Cal Poly Pomona

College of Engineering

California Polytechnic State University, Pomona

ECE3300L

Experiment #6

GROUP K
Hoang, Dalton - 016062800
Siu, Andy - 016205137

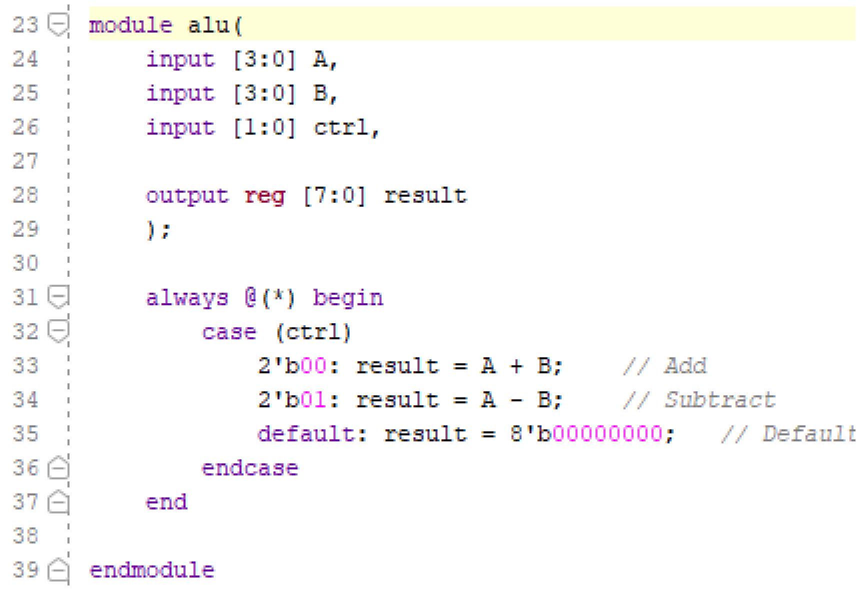
July 28th, 2025

Introduction:

In Lab 6, we designed and implemented a digital system on the Nexys A7-100T FPGA that integrates two BCD counters, an arithmetic logic unit (ALU), and a 7-segment display controller. The primary objective was to design an interactive counting system that allowed each counter to count up or down based on switch inputs. The outputs of these counters were then processed by a simple ALU capable of performing addition or subtraction. The final result was displayed on a 3-digit 7-segment display. The first two digits showed the ALU's result, while the third digit reflected the switch-controlled operational settings. Additional features included clock speed control using switches SW[4] through SW[0], direction control for each counter using switches SW[8] and SW[7], ALU operation selection using switches SW[5] and SW[6], and system-wide reset using BTN0 as the reset. LEDs were also used to monitor the raw binary output of the counters as a form of debugging. Overall, this lab provided hands-on experience in Verilog design and the integration of instantiated modules.

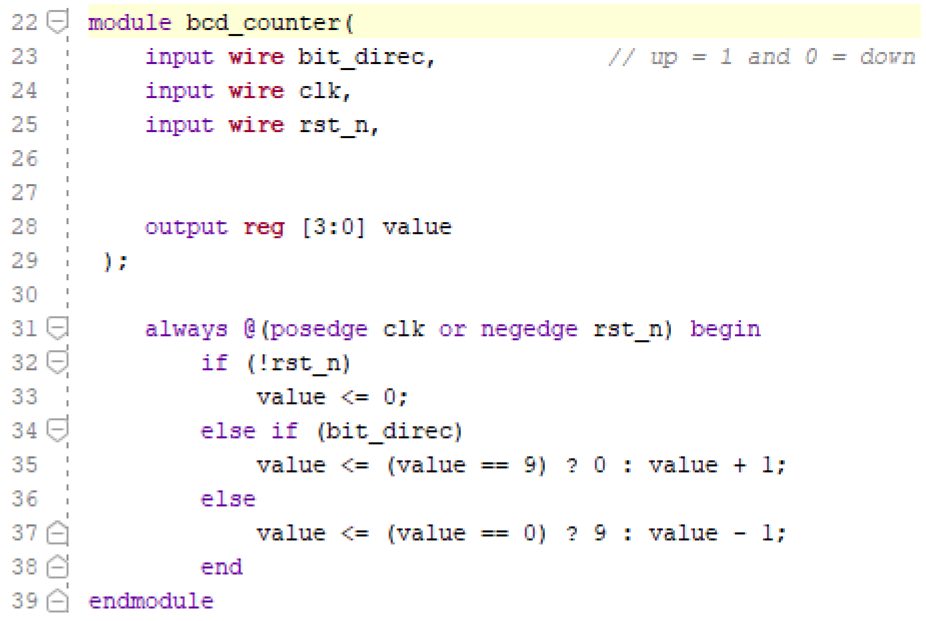
Design:

alu.v: implements an arithmetic logic unit that takes two 4-bit BCD inputs and a 2-bit control signal to perform either addition or subtraction



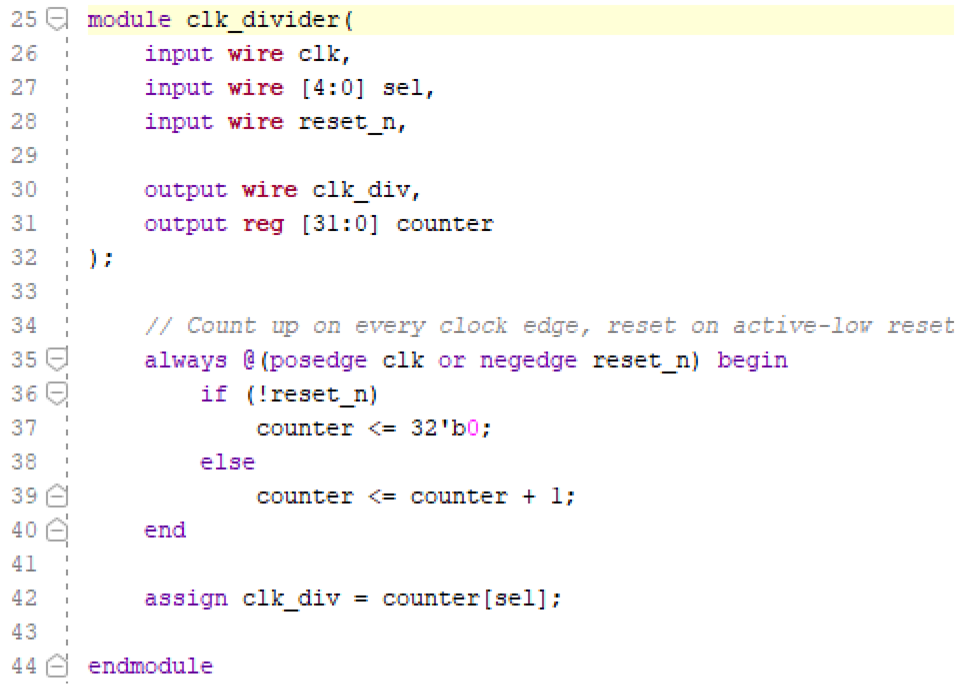
```
module alu(
    input [3:0] A,
    input [3:0] B,
    input [1:0] ctrl,

    output reg [7:0] result
    );

    always @(*) begin
        case (ctrl)
            2'b00: result = A + B;     // Add
            2'b01: result = A - B;     // Subtract
            default: result = 8'b00000000;   // Default
        endcase
    end

endmodule
```

bcd_counter.v: defines a BCD counter that increments or decrements a 4-bit value between 0 and 9 based on a direction control signal, and resets to 0 when the active-low reset is triggered.



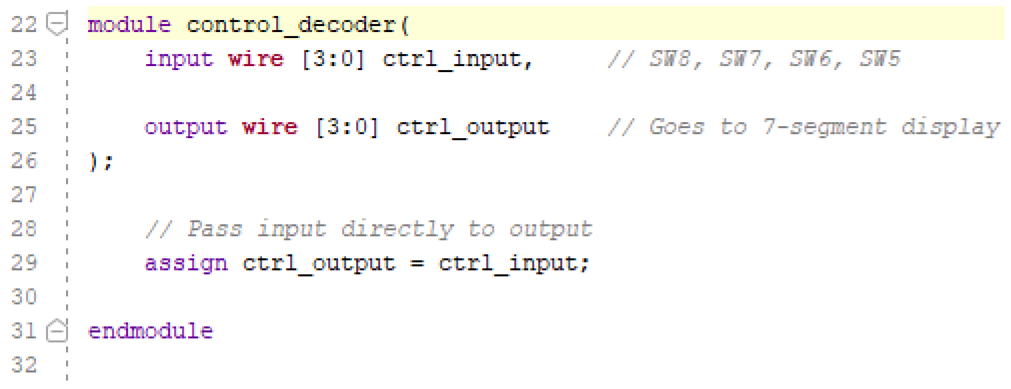
```verilog
module bcd_counter(
    input wire bit_direc,        // up = 1 and 0 = down
    input wire clk,
    input wire rst_n,


    output reg [3:0] value
 );

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n)
            value <= 0;
        else if (bit_direc)
            value <= (value == 9) ? 0 : value + 1;
        else
            value <= (value == 0) ? 9 : value - 1;
        end
endmodule
```

clk_divider.v: implements a clock divider that uses a 32-bit counter to generate a slower clock signal by selecting the MSB of the counter based on a 5-bit input select, with an active-low reset to zero the counter.



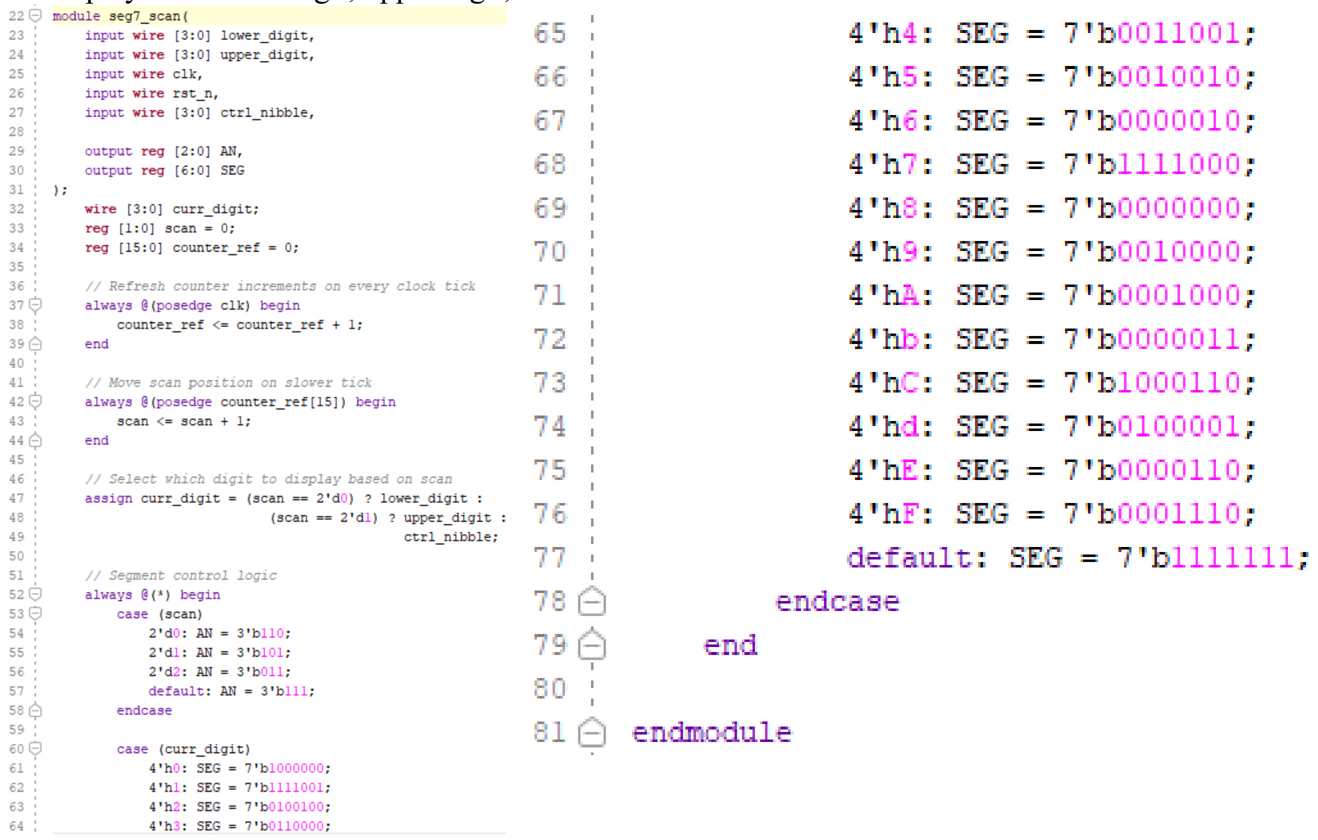
```verilog
module clk_divider(
    input wire clk,
    input wire [4:0] sel,
    input wire reset_n,

    output wire clk_div,
    output reg [31:0] counter
);

    // Count up on every clock edge, reset on active-low reset
    always @(posedge clk or negedge reset_n) begin
        if (!reset_n)
            counter <= 32'b0;
        else
            counter <= counter + 1;
    end

    assign clk_div = counter[sel];

endmodule
```

control_decoder.v: passes a 4-bit ctrl_in input directly to a 4-bit ctrl_out output, allowing switch settings to be displayed on a 7-segment display.



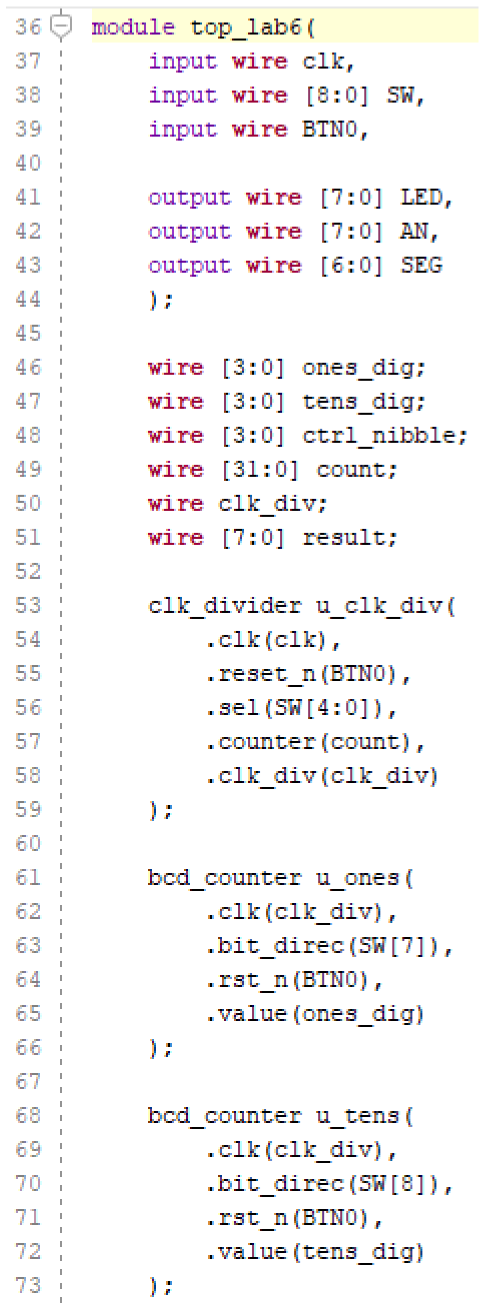
```verilog
module control_decoder(
    input wire [3:0] ctrl_input,     // SW8, SW7, SW6, SW5

    output wire [3:0] ctrl_output    // Goes to 7-segment display
);

    // Pass input directly to output
    assign ctrl_output = ctrl_input;

endmodule

```

seg7_scan.v: implements a 3-digit 7-segment display scan that cycles through and displays the lower digit, upper digit, and control nibble.

