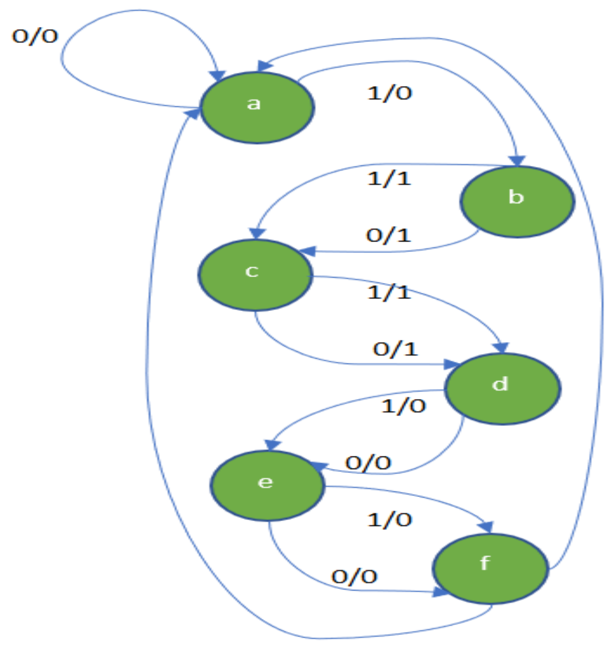
ELP Assignment 3 Report

Ritika Soni

2020MT10838

State Diagram for FSM :



Here, a represents the idle state. The system starts at idle state when we get input 1,it produces output sequence {0 1 1 0 0 0} ,irrespective of the input in the next 5 cycles.

State Transition Table :

Present State	*Next Sate at x = 0*	*Next State at x =1*	*Output at x = 0*	*Output at x = 1*
a	a	b	0	0
b	c	c	0	1
c	d	d	0	1
d	e	e	1	0
e	f	f	1	0
f	a	a	1	0

Constructing State table with binary coded states

The encoding of the states is done as follows:

- a maps to 000
- b maps to 001
- c maps to 010
- d maps to 011
- e maps to 100
- f maps to 101

Now, we require 1 bit to represent the output, Y.

Therefore, we need a total of **3 D Flip flops to implement the given FSM**, 1 for each bit to be encoded. The output Y can be implemented using simple combinational logic.

State	Present State			Input	Next State			Output	D Flip-Flop		
	Q2	Q1	Q0	x	$Q2^N$	$Q1^N$	$Q0^N$	Y	D2	D1	D0
a	0	0	0	0	0	0	0	0	0	0	0
a	0	0	0	1	0	0	1	0	0	0	1
b	0	0	1	0	0	1	0	1	0	1	0
b	0	0	1	1	0	1	0	1	0	1	0
c	0	1	0	0	0	1	1	1	0	1	1
c	0	1	0	1	0	1	1	1	0	1	1
d	0	1	1	0	1	0	0	0	1	0	0
d	0	1	1	1	1	0	0	0	1	0	0
e	1	0	0	0	1	0	1	0	1	0	1
e	1	0	0	1	1	0	1	0	1	0	1
f	1	0	1	0	0	0	0	0	0	0	0
f	1	0	1	1	0	0	0	0	0	0	0

Karnaugh maps to generate each of the inputs

Using the K-maps shown below, we find the minimised expressions for flip-flops D2, D1, D0 and Y in terms of the current state variables: Q2, Q1, Q0, and the input x.

$$Q2^N = Q_2Q_0' + Q_1Q_0$$

Q_0x --- Q_2Q_1 |	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	x	x	x	x
10	1	1	0	0

$$Q1^N = Q_1Q_0' + Q_2'Q_1'Q_0$$

Q_0x --- Q_2Q_1 |	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	x	x	x	0
10	0	0	0	0

$$Q0^N = Q_0x + Q_2Q_0' + Q_1Q_0'$$

Q_0x --- Q_2Q_1 |	00	01	11	10
00	0	1	0	0
01	1	1	0	0
11	x	x	x	x
10	1	1	0	0

Output = $Q_2'Q_1'Q_0 + Q_1Q_0'$

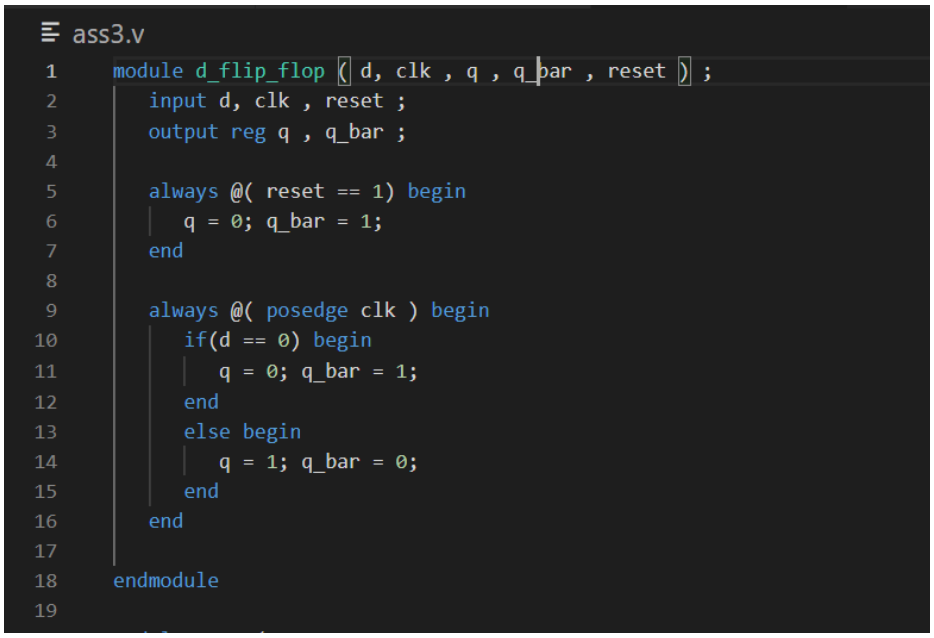
Q_0 x — Q_2Q_1 |	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	x	x	x	x
10	0	0	0	0

Since we are using D-FFs, the bits used to represent the next state and the output are the same as the inputs to the D-FFs. Therefore **D0 = $Q0^N$, D1 = $Q1^N$, D2 = $Q2^N$**

The simplified expressions obtained from the K-maps drawn above are as follows:

- **D0 =** $Q_0'x + Q_2Q_0' + Q_1Q_0'$
- **D1** = $Q_1Q_0' + Q_2'Q_1'Q_0$
- **D2** = $Q_2Q_0' + Q_1Q_0$
- **Output** = $Q_1Q_0' + Q_2'Q_1'Q_0$

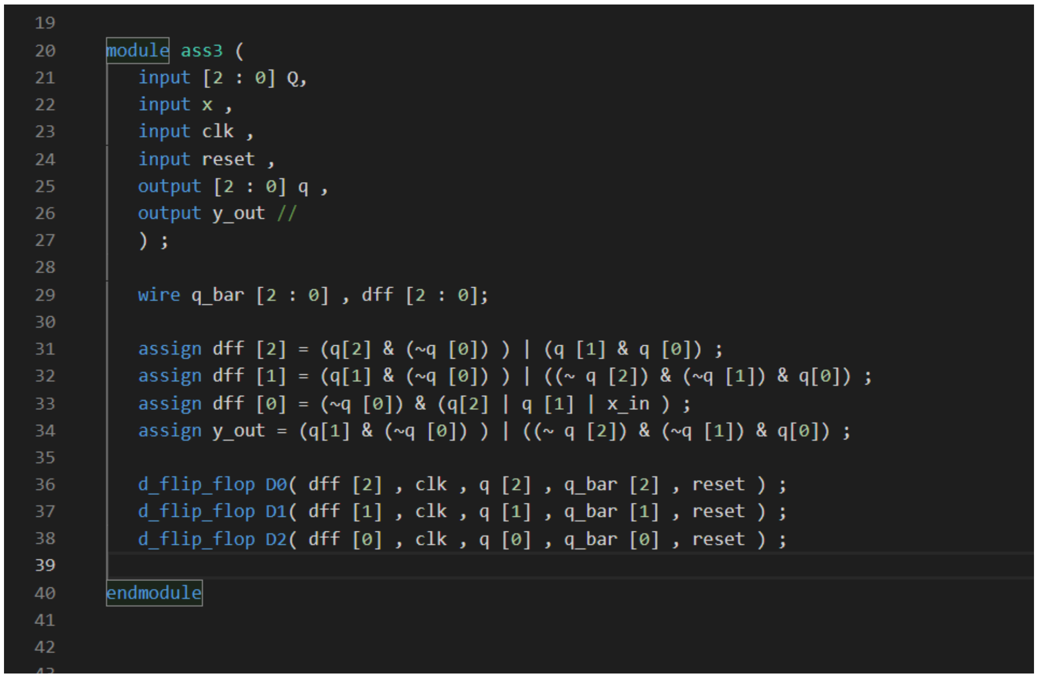
Verilog Code for D Flip Flop :



