**3 digits BCD Counter on Multiplexed Seven Segment Display**

**Lab no# 06**

****

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**

**May** 27, 2022

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**Objective:**

* Learn to use time multiplexed 3 digit Seven Segment display

**Block Diagram:**

****

**Lab Task01:** Implement a BCD counter that runs from 0 to 9 and shows each BCD digit on the seven segment display.

**Source Code:**

module BCD\_Counter(seg,en,reset,clk);  //BCD 1 digit counter

input reset,clk;

output [6:0]seg;

reg [3:0] out;

wire clk\_1MHz;

output [2:0] en;

assign en=3'b110;

 //we can also on remaining 2 LEDs. same digit will display on other two LEDsdue to common inputs.

clk\_divider cd(clk\_1MHz,clk,reset);

seven\_seg\_Dec s1(seg,out);

always @(posedge clk\_1MHz)

       if(reset)

            out=4'd0;

         else

         begin

             out=out+1'b1;

             if(out==4'd10)

                out=4'd0;

         end

endmodule

module clk\_divider(clk\_1MHz,clk\_100MHz,reset);  //clock divider module

input clk\_100MHz;

output clk\_1MHz;

input reset;

reg[26:0]c;

reg clk\_1MHz;

always @(posedge clk\_100MHz)

begin

     if(reset)

      begin

      c=0;

      clk\_1MHz=1;

      end

      else

      begin

      c=c+1'b1;

      if(c==10000000)

      begin

      clk\_1MHz=~clk\_1MHz;

      c=0;

      end

     end

end

endmodule

module  seven\_seg\_Dec(seg,in);  //seven segments display module

input [3:0]in;

output [6:0]seg;

assign seg=(in==4'b0000)? 7'b1000000:

           (in==4'b0001)? 7'b1111001:

              (in==4'b0010)? 7'b0100100:

              (in==4'b0011)? 7'b0110000:

              (in==4'b0100)? 7'b0011001:

              (in==4'b0101)? 7'b0010010:

              (in==4'b0110)? 7'b0000010:

