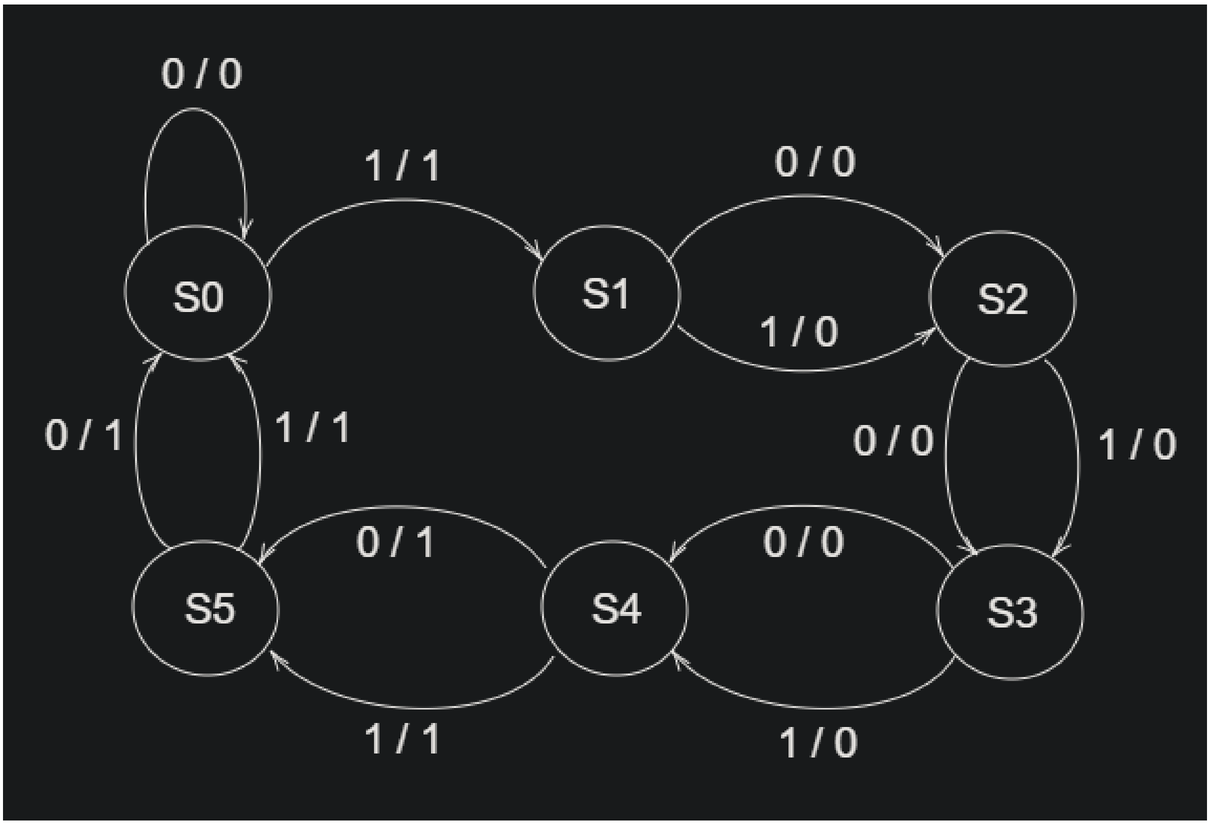
Experiment 7: Exercise with Verilog Simulations – FSM

Design a sequence generator that generates the following serial sequence:
Since my entry number ends in 43, the sequence is {1,0,0,0,1,1}

Design steps:
The circuit has one input, X, and one serial output, Y. If X is 1 at the rising clock edge, the machine should generate the desired output sequence as Y. If X is 1 while the circuit is busy generating its output sequence, the input X should be ignored. If X is 0, and the circuit is idle, the output Y should be 0.

State diagram (Mealy Machine)



Therefore, we will be needing 4 flip flops. 3 for each bit of a state, and finally 1 for output
State 0: 000

State 1: 001
State 2: 010
State 3: 011
State 4: 100
State 5: 101
State 6: 110

State Table:

Present State (Q1Q2Q3)	Input (X)	Next State (Q1Q2Q3)*	D1	D2	D3	Output (Y)
000	0	000	0	0	0	0
000	1	001	0	0	1	1
001	0	010	0	1	0	0
001	1	010	0	1	0	0
010	0	011	0	1	1	0
010	1	011	0	1	1	0
011	0	100	1	0	0	0
011	1	100	1	0	0	0
100	0	101	1	0	1	1
100	1	101	1	0	1	1
101	0	000	0	0	0	1
101	1	000	0	0	0	1

In following K-maps, for simplicity
A=Q1
B=Q2
C=Q3
D=Input(X)

D1: BC + AC’

Map

	C’.D’	C’.D	C.D	C.D’
A’.B’	0	0	0	0
A’.B	0	0	1	1
A.B	x	x	x	x
A.B’	1	1	0	0

Groups

(6,7,14,15)	B.C
(8,9,12,13)	A.C’

D2: BC’ + A’B’C

Map

	C'.D'	C'.D	C.D	C.D'
A'.B'	0	0	1	1
A'.B	1	1	0	0
A.B	x	x	x	x
A.B'	0	0	0	0

Groups

(4,5,12,13)	B.C'
(2,3)	A'.B'.C

D3: C'D + BC' + AC'

Map

	C'.D'	C'.D	C.D	C.D'
A'.B'	0	1	0	0
A'.B	1	1	0	0
A.B	x	x	x	x
A.B'	1	1	0	0

Groups

(1,5,9,13)	C'.D
(4,5,12,13)	B.C'
(8,9,12,13)	A.C'

Y: A + B'C'D

Map

	C'.D'	C'.D	C.D	C.D'
A'.B'	0	1	0	0
A'.B	0	0	0	0
A.B	x	x	x	x
A.B'	1	1	1	1

Groups

(8,9,10,11,12,13,14,15)	A
(1,9)	B'.C'.D

Therefore,

D1= Q2.Q3 + Q1.Q3’
D2= Q2.Q3’ + Q1’.Q2’.Q3
D3= Q3’.X + Q2.Q3’ + Q1.Q3’
Y= Q1 + Q2’.Q3’.X

Verilog Code:

1. D_ff.v

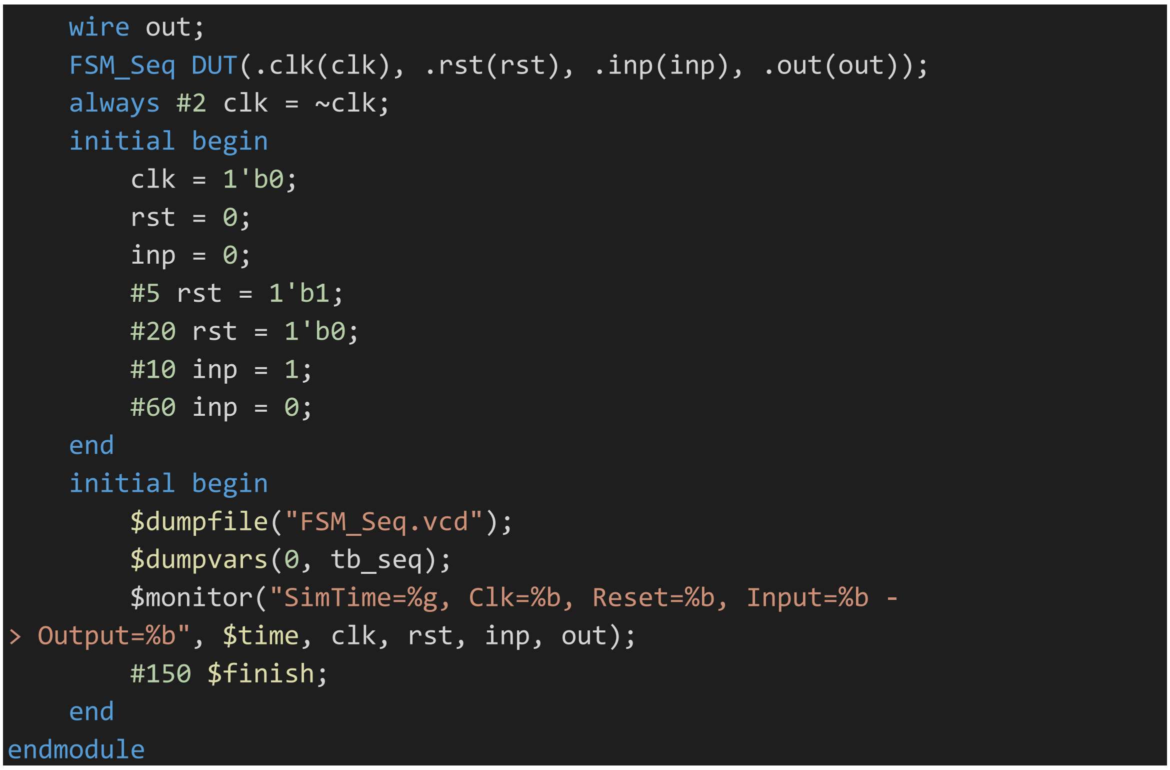
```
module D_ff(d, clk, q, rst, set);
    input d, clk, rst, set;
    output q;
    reg q;

    always @(posedge clk) begin
        if (rst) begin
            if (set) begin
                q <= 1;
            end
            else begin
                q <=0;
            end
        end else begin
        q <= d;
        end
    end
endmodule
```

