



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

```
22 |
23 | module alu(
24 |     input [3:0] A,
25 |     input [3:0] B,
26 |     input [1:0] ctrl,
27 |     output reg [7:0] result
28 | );
29 |
30 | always @(*) begin
31 |     case (ctrl)
32 |         2'b00: result = A + B;    // Add
33 |         2'b01: result = A - B;    // Subtract
34 |         default: result = 8'b00000000; // Default
35 |     endcase
36 | end
37 |
38 | 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.

```

22 module bcd_counter(
23     input wire clk,
24     input wire rst_n,
25     input wire dir,          // up = 1 and 0 = down
26     output reg [3:0] value
27 );
28
29 always @(posedge clk or negedge rst_n) begin
30     if (!rst_n)
31         value <= 0;
32     else if (dir)
33         value <= (value == 9) ? 0 : value + 1;
34     else
35         value <= (value == 0) ? 9 : value - 1;
36     end
37 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.

```

24
25 module clk_divider(
26     input wire clk,
27     input wire [4:0] sel,
28     input wire reset_n,
29
30     output wire clk_div,
31     output reg [31:0] counter
32 );
33
34 // Count up on every clock edge, reset on active-low reset
35 always @(posedge clk or negedge reset_n) begin
36     if (!reset_n)
37         counter <= 32'b0;
38     else
39         counter <= counter + 1;
40     end
41
42     assign clk_div = counter[sel];
43
44 endmodule
45

```

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.

```

22 module control_decoder(
23     input wire [3:0] ctrl_in,    // SW8, SW7, SW6, SW5
24     output wire [3:0] ctrl_out  // Goes to 7-segment display
25 );
26
27     // Pass input directly to output
28     assign ctrl_out = ctrl_in;
29
30 endmodule
31

```

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

```

21 //////////////////////////////////////////////////
22 module seg7_scan(
23     input wire [3:0] lower_digit,
24     input wire [3:0] upper_digit,
25     input wire clk,
26     input wire rst_n,
27     input wire [3:0] ctrl_nibble,
28
29     output reg [2:0] AN,
30     output reg [6:0] SEG
31 );
32     wire [3:0] current_digit;
33     reg [1:0] scan = 0;
34     reg [15:0] refresh_counter = 0;
35
36     // Refresh counter increments on every clock tick
37     always @(posedge clk) begin
38         refresh_counter <= refresh_counter + 1;
39     end
40
41     // Advance scan position on slower tick
42     always @(posedge refresh_counter[15]) begin
43         scan <= scan + 1;
44     end
45
46     // Select which digit to display based on scan
47     assign current_digit = (scan == 2'd0) ? lower_digit :
48                             (scan == 2'd1) ? upper_digit :
49                             ctrl_nibble;
50
51     // Segment control logic
52     always @(*) begin
53         case (scan)
54             2'd0: AN = 3'b110;
55             2'd1: AN = 3'b101;
56             2'd2: AN = 3'b011;
57             default: AN = 3'b111;
58         endcase
59
60         case (current_digit)
61             4'h0: SEG = 7'b1000000;
62             4'h1: SEG = 7'b1111001;
63             4'h2: SEG = 7'b0100100;
64             4'h3: SEG = 7'b0110000;
65             4'h4: SEG = 7'b0011001;
66             4'h5: SEG = 7'b0010010;
67             4'h6: SEG = 7'b0000010;
68             4'h7: SEG = 7'b1111000;
69             4'h8: SEG = 7'b0000000;
70             4'h9: SEG = 7'b0010000;
71             4'hA: SEG = 7'b0001000;
72             4'hb: SEG = 7'b0000011;
73             4'hC: SEG = 7'b1000110;
74             4'hd: SEG = 7'b0100001;
75             4'hE: SEG = 7'b0000110;
76             4'hF: SEG = 7'b0001110;
77             default: SEG = 7'b1111111;
78         endcase
79     end
80
81 endmodule
82

