HACETTEPE UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BBM 233 LOGIC DESIGN LAB – 2021 FALL

# Lab Experiment 5 Verilog Sequential

December 23, 2021

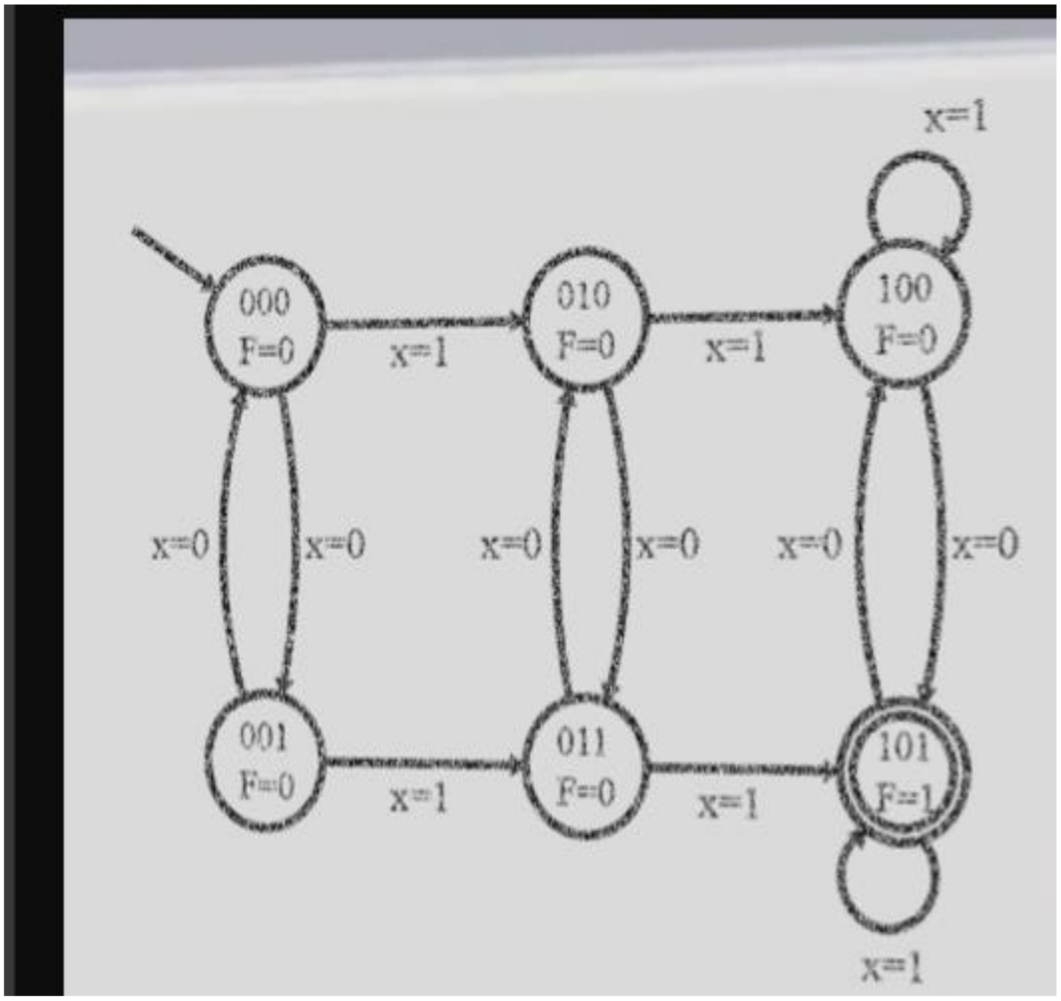
Student Name:
Umut GÜNGÖR

Student Number:
b21946198

## 1 Problem Definition

We are asked to design a sequential circuit. The state diagram is given. Firstly, we have to convert it to state table. Then, we obtain 4 different Karnaugh Maps. We are going to use D flip-flops in this experiment.

## 2 Convert State Diagram to State Table



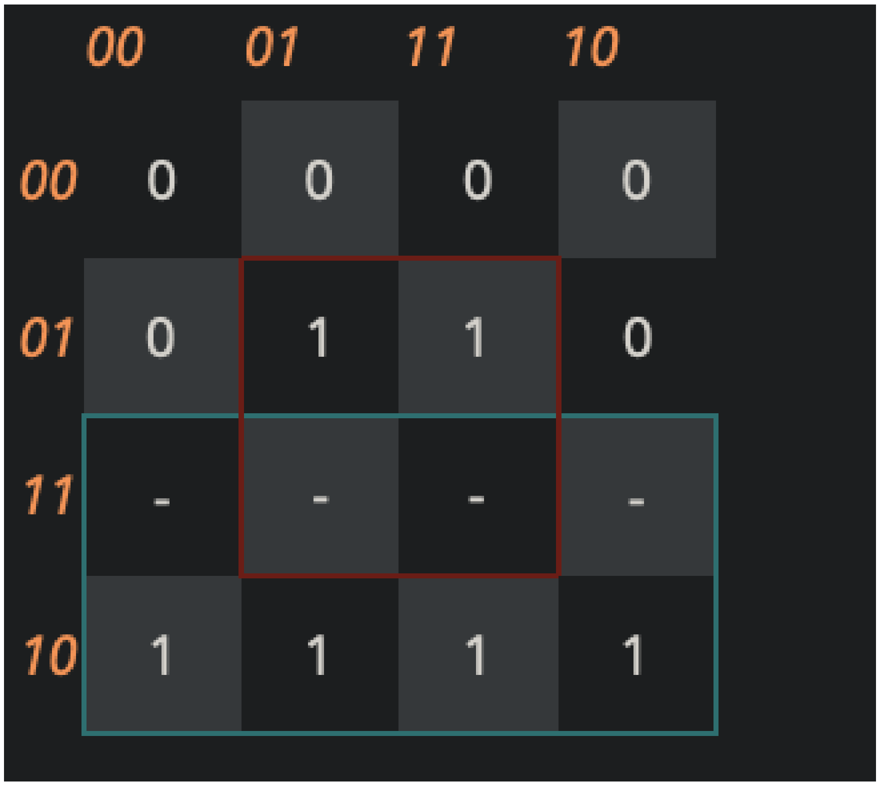
State Diagram

| Present State | | | Input | Next State | | | Output |
|---|---|---|---|---|---|---|---|
| A | B | C | x | A+ | B+ | C+ | F |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | X | X | X | X |
| 1 | 1 | 0 | 1 | X | X | X | X |
| 1 | 1 | 1 | 0 | X | X | X | X |
| 1 | 1 | 1 | 1 | X | X | X | X |

State Table

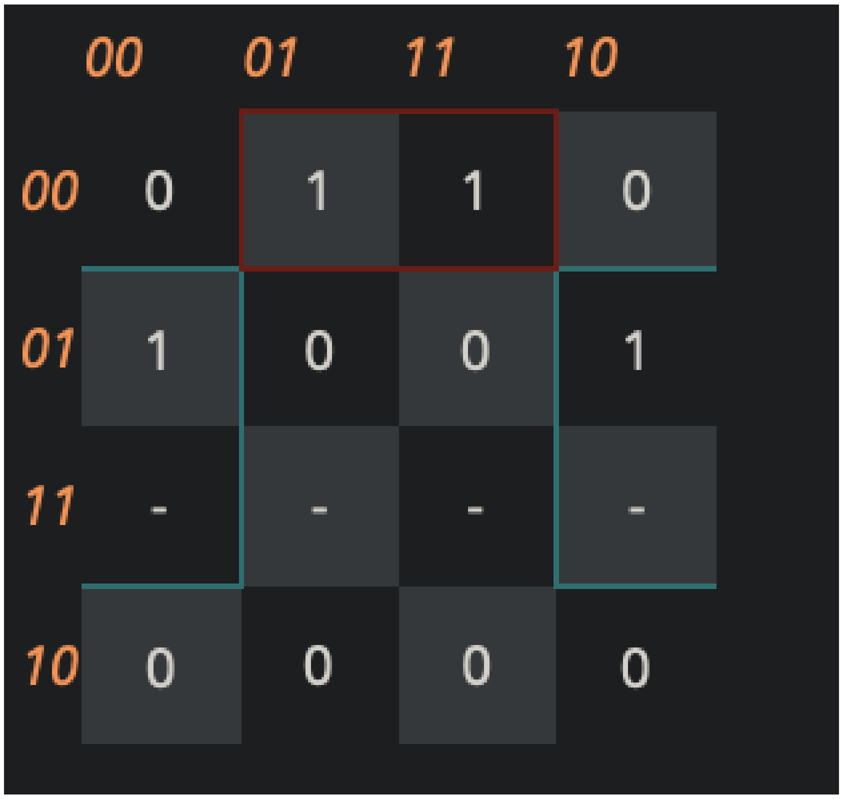
## 3 K-Maps

We have 3 states so, we need 3 D flip-flops. They store the state information. I named these flip-flops DA, DB and DC.



