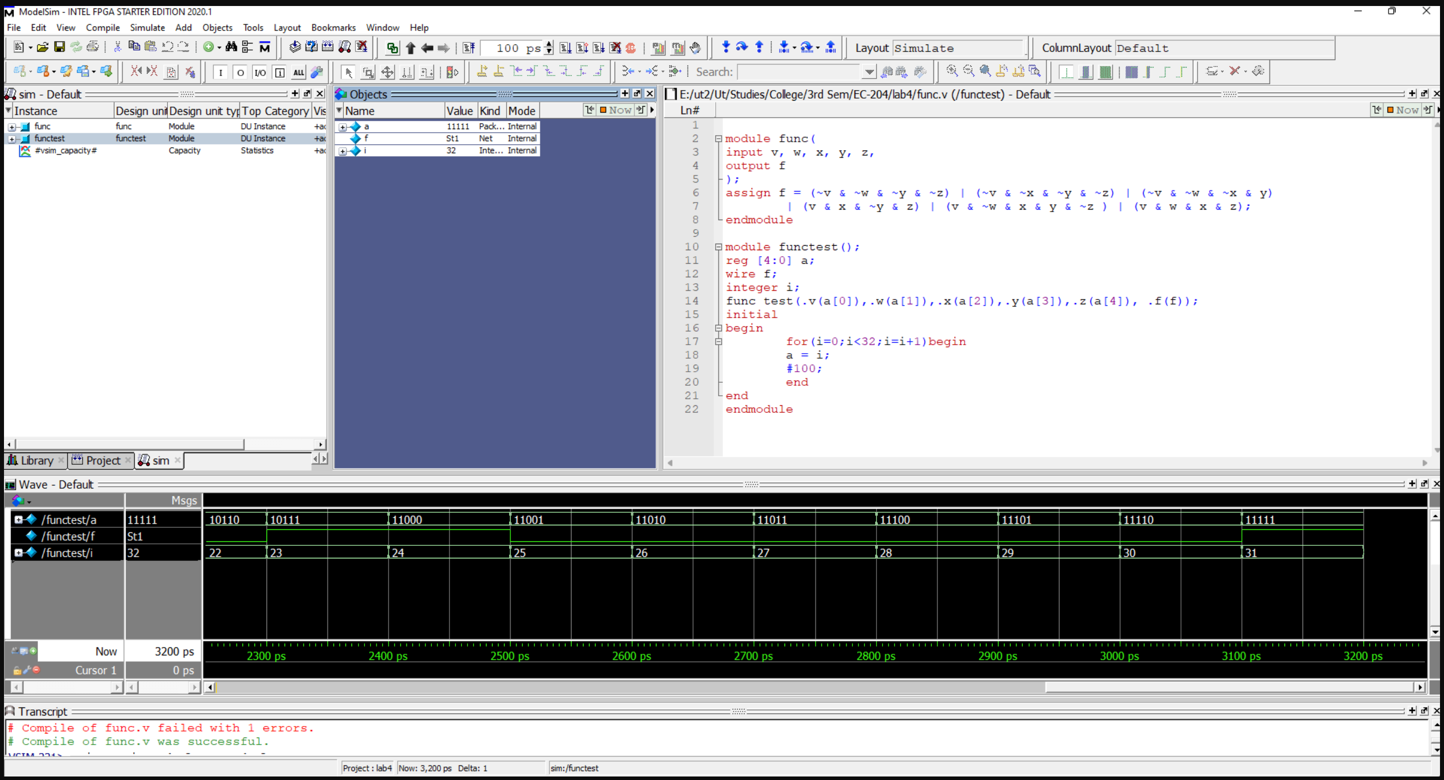
```
Wave - Default
                              Msgs
/NumberOfOnestest/i          -No Data-  42900000  85800000  128700000  171600000  214500000  257400000  300300000  343200000  386100000  429000000
/NumberOfOnestest/a          -No Data-  0000001...  0000010100011101 0...  00000111101010111...  00001010001110100...  00001100110010010...  00001111010101111...  00010001111001100...  00010100011101001...  00010111000000110...  00011001100100100...
/NumberOfOnestest/count      -No Data-  001010  001111  001010  001001  001111  001110  001010  001100  001001
Now       1100 ps    200 ps  300 ps  400 ps  500 ps  600 ps  700 ps  800 ps  900 ps  1000 ps  1100 ps
Cursor 1  1116 ps    1116 ps
```

6] Implement F(v,w,x,y,z) = ∑ (0, 2, 3, 4, 8, 21, 22, 29, 31)

```
module func(
input v, w, x, y, z,
output f
);
assign f = (~v & ~w & ~y & ~z) | (~v & ~x & ~y & ~z) | (~v & ~w &
~x & y)
    | (v & x & ~y & z) | (v & ~w & x & y & ~z ) | (v & w & x & z);
endmodule

module functest();
reg [4:0] a;
wire f;
integer i;
func test(.v(a[0]),.w(a[1]),.x(a[2]),.y(a[3]),.z(a[4]), .f(f));
initial
begin
    for(i=0;i<32;i=i+1)begin
    a = i;
    #100;
    end
end
endmodule
```



```
Wave - Default
                  Msgs
/functest/a      11111   10110  10111  11000  11001  11010  11011  11100  11101  11110  11111
/functest/f      St1
/functest/i      32      22  23  24  25  26  27  28  29  30  31
Now       3200 ps    2300 ps  2400 ps  2500 ps  2600 ps  2700 ps  2800 ps  2900 ps  3000 ps  3100 ps  3200 ps
Cursor 1  0 ps
```



E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/func.v (/functest) - Default

```
module func(
input v, w, x, y, z,
output f
);
assign f = (~v & ~w & ~y & ~z) | (~v & ~x & ~y & ~z) | (~v & ~w & ~x & y)
        | (v & x & ~y & z) | (v & ~w & x & y & ~z ) | (v & w & x & z);
endmodule

module functest();
reg [4:0] a;
wire f;
integer i;
func test(.v(a[0]),.w(a[1]),.x(a[2]),.y(a[3]),.z(a[4]), .f(f));
initial
begin
        for(i=0;i<32;i=i+1)begin
        a = i;
        #100;
        end
end
endmodule
```

Wave - Default

```
/functest/a   11111   10110 10111 11000 11001 11010 11011 11100 11101 11110 11111
/functest/f   St1
/functest/i   32      22 23 24 25 26 27 28 29 30 31

Now       3200 ps   2300 ps 2400 ps 2500 ps 2600 ps 2700 ps 2800 ps 2900 ps 3000 ps 3100 ps 3200 ps
Cursor 1  0 ps
```

