COL215
Digital Logic & System Design

Assignment 2

Submitted by:

Dhairya Kuchhal	(2024CS50396)
Siddhant Agrawal	(2024CS50469)

August 21, 2025

1 Task

In this lab, we were tasked with designing and implementing a circuit on the Basys3 FPGA board that utilizes the 10 onboard switches (SW0–SW9) as inputs for each of the four seven-segment displays, showing the corresponding decimal digit (0–9) on them. The design was executed in behavioral Verilog and mapped using the constraints file specific to Basys3. The system prioritizes the highest-numbered switch when multiple switches are activated. The outputs are displayed on all the seven-segment displays based on the digit assigned to each of the anodes. This process includedd the following steps:

- Capture and store the four digits using the input slider switches on the Basys3 board
- Display the four digits on the seven segment displays of the board.

Finally, we had to understand the synthesis results. We reused our seven-segment display code from Assignment 2.

2 Design Decisions

We presume that the reader has a background understanding of the code and functionality from Assignment 2, particularly how we utilized casex for priority assignment. In this instance, we required a method to acquire and store the digits for each display we intended to show on the board. The Basys Board features common cathodes for its four displays (the four anodes), requiring a way to differentiate among them if we wish to exhibit different digits on each one. Initially, we had to devise a method for capturing the digits we aimed to display, followed by the process of displaying them individually. For the latter task, we'll employ an innovative approach, taking into account that humans cannot easily detect closely spaced flashing lights

2.1 Capturing the digits

We chose to use a non-blocking assignment to capture the digits whenever the clock edges positively (with a 10ns cycle). As a result, the digits are updated in real-time based on the changes made to the switches, and these digits are recorded depending on which of the SW[10] - SW[13] switches are activated, with SW[10] representing the least significant digit and SW[13] representing the most significant. In essence, we assigned a switch specifically for capturing the digits.

2.2 Displaying the digits

To achieve this, we utilize a 17-bit counter capable of counting up to 100,000 (and even more, but for our needs, this is adequate), along with a 2-bit active_anode counter that tracks which anode is currently active. Every 1 millisecond, the anodes are cycled through. The active anode allows us to display only one digit at a time, showing the digit that corresponds to the stored value. In theory, if we activated the other display digits, they would also present the same number as the one intended for display; this is a limitation of the Basys Board, which cannot show different digits simultaneously. Nonetheless, by cleverly flashing the displays at 1-millisecond intervals, these flashes become imperceptible to the human eye, creating the illusion of multiple digits being displayed on separate anodes.

3 Verilog Code

3.1 Code for Displaying digits

```verilog
module seven_seg_decoder(
input [9:0] value,
output reg [6:0] seg
);

always @(*) begin
    casex (value)
        10'b1xxxxxxxxx: seg = 7'b0010000; // 9
        10'b01xxxxxxxx: seg = 7'b0000000; // 8
        10'b001xxxxxxx: seg = 7'b1111000; // 7
        10'b0001xxxxxx: seg = 7'b0000010; // 6
        10'b00001xxxxx: seg = 7'b0010010; // 5
        10'b000001xxxx: seg = 7'b0011001; // 4
        10'b0000001xxx: seg = 7'b0110000; // 3
        10'b00000001xx: seg = 7'b0100100; // 2
        10'b000000001x: seg = 7'b1111001; // 1
        10'b0000000001: seg = 7'b1000000; // 0
        default: seg = 7'b1111111; // blank
    endcase
end
endmodule

module main(
    input clk,
    input [13:0] sw,
    output reg [6:0] led,
    output [3:0] anode,
    output dp
    );

    assign dp = 1;

    reg [9:0] digit0, digit1, digit2, digit3;
     // capture the digits
    always @(posedge clk) begin
        if (sw[10]) digit0 <= sw[9:0];
        if (sw[11]) digit1 <= sw[9:0];
        if (sw[12]) digit2 <= sw[9:0];
        if (sw[13]) digit3 <= sw[9:0];
    end

    reg [16:0] counter = 0; // 17-bit counter
    reg [1:0] active_anode = 0; // 2-bit module seven_seg_decoder

    always @(posedge clk) begin
        counter <= counter + 1;
        if (counter == 17'd999) begin
            counter <= 0;
            active_anode <= (active_anode==2'd3) ? 2'd0 : active_anode + 1;
        end
    end
    // temporary wire for decoder output
    wire [6:0] seg_out;
     // instantiate decoder module
    seven_seg_decoder decoder_inst(
        .value((active_anode==2'd0) ? digit0 :
        (active_anode==2'd1) ? digit1 :
        (active_anode==2'd2) ? digit2 :
        digit3),
    .seg(seg_out)
    );
     // assign to led
    always @(*) begin
        led = seg_out;
    end
     // anode logic (active-low)
    assign anode[0] = ~(active_anode == 2'd0);
    assign anode[1] = ~(active_anode == 2'd1);
    assign anode[2] = ~(active_anode == 2'd2);
    assign anode[3] = ~(active_anode == 2'd3);
endmodule
```

