CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA

COLLEGE OF ENGINEERING

LAB 6

Two Independent BCD Up/Down Counters

ECE 3300L Summer 2025

Digital Circuit Design using Verilog

Professor Mohamed Aly

By: Clay Kim and Changwe Musonda

Objective

This lab teaches you how to create a digital project with two independent BCD up/down counters, a 4-bit ALU, and a 7-segment display decoder.. The goal is to reinforce understanding of sequential logic, arithmetic logic, and display interfacing by creating a system responsive to switch inputs and pushbuttons on the Nexys A7 FPGA board. This project equals an assessment of logical understanding of digital components and physical work with chips on a board.

Hardware Pin Mapping

- **CLK:** 100MHz On-board clock
- **SW0 –** SW4: SW[4:0] Speed select (0 = slowest ... 31 = fastest)
- **SW5-SW6:** SW[6:5] ALU control bits(00 = add, 01 = sub)
- **Signal:** Port/Pin
- **SW7:** SW7 Units counter direction (1 = up, 0 = down)
- **SW8:** SW8 Tens counter direction (1 = up, 0 = down)
- **BTN0:** PushBtn0 Reset both counters (active low)
- **AN[2:0]:** AN0 – AN2 Digit enables for 3-digit scan
- **SEG[6:0]:** SEG A-G Seven-segment
- **LED[3:0]:** LED0 - LED3 Units counter BCD
- **LED[7:4]:** LED4 – LED7 Tens counter BCD

Verilog Code:

alu.v

```
module alu (
   input wire [3:0] A,        // units BCD
   input wire [3:0] B,        // tens BCD
   input wire [1:0] ctrl,     // {SW6, SW5}
   output reg [7:0] result
);
   always @(*) begin
      case (ctrl)
         2'b00: result = A + B;                   // Add (0–18)
         2'b01: result = (A >= B) ? (A - B) : ((A + 10) - B); // Subtract, wrap if underflow
         default: result = 8'd0;
      endcase
   end
endmodule
```

bcd_counter.v

```
module bcd_counter (
   input wire clk,
    input wire reset_n,      // Active-low reset
    input wire dir,          // 1 = up, 0 = down
    output reg [3:0] q       // 4-bit BCD output (0–9)
);
    always @(posedge clk or negedge reset_n) begin
        if (!reset_n)
            q <= 0;
        else if (dir) begin
            if (q == 9)
                q <= 0;
            else
                q <= q + 1;
        end
        else begin
            if (q == 0)
                q <= 9;
            else
                q <= q - 1;
        end
    end
endmodule
```

clock_divider.v

```
module clock_divider (
    input wire clk,                 // 100 MHz input clock
    input wire [4:0] sel,           // SW[4:0] selects division speed
    output wire clk_div             // Output divided clock
);
    reg [31:0] count = 0;

    always @(posedge clk) begin
        count <= count + 1;
    end

    assign clk_div = count[sel];
endmodule
```

control_decoder.v

```
module control_decoder (
    input  wire [3:0] nibble,       // {SW8, SW7, SW6, SW5}
    output wire [3:0] ctrl_nibble   // For display
);
    assign ctrl_nibble = nibble;
endmodule
```

seg7_scan.v

```
module seg7_scan(
    input wire clk,                    // 100 MHz clock
    input wire [7:0] result,           // ALU result
    input wire [3:0] ctrl_nibble,      // Control bits
    output reg [2:0] AN,               // Active low digit enables
    output reg [6:0] SEG               // Segments (a–g)
);

    reg [15:0] scan_cnt = 0;
    reg [1:0] digit_idx = 0;
    reg [3:0] value;

    // Scan timing ~1kHz (assuming 100 MHz clock)
    always @(posedge clk) begin
        if (scan_cnt == 16'd49999) begin // 100MHz/1kHz/3 = 33333, using 50000 for simplicity
            scan_cnt <= 0;
            digit_idx <= digit_idx + 1;
        end else begin
            scan_cnt <= scan_cnt + 1;
        end
    end

    always @(*) begin
        case (digit_idx)
            2'd0: begin AN = 3'b110; value = result[3:0]; end // AN0 = display result low nibble
            2'd1: begin AN = 3'b101; value = result[7:4]; end // AN1 = display result high nibble
            2'd2: begin AN = 3'b011; value = ctrl_nibble; end // AN2 = display control nibble
            default: begin AN = 3'b111; value = 4'd0; end
        endcase
    end

    // 7-segment decoder for common anode (active low)
    always @(*) begin
        case (value)
            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
```

