EC-204

<LAB- 4>

NITK SURATHKAL

INBASEKARAN.P

201EC226

Prof: Sumam S

1. 2 to 4 decoder using (a) concurrent signal assignment statements (b) case statement

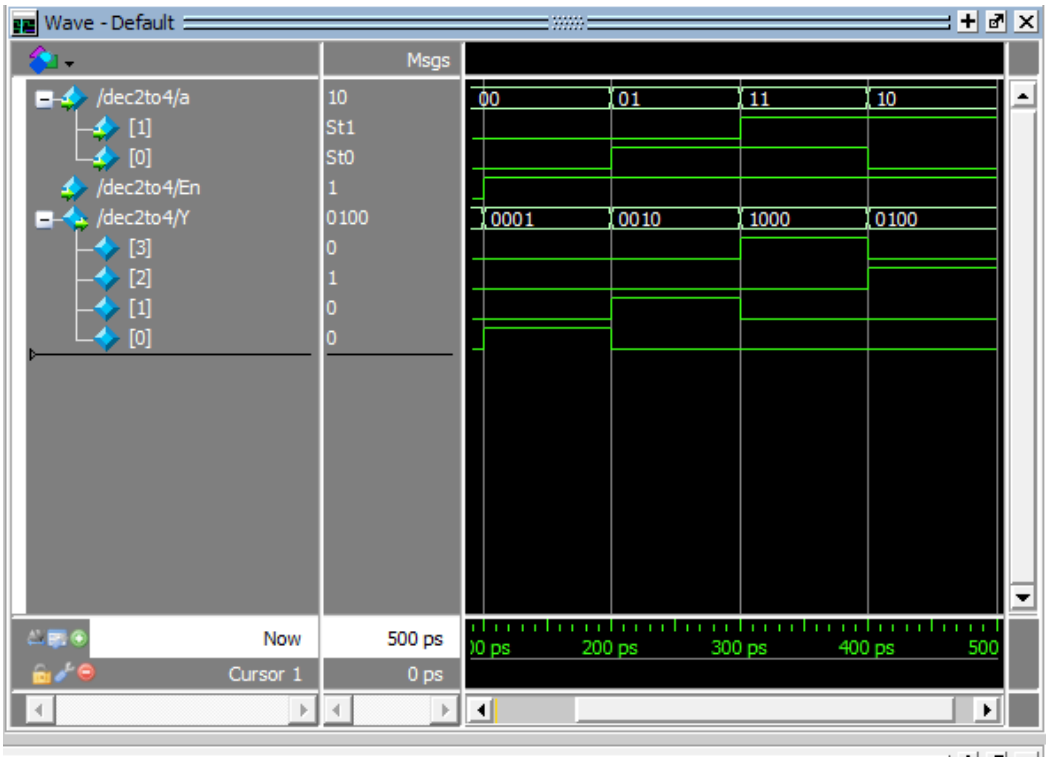
Solution:

a) Concurrent Signal assignments

dec2to4.v

```
1   module dec2to4(a, En, Y);
2     input [1:0] a;
3     input En;
4     output reg [3:0] Y;
5     /*
6     // Can be done using multiple assign statements
7     assign Y[0] = (En & ~a[1] & ~a[0]);
8     assign Y[1] = (En & ~a[1] & a[0]);
9     assign Y[2] = (En & a[1] & ~a[0]);
10    assign Y[3] = (En & a[1] & a[0]);
11    */
12    // or a short version with one assign statement
13    // When enable is one 1 is right shifted by a bits
14    assign Y = En ? (1<<a) : (4'b0000);
15  endmodule
```

Simulation waveform

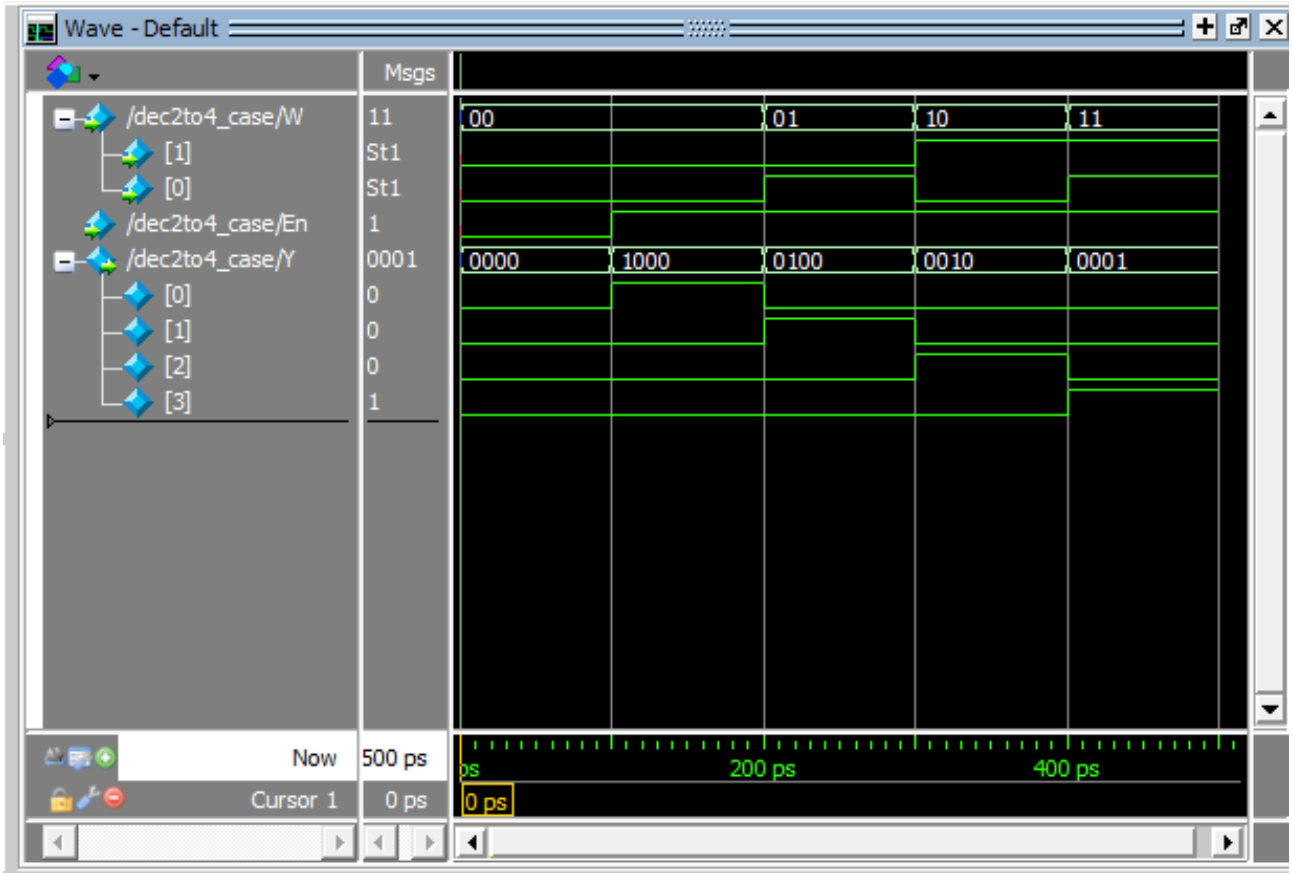


b) Case statements

dec2to4_case.v

```
module dec2to4_case (W, En, Y);
// input lines 2 bit
  input [1:0] W;
// enable input
  input En;
// output line 4 bit
  output reg [0:3] Y;
  always @(W, En)
    begin
      // if enable is 0 then output is alaways 0
      if (En == 0)
        Y = 4'b0000;
      else
    case (W)
    // output mapped according to the truth table
      0: Y = 4'b1000;
      1: Y = 4'b0100;
      2: Y = 4'b0010;
      3: Y = 4'b0001;
    endcase
  end
endmodule
```

Simulation waveform



2. 16:1 Mux

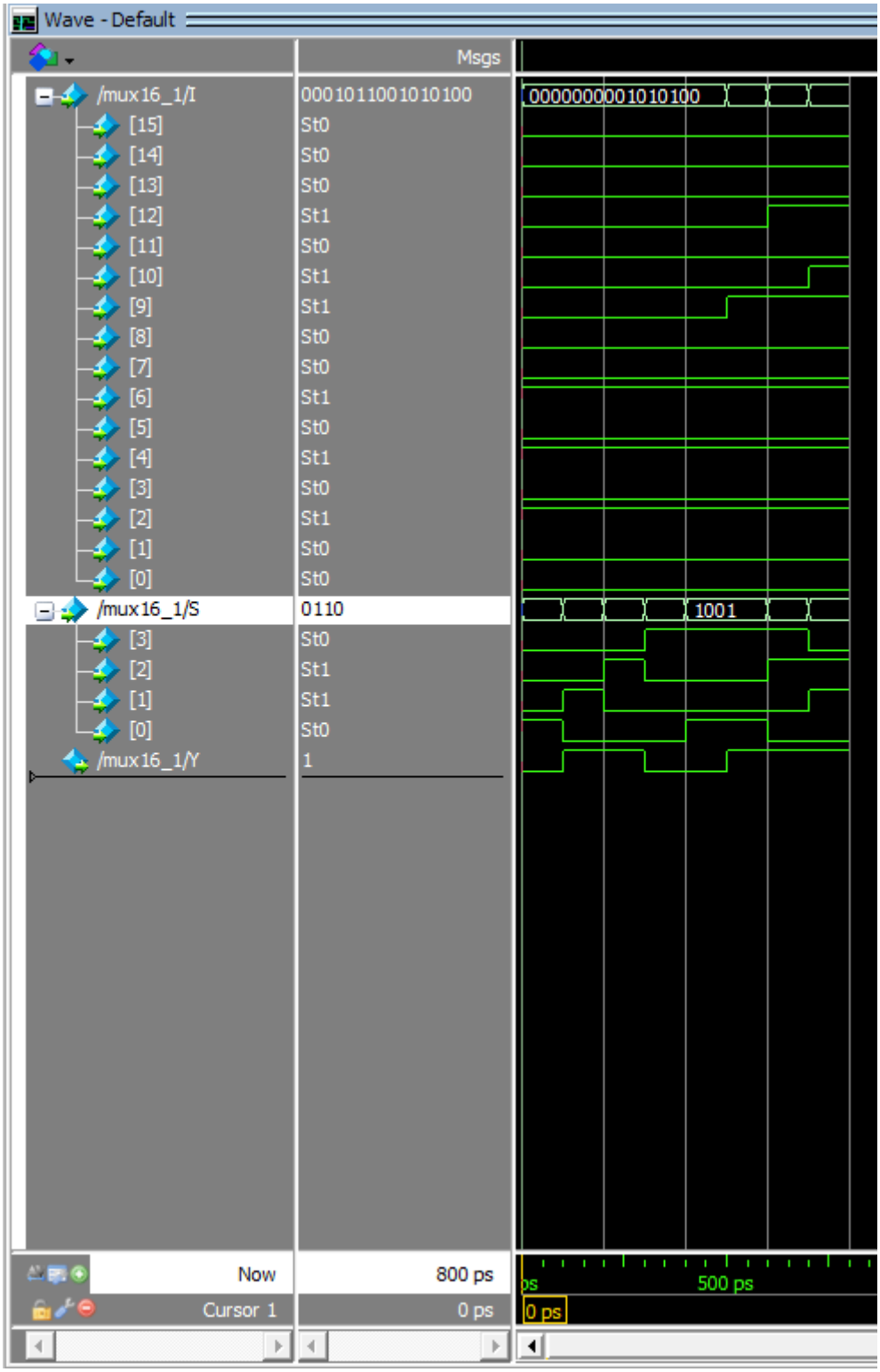
Solution:

Verilog code

question2 > mux16_1.v

```
1   module mux16_1(I,S,Y);
2   // input I 16 bit
3     input [15:0] I;
4   // selector 4 bit
5     input [3:0] S;
6   // output Y
7     output reg Y;
8   // Bit Select operation
9     assign Y = I[S];
10  endmodule
```

Simulation


Wave - Default
Msgs
/mux16_1/I
0001011001010100
0000000001010100
[15]
St0
[14]
St0
[13]
St0
[12]
St1
[11]
St0
[10]
St1
[9]
St1
[8]
St0
[7]
St0
[6]
St1
[5]
St0
[4]
St1
[3]
St0
[2]
St1
[1]
St0
[0]
St0
/mux16_1/S
0110
1001
[3]
St0
[2]
St1
[1]
St1
[0]
St0
/mux16_1/Y
1
Now
800 ps
500 ps
Cursor 1
0 ps
0 ps

3. Functionality of 74x148 priority encoder

Solution:

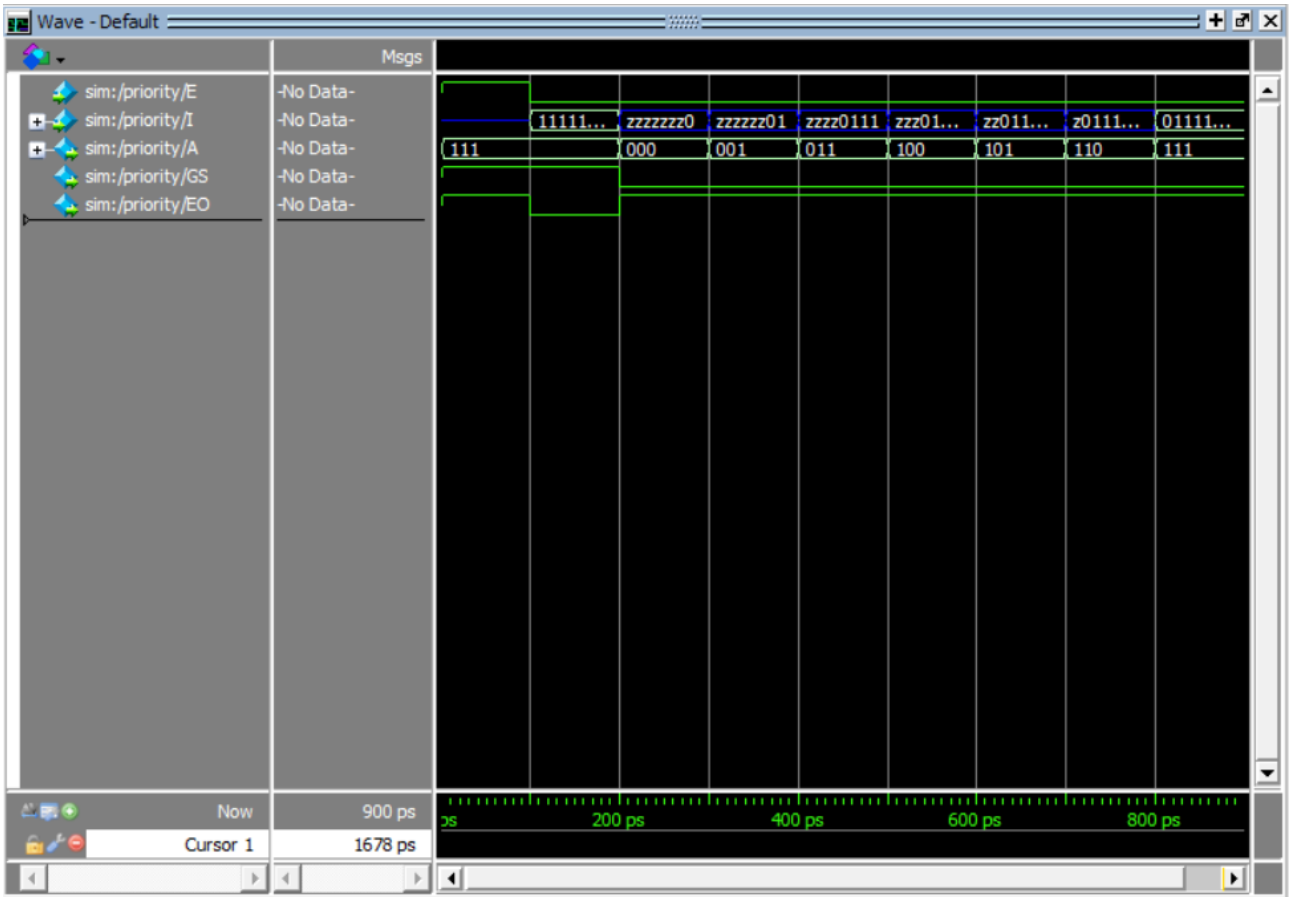
Verilog code

question3 > priority.v

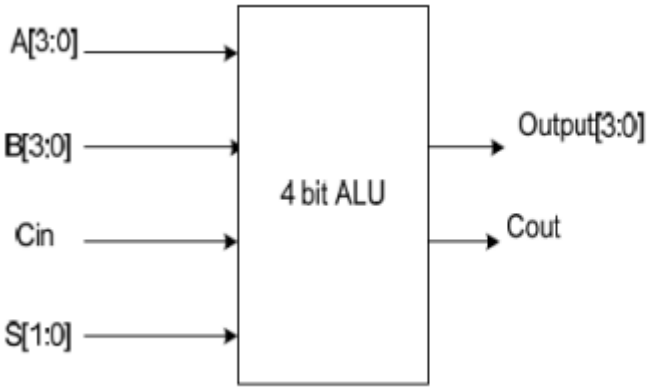
```
1   module prirority(E, I, A, GS, EO);
2     input E; input [7:0]I;
3     output reg [2:0]A;
4     output reg GS;
5     output reg EO;
6     always @(E, I) begin
7   // active low enable
8       if(E == 1)
9       begin
10        GS = 1;
11        EO = 1;
12        A = 3'd7;
13      end
14      else
15      begin
16  // cases for all the inputs acc to the truth table
17        casex (I)
18          8'bxxxxxxx0: begin
19              A = 0;
20              GS = 0;
21              EO = 1;
22            end
23          8'bxxxxxx01: begin
24              A = 3'd1;
25              GS = 0;
26              EO = 1;
27            end
28          8'bxxxxx011: begin
29              A = 3'd2;
30              GS = 0;
31              EO = 1;
```

```
29                A = 3'd2;
30                GS = 0;
31                EO = 1;
32              end
33              8'bxxxx0111: begin
34                A = 3'd3;
35                GS = 0;
36                EO = 1;
37              end
38             8'bxxx01111: begin
39                A = 3'd4;
40                GS = 0;
41                EO = 1;
42              end
43              8'bxx011111: begin
44                A = 3'd5;
45                GS = 0;
46                EO = 1;
47              end
48              8'bx0111111: begin
49                A = 3'd6;
50                GS = 0;
51                EO = 1;
52              end
53              8'b01111111: begin
54                A = 3'd7;
55                GS = 0;
56                EO = 1;
57              end
58              8'b11111111: begin
59                A = 3'd7;
60                GS = 1;
61                EO = 0;
62              end
63          endcase
64        end
65      end
66    endmodule
67
68
```