ass3.v

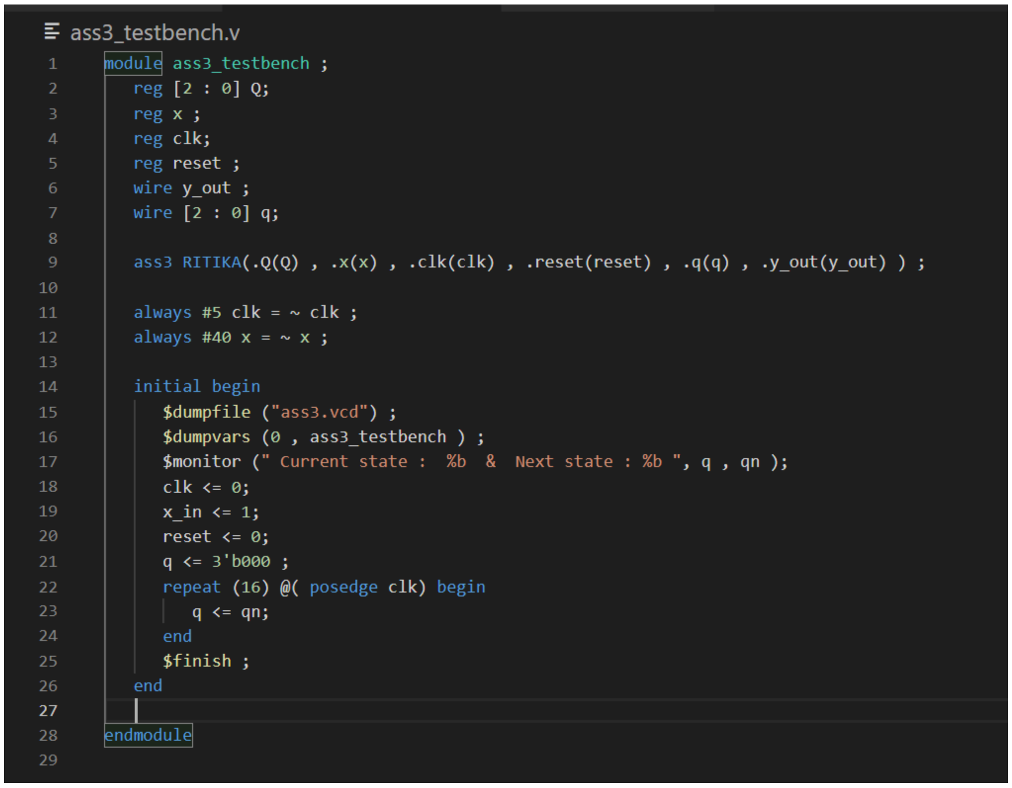
```
module d_flip_flop ( d, clk , q , q_bar , reset ) ;
   input d, clk , reset ;
   output reg q , q_bar ;

   always @( reset == 1) begin
      q = 0; q_bar = 1;
   end

   always @( posedge clk ) begin
      if(d == 0) begin
         q = 0; q_bar = 1;
      end
      else begin
         q = 1; q_bar = 0;
      end
   end

endmodule
```

Verilog Code for Sequence Generation :



```verilog

module ass3 (
    input [2 : 0] Q,
    input x ,
    input clk ,
    input reset ,
    output [2 : 0] q ,
    output y_out //
    ) ;

    wire q_bar [2 : 0] , dff [2 : 0];

    assign dff [2] = (q[2] & (~q [0]) ) | (q [1] & q [0]) ;
    assign dff [1] = (q[1] & (~q [0]) ) | ((~ q [2]) & (~q [1]) & q[0]) ;
    assign dff [0] = (~q [0]) & (q[2] | q [1] | x_in ) ;
    assign y_out = (q[1] & (~q [0]) ) | ((~ q [2]) & (~q [1]) & q[0]) ;

    d_flip_flop D0( dff [2] , clk , q [2] , q_bar [2] , reset ) ;
    d_flip_flop D1( dff [1] , clk , q [1] , q_bar [1] , reset ) ;
    d_flip_flop D2( dff [0] , clk , q [0] , q_bar [0] , reset ) ;

endmodule


