**Digital Lock (sequence Detector)**

**Lab no# 08**

****

Spring 2022

CSE-308L Digital Systems Design lab

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

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:

**Dr: Ma’am Madeha sheer**

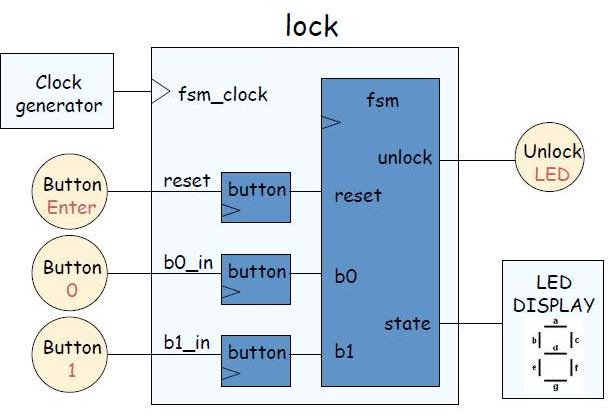
**June** 16, 2022

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Objective:**

* Build an electronic combination lock with reset button, two number buttons (0 and 1), and an unlock output.

**Block Diagram:** A combinational digital lock has three input buttons for Reset, entering a “0” and entering a “1” and output button UNLOCK and state which shows in which state the machine is currently in. The state output is connected to the seven segment display on the S6BOARD. The Module button in the below diagram is an abstraction of the synchronizer and level to pulse converter from the previous lab. The state transition diagram is given in the following figures.



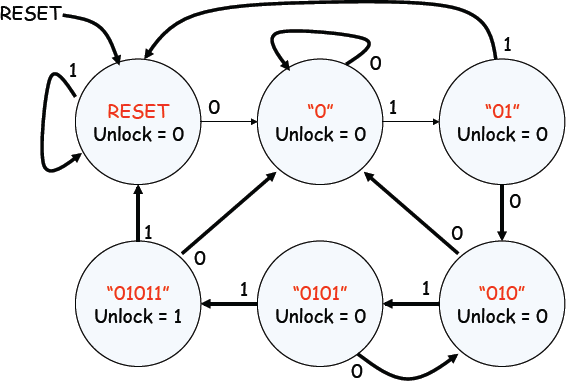
**Note:** As there are only two distinct digits in code I,e 0 and 1 so we use two buttons 0 and 1. Here we ignore the off state of both buttons. We can’t use one button to get a sequence like 0110 or 0010 etc by using both on and off state of button. By using one button we can get a sequence like 10101010… Or 01010101 ... [0 mean off button 1 mean on button].

**Task01: Design a sequence detector or digital lock that detect “01011”.**

**State Transition Diagram:**

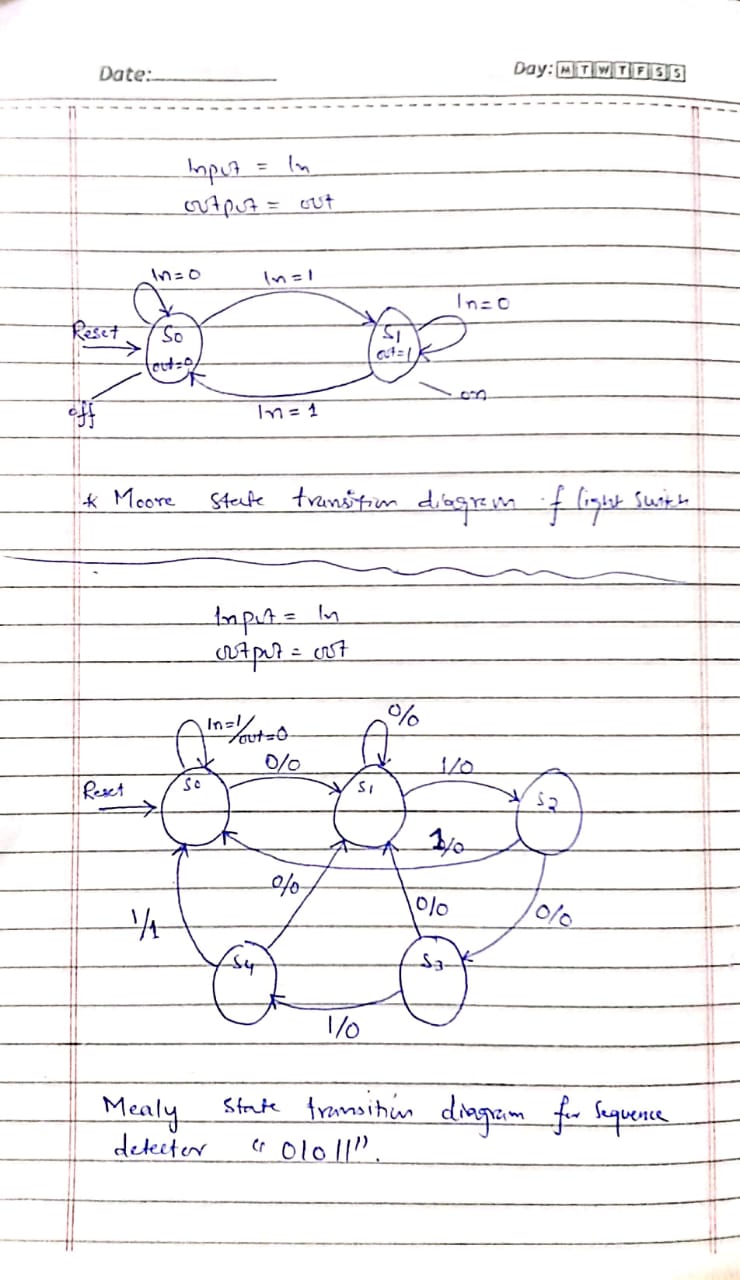
**Moore Machine (non-overlapping):**

**Output**=Unlock

**Inputs**=0 and 1

Mealy Machine (non-overlapping):

Note: 6 states are also possible.



**Source Code: (Two always block used)**

**Moore Machine:**

module digital\_Lock\_Moore(out,enable,seg7,zero,one,clk\_100Mhz,reset);  //sequence detector 01011

input zero,one,clk\_100Mhz,reset;

output [2:0]enable;

output reg out;

output [6:0]seg7;

wire L2P\_out0,L2P\_out1,clk\_5hz;

reg [2:0]state,next\_state;

parameter s0=3'd0,s1=3'd1,s2=3'd2,s3=3'd3,s4=3'd4,s5=3'd5;

clk\_divider cd(clk\_5hz,clk\_100Mhz,reset);

L2P\_Converter l2p0(L2P\_out0,zero,clk\_5hz,reset);

L2P\_Converter l2p1(L2P\_out1,one,clk\_5hz,reset);

seven\_seg ss(seg7,state);

assign enable=3'b110;

always@(posedge clk\_5hz,posedge reset)

begin

        if(reset) //if reset inital state= s0.

                  state=s0;

          else

                  state=next\_state;

end

always @(\*)      //Moore Machine

begin

      case(state)

        s0:

           begin

                          out=0;

                  if(L2P\_out0)

                          next\_state=s1;

                  else if(L2P\_out1)

                          next\_state=s0;

                  else

                          next\_state=state;

           end

      s1:

           begin

                          out=0;

                  if(L2P\_out0)

                          next\_state=s1;

                  else if(L2P\_out1)

                          next\_state=s2;

                  else

                          next\_state=state;

           end

      s2:

           begin

                          out=0;

                  if(L2P\_out0)

                          next\_state=s3;

                  else if(L2P\_out1)

                          next\_state=s0;

                  else

                          next\_state=state;

           end

      s3:

           begin

                          out=0;

                  if(L2P\_out0)

                          next\_state=s1;

                  else if(L2P\_out1)

                          next\_state=s4;

                  else

                          next\_state=state;

           end

      s4:

           begin

                          out=0;

                  if(L2P\_out0)

                          next\_state=s3;

                  else if(L2P\_out1)

                          next\_state=s5;

                  else

                          next\_state=state;

           end

        s5:

          begin

                          out=1;

                  if(L2P\_out0)

                          next\_state=s1;

                  else if(L2P\_out1)

                          next\_state=s0;

                  else

                          next\_state=state;

           end

          default: //if initally no reset default case will execute. if there is no default case then random case will be executing.

          begin

                 next\_state=s0;

                    out=0;

          end

        endcase

end

endmodule

module clk\_divider(clk\_5hz,clk\_100Mhz,reset);  //clock divider

input clk\_100Mhz,reset;

output reg clk\_5hz;

integer c;    //we can also delacre c as a reg.

always @(posedge clk\_100Mhz)

        if(reset)

        begin

                c=0;

                clk\_5hz=1;

        end

        else

        begin

                c=c+1'b1;

                if(c==10000000)

                begin

                        clk\_5hz=~clk\_5hz;

                        c=0;

                end

        end

endmodule

module D\_FF(Q,D,clk\_5hz,reset);  //D\_FF.

input D,clk\_5hz,reset;

output reg Q;

always @(posedge clk\_5hz)

       if(reset)

                 Q=0;

         else

                 Q=D;

endmodule

module synchronizer(syn\_out,in,clk\_5hz,reset);   //synchronizer

input in,clk\_5hz,reset;

output syn\_out;

wire FF1\_out;

//we can also design synchronizer without using D\_FFs.

D\_FF ff1(FF1\_out,in,clk\_5hz,reset);

D\_FF ff2(syn\_out,FF1\_out,clk\_5hz,reset);

endmodule

module L2P\_Converter(L2P\_out,in,clk\_5hz,reset); //level to pulse converter.

input in,clk\_5hz,reset;

output reg L2P\_out;

wire syn\_out;

synchronizer ss1(syn\_out,in,clk\_5hz,reset);

parameter s0=0,s1=1;  //posiible stats . signal level either 1 or 0.

reg state,next\_state;

always @(\*)           //star '\*' means this block is sensitive to all inputs used inside it.

begin

       case(state)    //stats diagram code of L2P.

       s0:

            if(syn\_out)

                begin

                     L2P\_out=1;

                      next\_state=s1;

                end

                else

                begin

                     L2P\_out=0;

                      next\_state=s0;

               end

       s1:

            if(syn\_out)

                begin

                     L2P\_out=0;

                      next\_state=s1;

                end

                else

                begin

                     L2P\_out=0;

                      next\_state=s0;

               end

      endcase

end

always @(posedge clk\_5hz)  //reset block

begin

       if(reset)

               state=s0;

       else

                 state=next\_state;

end

endmodule

module seven\_seg(seg7,state);   //7\_segment display

input [2:0]state;

output [6:0] seg7;     // as we have 0 to 4 states only. so only these five states will be used.

assign seg7=(state==3'b000)? 7'b1000000:

           (state==3'b001)? 7'b1111001:

           (state==3'b010)? 7'b0100100:

           (state==3'b011)? 7'b0110000:

           (state==3'b100)? 7'b0011001:

           (state==3'b101)? 7'b0010010:

           (state==3'b110)? 7'b0000010:

           (state==3'b111)? 7'b1111000:7'b1111111;

endmodule

**Mealy Code:**

always@(posedge clk\_5hz,posedge reset)

begin

        if(reset)  //if reset inital state= s0.

                  state=s0;

          else

                  state=next\_state;

end

always @(\*)               //Mealy Machine...

begin

      case(state)

        s0:

            if(L2P\_out0)

            begin

                 out=0;

                 next\_state=s1;

            end

            else

            if(L2P\_out1)

            begin

                 out=0;

                  next\_state=s0;

            end

            else

            begin

                 out=out;

                  next\_state=state;

            end

      s1:

           if(L2P\_out0)

            begin

                 out=0;

                 next\_state=s1;

            end

            else

            if(L2P\_out1)

            begin

                 out=0;

                  next\_state=s2;

            end

            else

            begin

                 out=out;

                  next\_state=state;

            end

      s2:

           if(L2P\_out0)

            begin

                 out=0;

                 next\_state=s3;

            end

            else

            if(L2P\_out1)

            begin

                 out=0;

                  next\_state=s0;

            end

            else

            begin

                 out=out;

                  next\_state=state;

            end

      s3:

           if(L2P\_out0)

            begin

                 out=0;

                 next\_state=s1;

            end

            else

            if(L2P\_out1)

            begin

                 out=0;

                  next\_state=s4;

            end

            else

            begin

                 out=out;

                  next\_state=state;

            end

      s4:

           if(L2P\_out0)

            begin

                 out=0;

                 next\_state=s1;

            end

            else

            if(L2P\_out1)

            begin

                 out=1;

                  next\_state=s0;

            end

            else

            begin

                 out=out;

                  next\_state=state;

            end

          default:    //if initally no reset default case will execute. if there is no default case then random case will be executing.

          begin

                 next\_state=s0;

                    out=0;

          end

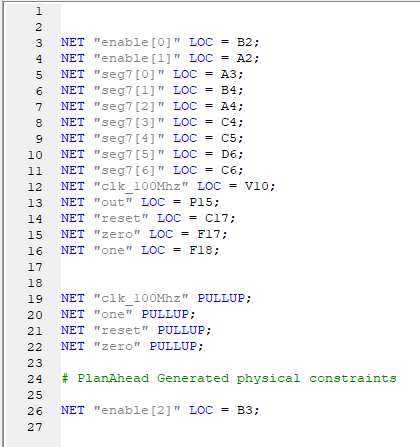
        endcase

end

endmodule

The Rest part of code is same as Moore.