3.2 Code for Simulation and testing

```verilog
module tb_seven_segment_all();
    reg clk;
    reg [13:0] SW;
    wire [6:0] led;
    wire [3:0] anode;
    main uut (
    .clk(clk),
    .sw(SW),
    .led(led),
    .anode(anode)
    );
    initial clk = 0;
    always #5 clk = ~clk;

    initial begin
```

```
        SW = 14'b0;

        SW[10] = 1; SW[9:0] = 10'b1000000000; #10000 SW[10] = 0;
        #10000 SW[11] = 1; SW[9:0] = 10'b0100000000; #10000 SW[11] = 0;
        #10000 SW[12] = 1; SW[9:0] = 10'b0010000000; #10000 SW[12] = 0;
        #10000 SW[13] = 1; SW[9:0] = 10'b0001000000; #10000 SW[13] = 0;
        #10000 SW[12] = 1; SW[9:0] = 10'b0000000010; #10000 SW[12] = 0;
        #10000 SW[10] = 1; SW[9:0] = 10'b0000010000; #10000 SW[10] = 0;
        $stop;
    end
endmodule
```

3.3 Mapping of Pins on Basys3

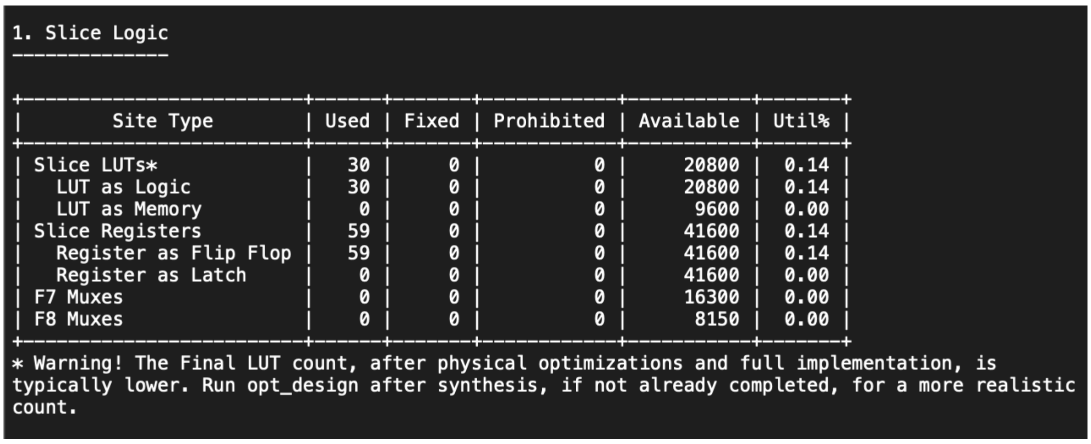
The constraint file (basys3.xdc) was edited to bind logical inputs and outputs to physical switches and the display segments in a user-friendly, simple left to right manner. The following table summarizes the mapping:

Signal Type	FGPA Pin	Description
Input	SW[0...13]	Switches
Output	W4, V4, U4, U2	Anodes
Output	W7, W6, U8, V8, U5, V5, U7	Cathodes
Output	V7	Decimal Point

