DIGITAL SYSTEM DESIGN LAB CSE-308L

SEMESTER:6TH

LAB REPORT # 6

BCD COUNTER

Submitted By: *Zainab Khalid*
Registration No:*19PWCSE1743*
Section: A
Submitted to: *Mam Madeeha Sher*

DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING
UNIVERSITY OF ENGINEERING AND TECHNOLOGY PESHAWAR

LAB NO 6

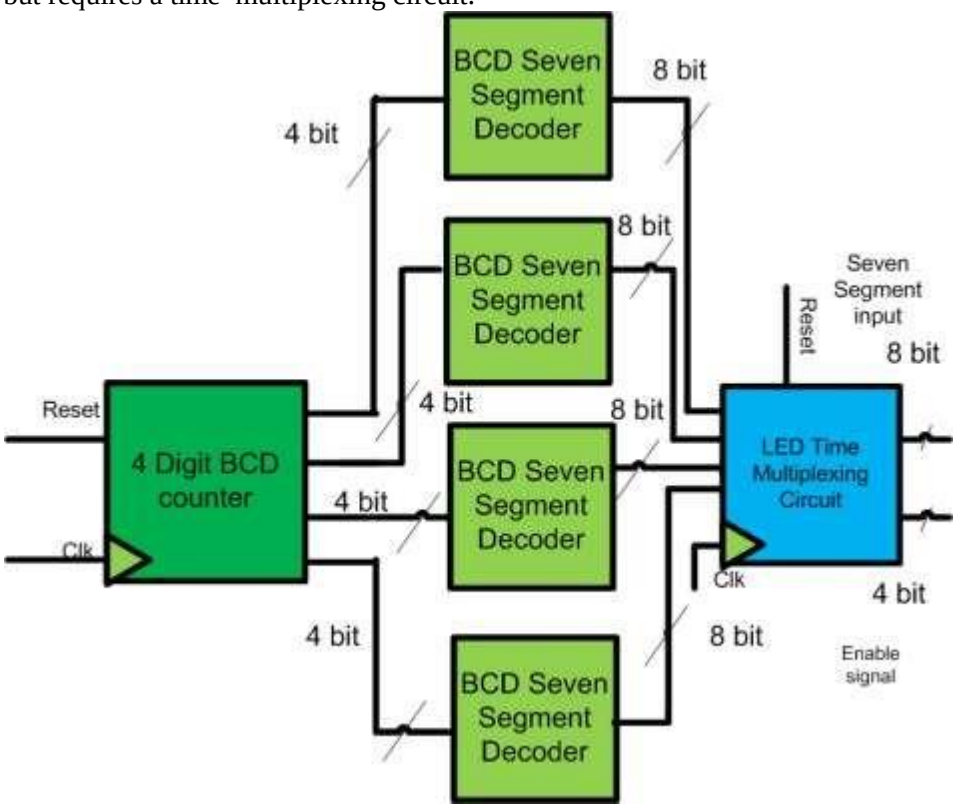
4 DIGIT BCD COUNTER ON MULTIPLEXED SEVEN SEGMENT DISPLAY

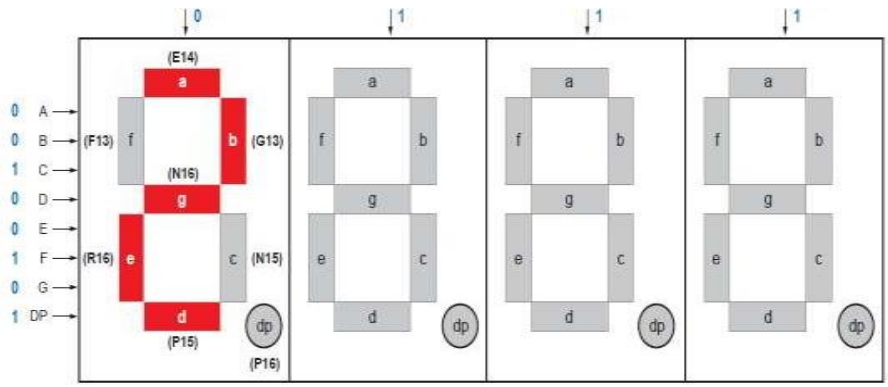
Objective:

