ECE 3300L Lab 3 - Switch-to-7-Segment Display Interface o

By: Priyanka Ravinder and Raj Gokidi

California State Polytechnic University, Pomona

July 11th, 2025

Introduction

The primary objective of this laboratory is to design and implement a binary-to-hexadecimal decoder circuit. This system accepts a 4-bit binary value from the user and outputs the corresponding hexadecimal on a 7-segment display. The implementation is on a Nexys A7 FPGA and written in Verilog.

Verilog Code

Seg7_driver.v

```
`timescale 1ns / 1ps

module seg7_driver(
    input clk,
    input [15:0] SW,
    output [15:0] LED,
    output [6:0] Cnode,
    output dp,
    output [7:0] AN
);

    reg [19:0] tmp = 0;
    reg [3:0] digit;
    reg [6:0] raw_seg;
    wire [2:0] s;

    assign dp = 1'b1;
    assign LED = SW;
    assign s = tmp[17:15];

    always @(posedge clk)
        tmp <= tmp + 1;

    always @(*) begin
        case (s)
            3'd0: digit = SW[3:0];
            3'd1: digit = SW[7:4];
            3'd2: digit = SW[11:8];
            3'd3: digit = SW[15:12];
            default: digit = 4'b0000;
        endcase
    end
```

```
always @(*) begin
    case (digit)
        4'd0: raw_seg = 7'b0000001;
        4'd1: raw_seg = 7'b1001111;
        4'd2: raw_seg = 7'b0010010;
        4'd3: raw_seg = 7'b0000110;
        4'd4: raw_seg = 7'b1001100;
        4'd5: raw_seg = 7'b0100100;
        4'd6: raw_seg = 7'b0100000;
        4'd7: raw_seg = 7'b0001111;
        4'd8: raw_seg = 7'b0000000;
        4'd9: raw_seg = 7'b0001100;
        4'd10: raw_seg = 7'b0001000;
        4'd11: raw_seg = 7'b1100000;
        4'd12: raw_seg = 7'b0110001;
        4'd13: raw_seg = 7'b1000010;
        4'd14: raw_seg = 7'b0110000;
        4'd15: raw_seg = 7'b0111000;
        default: raw_seg = 7'b1111111;
    endcase
end

assign Cnode = {
    raw_seg[0],
    raw_seg[1],
    raw_seg[2],
    raw_seg[3],
    raw_seg[4],
    raw_seg[5],
    raw_seg[6]
};

reg [7:0] AN_tmp;
always @(*) begin
    case(s)
        3'd0: AN_tmp = 8'b11111110;
```

```
            3'd1: AN_tmp = 8'b11111101;
            3'd2: AN_tmp = 8'b11111011;
            3'd3: AN_tmp = 8'b11110111;
            default: AN_tmp = 8'b11111111;
        endcase
    end

    assign AN = AN_tmp;

endmodule
```

Top.v

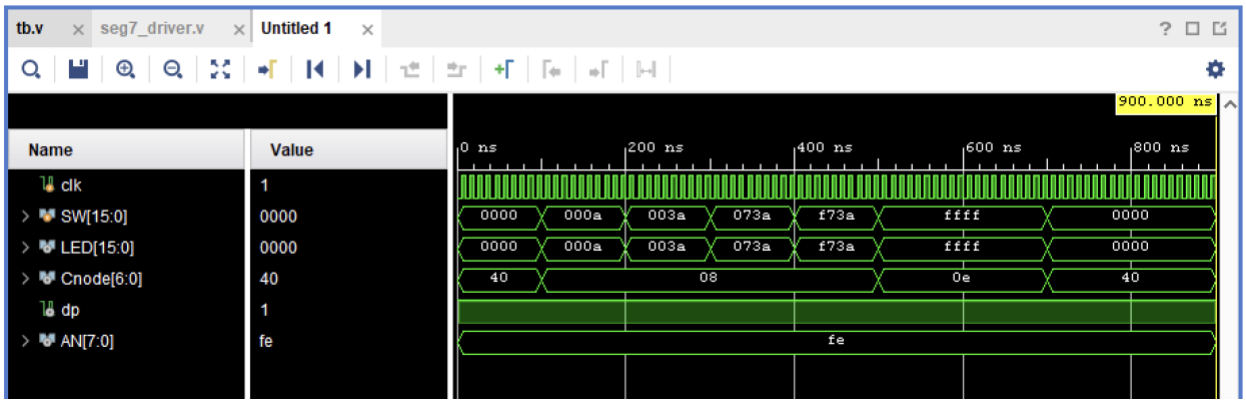
```
module top(
    input clk,
    input [15:0] SW,
    output [15:0] LED,
    output [6:0] Cnode,
    output dp,
    output [7:0] AN
);

    seg7_driver u0 (
        .clk(clk),
        .SW(SW),
        .LED(LED),
        .Cnode(Cnode),
        .dp(dp),
        .AN(AN)
    );

Endmodule
```

Seg7_driver_tb.v

```
`timescale 1ns / 1ps

module seg7_driver_tb;


    reg clk;
    reg [15:0] SW;


    wire [15:0] LED;
    wire [6:0] Cnode;
    wire dp;
    wire [7:0] AN;
```

```
    seg7_driver uut (
        .clk(clk),
        .SW(SW),
        .LED(LED),
        .Cnode(Cnode),
        .dp(dp),
        .AN(AN)
    );


    always #5 clk = ~clk;

    initial begin
        clk = 0;
        SW = 16'b0000_0000_0000_0000;

        #100;

        SW[3:0] = 4'hA;
        #100;

        SW[7:4] = 4'h3;
        #100;

        SW[11:8] = 4'h7;
        #100;

        SW[15:12] = 4'hF;
        #100;

        SW = 16'hFFFF;
        #200;

        SW = 16'h0000;
        #200;

        $finish;
    end

endmodule
```

Waveform



LUT and FF Utilizations

Resource	Utilization	Available	Utilization %
LUT	14	63400	0.02
FF	20	126800	0.02
IO	50	210	23.81

Contribution

Priyanka Ravinder: Implementation, demo, Verilog

Raj Gokidi: Test bench, waveform sim, report

Reflection

This lab provided valuable hands-on experience in translating digital logic theory into a functional hardware implementation. Through this process, a much clearer understanding of how hardware description languages control physical components like switches and displays was gained. The project was particularly helpful in demonstrating the direct connection between the Verilog code written and the resulting hardware behavior.

Youtube link: https://youtu.be/gPbV3EGgrZY