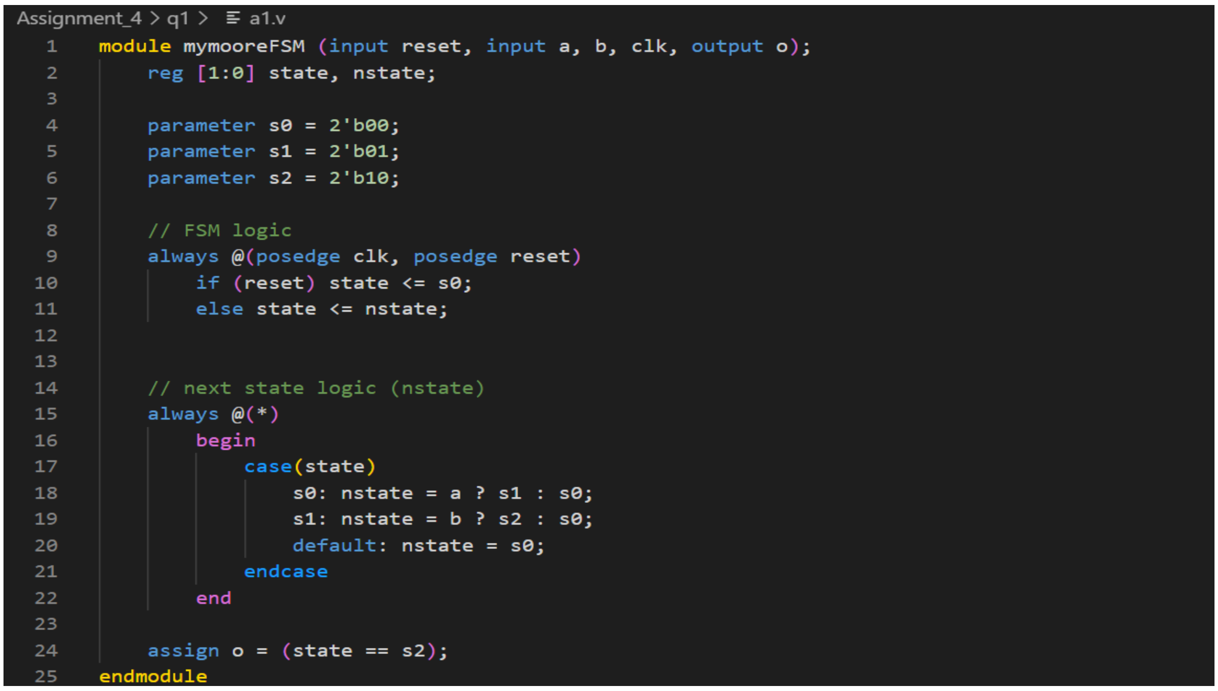
Computer Organization

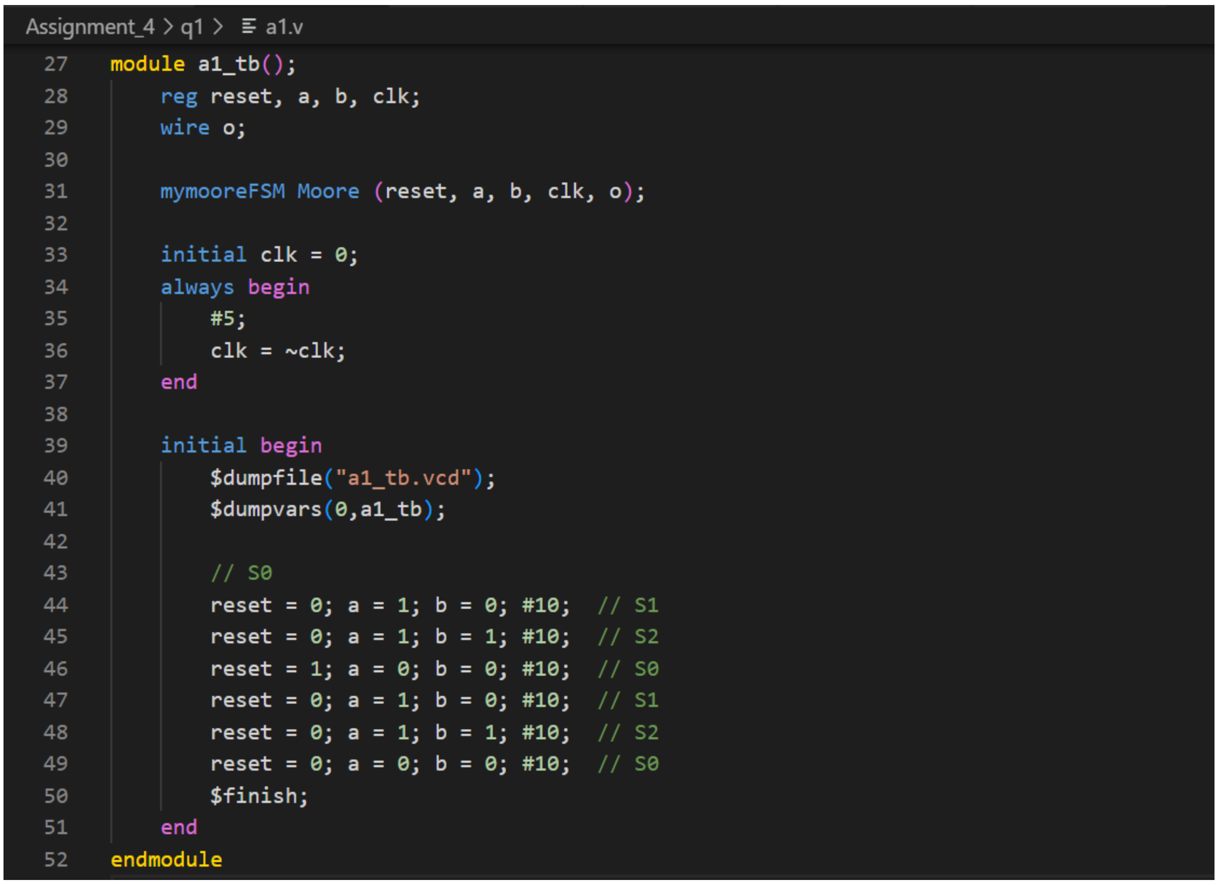
Assignment 4

Sambit Sahoo (22277)

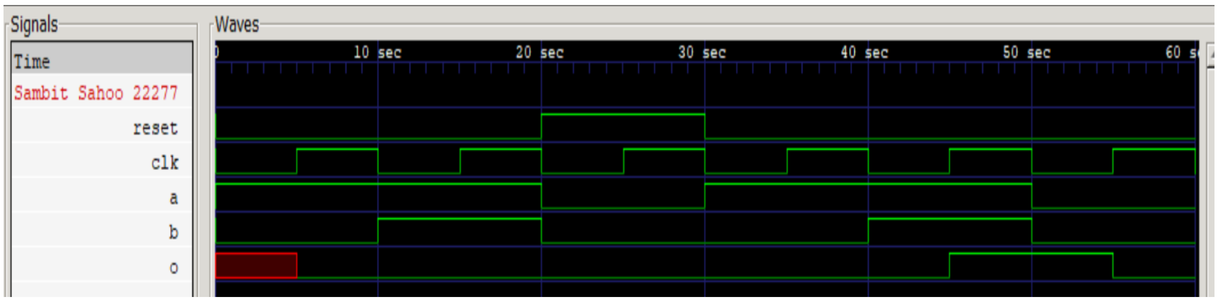
Q1 - Verilog based Moore machine as shown in question.