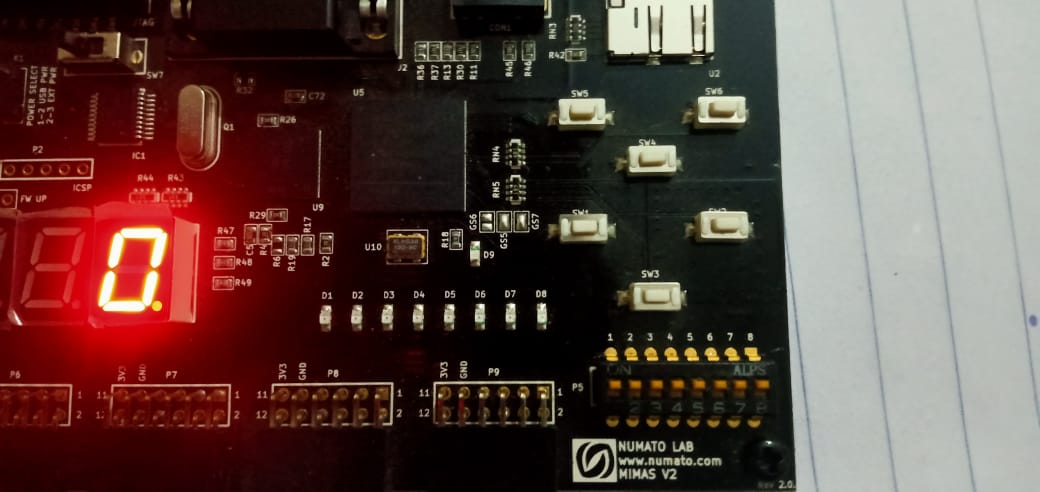
**UCF File:**



**Output: (Moore output)**

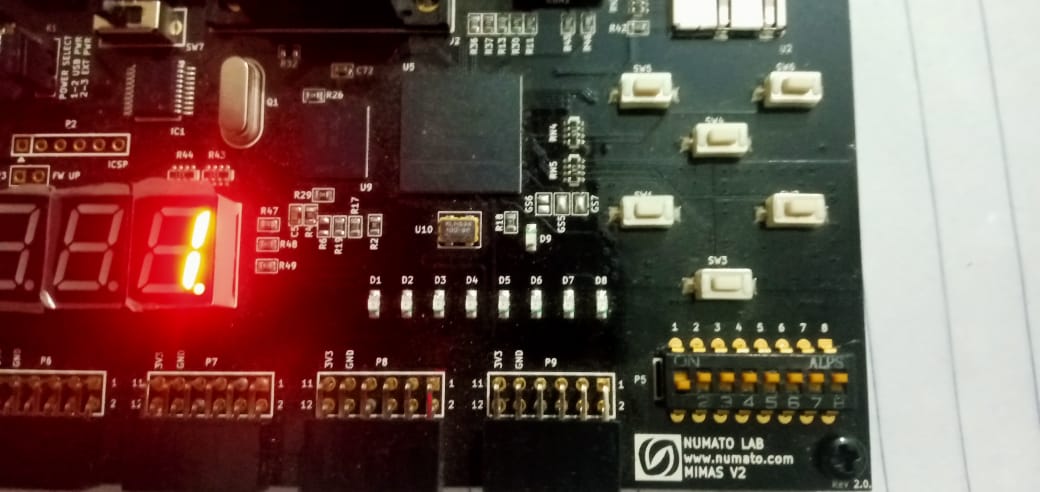
* The input Pins are Dip switch no 1 for 0 input and Dip switch no 2 for 1 input and reset switch is dip switch no 8.
* States are s0=0, s1=1, s2=2, s3=3, s4=4, s5=5.

Initial state=s0; out=0; input= no input.



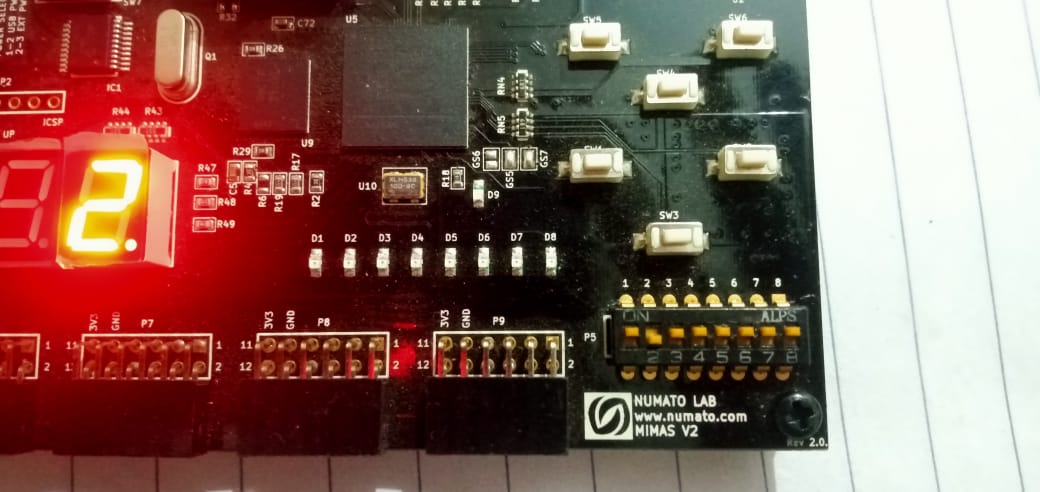
In=0; state=s1; out=0;

(Initially it was in s0 when first correct input 1 entered it goes to s1 state).