2. FSM_Seq.v

```
module FSM_Seq(clk, rst, inp, out);
    input clk, rst, inp;
    output out;
    wire q1, q2, q3;
    D_ff d1((q2&&q3)||(q1&&~q3), clk, q1, rst, 1'b0);
    D_ff d2((q2&&~q3)||(~q1&&~q2&&q3), clk, q2, rst, 1'b0);
    D_ff d3((~q3&&inp)||(q2&&~q3)||(q1&&~q3), clk, q3, rst, 1'b0);
    D_ff d4(q1||(~q2&&~q3&&inp) , clk, out, rst, 1'b0);
endmodule
```

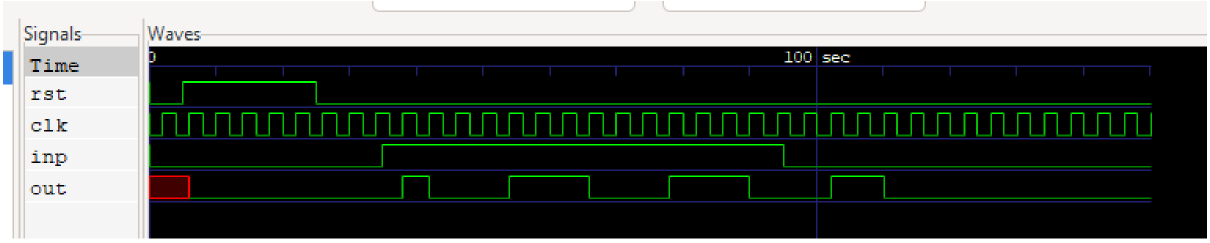
3. Tb_seq.v

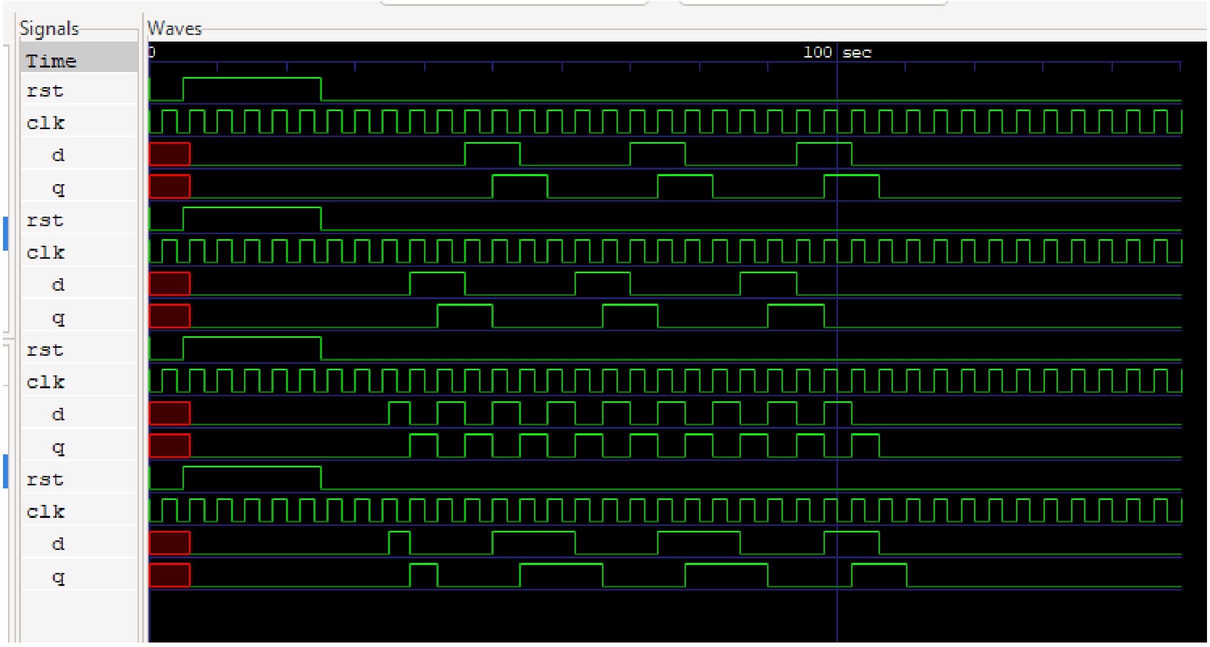
```
module tb_seq();
    reg clk, rst, inp;
```

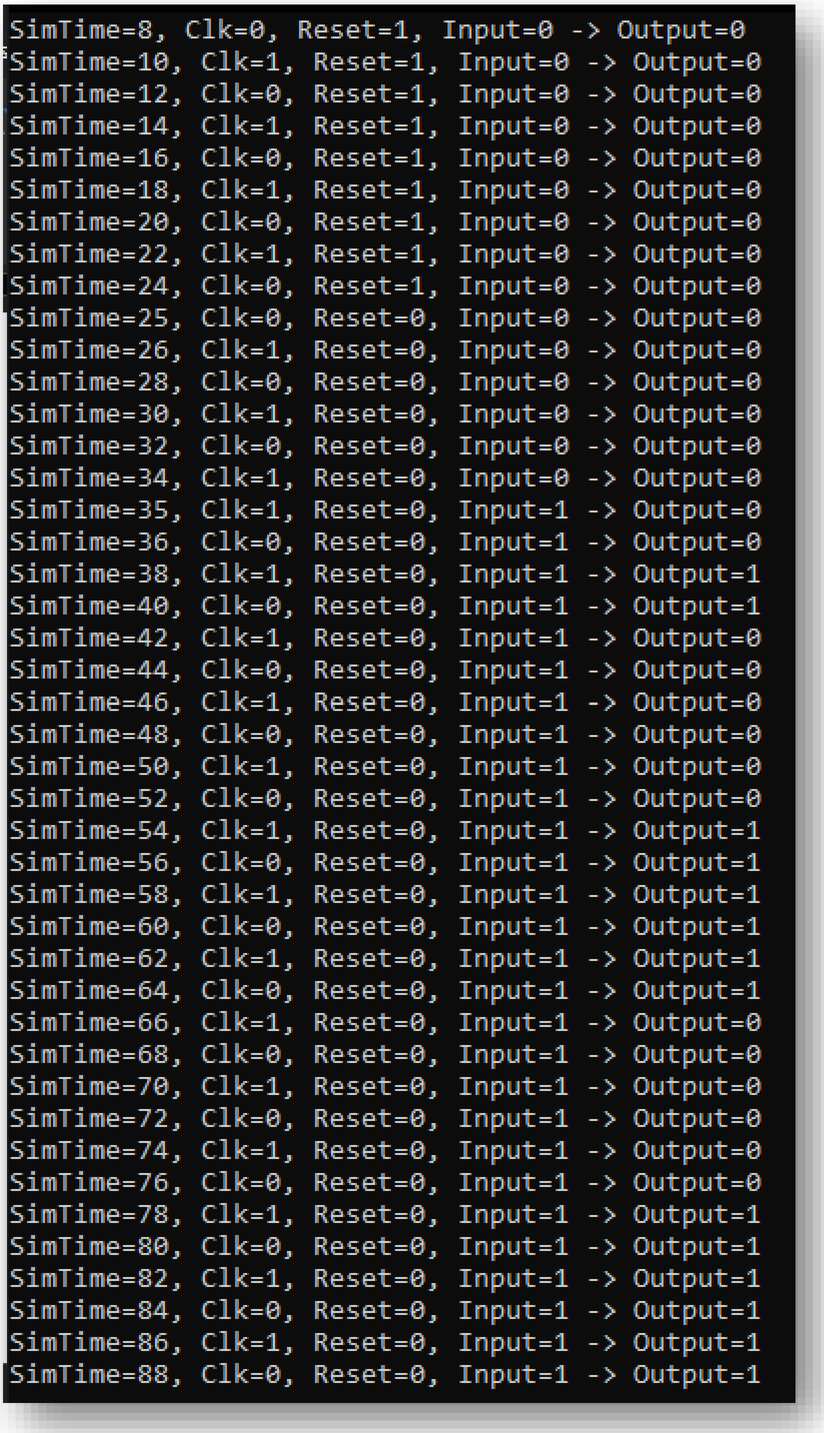


```
    wire out;
    FSM_Seq DUT(.clk(clk), .rst(rst), .inp(inp), .out(out));
    always #2 clk = ~clk;
    initial begin
        clk = 1'b0;
        rst = 0;
        inp = 0;
        #5 rst = 1'b1;
        #20 rst = 1'b0;
        #10 inp = 1;
        #60 inp = 0;
    end
    initial begin
        $dumpfile("FSM_Seq.vcd");
        $dumpvars(0, tb_seq);
        $monitor("SimTime=%g, Clk=%b, Reset=%b, Input=%b -
> Output=%b", $time, clk, rst, inp, out);
        #150 $finish;
    end
endmodule
```