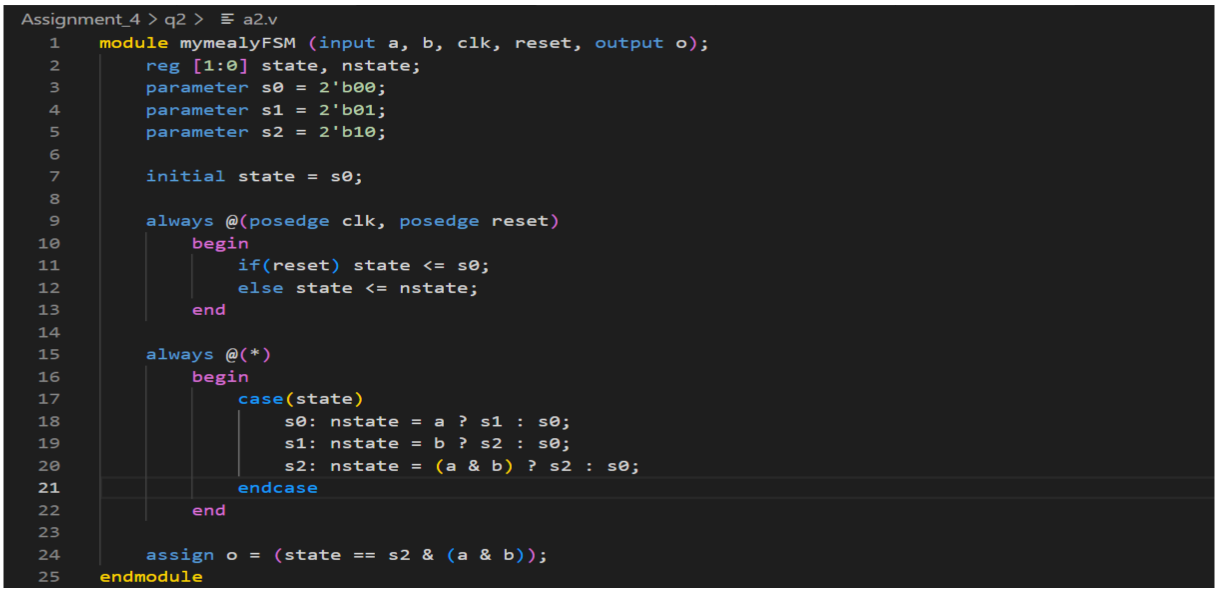
```verilog
module mymooreFSM (input reset, input a, b, clk, output o);
    reg [1:0] state, nstate;

    parameter s0 = 2'b00;
    parameter s1 = 2'b01;
    parameter s2 = 2'b10;

    // FSM logic
    always @(posedge clk, posedge reset)
        if (reset) state <= s0;
        else state <= nstate;


    // next state logic (nstate)
    always @(*)
        begin
            case(state)
                s0: nstate = a ? s1 : s0;
                s1: nstate = b ? s2 : s0;
                default: nstate = s0;
            endcase
        end

    assign o = (state == s2);
endmodule
```



```verilog
module a1_tb();
    reg reset, a, b, clk;
    wire o;

    mymooreFSM Moore (reset, a, b, clk, o);

    initial clk = 0;
    always begin
        #5;
        clk = ~clk;
    end

    initial begin
        $dumpfile("a1_tb.vcd");
        $dumpvars(0,a1_tb);

        // S0
        reset = 0; a = 1; b = 0; #10;  // S1
        reset = 0; a = 1; b = 1; #10;  // S2
        reset = 1; a = 0; b = 0; #10;  // S0
        reset = 0; a = 1; b = 0; #10;  // S1
        reset = 0; a = 1; b = 1; #10;  // S2
        reset = 0; a = 0; b = 0; #10;  // S0
        $finish;
    end
endmodule
```

