



COL215 – Digital Logic and System Design

**Department of Computer Science &
Engineering, IIT Delhi**

Semester I, 2025-26

Lab Assignment 3(Friday) 7-Segment Display

Developed By :

Ritik Raj 2024CS10086

Bharmal Bhakar 2024CS10403

Introduction

In this assignment, we captured and displayed a 4-digit decimal number on the 7-Segment Display devices of the Basys3 board. We used time-multiplexing to drive all 4 digits using a single set of cathode signals and individual anode selectors.

This was done in 2 parts:

1. Capturing and storing the four digits using the input slider switches on the Basys3 board.
2. Display the four digits on the 7-Segment LED display devices of the board.

Design Decisions

We have taken reference from our lab 2 assignment which was about to display one digit on a 7 segment display. We implemented our module by using the following approaches in our code of assignment 2:

1. 4-Digit Multiplexing Approach

We have chosen a time-multiplexing method. This reduces the number of required connections and ensures efficient resource usage, as only one digit is driven at a time but at a high refresh rate, making all digits appear continuously lit to the human eye.

```
always @(posedge clk) begin
    counter <= counter + 1;
    if (counter == 17'd99999) begin // 1 ms
        counter <= 0;
        active_anode <= (active_anode == 2'd3) ? 2'd0 : active_anode + 1;
    end
end
```

2. Switch-Based Data Entry

Ten input switches (sw[9:0]) were used to represent digits 0–9 in a one-hot encoding. Four additional control switches (sw10–sw13) were assigned to select which digit's data (units, tens, hundreds, thousands) gets stored, providing flexibility in updating any digit individually.

```
input wire sw10, sw11, sw12, sw13,
input wire [9:0] sw,
```

3. Separate Registers for Each Digit

Individual registers (dataout0–dataout3) were used to store the selected digit values. This structure simplifies the update logic and avoids overwriting other digits when one digit is being changed.

```
always @(posedge clk) begin
    if (sw10) begin
        dataout0[9:0] <= sw[9:0];
    end else if (sw11) begin
        dataout1[9:0] <= sw[9:0];
    end else if (sw12) begin
        dataout2[9:0] <= sw[9:0];
    end else if (sw13) begin
        dataout3[9:0] <= sw[9:0];
    end
end
```

4. Clock Divider for Multiplexing Speed

A 17-bit counter was implemented to divide the FPGA's 100 MHz clock down to ~1 kHz for smooth multiplexing. This ensures there is no flicker and the display is stable to the human eye. The display-driving logic uses if–else conditions to decode the one-hot input into 7-segment patterns. This makes the design simple to understand and debug for a beginner-level FPGA project.

5. Active-Low Anode Control

Since the Basys3 7-segment displays are common-anode type, active-low signals were used for an0 – an3. This matches the hardware design of the board and avoids extra inverters.

```
// ----- active_anode -----
always @(*) begin
    case (active_anode)
        2'd0: current_digit = dataout0;
        2'd1: current_digit = dataout1;
        2'd2: current_digit = dataout2;
        2'd3: current_digit = dataout3;
        default: current_digit = 10'b0;
    endcase
end
```

6. Variable used

Clk - system clock runs with rising edge

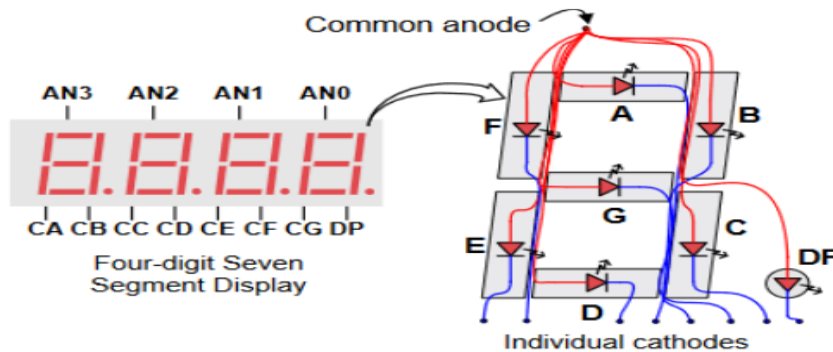
a - g represents 7 segments of the display(i.e. cathodes)

an0 -an3 - anodes of all 4 display

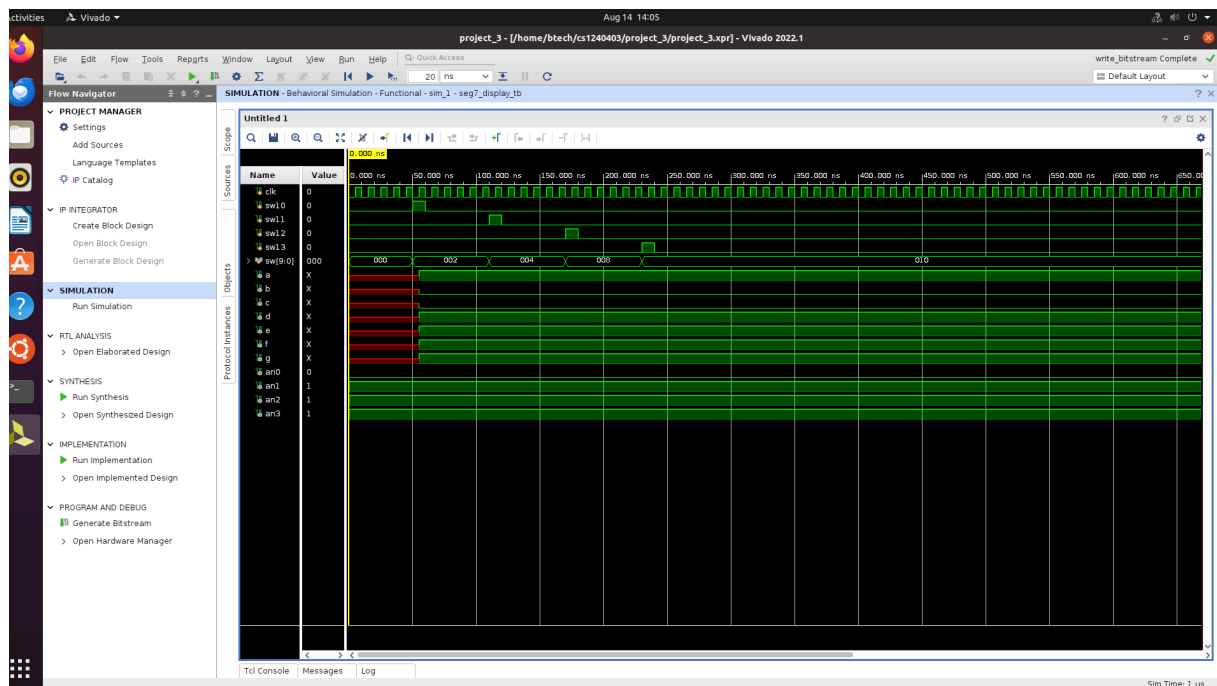
Active_anode - 2-bit register that keeps track of which digit is currently active in the multiplexing cycle.

Counter - 7-bit register that counts clock cycles to generate the 1 ms time slot for multiplexing.

Current_digit - It comes from one of dataout0..3 depending on active_anode.



Simulation snapshots



The simulation verifies the 7-segment display module, showing correct segment outputs (a–g) for different switch inputs (sw1–sw19) in sync with the clock. The an0–an3 signals enable the respective display digit as inputs change over time.



Synthesis report

Part Resources:

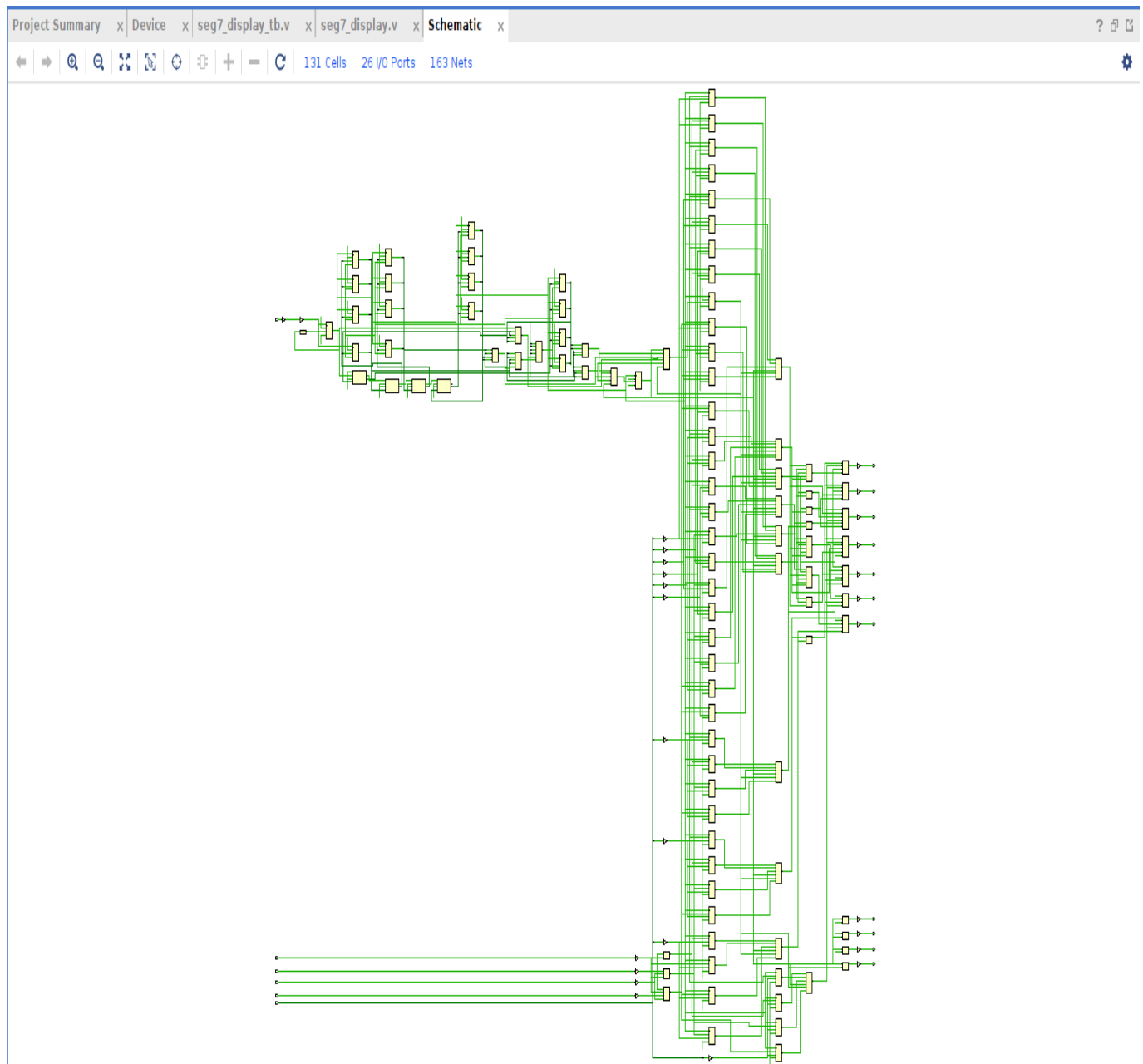
DSPs: 90 (col length:60)

BRAMs: 100 (col length: RAMB18 60 RAMB36 30)

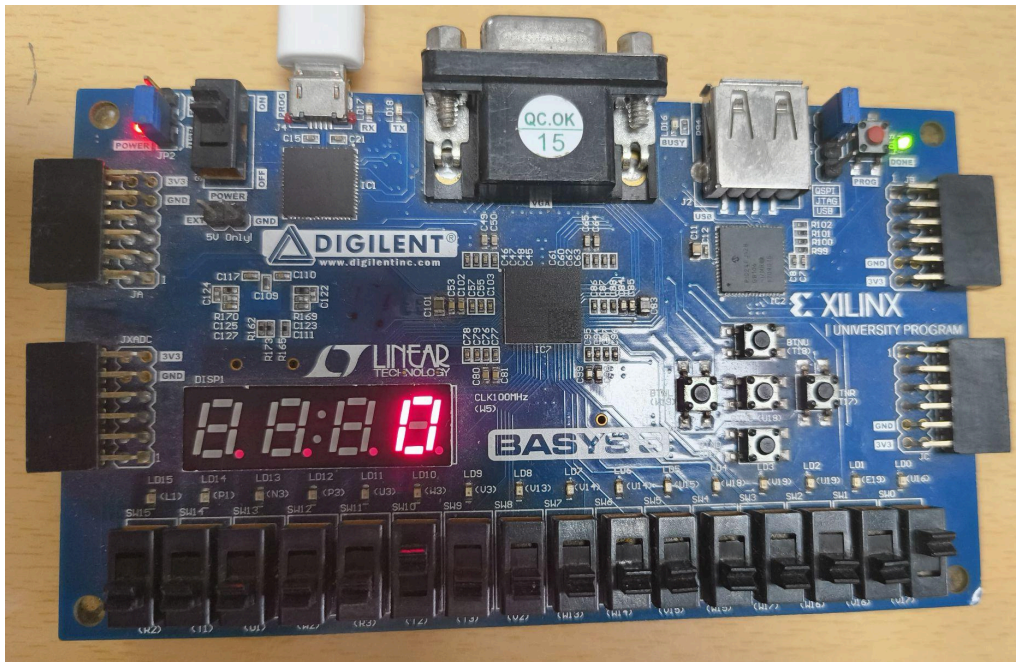
	Cell	Count
1	BUFG	1
2	CARRY4	4
3	LUT1	1
4	LUT2	9
5	LUT3	2
6	LUT4	6
7	LUT5	6
8	LUT6	17
9	FDRE	59
10	IBUF	15
11	OBUF	11

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP
▼  synth_1	constrs_1	Synthesis Out-of-date												34	59	0	0	0
 impl_1	constrs_1	Implementation Out-of-date	5.985	0.000	0.261	0.000		0.000	0.074	0	25 Warn			32	59	0	0	0

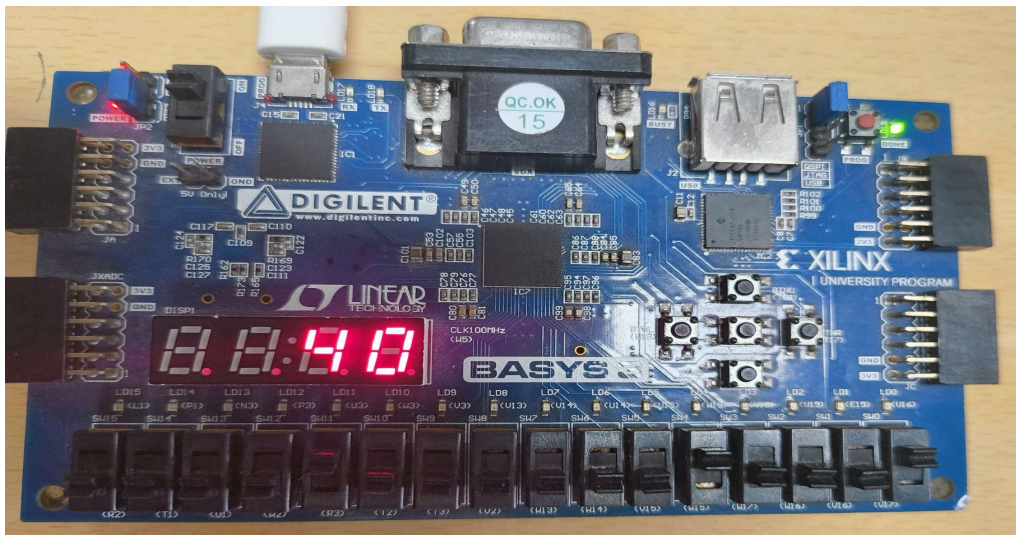
Generated schematics



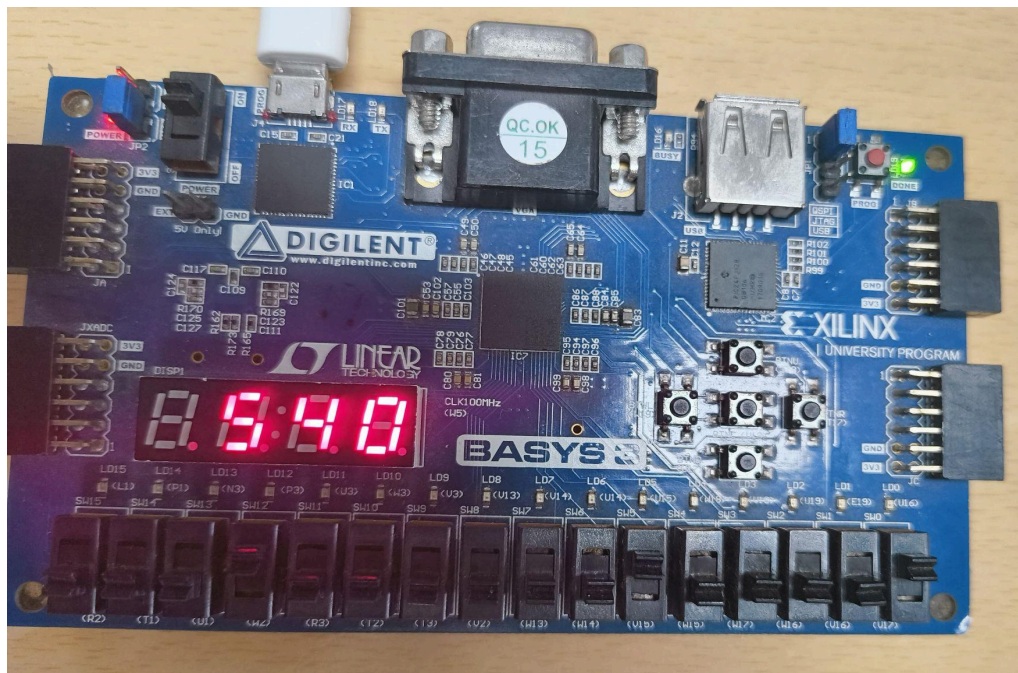
On board implementation



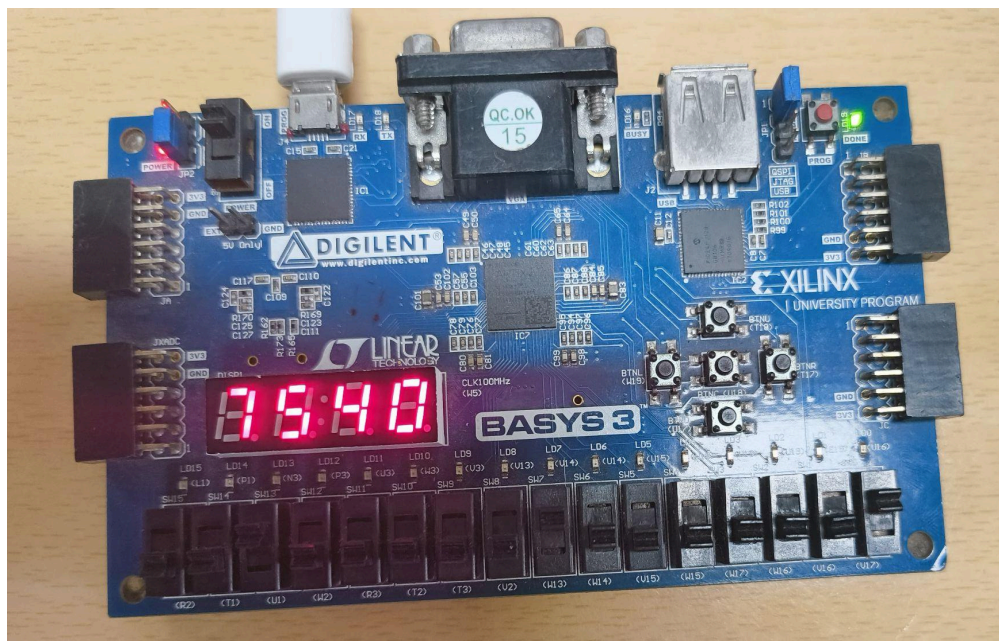
When sw10 is on and sw0 is on then display SD0 shows 0
And rest are off.



When sw10 if off(data is stored) and sw11 is turned on and now
when sw4 is turned on with sw0 the display shows 4 and 0.



Now when again sw11 is off(data is stored) and sw12 is turned on, when sw5 is turned on the display shows 5, 4 and 0.



Now only sw13 is turned (previous data is stored which can be seen on display) and then when sw7 is turned on the display will show 7, 5, 4 and 0.

Conclusion

The implemented Verilog design successfully captures 4 separate decimal digits via slider switches and displays them correctly on the 4-digit 7-segment display using time-multiplexed control. The circuit performs correctly under simulation and hardware testing on the Basys3 board.