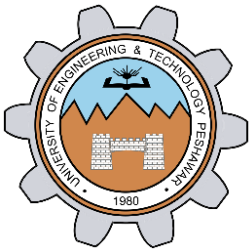
FSM OF BCD COUNTER

LAB # 06



Spring 2021

CSE-308L Digital System Design Lab

Submitted by: **ASHIQ ULLAH**

Registration No.: **18PWCSE1695**

Class Section: **B**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Student Signature: _____

Submitted to:

Engr. Madiha Sher

Wednesday, June 16, 2021

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

OBJECTIVES:

This lab will enable students to:

- Code using Behavioral level modeling
- Implement FSM of BCD Counter

TASK:

Develop a Verilog model for the FSM of BCD Counter (0 – 9), which rolls over when it reaches 9 to 0.

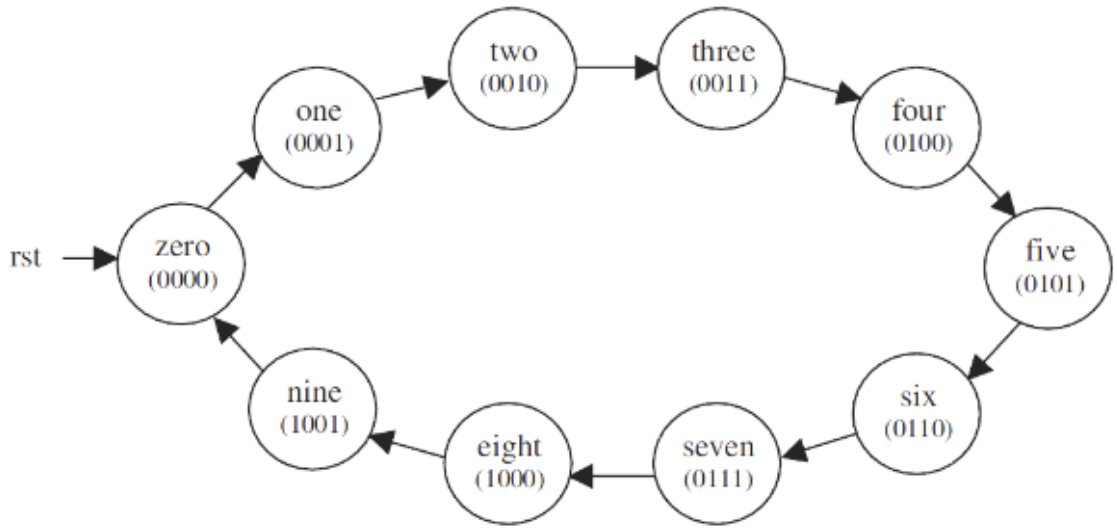


Fig: BCD Counter FSM

CODE:

```
module BCD_Counter(CLK, RST, OUT);
        input CLK, RST;                                 //two 1-bit inputs
        output [3:0] OUT;                               //4-bit output
        reg [3:0] state, next_state;

        parameter [3:0] S0 = 4'b0000, S1 = 4'b0001, S2 = 4'b0010,
                        S3 = 4'b0011, S4 = 4'b0100, S5 = 4'b0101,
                        S6 = 4'b0110, S7 = 4'b0111, S8 = 4'b1000,
                        S9 = 4'b1001;

        always @(posedge CLK)                           //Positive edge triggered
                state = next_state;

        always @(state or RST)
                if (RST)                                //if reset == 1
                        state = 4'b0000;
                else
                        case (state)
                                S0:     next_state = S1;
                                S1:     next_state = S2;
                                S2:     next_state = S3;
                                S3:     next_state = S4;
                                S4:     next_state = S5;
                                S5:     next_state = S6;
                                S6:     next_state = S7;
                                S7:     next_state = S8;
                                S8:     next_state = S9;
                                S9:     next_state = S0;
                        endcase
        assign OUT = state;                             //assign state values to the output
endmodule


```

TestBench:

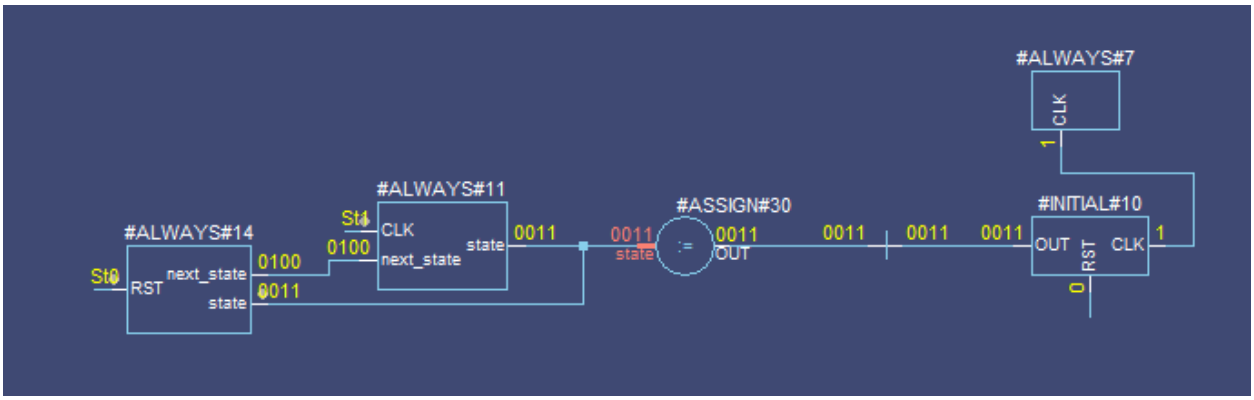
```
module test_BCD;
        reg CLK, RST;
        wire [3:0] OUT;

        BCD_Counter C1(CLK, RST, OUT);

        always
                #4 CLK = ~CLK;

        initial
        begin
                $display("RST | D_OUTPUT | B_OUTPUT");
                CLK = 0; RST = 0;
                $monitor("   %b    |          %d          |    %b",RST,OUT,OUT);
                #3 RST = 1;
                #5 RST = 0;

                #200 $finish;
        end
endmodule
```

OUTPUTS

Truth Table:

```
run
# RST | D_OUTPUT | B_OUTPUT
#   0 |        x |     xxxx
#   1 |        0 |     0000
#   0 |        0 |     0000
#   0 |        1 |     0001
#   0 |        2 |     0010
#   0 |        3 |     0011
#   0 |        4 |     0100
#   0 |        5 |     0101
#   0 |        6 |     0110
#   0 |        7 |     0111
#   0 |        8 |     1000
#   0 |        9 |     1001
#   0 |        0 |     0000
#   0 |        1 |     0001
#   0 |        2 |     0010
#   0 |        3 |     0011
#   0 |        4 |     0100
#   0 |        5 |     0101
#   0 |        6 |     0110
#   0 |        7 |     0111
#   0 |        8 |     1000
#   0 |        9 |     1001
#   0 |        0 |     0000
#   0 |        1 |     0001
#   0 |        2 |     0010
#   0 |        3 |     0011
```

Data Flow:



Wave Form: