Assignment_2

Ex1

Code:

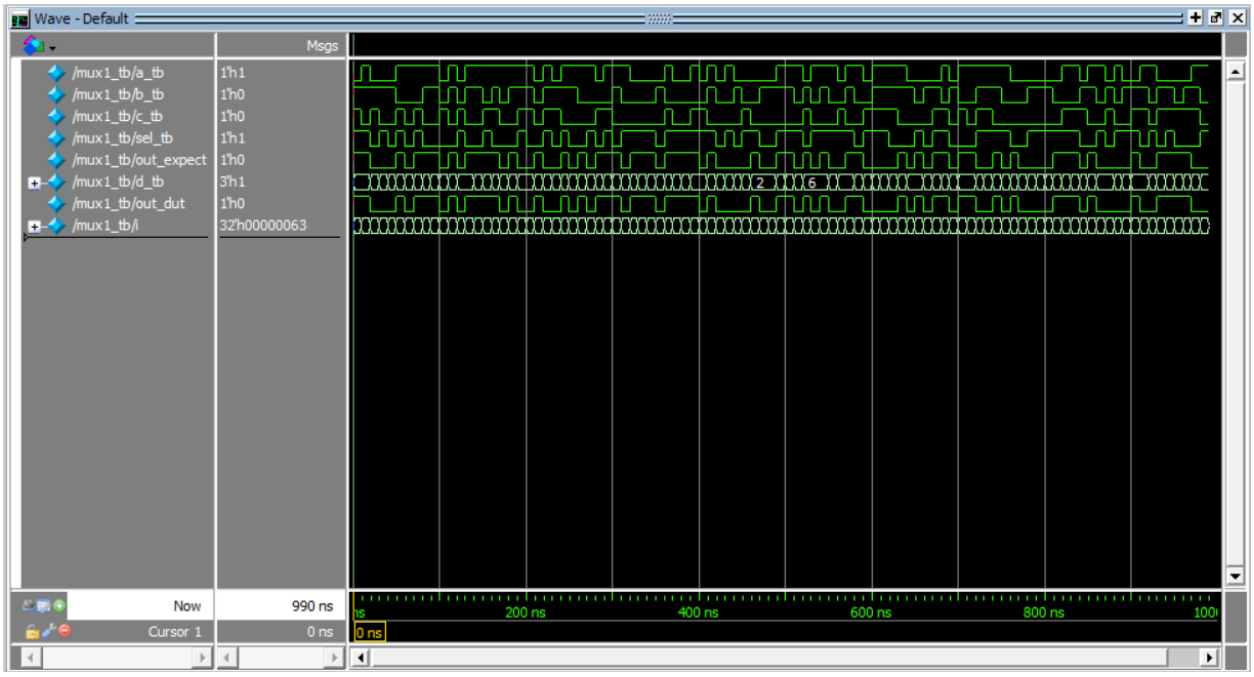
```
1   module mux1(A,B,C,sel,D,out,out_bar);
2   input A,B,C,sel;
3   input [2:0] D;
4   output reg out,out_bar;
5
6   always @(*) begin
7       if (sel == 0)begin
8           out= (D[0] & D[1]) | D[2];
9           out_bar= ~out;
10      end
11      else begin
12          out= ~(A ^ B ^ C);
13          out_bar= ~out;
14      end
15  end
16  endmodule
```

Output:

Testbench:

```
module mux1_tb();

reg a_tb, b_tb, c_tb, sel_tb, out_expect;
reg  [2:0] d_tb;
wire out_dut;

mux1 DUT(a_tb, b_tb, c_tb, sel_tb, d_tb , out_dut);

integer i;
initial begin
    for(i=0 ; i < 99 ; i= i + 1)begin
        a_tb= $random;
        b_tb= $random;
        c_tb= $random;
        sel_tb= $random;
        d_tb= $random;
        if (sel_tb == 0)begin
         out_expect= (d_tb[0] & d_tb[1]) | d_tb[2];
        end
        else begin
         out_expect= ~(a_tb ^ b_tb ^ c_tb);

        end
        #10
        if(out_expect != out_dut)begin
        $display("error_mux1 output is incorrect");
        $stop;
        end
    end
    $stop;
end
initial begin
$monitor("a= %b , b= %b , c= %b , d= %b ,sel= %b ,out_expect= %b",a_tb,b_tb,c_tb,d_tb,sel_tb,out_expect);
end
endmodule
```

Out:

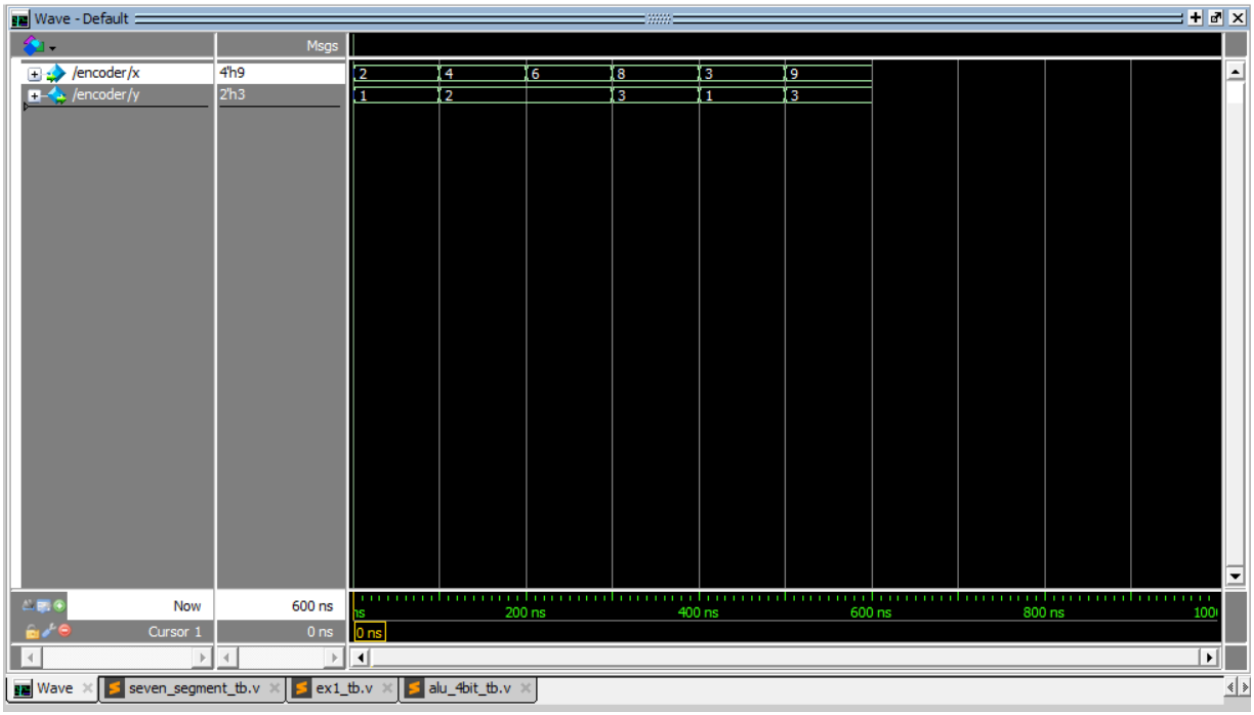


Ex2

Code:

```
1   module encoder(x,y);
2   input [3:0] x;
3   output reg [1:0] y;
4
5   always @(*)begin
6       if(x[3] == 1)
7       y= 2'b11;
8       else if({x[3],x[2]} == 2'b01)
9       y= 2'b10;
10      else if({x[3],x[2],x[1]} == 3'b001)
11      y= 2'b01;
12      else
13      y= 2'b00;
14
15  end
16  endmodule
```

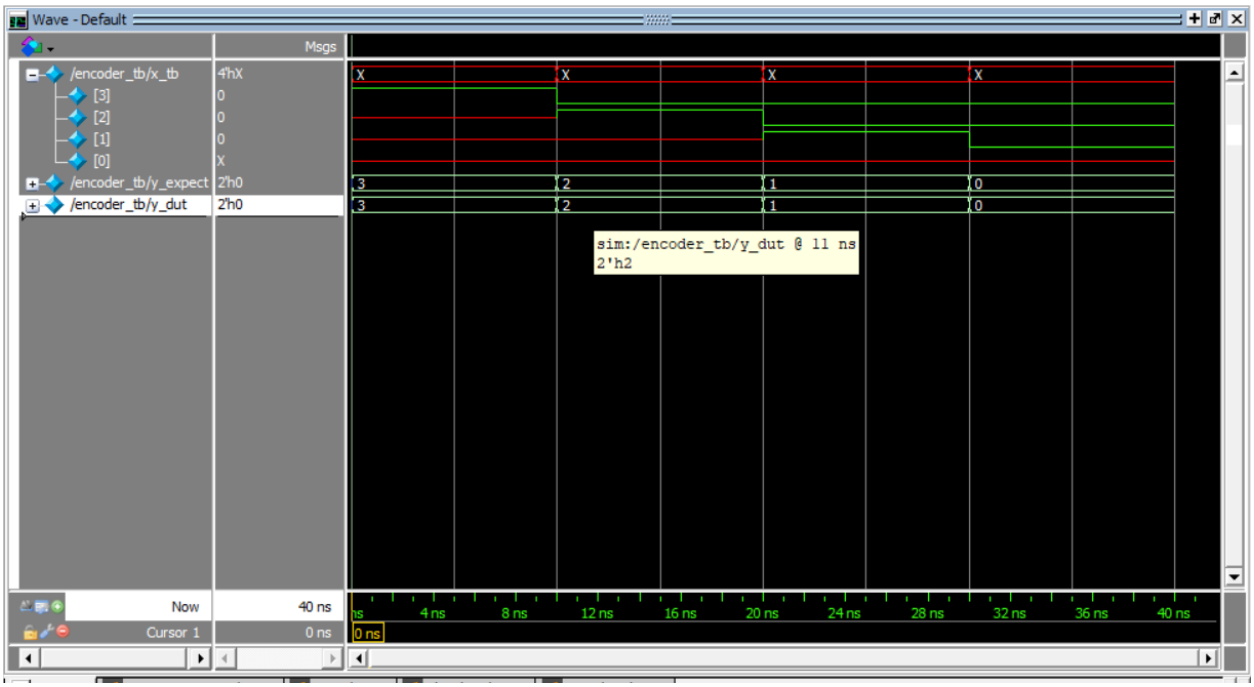
Out:



Testbench:

```
1  module encoder_tb();
2  reg [3:0] x_tb;
3  reg [1:0] y_expect;
4  wire [1:0] y_dut;
5
6  encoder DUT(x_tb,y_dut);
7
8  initial begin
9
10     #0 x_tb[3] = 1; y_expect= 3;
11     #10 x_tb[3]= 0; x_tb[2]= 1; y_expect=2;
12     #10 x_tb[3]= 0; x_tb[2]= 0; x_tb[1]= 1; y_expect= 1;
13     #10 x_tb[3]= 0; x_tb[2]= 0; x_tb[1]= 0; y_expect= 0;
14     #10
15     if(y_dut != y_expect)begin
16     $display("error-encoder output is incorrect");
17     $stop;
18     end
19     $stop;
20 end
21 initial begin
22 $monitor("x[0]=%b,x[1]=%b,x[2]=%b,x[3]=%b,y=%b",x_tb[0] ,x_tb[1] ,x_tb[2] ,x_tb[3] ,y_expect);
23 end
24 endmodule
```

Out:



Ex3

Code:

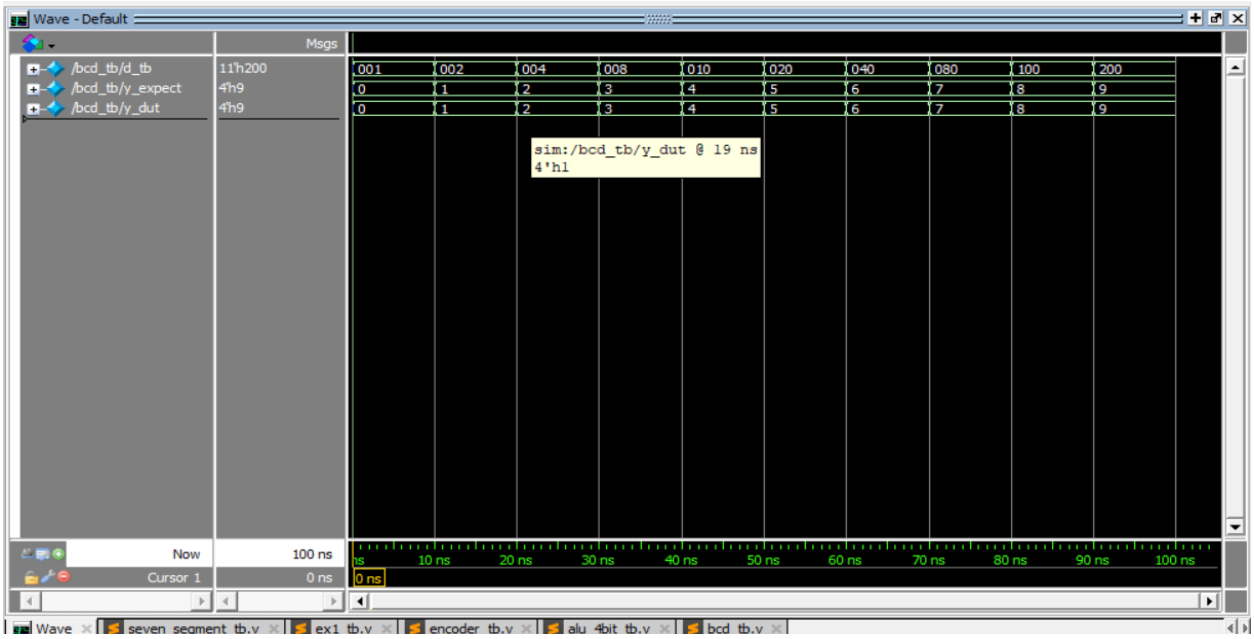
```
1   module bcd (D,Y);
2   input [9:0] D;
3   output reg [3:0] Y;
4
5   always @(*) begin
6       case(D)
7       10'b0000000001: Y= 0;
8       10'b0000000010: Y= 1;
9       10'b0000000100: Y= 2;
10      10'b0000001000: Y= 3;
11      10'b0000010000: Y= 4;
12      10'b0000100000: Y= 5;
13      10'b0001000000: Y= 6;
14      10'b0010000000: Y= 7;
15      10'b0100000000: Y= 8;
16      10'b1000000000: Y= 9;
17      default: Y= 0;
18      endcase
19  end
20  endmodule
```

Out:

Testbench:

```
module bcd_tb();
reg [10:0] d_tb;
reg [3:0] y_expect;
wire [3:0] y_dut;

bcd DUT(d_tb,y_dut);

initial begin
    #0 d_tb= 10'b0000000001; y_expect= 0;
    #10 d_tb= 10'b0000000010; y_expect= 1;
    #10 d_tb= 10'b0000000100; y_expect= 2;
    #10 d_tb= 10'b0000001000; y_expect= 3;
    #10 d_tb= 10'b0000010000; y_expect= 4;
    #10 d_tb= 10'b0000100000; y_expect= 5;
    #10 d_tb= 10'b0001000000; y_expect= 6;
    #10 d_tb= 10'b0010000000; y_expect= 7;
    #10 d_tb= 10'b0100000000; y_expect= 8;
    #10 d_tb= 10'b1000000000; y_expect= 9;
    #10
    if(y_dut != y_expect)begin
    $display("error-encoder output is incorrect");
    $stop;
    end
    $stop;
end
initial begin
$monitor("D=%b,y_expect=%b,y_dut=%b",d_tb ,y_expect ,y_dut );
end
endmodule
```

Out:

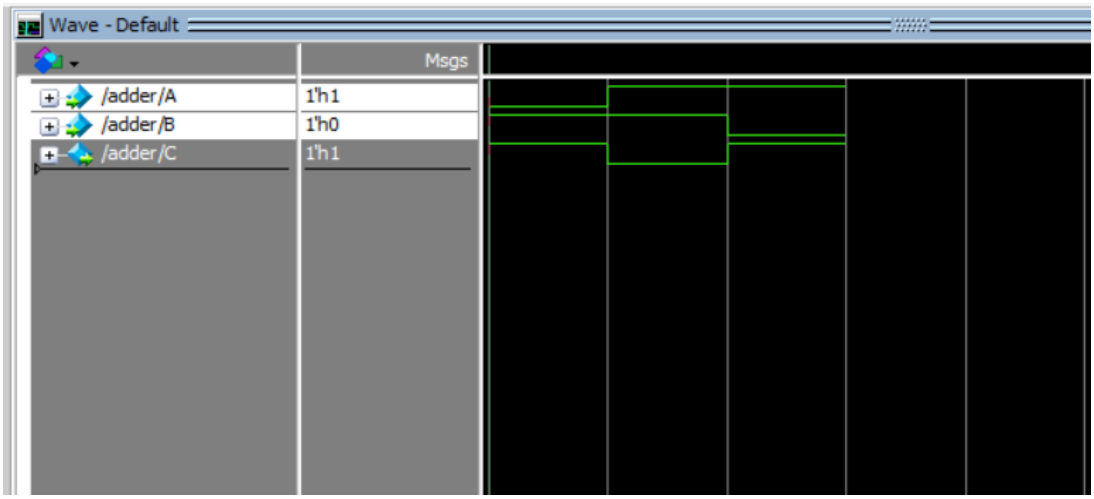


Ex4

Code:

```
1  module adder(A, B, C);
2  parameter N= 1;
3  input [N-1:0] A, B;
4  output [N-1:0] C;
5
6  assign C= A + B;
7
8  endmodule
```

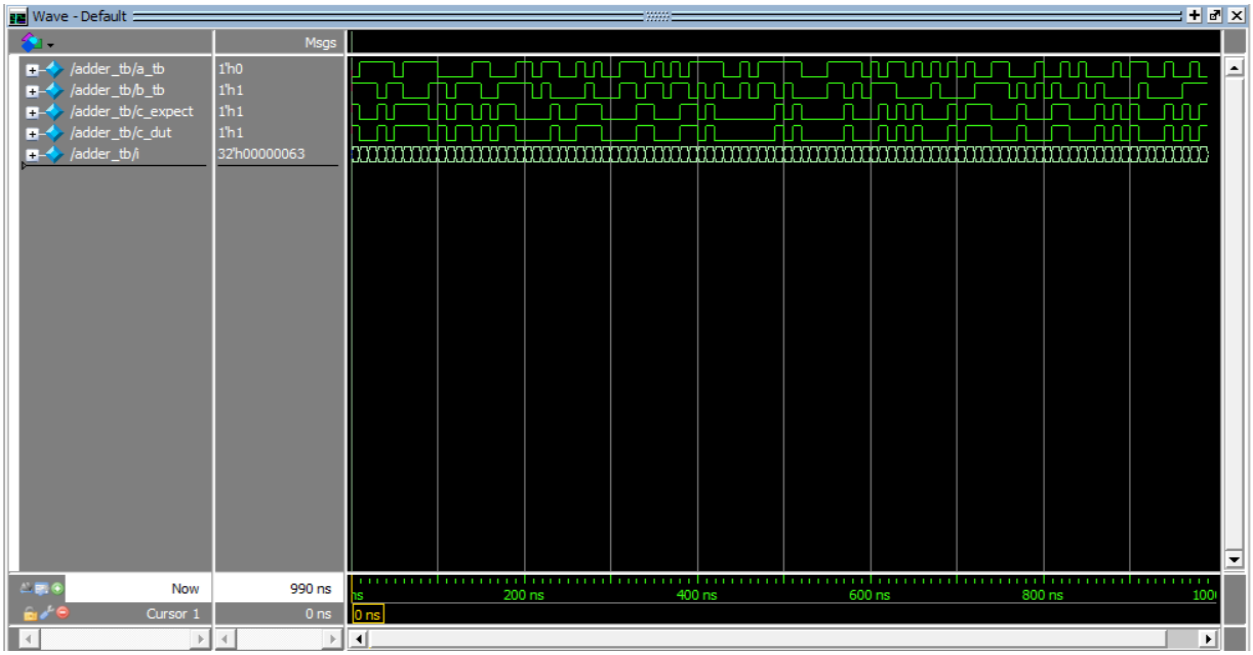
Out:



Testbench:

```
1   module adder_tb();
2   parameter N_tb= 1;
3   reg  [N_tb-1:0] a_tb, b_tb, c_expect;
4   wire[N_tb-1:0] c_dut;
5
6   adder #(1) DUT(a_tb,b_tb,c_dut);
7
8   integer i;
9   initial begin
10      for(i=0; i < 99 ; i = i + 1)begin
11          a_tb = $random;
12          b_tb = $random;
13          c_expect= a_tb + b_tb;
14          #10
15          if(c_dut != c_expect)begin
16          $display("error-adder output is incorrect");
17          $stop;
18          end
19      end
20      $stop;
21  end
22  initial begin
23  $monitor("a=%d, b=%d , c=%d",a_tb,b_tb,c_expect);
24  end
25  endmodule
```

Out:

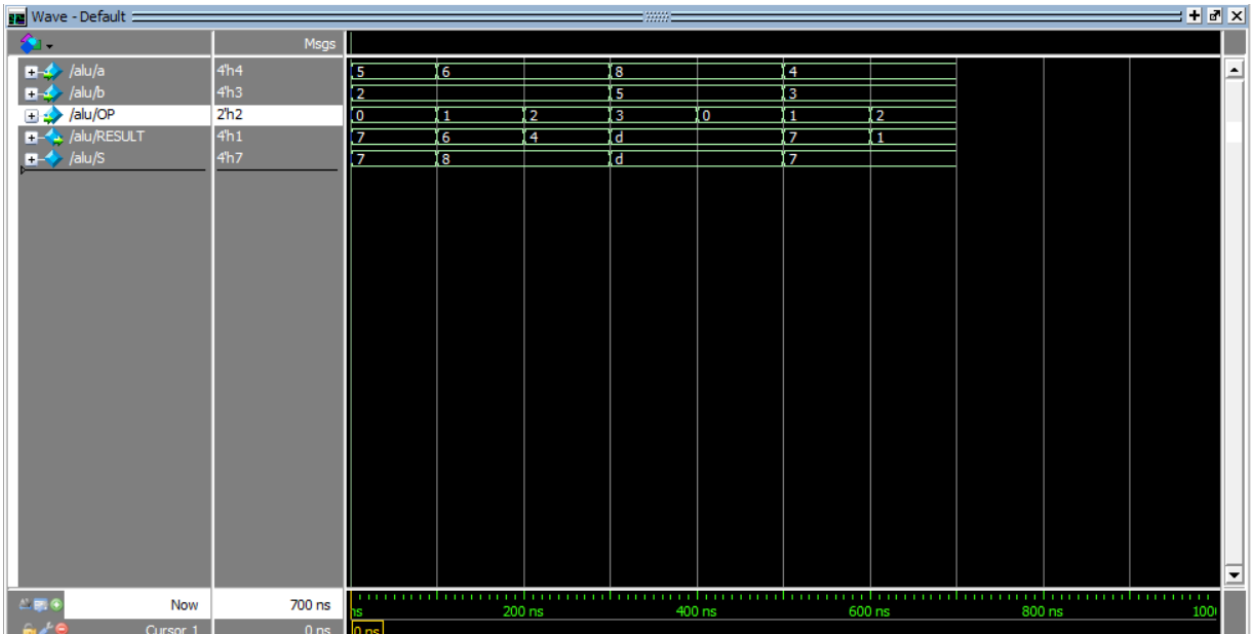


Ex5

Code:

```
1   module alu(a,b,OP,RESULT);
2   parameter N4= 4;
3   input  [N4-1:0] a,b;
4   input [1:0] OP;
5   output reg [N4-1:0] RESULT;
6   wire [N4-1:0] S;
7
8   adder #(.N(N4)) HA1(.A(a),.B(b),.C(S));
9
10
11  always @(*) begin
12      case(OP)
13      2'b00 : RESULT= S;
14      2'b10 : RESULT= a - b;
15      2'b01 : RESULT= a | b;
16      2'b11 : RESULT= a ^ b;
17      endcase
18  end
19  endmodule
```