Simulation waveform



4. ALU design



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

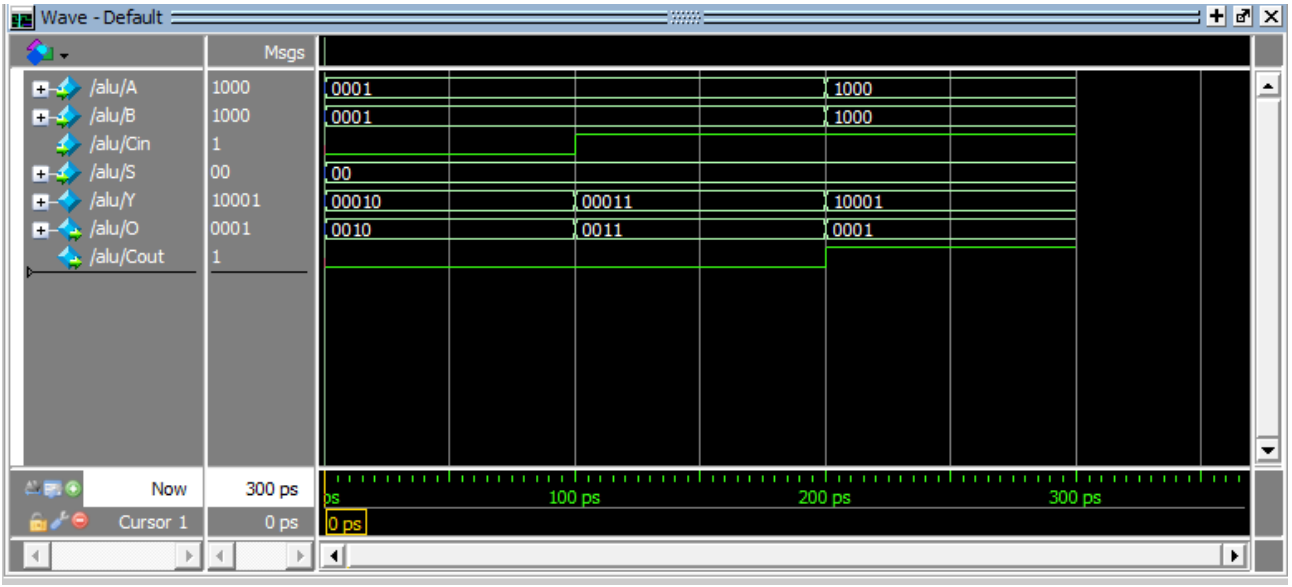
Solution:

Verilog code

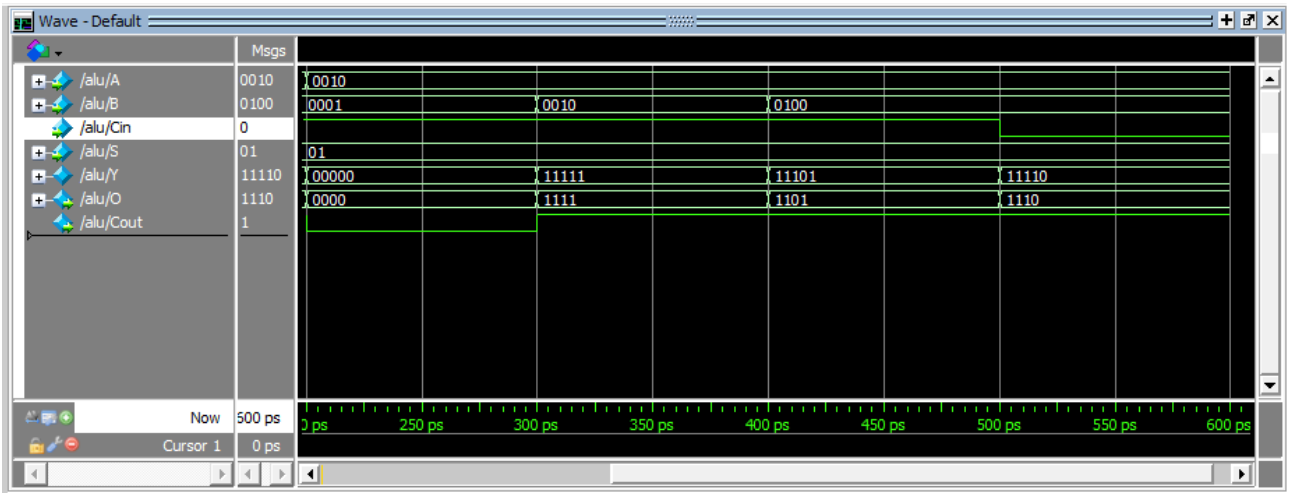
question4 > alu.v

```
1   module alu ( A, B, Cin, S, O, Cout);
2     input [3:0] A,B;
3     input Cin;
4     input [1:0]S;
5     reg [4:0] Y;
6     output reg [3:0] O;
7     output reg Cout;
8     always @(A,B,Cin,S) begin
9       // case for the given selector variable
10      case(S)
11      // addn
12        0 : Y = A + B + Cin;
13        // sub
14        1 : Y = A - B - Cin;
15        // bitwise and
16        2 : Y = A & B;
17        // bitwise or
18        3 : Y = A | B;
19      endcase
20      // assigining output to Cout and O
21      {Cout, O} = Y;
22    end
23  endmodule
```

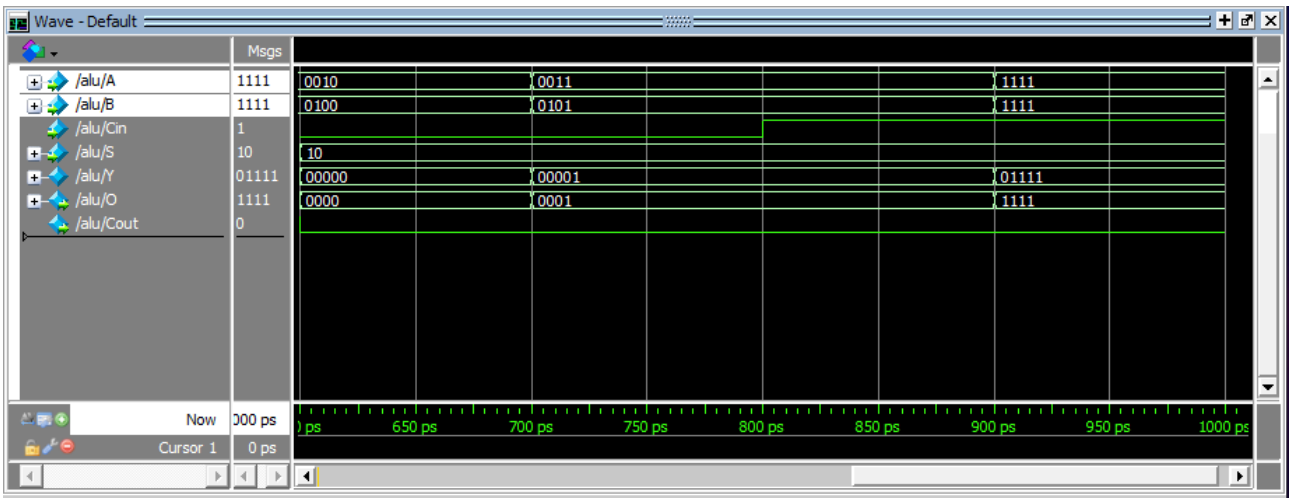
Simulation waveform for S = 00



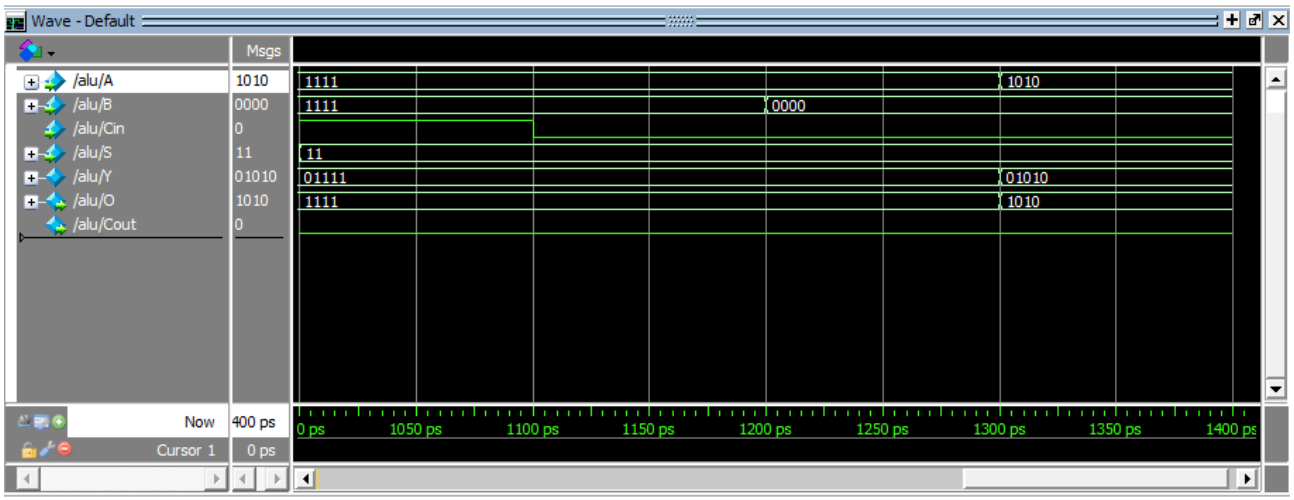
Simulation waveform for S = 01



Simulation waveform for S = 10



Simulation waveform for S = 11



5. Count number of 1s in a 32-bit number.

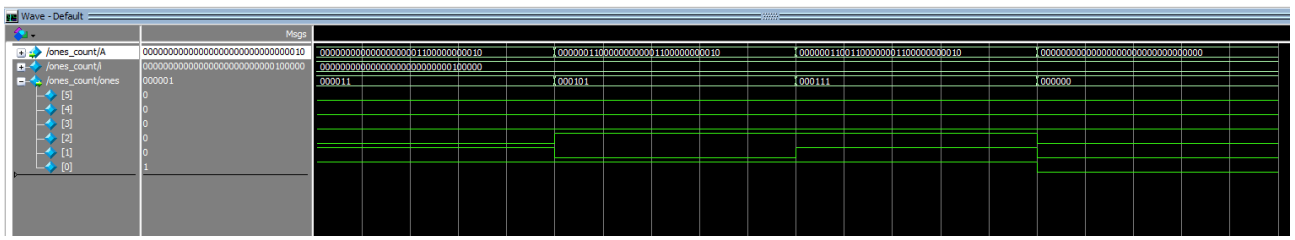
Solution:

