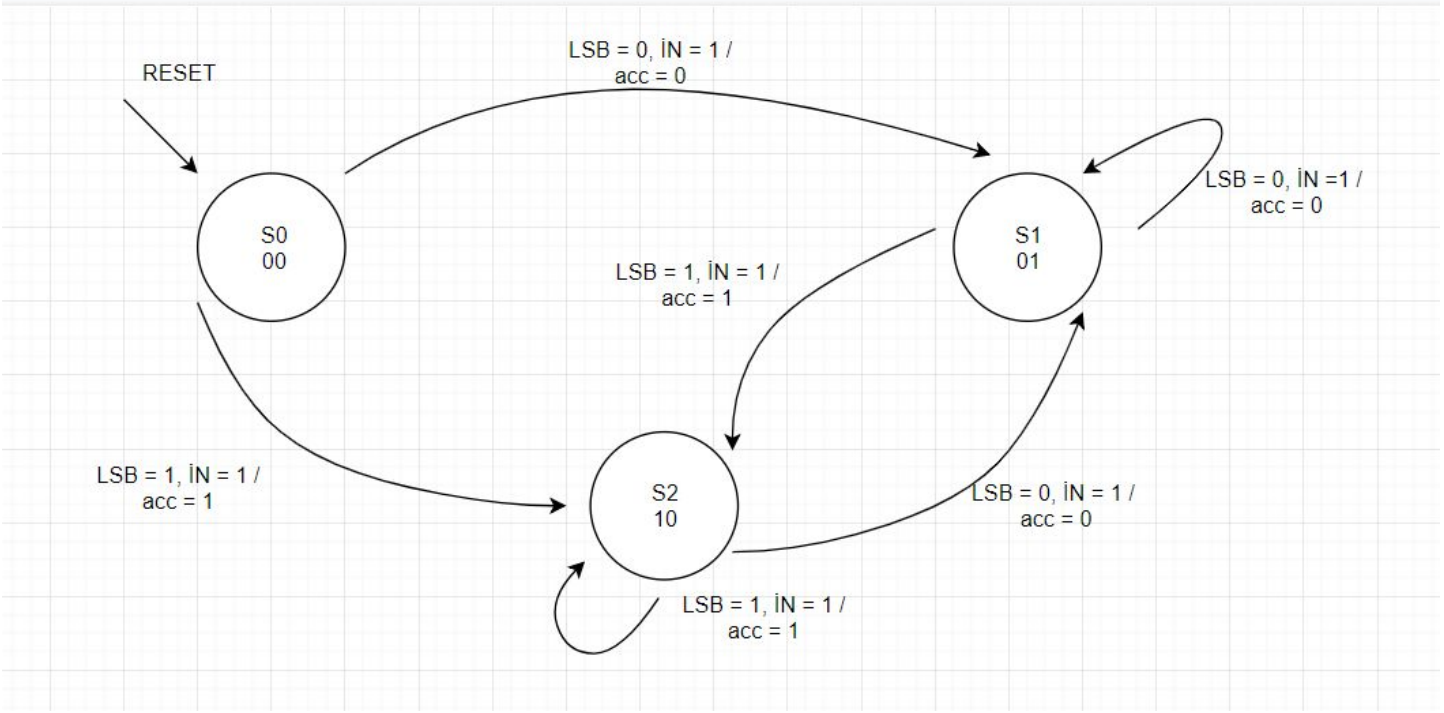
DIGITAL DESIGN – CS223
LAB 5
REPORT

İREM SEVEN
21704269

Date: 17.04.19
Trainer Pack No: 11

Fsm Diagram for Multiplier



00 = initial state
01 = no addition
10 = addition by acummulator

SystemVerilog Code

```
module multiplier
(
   input logic clk,
   input logic reset,
   input logic in,
   input logic [6:0] B,
   input logic [6:0] M,
   output logic [7:0] out
);

typedef enum logic [1:0] {S0,S1,S2} statetype;
   statetype state, nextstate;

logic [7:0] partialB;
logic [3:0] partialM;
assign partialM = M;
logic [5:0] count;


always_ff@ (posedge clk)                    //ACCUMULATOR
begin
        if (reset == 1'b1)
                out <= 8'b0000_0000;
        else if (in == 1 && partialM[0] == 1)
        begin
      partialB = partialM[0] * B;
      out = partialB + out;
   end
end

always_ff@ (posedge clk)                    //SHIFT REGISTER B 8 bit
begin
        if (reset == 1'b1)
        begin
   out = 8'b0000_0000;
   partialB <= out;
   end
        else if(in == 1'b1)
                partialB = partialB << 1;
end

always_ff@ ( posedge clk)                   //SHIFT REGISTER M 4 bit
begin
        if (reset == 1'b1)
                partialM <= 8'd0000_0000;
        else if(in == 1'b1)
                partialM = partialM >> 1;
end
```

```
always_ff@ (posedge clk)                    //STATE REGISTER
begin
        if (reset == 1'b1)
                state <= S0;
        else if (in == 1)
                state <= nextstate;
end


always@(posedge clk)                                //FSM COMB
begin
        case(state)
                S0:
                begin
                        if(in == 1 && partialM[0] == 1 )
                                nextstate <= S1;
                        else if(in == 1 && partialM[0] == 0)
                                nextstate <= S2;
                end

                S1:
                begin
                        if(in == 1 && partialM[0] == 1)
                                nextstate <= S1;
                        else if(in == 1 && partialM[0] == 0)
                                nextstate <= S2;
                end

                S2:
                begin
        if(in == 1 && partialM[0] == 1)
            nextstate <= S1;
        else if(in == 1 && partialM[0] == 0)
            nextstate <= S2;
                end
        endcase
end

// Output Logic
      always @ (posedge clk)
      begin
        if (state == S0)
          out = 8'b0000_0000;
        else if (state == S1)
          out = out;
        else if (state == S2)
          out = out;
      end

endmodule
```

TestBench Code

```
module Testbench();

    input logic clk,
    input logic reset,
    input logic in,
    input logic [6:0] B,
    input logic [6:0] M,
    output logic [7:0] out


always
begin
clk = 0; #5; clk = 1; #5;
end

multiplier dut(clk, reset, in, B , M, out);

initial begin
#20;
reset = 0;
in = 1;
M = 1001;
B  = 1111;
#20;
in = 0;
#20
in = 1;
M = 1100;
B  = 1101;
#20;
reset = 1;
end

endmodule
```