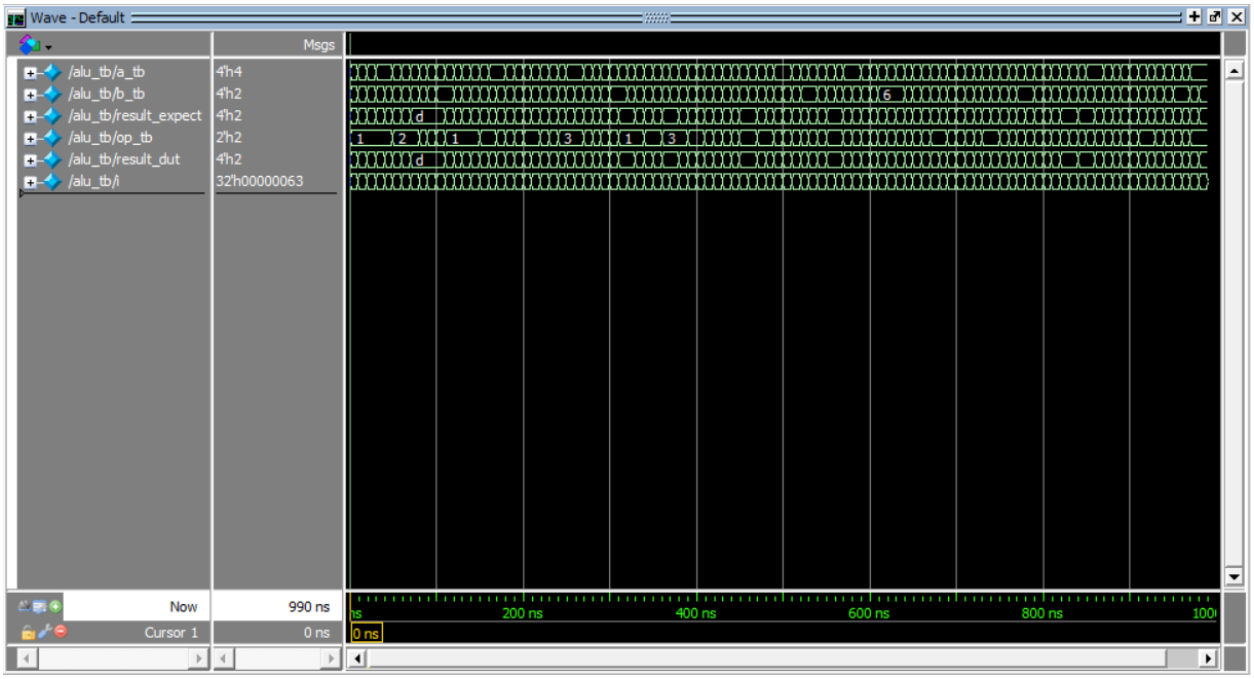
Out:



Testbench:

```
module alu_tb();
parameter N4_tb= 4;
reg [N4_tb-1:0] a_tb,b_tb,result_expect;
reg [1:0] op_tb;
wire [N4_tb - 1:0] result_dut;

alu #(N4_tb) DUT(a_tb,b_tb,op_tb,result_dut);

integer i;
initial begin
    for(i=0; i < 99; i= i + 1)begin
        a_tb=$random;
        b_tb=$random;
        op_tb=$random;
    case(op_tb)
    2'b00 : result_expect= a_tb + b_tb;
    2'b10 : result_expect= a_tb - b_tb;
    2'b01 : result_expect= a_tb | b_tb;
    2'b11 : result_expect= a_tb ^ b_tb;
    endcase
    #10
    if(result_expect != result_dut)begin
    $display("error-alu out is incorrect");
    $stop;
    end
    end
    $stop;
end
initial begin
$monitor("a=%b, b=%b , op=%b, result_dut=%b,result_expect=%b",a_tb,b_tb,op_tb, result_dut,result_expect);
end
endmodule

```

Out:

