

Lab 4 Report

by

Jonathan Huynh #016137719

Adam Godfrey #015981472

Instructor: Dr. Mohamed Aly

Class: ECE 3300L .E02-OU - Verilog Design

July 11th, 2025

Code with Explanation:

7 Segment Driver	
<pre>`timescale 1ns / 1ps module seg7_driver(input clk, // system clock input rst_n, // active-low reset input [15:0] SW, // 16-bit switch input output reg [6:0] Cnode, // segment outputs A-G output dp, // decimal point (unused) output [7:0] AN, // active-low digit enables output [15:0] LED // led output); reg [19:0] refresh_counter; // controls refresh rate reg [3:0] current_nibble; // current digit to display assign dp = 1'b1; // keep decimal point off // HEX to 7-segment conversion always @(current_nibble) case (current_nibble) 4'd0: Cnode = 7'b1000000; 4'd1: Cnode = 7'b1001111; 4'd2: Cnode = 7'b0100100; 4'd3: Cnode = 7'b0110000; 4'd4: Cnode = 7'b0011001; 4'd5: Cnode = 7'b0010010; 4'd6: Cnode = 7'b0000010; 4'd7: Cnode = 7'b1111000; 4'd8: Cnode = 7'b0000000; 4'd9: Cnode = 7'b0010000; 4'd10: Cnode = 7'b0001000; // A 4'd11: Cnode = 7'b0000011; // b 4'd12: Cnode = 7'b1000110; // C 4'd13: Cnode = 7'b0100001; // d 4'd14: Cnode = 7'b0000110; // E 4'd15: Cnode = 7'b0001110; // F default: Cnode = 7'b1111111; endcase</pre>	<p>The first part sets up the inputs and outputs: the clock (clk), reset (rst_n), switch input (SW), the 7-segment display outputs (Cnode), the digit select lines (AN), and an LED output that mirrors the switch state. We defined a refresh counter to control the refresh rate of the display. This counter is used to cycle through each digit quickly enough so all digits appear turned on at the same time to the human eye. We also assigned the decimal point (dp) to always stay off by setting it to 1.</p> <p>The main logic works by taking the top 3 bits of the refresh counter to get a value from 0 to 7, which we call digit_index. This tells the system which digit position is currently being displayed. Depending on that index, we select one of the 4-bit nibbles from the SW input. Since we only have 4 nibbles (16 bits), but 8 digits on the display, we just repeated the same 4 nibbles across all 8 digit positions. After getting the nibble, we use a case statement to map that nibble to the correct 7-segment pattern. This way, the correct segments (A–G) light up to display a hex digit (0–F). The conversion is done in a combinational block so it updates instantly when the nibble changes.</p> <p>To make sure only one digit is active at a time, we used another case block to create an active_digit signal that controls which digit is turned on. Since the digit enables are active-low, we set one bit to 0 at a time based on the current digit_index, and all others to 1. Finally, we connected active_digit to the AN output and passed the switch values directly to the LED output for feedback. This project helped us understand how to implement multiplexing in hardware and how to break up a multi-digit display task into smaller logic blocks that work together.</p>

```

// counter increment
always @(posedge clk or negedge rst_n)
    if (!rst_n)
        refresh_counter <= 0;
    else
        refresh_counter <= refresh_counter +
1;

// extract nibble to display
wire [2:0] digit_index =
refresh_counter[19:17];
always @(digit_index or SW)
    case (digit_index)
        3'd0: current_nibble = SW[3:0];
        3'd1: current_nibble = SW[7:4];
        3'd2: current_nibble = SW[11:8];
        3'd3: current_nibble = SW[15:12];
        3'd4: current_nibble = SW[3:0];
        3'd5: current_nibble = SW[7:4];
        3'd6: current_nibble = SW[11:8];
        3'd7: current_nibble = SW[15:12];
        default: current_nibble = 4'b0000;
    endcase

// control which digit is active
reg [7:0] active_digit;
always @(digit_index)
    case (digit_index)
        3'd0: active_digit = 8'b11111110;
        3'd1: active_digit = 8'b11111101;
        3'd2: active_digit = 8'b11111011;
        3'd3: active_digit = 8'b11110111;
        3'd4: active_digit = 8'b11101111;
        3'd5: active_digit = 8'b11011111;
        3'd6: active_digit = 8'b10111111;
        3'd7: active_digit = 8'b01111111;
        default: active_digit = 8'b11111111;
    endcase

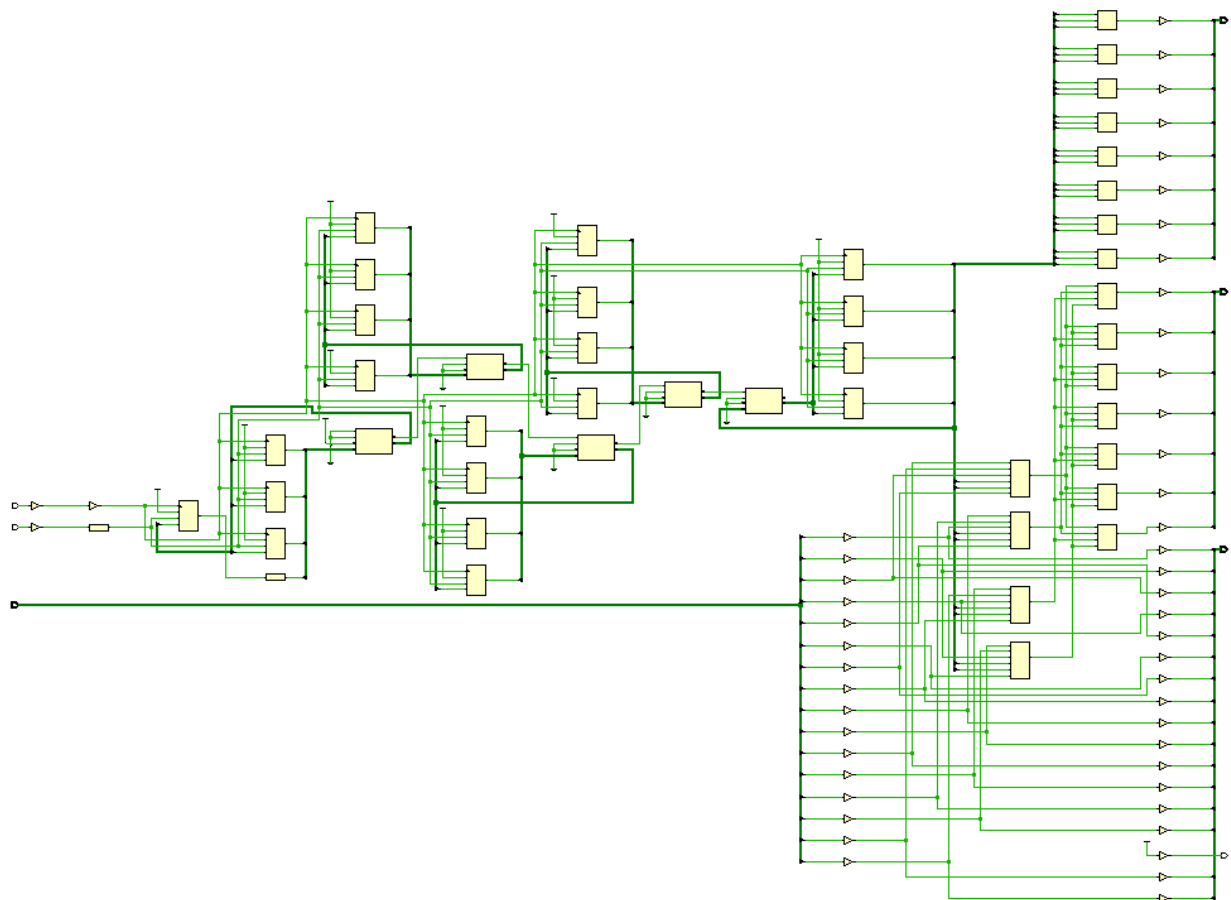
assign AN = active_digit;
assign LED = SW;

endmodule

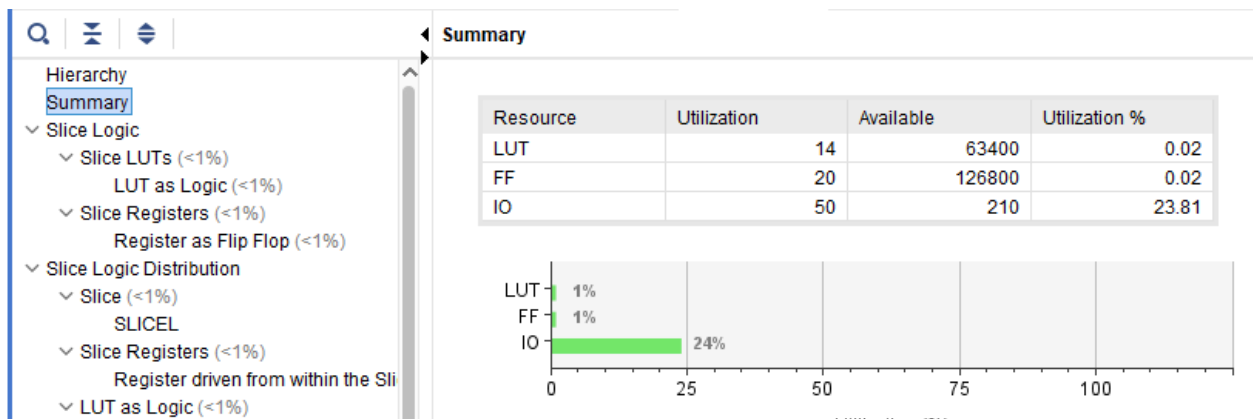
```

Screenshot Proofs:

Schematic:



Resource Utilization Table:



Power Utilization:

Settings

Summary (0.126 W, Margin: N/A)

Power Supply

Utilization Details

Hierarchical (0.028 W)

Clocks (0.001 W)

Signals (<0.001 W)

Data (<0.001 W)

Set/Reset (0 W)

Logic (<0.001 W)

I/O (0.027 W)

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power: 0.126 W

Design Power Budget: Not Specified

Process: typical

Power Budget Margin: N/A

Junction Temperature: 25.6°C

Thermal Margin: 59.4°C (12.9 W)

Ambient Temperature: 25.0 °C

Effective θJA: 4.6°C/W

Power supplied to off-chip devices: 0 W

Confidence level: Low

[Launch Power Constraint Advisor](#) to find and fix invalid switching activity

On-Chip Power

23%

77%

Dynamic: 0.028 W (23%)

Clocks: 0.001 W (2%)

Signals: <0.001 W (1%)

Logic: <0.001 W (1%)

I/O: 0.027 W (96%)

Device Static: 0.097 W (77%)

Timing Summary:

Tcl Console Messages Log Reports Design Runs Timing x

Design Timing Summary

General Information

Timer Settings

Design Timing Summary

Clock Summary (1)

Methodology Summary (16)

Check Timing (16)

Intra-Clock Paths

Inter-Clock Paths

Other Path Groups

User Ignored Paths

Setup

Hold

Pulse Width

Worst Negative Slack (WNS): 7.440 ns

Worst Hold Slack (WHS): 0.324 ns

Worst Pulse Width Slack (WPWS): 4.500 ns

Total Negative Slack (TNS): 0.000 ns

Total Hold Slack (THS): 0.000 ns

Total Pulse Width Negative Slack (TPWS): 0.000 ns

Number of Failing Endpoints: 0

Number of Failing Endpoints: 0

Number of Failing Endpoints: 0

Total Number of Endpoints: 20

Total Number of Endpoints: 20

Total Number of Endpoints: 21

All user specified timing constraints are met.

Simulation Waveform:



Partner Contributions:

Team Member	Contribution	% Effort
Jonathan Huynh	Implementation, Debugging, Demo, Written Report	50%
Adam Godfrey	Verilog Code, Test Bench	50%