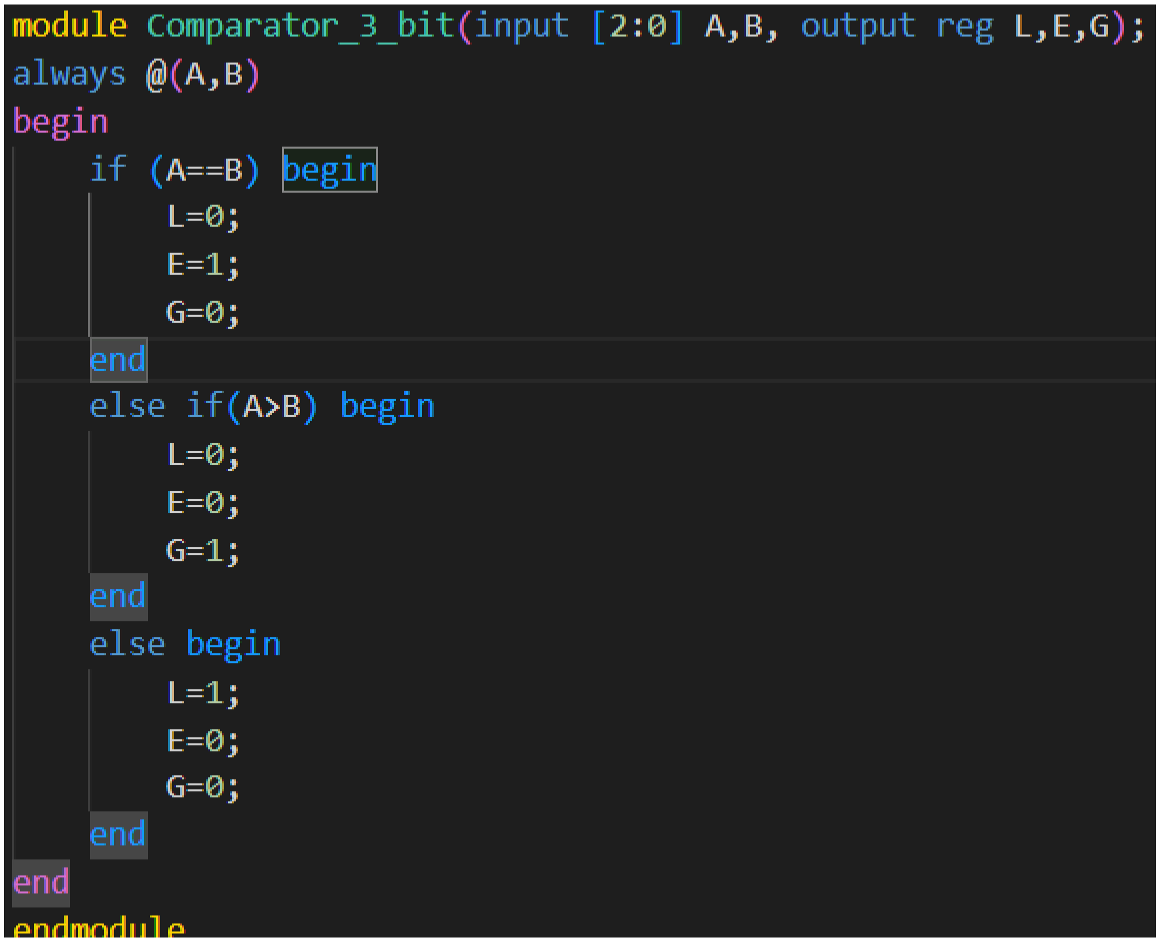
COS ASSIGNMENT-2 REPORT

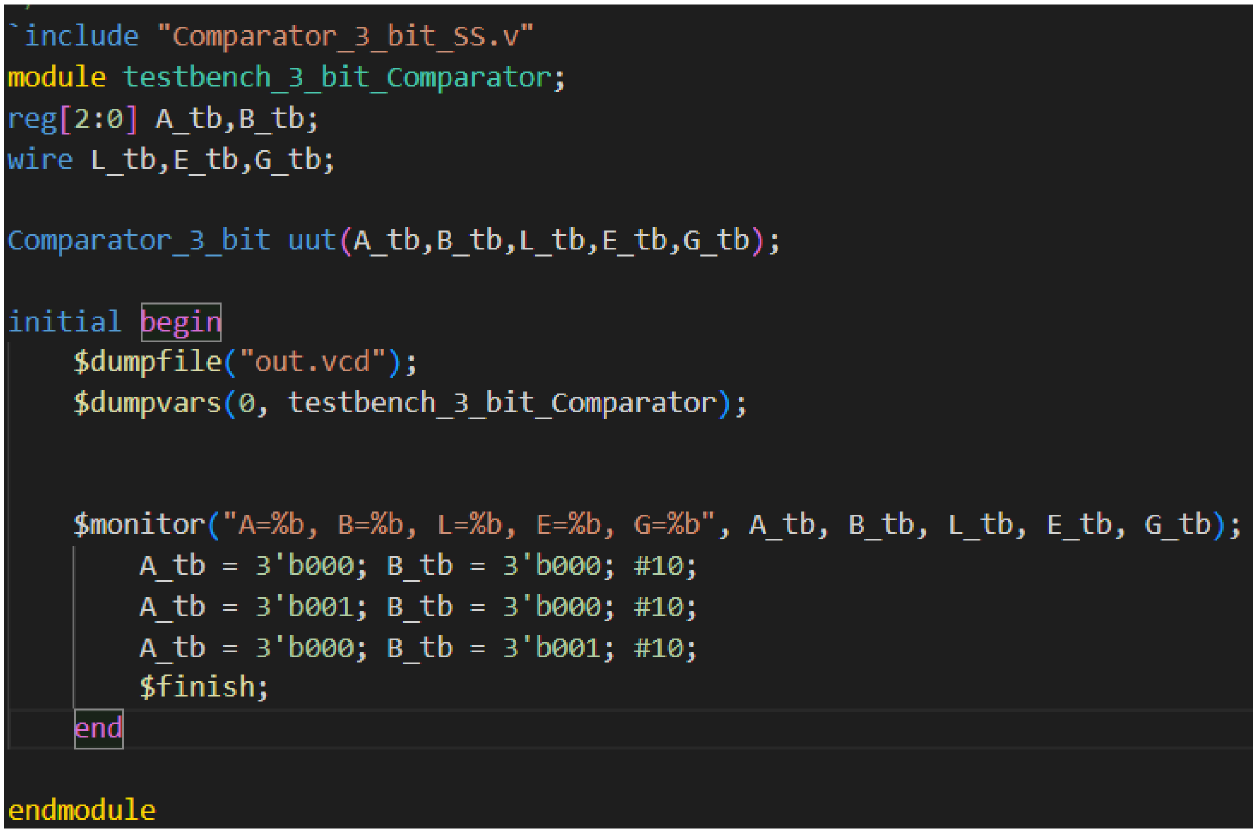
Group Member 1: Sahil kumar
Group Member 2: Shravan Ram

1.Write a verilog code for a 3-bit comparator circuit along with the test bench.
Code:



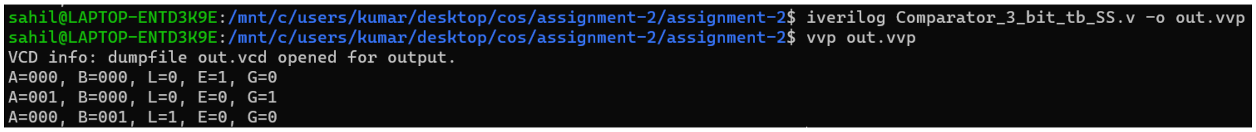
```
module Comparator_3_bit(input [2:0] A,B, output reg L,E,G);
always @(A,B)
begin
    if (A==B) begin
        L=0;
        E=1;
        G=0;
    end
    else if(A>B) begin
        L=0;
        E=0;
        G=1;
    end
    else begin
        L=1;
        E=0;
        G=0;
    end
end
endmodule
```

Test Bench:



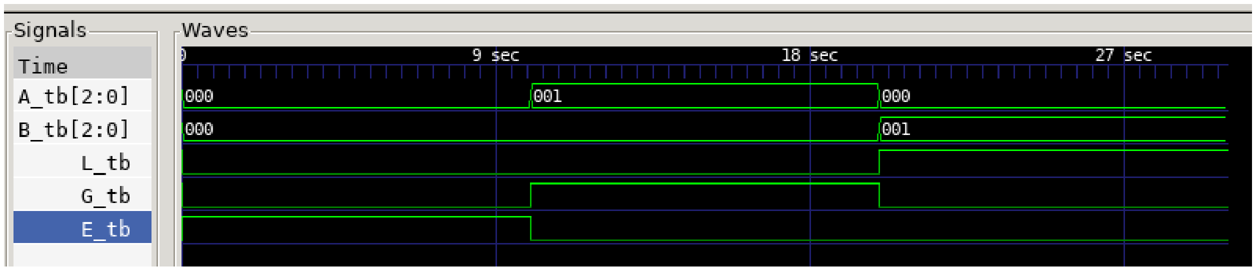
```
`include "Comparator_3_bit_SS.v"
module testbench_3_bit_Comparator;
reg[2:0] A_tb,B_tb;
wire L_tb,E_tb,G_tb;

Comparator_3_bit uut(A_tb,B_tb,L_tb,E_tb,G_tb);

initial begin
    $dumpfile("out.vcd");
    $dumpvars(0, testbench_3_bit_Comparator);


    $monitor("A=%b, B=%b, L=%b, E=%b, G=%b", A_tb, B_tb, L_tb, E_tb, G_tb);
        A_tb = 3'b000; B_tb = 3'b000; #10;
        A_tb = 3'b001; B_tb = 3'b000; #10;
        A_tb = 3'b000; B_tb = 3'b001; #10;
        $finish;
    end

endmodule
```

Terminal Output:

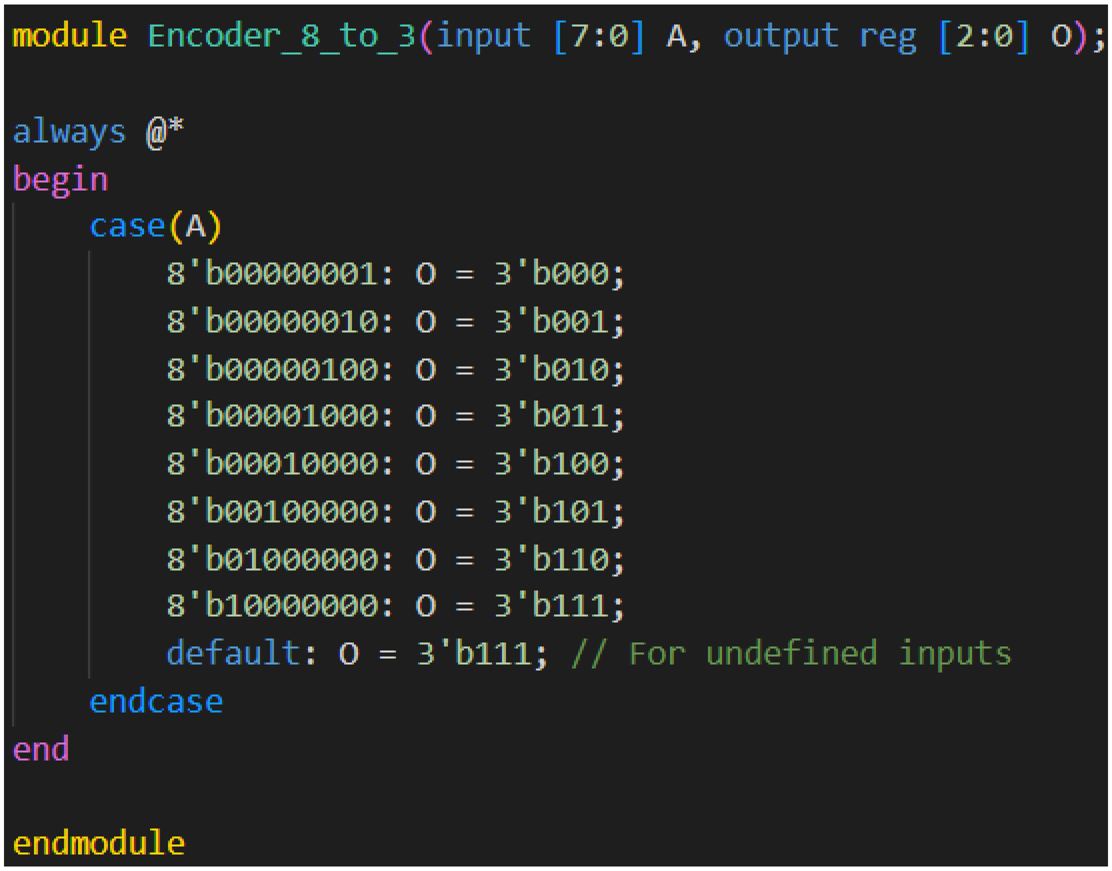


```
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ iverilog Comparator_3_bit_tb_SS.v -o out.vvp
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ vvp out.vvp
VCD info: dumpfile out.vcd opened for output.
A=000, B=000, L=0, E=1, G=0
A=001, B=000, L=0, E=0, G=1
A=000, B=001, L=1, E=0, G=0
```

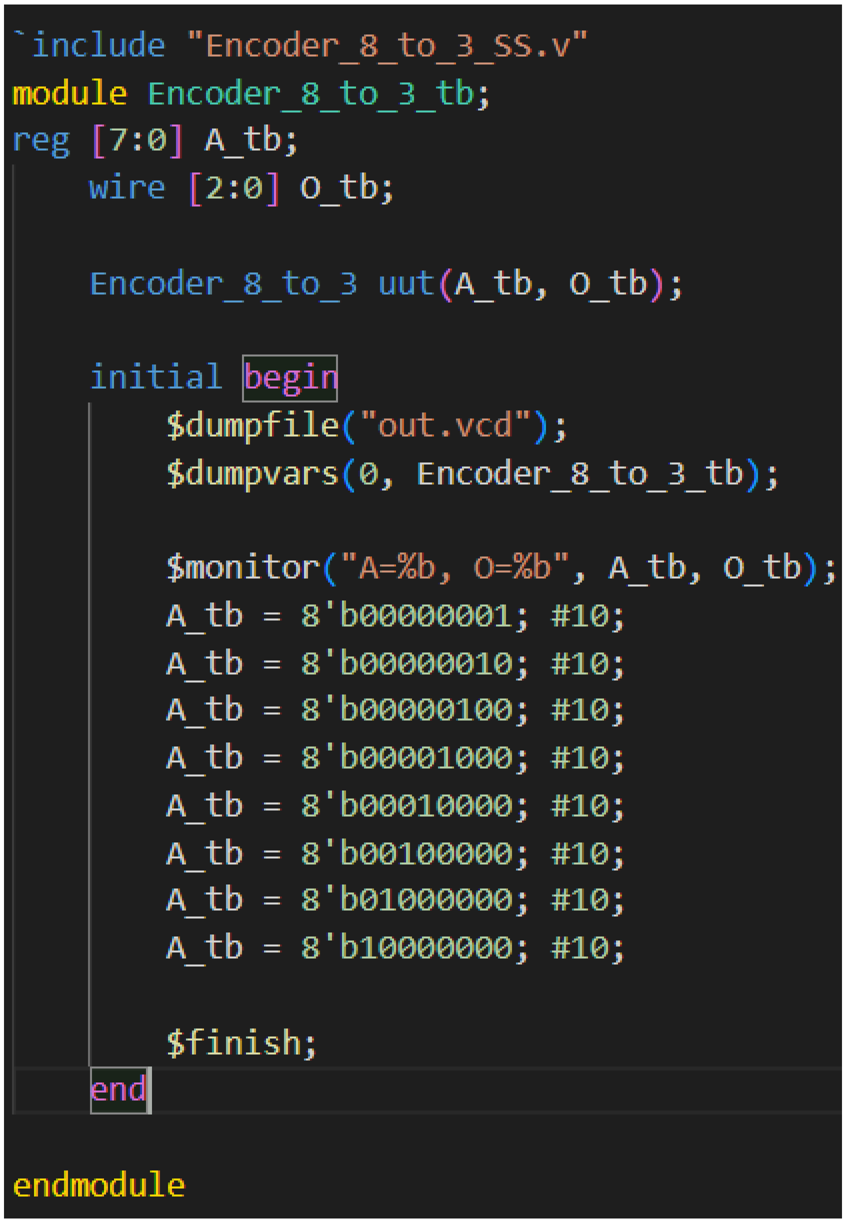
GTK wave output:



2. Write a verilog code for 8 to 3 Encoder circuit along with the test bench.
Code:

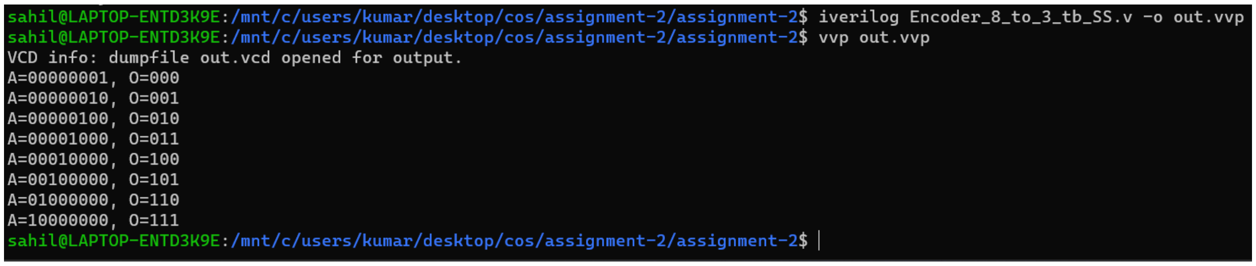

```
module Encoder_8_to_3(input [7:0] A, output reg [2:0] O);

always @*
begin
    case(A)
        8'b00000001: O = 3'b000;
        8'b00000010: O = 3'b001;
        8'b00000100: O = 3'b010;
        8'b00001000: O = 3'b011;
        8'b00010000: O = 3'b100;
        8'b00100000: O = 3'b101;
        8'b01000000: O = 3'b110;
        8'b10000000: O = 3'b111;
        default: O = 3'b111; // For undefined inputs
    endcase
end

endmodule
```

Testbench:



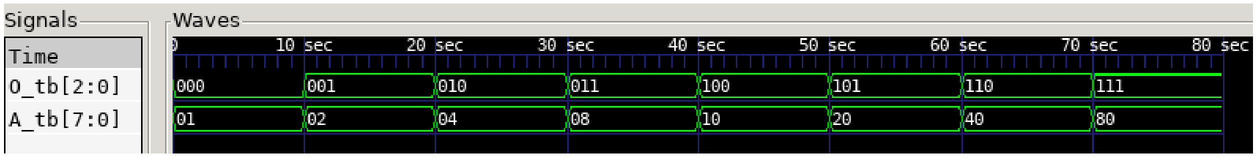
```verilog
`include "Encoder_8_to_3_SS.v"
module Encoder_8_to_3_tb;
reg [7:0] A_tb;
    wire [2:0] O_tb;

    Encoder_8_to_3 uut(A_tb, O_tb);

    initial begin
        $dumpfile("out.vcd");
        $dumpvars(0, Encoder_8_to_3_tb);

        $monitor("A=%b, O=%b", A_tb, O_tb);
        A_tb = 8'b00000001; #10;
        A_tb = 8'b00000010; #10;
        A_tb = 8'b00000100; #10;
        A_tb = 8'b00001000; #10;
        A_tb = 8'b00010000; #10;
        A_tb = 8'b00100000; #10;
        A_tb = 8'b01000000; #10;
        A_tb = 8'b10000000; #10;

        $finish;
    end

endmodule
```

Terminal Output:

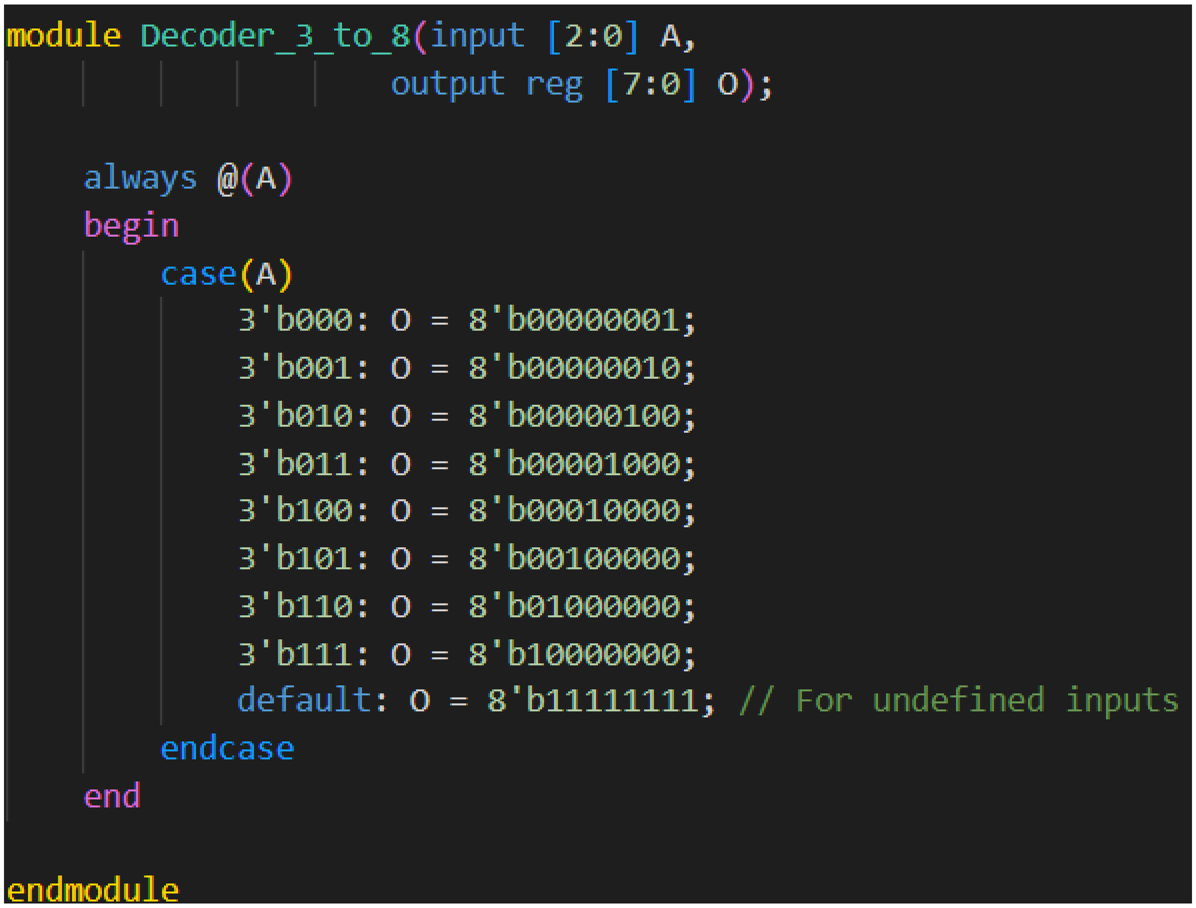


```
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ iverilog Encoder_8_to_3_tb_SS.v -o out.vvp
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ vvp out.vvp
VCD info: dumpfile out.vcd opened for output.
A=00000001, O=000
A=00000010, O=001
A=00000100, O=010
A=00001000, O=011
A=00010000, O=100
A=00100000, O=101
A=01000000, O=110
A=10000000, O=111
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ 
```

GTK wave Output:

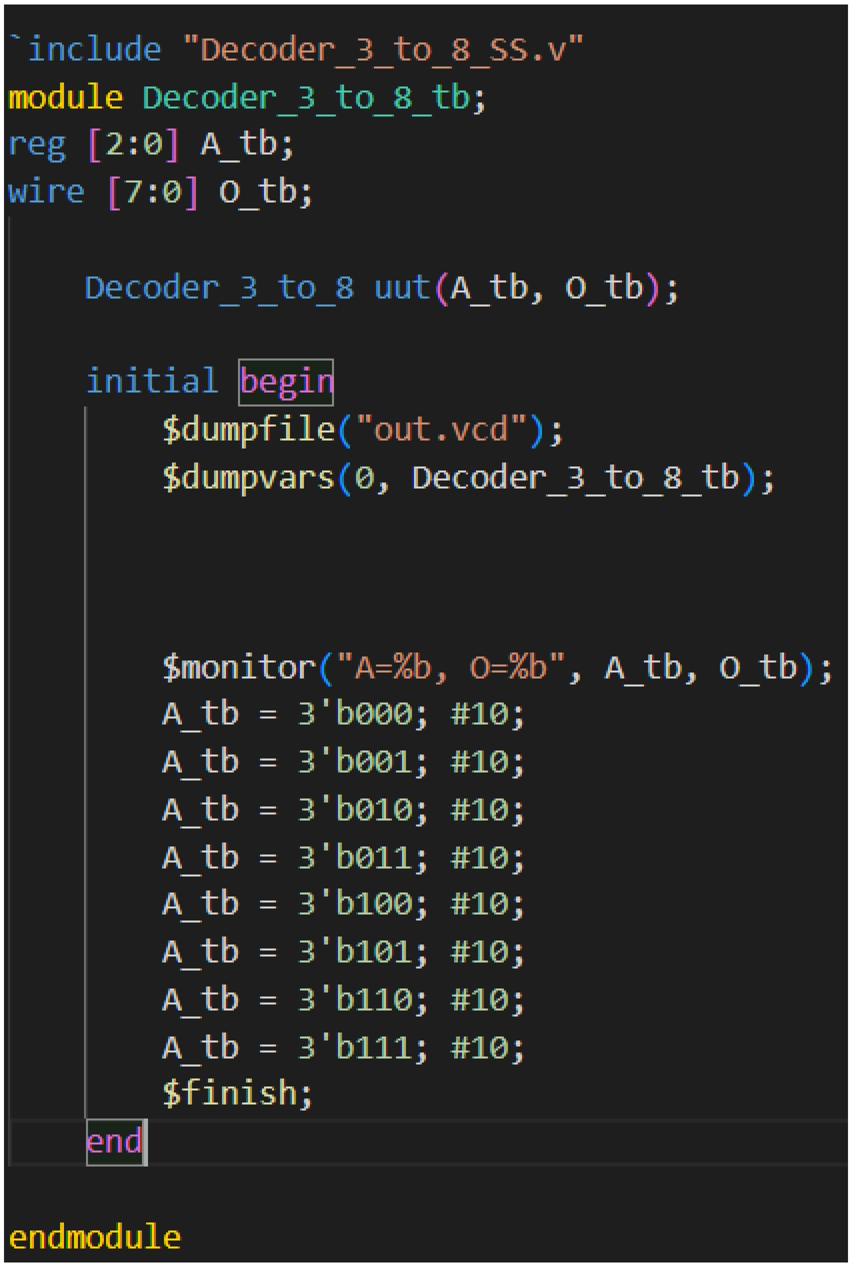


3. Write a verilog code for 3 to 8 Decoder circuit along with the test bench.
Code:



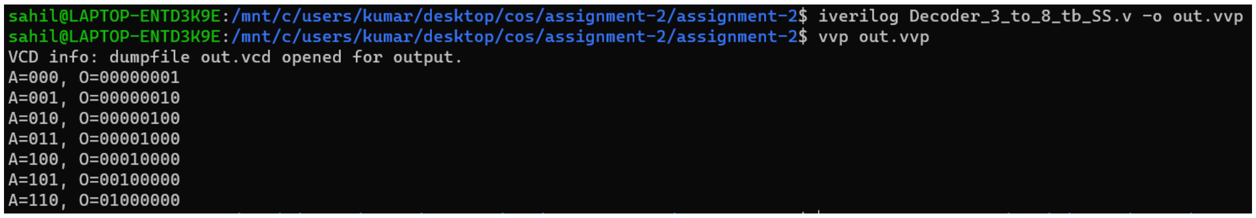
```
module Decoder_3_to_8(input [2:0] A,
                      output reg [7:0] O);

    always @(A)
    begin
        case(A)
            3'b000: O = 8'b00000001;
            3'b001: O = 8'b00000010;
            3'b010: O = 8'b00000100;
            3'b011: O = 8'b00001000;
            3'b100: O = 8'b00010000;
            3'b101: O = 8'b00100000;
            3'b110: O = 8'b01000000;
            3'b111: O = 8'b10000000;
            default: O = 8'b11111111; // For undefined inputs
        endcase
    end

endmodule
```

Testbench:



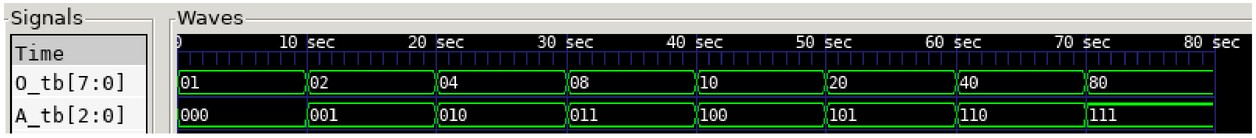
```verilog
`include "Decoder_3_to_8_SS.v"
module Decoder_3_to_8_tb;
reg [2:0] A_tb;
wire [7:0] O_tb;

    Decoder_3_to_8 uut(A_tb, O_tb);

    initial begin
        $dumpfile("out.vcd");
        $dumpvars(0, Decoder_3_to_8_tb);



        $monitor("A=%b, O=%b", A_tb, O_tb);
        A_tb = 3'b000; #10;
        A_tb = 3'b001; #10;
        A_tb = 3'b010; #10;
        A_tb = 3'b011; #10;
        A_tb = 3'b100; #10;
        A_tb = 3'b101; #10;
        A_tb = 3'b110; #10;
        A_tb = 3'b111; #10;
        $finish;
    end

endmodule
```

Terminal Output:

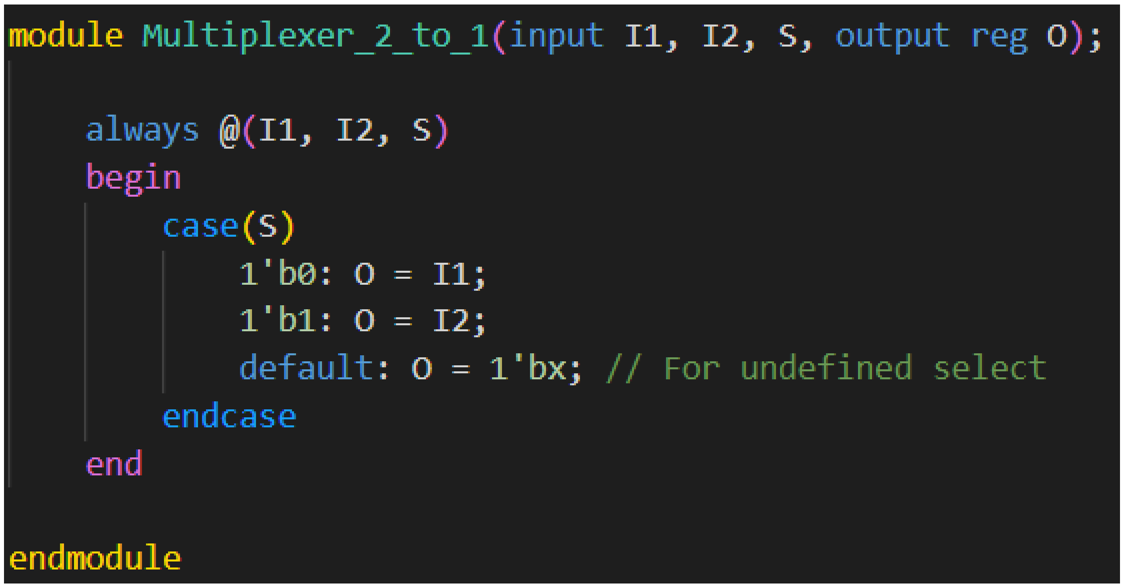


```
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ iverilog Decoder_3_to_8_tb_SS.v -o out.vvp
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ vvp out.vvp
VCD info: dumpfile out.vcd opened for output.
A=000, O=00000001
A=001, O=00000010
A=010, O=00000100
A=011, O=00001000
A=100, O=00010000
A=101, O=00100000
A=110, O=01000000
```

GTK wave Output:

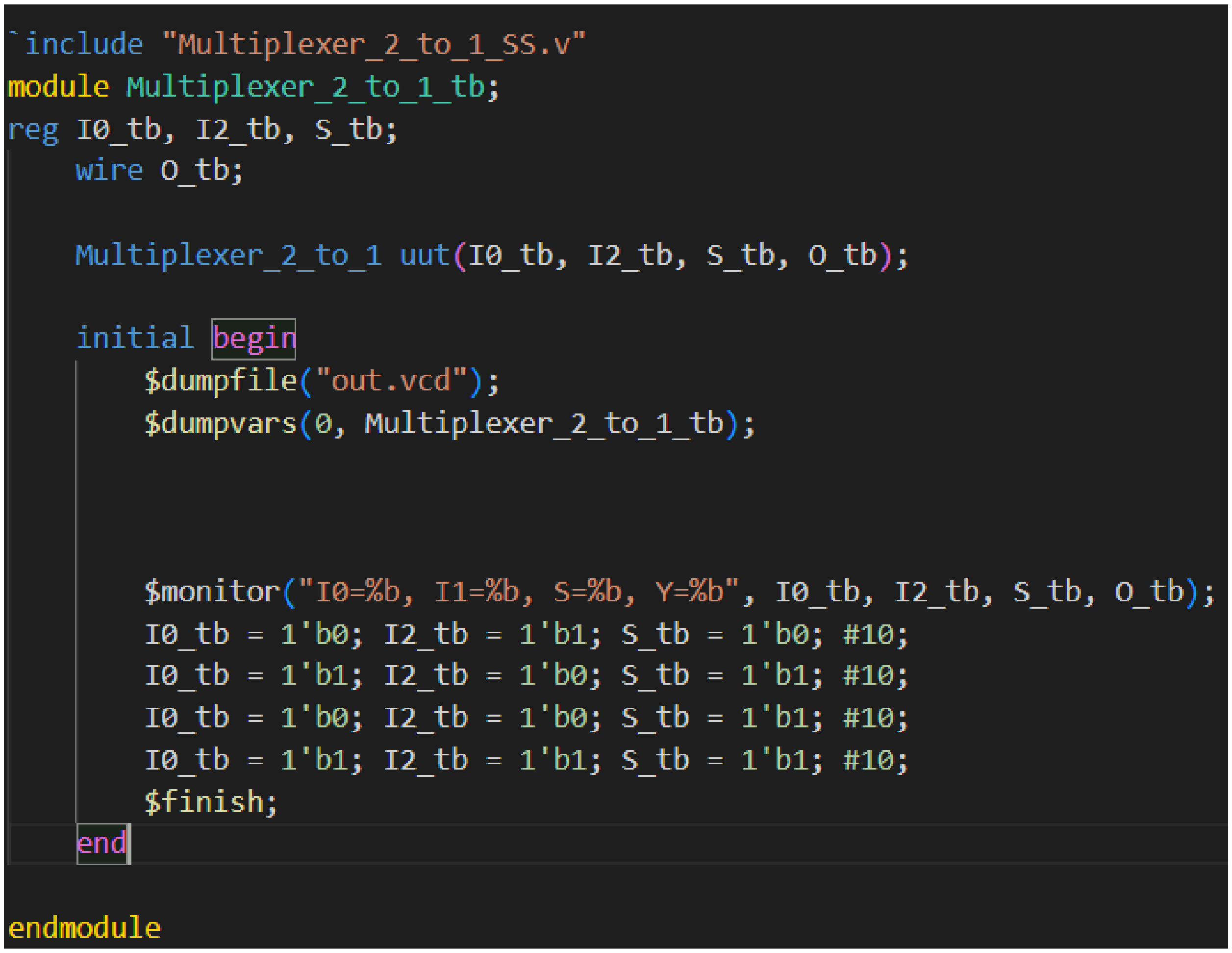


4. Write a verilog code for 2-to-1 Multiplexer circuit along with the test bench.
Code:



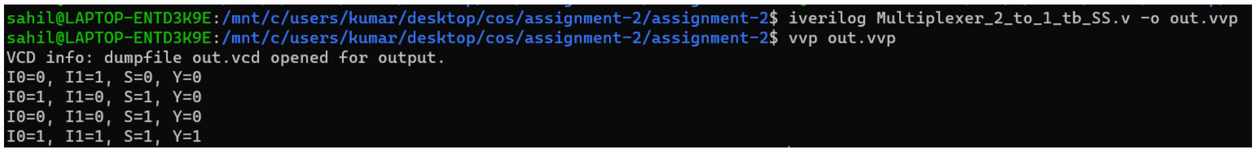
```
module Multiplexer_2_to_1(input I1, I2, S, output reg O);

    always @(I1, I2, S)
    begin
        case(S)
            1'b0: O = I1;
            1'b1: O = I2;
            default: O = 1'bx; // For undefined select
        endcase
    end

endmodule
```

Testbench:



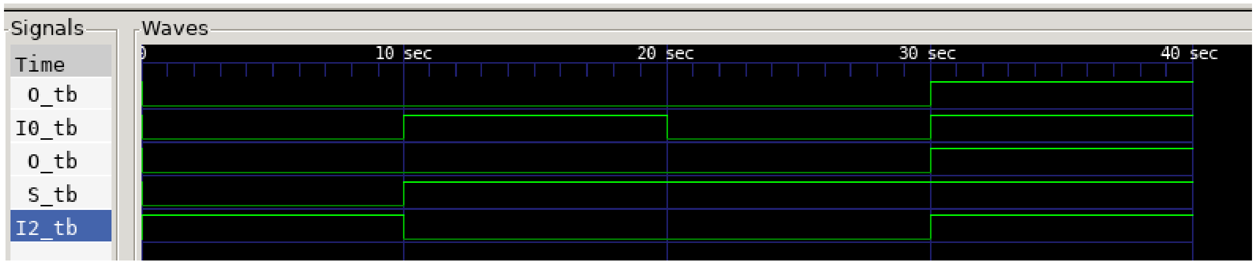
```
`include "Multiplexer_2_to_1_SS.v"
module Multiplexer_2_to_1_tb;
reg I0_tb, I2_tb, S_tb;
    wire O_tb;

    Multiplexer_2_to_1 uut(I0_tb, I2_tb, S_tb, O_tb);

    initial begin
        $dumpfile("out.vcd");
        $dumpvars(0, Multiplexer_2_to_1_tb);



        $monitor("I0=%b, I1=%b, S=%b, Y=%b", I0_tb, I2_tb, S_tb, O_tb);
        I0_tb = 1'b0; I2_tb = 1'b1; S_tb = 1'b0; #10;
        I0_tb = 1'b1; I2_tb = 1'b0; S_tb = 1'b1; #10;
        I0_tb = 1'b0; I2_tb = 1'b0; S_tb = 1'b1; #10;
        I0_tb = 1'b1; I2_tb = 1'b1; S_tb = 1'b1; #10;
        $finish;
    end

endmodule
```

Terminal Output:

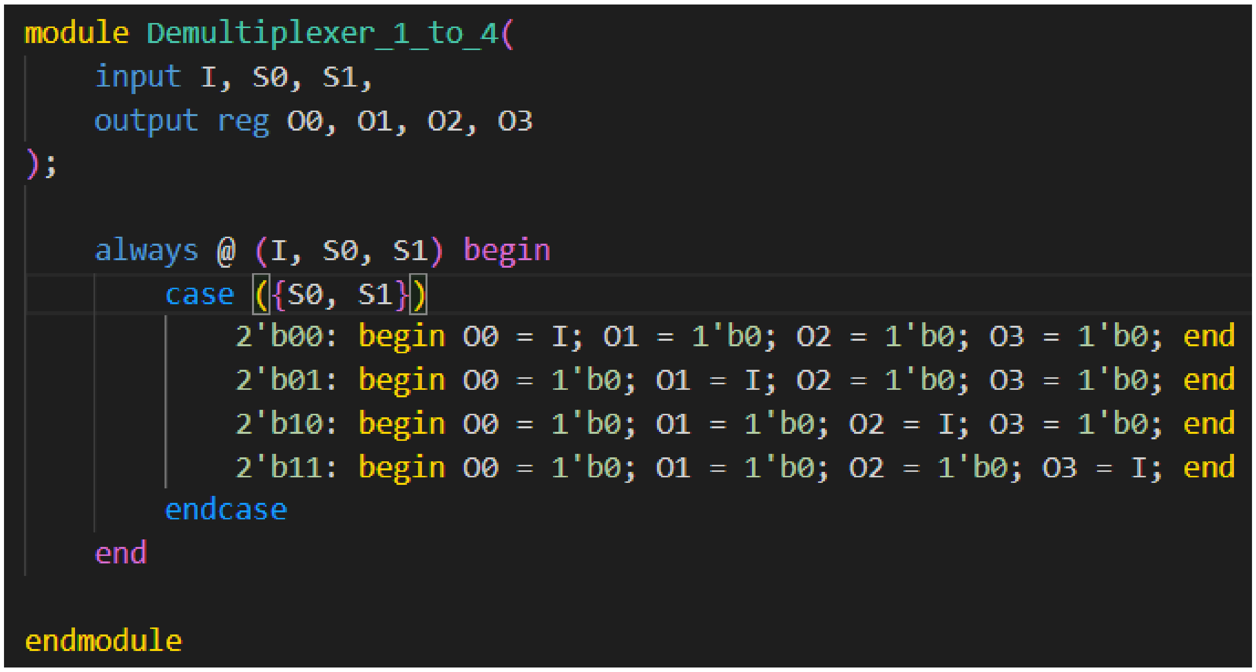


```
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ iverilog Multiplexer_2_to_1_tb_SS.v -o out.vvp
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ vvp out.vvp
VCD info: dumpfile out.vcd opened for output.
I0=0, I1=1, S=0, Y=0
I0=1, I1=0, S=1, Y=0
I0=0, I1=0, S=1, Y=0
I0=1, I1=1, S=1, Y=1
```

GTK wave Output:

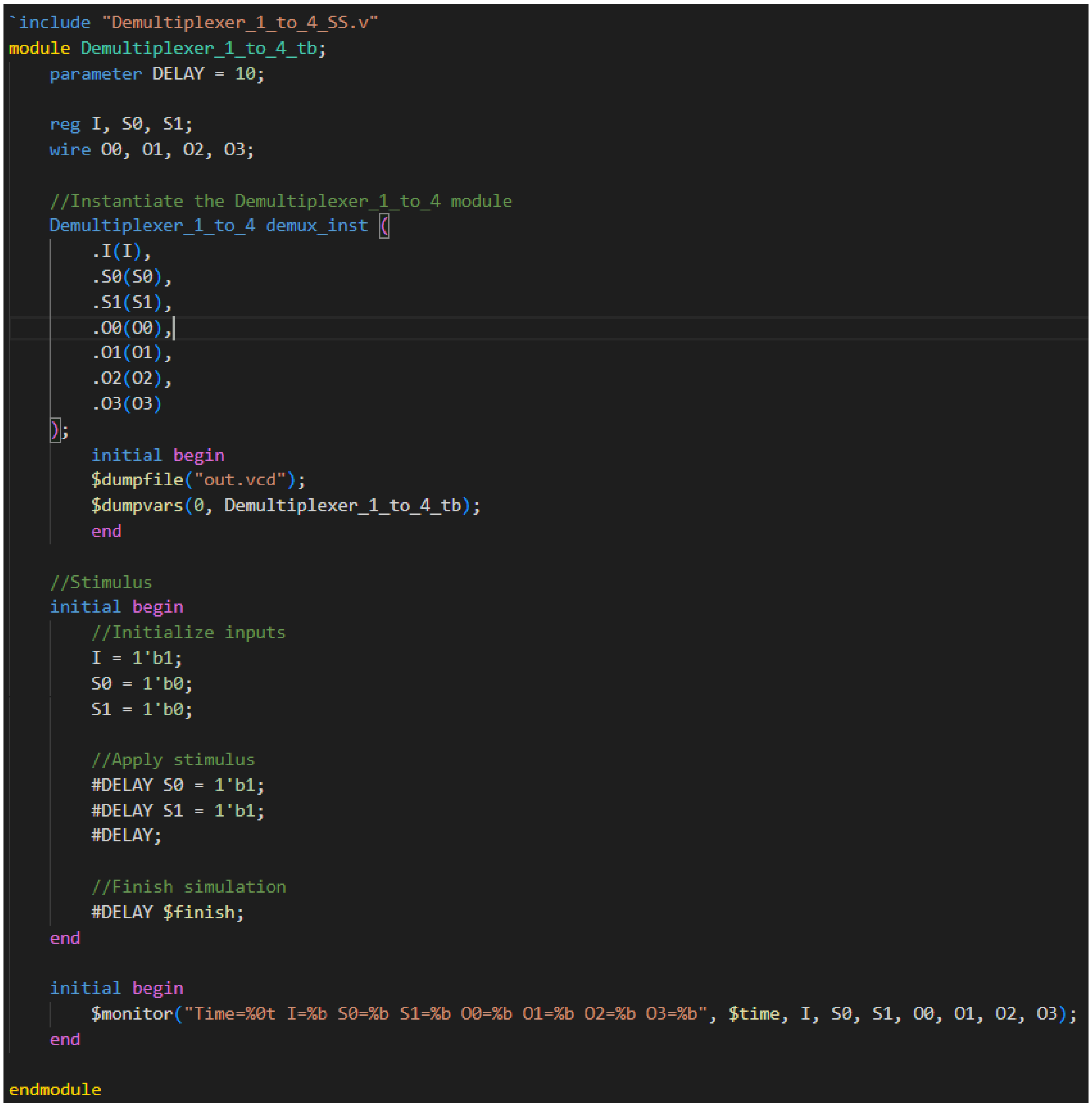


5. Write a verilog code for 1-to-4 Demultiplexer circuit along with the test bench.
Code:



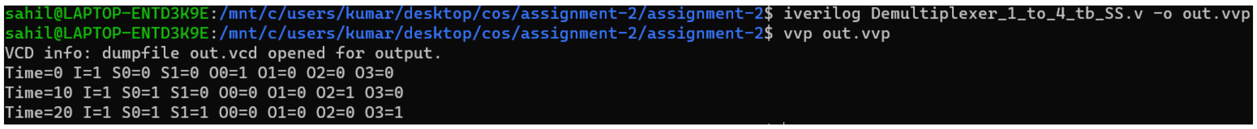
```
module Demultiplexer_1_to_4(
    input I, S0, S1,
    output reg O0, O1, O2, O3
);

    always @ (I, S0, S1) begin
        case ({S0, S1})
            2'b00: begin O0 = I; O1 = 1'b0; O2 = 1'b0; O3 = 1'b0; end
            2'b01: begin O0 = 1'b0; O1 = I; O2 = 1'b0; O3 = 1'b0; end
            2'b10: begin O0 = 1'b0; O1 = 1'b0; O2 = I; O3 = 1'b0; end
            2'b11: begin O0 = 1'b0; O1 = 1'b0; O2 = 1'b0; O3 = I; end
        endcase
    end

endmodule
```

Testbench:



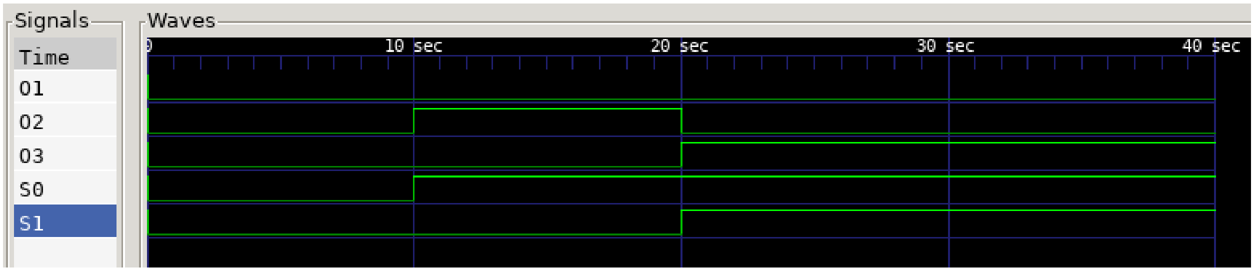
```
`include "Demultiplexer_1_to_4_SS.v"
module Demultiplexer_1_to_4_tb;
    parameter DELAY = 10;

    reg I, S0, S1;
    wire O0, O1, O2, O3;

    //Instantiate the Demultiplexer_1_to_4 module
    Demultiplexer_1_to_4 demux_inst (
        .I(I),
        .S0(S0),
        .S1(S1),
        .O0(O0),
        .O1(O1),
        .O2(O2),
        .O3(O3)
    );
        initial begin
        $dumpfile("out.vcd");
        $dumpvars(0, Demultiplexer_1_to_4_tb);
        end

    //Stimulus
    initial begin
        //Initialize inputs
        I = 1'b1;
        S0 = 1'b0;
        S1 = 1'b0;

        //Apply stimulus
        #DELAY S0 = 1'b1;
        #DELAY S1 = 1'b1;
        #DELAY;

        //Finish simulation
        #DELAY $finish;
    end

    initial begin
        $monitor("Time=%0t I=%b S0=%b S1=%b O0=%b O1=%b O2=%b O3=%b", $time, I, S0, S1, O0, O1, O2, O3);
    end

endmodule
```

Terminal Output:



```
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ iverilog Demultiplexer_1_to_4_tb_SS.v -o out.vvp
sahil@LAPTOP-ENTD3K9E:/mnt/c/users/kumar/desktop/cos/assignment-2/assignment-2$ vvp out.vvp
VCD info: dumpfile out.vcd opened for output.
Time=0 I=1 S0=0 S1=0 O0=1 O1=0 O2=0 O3=0
Time=10 I=1 S0=1 S1=0 O0=0 O1=0 O2=1 O3=0
Time=20 I=1 S0=1 S1=1 O0=0 O1=0 O2=0 O3=1
```

GTK wave output:


Signals
Waves
Time
O1
O2
O3
S0
S1
0
10 sec
20 sec
30 sec
40 sec