**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

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);

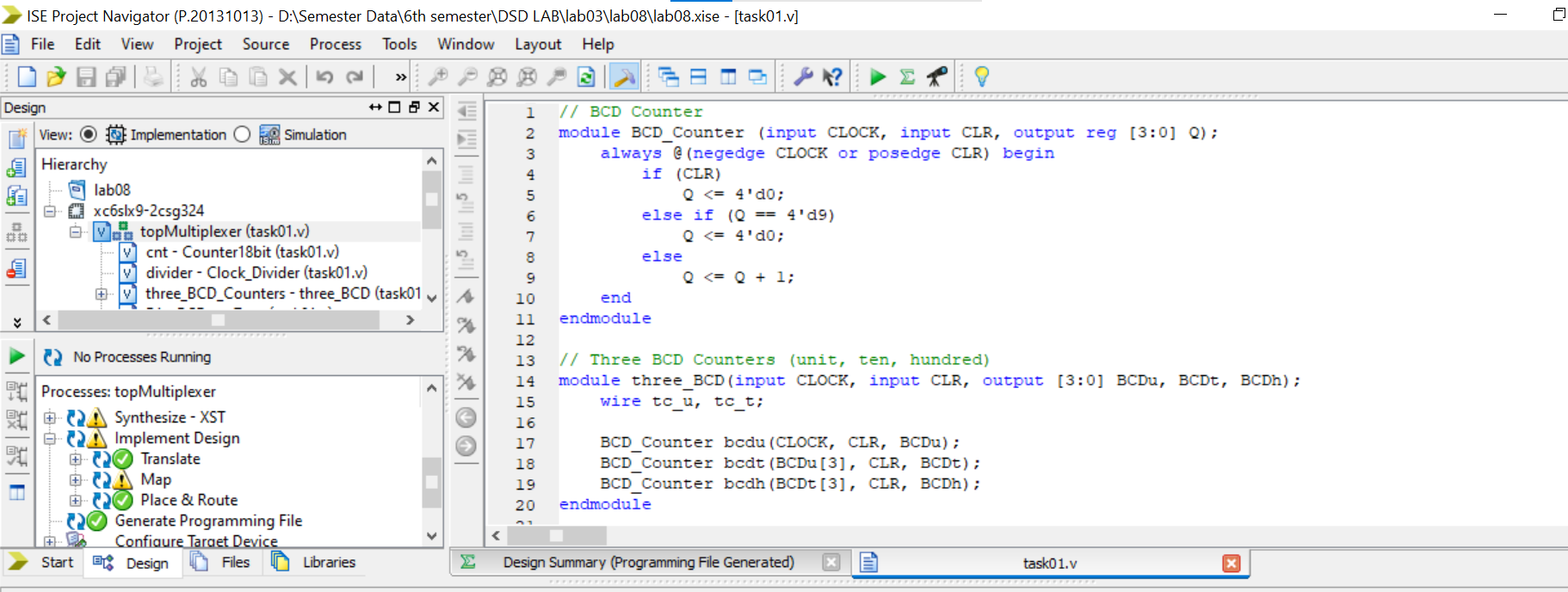
BCD\_to\_7seg B2(BCDt, bcd2);

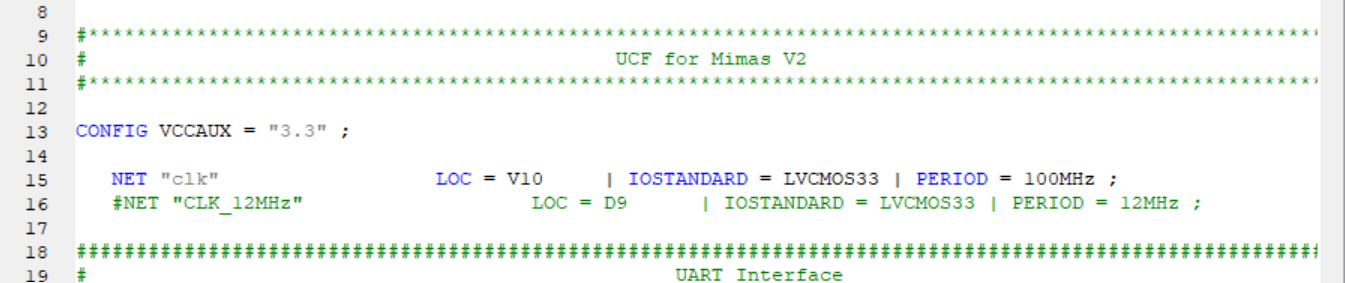
BCD\_to\_7seg B3(BCDh, bcd3);

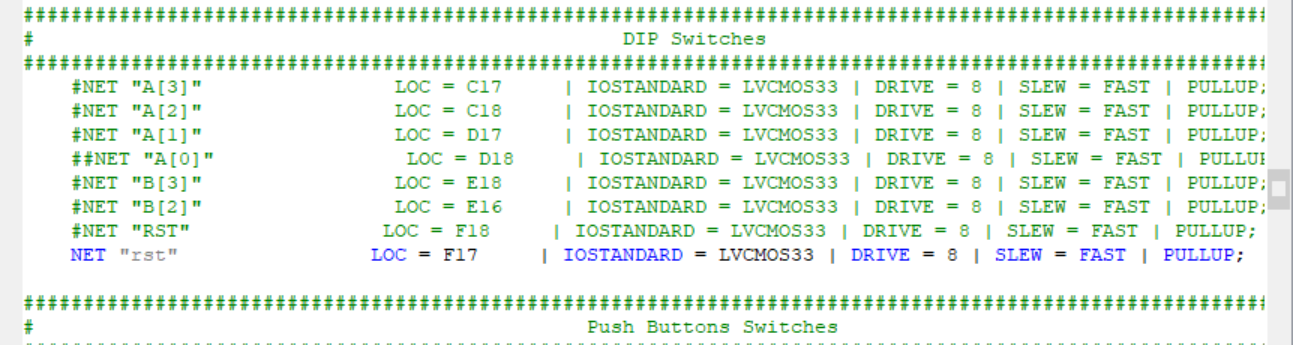
mux4X1 Mux(bcd1, bcd2, bcd3, counter\_out[17:16], out\_mux);

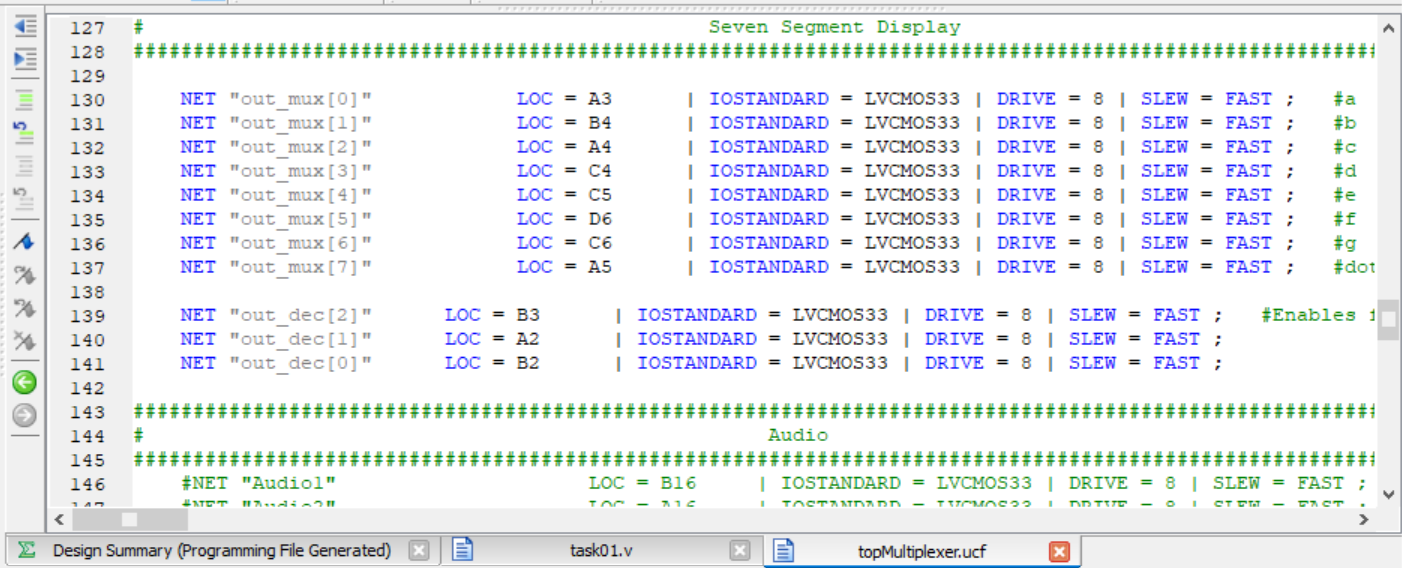
decoder2X4 d1(counter\_out[17:16], out\_dec);

endmodule

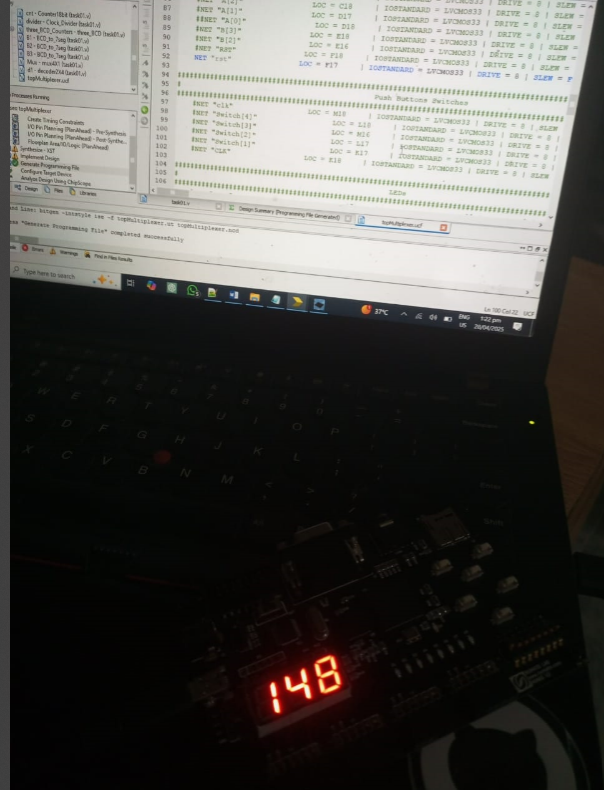
****

****

****

****

**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.