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

Digital Circuit Design Using Verilog Laboratory
ECE 3300L

Lab 6

Dual BCD Up/Down Counters, ALU, and Control Display on 7-Segment

By

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

7/28/2025

Design:

This labs design is heavily based on the previous weeks design. However, this design implements two separate counters that are displayed via leds on the board, the values from these counters are then processed through as ALU in either subtraction or addition mode based on the users input. This output is then sent into the 7-segment driver that displays the results on the first two 7-segment digits, the third 7-segment digit is reserved for displaying the input logic for the processing. It displays the bit value in hex for the two direction switches and the two ALU processing switches. Just like lab 5, the clock speed for the counters is processed by the clock divider module and the 32x1 mux that allows the user to change the clock speed with the on-board switches.

Code:

top_lab6.v

```
`timescale 1ns / 1ps

module top_lab6(
    input wire CLK,
    input wire [4:0] SW, //Select
    input wire [1:0] ALU,
    input wire RST, // reset
    input wire dir0, // dir0
    input wire dir1, // dir1
    input wire DP,
    output wire [6:0] SEG,
    output wire [7:0] AN,
    output wire [7:0] LED
);

    wire rst_n = ~RST;

    wire [31:0] cnt;
    wire clk_out;
    wire [3:0] value0, value1;
    wire [7:0] result;
    wire [3:0] result_units, result_tens;

    wire [3:0] ControlNibble;
```

```
 assign DP = 0;
 assign LED[3:0] = value0;
 assign LED[7:4] = value1;

control_decoder DC(
     .Sw4(ALU[0]),
     .Sw3(ALU[1]),
     .Sw2(dir0),
     .Sw1(dir1),
     .ctrl_nibble(ControlNibble)
 );

 clock_divider u_clkdiv (
     .clk(CLK),
     .rst_n(rst_n),
     .cnt(cnt)
 );

 mux32x1 u_mux (
     .cnt(cnt),
     .sel(SW),
     .clk_out(clk_out)
 );

 bcd_up_down_counter BCD0_COUNTER (
     .clk_processed(clk_out),
     .rst_n(rst_n),
     .dir(dir0),
     .digit(value0)
 );

 bcd_up_down_counter BCD1_COUNTER (
     .clk_processed(clk_out),
     .rst_n(rst_n),
     .dir(dir1),
     .digit(value1)
 );
```

```
    binary_to_bcd u_bcd (
        .binary(result),
        .units(result_units),
        .tens(result_tens)
    );

    seg7_scan u_seg (
        .clk(CLK),
        .rst_n(rst_n),
        .onesPlace(result_units),
        .tensPlace(result_tens),
        .digitCtrl(ControlNibble),
        .SEG(SEG),
        .AN(AN)
    );

    alu ADDSUB(
        .A(value0),
        .B(value1),
        .ctrl(ALU),
        .out(result)
    );

endmodule
```

control_decoder.v

```
`timescale 1ns / 1ps
module control_decoder(
    input wire Sw4,
    input wire Sw3,
    input wire Sw2,
    input wire Sw1,
    output [3:0] ctrl_nibble
    );

    wire [3:0] nibble = {Sw1, Sw2, Sw3, Sw4};

    assign ctrl_nibble = nibble;
```

```
endmodule
```

clock_divider.v

```
`timescale 1ns / 1ps

module clock_divider(
    input wire clk,
    input wire rst_n,
    output reg [31:0] cnt
);

    always @(posedge clk or negedge rst_n)
        begin
            if (!rst_n)
                cnt <= 0;
            else
                cnt <= cnt + 1;
        end

endmodule
```

mux32x1.v

```
`timescale 1ns / 1ps

module mux32x1(
    input [31:0] cnt,
    input [4:0] sel,
    output clk_out
    );

    assign clk_out = cnt[31 - sel];

endmodule
```