Verilog code

question5 > ones_count.v

```
1   module ones_count(A, ones);
2       input [31:0] A;
3       output reg [5:0] ones;
4       integer i;
5       always@(A)
6         begin
7             ones = 0;  //initialize count variable.
8             for(i=0;i<32;i=i+1)   //for all the bits.
9                 ones = ones + A[i]; //Add the bit to the count.
10        end
11  endmodule
12
```

Simulation waveform



6. Implement the following function

F(v,w,x,y,z) = Σ (0, 2, 3, 4, 8, 21, 22, 29, 31)

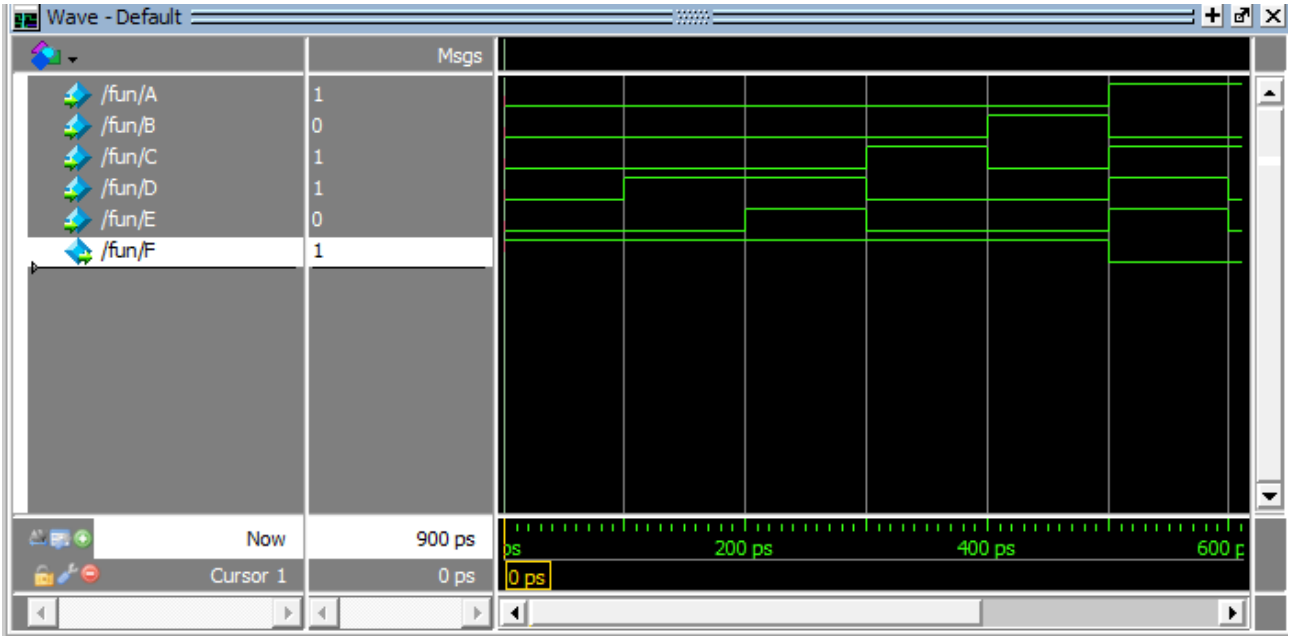
Solution:

Verilog code

question6 > fun.v

```
1   module fun(A,B,C,D,E,F);
2     input A,B,C,D,E;
3     output reg F;
4     always @(A,B,C,D,E) begin
5       // setting the value of F to 1 for the min terems
6         case ({A,B,C,D,E})
7         0 : F = 1'b1;
8         2 : F = 1'b1;
9         3 : F = 1'b1;
10        4 : F = 1'b1;
11        8 : F = 1'b1;
12        21 : F = 1'b1;
13        22 : F = 1'b1;
14        29 : F = 1'b1;
15        31 : F = 1'b1;
16        // setting the value as 0 for others
17        default: F = 0;
18    endcase
19    end
20  endmodule
```

Simulation waveform



7. 16-bit magnitude comparator.