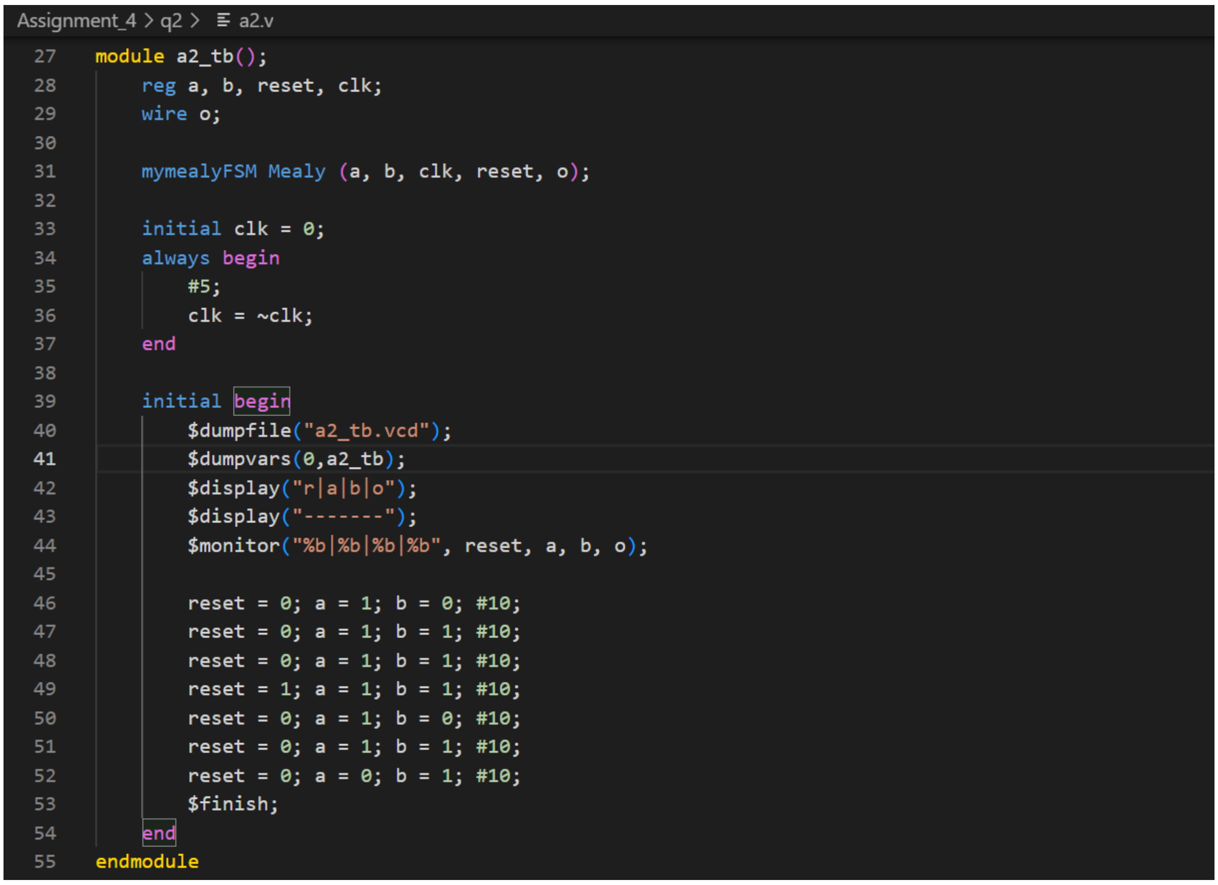


Q2 - Verilog based Mealy machine as given in question.



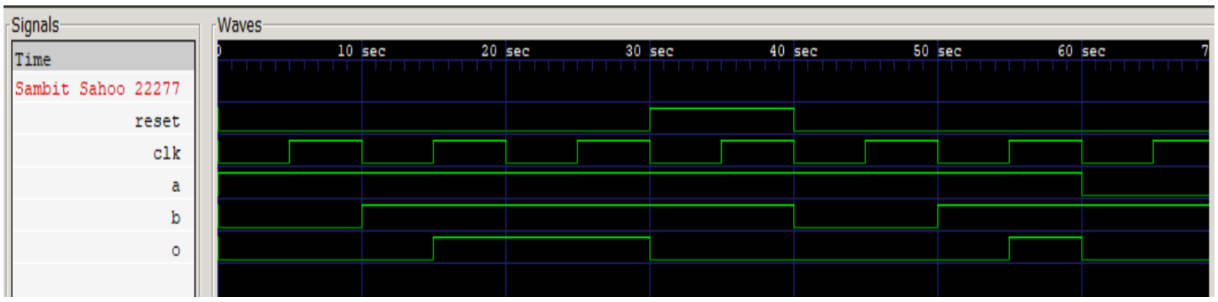
Assignment_4 > q2 > a2.v

```verilog
module mymealyFSM (input a, b, clk, reset, output o);
    reg [1:0] state, nstate;
    parameter s0 = 2'b00;
    parameter s1 = 2'b01;
    parameter s2 = 2'b10;

    initial state = s0;

    always @(posedge clk, posedge reset)
        begin
            if(reset) state <= s0;
            else state <= nstate;
        end

    always @(*)
        begin
            case(state)
                s0: nstate = a ? s1 : s0;
                s1: nstate = b ? s2 : s0;
                s2: nstate = (a & b) ? s2 : s0;
            endcase
        end

    assign o = (state == s2 & (a & b));
endmodule
```

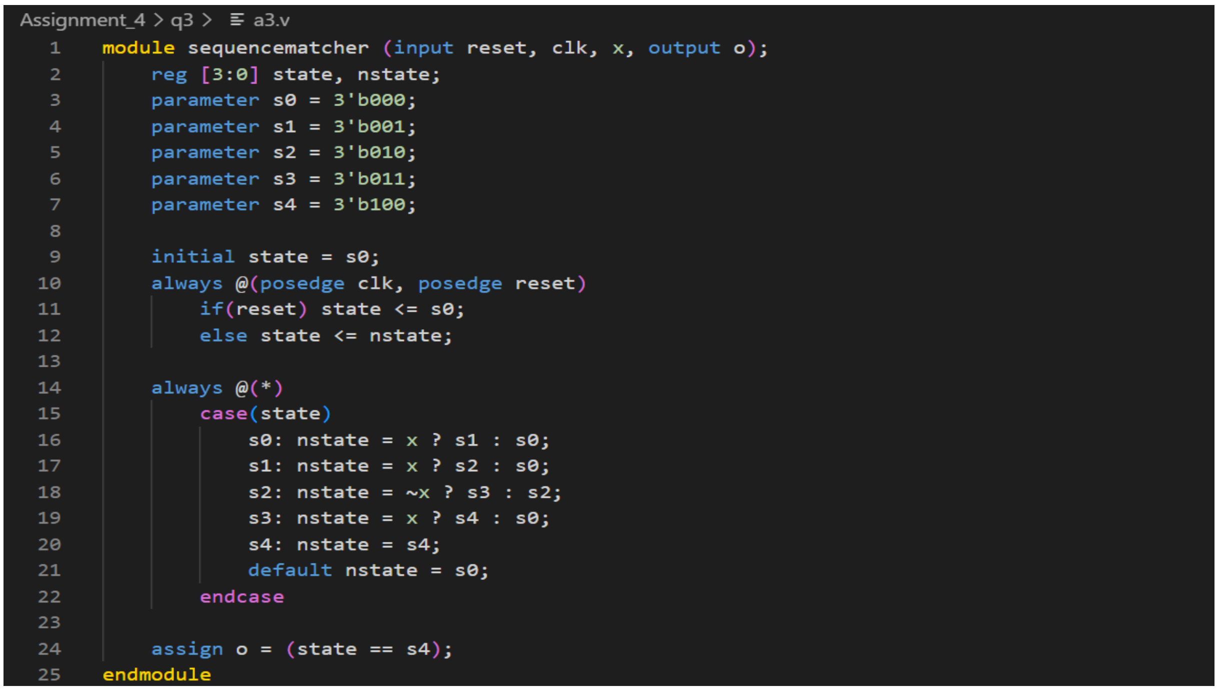


Assignment_4 > q2 > a2.v

```verilog
module a2_tb();
    reg a, b, reset, clk;
    wire o;

    mymealyFSM Mealy (a, b, clk, reset, o);

    initial clk = 0;
    always begin
        #5;
        clk = ~clk;
    end

    initial begin
        $dumpfile("a2_tb.vcd");
        $dumpvars(0,a2_tb);
        $display("r|a|b|o");
        $display("-------");
        $monitor("%b|%b|%b|%b", reset, a, b, o);

        reset = 0; a = 1; b = 0; #10;
        reset = 0; a = 1; b = 1; #10;
        reset = 0; a = 1; b = 1; #10;
        reset = 1; a = 1; b = 1; #10;
        reset = 0; a = 1; b = 0; #10;
        reset = 0; a = 1; b = 1; #10;
        reset = 0; a = 0; b = 1; #10;
        $finish;
    end
endmodule
```