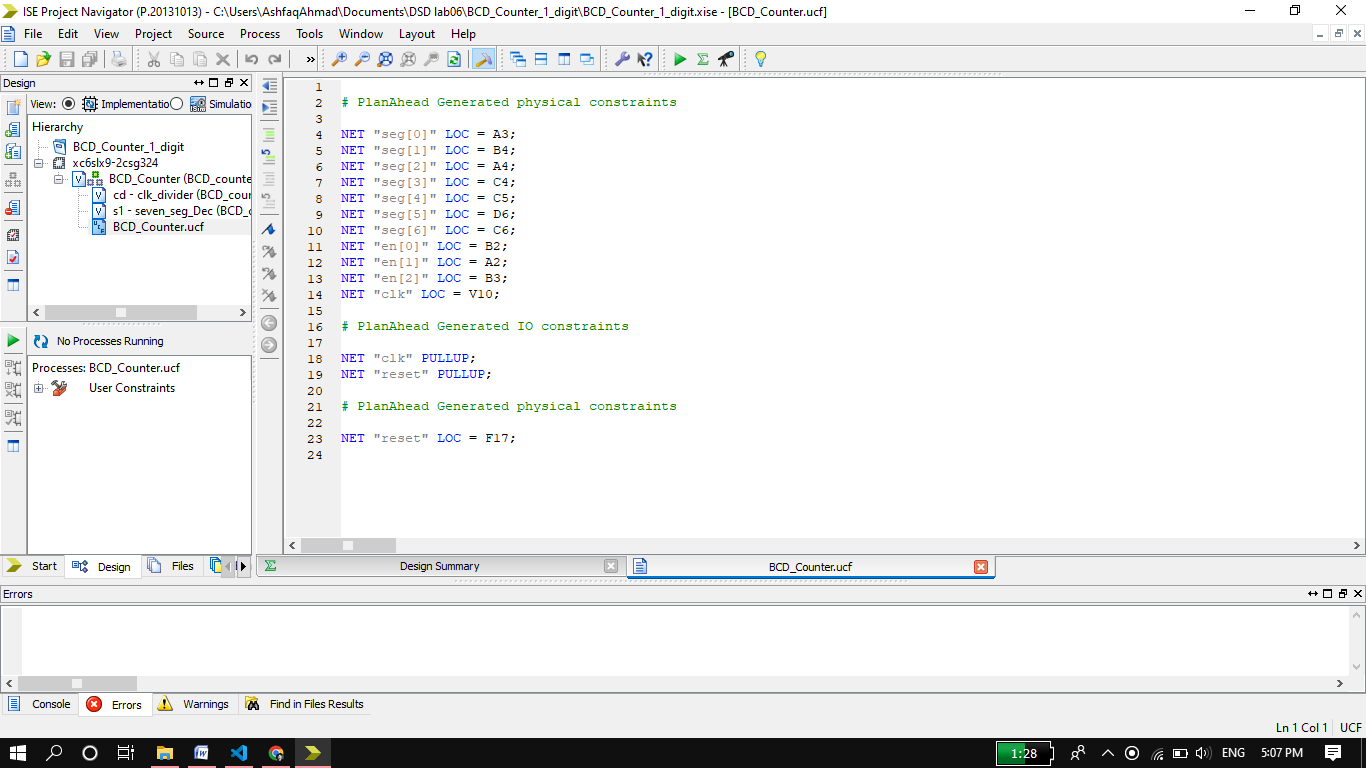
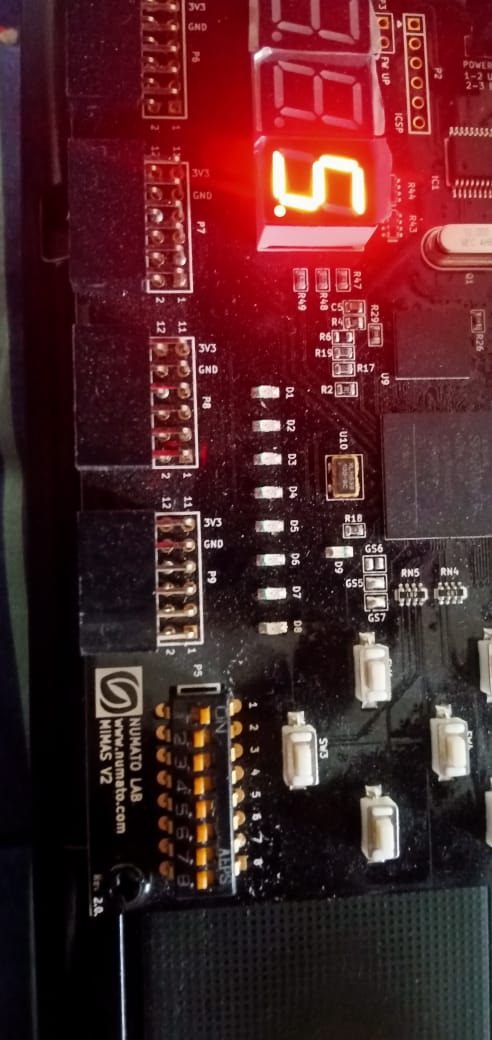
              (in==4'b0111)? 7'b1111000:

              (in==4'b1000)? 7'b0000000:

              (in==4'b1001)? 7'b0010000:7'b1111111;

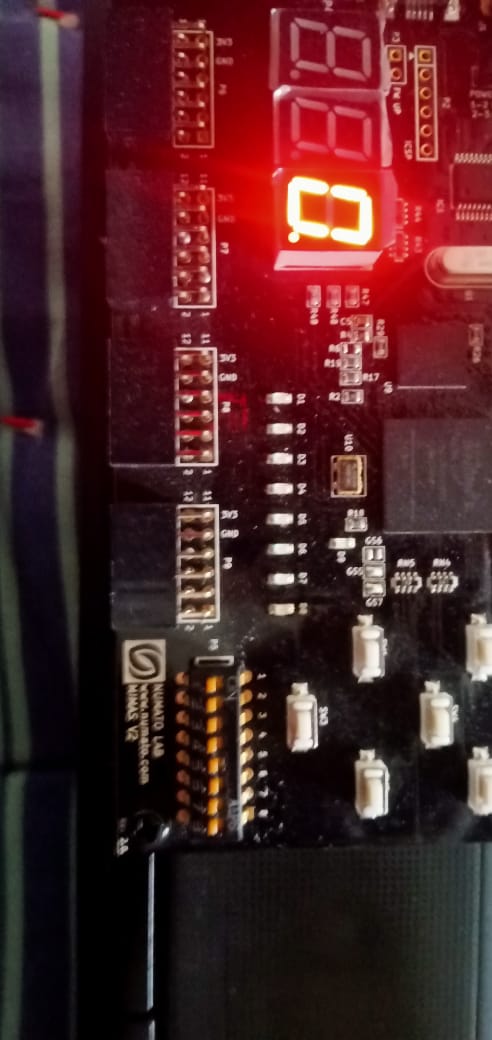
endmodule

**UCF file:**

****

**Output:**

**When reset=1; Dip switch 1 is a reset.**

**When reset=0;**

**Lab Task02:** Implement a BCD counter that runs from 000 to 999 and shows each BCD digit on the seven segment display.

**Source Code:**

**Top level Module:**

module Top\_Level(seg,clk\_100Mhz,reset,enable); //these are the only I/O which we connect with circuit.

input clk\_100Mhz,reset;  //as for these inputs values are given from circuit so they are define as input.

output [6:0]seg;    //to seg and enable we assign values so they are define as output.

output [2:0] enable;

wire [3:0] out,U,T,H;

wire clk\_5Mhz;

//sensitivity of reset and clock should be same in all modules.

clk\_divider cd(clk\_5Mhz,clk\_100Mhz,reset);

U\_T\_H\_counting UTH(U,T,H,reset,clk\_5Mhz);

Led\_multiplexing LM(out,enable,clk\_100Mhz,reset,U,T,H);

led\_segment ls(seg,out);

endmodule

**Clock Divider Module:**

module clk\_divider(clk\_5Mhz,clk\_100Mhz,reset);

input clk\_100Mhz,reset;

output clk\_5Mhz;  //here we assign value

reg clk\_5Mhz;

reg [23:0]c;    //24 bits is the size of 10Mhz.

always @(posedge clk\_100Mhz)

if(reset)

begin

    c=0;

    clk\_5Mhz=0;

end

else

begin

    c=c+1'b1;

    if(c==10000000)

    begin

        clk\_5Mhz=~clk\_5Mhz;

        c=0;

    end

end

endmodule

**0 to 999 Counting Module:**

module U\_T\_H\_counting(U,T,H,reset,clk\_5Mhz);

input reset,clk\_5Mhz;

output reg [3:0] U,T,H;

always @(posedge clk\_5Mhz)

if(reset)

begin

    U=0;

    T=0;

    H=0;

end

else

begin

    U=U+1'b1;

    if(U==10)

    begin

        U=0;

        T=T+1'b1;

        if(T==10)

        begin

            T=0;

            H=H+1'b1;

            if(H==10)

            begin

                U=0;

                T=0;

                H=0;

            end

        end

    end

end

endmodule

**LEDs Multiplexing Module:**

* Multiplexing of LEDs occur so fast that we can’t detect any one of them off.

module Led\_multiplexing(out,enable,clk\_100Mhz,reset,U,T,H);

input clk\_100Mhz,reset;

input [3:0] U,T,H;

output reg [2:0] enable;

output reg [3:0] out;

reg [13:0] delay; //14 bits is the size of 10000.

always @(posedge clk\_100Mhz) //in this module clock will be 100Mhz.