Outputs and waveforms:







```
SimTime=8, Clk=0, Reset=1, Input=0 -> Output=0
SimTime=10, Clk=1, Reset=1, Input=0 -> Output=0
SimTime=12, Clk=0, Reset=1, Input=0 -> Output=0
SimTime=14, Clk=1, Reset=1, Input=0 -> Output=0
SimTime=16, Clk=0, Reset=1, Input=0 -> Output=0
SimTime=18, Clk=1, Reset=1, Input=0 -> Output=0
SimTime=20, Clk=0, Reset=1, Input=0 -> Output=0
SimTime=22, Clk=1, Reset=1, Input=0 -> Output=0
SimTime=24, Clk=0, Reset=1, Input=0 -> Output=0
SimTime=25, Clk=0, Reset=0, Input=0 -> Output=0
SimTime=26, Clk=1, Reset=0, Input=0 -> Output=0
SimTime=28, Clk=0, Reset=0, Input=0 -> Output=0
SimTime=30, Clk=1, Reset=0, Input=0 -> Output=0
SimTime=32, Clk=0, Reset=0, Input=0 -> Output=0
SimTime=34, Clk=1, Reset=0, Input=0 -> Output=0
SimTime=35, Clk=1, Reset=0, Input=1 -> Output=0
SimTime=36, Clk=0, Reset=0, Input=1 -> Output=0
SimTime=38, Clk=1, Reset=0, Input=1 -> Output=1
SimTime=40, Clk=0, Reset=0, Input=1 -> Output=1
SimTime=42, Clk=1, Reset=0, Input=1 -> Output=0
SimTime=44, Clk=0, Reset=0, Input=1 -> Output=0
SimTime=46, Clk=1, Reset=0, Input=1 -> Output=0
SimTime=48, Clk=0, Reset=0, Input=1 -> Output=0
SimTime=50, Clk=1, Reset=0, Input=1 -> Output=0
SimTime=52, Clk=0, Reset=0, Input=1 -> Output=0
SimTime=54, Clk=1, Reset=0, Input=1 -> Output=1
SimTime=56, Clk=0, Reset=0, Input=1 -> Output=1
SimTime=58, Clk=1, Reset=0, Input=1 -> Output=1
SimTime=60, Clk=0, Reset=0, Input=1 -> Output=1
SimTime=62, Clk=1, Reset=0, Input=1 -> Output=1
SimTime=64, Clk=0, Reset=0, Input=1 -> Output=1
SimTime=66, Clk=1, Reset=0, Input=1 -> Output=0
SimTime=68, Clk=0, Reset=0, Input=1 -> Output=0
SimTime=70, Clk=1, Reset=0, Input=1 -> Output=0
SimTime=72, Clk=0, Reset=0, Input=1 -> Output=0
SimTime=74, Clk=1, Reset=0, Input=1 -> Output=0
SimTime=76, Clk=0, Reset=0, Input=1 -> Output=0
SimTime=78, Clk=1, Reset=0, Input=1 -> Output=1
SimTime=80, Clk=0, Reset=0, Input=1 -> Output=1
SimTime=82, Clk=1, Reset=0, Input=1 -> Output=1
SimTime=84, Clk=0, Reset=0, Input=1 -> Output=1
SimTime=86, Clk=1, Reset=0, Input=1 -> Output=1
SimTime=88, Clk=0, Reset=0, Input=1 -> Output=1
```