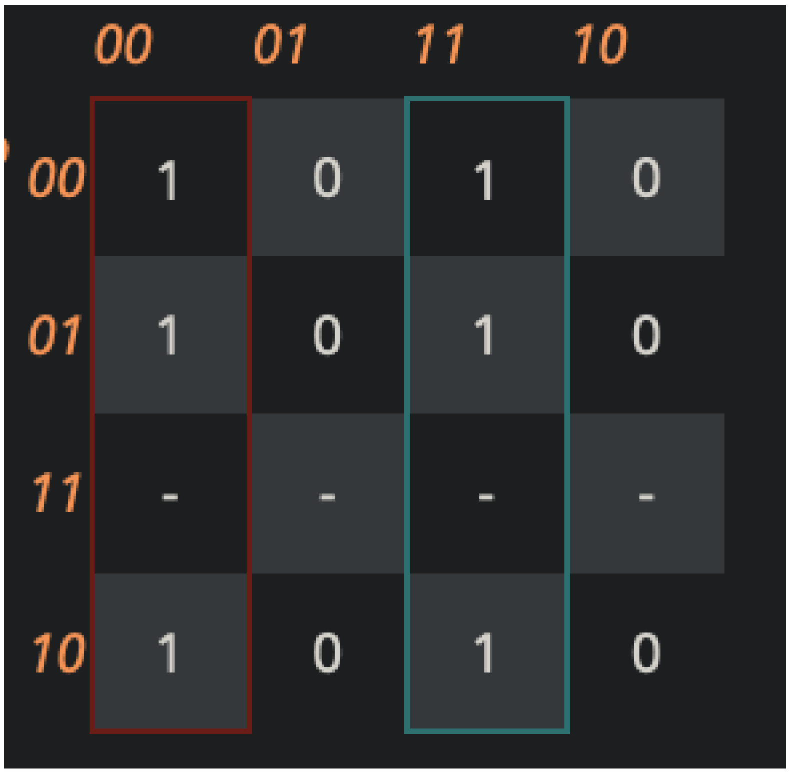
| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 1 | 0 |
| 11 | - | - | - | - |
| 10 | 1 | 1 | 1 | 1 |

DA = A + Bx



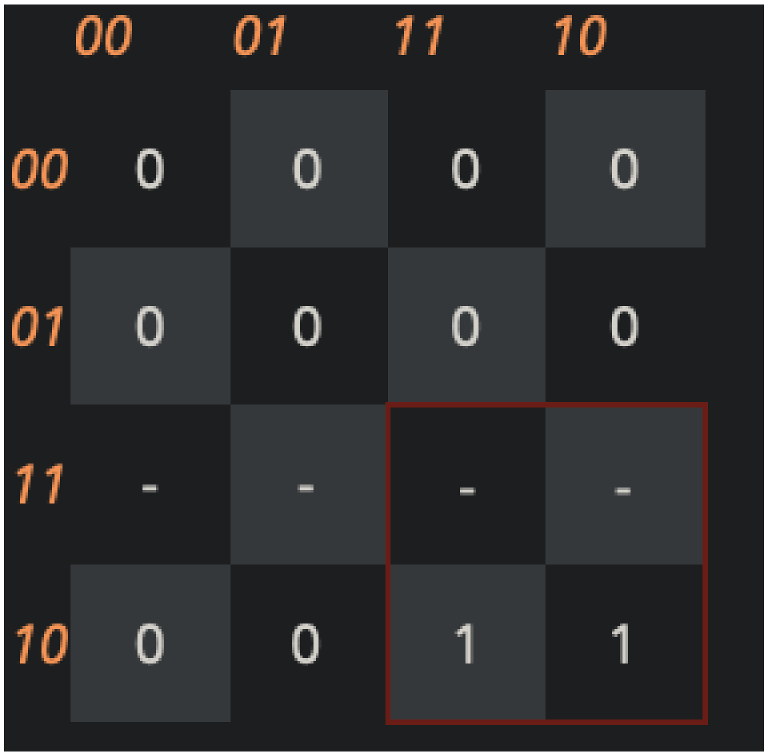
| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 1 | 0 |
| 01 | 1 | 0 | 0 | 1 |
| 11 | - | - | - | - |
| 10 | 0 | 0 | 0 | 0 |