Transcript

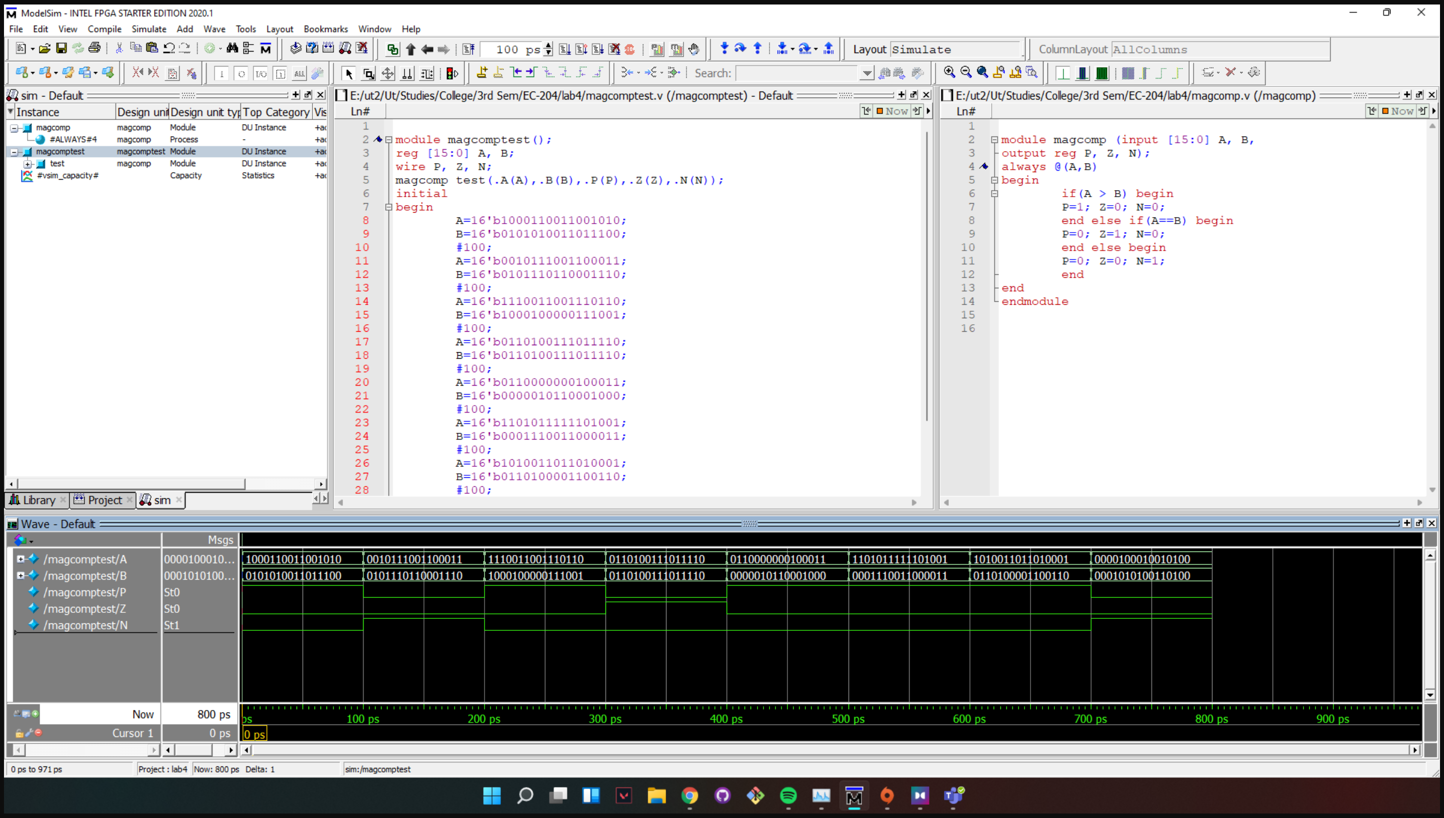
```
# Compile of func.v failed with 1 errors.
# Compile of func.v was successful.
```

Project : lab4 | Now: 3,200 ps Delta: 1 | sim:/functest

7] 16 bit magnitude comparator

```
module magcomp (input [15:0] A, B,
output reg P, Z, N);
always @(A,B)
begin
    if(A > B) begin
    P=1; Z=0; N=0;
    end else if(A==B) begin
    P=0; Z=1; N=0;
    end else begin
    P=0; Z=0; N=1;
    end
end
endmodule

module magcomptest();
reg [15:0] A, B;
wire P, Z, N;
magcomp test(.A(A),.B(B),.P(P),.Z(Z),.N(N));
initial
begin
    A=16'b1000110011001010;
    B=16'b0101010011011100;
    #100;
```

```verilog
    A=16'b0010111001100011;
    B=16'b0101110110001110;
    #100;
    A=16'b1110011001110110;
    B=16'b1000100000111001;
    #100;
    A=16'b0110100111011110;
    B=16'b0110100111011110;
    #100;
    A=16'b0110000000100011;
    B=16'b0000010110001000;
    #100;
    A=16'b1101011111101001;
    B=16'b0001110011000011;
    #100;
    A=16'b1010011011010001;
    B=16'b0110100001100110;
    #100;
    A=16'b0000100010010100;
    B=16'b0001010100110100;
    #100;
end
endmodule
```



