**Design of Light Switch**

**Lab no# 07**

****

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** 11, 2022

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Objective:**

* Build a light switch controller such that when the push button is pressed the light if “off” turns “on” and if “on” turns “off”.

**Diagram

Description automatically generatedBlock Diagram:**

* Connect the button input to a dip or push button on the S6 board and light output to LED.

**Synchronizer Circuit:**

These two registers are two D\_FFs.

Diagram

Description automatically generated

**Level to Pulse FSM:**

Given register is D\_FFs.

Diagram

Description automatically generated

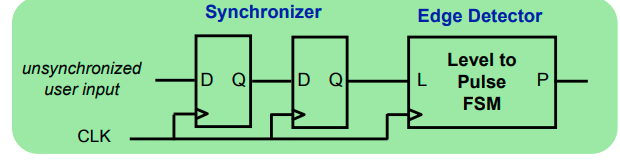
**Task 01:** Design a Level to Pulse Converter:

**Description:**

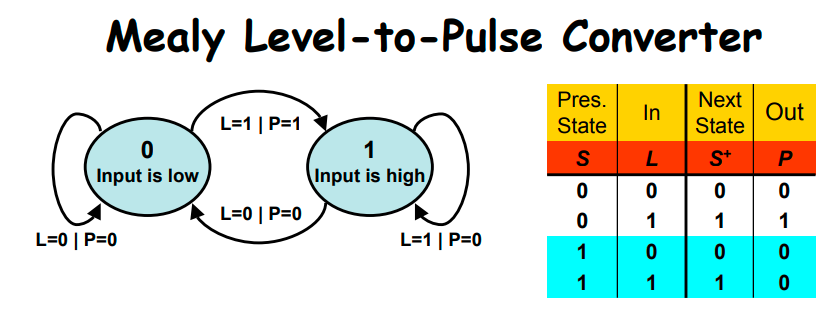
For level to pulse first we convert user input (button) to synchronized input by using two D\_FFs (we can also use one but two for accurate synchronization).

We can also synchronize user input by simple assigning it to other variable on clock edge in separate module.

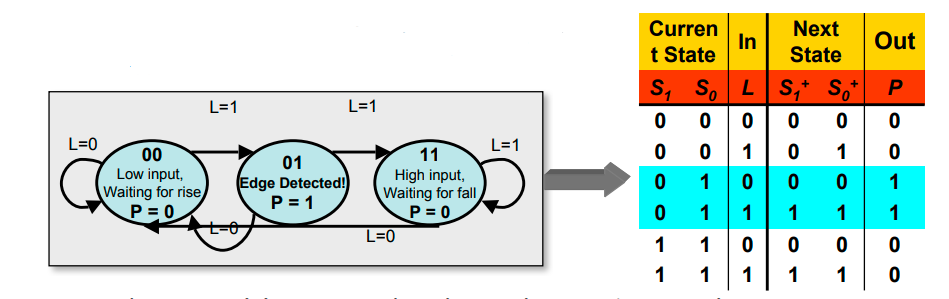
**Block Diagram:**



**State Transition Diagram of Level to Pulse converter:**



**In case of Moore machine,**



**Source Code:**

module L2P\_Converter(out,in,clk\_100Mhz,reset);

input in,clk\_100Mhz,reset;

output reg out;

wire clk\_1Mhz,syn\_out;

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

reg state,next\_state;

clk\_divider cd(clk\_1Mhz,clk\_100Mhz,reset);

synchronizer syn(syn\_out,in,clk\_1Mhz,reset);

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

begin

       case(state)    //stats diagram code

       s0:

            if(syn\_out)

                begin

                     out=1;

                      next\_state=s1;

                end

                else

                begin

                     out=0;

                      next\_state=s0;

               end

       s1:

            if(syn\_out)

                begin

                     out=0;

                      next\_state=s1;

                end

                else

                begin

                     out=0;

                      next\_state=s0;

               end

      endcase

end

//we can also remove given always block and add default case in case statement.

always @(posedge clk\_1Mhz)  //reset block

begin

       if(reset)

               state=s0;

       else

                 state=next\_state;

end

endmodule

module clk\_divider(clk\_1Mhz,clk\_100Mhz,reset);  //clock divider

input clk\_100Mhz,reset;

output reg clk\_1Mhz;

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

always @(posedge clk\_100Mhz)

        if(reset)

                  begin

                  c=0;

                  clk\_1Mhz=1;

                  end

                  else

                  begin

                  c=c+1'b1;

                  if(c==50000000)

                  begin

                                             clk\_1Mhz=~clk\_1Mhz;

c=0;

 end

 end

endmodule

module D\_FF(Q,D,clk\_1Mhz,reset);  //D\_FF

input D,clk\_1Mhz,reset;

output reg Q;

always @(posedge clk\_1Mhz)

       if(reset)

                 Q=0;

         else

                 Q=D;

endmodule

module synchronizer(syn\_out,in,clk\_1Mhz,reset);   //synchronizer

input in,clk\_1Mhz,reset;

output syn\_out;

wire FF1\_out;

D\_FF ff1(FF1\_out,in,clk\_1Mhz,reset);

D\_FF ff2(syn\_out,FF1\_out,clk\_1Mhz,reset);

endmodule

**Output:**

* Input is Dip switch no 1 and reset is dip switch no 8.
* States are s0=0 and s1=1.

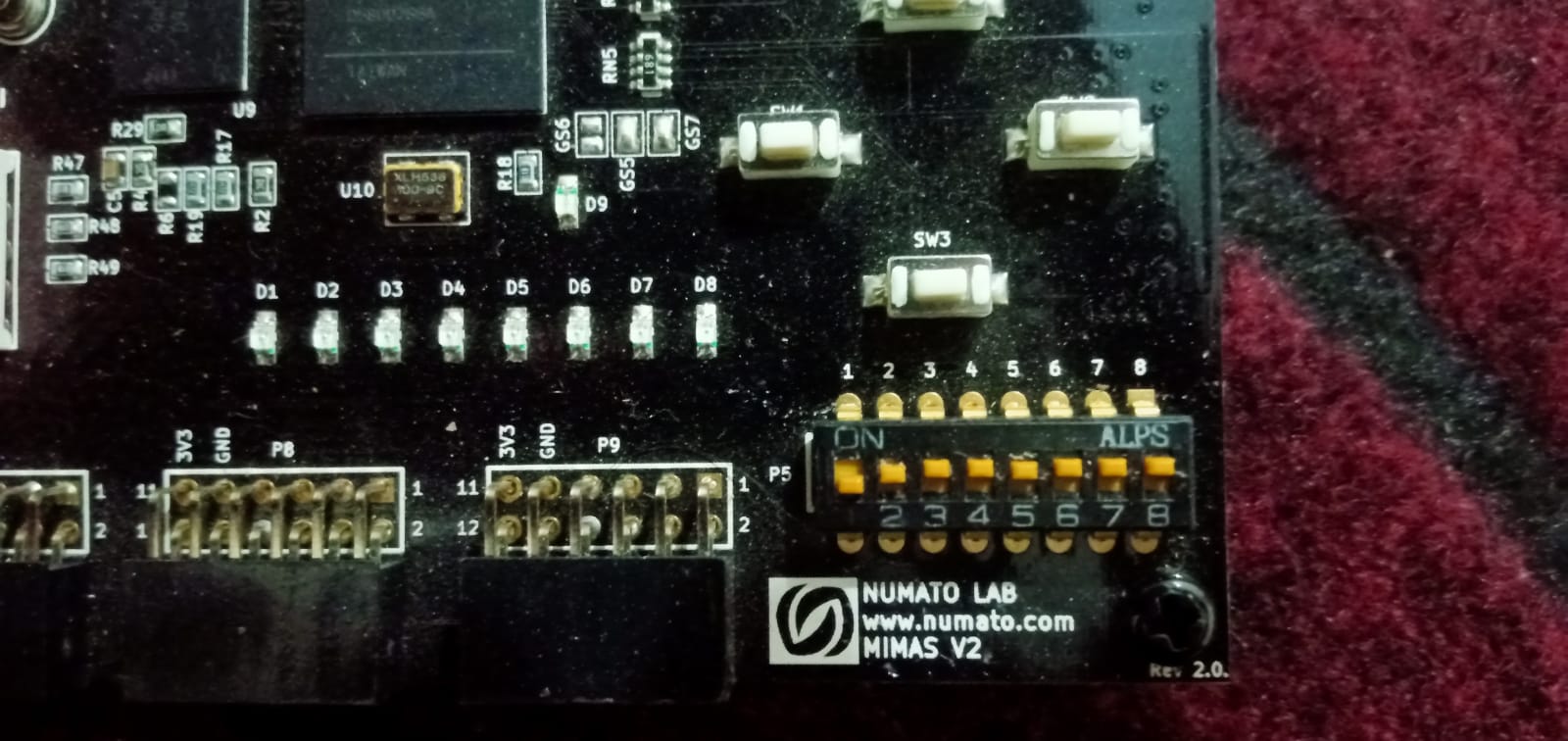
In=0; state=s0; out=0; next state=s0.

****

In=1; state=s0; out=1; next state=s1.

****

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

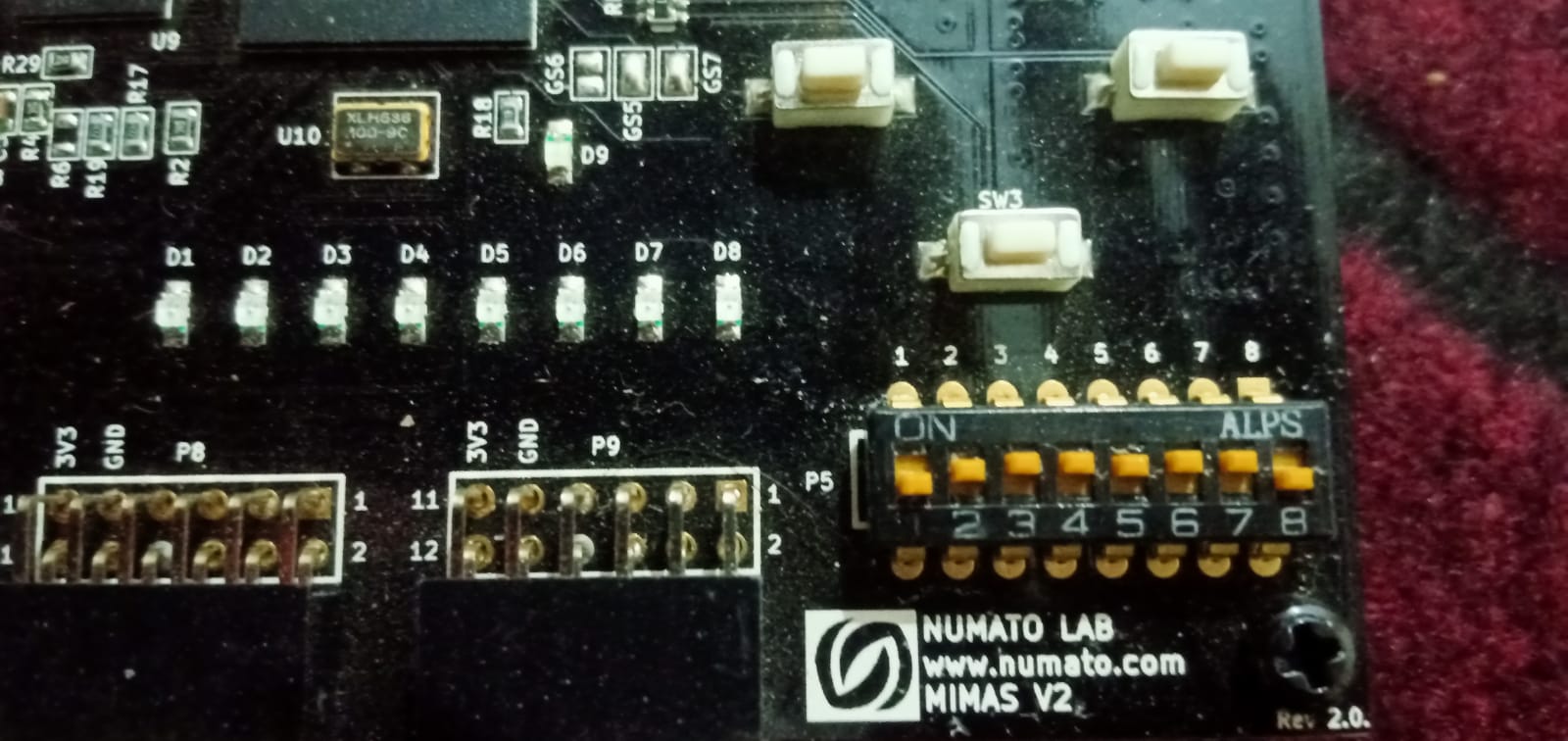
****

In=0; state=s1; out=0; next state=s0.

****

**Reset state:**

Even in=1 but out=0 and state=s0.