Table 1: Mapping of inputs and outputs on Basys 3 FPGA board

4 Synthesis Report

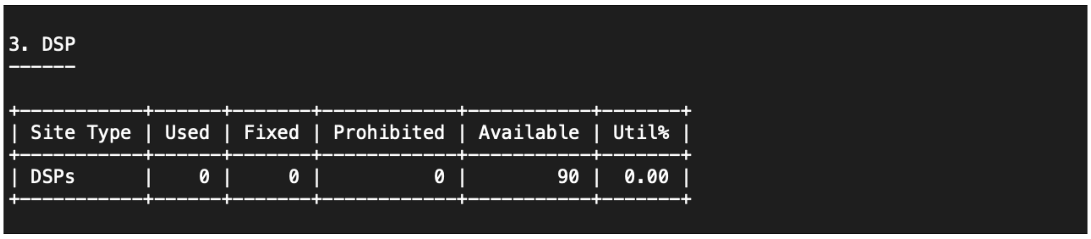
The following tables show the main resource counts. Other details are present in the synthesis report.



```
1. Slice Logic
--------------

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs*             |   30 |     0 |          0 |     20800 |  0.14 |
|   LUT as Logic          |   30 |     0 |          0 |     20800 |  0.14 |
|   LUT as Memory         |    0 |     0 |          0 |      9600 |  0.00 |
| Slice Registers         |   59 |     0 |          0 |     41600 |  0.14 |
|   Register as Flip Flop |   59 |     0 |          0 |     41600 |  0.14 |
|   Register as Latch     |    0 |     0 |          0 |     41600 |  0.00 |
| F7 Muxes                |    0 |     0 |          0 |     16300 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |      8150 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
* Warning! The Final LUT count, after physical optimizations and full implementation, is
typically lower. Run opt_design after synthesis, if not already completed, for a more realistic
count.
```

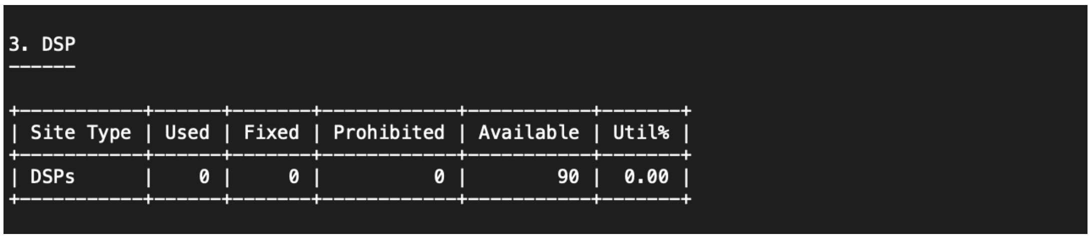
Figure 1: Flip-Flops and LUT



```
3. DSP
------

+-----------+------+-------+------------+-----------+-------+
| Site Type | Used | Fixed | Prohibited | Available | Util% |
+-----------+------+-------+------------+-----------+-------+
| DSPs      |    0 |     0 |          0 |        90 |  0.00 |
+-----------+------+-------+------------+-----------+-------+
```

Figure 2: DSP usage



```
3. DSP
------

+-----------+------+-------+------------+-----------+-------+
| Site Type | Used | Fixed | Prohibited | Available | Util% |
+-----------+------+-------+------------+-----------+-------+
| DSPs      |    0 |     0 |          0 |        90 |  0.00 |
+-----------+------+-------+------------+-----------+-------+
```

Figure 3: BRAM usage

5 Proof of working

After we ran the synthesis and implementation, we generated the bitstream to program the FPGA board.