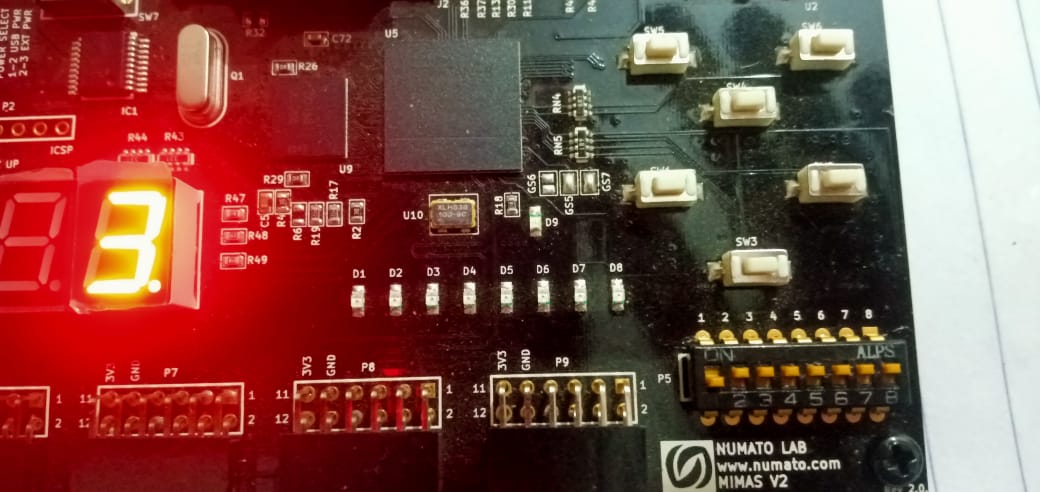
In=1; state=s2; out=0;

(First it was in s1 when next correct input 1 entered it goes to s2 state).



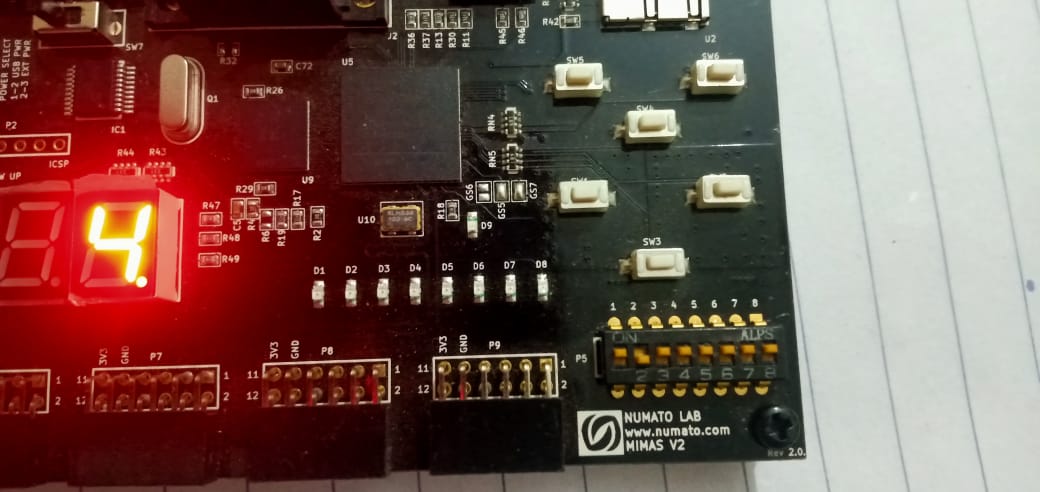
In=0; state=s3; out=0;

(First it was in s2 when next correct input 0 entered it goes to s3 state).



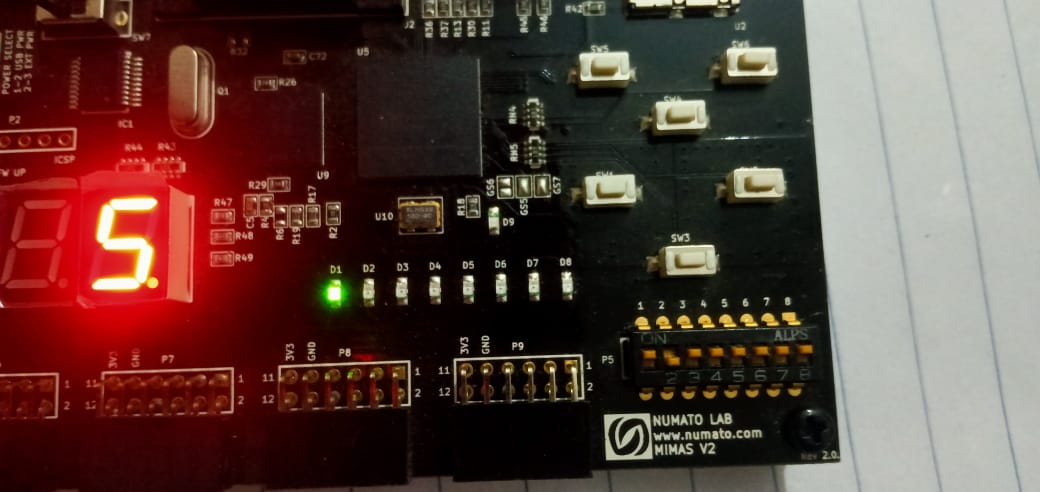
In=1; state=s4; out=0;

(First it was in s3 when next correct input 1 entered it goes to s4 state).



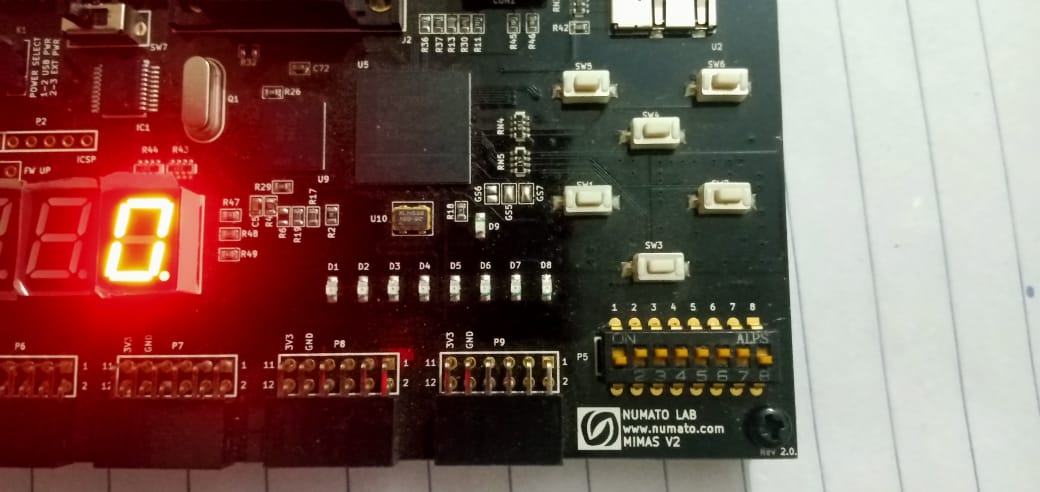
In=1; state=s5; out=1;

(First it was in s4 when next correct input 0 entered it goes to s5 state).



**Reset state:**

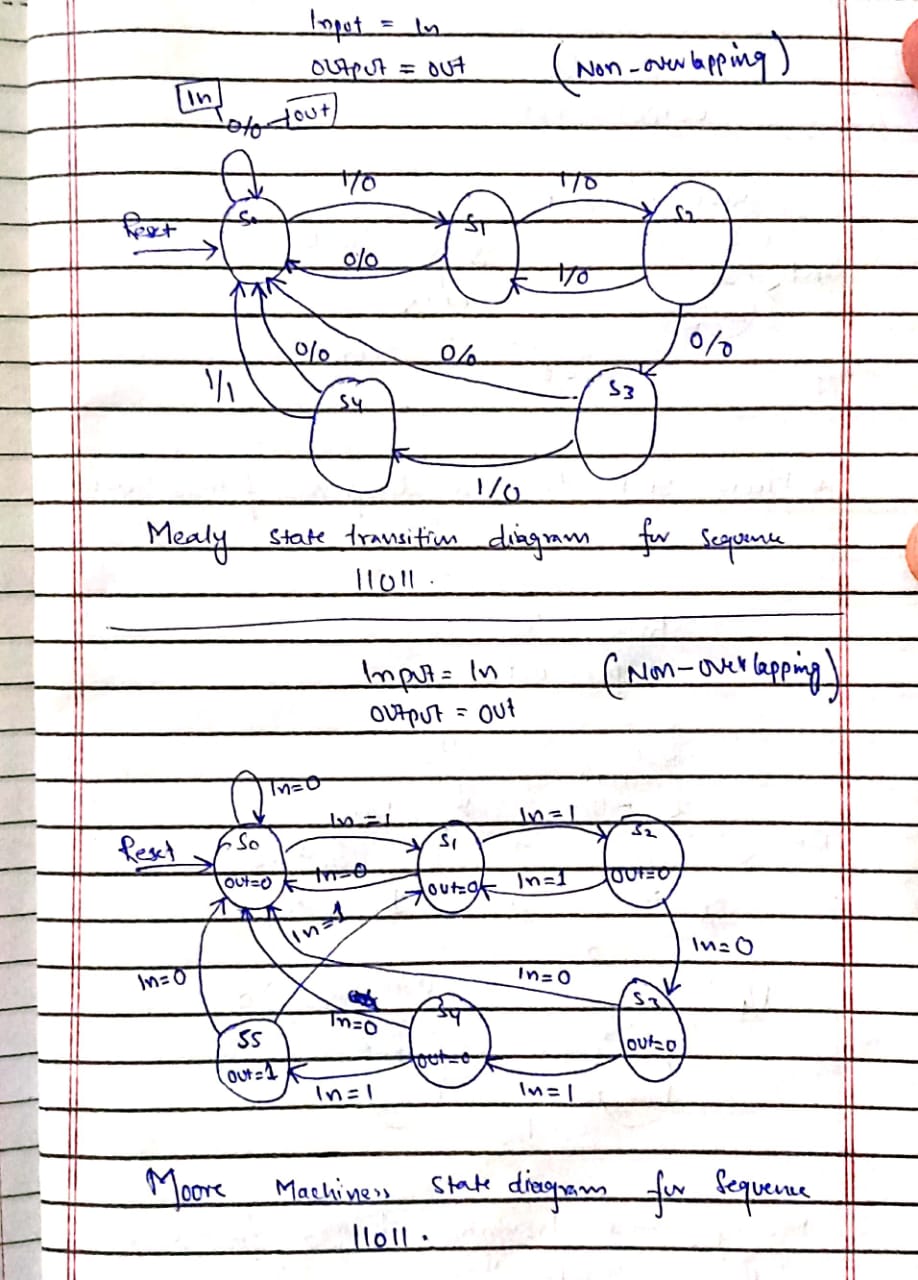
Even input is 0 but output is zero and state is s0.



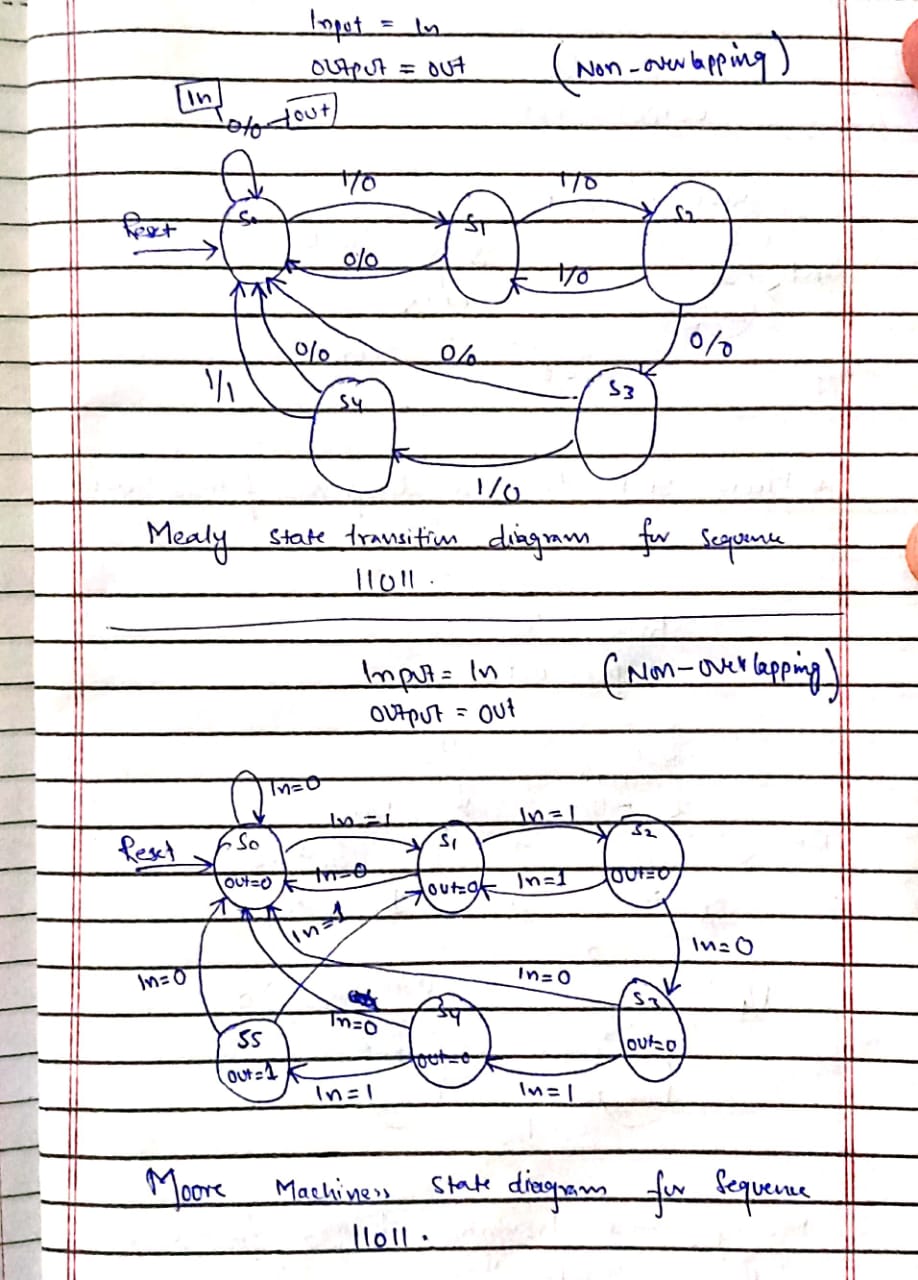
**Task 02: Change the functionality of the lock such that it unlocks on the sequence of 11011.**

**State Transition Diagram:**

**Moore Machine:**



**Mealy Machine:**



**Source Code: (one always block used)**

**Moore Machine:**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*sequence detector 11011\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

module digital\_lock\_Moore(seg7,out,zero,one,clk\_100Mhz,reset,enable);

input zero,one,clk\_100Mhz,reset;

output [2:0]enable;

output reg out;

output [7:0]seg7;

wire syn\_out1,syn\_out2,L2P\_out0,L2P\_out1,clk\_1hz;

reg [2:0]state;

parameter s0=3'd0,s1=3'd1,s2=3'd2,s3=3'd3,s4=3'd4,s5=3'd5;

//if i use s0=0 and s1=1 etc. then it give warning becuase these decimal are by default 32 bits and

// state register is 3 bits the warning will be "32-bits truncated to 3  bits".

clk\_divider cd(clk\_1hz,clk\_100Mhz,reset);

synchronizer syn1(syn\_out1,zero,clk\_1hz,reset);

synchronizer syn2(syn\_out2,one,clk\_1hz,reset);

L2P\_Converter l2p1(L2P\_out0,syn\_out1,clk\_1hz,reset);

L2P\_Converter l2p2(L2P\_out1,syn\_out2,clk\_1hz,reset);

seven\_seg ss(seg7,state);

assign enable=3'b110;

always@(posedge clk\_1hz,posedge reset)   //Moore Machine

begin

     if(reset)

      begin

               state=s0;

                 out=0;

      end

      else

     case(state)

                s0:

                  begin

                         out=0;

                         if(L2P\_out0)

                                    state=s0;

                         else if(L2P\_out1)

                                    state=s1;

                         else

                                    state=state;

                  end

                s1:

                   begin

                         out=0;

                         if(L2P\_out0)

                                    state=s0;

                         else if(L2P\_out1)

                                    state=s2;

                         else

                                    state=state;

                  end

                s2:

                  begin

                         out=0;

                         if(L2P\_out0)

                                    state=s3;

                         else if(L2P\_out1)

                                    state=s1;

                         else

                                    state=state;

                  end

                s3:

                 begin

                         out=0;

                         if(L2P\_out0)

                                    state=s0;

                         else if(L2P\_out1)

                                    state=s4;

                         else

                                    state=state;

                  end

                s4:

                  begin

                         out=0;

                         if(L2P\_out0)

                                    state=s0;

                         else if(L2P\_out1)

                                    state=s5;

                         else

                                    state=state;

                  end

                s5:

                 begin

                         out=1;

                         if(L2P\_out0)

                                    state=s0;

                         else if(L2P\_out1)

                                    state=s1;

                         else

                                    state=state;

                  end

                 default:   //if there is no reset then default state will execute . if there is no default and no reset then any random state will execute.

                  begin

                       state=s0;

                         out=0;

                  end

        endcase

end

endmodule

module clk\_divider(clk\_1hz,clk\_100Mhz,reset);  //clock divider

input clk\_100Mhz,reset;

output reg clk\_1hz;

integer c;    //we can also delacre c as a reg.

always @(posedge clk\_100Mhz)

        if(reset)

        begin

                c=0;

                clk\_1hz=1;

        end

        else

        begin

                c=c+1'b1;

                if(c==50000000)

                begin

                        clk\_1hz=~clk\_1hz;

                        c=0;

                end

        end

endmodule

module D\_FF(Q,D,clk\_1hz,reset);  //D\_FF.

input D,clk\_1hz,reset;

output reg Q;

always @(posedge clk\_1hz)

       if(reset)

                 Q=0;

         else

                 Q=D;

endmodule

module synchronizer(syn\_out,in,clk\_1hz,reset);   //synchronizer

input in,clk\_1hz,reset;

output syn\_out;

wire FF1\_out;

//we can also design synchronizer without using D\_FFs.

