California Polytechnic State University, Pomona
Department of Electrical and Computer Engineering

Digital Circuit Design Using Verilog Laboratory
ECE 3300L

Lab 4

Switch-to-7-Segment Display Interface

By

Jetts Crittenden (ID# 015468128) and
Evan Tram(#ID 016404570)

7/11/2025

Design:

For this lab, it is important to look at how the 7-segment displays are addressed. There are 8 outputs for each element of the 7-segment and 8 different anodes. By changing which anode value, we can select which display we want to turn on. Since all of the data pins are shared between all of the displays, we have to rapidly shift through the anodes and change the values of the pins, giving the illusion that all of the 7-segments are displaying simultaneously. This is called multiplexing. Our design works by taking the switch inputs as individual bits that control separate numbers that vary from 0-15. We then route these values through the left and right displays to show their decimal and hexadecimal value. This is achieved by using a lookup table for the 10's place to convert to BCD. We then have a counter that lessens the frequency at which it is updated in order to set the refresh rate to a visible level. This counter then selects digits 0-7 depending on the position of the counter, which then displays the individual digits.

Code:

```
`timescale 1ns / 1ps
module seg7_driver(
    input clk,
    input rst_n,
    input [15:0] SW,
    output [15:0] LED,
    output reg [6:0] Cnode,
    output dp,
    output [7:0] AN
    );

    reg [19:0] tmp;
    wire [2:0] s;
    reg [3:0] digits [7:0];
    reg [7:0] AN_tmp;

    assign dp = 1'b1;
    assign LED = SW;
    assign s = tmp[19:17];
    assign AN = AN_tmp;
```

```
    // Lookup tables for decimal conversion
    reg [3:0] tens_lut [15:0];
    reg [3:0] units_lut [15:0];

    always @(posedge clk or posedge rst_n) begin
    if (rst_n) begin
        tens_lut[ 0] = 16; units_lut[ 0] = 0;
        tens_lut[ 1] = 16; units_lut[ 1] = 1;
        tens_lut[ 2] = 16; units_lut[ 2] = 2;
        tens_lut[ 3] = 16; units_lut[ 3] = 3;
        tens_lut[ 4] = 16; units_lut[ 4] = 4;
        tens_lut[ 5] = 16; units_lut[ 5] = 5;
        tens_lut[ 6] = 16; units_lut[ 6] = 6;
        tens_lut[ 7] = 16; units_lut[ 7] = 7;
        tens_lut[ 8] = 16; units_lut[ 8] = 8;
        tens_lut[ 9] = 16; units_lut[ 9] = 9;
        tens_lut[10] = 1; units_lut[10] = 0;
        tens_lut[11] = 1; units_lut[11] = 1;
        tens_lut[12] = 1; units_lut[12] = 2;
        tens_lut[13] = 1; units_lut[13] = 3;
        tens_lut[14] = 1; units_lut[14] = 4;
        tens_lut[15] = 1; units_lut[15] = 5;
    end
    end

    // Update counter
    always @(posedge clk or posedge rst_n)
        if (rst_n)
            tmp <= 0;
        else
            tmp <= tmp + 1;

    // Update digit values from SW input
    always @(posedge clk) begin
        digits[0] <= units_lut[SW[3:0]];
        digits[1] <= tens_lut[SW[3:0]];
        digits[2] <= units_lut[SW[7:4]];
        digits[3] <= tens_lut[SW[7:4]];
        digits[4] <= SW[3:0];
```

```
        digits[5] <= SW[7:4];
        digits[6] <= SW[11:8];
        digits[7] <= SW[15:12];
    end

    // Select active digit
    always @(posedge clk) 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;
            3'd4: AN_tmp <= 8'b11101111;
            3'd5: AN_tmp <= 8'b11011111;
            3'd6: AN_tmp <= 8'b10111111;
            3'd7: AN_tmp <= 8'b01111111;
            default: AN_tmp <= 8'b11111111;
        endcase
    end

    // 7-segment encoding
    always @(posedge clk) begin
        case (digits[s])
            4'd0:  Cnode <= 7'b1000000;
            4'd1:  Cnode <= 7'b1111001;
            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'b0011000;
            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
    end

endmodule
```

XDC Snippet:

Switches 0 - 15 were set to be used along with the anode and cathode pins of the 7-segment displays. 16 LEDS were set to be used for the switch indicators.

```
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 }
[get_ports { clk }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5}
[get_ports {clk}];
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 }
[get_ports { SW[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 }
[get_ports { SW[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 }
[get_ports { SW[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 }
[get_ports { SW[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 }
[get_ports { SW[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 }
[get_ports { SW[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 }
[get_ports { SW[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 }
[get_ports { SW[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 }
[get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 }
[get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 }
[get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 }
```

```
[get_ports { SW[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 }
[get_ports { SW[12] }]; #IO_L24P_T3_35 Sch=sw[12]
set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 }
[get_ports { SW[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 }
[get_ports { SW[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 }
[get_ports { SW[15] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]
## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 }
[get_ports { LED[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 }
[get_ports { LED[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 }
[get_ports { LED[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 }
[get_ports { LED[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 }
[get_ports { LED[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 }
[get_ports { LED[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 }
[get_ports { LED[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 }
[get_ports { LED[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 }
[get_ports { LED[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 }
[get_ports { LED[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 }
[get_ports { LED[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 }
[get_ports { LED[11] }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 }
[get_ports { LED[12] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 }
[get_ports { LED[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 }
```

```
[get_ports { LED[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 }
[get_ports { LED[15] }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]
##7 segment display
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 }
[get_ports { Cnode[0] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 }
[get_ports { Cnode[1] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 }
[get_ports { Cnode[2] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 }
[get_ports { Cnode[3] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 }
[get_ports { Cnode[4] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 }
[get_ports { Cnode[5] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 }
[get_ports { Cnode[6] }]; #IO_L4P_T0_D04_14 Sch=cg
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 }
[get_ports { dp }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 }
[get_ports { AN[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 }
[get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 }
[get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 }
[get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 }
[get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 }
[get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 }
[get_ports { AN[6] }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 }
[get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]
##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 }
[get_ports { rst_n }]; #IO_L9P_T1_DQS_14 Sch=btnc
```

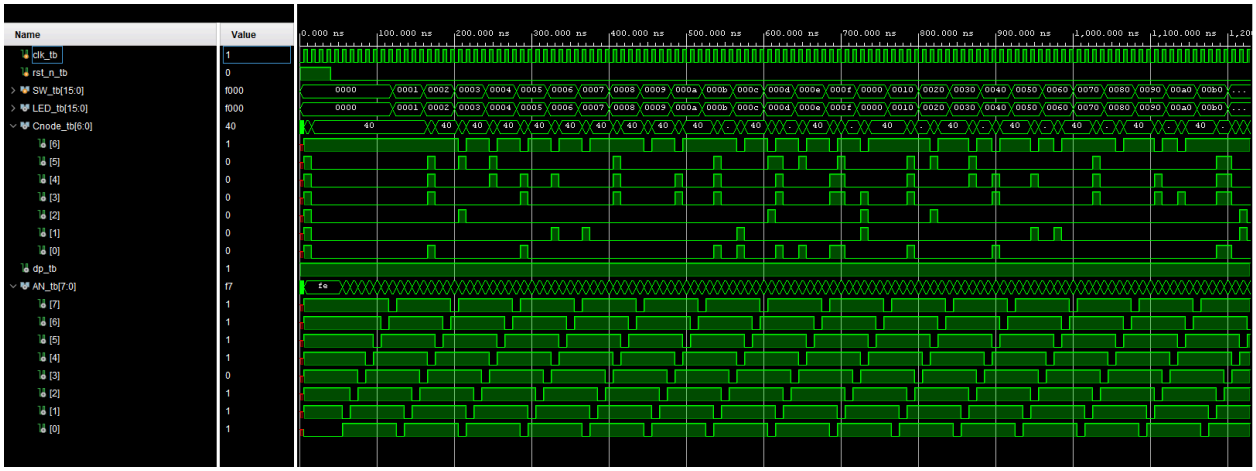
Simulation:

Test Bench Code:

```
`timescale 1ns / 1ps

module seg7_driver_tb(

    );
    reg clk_tb, rst_n_tb;
    reg [15:0] SW_tb;
    wire [15:0] LED_tb;
    wire [6:0] Cnode_tb;
    wire dp_tb;
    wire [7:0] AN_tb;

    seg7_driver tb(
        .clk(clk_tb),
        .rst_n(rst_n_tb),
        .SW(SW_tb),
        .LED(LED_tb),
        .Cnode(Cnode_tb),
        .dp(dp_tb),
        .AN(AN_tb)
    );

    initial clk_tb = 0;
    always #5 clk_tb = ~clk_tb;

    task test_group(input [3:0] value, input integer group);
        begin
            SW_tb = 16'b0;

            case(group)
                0: SW_tb[3:0] = value;
                1: SW_tb[7:4] = value;
                2: SW_tb[11:8] = value;
                3: SW_tb[15:12] = value;
            endcase

            #40;
```

```
            $display("SW[%0d:%0d] = %h, LED = %h, Cnode = %b, AN =
%b",
                group*4+3, group*4, value, LED_tb, Cnode_tb, AN_tb);

        end
    endtask

    integer w,x,y,z;
    initial
        begin
            SW_tb = 16'b0;
            rst_n_tb = 1;
            #40
            rst_n_tb = 0;
            #40

            for(w = 0; w < 16; w = w + 1)
                begin
                    test_group(w, 0);
                end
            for(x = 0; x < 16; x = x + 1)
                begin
                    test_group(x, 1);
                end

            for(y = 0; y < 16; y = y + 1)
                begin
                    test_group(y, 2);
                end

             for(z = 0; z < 16; z = z + 1)
                begin
                    test_group(z, 3);
                end
            $finish;
        end
endmodule
```

Sample Waveform:



In this simulation, the refresh rate is adjusted to be faster so it can be captured by the simulation window. You can see that the anodes rapidly turn on and off sequentially in order to display all of the decimal and hex values. In hardware, the refresh rate is much slower for it to be visible to the eye.

Implementation: Resource utilization table(s):

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	22	0	0	63400	0.03
LUT as Logic	22	0	0	63400	0.03
LUT as Memory	0	0	0	19000	0.00
Slice Registers	59	0	0	126800	0.05
Register as Flip Flop	59	0	0	126800	0.05
Register as Latch	0	0	0	126800	0.00
F7 Muxes	4	0	0	31700	0.01
F8 Muxes	0	0	0	15850	0.00

Look Up Tables Used:
22 total, all for logic
Memory LUTs: 0 used

Registers:
Used: 59 flip-flops
Latches: 0 used

Muxes:
F7 Muxes: 4 used
F8 Muxes: 0 used

Timing summary:

```
------------------------------------------------------------------------------------------------
| Design Timing Summary
| ---------------------
------------------------------------------------------------------------------------------------

    WNS(ns)      TNS(ns)  TNS Failing Endpoints  TNS Total Endpoints      WHS(ns)
    -------      -------  ---------------------  -------------------      -------
      6.045        0.000                      0                   43        0.253


------------------------

  WHS(ns)      THS(ns)  THS Failing Endpoints  THS Total Endpoints     WPWS(ns)
  -------      -------  ---------------------  -------------------     --------
    0.253        0.000                      0                   43        4.500
```

WNS (Worst Negative Slack): 6.045 ns
TNS (Total Negative Slack): 0.00 ns
WHS (Worst Hold Slack): 0.253 ns
PWWS (Worst Pulse Width Slack): 4.500 ns
Period: 10 ns - Worst Slack Under clock period

Group video link:

https://youtu.be/24z85Tas4oo?si=Gn3qJsFoKFkw4eSZ

Contributions:

Contributions on this lab were equivalent. Both lab members worked on the testbench and the main module. Much discussion was on the digit implementation and how to organize the display and debug the testbench timing. The demo video was recorded by Jetts Crittenden, and the testbench waveform was captured by Evan Tram. Both team members worked on the lab report.