    begin

        if(reset)

        begin

            enable=3'b000; //all three leds opened

            out=0; //all three leds will display 0 at reset=1.

            delay=0;

        end

        else

        begin

            if(delay<10000)

            delay=delay+1'b1;

            else

            begin

                delay=0;

                if(enable==3'b000 || enable==3'b011)

                begin

                    enable=3'b110;

                    out=U;

                end

                else if(enable==3'b110)

                begin

                    enable=3'b101;

                    out=T;

                end

                else if(enable==3'b101)

                begin

                    enable=3'b011;

                    out=H;

                end

            end

        end

    end

endmodule

**Seven Segments Module:**

module led\_segment(seg,out);  //it is combinational circuit

input [3:0]out;

output wire [6:0] seg; //outside always block value assign to wire. in assign case value can also assign to output.

parameter [6:0] zero=7'b1000000,one=7'b1111001,two=7'b0100100,

three=7'b0110000,four=7'b0011001,five=7'b0010010,six=7'b0000010,

seven=7'b1111000,eight=7'b0000000,nine=7'b0010000;

assign seg= (out==4'd0)? zero   :

            (out==4'd1)? one    :

            (out==4'd2)? two    :

            (out==4'd3)? three  :

            (out==4'd4)? four   :

            (out==4'd5)? five   :

            (out==4'd6)? six    :

            (out==4'd7)? seven  :

            (out==4'd8)? eight  :

            (out==4'd9)? nine   : 7'b1111111;

endmodule

* Sensitivity of **Reset** input will be same in all module I,e either active low or active high otherwise circuit will not work properly,
* Sensitivity of clock should be same in all modules.

**Second Method of Coding:**

**0 to 999 Counter Module:**

module BCD\_Counter(seg,en,reset,clk);

input reset,clk;  //these are declare as input because it take values from circuit

output [6:0]seg;

wire clk\_1MHz;

output [2:0] en;  //seg and en declare as output because we assign value to it.

reg [3:0] U,T,H; //inside always block values assigned to these variables so defined as reg.

//sensitivity of reset and clock should be same in all modules.

clk\_divider cd(clk\_1MHz,clk,reset);

seven\_seg\_Dec s1(seg,clk,U,T,H,en,reset);

always @(posedge clk\_1MHz)

if(reset)

begin

    U=0;

    T=0;

    H=0;

end

else

begin

    U=U+1'b1;

    if(U==10)

    begin

        U=0;

        T=T+1'b1;

        if(T==10)

        begin

            T=0;

            H=H+1'b1;

            if(H==10)

            begin

                U=0;

                T=0;

                H=0;

            end

        end

    end

end

endmodule

**Clock Divider:**

module clk\_divider(clk\_1MHz,clk\_100MHz,reset);

input clk\_100MHz;

output clk\_1MHz;

input reset;

reg[26:0]c;

reg clk\_1MHz;

always @(posedge clk\_100MHz)

begin

    if(reset)

    begin

        c=0;

        clk\_1MHz=1;

    end

    else

    begin

        c=c+1'b1;

        if(c==10000000)

        begin

            clk\_1MHz=~clk\_1MHz;

            c=0;

        end

    end

end

endmodule

**Seven segments and LEDS Multiplexing Module**

* Multiplexing of LEDs occur so fast that we can’t detect any one of them off.

module  seven\_seg\_Dec(seg,clk,U,T,H,en,reset);

input [3:0]U,T,H;

output wire [6:0]seg;

input reset,clk;

output reg [2:0] en;

reg [3:0]in;  //it is used in always block and values assign to it so it's a reg

assign seg=(in==4'b0000)? 7'b1000000:

           (in==4'b0001)? 7'b1111001:

              (in==4'b0010)? 7'b0100100:

              (in==4'b0011)? 7'b0110000:

              (in==4'b0100)? 7'b0011001:

              (in==4'b0101)? 7'b0010010:

              (in==4'b0110)? 7'b0000010:

              (in==4'b0111)? 7'b1111000:

              (in==4'b1000)? 7'b0000000:

              (in==4'b1001)? 7'b0010000:7'b1111111;

reg [12:0] delay;

always @(posedge clk)

begin

    if(reset)

    begin

        en=3'b110;

        in=0; //here all three LED set to 0 initially. Here inputs common to all

//three LED

        delay = 0;

    end

    else

    begin

        if(delay<6000)

        delay = delay + 1'b1;

        else

        begin

            delay = 0;

            if(en==3'b110) //Multiplexing starts here.

//Multiplexing is so fast that we can’t detect any LED off.

            begin

                en=3'b101;

                in=T;

            end

            else if(en==3'b101)

            begin

                en=3'b011;

                in=H;

            end

            else if(en==3'b011)

            begin

                en=3'b110;

                in=U;

            end

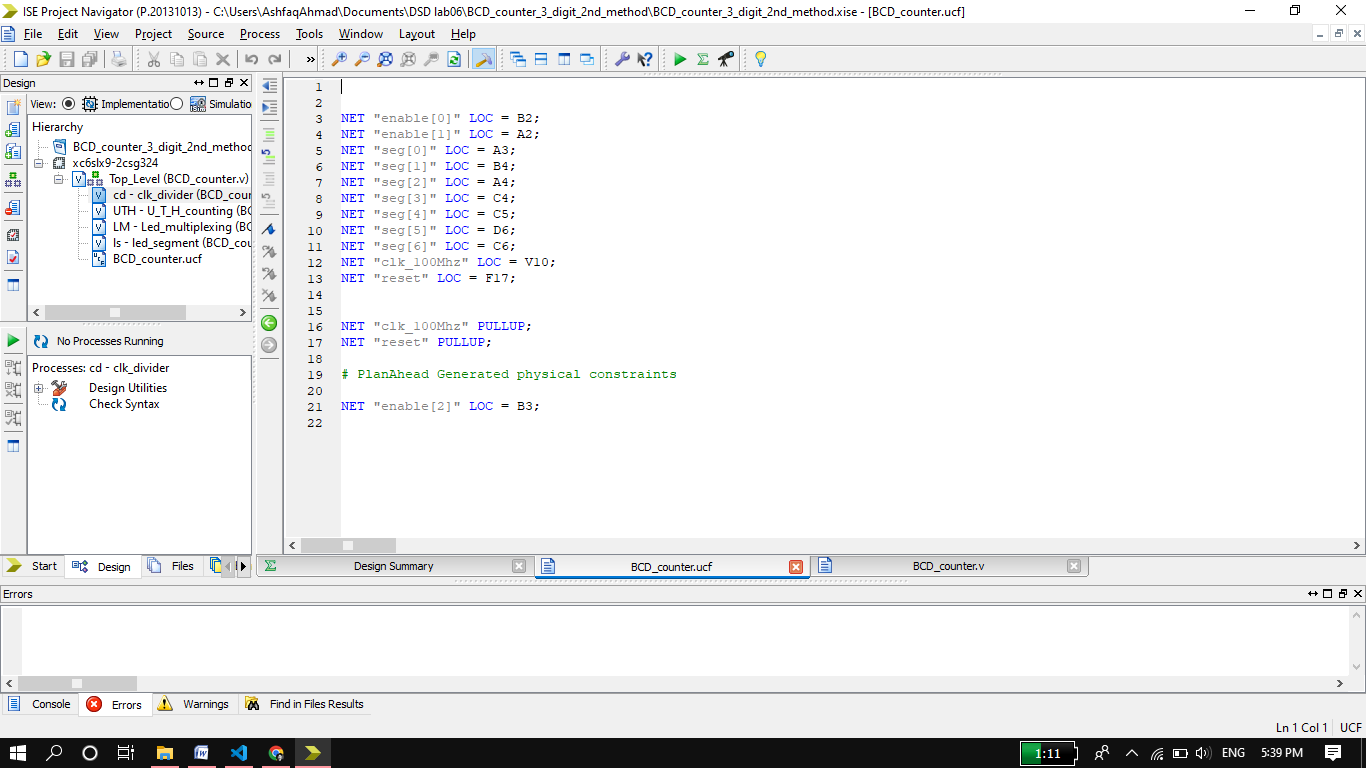
        end

    end

end

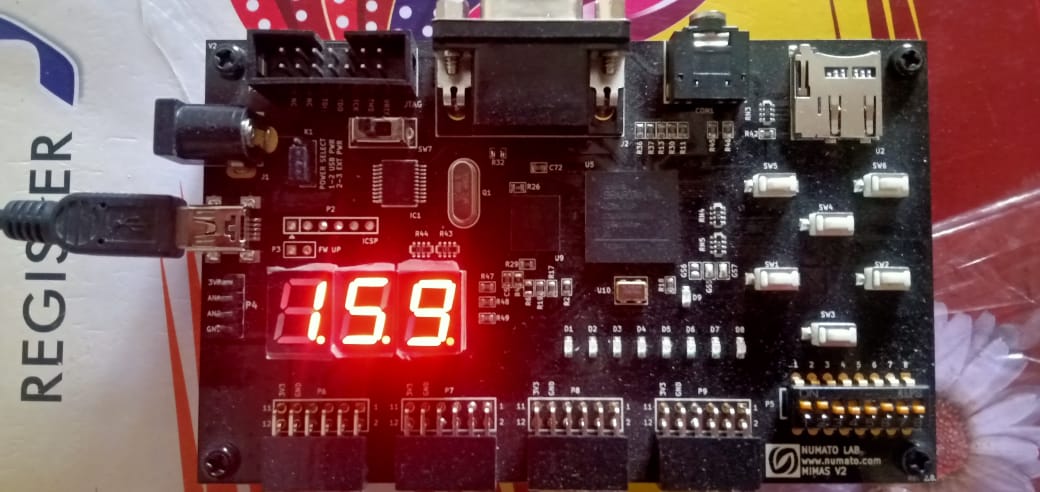
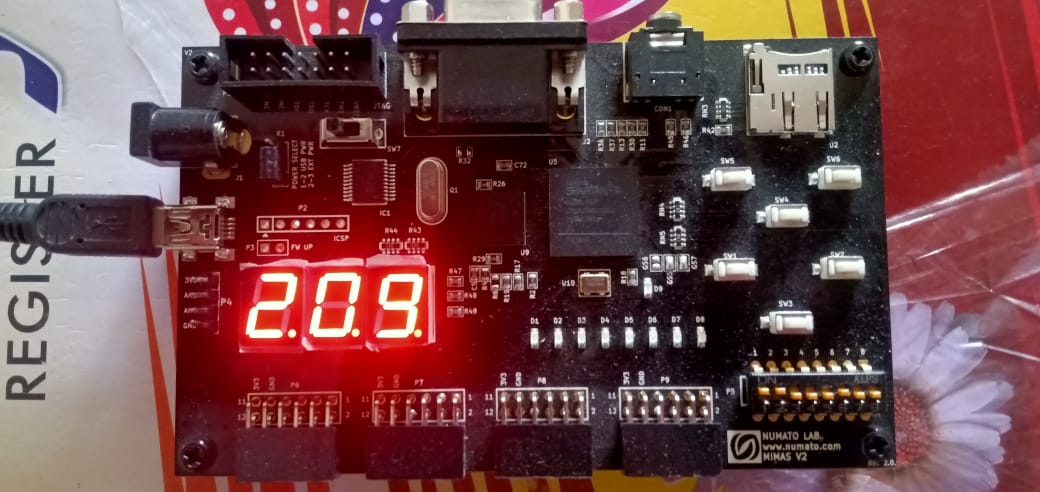
endmodule

**UCF file**: (All three program have same UCF file).

****

**Output:**

**When reset =1; Dip switch 1 is reset.**

**** ****

When Reset=0; reset is first Dip switch.