```

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.

```

35 :
36 module top_lab6(
37     input wire clk,
38     input wire [8:0] SW,
39     input wire BTNO,
40
41     output wire [7:0] LED,
42     output wire [7:0] AN,
43     output wire [6:0] SEG
44 );
45
46 wire [3:0] ones;
47 wire [3:0] tens;
48 wire [3:0] ctrl_nibble;
49 wire [31:0] count;
50 wire clk_div;
51 wire [7:0] result;
52
53 clk_divider u_clk_div(
54     .clk(clk),
55     .reset_n(BTNO),
56     .sel(SW[4:0]),
57     .counter(count),
58     .clk_div(clk_div)
59 );
60
61 bcd_counter u_ones(
62     .clk(clk_div),
63     .dir(SW[7]),
64     .rst_n(BTNO),
65     .value(ones)
66 );
67
68 bcd_counter u_tens(
69     .clk(clk_div),
70     .dir(SW[8]),
71     .rst_n(BTNO),
72     .value(tens)
73 );
74
75 control_decoder u_ctrl_dec(
76     .ctrl_in(SW[8:5]),
77     .ctrl_out(ctrl_nibble)
78 );
79
80 alu u_alu(
81     .A(ones),
82     .B(tens),
83     .ctrl(SW[6:5]),
84     .result(result)
85 );
86
87 seg7_scan u_display(
88     .clk(clk),
89     .rst_n(BTNO),
90     .lower_digit(result[3:0]),
91     .upper_digit(result[7:4]),
92     .ctrl_nibble(ctrl_nibble),
93     .seg(SEG),
94     .an(AN)
95 );
96
97 assign LED = result;
98 assign AN[7:3] = 5'b11111;
99
100 endmodule

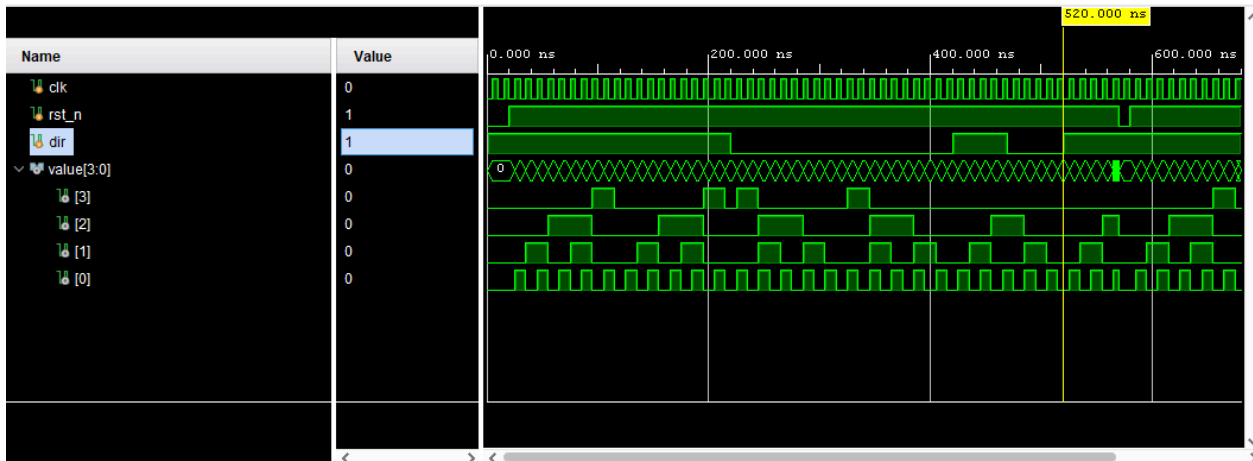
```

Simulation:

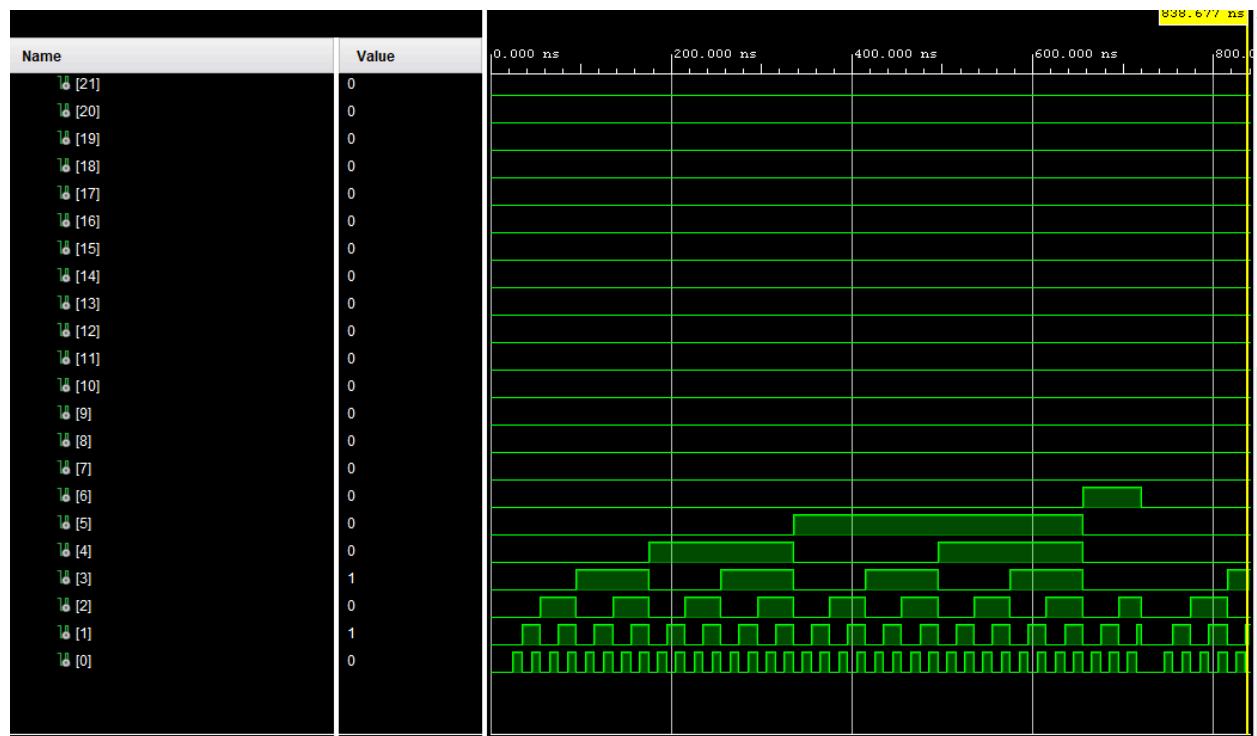
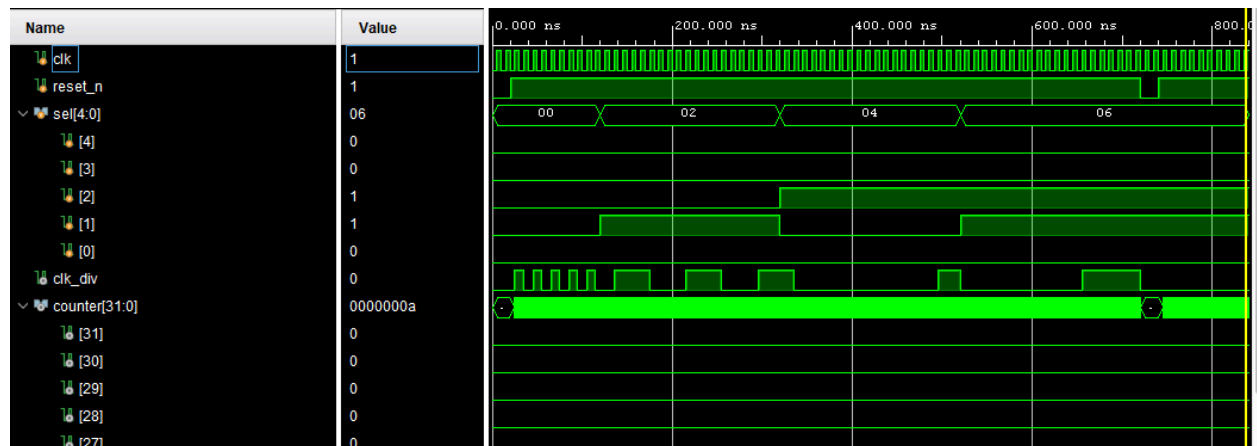
alu_tb.v: Adds and subtracts digits 0 - 9



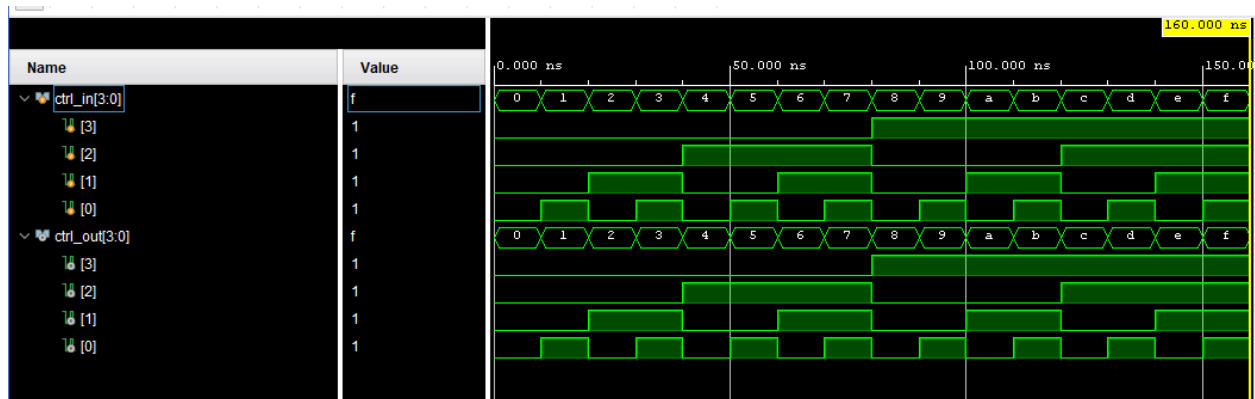
`bcd_counter_tb.v`: counts up from 0 - 9 and back down 9 - 0



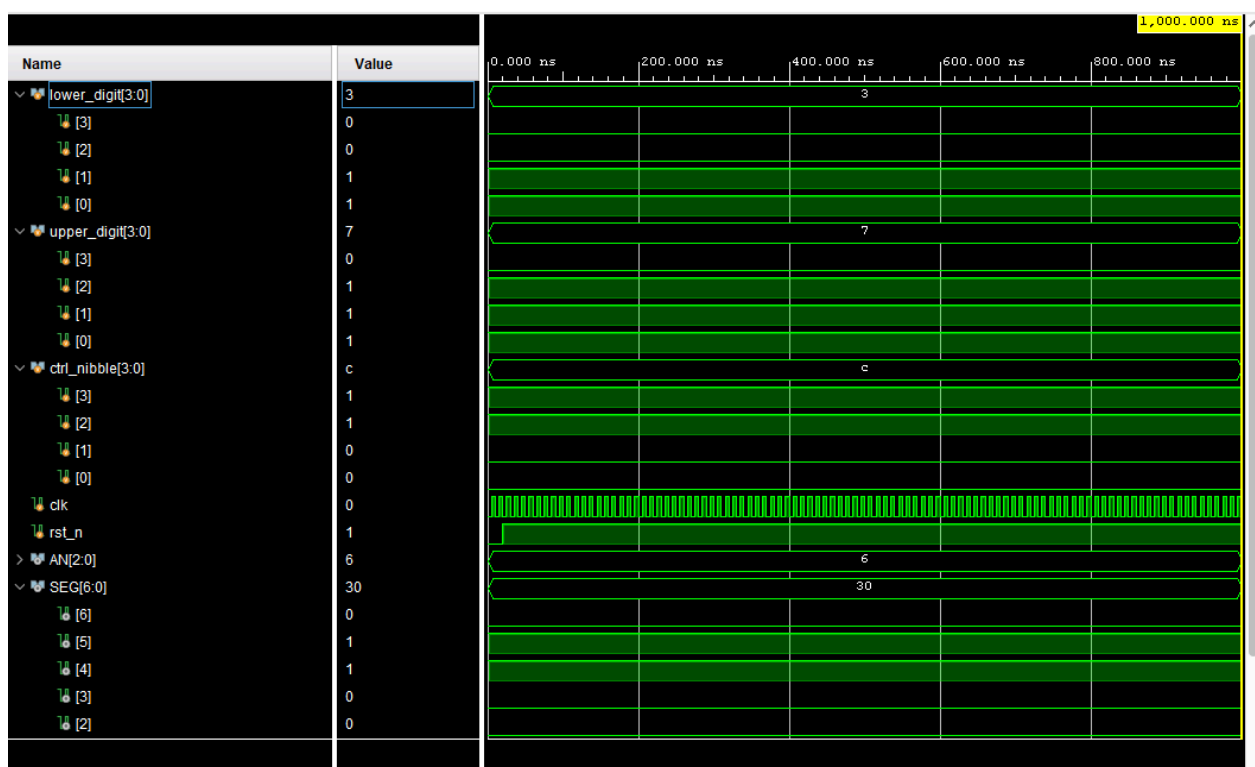
clk_divider_tb.v: Verify increment and toggle on rising edge



control_decoder_tb.v: Verify inputs = outputs

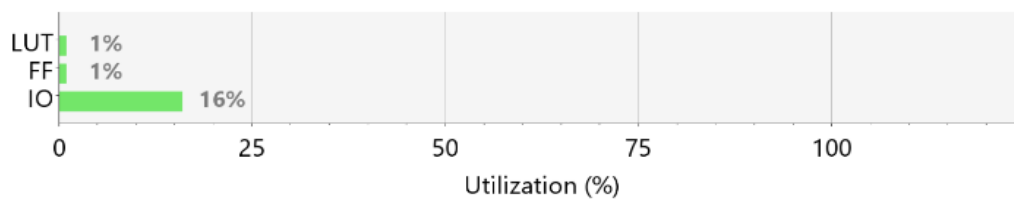


`seg7_scan_tb.v`: Correctly cycles through and displays the lower, upper, and control digits on a 3-digit 7-segment display



Implementation:

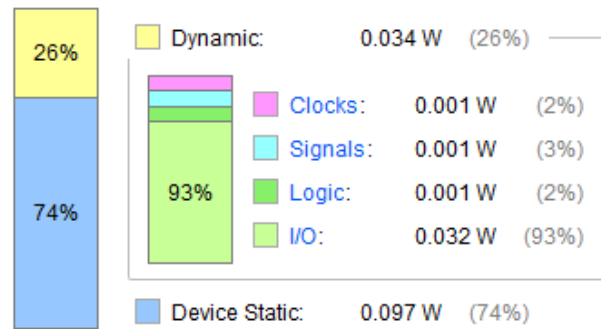
Resource	Utilization	Available	Utilization %
LUT	26	63400	0.04
FF	60	126800	0.05
IO	33	210	15.71



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 θ_{JA} : 4.6°C/W
Power supplied to off-chip devices: 0 W
Confidence level: Low

On-Chip Power



Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 7.237 ns	Worst Hold Slack (WHS): 0.265 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 48	Total Number of Endpoints: 48	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