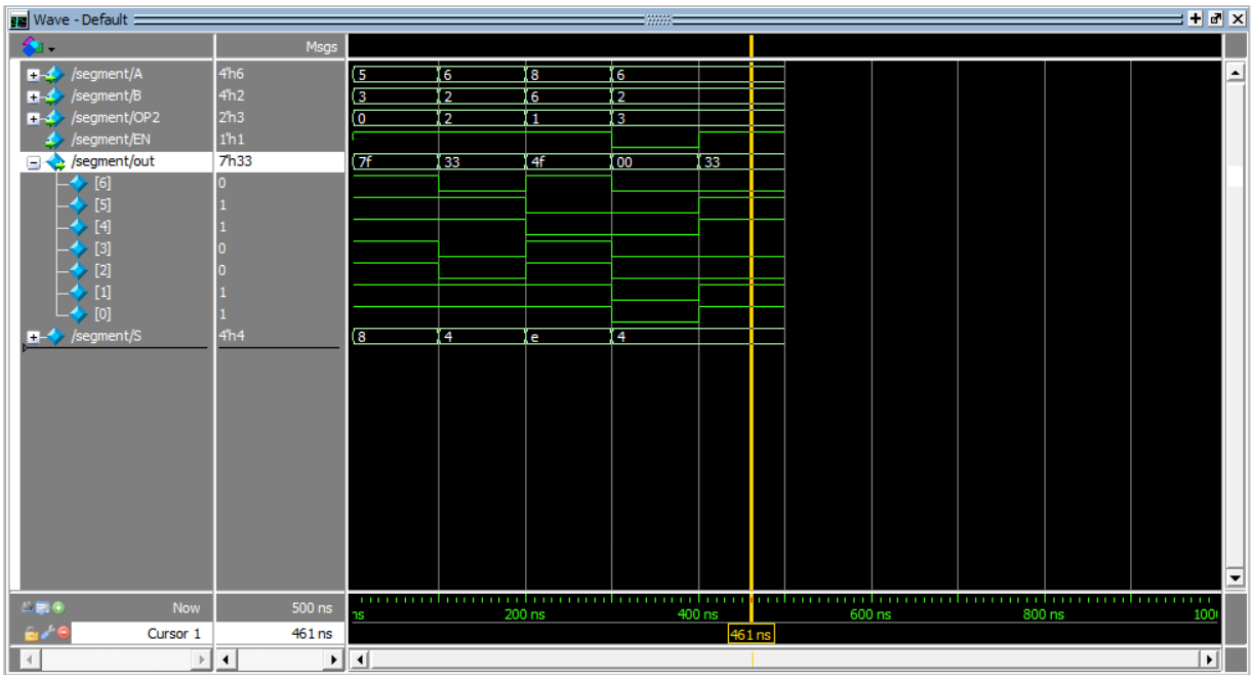


Ex6

Code:

```
module segment(A,B,OP2,EN,out);
parameter N7= 4;
input [N7-1:0] A, B;
 input [1:0] OP2;
input EN;
output reg [6:0] out;

wire [N7-1:0] S;

alu #(.N4(N7)) FA(.a(A),.b(B),.OP(OP2),.RESULT(S));

always @(*) begin
    if(EN == 1)begin
        case(S)
        4'b0000: out= 7'b1111110;
        4'b0001: out= 7'b0110000;
        4'b0010: out= 7'b1101101;
        4'b0011: out= 7'b1111001;
        4'b0100: out= 7'b0110011;
        4'b0101: out= 7'b1011011;
        4'b0110: out= 7'b1011111;
        4'b0111: out= 7'b1110000;
        4'b1000: out= 7'b1111111;
        4'b1001: out= 7'b1111011;
        4'b1010: out= 7'b1110111;
        4'b1011: out= 7'b0011111;
        4'b1100: out= 7'b1001110;
        4'b1101: out= 7'b0111101;
        4'b1110: out= 7'b1001111;
        4'b1111: out= 7'b1000111;
        endcase
    end
    else begin
        out=0;

    end
end
endmodule
```