DB = Bx’ + A’B’x



| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 1 | 0 |
| 01 | 1 | 0 | 1 | 0 |
| 11 | - | - | - | - |
| 10 | 1 | 0 | 1 | 0 |

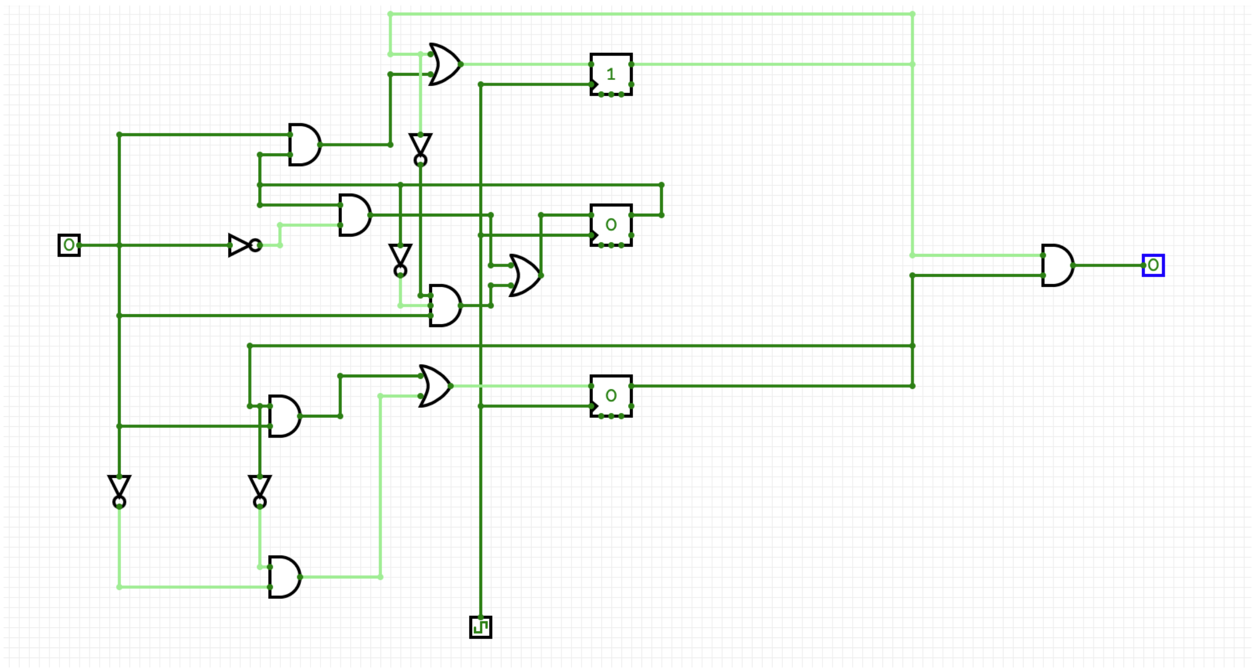
DC = C’x’ + Cx



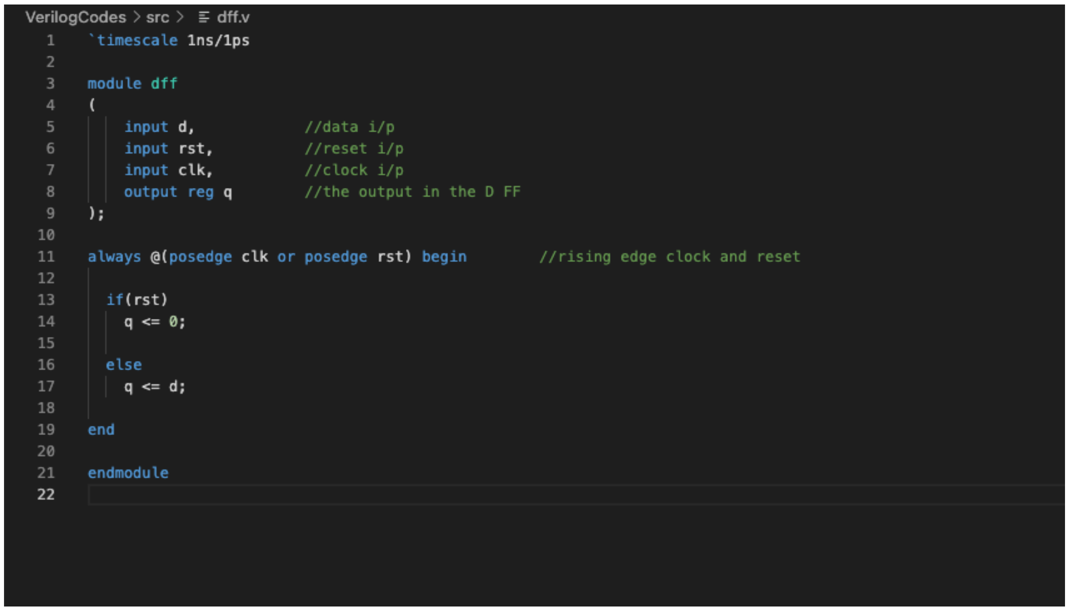
| | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | - | - | - | - |
| 10 | 0 | 0 | 1 | 1 |

F = AC

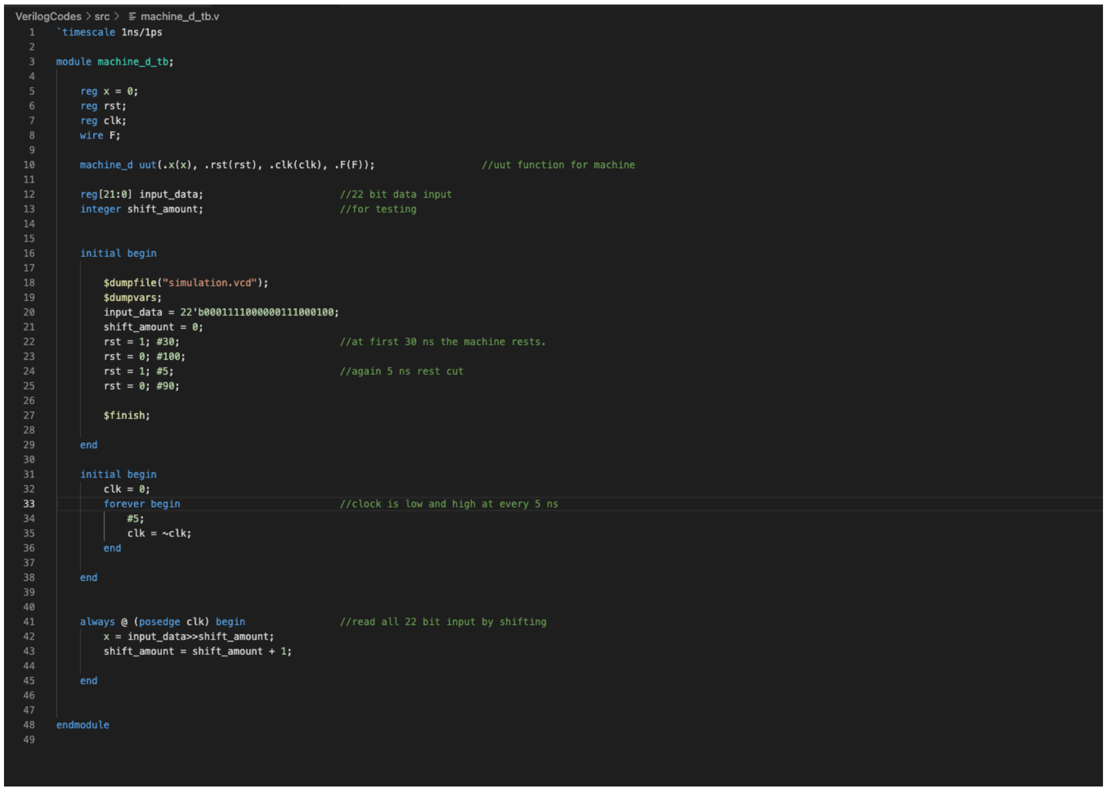
## 4 Circuit



## 5 Verilog Codes

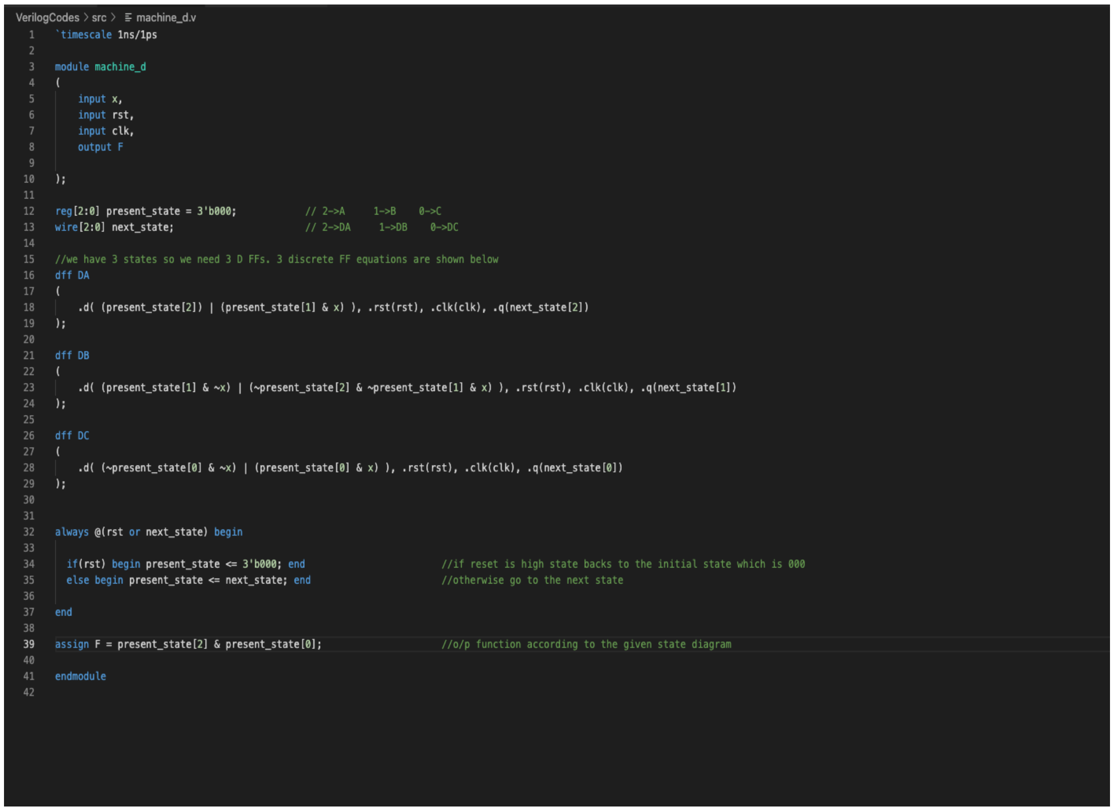


```verilog
`timescale 1ns/1ps

module dff
(
    input d,            //data i/p
    input rst,          //reset i/p
    input clk,          //clock i/p
    output reg q        //the output in the D FF
);

always @(posedge clk or posedge rst) begin        //rising edge clock and reset

  if(rst)
    q <= 0;

  else
    q <= d;

end

endmodule
```



VerilogCodes > src > machine_d_tb.v

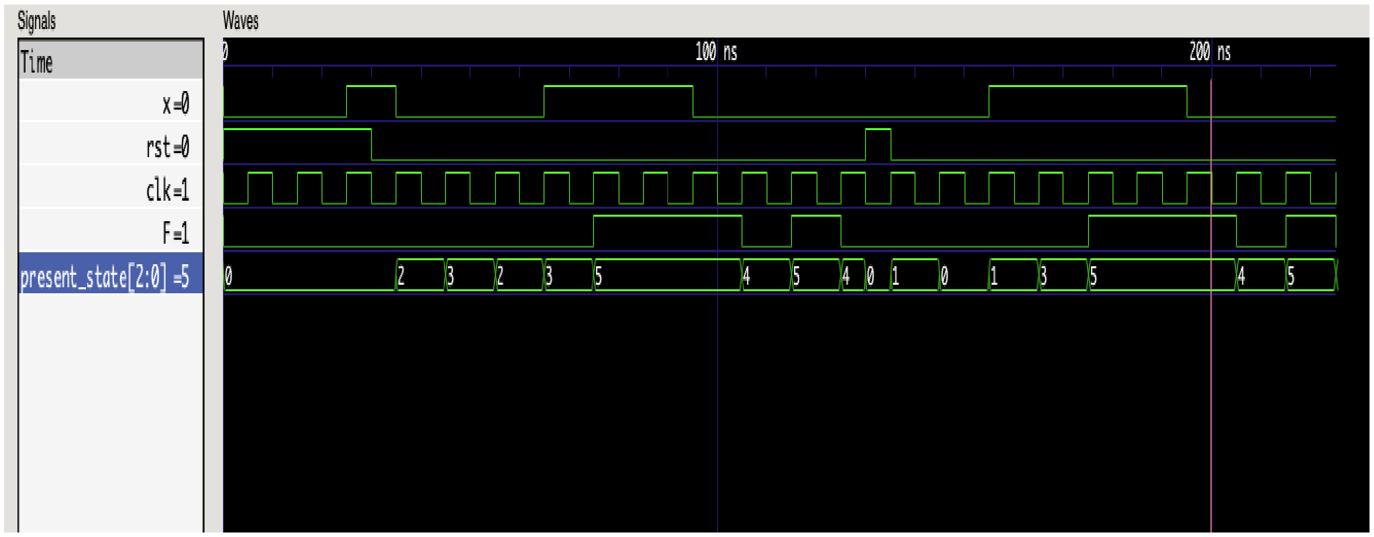
```verilog
`timescale 1ns/1ps

module machine_d_tb;

    reg x = 0;
    reg rst;
    reg clk;
    wire F;

    machine_d uut(.x(x), .rst(rst), .clk(clk), .F(F));                  //uut function for machine

    reg[21:0] input_data;                       //22 bit data input
    integer shift_amount;                       //for testing


    initial begin

        $dumpfile("simulation.vcd");
        $dumpvars;
        input_data = 22'b0001111000000111000100;
        shift_amount = 0;
        rst = 1; #30;                           //at first 30 ns the machine rests.
        rst = 0; #100;
        rst = 1; #5;                            //again 5 ns rest cut
        rst = 0; #90;

        $finish;

    end

    initial begin
        clk = 0;
        forever begin                           //clock is low and high at every 5 ns
            #5;
            clk = ~clk;
        end

    end


    always @ (posedge clk) begin                //read all 22 bit input by shifting
        x = input_data>>shift_amount;
        shift_amount = shift_amount + 1;

    end


endmodule
```



VerilogCodes > src > machine_d.v

```verilog
`timescale 1ns/1ps

module machine_d
(
    input x,
    input rst,
    input clk,
    output F

);

reg[2:0] present_state = 3'b000;            // 2->A     1->B    0->C
wire[2:0] next_state;                       // 2->DA     1->DB    0->DC

//we have 3 states so we need 3 D FFs. 3 discrete FF equations are shown below
dff DA
(
    .d( (present_state[2]) | (present_state[1] & x) ), .rst(rst), .clk(clk), .q(next_state[2])
);

dff DB
(
    .d( (present_state[1] & ~x) | (~present_state[2] & ~present_state[1] & x) ), .rst(rst), .clk(clk), .q(next_state[1])
);

dff DC
(
    .d( (~present_state[0] & ~x) | (present_state[0] & x) ), .rst(rst), .clk(clk), .q(next_state[0])
);


always @(rst or next_state) begin

  if(rst) begin present_state <= 3'b000; end                       //if reset is high state backs to the initial state which is 000
  else begin present_state <= next_state; end                      //otherwise go to the next state

end

assign F = present_state[2] & present_state[0];                    //o/p function according to the given state diagram

endmodule
```

## 6 Waveform



## 7 Results

We can see that in wavefrom the output of the circuit is only high when the state is 101 which can be shown in decimal 5. Since this is a Moore Machine output is not dependent on input. Clock and reset are rising edge sensitive. Clock's period is 10 nanoseconds. When reset is high which is 1 the machine backs to the initial state that is 000.

## 8 References

- BBM 233 Verilog pdf
- BBm 231 lecture notes
- Previous years student’s solutions
- chipverify.com