- Learn to use time multiplexed 4 digit Seven Segment display

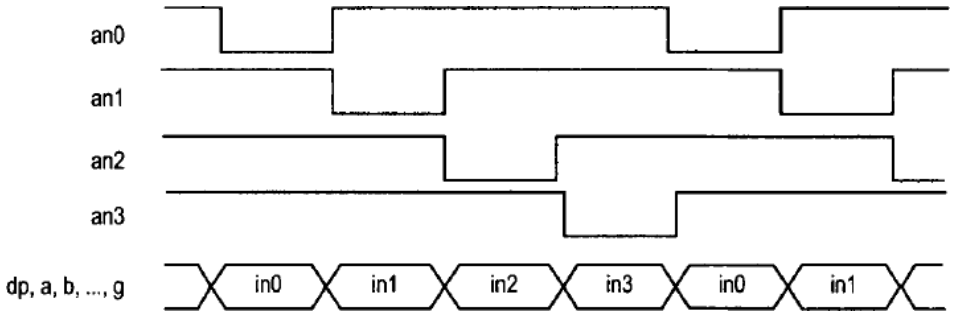
Block Diagram:

The S3 board has four seven-segment LED displays, each containing seven bars and one small round dot. To reduce the use of FPGA's I/0 pins, the S3 board uses a time-multiplexing sharing scheme. In this scheme, the four displays have their individual enable signals but share eight common signals to light the segments. All signals are active low (i.e., enabled when a signal is 0). The schematic of displaying a "2" on the leftmost LED is shown in Figure.
Note that the enable signal (i.e., an) is "0111". This configuration clearly can enable only one display at a time. We can time-multiplex the four LED patterns by enabling the four displays in turn, as shown in the simplified timing diagram. If the refreshing rate of the enable signal is fast enough, the human eye cannot distinguish the on and off intervals of the LEDs and perceives that all four displays are lit simultaneously. This scheme reduces the number of I/O pins from 32 to 12 (i.e., eight LED segments plus four enable signals) but requires a time- multiplexing circuit.





Timing Diagram of LED Multiplexing:



LED time Multiplexing:

The refresh rate of the enable signal has to be fast enough to fool our eyes but should be slow enough so that the LEDs can be turned on and off completely. The rate around the range 1000 Hz should work properly. In our design, we use an 18- bit binary counter for this purpose. The two MSBs are decoded to generate the enable signal and are used as the selection signal for multiplexing.

Lab Tasks:

Implement a BCD counter that runs from 0000 to 9999 and shows each BCD digit on the seven segment display.

Code:

```
module counter999(count7seg, rst,clk,en);
        output [6:0]count7seg;
        reg [3:0] U,T,H;
        input rst,clk;
        output [2:0]en;

        clock_divider c1(clk,rst,clk1hz);

        BCD b1(count7seg, rst,clk,en,U,T,H);
```

```
	always@(posedge clk1hz)
	begin
		if(rst)
		begin
			//count = 0;
			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
								H=H+1'b1;
								T=0;
							end
					end
			end
		end


endmodule


module clock_divider(input clk,input reset,output reg clk1hz);
integer c;
always @(posedge clk)
begin
	if(reset)
	begin
		clk1hz=1;
		c=0;
	end
	else
	begin
		c=c+1;
		if(c==10000000)
		begin
			clk1hz=~clk1hz;
			c=0;
		end
	end
end
endmodule


module BCD(out, rst,clk,en,U,T,H);

input [3:0] U, T, H;
output reg [2:0]en;
input clk,rst;
```

```
reg[3:0] in;
output [6:0] out;

assign out = (in==4'd0)?7'b1000000:(in==4'd1)?7'b1111001:(in==4'd2)?7'b0100100:
                        (in==4'd3)?7'b0110000:
                        (in==4'd4)?7'b0011001:
                        (in==4'd5)?7'b0010010:
                        (in==4'd6)?7'b0000010:
                        (in==4'd7)?7'b1111000:
                        (in==4'd8)?7'b0000000:
                        (in==4'd9)?7'b0010000:7'b1111111;

integer delay;
always @(posedge clk)
begin
        if(rst)
        begin
                in = 0;
                en = 3'b110;
                delay = 0;
        end
        else
        begin
                if(delay<10000)
                        delay = delay +1;
                else
                begin
                        delay = 0;
                        if(en==3'b110)
                        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:

```
NET "count7seg[0]" LOC = A3;
NET "count7seg[1]" LOC = B4;
NET "count7seg[2]" LOC = A4;
NET "count7seg[3]" LOC = C4;
NET "count7seg[4]" LOC = C5;
```

```
NET "count7seg[5]" LOC = D6;
NET "count7seg[6]" LOC = C6;

NET "rst" LOC = C17 | PULLUP;
NET "clk" LOC = V10 | PULLUP;

# PlanAhead Generated physical constraints

NET "en[0]" LOC = B3;
NET "en[1]" LOC = A3;
NET "en[2]" LOC = A3;


# PlanAhead Generated IO constraints
```

Output: