ECE 3300L.01 - Lab 6

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

Professor Mohamed Aly
Group Q: Kevin Tang(015429622) & Jared Mocling(015215057)

July 28th, 2025

Design:

clock_divider.v

```
module clock_divider(
   input clockIn,
   input [4:0] sel, // speed selector this uses SW[4:0]
   input rst_n, // active-low resert
   output clk_div

   );

   reg [31:0] counter;

   always @(posedge clockIn or negedge rst_n) begin
      if (!rst_n)
         counter <= 32'b0;     // Reset the counter
      else
         counter <= counter + 1;   // Increment on each rising clock edge
   end

  assign clk_div = counter[31 - sel]; // without 31 - : this will invert the physical switches

endmodule
```

bcd_counter.v

```
module bcd_counter(
   input clk_div,
   input rst_n,           // Active-low reset (BTN0)
   input dir_bit,         // 1 = count up, 0 = count down (SW [8:7])
   output reg [3:0] digit // Ones digit in BCD


   );


   always@(posedge clk_div or negedge rst_n) begin

      if (!rst_n) begin
         digit <= 4'b0 ;
         end
      else begin
```

```
          if (!dir_bit) begin
            if (digit == 4'd9) begin
            digit <= 4'd0;

            end
            else
            digit <= digit + 1;
            end
          else begin
            if (digit == 4'd0) begin
            digit <= 4'd9;

            end
            else
            digit <= digit - 1;
            end

         end
        end

endmodule
```

<u>alu.v</u>

```
`timescale 1ns / 1ps
module alu(
  input [3:0] A, //ones
    input [3:0] B,  // tens
    input [1:0] ctrl, // SW[6:5]
    output reg [7:0] result
    );
    reg [4:0] computation;

    always @(*) begin
    if (ctrl == 2'b00)
        computation = A + B; // addition
    else if (ctrl == 2'b01)
        computation = A - B; // subtraction
    else
        computation = 8'd0;

        result[7:5] = 0;
        result[4] = computation / 4'd10;
        result[3:0] = computation % 4'd10;
end


endmodule
```

control_decoder.v

```
module control_decoder(
  input [3:0] controlSW, // {SW [8-5]}
    output reg [3:0] ctrl_nibble
    );

    always@(*)begin
        ctrl_nibble = controlSW;
    end

endmodule
```

seg7_scan.v

```
module seg7_scan(
    input CLK,                  // clock (used to multiplex)
    input rst_n,                // Active-low reset
    input [11:0] bits,
    output reg [6:0] SEG,       // Segment outputs (active low)
    output [7:0] AN,            // 1 bit for each display, Digit select lines (active low)
    output DP
);
    assign DP = 1'b1; // turns off decimal point

    reg [3:0] digit;
    // Based on digit, the 7 segment will create an output between 0 - F
    always@(digit)
    case(digit)
             4'd0: SEG =7'b0000001;
            4'd1: SEG =7'b1001111;
            4'd2: SEG =7'b0010010;
            4'd3: SEG =7'b0000110;
            4'd4: SEG =7'b1001100;
            4'd5: SEG =7'b0100100;
            4'd6: SEG =7'b0100000;
```

```
          4'd7: SEG =7'b0001111;
          4'd8: SEG =7'b0000000;
          4'd9: SEG =7'b0001100;
          4'd10: SEG =7'b0001000; //A
          4'd11: SEG =7'b1100000; //b
          4'd12: SEG =7'b0110001; //c
          4'd13: SEG =7'b1000010; //d
          4'd14: SEG =7'b0110000; //E
          4'd15: SEG =7'b0111000; //F
          default: SEG = 7'b1111111;
   endcase

   reg [20:0] tmp;
   always@(posedge CLK or negedge rst_n)
   if(!rst_n) tmp<=0;
   else tmp<=tmp+1;

   wire [1:0] s = tmp[18:17];



   always@(s, bits)
   case (s)
      2'd0:digit=bits[3:0]; 2'd1:digit=bits[7:4];
      2'd2:digit=bits[11:8];
      default:digit=4'b0000;
  endcase


   reg [7:0] AN_tmp;
   always@(s)
   case(s)
     2'd0:AN_tmp=8'b11111110; 2'd1:AN_tmp=8'b11111101;
     2'd2:AN_tmp=8'b11111011;
     default:AN_tmp=8'b11111111;
  endcase

assign AN=AN_tmp;
```

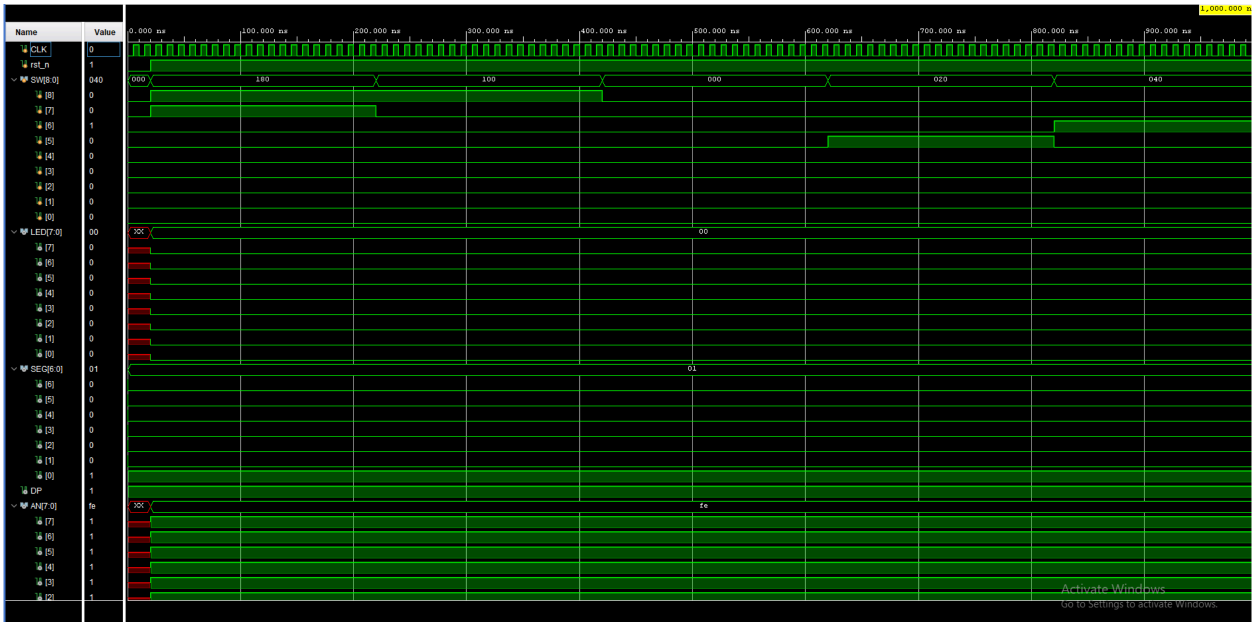
```
endmodule
```

<u>top_lab6.v</u>

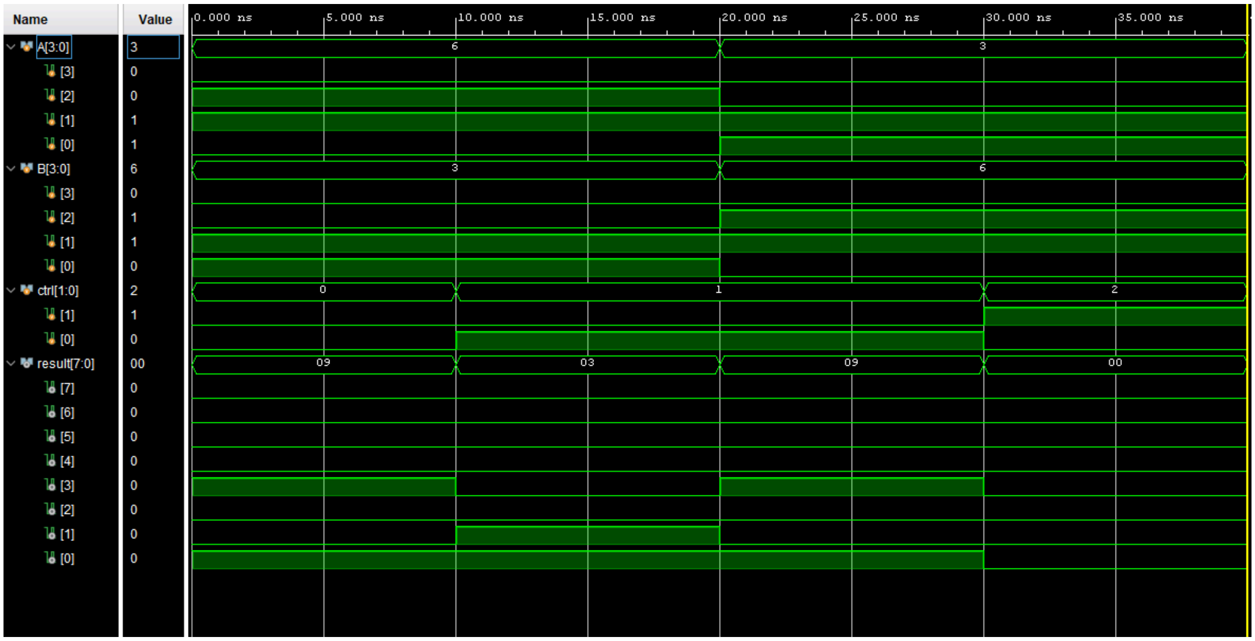
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 07/28/2025 09:34:25 PM
// Design Name:
// Module Name:
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module top_lab6(
    input CLK,
    input rst_n, //BTN0 in the XDC file, the !rst_n input will make the button ACTIVE-LOW
reset

    // SW[4:0] controls the speed
    // SW[6:5] controls the ALU bits
    //SW[8:7] controls the counter direction
    input [8:0] SW,

  // LED[7:4] debug for the tens BCD counter
  // LED[3:0]  debug for the units BCD counter
    output [7:0] LED,
    output [6:0] SEG, // for the segments
    output DP, // decimal point
    output [7:0] AN //only have 3 displays showing so AN [2:0], 1 bit for each display
    );
    wire clk_out;
    wire [3:0] digit0;
    wire [3:0] digit1;
    wire [7:0] finalResult;
    wire [3:0] ctrl_nibble;
    wire [11:0] bits;
    assign bits[11:0] = { ctrl_nibble[3:0], finalResult[7:0] };
    assign LED = {digit1[3:0], digit0[3:0]};

    clock_divider clk1 ( .clockIn(CLK), .sel(SW[4:0]), .rst_n(!rst_n), .clk_div(clk_out) );

    bcd_counter units ( .clk_div(clk_out), .rst_n(!rst_n), .dir_bit(SW[7]), .digit(digit0) );
    bcd_counter tens ( .clk_div(clk_out), .rst_n(!rst_n), .dir_bit(SW[8]), .digit(digit1) );

    alu addOrSubtract ( .A(digit0), .B(digit1), .ctrl(SW[6:5]), .result(finalResult) );

    control_decoder control ( .controlSW(SW[8:5]), .ctrl_nibble(ctrl_nibble) );

    seg7_scan display ( .CLK(CLK), .rst_n(!rst_n), .bits(bits), .SEG(SEG), .AN(AN),
.DP(DP) );


endmodule
```

<u>Testbench Waveform:</u>

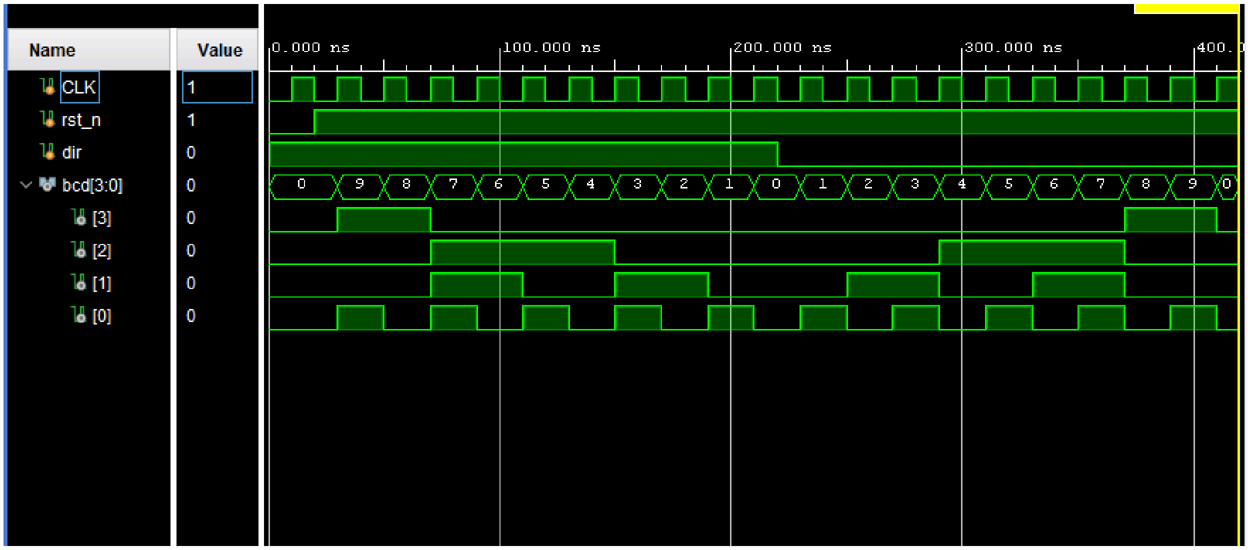
Top_lab6_tb.v



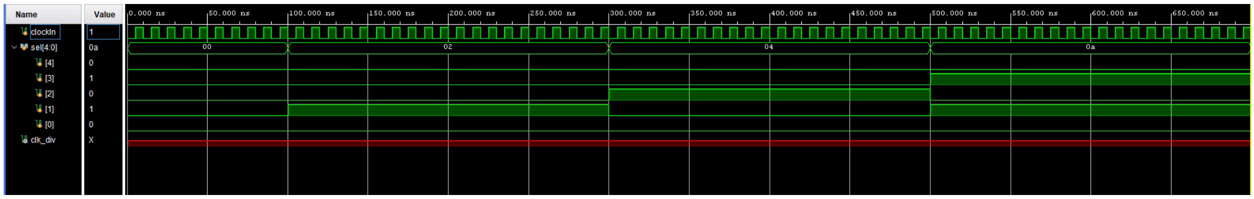
Alu_tb.v



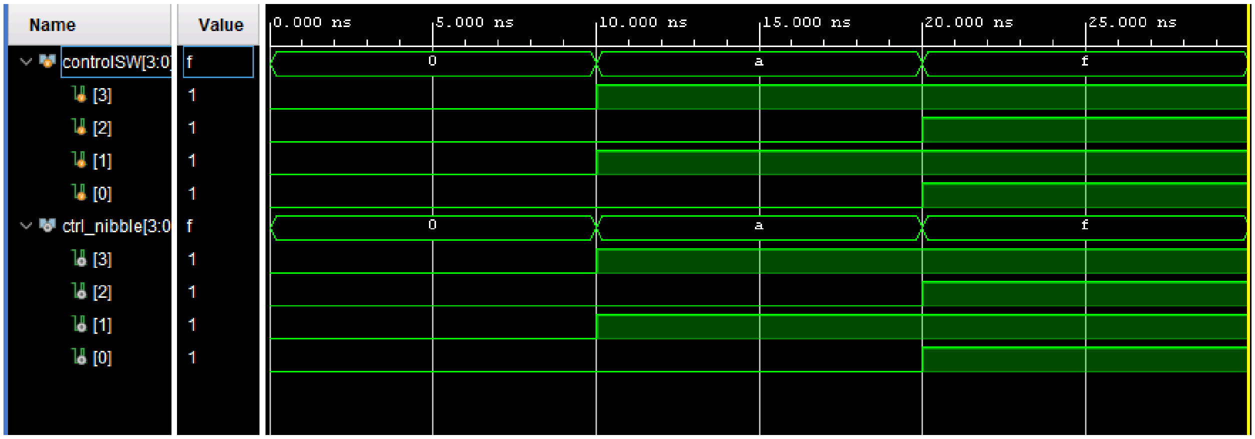
Bcd_counter_tb.v



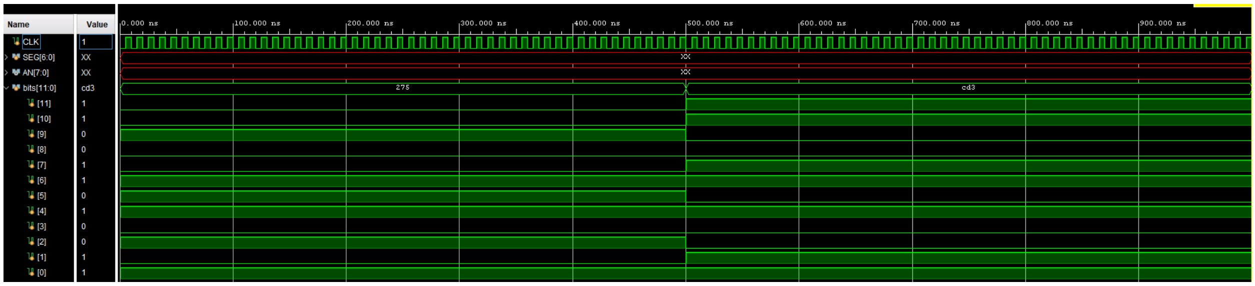
Clock_divider_tb.v



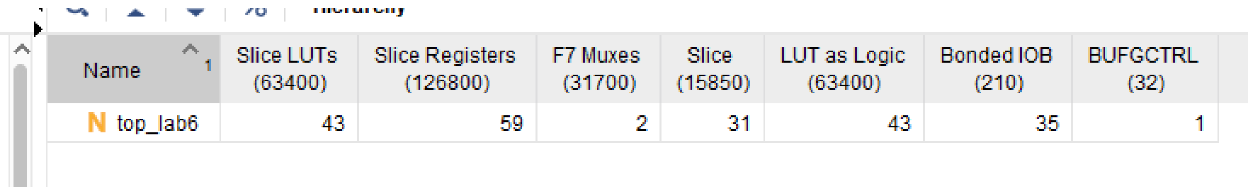
Control_decoder_tb.v



Seg7_scan_tb.v

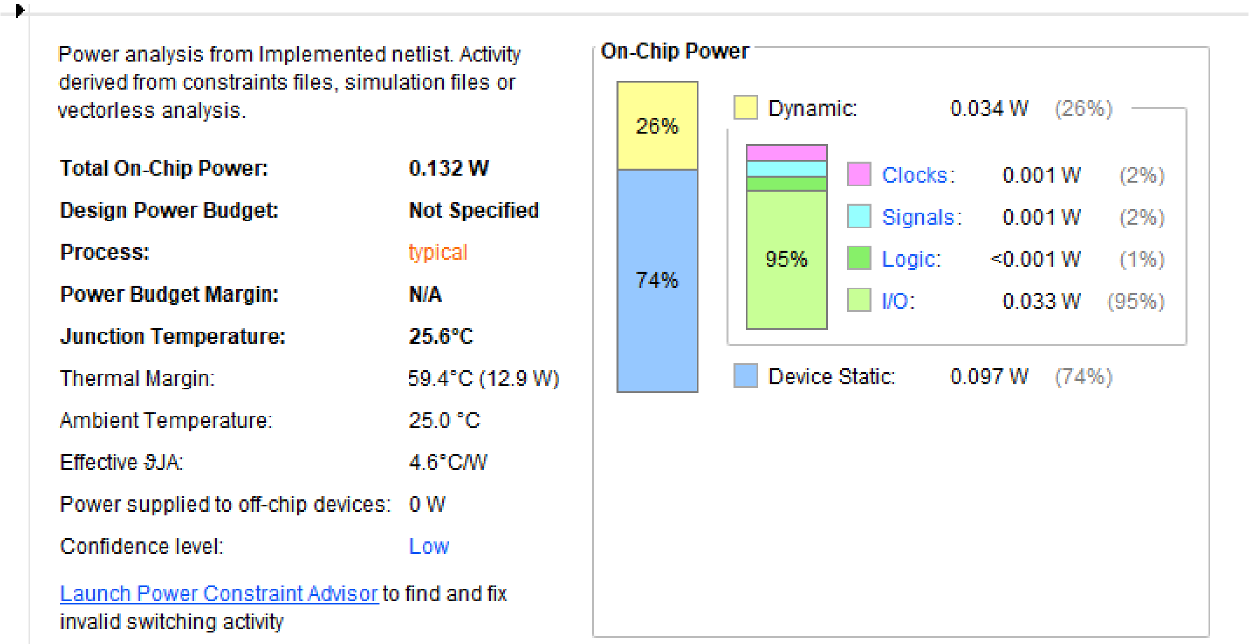


Utilization/Implementation:



Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	Slice (15850)	LUT as Logic (63400)	Bonded IOB (210)	BUFGCTRL (32)
N top_lab6	43	59	2	31	43	35	1

Power



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

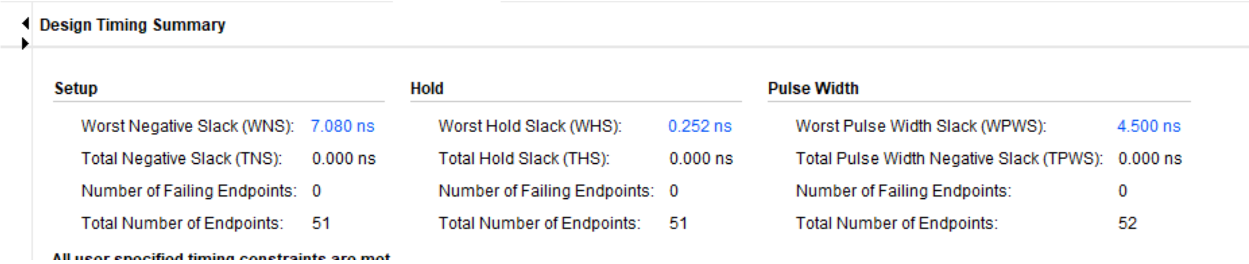
Total On-Chip Power:	**0.132 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.6°C**
Thermal Margin:	59.4°C (12.9 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	0.034 W	(26%)
Clocks:	0.001 W	(2%)
Signals:	0.001 W	(2%)
Logic:	<0.001 W	(1%)
I/O:	0.033 W	(95%)
Device Static:	0.097 W	(74%)

Time Summary:



Design Timing Summary

Setup		**Hold**		**Pulse Width**	
Worst Negative Slack (WNS):	7.080 ns	Worst Hold Slack (WHS):	0.252 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:	51	Total Number of Endpoints:	51	Total Number of Endpoints:	52

All user specified timing constraints are met.

Contributions:

Jared Mocling (50%) - board demo, compiled code, report.

Kevin Tang (50%)- compiled code, testbench code, Synthesis reports, report