bcd_up_down_counter.v

```
module bcd_up_down_counter(
    input wire clk_processed,
    input wire rst_n,
    input wire dir,
    output reg [3:0] digit
);

    always @(posedge clk_processed or negedge rst_n) begin
        if (!rst_n) begin
            digit <= 4'd0;
        end else begin
            if (dir) begin
                // Count up
                if (digit == 4'd9)
                    digit <= 4'd0;
                else
                    digit <= digit + 4'd1;
            end else begin
                // Count down
                if (digit == 4'd0)
                    digit <= 4'd9;
                else
                    digit <= digit - 4'd1;
            end
        end
    end

endmodule
```

binary_to_bcd.v

```
`timescale 1ns / 1ps

module binary_to_bcd(
    input wire [7:0] binary,
    output reg [3:0] units,
    output reg [3:0] tens
```

```
    );

    always @(*)
        begin
            case (binary)
                8'd0:  begin tens = 4'd0; units = 4'd0; end
                8'd1:  begin tens = 4'd0; units = 4'd1; end
                8'd2:  begin tens = 4'd0; units = 4'd2; end
                8'd3:  begin tens = 4'd0; units = 4'd3; end
                8'd4:  begin tens = 4'd0; units = 4'd4; end
                8'd5:  begin tens = 4'd0; units = 4'd5; end
                8'd6:  begin tens = 4'd0; units = 4'd6; end
                8'd7:  begin tens = 4'd0; units = 4'd7; end
                8'd8:  begin tens = 4'd0; units = 4'd8; end
                8'd9:  begin tens = 4'd0; units = 4'd9; end
                8'd10: begin tens = 4'd1; units = 4'd0; end
                8'd11: begin tens = 4'd1; units = 4'd1; end
                8'd12: begin tens = 4'd1; units = 4'd2; end
                8'd13: begin tens = 4'd1; units = 4'd3; end
                8'd14: begin tens = 4'd1; units = 4'd4; end
                8'd15: begin tens = 4'd1; units = 4'd5; end
                8'd16: begin tens = 4'd1; units = 4'd6; end
                8'd17: begin tens = 4'd1; units = 4'd7; end
                8'd18: begin tens = 4'd1; units = 4'd8; end
                default: begin tens = 4'd0; units = 4'd0; end
            endcase
        end

endmodule
```

seg7_scan.v

```
`timescale 1ns / 1ps

module seg7_scan(
    input wire clk,
    input wire rst_n,
    input wire [3:0] onesPlace,
    input wire [3:0] tensPlace,
```

```
    input wire [3:0] digitCtrl,
    output reg [6:0] SEG,
    output reg [7:0] AN
);

    reg [19:0] refresh_counter = 0;
    wire [1:0] sel;
    reg [3:0] digit_val;

    assign sel = refresh_counter[19:18];

    always @(posedge clk or negedge rst_n)
        begin
            if (!rst_n)
                refresh_counter <= 0;
            else
                refresh_counter <= refresh_counter + 1;
        end

    always @(*)
        begin
            case(sel)
                2'd0:
                    begin
                        AN = 8'b11111110;
                        digit_val = onesPlace;
                    end
                2'd1:
                    begin
                        AN = 8'b11111101;
                        digit_val = tensPlace;
                    end
                2'd2:
                    begin
                        AN = 8'b11111011;
                        digit_val = digitCtrl;
                    end
                default: AN = 8'b11111111;
            endcase
```

```
        end

    always@(*)
        begin
            case(digit_val)
                4'd0:  SEG = 7'b1000000; // 0
                4'd1:  SEG = 7'b1111001; // 1
                4'd2:  SEG = 7'b0100100; // 2
                4'd3:  SEG = 7'b0110000; // 3
                4'd4:  SEG = 7'b0011001; // 4
                4'd5:  SEG = 7'b0010010; // 5
                4'd6:  SEG = 7'b0000010; // 6
                4'd7:  SEG = 7'b1111000; // 7
                4'd8:  SEG = 7'b0000000; // 8
                4'd9:  SEG = 7'b0010000; // 9
                4'd10: SEG = 7'b0001000; // A
                4'd11: SEG = 7'b0000011; // b
                4'd12: SEG = 7'b1000110; // C
                4'd13: SEG = 7'b0100001; // d
                4'd14: SEG = 7'b0000110; // E
                4'd15: SEG = 7'b0001110; // F
    default: SEG = 7'b1111111; // Blank
endcase
        end

endmodule
```

