ECS 409/609
Assignment 4
Finite State Machine Assignment Solutions

Kunwar Arpit Singh

Submission Deadline: September 18, 2025

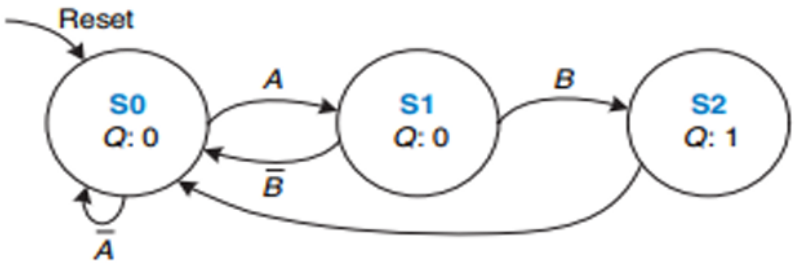
For code repository of this assignment: **Verilog Assignments**

---

1. Problem 1

Problem statement

Implement a Verilog based Moore machine as shown in the given figure.



Verilog source

```
module moore_machine (input CLK, input RESET, input A, input B, output reg Q);

    parameter S0 = 2'b00;
    parameter S1 = 2'b01;
    parameter S2 = 2'b10;

    reg [1:0] current_state;
    reg [1:0] next_state;

    always @(posedge CLK) begin
        if (RESET) begin
            current_state <= S0;
        end else begin
            current_state <= next_state;
        end
    end

```

```
    always @(*) begin
        case (current_state)
            S0: begin
                if (A)
                    next_state = S1;
                else
                    next_state = S0;
            end

            S1: begin
                if (B)
                    next_state = S2;
                else
                    next_state = S0;
            end

            S2: begin
                next_state = S0;
            end

            default: begin
                next_state = S0;
            end
        endcase
    end

    always @(*) begin
        case (current_state)
            S0: Q = 1'b0;
            S1: Q = 1'b0;
            S2: Q = 1'b1;
            default: Q = 1'b0;
        endcase
    end
endmodule
```

Testbench

```
module moore_machine_tb;

    reg clk;
    reg reset;
    reg A;
    reg B;
    wire Q;
    reg [8*24-1:0] name;

    moore_machine dut (.CLK(clk), .RESET(reset), .A(A), .B(B), .Q(Q));

    initial begin
```

```
13          clk = 0;
14          forever #5 clk = ~clk;
15      end
16
17      initial begin
18          name = "␣Kunwar␣Arpit␣Singh␣22185";
19          $display("Kunwar␣Arpit␣Singh");
20          $dumpfile("assign4_problem1_moore_machine.vcd");
21          $dumpvars(1, moore_machine_tb);
22          $display("Kunwar␣Arpit␣Singh");
23
24          $display("|␣CLK␣|␣RESET␣|␣A␣|␣B␣|␣State␣|␣Output␣Q␣␣|");
25          $monitor("|␣␣%b␣␣|␣␣␣%b␣␣␣|␣%b␣|␣%b␣|␣␣␣%b␣␣␣|␣␣␣␣␣%b␣␣␣␣␣|", clk,
                reset, A, B, dut.current_state, Q);
26
27          reset = 1; A = 0; B = 0;
28          @(posedge clk);
29          @(posedge clk);
30
31          reset = 0;
32          @(posedge clk);
33
34          A = 0; B = 1;
35          $display("For␣S0␣->␣S0␣(A=0)");
36          $display("|␣CLK␣|␣RESET␣|␣A␣|␣B␣|␣␣State␣|␣Output␣Q␣␣|");
37          @(posedge clk);
38
39          A = 1; B = 0;
40          $display("For␣S0␣->␣S1␣(A=1)");
41          $display("|␣CLK␣|␣RESET␣|␣A␣|␣B␣|␣␣State␣|␣Output␣Q␣␣|");
42          @(posedge clk);
43
44          A = 1; B = 0;
45          $display("For␣S1␣->␣S0␣(B=0)");
46          $display("|␣CLK␣|␣RESET␣|␣A␣|␣B␣|␣␣State␣|␣Output␣Q␣␣|");
47          @(posedge clk);
48
49          A = 1; B = 0;
50          $display("For␣transition␣to␣S1");
51          $display("|␣CLK␣|␣RESET␣|␣A␣|␣B␣|␣␣State␣|␣Output␣Q␣␣|");
52          @(posedge clk);
53
54          A = 0; B = 1;
55          $display("For␣S1␣->␣S2␣(B=1)");
56          $display("|␣CLK␣|␣RESET␣|␣A␣|␣B␣|␣␣State␣|␣Output␣Q␣␣|");
57          @(posedge clk);
58
59          A = 1; B = 1;
60          $display("For␣S2␣->␣S0␣(unconditional)");
61          $display("|␣CLK␣|␣RESET␣|␣A␣|␣B␣|␣␣State␣|␣Output␣Q␣␣|");
62          @(posedge clk);
```

```
63
64          @(posedge clk);
65          @(posedge clk);
66          $finish;
67      end
68  endmodule
```

Output in Terminal

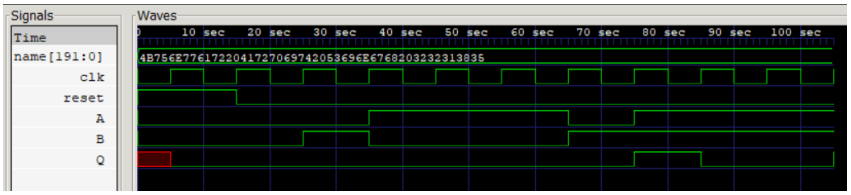
```
1  Kunwar Arpit Singh
2  VCD info: dumpfile assign4_problem1_moore_machine.vcd opened for output.
3  Kunwar Arpit Singh
4  | CLK | RESET | A | B | State | Output Q |
5  | 0 | 1 | 0 | 0 | xx | x |
6  | 1 | 1 | 0 | 0 | 00 | 0 |
7  | 0 | 1 | 0 | 0 | 00 | 0 |
8  | 1 | 0 | 0 | 0 | 00 | 0 |
9  | 0 | 0 | 0 | 0 | 00 | 0 |
10 For S0 -> S0 (A=0)
11 | CLK | RESET | A | B | State | Output Q |
12 | 1 | 0 | 0 | 1 | 00 | 0 |
13 | 0 | 0 | 0 | 1 | 00 | 0 |
14 For S0 -> S1 (A=1)
15 | CLK | RESET | A | B | State | Output Q |
16 | 1 | 0 | 1 | 0 | 00 | 0 |
17 | 0 | 0 | 1 | 0 | 00 | 0 |
18 For S1 -> S0 (B=0)
19 | CLK | RESET | A | B | State | Output Q |
20 | 1 | 0 | 1 | 0 | 01 | 0 |
21 | 0 | 0 | 1 | 0 | 01 | 0 |
22 For transition to S1
23 | CLK | RESET | A | B | State | Output Q |
24 | 1 | 0 | 1 | 0 | 00 | 0 |
25 | 0 | 0 | 1 | 0 | 00 | 0 |
26 For S1 -> S2 (B=1)
27 | CLK | RESET | A | B | State | Output Q |
28 | 1 | 0 | 0 | 1 | 01 | 0 |
29 | 0 | 0 | 0 | 1 | 01 | 0 |
30 For S2 -> S0 (unconditional)
31 | CLK | RESET | A | B | State | Output Q |
32 | 1 | 0 | 1 | 1 | 10 | 1 |
33 | 0 | 0 | 1 | 1 | 10 | 1 |
34 | 1 | 0 | 1 | 1 | 00 | 0 |
35 | 0 | 0 | 1 | 1 | 00 | 0 |
36 | 1 | 0 | 1 | 1 | 01 | 0 |
37 | 0 | 0 | 1 | 1 | 01 | 0 |
38 | 1 | 0 | 1 | 1 | 10 | 1 |
```

