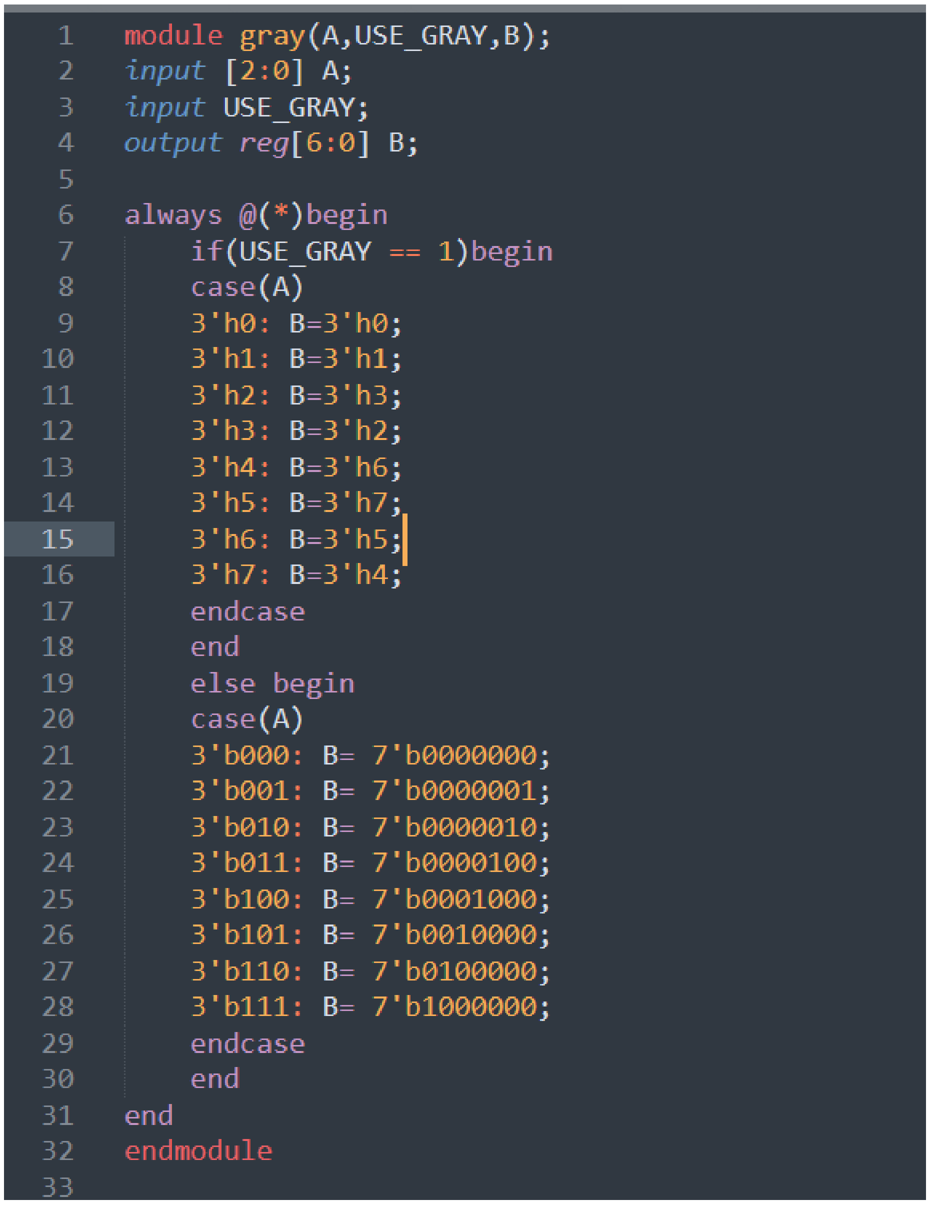
Assignment_2 (Extra)

