Lab 08

4 digit BCD Counter on Multiplexed Seven Segment Display

Spring 2025

Submitted by: **Mohsin Sajjad**

Registration No: **22pwsce2149**

Class Section: **A**

"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:

Engr. Faheem Jan

Month Day, Year (04 05, 2025)

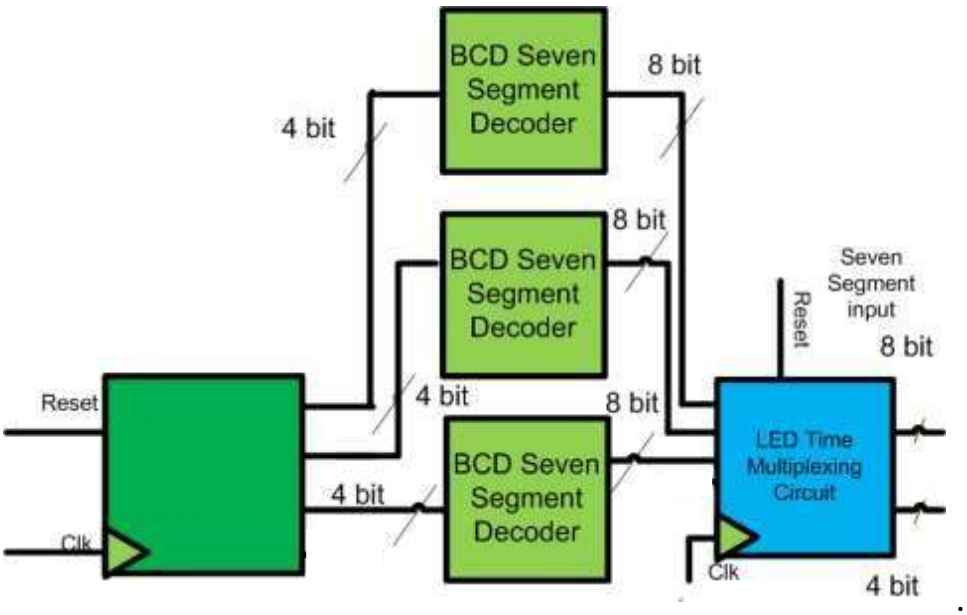
Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

4 digit BCD Counter on Multiplexed Seven Segment Display

Objective:

Learn to use time multiplexed 4 digit Seven Segment display



Lab Task:

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

CODE:

```
// BCD Counter
module BCD_Counter (input CLOCK, input CLR, output reg [3:0] Q);
   always @(negedge CLOCK or posedge CLR) begin
      if (CLR)
         Q <= 4'd0;
      else if (Q == 4'd9)
         Q <= 4'd0;
      else
         Q <= Q + 1;
   end
endmodule


// Three BCD Counters (unit, ten, hundred)
```

```
module three_BCD(input CLOCK, input CLR, output [3:0] BCDu, BCDt, BCDh);
    wire tc_u, tc_t;

    BCD_Counter bcdu(CLOCK, CLR, BCDu);
    BCD_Counter bcdt(BCDu[3], CLR, BCDt);
    BCD_Counter bcdh(BCDt[3], CLR, BCDh);
endmodule

// BCD to 7 Segment
module BCD_to_7seg(input [3:0] bcd, output reg [7:0] seg);
    always @(*) begin
        case (bcd)
            4'b0000: seg = 8'b11000000; // 0
            4'b0001: seg = 8'b11111001; // 1
            4'b0010: seg = 8'b10100100; // 2
            4'b0011: seg = 8'b10110000; // 3
            4'b0100: seg = 8'b10011001; // 4
            4'b0101: seg = 8'b10010010; // 5
            4'b0110: seg = 8'b10000010; // 6
            4'b0111: seg = 8'b11111000; // 7
            4'b1000: seg = 8'b10000000; // 8
            4'b1001: seg = 8'b10010000; // 9
            default: seg = 8'b11111111; // error
        endcase
    end
endmodule

// Clock Divider
module Clock_Divider(input clock_in, output reg clock_out);
    reg [27:0] counter = 28'd0;
    parameter DIVISOR = 28'd100000000;  // Adjust according to FPGA clock

    always @(posedge clock_in) begin
        if (counter == DIVISOR-1) begin
```

```
            counter <= 28'd0;
            clock_out <= ~clock_out;
        end else begin
            counter <= counter + 1;
        end
    end
endmodule

// 18-bit Counter
module Counter18bit(input clk, input rst, output reg [17:0] count);
    always @(posedge clk or posedge rst) begin
        if (rst)
            count <= 18'd0;
        else
            count <= count + 1;
    end
endmodule

// 4X1 Multiplexer
module mux4X1(
    input [7:0] bcd0,
    input [7:0] bcd1,
    input [7:0] bcd2,
    input [1:0] select,
    output reg [7:0] out
);
    always @(*) begin
        case(select)
            2'b00: out = bcd0;
            2'b01: out = bcd1;
            2'b10: out = bcd2;
            default: out = 8'b11111111; // blank
        endcase
    end
```

Like someone **clapping very fast** — 100 million claps per second.

But you only want to turn on a light once per second**.

So, you:

1. Count the claps.
2. When you've counted enough (say 100 million),
3. You flip a switch (turn the light ON if it was OFF, or OFF if it was ON).
4. Then you start counting again.

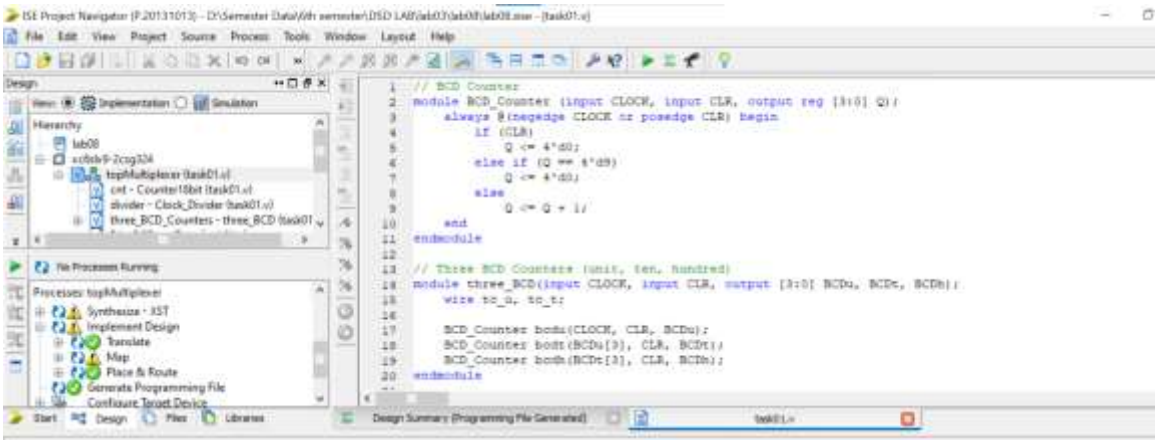
This way, even though claps are fast, you slow things down using **counting**.

Now in Verilog:
reg [27:0] counter = 0; // a big number to count fast clock pulses
```

This is your clap counter.
It increases every time the FPGA clock ticks.

if (counter == DIVISOR-1) begin
    counter <= 0;
    clock_out <= ~clock_out;  // flip the output signal
end
```

If the counter reaches the target (like 100 million), flip `clock_out`.
 Reset the counter to start counting again.

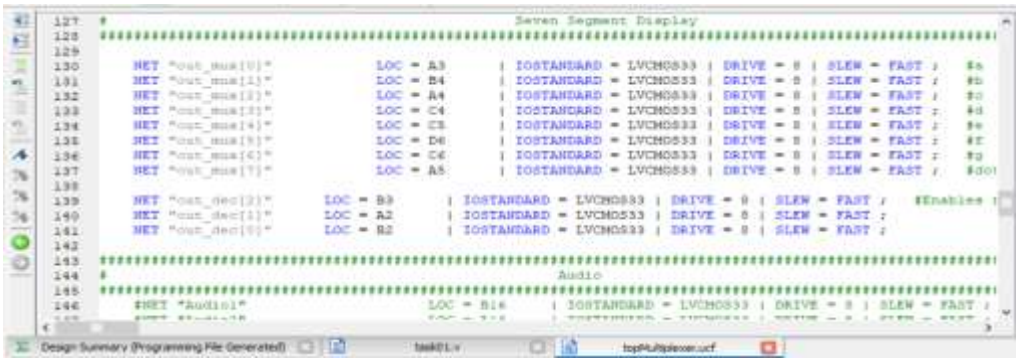
```
endmodule

// 2X4 Decoder
module decoder2X4(
    input [1:0] select,
    output reg [3:0] out
);
    always @(*) begin
        case(select)
            2'b00: out = 4'b1110;
            2'b01: out = 4'b1101;
            2'b10: out = 4'b1011;
            2'b11: out = 4'b0111;
        endcase
    end
endmodule

// Top Module
module topMultiplexer(
    input clk,
    input rst,
    output [3:0] out_dec,
    output [7:0] out_mux
);
    wire slow_clock;
    wire [3:0] BCDu, BCDt, BCDh;
    wire [7:0] bcd1, bcd2, bcd3;
    wire [17:0] counter_out;

    Counter18bit cnt(clk, rst, counter_out);
    Clock_Divider divider(clk, slow_clock);

    three_BCD three_BCD_Counters(slow_clock, rst, BCDu, BCDt, BCDh);
    BCD_to_7seg B1(BCDu, bcd1);
```

```
    BCD_to_7seg B2(BCDt, bcd2);
    BCD_to_7seg B3(BCDh, bcd3);

    mux4X1 Mux(bcd1, bcd2, bcd3, counter_out[17:16], out_mux);
    decoder2X4 d1(counter_out[17:16], out_dec);
endmodule
```