**Task 02:** Design a light switch.

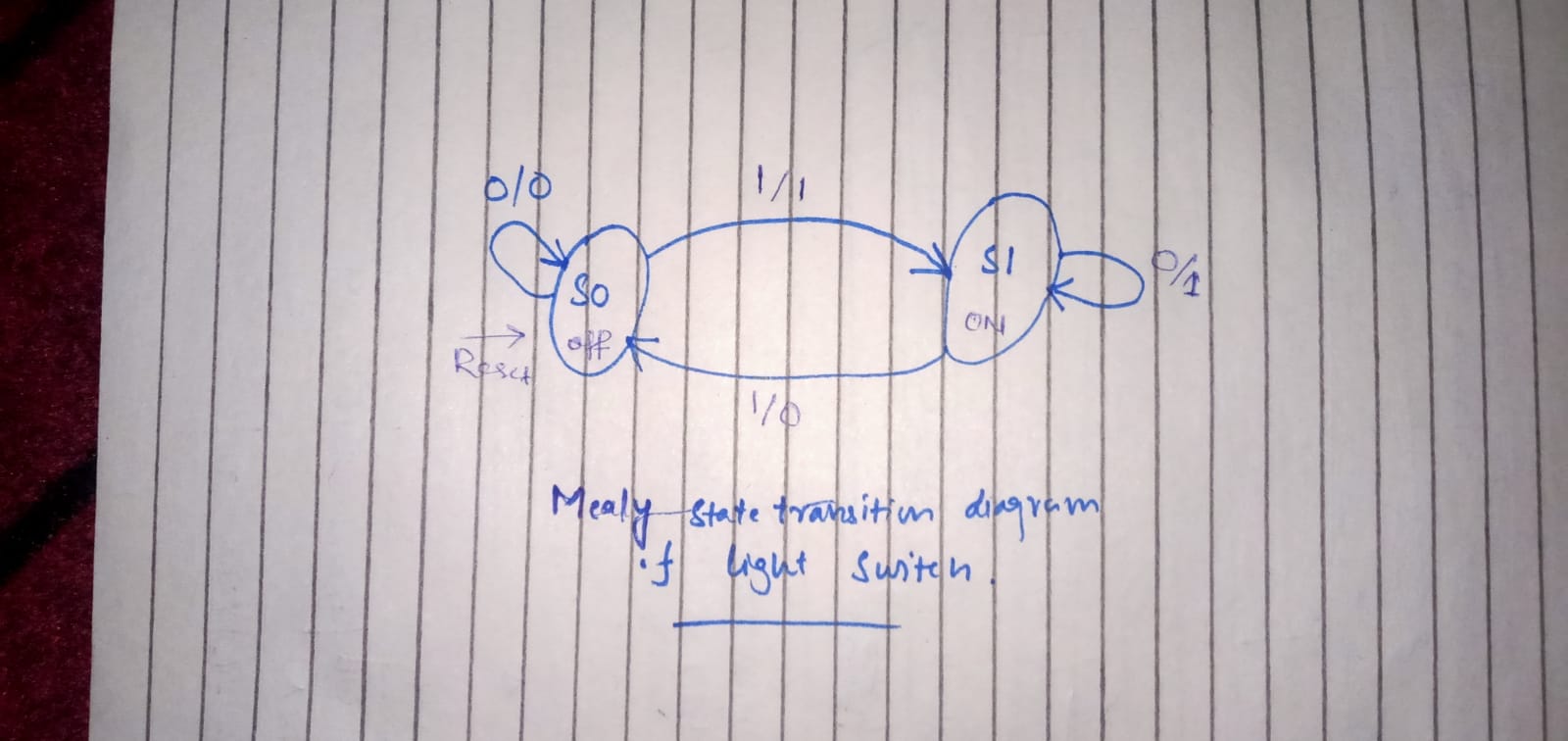
**Block Diagram:**

A screenshot of a computer

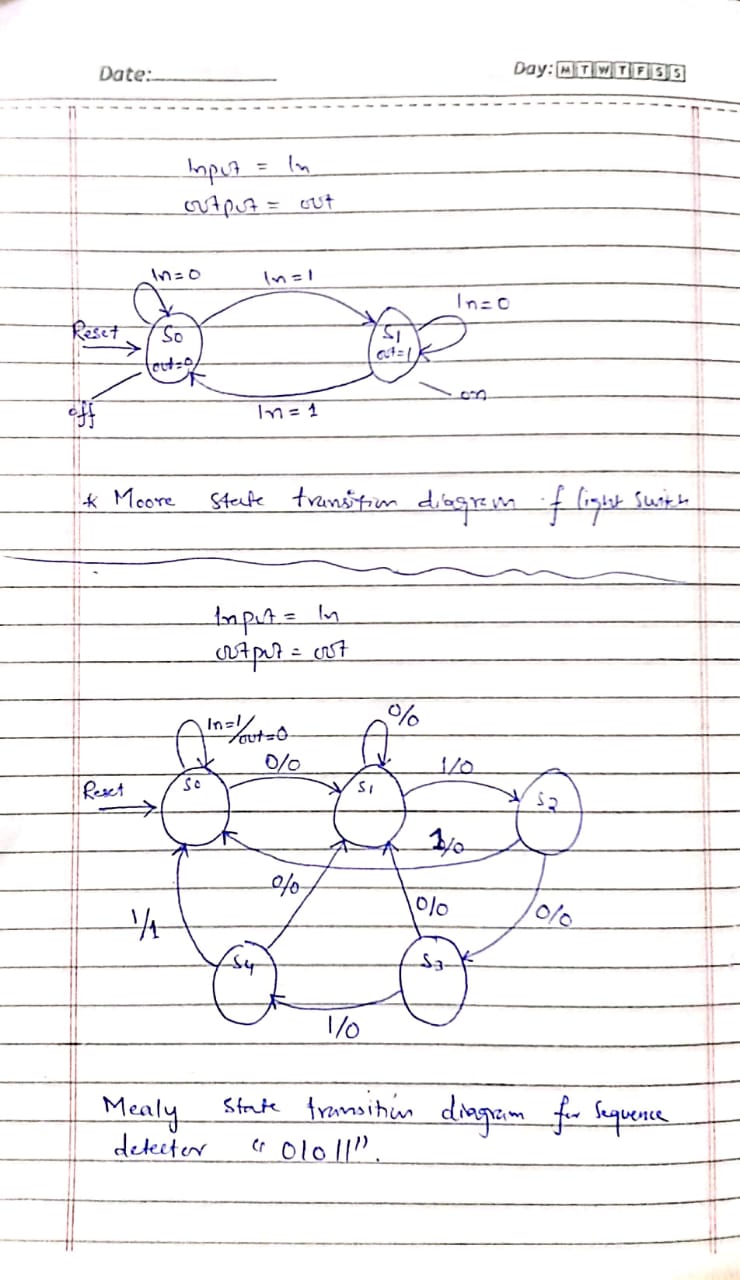
Description automatically generated with medium confidence

**States Transition Diagram:**

**Mealy Machine:**



**Moore Machine:**

****

**Source Code:**

module light\_switch(switch\_out,in,clk\_100Mhz,reset);  //light switch main module.

input in,clk\_100Mhz,reset;

output reg switch\_out;

wire clk\_1Mhz,syn\_out,L2P\_out;

reg state,next\_state;

parameter s0=0,s1=1;

clk\_divider cd(clk\_1Mhz,clk\_100Mhz,reset);

synchronizer syn(syn\_out,in,clk\_1Mhz,reset); //i can also instantiate synchronizer in L2p\_converter block.