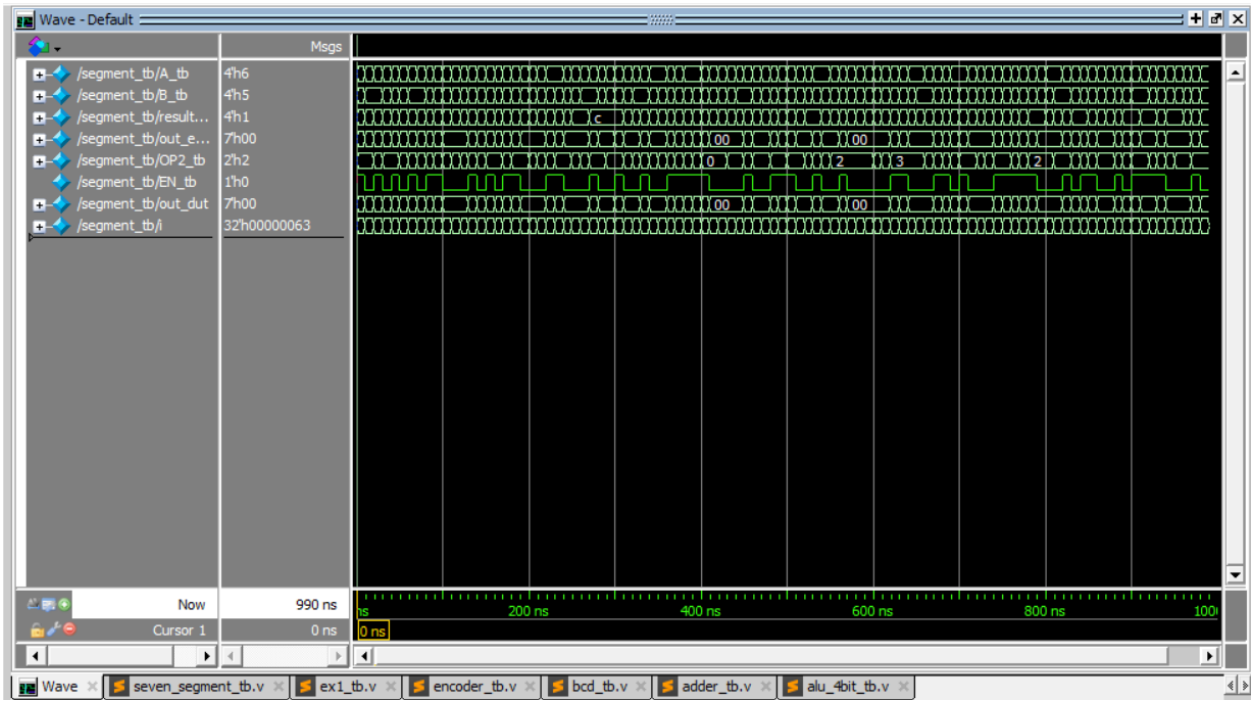
Out:


Wave - Default
Msgs
/segment/A
4'h6
/segment/B
4'h2
/segment/OP2
2'h3
/segment/EN
1'h1
/segment/out
7'h33
[6]
[5]
[4]
[3]
[2]
[1]
[0]
/segment/S
4'h4
5
6
8
6
3
2
6
2
0
2
1
3
7f
33
4f
00
33
1
1
0
0
1
1
8
4
e
4
Now
500 ns
Cursor 1
461 ns
200 ns
400 ns
600 ns
800 ns
461 ns

Testbench:

```
module segment_tb();
parameter N7_tb= 4;
reg  [N7_tb-1:0] A_tb, B_tb, result_expect;
reg [6:0] out_expect;
reg [1:0] OP2_tb;
reg EN_tb;

wire [6:0] out_dut;

segment #(N7_tb) DUT(A_tb,B_tb,OP2_tb,EN_tb,out_dut);

integer i;
initial begin
    for(i= 0 ; i < 99 ; i = i + 1)begin
        A_tb=$random;
        B_tb=$random;
        OP2_tb=$random;
        EN_tb= $random;
        case(OP2_tb)
            2'b00 : result_expect= A_tb + B_tb;
            2'b10 : result_expect= A_tb - B_tb;
            2'b01 : result_expect= A_tb | B_tb;
            2'b11 : result_expect= A_tb ^ B_tb;
        endcase
        if(EN_tb == 1)begin
            case(result_expect)
            4'b0000: out_expect= 7'b1111110;
            4'b0001: out_expect= 7'b0110000;
            4'b0010: out_expect= 7'b1101101;
            4'b0011: out_expect= 7'b1111001;
            4'b0100: out_expect= 7'b0110011;
            4'b0101: out_expect= 7'b1011011;
            4'b0110: out_expect= 7'b1011111;
            4'b0111: out_expect= 7'b1110000;
            4'b1000: out_expect= 7'b1111111;
            4'b1001: out_expect= 7'b1111011;
            4'b1010: out_expect= 7'b1110111;
            4'b1011: out_expect= 7'b0011111;
            4'b1100: out_expect= 7'b1001110;
            4'b1101: out_expect= 7'b0111101;
            4'b1110: out_expect= 7'b1001111;
            4'b1111: out_expect= 7'b1000111;
            endcase
        end
        else
        out_expect= 0;
        #10
        if(out_expect != out_dut)begin
        $display("error- 7 segment out is incorrect");
        $stop;
        end
    end
    $stop;
end
initial begin
$monitor("a=%b, b=%b , op=%b, EN=%b, result=%b, out_dut=%b,out_expect=%b",A_tb,B_tb,OP2_tb,EN_tb,result_expect,out_dut,out_expect);
end
endmodule
```

Out:


Wave - Default
Msgs
/segment_tb/A_tb
/segment_tb/B_tb
/segment_tb/result...
/segment_tb/out_e...
/segment_tb/OP2_tb
/segment_tb/EN_tb
/segment_tb/out_dut
/segment_tb/i
4'h6
4'h5
4'h1
7'h00
2'h2
1'h0
7'h00
32'h00000063
Now
990 ns
Cursor 1
0 ns
200 ns
400 ns
600 ns
800 ns
0 ns
Wave
seven_segment_tb.v
ex1_tb.v
encoder_tb.v
bcd_tb.v
adder_tb.v
alu_4bit_tb.v