D\_FF ff1(FF1\_out,in,clk\_1hz,reset);

D\_FF ff2(syn\_out,FF1\_out,clk\_1hz,reset);

endmodule

module L2P\_Converter(L2P\_out,syn\_out,clk\_1hz,reset); //level to pulse converter.

input syn\_out,clk\_1hz,reset;

output reg L2P\_out;

parameter s0=0,s1=1;  //posiible stats . signal level either 1 or 0.

reg state,next\_state; //by default reg size is 32 bits also by default integer size taken as 32 bit so s0 and s1 can store in state and next\_state without giving any warning.

always @(\*)           //star '\*' means this block is sensitive to all inputs used inside it.

begin

       case(state)    //stats diagram code of L2P.

       s0:            //this is mealy machine.

            if(syn\_out)

                begin

                     L2P\_out=1;

                      next\_state=s1;

                end

                else

                begin

                     L2P\_out=0;

                      next\_state=s0;

               end

       s1:

            if(syn\_out)

                begin

                     L2P\_out=0;

                      next\_state=s1;

                end

                else

                begin

                     L2P\_out=0;

                      next\_state=s0;

               end

      endcase

end

//we can also use single always block then no need of next\_register. as i used one alwyas block in main module.

always @(posedge clk\_1hz)  //reset block

begin

       if(reset)

               state=s0;

       else

                 state=next\_state;

end

endmodule

module seven\_seg(seg7,state);   //7\_segment display

input [2:0]state;

output [7:0] seg7;     // as we have 0 to 5 states only. so only these six states will be used.

assign seg7=(state==3'b000)? 8'b01000000:  //the last 8th bit for dit.

           (state==3'b001)? 8'b01111001:

           (state==3'b010)? 8'b00100100:

           (state==3'b011)? 8'b00110000:

           (state==3'b100)? 8'b00011001:

           (state==3'b101)? 8'b00010010:

           (state==3'b110)? 8'b00000010:

           (state==3'b111)? 8'b01111000:8'b01111111;

endmodule

**Mealy Machine:**

always@(posedge clk\_1hz,posedge reset)   //Mealy Machine

begin

     if(reset)    //initial state...

      begin

               state=s0;

                 out=0;

      end

      else

     case(state)

                s0:

                  if(L2P\_out0)

                  begin

                         out=0;

                         state=s0;  //both input not possible at  a time also we don't need.even if we entered both input at a time it will considered L2p\_out1 input.

                  end

                  else if(L2P\_out1)

                  begin

                         out=0;

                         state=s1;

                  end

                s1:

                  if(L2P\_out0)

                  begin

                         out=0;

                         state=s0;

                  end

                  else if(L2P\_out1)

                  begin

                         out=0;

                         state=s2;

                  end

                s2:

                  if(L2P\_out0)

                  begin

                         out=0;

                         state=s3;

                  end

                  else if(L2P\_out1)

                  begin

                         out=0;

                         state=s1;

                  end

                s3:

                 if(L2P\_out0)

                  begin

                         out=0;

                         state=s0;

                  end

                  else if(L2P\_out1)

                  begin

                         out=0;

                         state=s4;

                  end

                s4:

                  if(L2P\_out0)

                  begin

                         out=0;

                         state=s0;

                  end

                  else if(L2P\_out1)

                  begin

                         out=1;

                         state=s0;

                  end

                default:    //if initially not reset then default state execute. if there is no reset and no default state then random state execute.

                  begin

                       state=s0;

                         out=0;

                  end

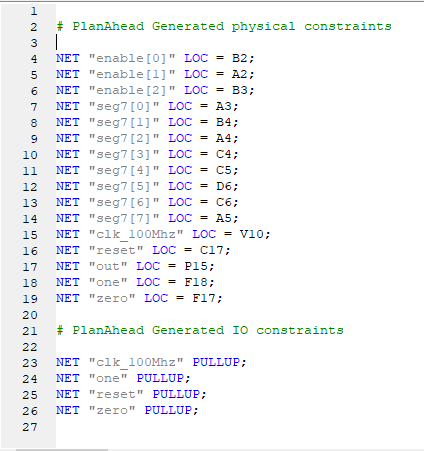
        endcase

end

endmodule

The rest part of code is same as Moore.

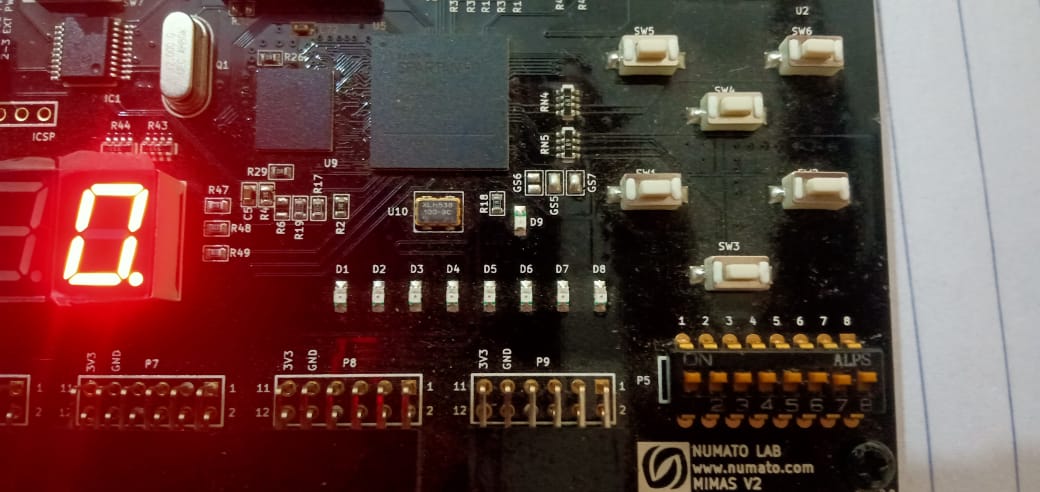
**UCF file:**



**Output: (Moore output)**

* The input Pins are Dip switch no 1 for 0 input and Dip switch no 2 for 1 input and reset switch is dip switch no 8.
* States are s0=0, s1=1, s2=2, s3=3, s4=4, s5=5.

Initial state=s0; out=0; input= no input.



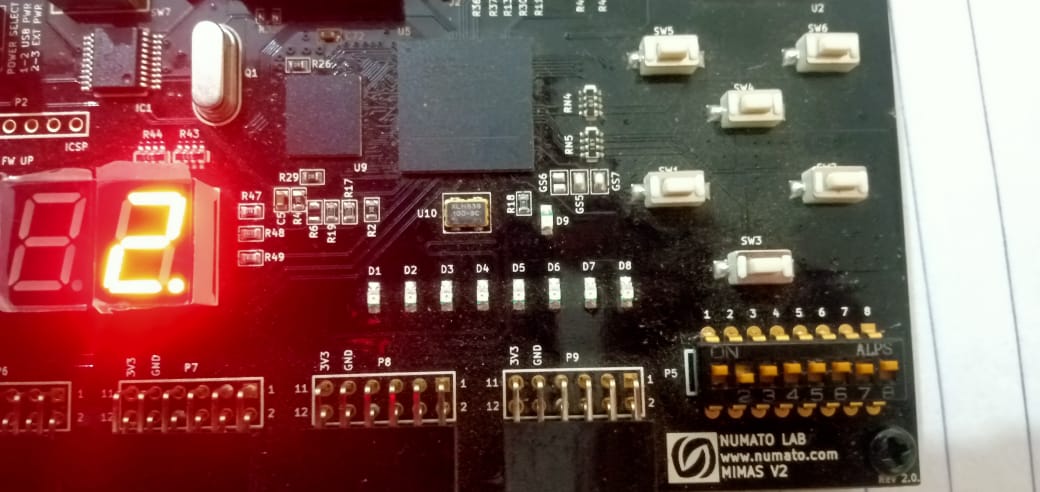
In=1; state=s1; out=0;

(Initially it was in s0 when first correct input 1 entered it goes to s1 state).