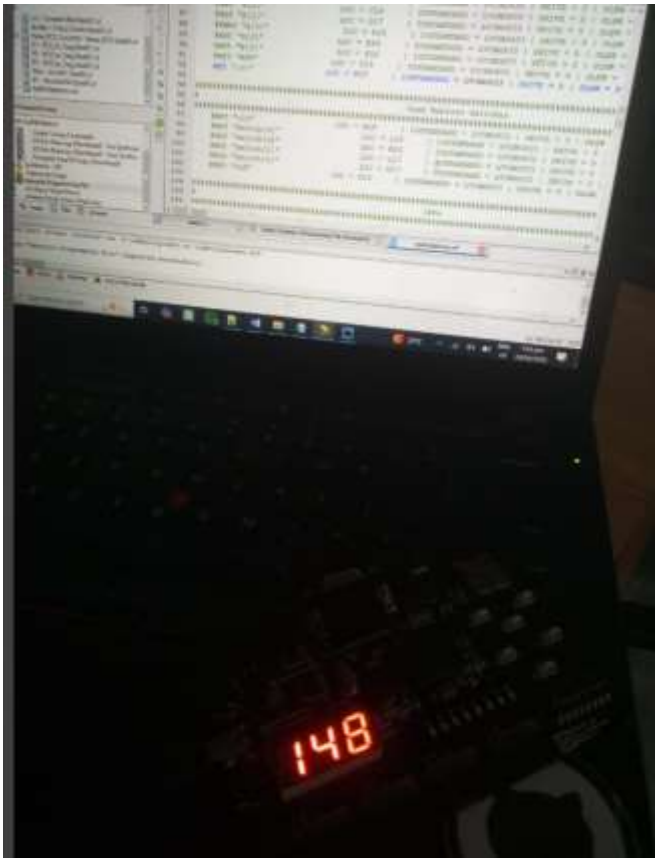
```
#**************************************************************************************************
#                                          UCF for Mimas V2
#**************************************************************************************************

CONFIG VCCAUX = "3.3" ;

   NET "clk"                  LOC = V10     | IOSTANDARD = LVCMOS33 | PERIOD = 100MHz ;
   #NET "CLK_12MHz"                LOC = D9      | IOSTANDARD = LVCMOS33 | PERIOD = 12MHz ;

##################################################################################################
#                                          UART Interface
```

```
##################################################################################################
#                                          DIP Switches
##################################################################################################
    #NET "A[3]"                LOC = C17     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP;
    #NET "A[2]"                LOC = C18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP;
    #NET "A[1]"                LOC = D17     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP;
    ##NET "A[0]"                LOC = D18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP
    #NET "B[3]"                LOC = E18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP;
    #NET "B[2]"                LOC = E16     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP;
    #NET "RST"                 LOC = F18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP;
    NET "rst"                  LOC = F17     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP;

##################################################################################################
#                                          Push Buttons Switches
```



```
#                                          Seven Segment Display
##################################################################################################

    NET "out_mux[0]"           LOC = A3      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #a
    NET "out_mux[1]"           LOC = B4      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #b
    NET "out_mux[2]"           LOC = A4      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #c
    NET "out_mux[3]"           LOC = C4      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #d
    NET "out_mux[4]"           LOC = C3      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #e
    NET "out_mux[5]"           LOC = D4      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #f
    NET "out_mux[6]"           LOC = C4      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #g
    NET "out_mux[7]"           LOC = A5      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #dot

    NET "out_dec[2]"      LOC = B3      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #Enables
    NET "out_dec[1]"      LOC = A2      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
    NET "out_dec[0]"      LOC = B2      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

##################################################################################################
#                                          Audio
##################################################################################################
    #NET "Audio1"                LOC = B16     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
```

OUTPUT:



Conclusion:

We implements a digital BCD counter system with display functionality. It includes a chained three-digit BCD counter for units, tens, and hundreds, each displayed on a 7-segment display. A clock divider and an 18-bit counter manage timing and multiplexing of displays. The `mux4X1` selects which BCD digit to show, while the `decoder2X4` activates the corresponding display. This design demonstrates modular digital design using counters, decoders, multiplexers, and display drivers. It is ideal for learning and implementing multi-digit BCD counting and display systems in FPGAs.