Output Screenshots

Terminal output (Moore Machine)

```
Kunwar Arpit Singh
VCD info: dumpfile assign4_problem1_moore_machine.vcd opened for output.
Kunwar Arpit Singh
| CLK | RESET | A | B | State | Output Q  |
|  0  |   1   | 0 | 0 |   xx  |     x     |
|  1  |   1   | 0 | 0 |   00  |     0     |
|  0  |   1   | 0 | 0 |   00  |     0     |
|  1  |   0   | 0 | 0 |   00  |     0     |
|  0  |   0   | 0 | 0 |   00  |     0     |
For S0 -> S0 (A=0)
| CLK | RESET | A | B |  State | Output Q  |
|  1  |   0   | 0 | 1 |   00   |     0     |
|  0  |   0   | 0 | 1 |   00   |     0     |
For S0 -> S1 (A=1)
| CLK | RESET | A | B |  State | Output Q  |
|  1  |   0   | 1 | 0 |   00   |     0     |
|  0  |   0   | 1 | 0 |   00   |     0     |
For S1 -> S0 (B=0)
| CLK | RESET | A | B |  State | Output Q  |
|  1  |   0   | 1 | 0 |   01   |     0     |
|  0  |   0   | 1 | 0 |   01   |     0     |
For transition to S1
| CLK | RESET | A | B |  State | Output Q  |
|  1  |   0   | 1 | 0 |   00   |     0     |
|  0  |   0   | 1 | 0 |   00   |     0     |
For S1 -> S2 (B=1)
| CLK | RESET | A | B |  State | Output Q  |
|  1  |   0   | 0 | 1 |   01   |     0     |
|  0  |   0   | 0 | 1 |   01   |     0     |
For S2 -> S0 (unconditional)
| CLK | RESET | A | B |  State | Output Q  |
|  1  |   0   | 1 | 1 |   10   |     1     |
|  0  |   0   | 1 | 1 |   10   |     1     |
|  1  |   0   | 1 | 1 |   00   |     0     |
|  0  |   0   | 1 | 1 |   00   |     0     |
|  1  |   0   | 1 | 1 |   01   |     0     |
|  0  |   0   | 1 | 1 |   01   |     0     |
|  1  |   0   | 1 | 1 |   10   |     1     |
```

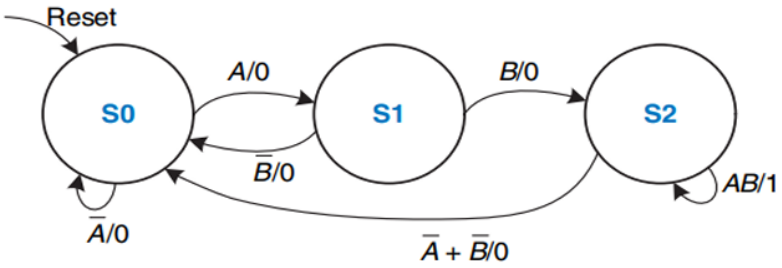
GTKWave waveform (Moore Machine)



2. Problem 2

Problem statement

Implement the Verilog based Mealy machine as given in the given figure.



Verilog source

```
module mealy_machine(input CLK, input RESET, input A, input B, output reg Y);

    parameter S0 = 2'b00;
    parameter S1 = 2'b01;
    parameter S2 = 2'b10;

    reg [1:0] current_state;
    reg [1:0] next_state;

    always @(posedge CLK) begin
        if (RESET)
            current_state <= S0;
        else
            current_state <= next_state;
    end

    always @(current_state or A or B) begin
        case (current_state)
            S0: begin
                if (A)
                    next_state = S1;
                else
                    next_state = S0;
            end
            S1: begin
                if (B)
                    next_state = S2;
                else
                    next_state = S0;
            end
            S2: begin
                if (A && B)
                    next_state = S2;
                else
```

```
                    next_state = S0;
            end
            default: begin
                next_state = S0;
            end
        endcase
    end

    always @(current_state or A or B) begin
        case (current_state)
            S0: Y = 0;
            S1: Y = 0;
            S2: begin
                if (A && B)
                    Y = 1;
                else
                    Y = 0;
            end
            default: Y = 0;
        endcase
    end

endmodule
```