```

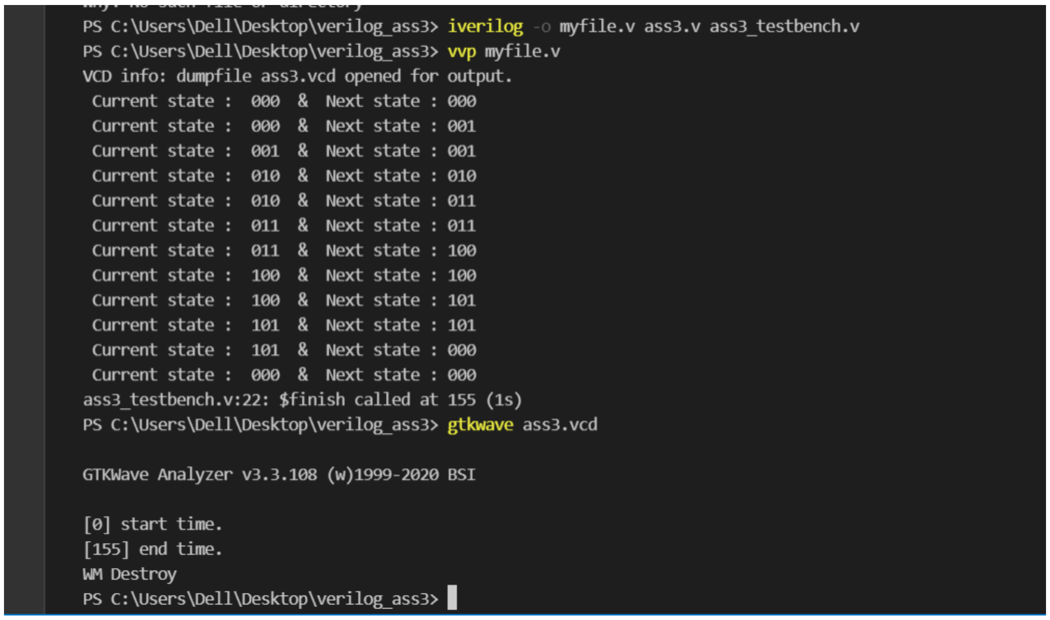
Verilog Code for Testbench :



ass3_testbench.v

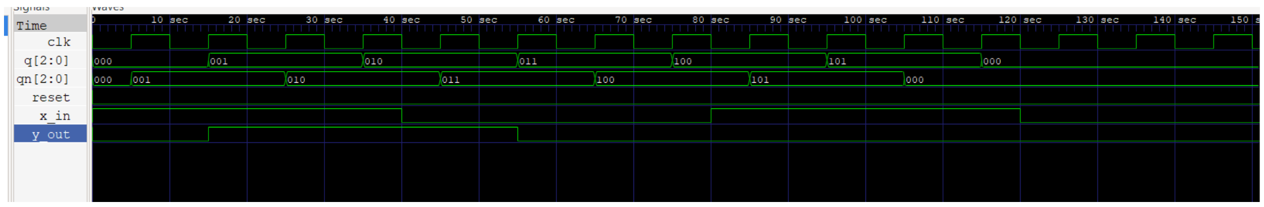
```verilog
module ass3_testbench ;
    reg [2 : 0] Q;
    reg x ;
    reg clk;
    reg reset ;
    wire y_out ;
    wire [2 : 0] q;

    ass3 RITIKA(.Q(Q) , .x(x) , .clk(clk) , .reset(reset) , .q(q) , .y_out(y_out) ) ;

    always #5 clk = ~ clk ;
    always #40 x = ~ x ;

    initial begin
        $dumpfile ("ass3.vcd") ;
        $dumpvars (0 , ass3_testbench ) ;
        $monitor (" Current state :  %b  &  Next state : %b ", q , qn );
        clk <= 0;
        x_in <= 1;
        reset <= 0;
        q <= 3'b000 ;
        repeat (16) @( posedge clk) begin
            q <= qn;
        end
        $finish ;
    end

endmodule

```

Output of Testbench :



```
PS C:\Users\Dell\Desktop\verilog_ass3> iverilog -o myfile.v ass3.v ass3_testbench.v
PS C:\Users\Dell\Desktop\verilog_ass3> vvp myfile.v
VCD info: dumpfile ass3.vcd opened for output.
 Current state :  000  &  Next state : 000
 Current state :  000  &  Next state : 001
 Current state :  001  &  Next state : 001
 Current state :  010  &  Next state : 010
 Current state :  010  &  Next state : 011
 Current state :  011  &  Next state : 011
 Current state :  011  &  Next state : 100
 Current state :  100  &  Next state : 100
 Current state :  100  &  Next state : 101
 Current state :  101  &  Next state : 101
 Current state :  101  &  Next state : 000
 Current state :  000  &  Next state : 000
ass3_testbench.v:22: $finish called at 155 (1s)
PS C:\Users\Dell\Desktop\verilog_ass3> gtkwave ass3.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[155] end time.
WM Destroy
PS C:\Users\Dell\Desktop\verilog_ass3>
```

Waveform of the Output by Gtkwave :



Here we can see clearly, that when the FSM in the idle state, if the input is 0, then it stays in the idle state. If the input is 1, it starts generating the desired output sequence, irrespective of the input in the next 5 clock cycles. **In my case the output is 0 1 1 0 0 0 which can be clearly seen in the gtkwave (last waveform).**

Thank You