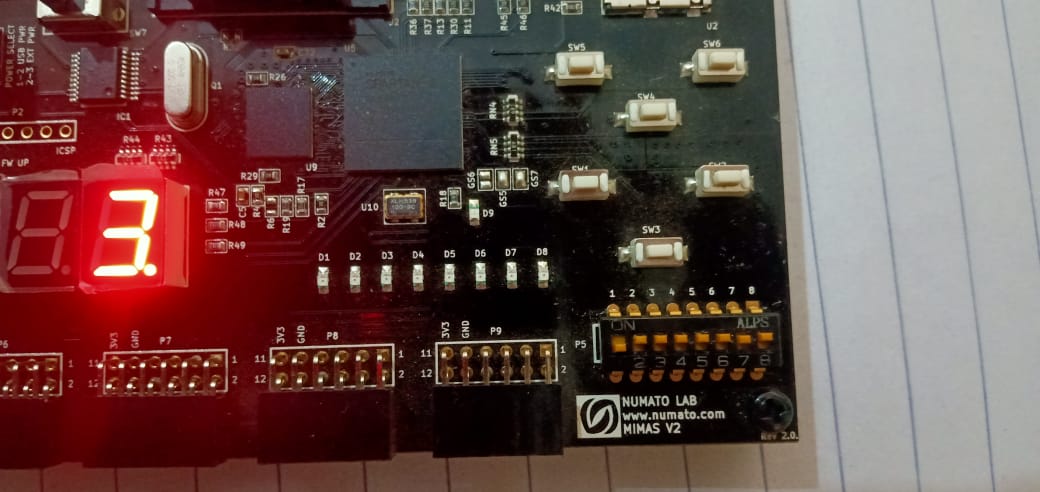
In=1; state=s2; out=0;

(First it was in s1 when next correct input 1 entered it goes to s2 state).



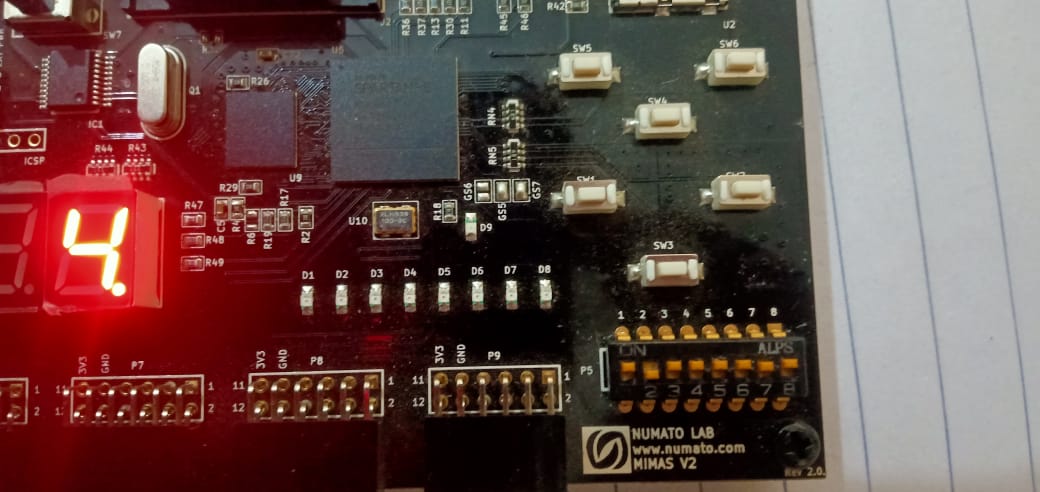
In=0; state=s3; out=0;

(First it was in s2 when next correct input 0 entered it goes to s3 state).



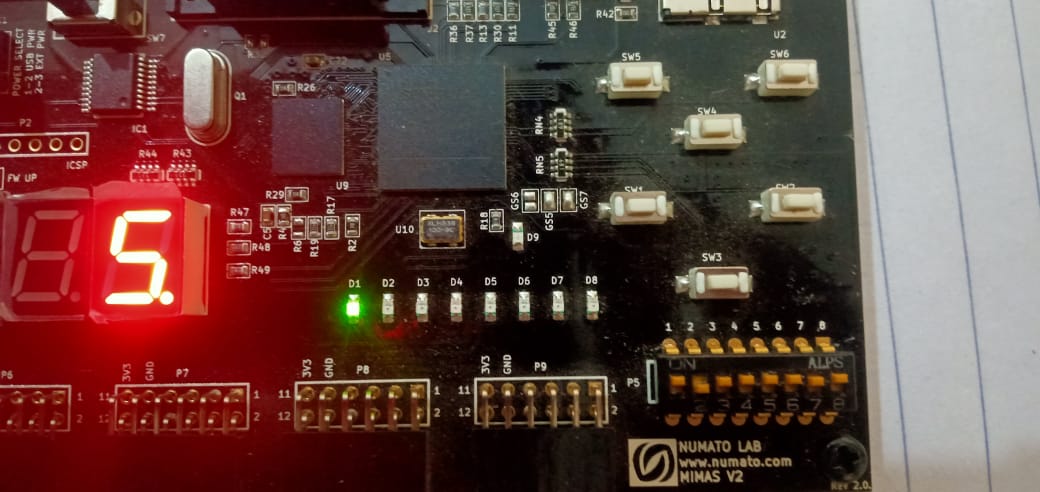
In=1; state=s4; out=0;

(First it was in s3 when next correct input 1 entered it goes to s4 state).



In=1; state=s5; out=1;

(First it was in s4 when next correct input 0 entered it goes to s5 state).



**Reset state:**

Even input is one but output is zero and state is s0.