L2P\_Converter l2p(L2P\_out,syn\_out,clk\_1Mhz,reset);

always@(\*)     //light switch state digram code

       case(state)

       s0:

          if(L2P\_out)

             begin

                   switch\_out=1;

                   next\_state=s1;

             end

           else

             begin

                      switch\_out=0;

                         next\_state=s0;

            end

        s1:

           if(L2P\_out)

            begin

                      switch\_out=0;

                        next\_state=s0;

            end

            else

            begin

                      switch\_out=1;

                         next\_state=s1;

         end

     endcase

//we can also remove given always block and add default case in case statement.

always@(posedge clk\_1Mhz)

      if(reset)

              state=s0;

      else

              state=next\_state;

endmodule

module clk\_divider(clk\_1Mhz,clk\_100Mhz,reset);  //clock divider

input clk\_100Mhz,reset;

output reg clk\_1Mhz;

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

always @(posedge clk\_100Mhz)

            if(reset)

            begin

                    c=0;

                    clk\_1Mhz=1;

            end

            else

            begin

                    c=c+1'b1;

                    if(c==20000000) //here i reduced value of c so the output clock is not 1\_Mhz.

                    begin

                            clk\_1Mhz=~clk\_1Mhz;

                            c=0;

                    end

            end

endmodule

//I have designed synchronizer without using Flip flops. but funcationality same.

module synchronizer(syn\_out,in,clk\_1Mhz,reset);   //synchronizer

input in,clk\_1Mhz,reset;

output reg syn\_out;

reg FF1\_out;

always @(posedge clk\_1Mhz)

       if(reset)

         syn\_out=0;

         else

         begin

             FF1\_out=in;  //here also two FF will be generated. but coding method is different.

              syn\_out=FF1\_out;

         end

endmodule

module L2P\_Converter(L2P\_out,syn\_out,clk\_1Mhz,reset); //level to pulse converter.

input syn\_out,clk\_1Mhz,reset;

output reg L2P\_out;

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\_1Mhz)  //reset block

begin

       if(reset)

               state=s0;

       else

                 state=next\_state;

end

endmodule

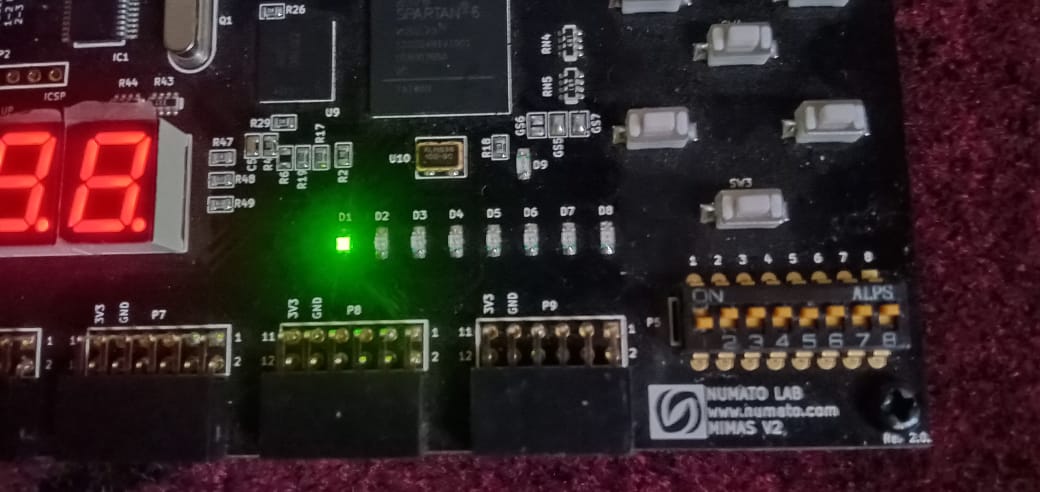
**Output:**

* The input Pin is Dip switch no 1 and reset switch is dip switch no 8.
* States are s0=0 and s1=1.

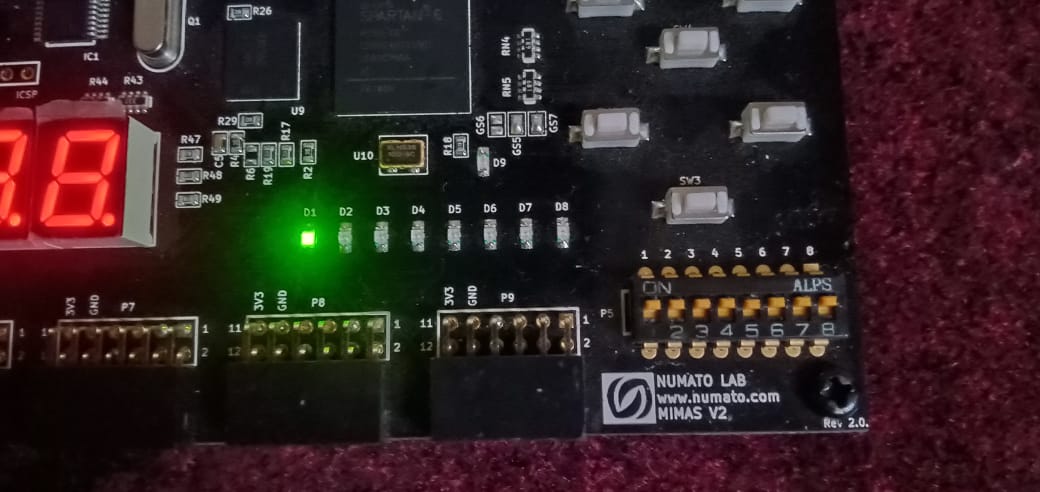
In=0; state=s0; out=0; next state=s0.



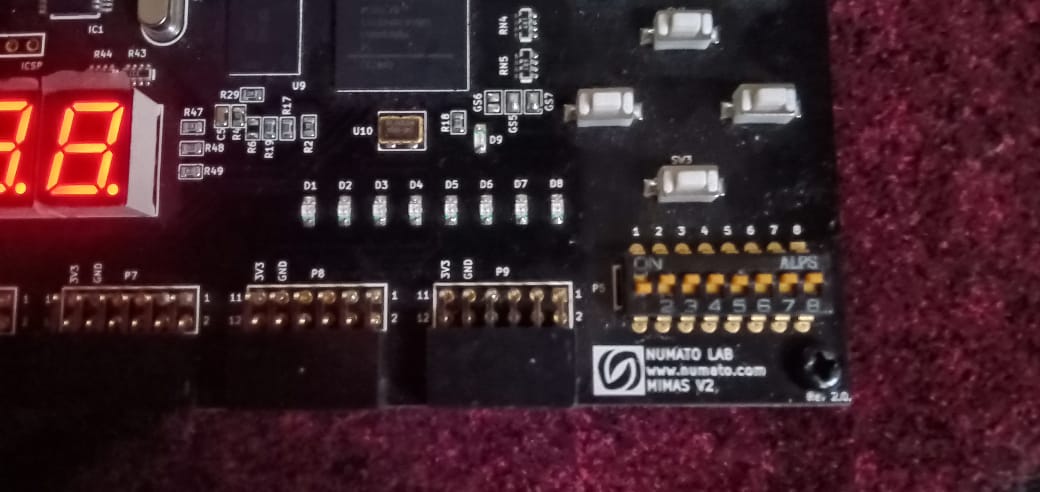
In=1; state=s0; out=1; next state=s1.



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



In=1; state=s1; out=0; next state=s0.



**Reset state:**

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