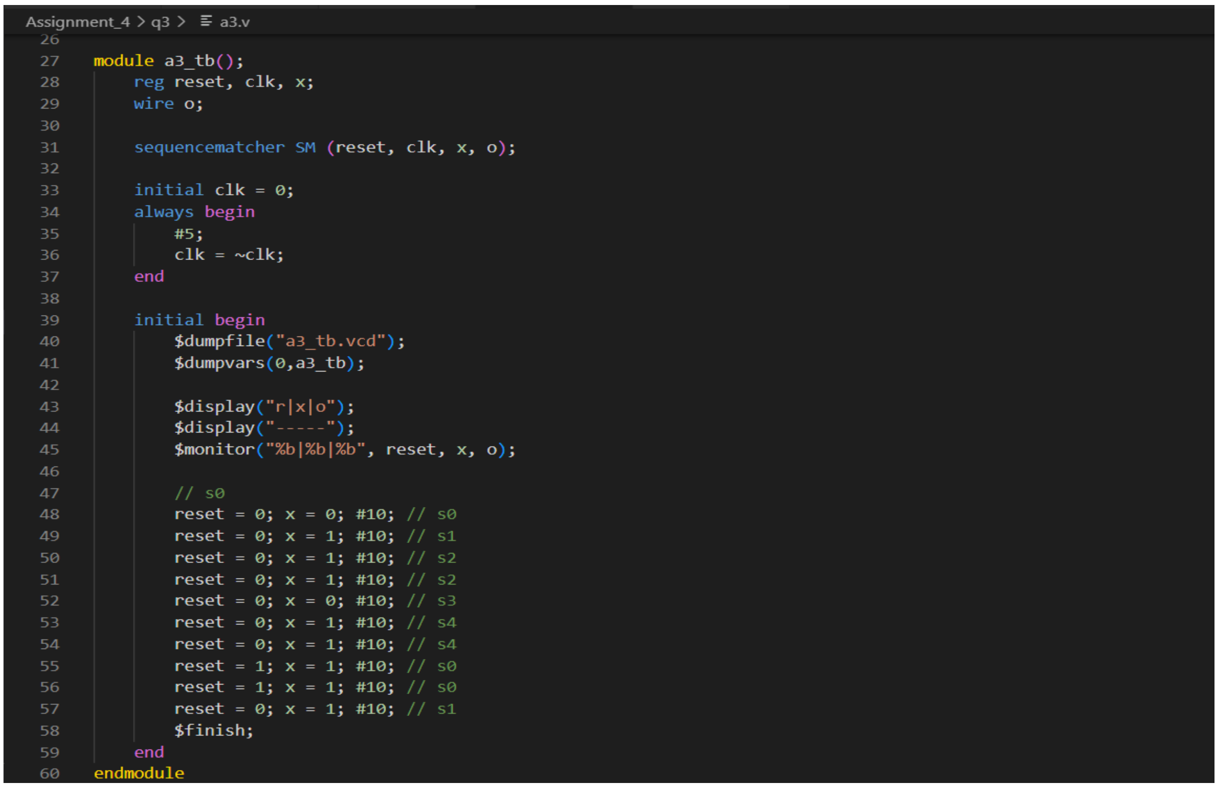


Q3- FSM in verilog that detects the input sequence 1101 on input variable x.



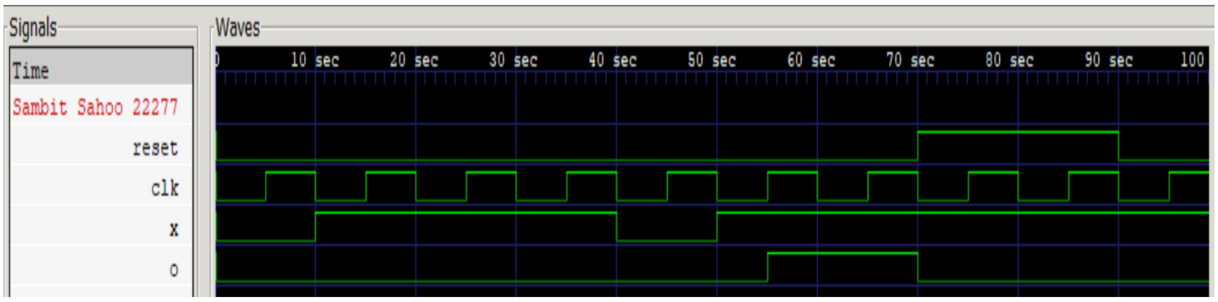
Assignment_4 > q3 > a3.v

```verilog
module sequencematcher (input reset, clk, x, output o);
    reg [3:0] state, nstate;
    parameter s0 = 3'b000;
    parameter s1 = 3'b001;
    parameter s2 = 3'b010;
    parameter s3 = 3'b011;
    parameter s4 = 3'b100;

    initial state = s0;
    always @(posedge clk, posedge reset)
        if(reset) state <= s0;
        else state <= nstate;

    always @(*)
        case(state)
            s0: nstate = x ? s1 : s0;
            s1: nstate = x ? s2 : s0;
            s2: nstate = ~x ? s3 : s2;
            s3: nstate = x ? s4 : s0;
            s4: nstate = s4;
            default nstate = s0;
        endcase

    assign o = (state == s4);
endmodule
```