alu.v

```
`timescale 1ns / 1ps
module alu(
input wire [4:0] A,
input wire [4:0] B,
input wire [1:0] ctrl,
output reg [7:0] out
    );
 always @(*) begin
        case (ctrl)
            2'b00: begin
```

```
                // ADD
                 out = A + B;
            end
            2'b01: begin
                // SUB with wraparound
                if (A >= B)
                    out = A - B;
                else
                   out  = (A + 4'd10) - B;
            end
            default: begin
                out = 8'd0;
            end
        endcase
    end

endmodule
```

XDC Snippet:

```
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 { ALU[0] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 }
[get_ports { ALU[1] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 }
[get_ports { dir0 }]; #IO_L5N_T0_D07_14 Sch=sw[7]
set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 }
[get_ports { dir1 }]; #IO_L24N_T3_34 Sch=sw[8]
## 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]

##7 segment display

set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 }
[get_ports { SEG[0] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 }
[get_ports { SEG[1] }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 }
[get_ports { SEG[2] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 }
[get_ports { SEG[3] }]; #IO_L17P_T2_A26_15 Sch=cd
```

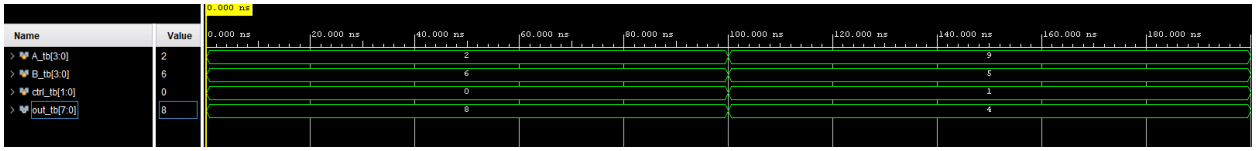
```
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 }
[get_ports { SEG[4] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 }
[get_ports { SEG[5] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 }
[get_ports { SEG[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 C12   IOSTANDARD LVCMOS33 }
[get_ports { CPU_RESETN }]; #IO_L3P_T0_DQS_AD1P_15 Sch=cpu_resetn

set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 }
[get_ports { RST }]; #IO_L9P_T1_DQS_14 Sch=btnc
```

Simulation:

Testbench code was withheld from report to not distract the report. All testbench code is found in the github repository.

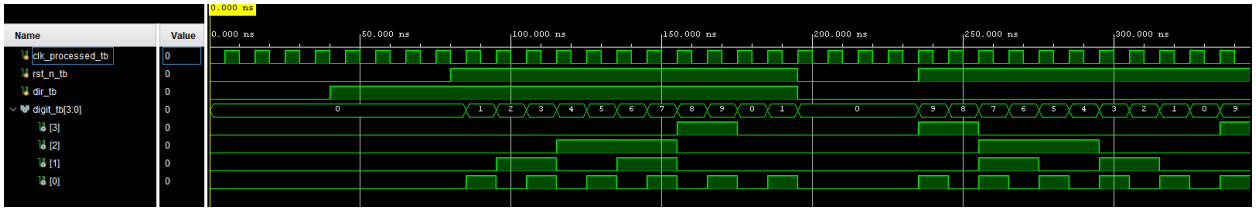
alu_tb.v

Simulation Waveform:



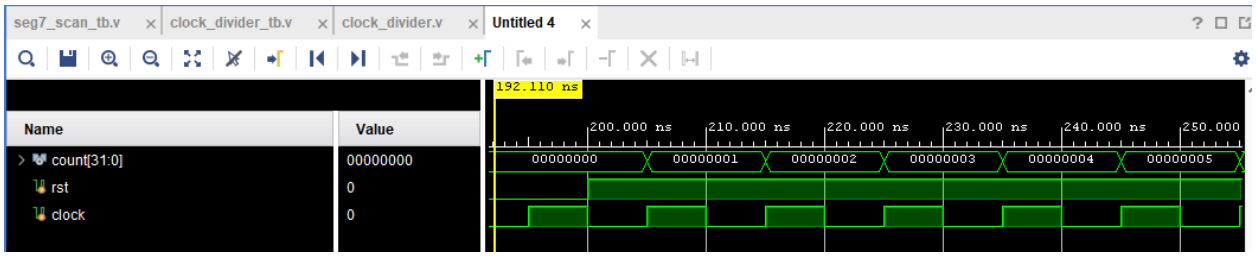
bcd_up_down_counter_tb.v

Simulation Waveform:



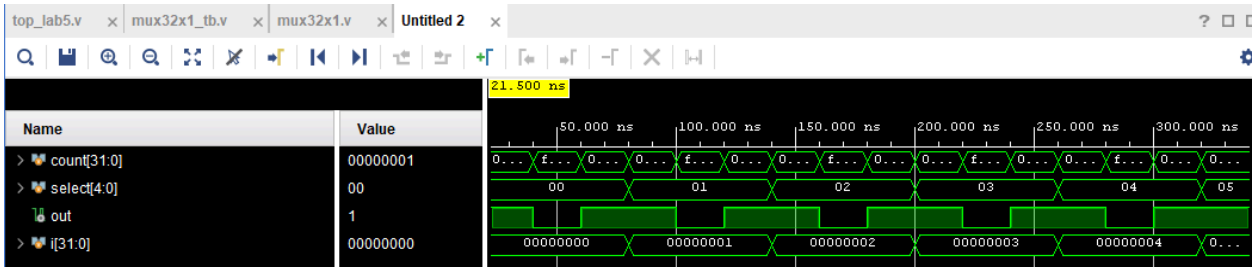
clock_divider_tb.v

Simulation Wavefrom:



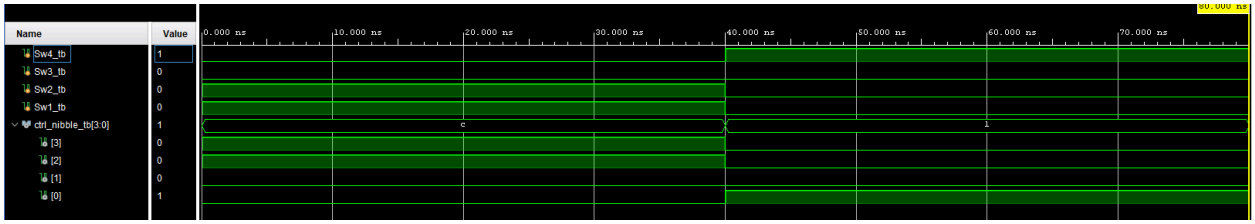
mux32x1_tb.v

Simulation Waveform:



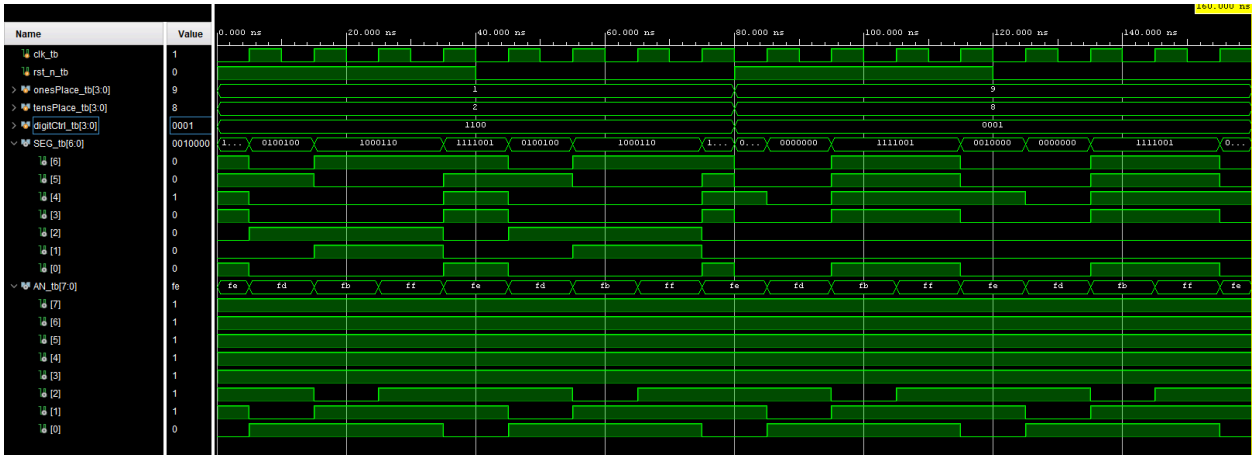
control_decorder_tb.v

Simulation Waveform:



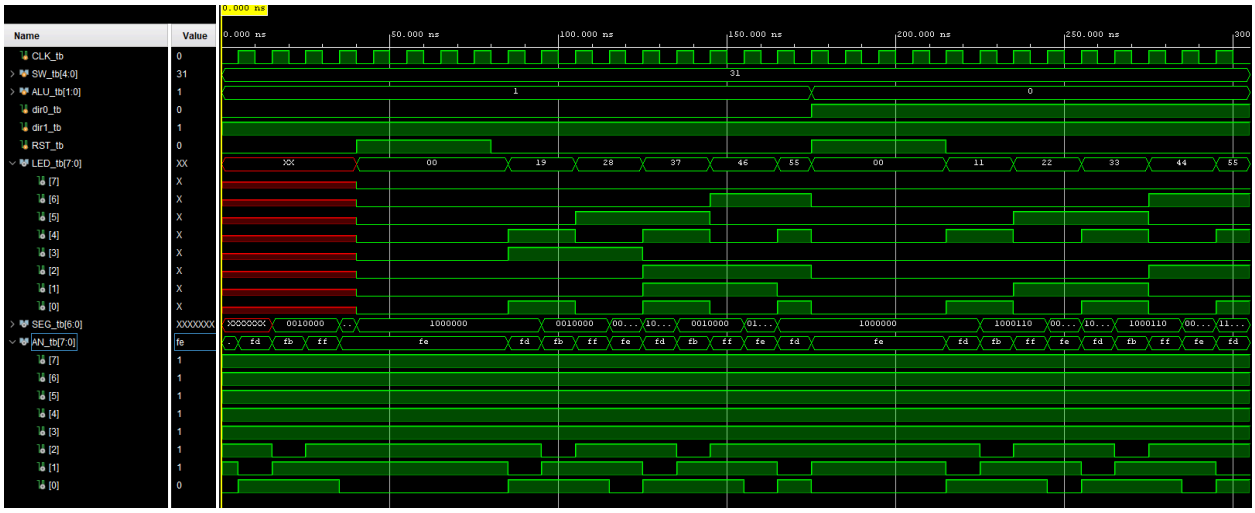
seg7_scan_tb.v

Simulation Waveform:



top_lab6_tb.v

Simulation Waveform:



Implementation: Resource utilization table(s):

Slice Logic Utilization:

LUTs Used: 55 out of 63,400 → 0.09%

Registers Used: 64 out of 126,800 → 0.05%: 60 FF

Muxes:

F7: 4 used → 0.01%

F8: 0 used → 0.00%

Registers:

All 64 registers have clock enable and asynchronous reset.

IO:

Bonded IOBs: 34 used out of 210 → 16.19%

All other IO-related resources 0.00%

Timing:

Worst Negative Slack: 6.851 ns; 52 end points

Worst Hold Slack: 0.213 ns; 52 end points

Worst Pulse Width Slack: 4.500 ns; 53 end points

Group video link:

https://youtu.be/tDmsNwrCuAU?si=6XTLR6KtyroO18NP

Contributions:

Most code for this project was derived from the previous lab which was a contribution of both Jetts Crittenden and Evan Tram. The remainder of the work for the verilog code was done in equal parts by both lab members where they collaborated and improved on eachothers work. Testbenches were developed and tested in the same way. The simulation screenshots were captured by Evan Tram while the implementation data and formatting was done by Jetts Crittenden. Equal work was shared between the two members.