**Lab 09**

**Design of a Light switch**



**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 (11 05, 2025)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**Design of a Light switch**

**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”.

**Block Diagram:**

A screenshot of a computer

Description automatically generated with medium confidence

**CODE:**

**module light\_switch (**

**input clk,**

**input rst,**

**input button,**

**output reg led**

**);**

**parameter S0 = 2'b00; // LED off**

**parameter S1 = 2'b01; // LED on**

**reg [1:0] state, next\_state;**

**// Sequential block**

**always @(posedge clk or posedge rst) begin**

**if (rst) begin**

**state <= S0;**

**led <= 0;**

**end else begin**

**state <= next\_state;**

**case (next\_state)**

**S0: led <= 0;**

**S1: led <= 1;**

**default: led <= 0;**

**endcase**

**end**

**end**

**// Combinational block**

**always @(\*) begin**

**case(state)**

**S0: next\_state = (button) ? S1 : S0;**

**S1: next\_state = (button) ? S0 : S1;**

**default: next\_state = S0;**

**endcase**

**end**

**endmodule**

**// Clock Divider**

**module Clock\_Divider (**

**input clock\_in,**

**output reg clock\_out = 0**

**);**

**reg [27:0] counter = 28'd0;**

**parameter DIVISOR = 28'd100000000;**

**always @(posedge clock\_in) begin**

**if (counter == DIVISOR - 1) begin**

**counter <= 0;**

**clock\_out <= ~clock\_out;**

**end else begin**

**counter <= counter + 1;**

**end**

**end**

**endmodule**

**// D Flip-Flop**

**module df1 (**

**output reg q,**

**input d,**

**input clk,**

**input rst**

**);**

**always @(posedge clk or negedge rst) begin**

**if (!rst)**

**q <= 1'b0;**

**else**

**q <= d;**

**end**

**endmodule**

**// Synchronizer (2 DFFs in series)**

**module Synchronizer (**

**output sb,**

**input d,**

**input clk,**

**input rst**

**);**

**wire q;**

**df1 inst1(q, d, clk, rst);**

**df1 inst2(sb, q, clk, rst);**

**endmodule**

**// Level to Pulse Generator**

**module level\_to\_pulse (**

**input wire clk,**

**input wire rst,**

**input wire level,**

**output wire pulse**

**);**

**wire q;**

**df1 dff (**

**.q(q),**

**.d(level),**

**.clk(clk),**

**.rst(rst)**

**);**

**assign pulse = level & ~q; // Rising edge detection**

**endmodule**

**// Top Module**

**module top (**

**input wire btn, // Raw button input**

**input wire RST, // Asynchronous reset**

**input wire clk, // System clock**

**output wire led // Output LED**

**);**

**wire sync\_button; // Synchronized button signal**

**wire slow\_clock; // Divided (slow) clock**

**wire pulse\_button; // One-cycle pulse on button press**

**// Clock Divider to reduce system clock frequency**

**Clock\_Divider c1 (**

**.clock\_in(clk),**

**.clock\_out(slow\_clock)**

**);**

**// Synchronize button input to slow clock**

**Synchronizer s1 (**

**.clk(slow\_clock),**

**.d(btn),**

**.rst(RST),**

**.sb(sync\_button)**

**);**

**// Convert level-based button press to one-clock-cycle pulse**

**level\_to\_pulse l1 (**

**.clk(slow\_clock),**

**.rst(RST),**

**.level(sync\_button),**

**.pulse(pulse\_button)**

**);**

**// Light switch FSM**

**light\_switch l2 (**

**.clk(slow\_clock),**

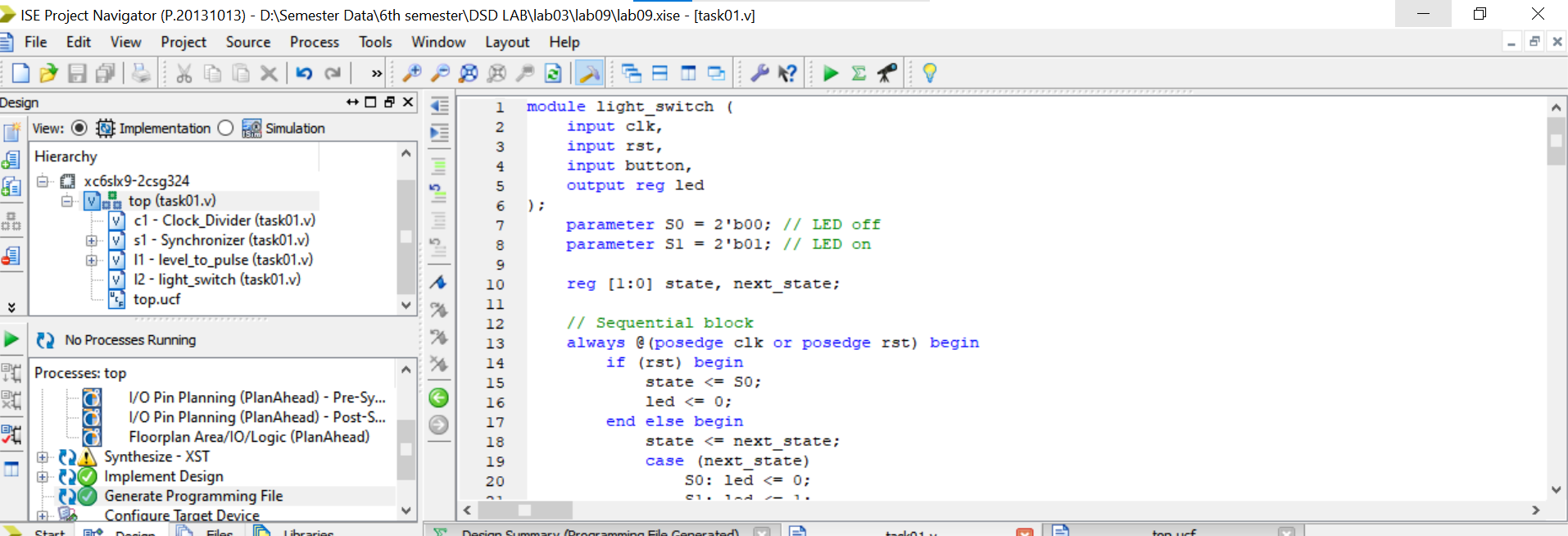
**.rst(RST),**

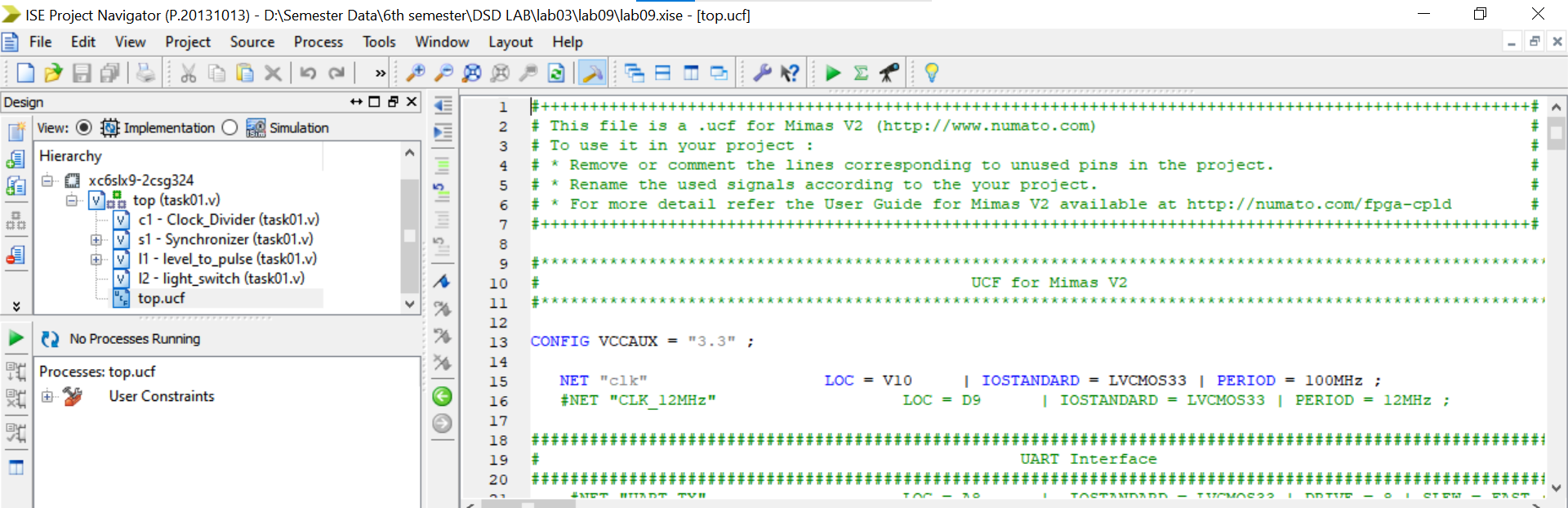
**.button(pulse\_button),**

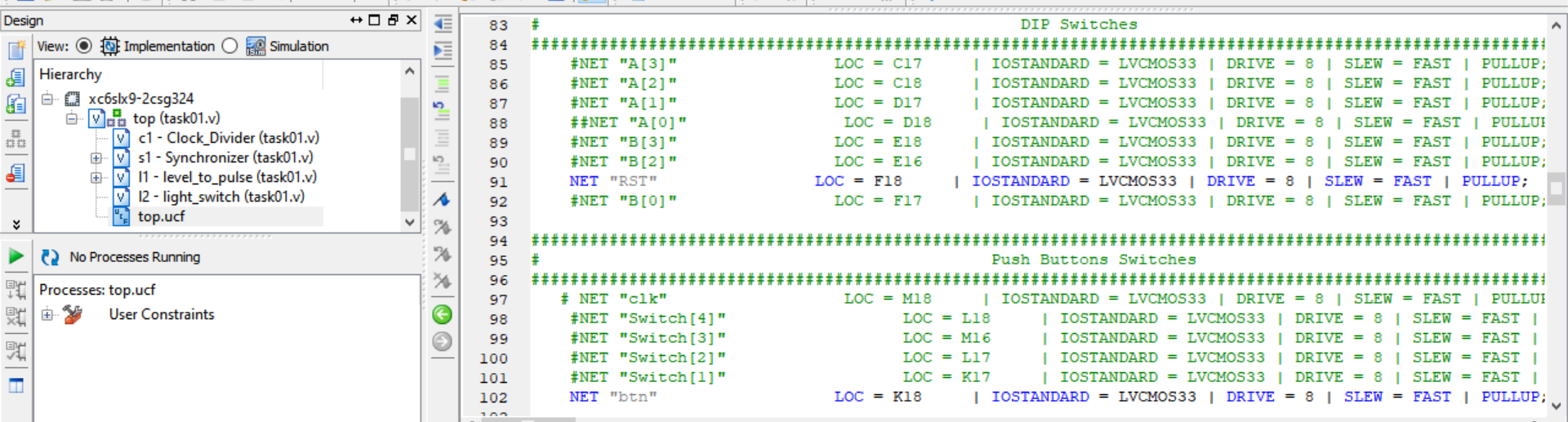
**.led(led)**

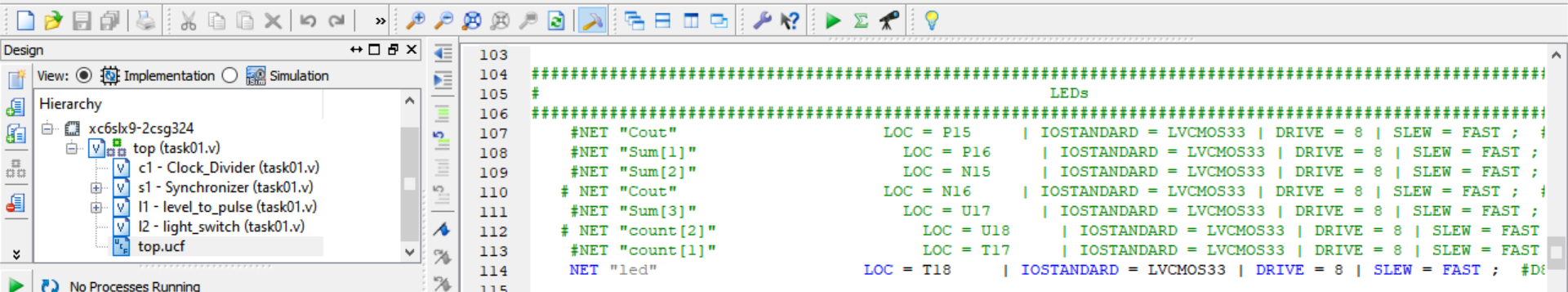
**);**

**endmodule**

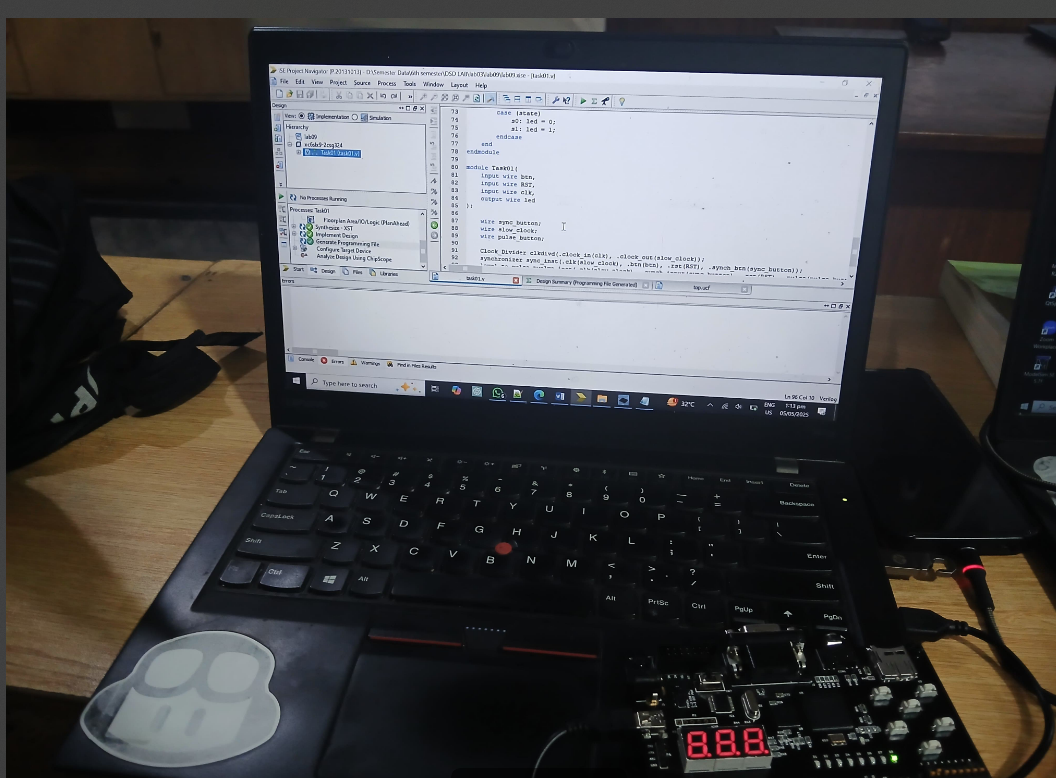
****

****

****

****

**Output:**

****

**Conclusion:**

This Verilog project creates a reliable light switch using an FSM. A clock divider slows down the system clock. Button signals are synchronized and turned into short pulses. The FSM changes the LED state with each button press. This setup ensures smooth LED control without glitches.