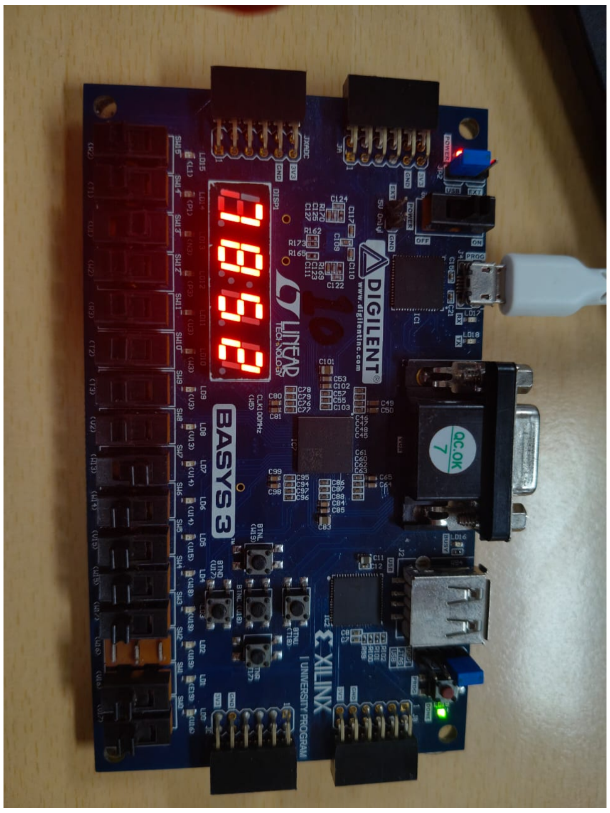
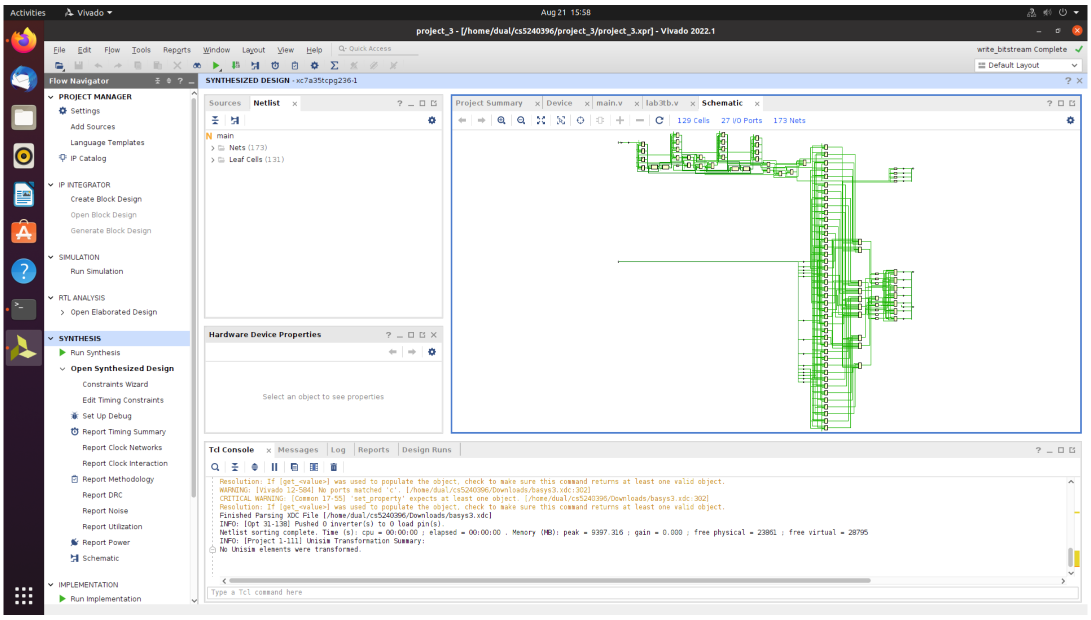


Figure 4: Snapshots showing simultaneous working of the 7-segment display for all anodes on the Basys3 board

6 Generated Schematics



```
Resolution: If [get_<value>] was used to populate the object, check to make sure this command returns at least one valid object.
WARNING: [Vivado 12-584] No ports matched 'c'. [/home/dual/cs5240396/Downloads/basys3.xdc:302]
CRITICAL WARNING: [Common 17-55] 'set_property' expects at least one object. [/home/dual/cs5240396/Downloads/basys3.xdc:302]
Resolution: If [get_<value>] was used to populate the object, check to make sure this command returns at least one valid object.
Finished Parsing XDC File [/home/dual/cs5240396/Downloads/basys3.xdc]
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 9397.316 ; gain = 0.000 ; free physical = 23861 ; free virtual = 28795
INFO: [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.
```

Figure 5: Generated Schematic design

7 Conclusion

This assignment involved designing a 4-digit input and display system on the Basys3 FPGA using Verilog. Switch inputs were captured into digit registers via clocked sampling, with selector switches determining the active digit. A circular multiplexing scheme refreshed each digit every 1 ms, ensuring a stable display. The design was verified in simulation and hardware, successfully demonstrating input capture, storage, and display. Overall, the project reinforced key concepts of synchronous control and hardware multiplexing, yielding a reliable and efficient solution.