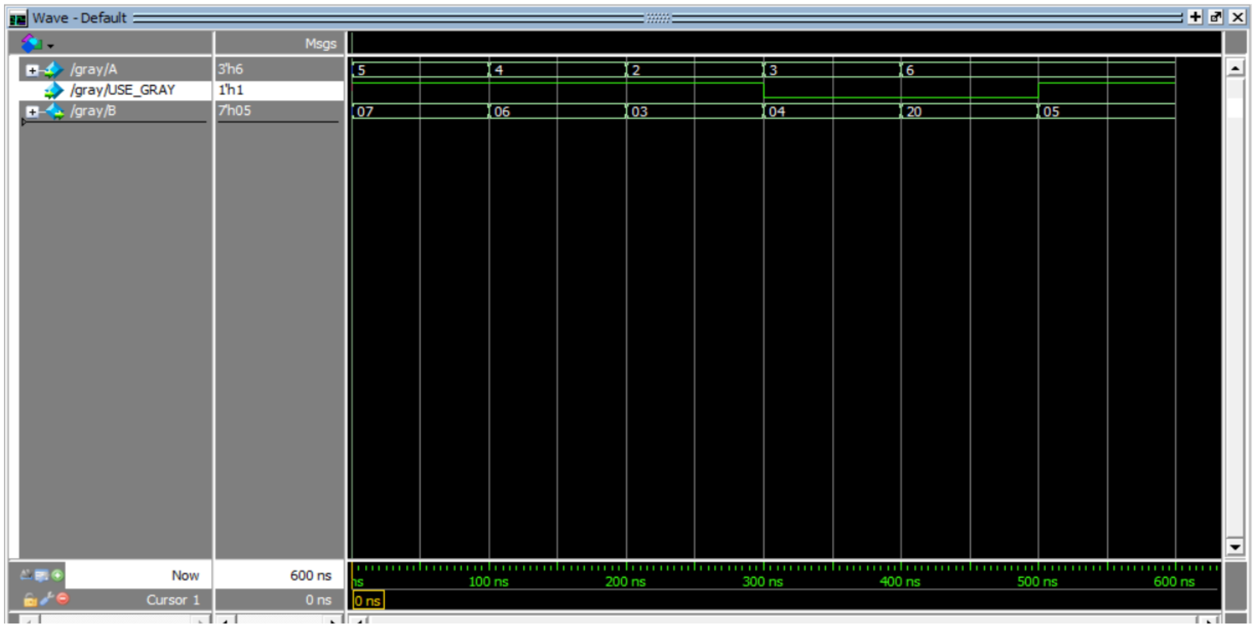
Ex1

Code:

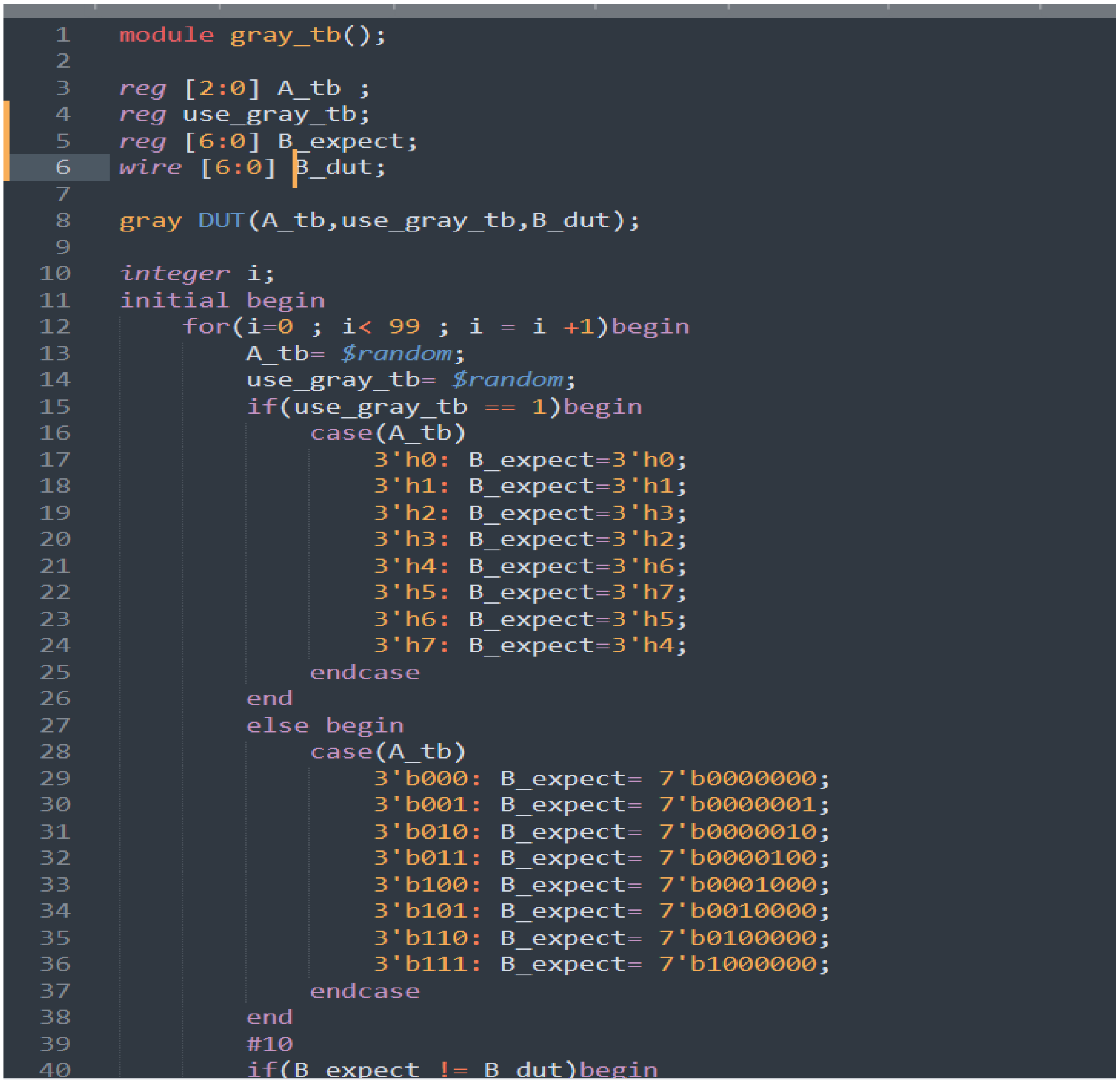


```
module gray(A,USE_GRAY,B);
input [2:0] A;
input USE_GRAY;
output reg[6:0] B;

always @(*)begin
    if(USE_GRAY == 1)begin
    case(A)
    3'h0: B=3'h0;
    3'h1: B=3'h1;
    3'h2: B=3'h3;
    3'h3: B=3'h2;
    3'h4: B=3'h6;
    3'h5: B=3'h7;
    3'h6: B=3'h5;
    3'h7: B=3'h4;
    endcase
    end
    else begin
    case(A)
    3'b000: B= 7'b0000000;
    3'b001: B= 7'b0000001;
    3'b010: B= 7'b0000010;
    3'b011: B= 7'b0000100;
    3'b100: B= 7'b0001000;
    3'b101: B= 7'b0010000;
    3'b110: B= 7'b0100000;
    3'b111: B= 7'b1000000;
    endcase
    end
end
endmodule
```

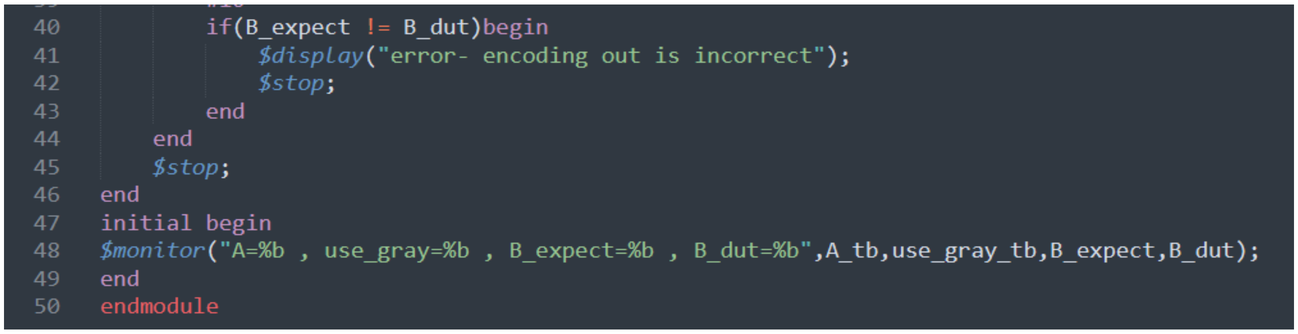
Out:



Testbench:

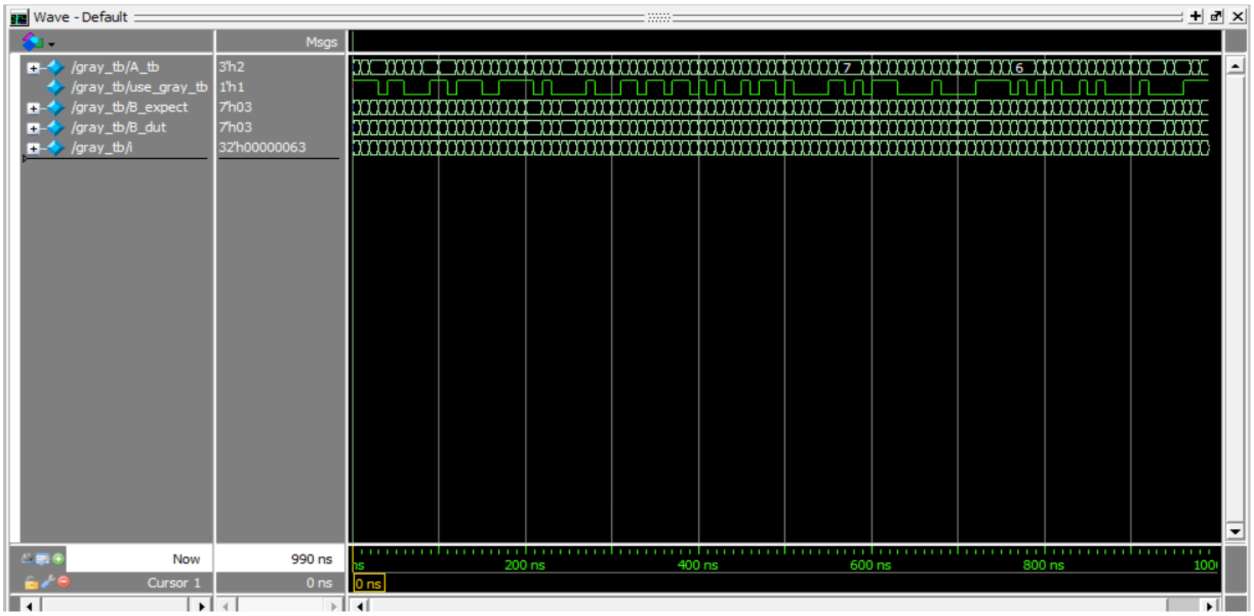


```verilog
module gray_tb();

reg [2:0] A_tb ;
reg use_gray_tb;
reg [6:0] B_expect;
wire [6:0] B_dut;

gray DUT(A_tb,use_gray_tb,B_dut);

integer i;
initial begin
    for(i=0 ; i< 99 ; i = i +1)begin
        A_tb= $random;
        use_gray_tb= $random;
        if(use_gray_tb == 1)begin
            case(A_tb)
                3'h0: B_expect=3'h0;
                3'h1: B_expect=3'h1;
                3'h2: B_expect=3'h3;
                3'h3: B_expect=3'h2;
                3'h4: B_expect=3'h6;
                3'h5: B_expect=3'h7;
                3'h6: B_expect=3'h5;
                3'h7: B_expect=3'h4;
            endcase
        end
        else begin
            case(A_tb)
                3'b000: B_expect= 7'b0000000;
                3'b001: B_expect= 7'b0000001;
                3'b010: B_expect= 7'b0000010;
                3'b011: B_expect= 7'b0000100;
                3'b100: B_expect= 7'b0001000;
                3'b101: B_expect= 7'b0010000;
                3'b110: B_expect= 7'b0100000;
                3'b111: B_expect= 7'b1000000;
            endcase
        end
        #10
```



```verilog
        if(B_expect != B_dut)begin
            $display("error- encoding out is incorrect");
            $stop;
        end
    end
    $stop;
end
initial begin
$monitor("A=%b , use_gray=%b , B_expect=%b , B_dut=%b",A_tb,use_gray_tb,B_expect,B_dut);
end
endmodule
```

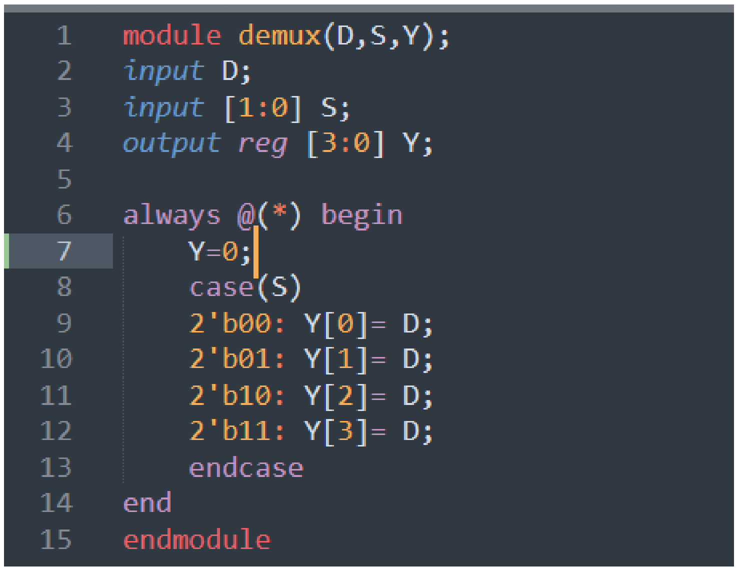
Out:



```
/gray_tb/A_tb          3'h2
/gray_tb/use_gray_tb   1'h1
/gray_tb/B_expect      7'h03
/gray_tb/B_dut         7'h03
/gray_tb/i             32'h00000063

Now        990 ns
Cursor 1   0 ns
```

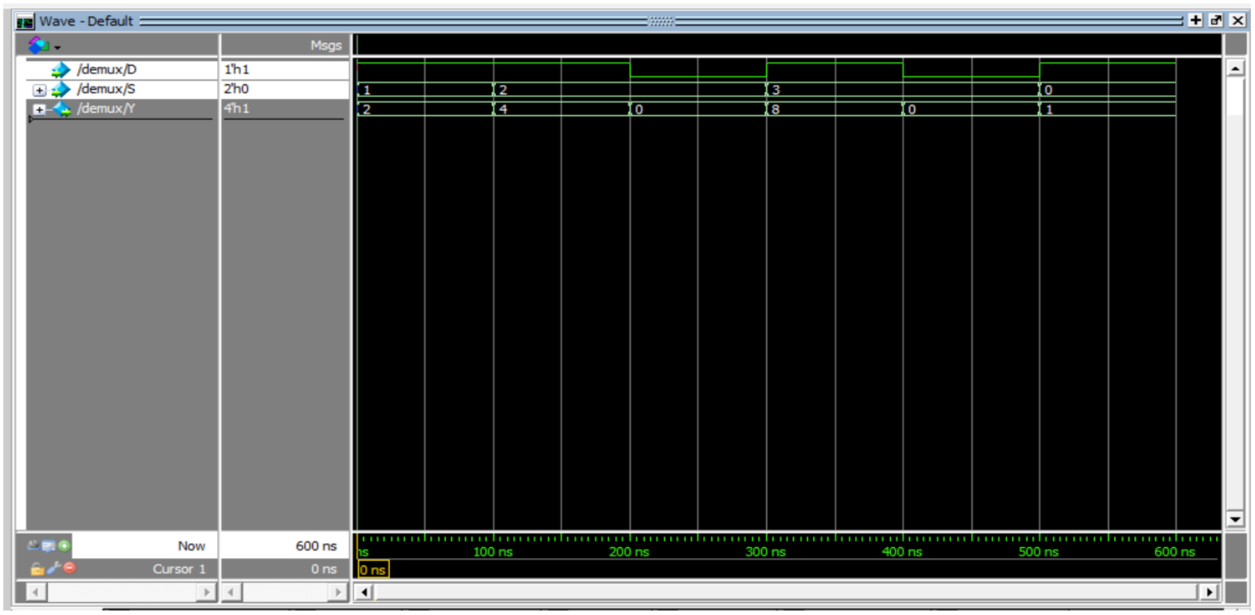
Ex2:

Code:

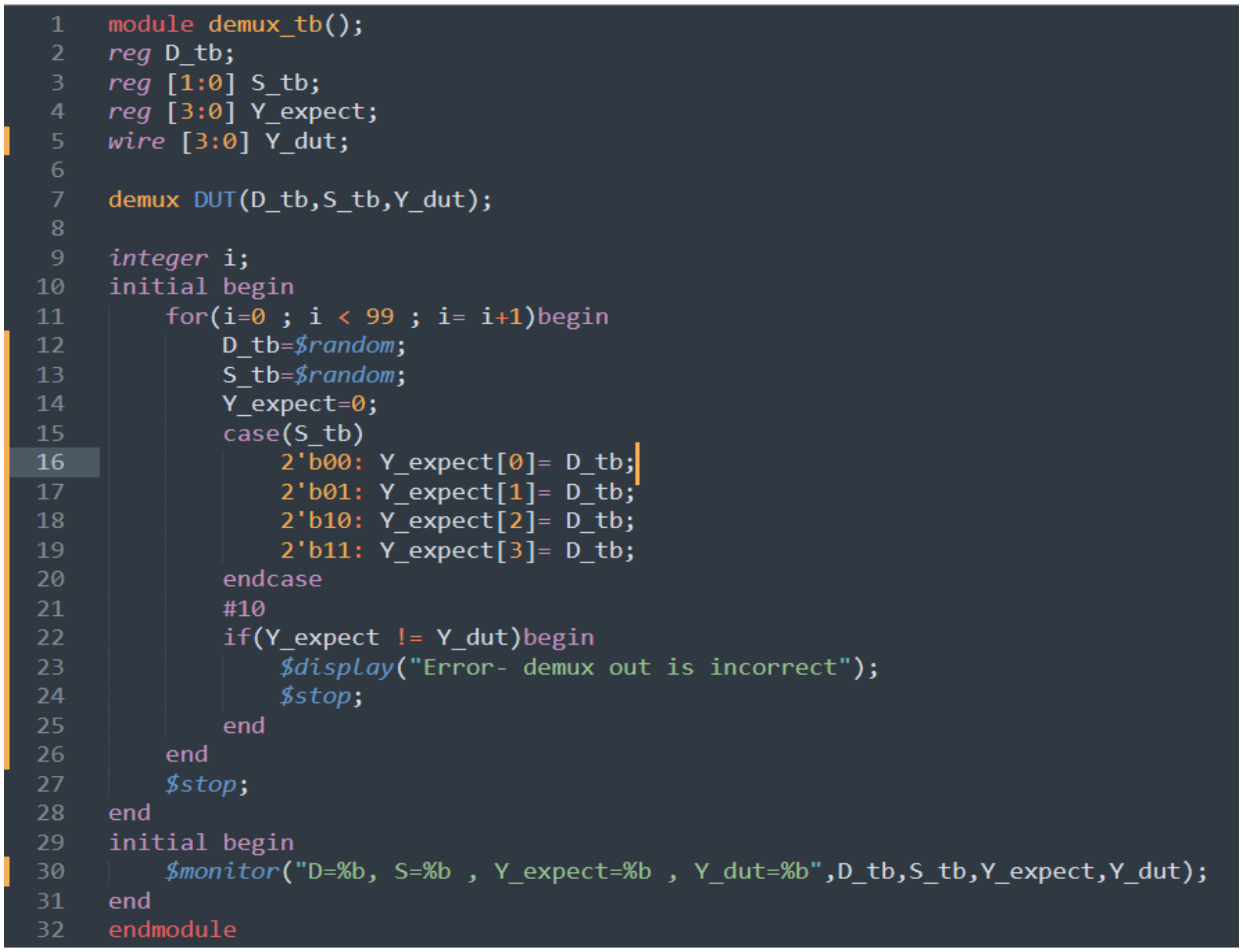


```
module demux(D,S,Y);
input D;
input [1:0] S;
output reg [3:0] Y;

always @(*) begin
    Y=0;
    case(S)
    2'b00: Y[0]= D;
    2'b01: Y[1]= D;
    2'b10: Y[2]= D;
    2'b11: Y[3]= D;
    endcase
end
endmodule
```

Out:

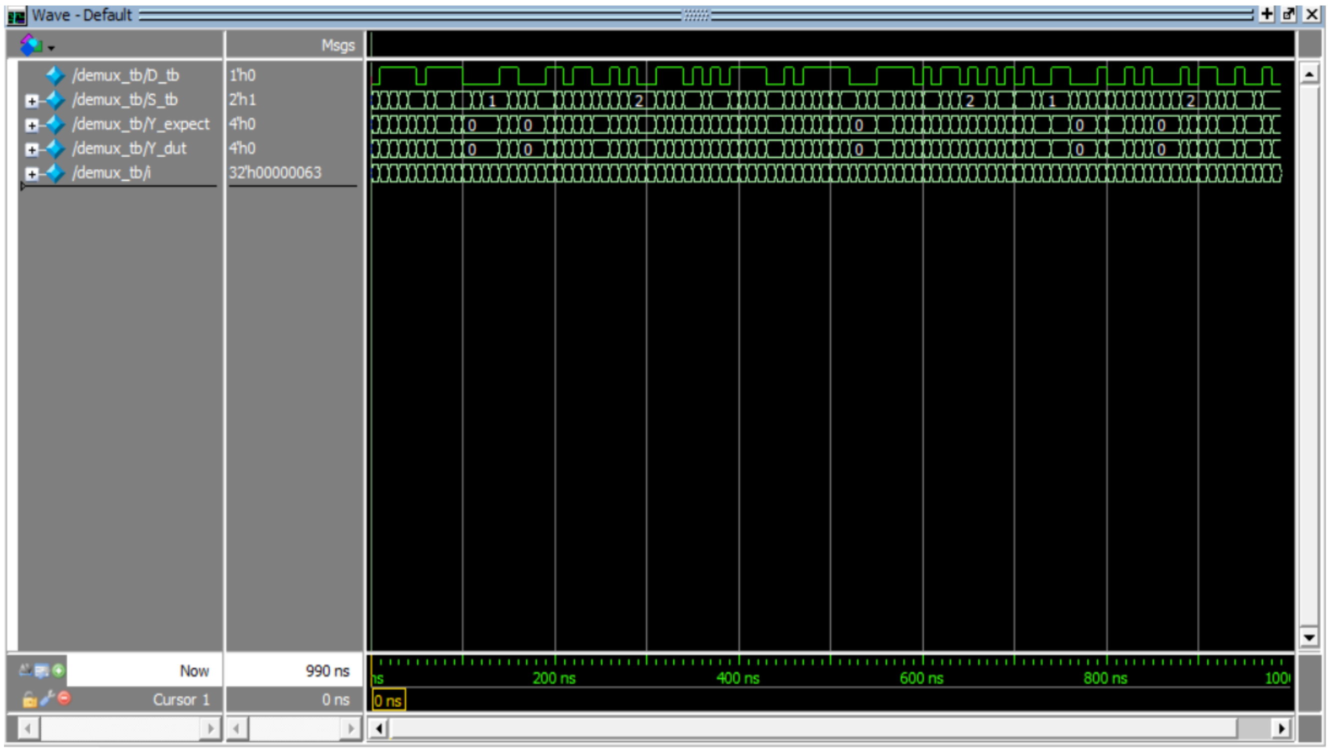


Testbench:



```
module demux_tb();
reg D_tb;
reg [1:0] S_tb;
reg [3:0] Y_expect;
wire [3:0] Y_dut;

demux DUT(D_tb,S_tb,Y_dut);

integer i;
initial begin
    for(i=0 ; i < 99 ; i= i+1)begin
        D_tb=$random;
        S_tb=$random;
        Y_expect=0;
        case(S_tb)
            2'b00: Y_expect[0]= D_tb;
            2'b01: Y_expect[1]= D_tb;
            2'b10: Y_expect[2]= D_tb;
            2'b11: Y_expect[3]= D_tb;
        endcase
        #10
        if(Y_expect != Y_dut)begin
            $display("Error- demux out is incorrect");
            $stop;
        end
    end
    $stop;
end
initial begin
    $monitor("D=%b, S=%b , Y_expect=%b , Y_dut=%b",D_tb,S_tb,Y_expect,Y_dut);
end
endmodule
```

Out:



```
/demux_tb/D_tb        1'h0
/demux_tb/S_tb        2'h1
/demux_tb/Y_expect    4'h0
/demux_tb/Y_dut       4'h0
/demux_tb/i           32'h00000063
Now                   990 ns
Cursor 1              0 ns
```