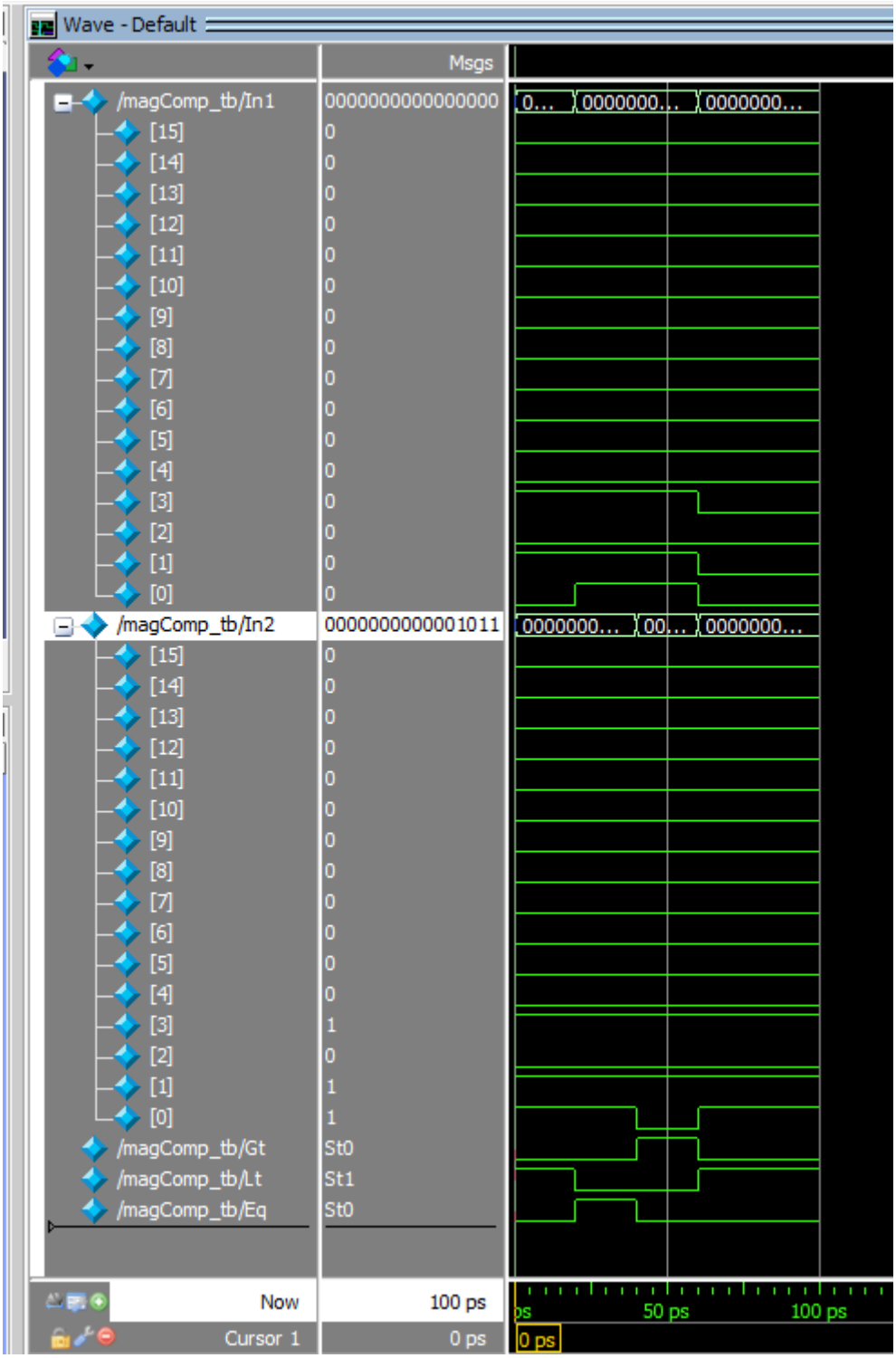
Solution:

Verilog code

question7 > magComp.v

```
1   module magComp ( In1, In2, Gt, Lt, Eq );
2     input [15:0]  In1, In2; //The two 16-bit Inputs In1 and In2
3     output reg  Gt, Lt, Eq; //The Outputs of comparison
4     always @ (In1 or In2) //Check the state of the input lines
5     begin
6       Gt <= ( In1 > In2 )? 1'b1 : 1'b0;
7       Lt <= ( In1 < In2 )? 1'b1 : 1'b0;
8       Eq <= ( In1 == In2)? 1'b1 : 1'b0;
9     end
10  endmodule
```

Simulation waveform



8. Count the number of leading 0's in an 8-bit number.

Solution:

Verilog code

question8 > zero_count_beg.v

```
1   module zero_count_beg(A, zeros);
2       input [0:7] A;
3       output reg [3:0] zeros;
4       integer i;
5       always@(A)
6         begin
7             zeros = 0;  //initialize count variable.
8             for(i=0; i<8 && A[i] == 1'b0; i=i+1)   //for all the bits.
9             begin
10              zeros = zeros + 1;
11            end
12        end
13  endmodule
14
```

Simulation waveform