E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/magcomptest.v (/magcomptest) - Default

```verilog

module magcomptest();
reg [15:0] A, B;
wire P, Z, N;
magcomp test(.A(A),.B(B),.P(P),.Z(Z),.N(N));
initial
begin
        A=16'b1000110011001010;
        B=16'b0101010011011100;
        #100;
        A=16'b0010111001100011;
        B=16'b0101110110001110;
        #100;
        A=16'b1110011001110110;
        B=16'b1000100000111001;
        #100;
        A=16'b0110100111011110;
        B=16'b0110100111011110;
        #100;
        A=16'b0110000000100011;
        B=16'b0000010110001000;
        #100;
        A=16'b1101011111101001;
        B=16'b0001110011000011;
        #100;
        A=16'b1010011011010001;
        B=16'b0110100001100110;
        #100;
```

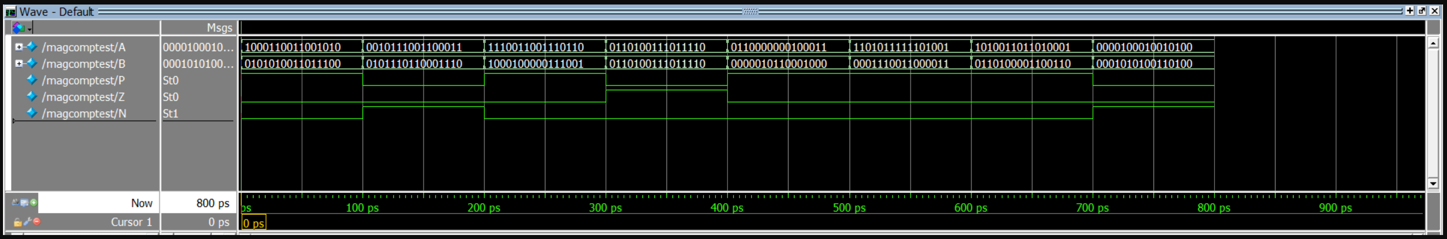
E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/magcomp.v (/magcomp)

```verilog

module magcomp (input [15:0] A, B,
output reg P, Z, N);
always @(A,B)
begin
        if(A > B) begin
        P=1; Z=0; N=0;
        end else if(A==B) begin
        P=0; Z=1; N=0;
        end else begin
        P=0; Z=0; N=1;
        end
end
endmodule


```

Wave - Default

```
/magcomptest/A   0000100010...  1000110011001010 0010111001100011 1110011001110110 0110100111011110 0110000000100011 1101011111101001 1010011011010001 0000100010010100
/magcomptest/B   0001010100...  0101010011011100 0101110110001110 1000100000111001 0110100111011110 0000010110001000 0001110011000011 0110100001100110 0001010100110100
/magcomptest/P   St0
/magcomptest/Z   St0
/magcomptest/N   St1

Now     800 ps
Cursor 1  0 ps
```