Testbench

```
module mealy_machine_tb;

    reg clk;
    reg reset;
    reg A;
    reg B;
    wire Y;
    reg [8*24-1:0] name;
    mealy_machine dut (.CLK(clk), .RESET(reset), .A(A), .B(B), .Y(Y));

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

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign4_problem2_mealy_machine.vcd");
        $dumpvars(1, mealy_machine_tb);
        $display("Kunwar␣Arpit␣Singh");
        $display("|␣Reset␣|␣A␣|␣B␣|␣␣State␣|␣␣Output␣Y␣|");
        $monitor("|␣␣␣%b␣␣␣|␣%b␣|␣%b␣|␣␣␣%b␣␣␣|␣␣␣␣␣%b␣␣␣␣␣|", reset, A, B, dut
            .current_state, Y);
```

```
24
25          reset = 1; A = 0; B = 0;
26          #15;
27          reset = 0;
28          #10;
29          A = 0; B = 0; #10;
30          A = 1; B = 0; #10;
31
32          A = 1; B = 0; #10;
33          A = 1; B = 0; #10;
34          A = 0; B = 1; #10;
35
36          A = 1; B = 1; #10;
37          A = 1; B = 1; #10;
38
39          A = 0; B = 1; #10;
40
41          A = 1; B = 0; #10;
42          A = 0; B = 1; #10;
43
44          A = 1; B = 0; #10;
45
46          #20;
47          $finish;
48      end
49
50  endmodule
```

Output in Terminal

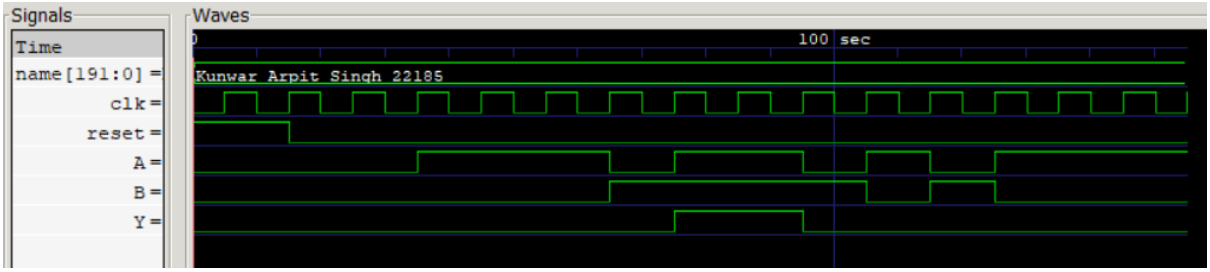
```
1  Kunwar Arpit Singh
2  VCD info: dumpfile assign4_problem2_mealy_machine.vcd opened for output.
3  Kunwar Arpit Singh
4  | Reset | A | B | State | Output Y |
5  | 1 | 0 | 0 | xx | 0 |
6  | 1 | 0 | 0 | 00 | 0 |
7  | 0 | 0 | 0 | 00 | 0 |
8  | 0 | 1 | 0 | 01 | 0 |
9  | 0 | 1 | 0 | 00 | 0 |
10 | 0 | 1 | 0 | 01 | 0 |
11 | 0 | 0 | 1 | 10 | 0 |
12 | 0 | 1 | 1 | 10 | 1 |
13 | 0 | 0 | 1 | 00 | 0 |
14 | 0 | 1 | 0 | 01 | 0 |
15 | 0 | 0 | 1 | 10 | 0 |
16 | 0 | 1 | 0 | 00 | 0 |
17 | 0 | 1 | 0 | 01 | 0 |
18 | 0 | 1 | 0 | 00 | 0 |
19 | 0 | 1 | 0 | 01 | 0 |
```

Output Screenshots

Terminal output (Mealy Machine)

```
Kunwar Arpit Singh
VCD info: dumpfile assign4_problem2_mealy_machine.vcd opened for output.
Kunwar Arpit Singh
| Reset | A | B |  State |  Output Y |
|   1   | 0 | 0 |   xx   |     0     |
|   1   | 0 | 0 |   00   |     0     |
|   0   | 0 | 0 |   00   |     0     |
|   0   | 1 | 0 |   01   |     0     |
|   0   | 1 | 0 |   00   |     0     |
|   0   | 1 | 0 |   01   |     0     |
|   0   | 0 | 1 |   10   |     0     |
|   0   | 1 | 1 |   10   |     1     |
|   0   | 0 | 1 |   00   |     0     |
|   0   | 1 | 0 |   01   |     0     |
|   0   | 0 | 1 |   10   |     0     |
|   0   | 1 | 0 |   00   |     0     |
|   0   | 1 | 0 |   01   |     0     |
|   0   | 1 | 0 |   00   |     0     |
|   0   | 1 | 0 |   01   |     0     |
```

GTKWave waveform (Mealy Machine)



3. Problem 3

Problem statement

Implement an FSM in verilog that detects the input sequence 1101 on input variable x. The Verilog code should detect the desired input sequence every time it occurs, even if embedded in a sequence of bits. When the Verilog code detects the desired input sequence and output, Z should be 1; otherwise, Z should be zero. On resetting the state machine, your verilog code should return to the initial state S0.

Verilog source

```
module fsm_implementation(input CLK, input RESET, input x, output reg Z);

    parameter S0 = 2'b00;
    parameter S1 = 2'b01;
    parameter S2 = 2'b10;
    parameter S3 = 2'b11;

    reg [1:0] current_state;
    reg [1:0] next_state;

    always @(posedge CLK or posedge RESET) begin
        if(RESET)
            current_state <= S0;
        else
            current_state <= next_state;
    end

    always @(current_state or x) begin
        case (current_state)
            S0:
                if(x)
                    next_state = S1;
                else
                    next_state = S0;
            S1:
                if(x)
                    next_state = S2;
                else
                    next_state = S0;
            S2:
                if(x)
                    next_state = S2;
                else
                    next_state = S3;
            S3:
                if(x)
                    next_state = S1;
                else
                    next_state = S0;
```

```
            default:
                next_state = S0;
        endcase
    end

    always @(current_state or x) begin
        if (current_state == S3 && x == 1'b1)
            Z = 1'b1;
        else
            Z = 1'b0;
    end

endmodule
```

Testbench

```
module fsm_implementation_tb;
    reg clk;
    reg reset;
    reg x;
    wire Z;
    reg [8*24-1:0] name;

    fsm_implementation dut (.CLK(clk), .RESET(reset), .x(x), .Z(Z));

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

    initial begin
        name = "␣Kunwar␣Arpit␣Singh␣22185";
        $display("Kunwar␣Arpit␣Singh");
        $dumpfile("assign4_problem3_FSM_implementation.vcd");
        $dumpvars(1, fsm_implementation_tb);
        $display("Kunwar␣Arpit␣Singh");

        reset = 1;
        x = 0;
        @(negedge clk);
        @(negedge clk);
        reset = 0;

        $display("\nFor␣Sequence:␣1101");
        $display("|␣Reset␣|␣Input␣x␣|␣State␣|␣Output␣Z␣|");
        @(negedge clk) x = 1;
        @(negedge clk) x = 1;
        @(negedge clk) x = 0;
        @(negedge clk) x = 1;
        @(negedge clk) x = 0;
```

```
35
36          $display("\nFor␣Sequence:␣011010");
37          $display("|␣Reset␣|␣Input␣x␣|␣State␣|␣Output␣Z␣|");
38          @(negedge clk) x = 0;
39          @(negedge clk) x = 1;
40          @(negedge clk) x = 1;
41          @(negedge clk) x = 0;
42          @(negedge clk) x = 1;
43          @(negedge clk) x = 0;
44
45          $display("\nFor␣Sequence:␣1101101");
46          $display("|␣Reset␣|␣Input␣x␣|␣State␣|␣Output␣Z␣|");
47          @(negedge clk) x = 1;
48          @(negedge clk) x = 1;
49          @(negedge clk) x = 0;
50          @(negedge clk) x = 1;
51          @(negedge clk) x = 1;
52          @(negedge clk) x = 0;
53          @(negedge clk) x = 1;
54          @(negedge clk) x = 0;
55
56          $display("\nFor␣Sequence:␣11101");
57          $display("|␣Reset␣|␣Input␣x␣|␣State␣|␣Output␣Z␣|");
58          @(negedge clk) x = 1;
59          @(negedge clk) x = 1;
60          @(negedge clk) x = 1;
61          @(negedge clk) x = 0;
62          @(negedge clk) x = 1;
63          @(negedge clk) x = 0;
64
65          $display("\nFor␣Sequence:␣0010100");
66          $display("|␣Reset␣|␣Input␣x␣|␣State␣|␣Output␣Z␣|");
67          @(negedge clk) x = 0;
68          @(negedge clk) x = 0;
69          @(negedge clk) x = 1;
70          @(negedge clk) x = 0;
71          @(negedge clk) x = 1;
72          @(negedge clk) x = 0;
73          @(negedge clk) x = 0;
74
75          #20 $finish;
76      end
77
78      // Monitor with timestamps for clearer debugging
79      initial begin
80          $monitor("|␣␣␣%b␣␣␣|␣␣␣␣%b␣␣␣␣|␣␣␣%b␣␣|␣␣␣␣␣%b␣␣␣␣|", reset, x, dut.
                current_state, Z);
81      end
82
83  endmodule
```

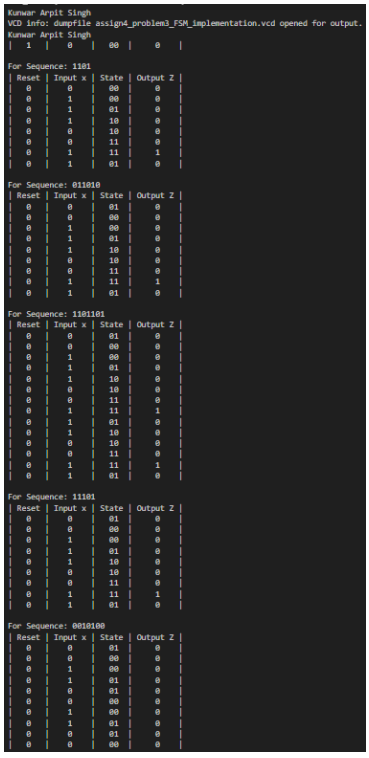
Output in Terminal

```
Kunwar Arpit Singh
VCD info: dumpfile assign4_problem3_FSM_implementation.vcd opened for output.
Kunwar Arpit Singh
| 1 | 0 | 00 | 0 |
For Sequence: 1101
| Reset | Input x | State | Output Z |
| 0 | 0 | 00 | 0 |
| 0 | 1 | 00 | 0 |
| 0 | 1 | 01 | 0 |
| 0 | 1 | 10 | 0 |
| 0 | 0 | 10 | 0 |
| 0 | 0 | 11 | 0 |
| 0 | 1 | 11 | 1 |
| 0 | 1 | 01 | 0 |
For Sequence: 011010
| Reset | Input x | State | Output Z |
| 0 | 0 | 01 | 0 |
| 0 | 0 | 00 | 0 |
| 0 | 1 | 00 | 0 |
| 0 | 1 | 01 | 0 |
| 0 | 1 | 10 | 0 |
| 0 | 0 | 10 | 0 |
| 0 | 0 | 11 | 0 |
| 0 | 1 | 11 | 1 |
| 0 | 1 | 01 | 0 |
For Sequence: 1101101
| Reset | Input x | State | Output Z |
| 0 | 0 | 01 | 0 |
| 0 | 0 | 00 | 0 |
| 0 | 1 | 00 | 0 |
| 0 | 1 | 01 | 0 |
| 0 | 1 | 10 | 0 |
| 0 | 0 | 10 | 0 |
| 0 | 0 | 11 | 0 |
| 0 | 1 | 11 | 1 |
| 0 | 1 | 01 | 0 |
| 0 | 1 | 10 | 0 |
| 0 | 0 | 10 | 0 |
| 0 | 0 | 11 | 0 |
| 0 | 1 | 11 | 1 |
| 0 | 1 | 01 | 0 |
For Sequence: 11101
| Reset | Input x | State | Output Z |
| 0 | 0 | 01 | 0 |
| 0 | 0 | 00 | 0 |
| 0 | 1 | 00 | 0 |
| 0 | 1 | 01 | 0 |
| 0 | 1 | 10 | 0 |
| 0 | 0 | 10 | 0 |
```

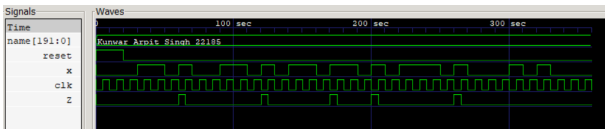
```
| 0 | 0 | 11 | 0 |
| 0 | 1 | 11 | 1 |
| 0 | 1 | 01 | 0 |
For Sequence: 0010100
| Reset | Input x | State | Output Z |
| 0 | 0 | 01 | 0 |
| 0 | 0 | 00 | 0 |
| 0 | 1 | 00 | 0 |
| 0 | 1 | 01 | 0 |
| 0 | 0 | 01 | 0 |
| 0 | 0 | 00 | 0 |
| 0 | 1 | 00 | 0 |
| 0 | 1 | 01 | 0 |
| 0 | 0 | 01 | 0 |
| 0 | 0 | 00 | 0 |
```

Output Screenshots

Terminal output (FSM Implementation)



GTKWave waveform (FSM Implementation)



4. Problem 4

Problem statement

You are given two one-bit input signals (P_A and P_B) and one-bit output signal (O) for the following modular equation $2N(P_A) + N(P_B) = 1 \pmod 4$. In this modular equation, $N(P_A)$ and $N(P_B)$ represent the total number of times the inputs P_A and P_B are high (i.e., logic 1) at each positive clock edge, respectively. The one-bit output signal, O, is set to 1 when the modular equation is satisfied (i.e., $2N(P_A) + N(P_B) = 1 \pmod 4$), and 0 otherwise. An example that sets $O = 1$ at the end of third cycle would be:

- (1^{st} cycle) $P_A = 0$ ($N(P_A) = 0$), $P_B = 0$ ($N(P_B) = 0$),
  $2N(P_A) + N(P_B) = 0 \pmod 4 \rightarrow O = 0$
- (2^{nd} cycle) $P_A = 1$ ($N(P_A) = 1$), $P_B = 1$ ($N(P_B) = 1$),
  $2N(P_A) + N(P_B) = 3 \pmod 4 \rightarrow O = 0$
- (3^{rd} cycle) $P_A = 1$ ($N(P_A) = 2$), $P_B = 0$ ($N(P_B) = 1$),
  $2N(P_A) + N(P_B) = 5 \pmod 4 \rightarrow O = 1$
- (4^{th} cycle) $P_A = 0$ ($N(P_A) = 2$), $P_B = 1$ ($N(P_B) = 2$),
  $2N(P_A) + N(P_B) = 6 \pmod 4 \rightarrow O = 0$

Considering this state diagram implementation, write the Verilog code based on the above description.

Verilog source

```
module state_diagram_implementation (input clk, input reset, input P_A, input
    P_B, output reg O);

    reg [1:0] state;
    reg [1:0] next_state;

    parameter STATE_0 = 2'b00;
    parameter STATE_1 = 2'b01;
    parameter STATE_2 = 2'b10;
    parameter STATE_3 = 2'b11;

    always @(posedge clk or posedge reset) begin
        if (reset) begin
            state <= STATE_0;
            O <= 1'b0;
        end else begin
            state <= next_state;
            O <= (next_state == STATE_1);
        end
    end

    always @(*) begin
        case (state)
```

```
            STATE_0: begin
                case ({P_A, P_B})
                    2'b00: next_state = STATE_0;
                    2'b01: next_state = STATE_1;
                    2'b10: next_state = STATE_2;
                    2'b11: next_state = STATE_3;
                endcase
            end
            STATE_1: begin
                case ({P_A, P_B})
                    2'b00: next_state = STATE_1;
                    2'b01: next_state = STATE_2;
                    2'b10: next_state = STATE_3;
                    2'b11: next_state = STATE_0;
                endcase
            end
            STATE_2: begin
                case ({P_A, P_B})
                    2'b00: next_state = STATE_2;
                    2'b01: next_state = STATE_3;
                    2'b10: next_state = STATE_0;
                    2'b11: next_state = STATE_1;
                endcase
            end
            STATE_3: begin
                case ({P_A, P_B})
                    2'b00: next_state = STATE_3;
                    2'b01: next_state = STATE_0;
                    2'b10: next_state = STATE_1;
                    2'b11: next_state = STATE_2;
                endcase
            end
            default: next_state = STATE_0;
        endcase
    end

endmodule
```