Assignment_4 > q3 > a3.v

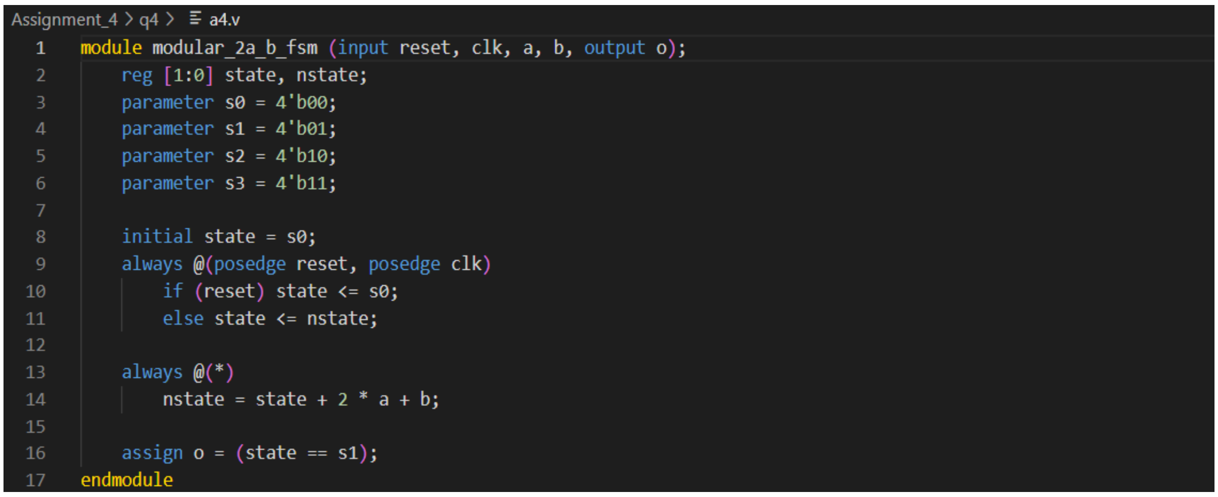
```verilog

module a3_tb();
    reg reset, clk, x;
    wire o;

    sequencematcher SM (reset, clk, x, o);

    initial clk = 0;
    always begin
        #5;
        clk = ~clk;
    end

    initial begin
        $dumpfile("a3_tb.vcd");
        $dumpvars(0,a3_tb);

        $display("r|x|o");
        $display("-----");
        $monitor("%b|%b|%b", reset, x, o);

        // s0
        reset = 0; x = 0; #10; // s0
        reset = 0; x = 1; #10; // s1
        reset = 0; x = 1; #10; // s2
        reset = 0; x = 1; #10; // s2
        reset = 0; x = 0; #10; // s3
        reset = 0; x = 1; #10; // s4
        reset = 0; x = 1; #10; // s4
        reset = 1; x = 1; #10; // s0
        reset = 1; x = 1; #10; // s0
        reset = 0; x = 1; #10; // s1
        $finish;
    end
endmodule
```



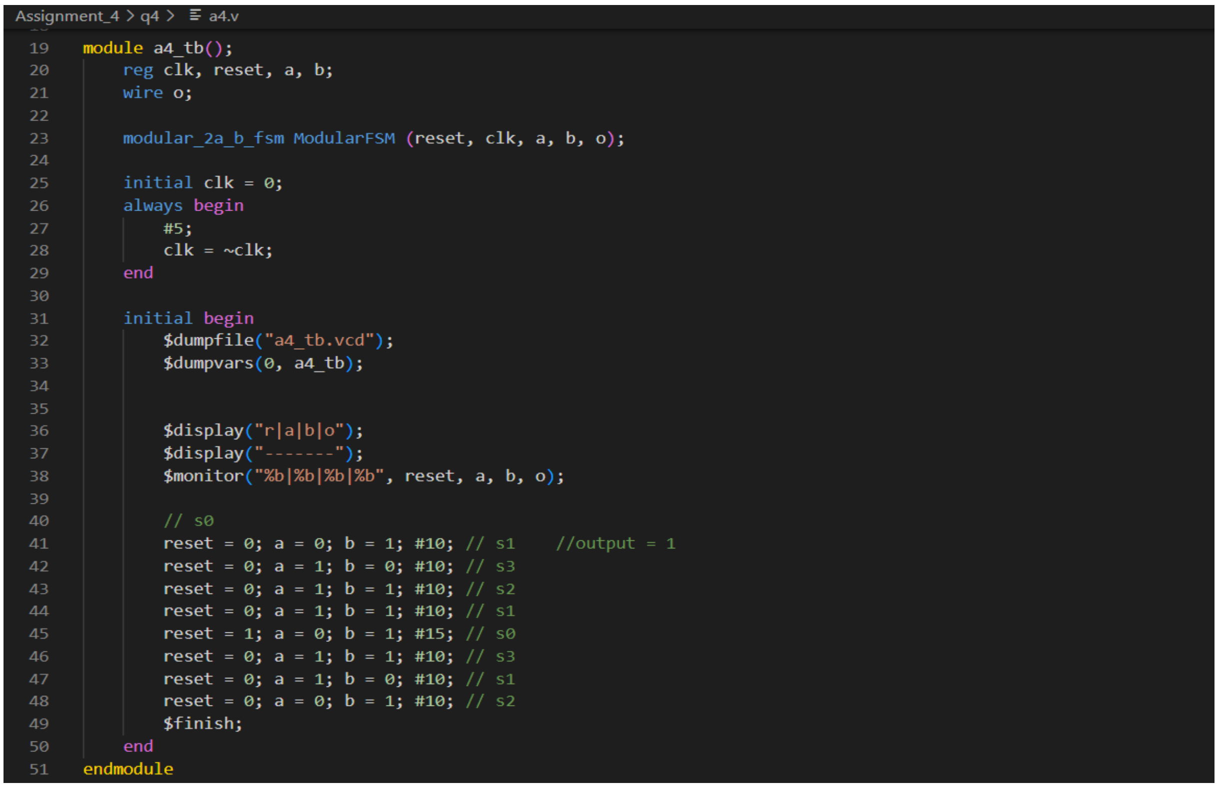
Q4. Verilog Code for Modular Equation Implementation

You are given two one-bit input signals P_A and P_B, and one-bit output signal O, for the following modular equation:

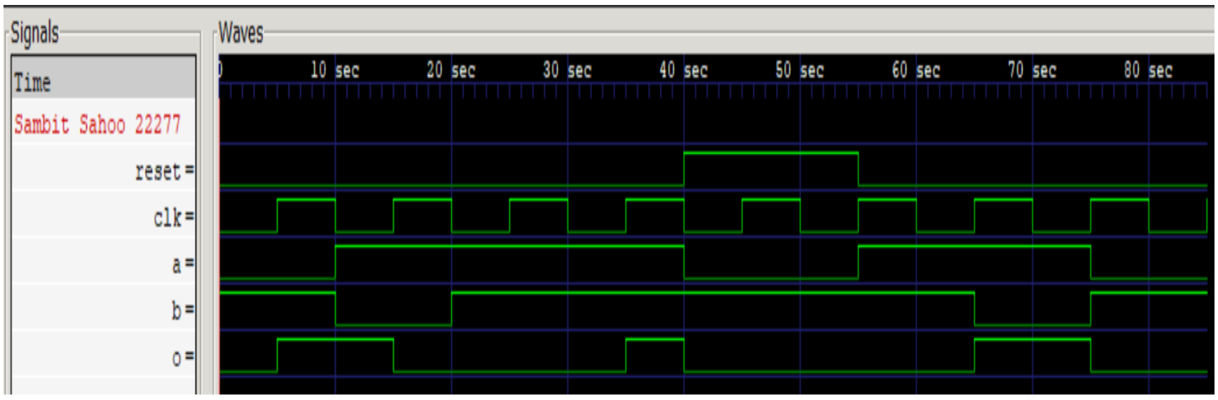
$$2N(P_A) + N(P_B) \equiv 1 \pmod 4$$



```verilog
module modular_2a_b_fsm (input reset, clk, a, b, output o);
    reg [1:0] state, nstate;
    parameter s0 = 4'b00;
    parameter s1 = 4'b01;
    parameter s2 = 4'b10;
    parameter s3 = 4'b11;

    initial state = s0;
    always @(posedge reset, posedge clk)
        if (reset) state <= s0;
        else state <= nstate;

    always @(*)
        nstate = state + 2 * a + b;

    assign o = (state == s1);
endmodule
```



```verilog
module a4_tb();
    reg clk, reset, a, b;
    wire o;

    modular_2a_b_fsm ModularFSM (reset, clk, a, b, o);

    initial clk = 0;
    always begin
        #5;
        clk = ~clk;
    end

    initial begin
        $dumpfile("a4_tb.vcd");
        $dumpvars(0, a4_tb);


        $display("r|a|b|o");
        $display("-------");
        $monitor("%b|%b|%b|%b", reset, a, b, o);

        // s0
        reset = 0; a = 0; b = 1; #10; // s1    //output = 1
        reset = 0; a = 1; b = 0; #10; // s3
        reset = 0; a = 1; b = 1; #10; // s2
        reset = 0; a = 1; b = 1; #10; // s1
        reset = 1; a = 0; b = 1; #15; // s0
        reset = 0; a = 1; b = 1; #10; // s3
        reset = 0; a = 1; b = 0; #10; // s1
        reset = 0; a = 0; b = 1; #10; // s2
        $finish;
    end
endmodule
```


Signals
Waves
Time
Sambit Sahoo 22277
reset =
clk =
a =
b =
o =
0
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
80 sec