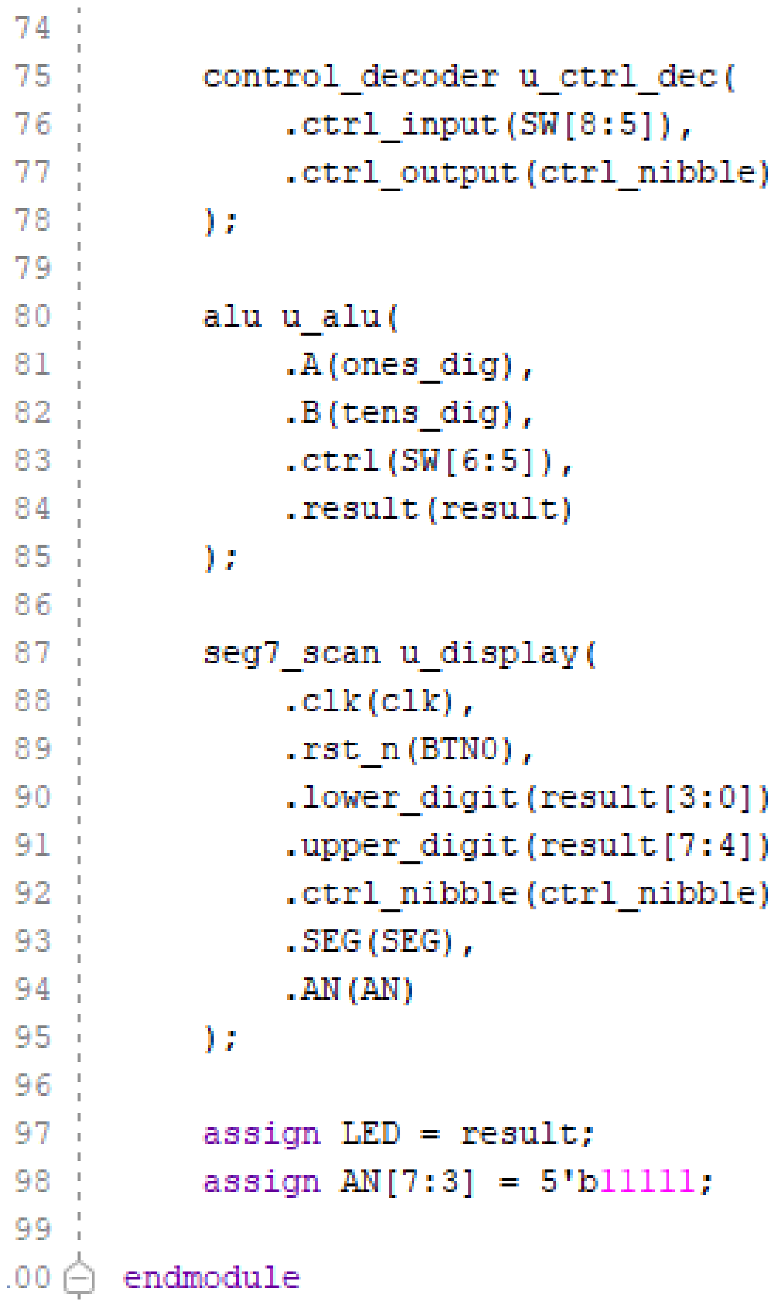


```verilog
module seg7_scan(
    input wire [3:0] lower_digit,
    input wire [3:0] upper_digit,
    input wire clk,
    input wire rst_n,
    input wire [3:0] ctrl_nibble,

    output reg [2:0] AN,
    output reg [6:0] SEG
);
    wire [3:0] curr_digit;
    reg [1:0] scan = 0;
    reg [15:0] counter_ref = 0;

    // Refresh counter increments on every clock tick
    always @(posedge clk) begin
        counter_ref <= counter_ref + 1;
    end

    // Move scan position on slower tick
    always @(posedge counter_ref[15]) begin
        scan <= scan + 1;
    end

    // Select which digit to display based on scan
    assign curr_digit = (scan == 2'd0) ? lower_digit :
                        (scan == 2'd1) ? upper_digit :
                                         ctrl_nibble;

    // Segment control logic
    always @(*) begin
        case (scan)
            2'd0: AN = 3'b110;
            2'd1: AN = 3'b101;
            2'd2: AN = 3'b011;
            default: AN = 3'b111;
        endcase

        case (curr_digit)
            4'h0: SEG = 7'b1000000;
            4'h1: SEG = 7'b1111001;
            4'h2: SEG = 7'b0100100;
            4'h3: SEG = 7'b0110000;
            4'h4: SEG = 7'b0011001;
            4'h5: SEG = 7'b0010010;
            4'h6: SEG = 7'b0000010;
            4'h7: SEG = 7'b1111000;
            4'h8: SEG = 7'b0000000;
            4'h9: SEG = 7'b0010000;
            4'hA: SEG = 7'b0001000;
            4'hb: SEG = 7'b0000011;
            4'hC: SEG = 7'b1000110;
            4'hd: SEG = 7'b0100001;
            4'hE: SEG = 7'b0000110;
            4'hF: SEG = 7'b0001110;
            default: SEG = 7'b1111111;
        endcase
    end

endmodule
```

top_lab6.v: connects all submodules to implement a hardware system that counts up/down in BCD, performs addition or subtraction based on switch settings, and displays the result and control state using a 3-digit 7-segment display and LEDs.



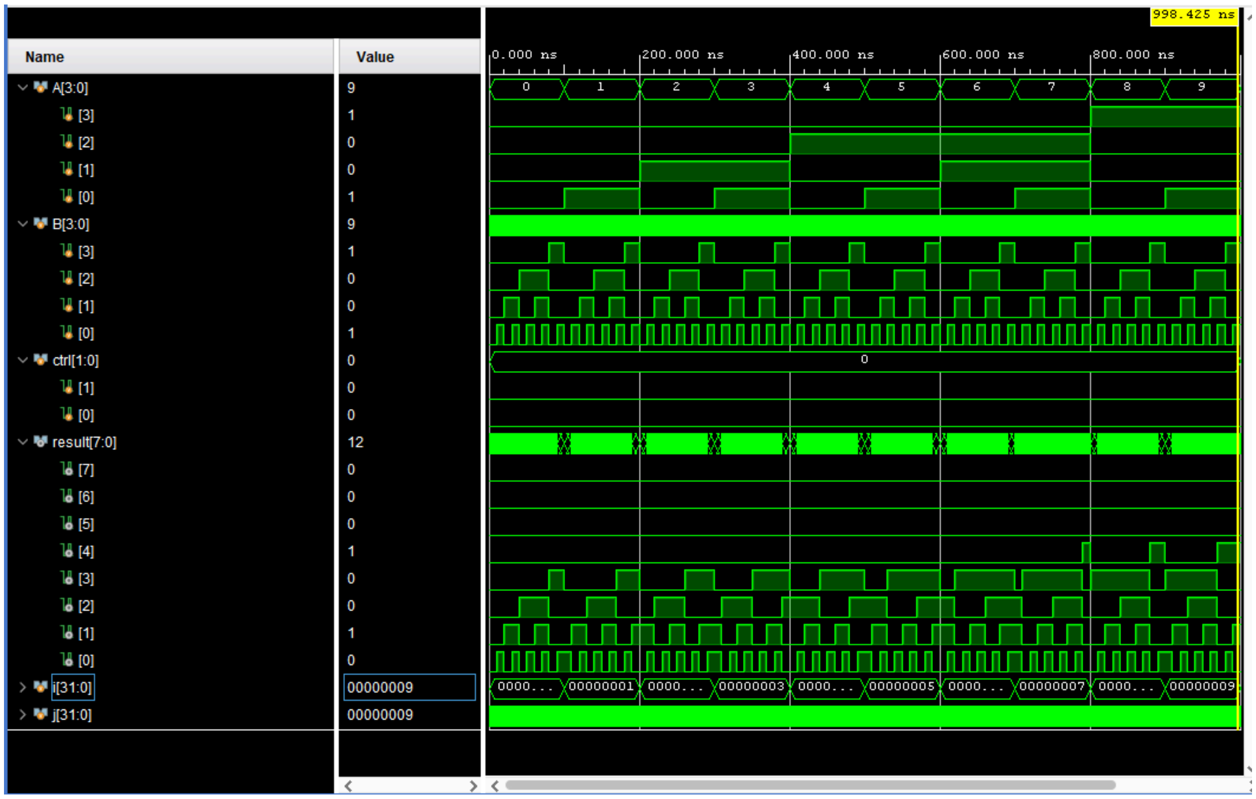
```verilog
module top_lab6(
    input wire clk,
    input wire [8:0] SW,
    input wire BTN0,

    output wire [7:0] LED,
    output wire [7:0] AN,
    output wire [6:0] SEG
    );

    wire [3:0] ones_dig;
    wire [3:0] tens_dig;
    wire [3:0] ctrl_nibble;
    wire [31:0] count;
    wire clk_div;
    wire [7:0] result;

    clk_divider u_clk_div(
        .clk(clk),
        .reset_n(BTN0),
        .sel(SW[4:0]),
        .counter(count),
        .clk_div(clk_div)
    );

    bcd_counter u_ones(
        .clk(clk_div),
        .bit_direc(SW[7]),
        .rst_n(BTN0),
        .value(ones_dig)
    );

    bcd_counter u_tens(
        .clk(clk_div),
        .bit_direc(SW[8]),
        .rst_n(BTN0),
        .value(tens_dig)
    );
```



```verilog

    control_decoder u_ctrl_dec(
        .ctrl_input(SW[8:5]),
        .ctrl_output(ctrl_nibble)
    );

    alu u_alu(
        .A(ones_dig),
        .B(tens_dig),
        .ctrl(SW[6:5]),
        .result(result)
    );

    seg7_scan u_display(
        .clk(clk),
        .rst_n(BTN0),
        .lower_digit(result[3:0])
        .upper_digit(result[7:4])
        .ctrl_nibble(ctrl_nibble)
        .SEG(SEG),
        .AN(AN)
    );

    assign LED = result;
    assign AN[7:3] = 5'b11111;

endmodule
```

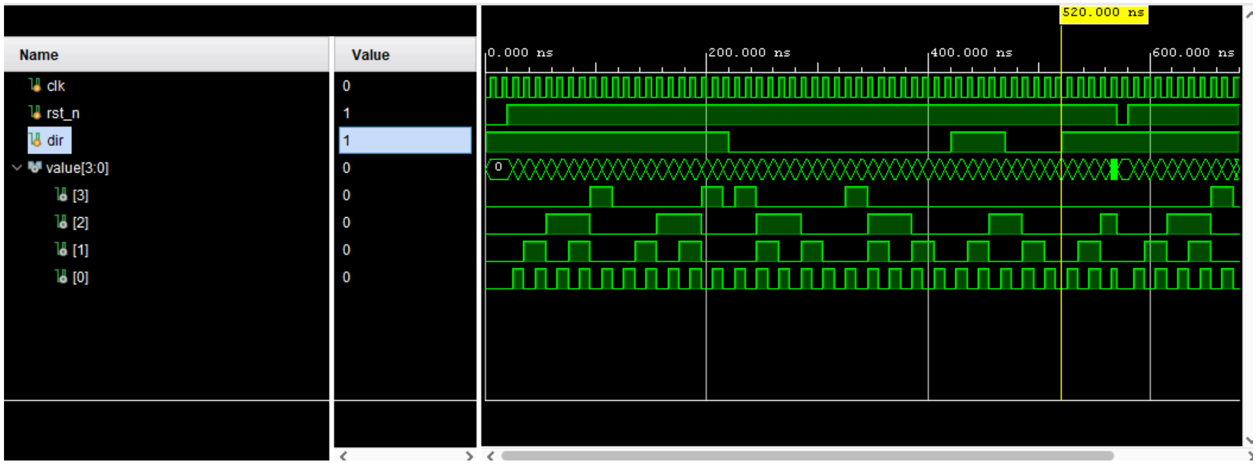
Simulation:

alu_tb.v: Adds and subtracts digits 0 - 9



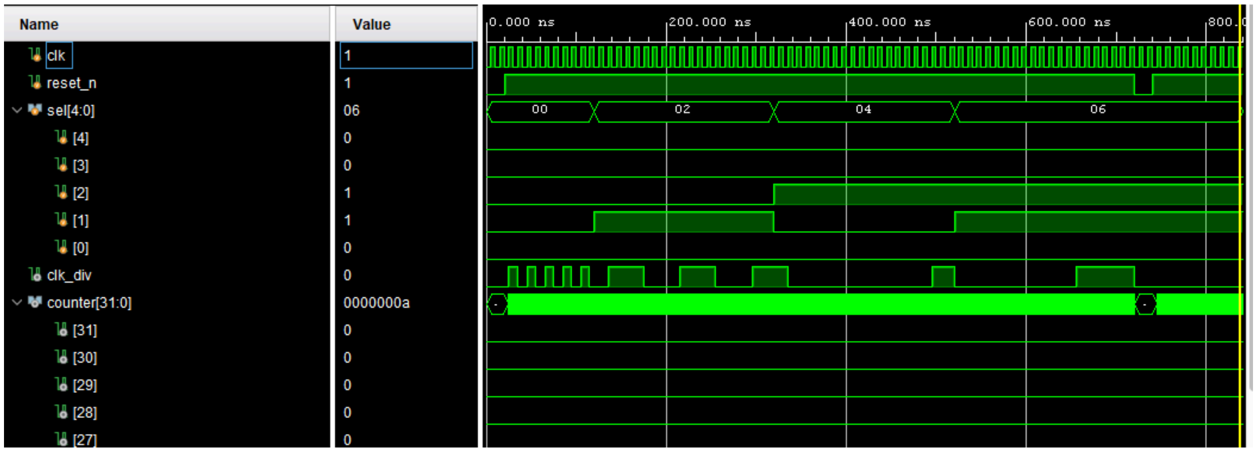
Name	Value
A[3:0]	9
[3]	1
[2]	0
[1]	0
[0]	1
B[3:0]	9
[3]	1
[2]	0
[1]	0
[0]	1
ctrl[1:0]	0
[1]	0
[0]	0
result[7:0]	12
[7]	0
[6]	0
[5]	0
[4]	1
[3]	0
[2]	0
[1]	1
[0]	0
i[31:0]	00000009
j[31:0]	00000009

bcd_counter_tb.v: counts up from 0 - 9 and back down 9 - 0

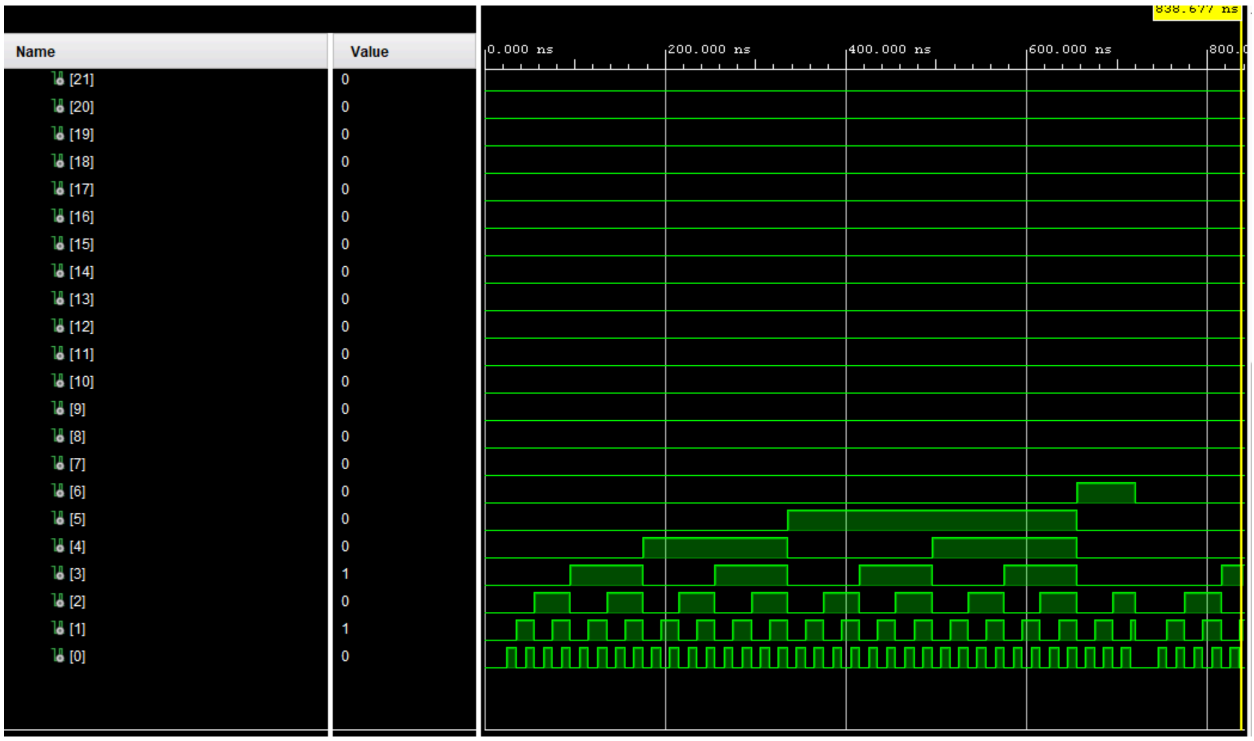


Name	Value
clk	0
rst_n	1
dir	1
value[3:0]	0
[3]	0
[2]	0
[1]	0
[0]	0

clk_divider_tb.v: Verify increment and toggle on rising edge

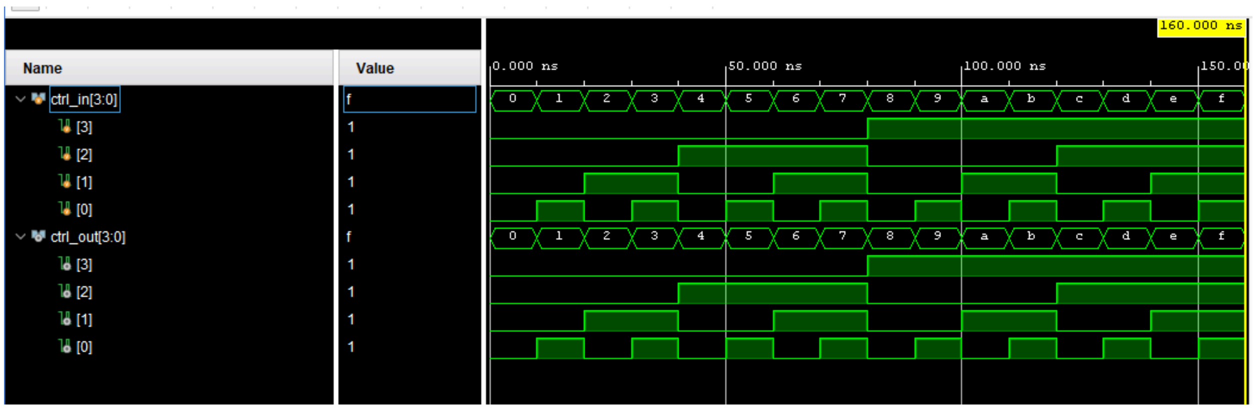


Name	Value
clk	1
reset_n	1
sel[4:0]	06
[4]	0
[3]	0
[2]	1
[1]	1
[0]	0
clk_div	0
counter[31:0]	0000000a
[31]	0
[30]	0
[29]	0
[28]	0
[27]	0

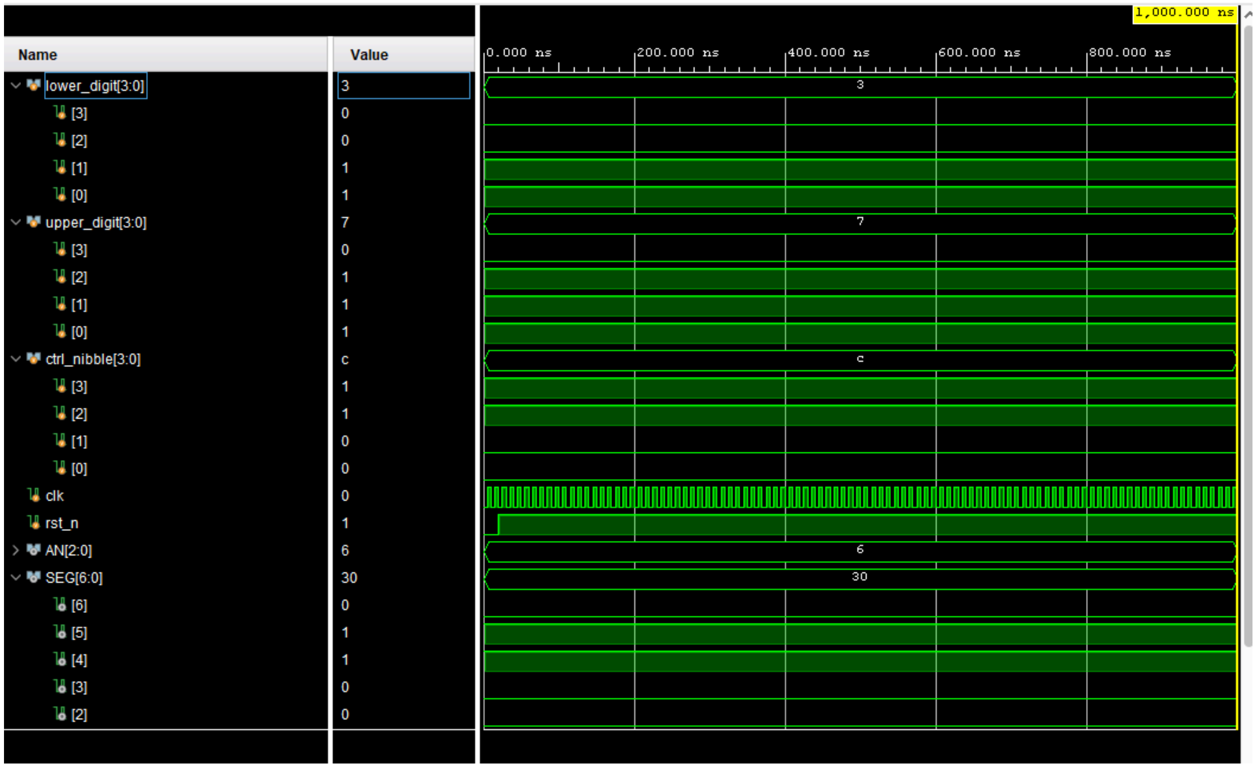


Name	Value
[21]	0
[20]	0
[19]	0
[18]	0
[17]	0
[16]	0
[15]	0
[14]	0
[13]	0
[12]	0
[11]	0
[10]	0
[9]	0
[8]	0
[7]	0
[6]	0
[5]	0
[4]	0
[3]	1
[2]	0
[1]	1
[0]	0

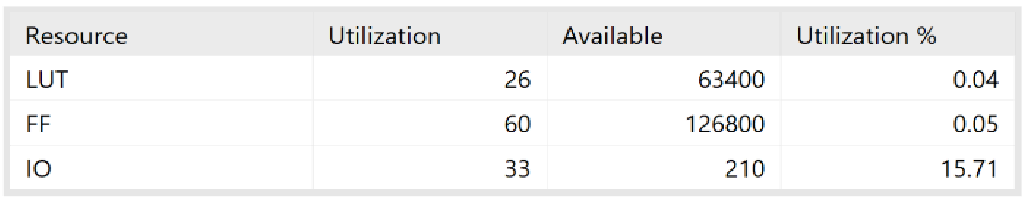
control_decoder_tb.v: Verify inputs = outputs



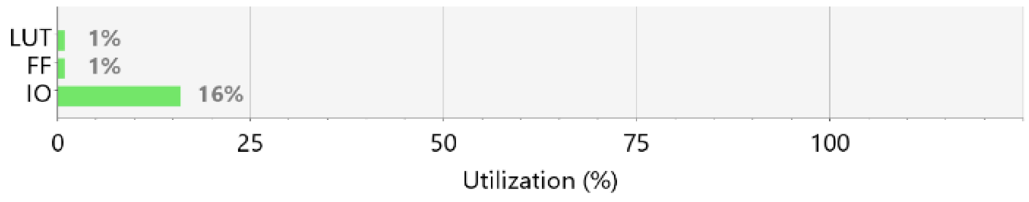
seg7_scan_tb.v: Correctly cycles through and displays the lower, upper, and control digits on a 3-digit 7-segment display

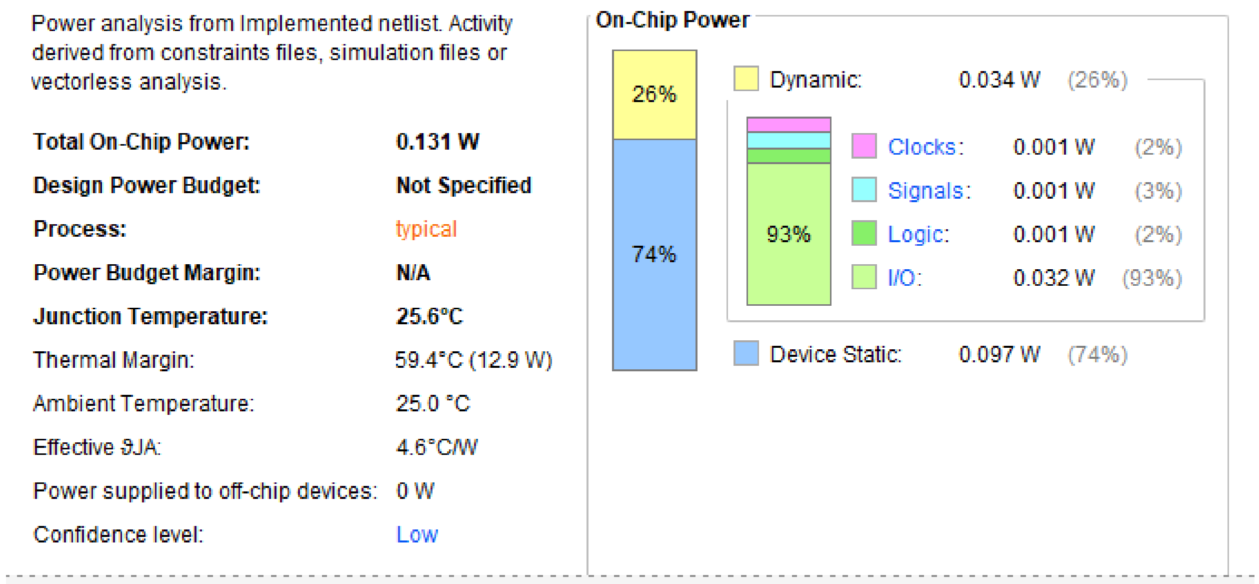


Implementation:



Resource	Utilization	Available	Utilization %
LUT	26	63400	0.04
FF	60	126800	0.05
IO	33	210	15.71

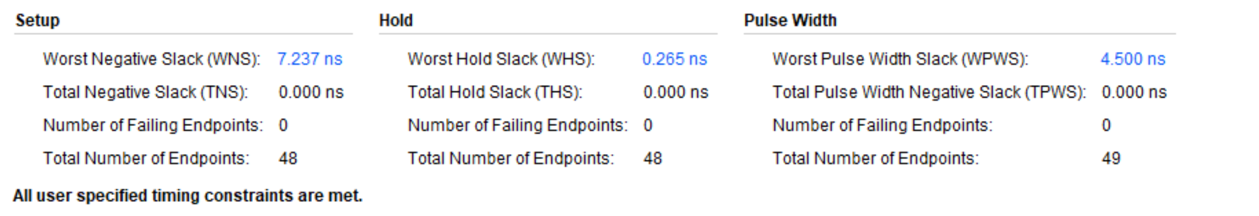



Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.131 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.6°C**
Thermal Margin:	59.4°C (12.9 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

On-Chip Power

🟨	Dynamic:	0.034 W	(26%)
🟪	Clocks:	0.001 W	(2%)
🟦	Signals:	0.001 W	(3%)
🟩	Logic:	0.001 W	(2%)
🟩	I/O:	0.032 W	(93%)
🟦	Device Static:	0.097 W	(74%)



Setup

Worst Negative Slack (WNS):	7.237 ns
Total Negative Slack (TNS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	48

Hold

Worst Hold Slack (WHS):	0.265 ns
Total Hold Slack (THS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	48

Pulse Width

Worst Pulse Width Slack (WPWS):	4.500 ns
Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	49

All user specified timing constraints are met.

Video Link: https://youtu.be/4higxxSDsZA

Contributions:

Andy Siu: (50%) Testbench, top, simulation, report, demo

Dalton Hoang: (50%) source files, top, report, implementation