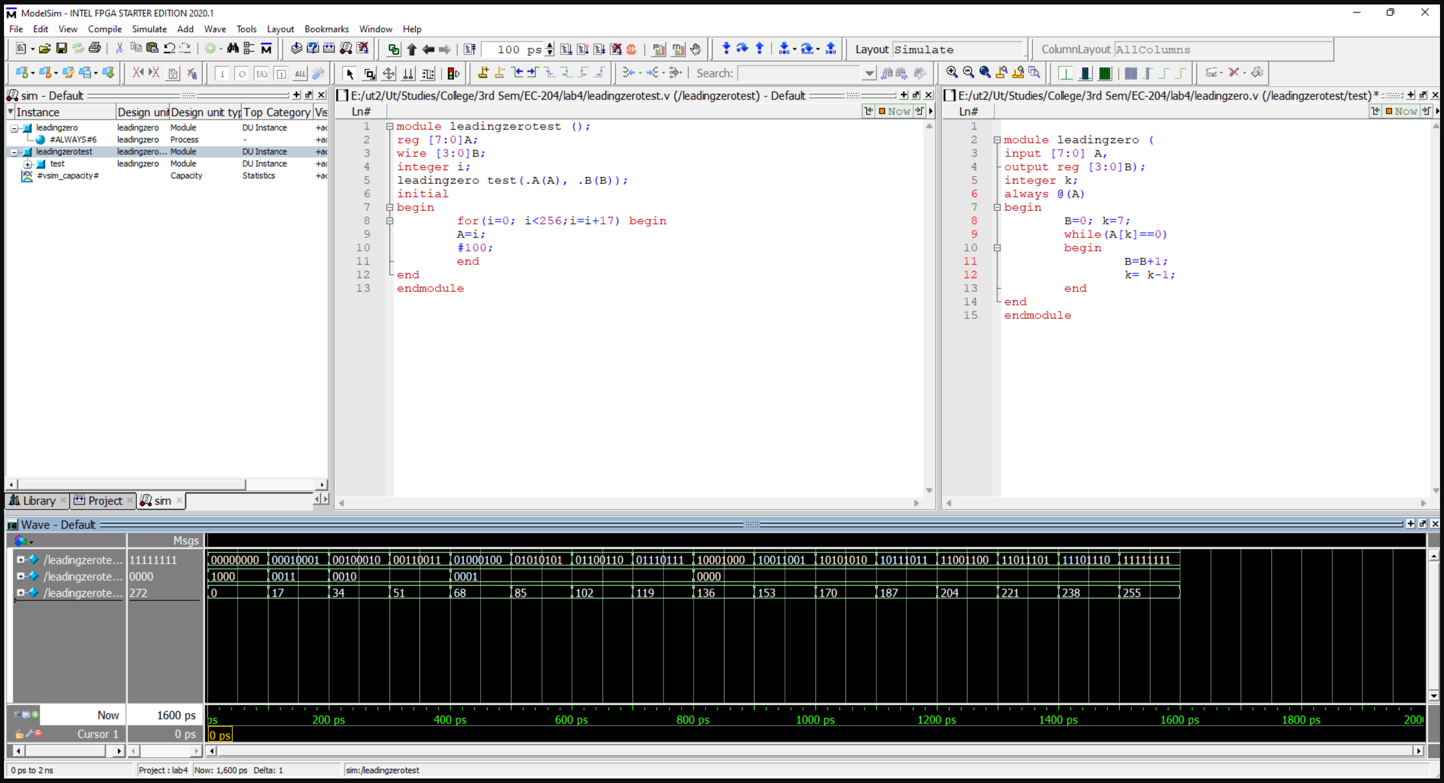
0 ps to 971 ps | Project : lab4 | Now: 800 ps Delta: 1 | sim:/magcomptest



```
Wave - Default
                     Msgs
/magcomptest/A   0000100010...  1000110011001010  0010111001100011  1110011001110110  0110100111011110  0110000000100011  1101011111101001  1010011011010001  0000100010010100
/magcomptest/B   0001010100...  0101010011011100  0101110110001110  1000100000111001  0110100111011110  0000010110001000  0001110011000011  0110100001100110  0001010100110100
/magcomptest/P   St0
/magcomptest/Z   St0
/magcomptest/N   St1
Now       800 ps   ps  100 ps  200 ps  300 ps  400 ps  500 ps  600 ps  700 ps  800 ps  900 ps
Cursor 1  0 ps     0 ps
```

8] Count the number of leading 0s in an 8 bit number

```
module leadingzero (
input [7:0] A,
output reg [3:0]B);
integer k;
always @(A)
begin
    B=0; k=7;
    while(A[k]==0)
    begin
        B=B+1;
        k= k-1;
    end
end
endmodule
module leadingzerotest ();
reg [7:0]A;
wire [3:0]B;
integer i;
leadingzero test(.A(A), .B(B));
initial
begin
    for(i=0; i<256;i=i+17) begin
    A=i;
    #100;
    end
end
endmodule
```


ModelSim - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help
Layout Simulate
ColumnLayout AllColumns
sim - Default
Library Project sim
Wave - Default
Now 1600 ps
Cursor 1 0 ps
0 ps to 2 ns
Project : lab4 Now: 1,600 ps Delta: 1
sim:/leadingzerotest

E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/leadingzerotest.v (/leadingzerotest) - Default

```
module leadingzerotest ();
reg [7:0]A;
wire [3:0]B;
integer i;
leadingzero test(.A(A), .B(B));
initial
begin
        for(i=0; i<256;i=i+17) begin
        A=i;
        #100;
        end
end
endmodule
```

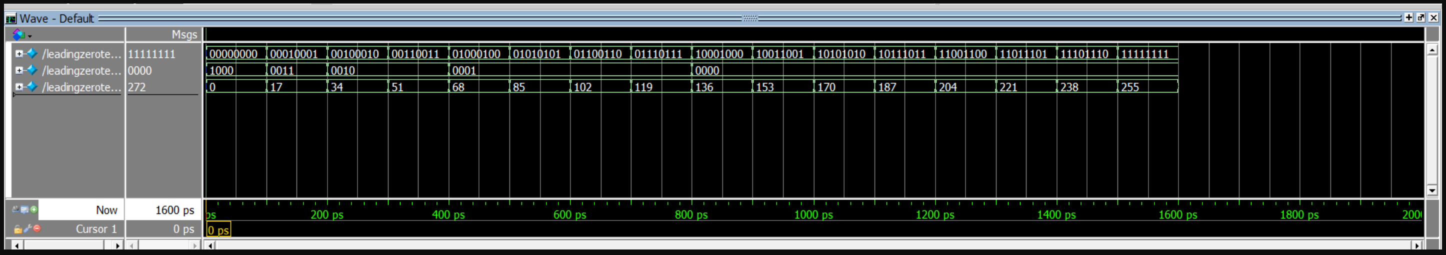
E:/ut2/Ut/Studies/College/3rd Sem/EC-204/lab4/leadingzero.v (/leadingzerotest/test) *

```

module leadingzero (
input [7:0] A,
output reg [3:0]B);
integer k;
always @(A)
begin
        B=0; k=7;
        while(A[k]==0)
        begin
                B=B+1;
                k= k-1;
        end
end
endmodule
```

Instance	Design unit	Design unit type	Top Category
leadingzero	leadingzero	Module	DU Instance
#ALWAYS#6	leadingzero	Process	-
leadingzerotest	leadingzero...	Module	DU Instance
test	leadingzero	Module	DU Instance
#vsim_capacity#		Capacity	Statistics

```
/leadingzerote...  11111111  00000000 00010001 00100010 00110011 01000100 01010101 01100110 01110111 10001000 10011001 10101010 10111011 11001100 11011101 11101110 11111111
/leadingzerote...  0000      1000 0011 0010 0001 0000
/leadingzerote...  272       0 17 34 51 68 85 102 119 136 153 170 187 204 221 238 255
```

Wave:



```
/leadingzerote...  11111111  00000000 00010001 00100010 00110011 01000100 01010101 01100110 01110111 10001000 10011001 10101010 10111011 11001100 11011101 11101110 11111111
/leadingzerote...  0000      1000 0011 0010 0001 0000
/leadingzerote...  272       0 17 34 51 68 85 102 119 136 153 170 187 204 221 238 255
```