Testbench

```
module state_diagram_implementation_tb;

    // Testbench signals
    reg clk;
    reg reset;
    reg P_A;
    reg P_B;
    wire O;
    reg [8*24-1:0] name;
```

```
10      state_diagram_implementation uut (.clk(clk), .reset(reset), .P_A(P_A), .P_B
            (P_B), .O(O));
11
12      always #5 clk = ~clk;
13
14      integer N_PA, N_PB;
15      integer expected_value;
16      integer cycle_count;
17
18      initial begin
19          name = "␣Kunwar␣Arpit␣Singh␣22185";
20          $display("Kunwar␣Arpit␣Singh");
21          $dumpfile("assign4_problem4_state_diagram_implementation.vcd");
22          $dumpvars(1, state_diagram_implementation_tb);
23          $display("Kunwar␣Arpit␣Singh");
24
25          clk = 0;
26          reset = 1;
27          P_A = 0;
28          P_B = 0;
29          N_PA = 0;
30          N_PB = 0;
31          cycle_count = 0;
32
33          #10;
34          reset = 0;
35
36          $display("|␣Cycle␣|␣P_A␣|␣P_B␣|␣N(P_A)␣|␣N(P_B)␣|␣2*N(P_A)+N(P_B)␣|␣
                mod4␣|␣O␣|␣Expected␣|");
37          $display("-------------------------------------------------------");
38
39          test_cycle(0, 0);
40          test_cycle(1, 1);
41          test_cycle(1, 0);
42          test_cycle(0, 1);
43
44          test_cycle(1, 1);
45          test_cycle(0, 0);
46          test_cycle(1, 0);
47          test_cycle(0, 1);
48
49          #20;
50          $finish;
51      end
52
53      task test_cycle;
54          input pa_val;
55          input pb_val;
56          begin
57              P_A = pa_val;
58              P_B = pb_val;
```

```
59
60              @(posedge clk);
61
62              if (pa_val) N_PA = N_PA + 1;
63              if (pb_val) N_PB = N_PB + 1;
64
65              cycle_count = cycle_count + 1;
66
67              #2;
68
69              expected_value = (2 * N_PA + N_PB) % 4;
70
71              $display("|␣␣␣%0d␣␣␣|␣␣%0d␣␣|␣␣%0d␣␣|␣␣␣␣%0d␣␣␣|␣␣␣␣%0d␣␣␣|␣␣␣␣␣␣␣␣
                    %0d␣␣␣␣␣␣␣␣␣|␣␣%0d␣␣␣|␣%0d␣|␣␣␣␣%0d␣␣␣␣␣|", cycle_count, P_A, P_B,
                     N_PA, N_PB, 2*N_PA + N_PB, expected_value, 0, (expected_value
                    == 1));
72
73          end
74      endtask
75
76  endmodule
```

Output in Terminal

```
1  Kunwar Arpit Singh
2  VCD info: dumpfile assign4_problem4_state_diagram_implementation.vcd opened
       for output.
3  Kunwar Arpit Singh
4  | Cycle | P_A | P_B | N(P_A) | N(P_B) | 2*N(P_A)+N(P_B) | mod4 | O | Expected
       |
5  ---------------------------------------------------------------------------------

6  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
7  | 2 | 1 | 1 | 1 | 1 | 3 | 3 | 0 | 0 |
8  | 3 | 1 | 0 | 2 | 1 | 5 | 1 | 1 | 1 |
9  | 4 | 0 | 1 | 2 | 2 | 6 | 2 | 0 | 0 |
10 | 5 | 1 | 1 | 3 | 3 | 9 | 1 | 1 | 1 |
11 | 6 | 0 | 0 | 3 | 3 | 9 | 1 | 1 | 1 |
12 | 7 | 1 | 0 | 4 | 3 | 11 | 3 | 0 | 0 |
13 | 8 | 0 | 1 | 4 | 4 | 12 | 0 | 0 | 0 |
```

Output Screenshots

Terminal output (State Diagram Implementation)

```
Kunwar Arpit Singh
VCD info: dumpfile assign4_problem4_state_diagram_implementation.vcd opened for output.
Kunwar Arpit Singh
| Cycle | P_A | P_B | N(P_A) | N(P_B) | 2*N(P_A)+N(P_B) | mod4 | O | Expected |
-------------------------------------------------------------------------------
|   1   |  0  |  0  |    0   |    0   |        0        |  0   | 0 |    0     |
|   2   |  1  |  1  |    1   |    1   |        3        |  3   | 0 |    0     |
|   3   |  1  |  0  |    2   |    1   |        5        |  1   | 1 |    1     |
|   4   |  0  |  1  |    2   |    2   |        6        |  2   | 0 |    0     |
|   5   |  1  |  1  |    3   |    3   |        9        |  1   | 1 |    1     |
|   6   |  0  |  0  |    3   |    3   |        9        |  1   | 1 |    1     |
|   7   |  1  |  0  |    4   |    3   |        11       |  3   | 0 |    0     |
|   8   |  0  |  1  |    4   |    4   |        12       |  0   | 0 |    0     |
```

GTKWave waveform (State Diagram Implementation)