switch_led_interface.v

```
module switch_led_interface(
       input wire [15:0] sw,
       output wire [15:0] led

    );
    assign sw = led;
endmodule
```

top_lab6.v

```
module top_lab6(
    input  wire clk,              // 100 MHz
    input  wire [8:0] SW,         // SW[8:0]
    input  wire BTN0,             // Reset, active-low
    output wire [7:0] LED,        // Debug LEDs
    output wire [7:0] AN,         // Digit enable
    output wire [6:0] SEG         // 7-segment
);

    wire clk_div;
    wire [3:0] units_bcd, tens_bcd;
    wire [7:0] alu_result;
    wire [3:0] ctrl_nibble;

    // Clock Divider
    clock_divider u_clkdiv (
        .clk(clk),
        .sel(SW[4:0]),
        .clk_div(clk_div)
    );

    // Units Counter
    bcd_counter u_units (
        .clk(clk_div),
        .reset_n(BTN0),
        .dir(SW[7]),
        .q(units_bcd)
    );
```

```
// Tens Counter
bcd_counter u_tens (
    .clk(clk_div),
    .reset_n(BTN0),
    .dir(SW[8]),
    .q(tens_bcd)
);

// ALU
alu u_alu (
    .A(units_bcd),
    .B(tens_bcd),
    .ctrl(SW[6:5]),
    .result(alu_result)
);

// Control Decoder
control_decoder u_ctrl (
    .nibble({SW[8], SW[7], SW[6], SW[5]}),
    .ctrl_nibble(ctrl_nibble)
);

// 7-seg Scanner
seg7_scan u_seg7 (
    .clk(clk),
    .result(alu_result),
    .ctrl_nibble(ctrl_nibble),
    .AN(AN),
    .SEG(SEG)
    );


    // Debug LEDs

    assign LED[3:0] = units_bcd;

    assign LED[7:4] = tens_bcd;

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

endmodule
```

Testbench:

alu_tb.v

```
module alu_tb;
    // Declare inputs as regs and outputs as wires
    reg [3:0] A, B;
    reg [2:0] opcode;
    wire [3:0] Y;
    wire cout, zero;

    // Instantiate the ALU
    alu uut (
        .A(A),
        .B(B),
        .opcode(opcode),
        .Y(Y),
        .cout(cout),
        .zero(zero)
    );

    initial begin
        // Monitor values
        $monitor("Time=%0t A=%h B=%h opcode=%b | Y=%h cout=%b zero=%b", $time, A, B, opcode, Y, cout,
zero);

        // Test case 1: Addition
        A = 4'h3; B = 4'h5; opcode = 3'b000; #10;
        // Test case 2: Subtraction
        A = 4'h7; B = 4'h2; opcode = 3'b001; #10;
        // Test case 3: AND
        A = 4'hF; B = 4'hA; opcode = 3'b010; #10;
        // Test case 4: OR
        A = 4'h6; B = 4'h9; opcode = 3'b011; #10;
        // Test case 5: XOR
        A = 4'hC; B = 4'h7; opcode = 3'b100; #10;
        // Test case 6: Zero output
        A = 4'h0; B = 4'h0; opcode = 3'b000; #10;

        // Add more tests as needed for your ALU functions

        $finish;
    end
endmodule
```

bcd_counter_tb.v

```
module bcd_counter_tb(

    );
endmodule
```

clock_divider_tb.v

```
module clock_divider_tb;
    reg clk, reset;
    wire clk_out;

    clock_divider uut (
        .clk(clk),
        .reset(reset),
        .clk_out(clk_out)
    );

    initial clk = 0;
    always #2 clk = ~clk; // Fast clock for sim

    initial begin
        reset = 1; #10;
        reset = 0; #10;
        #200; // Wait for clock_out toggles
        $finish;
    end

    initial begin
        $monitor("Time=%0t clk=%b clk_out=%b", $time, clk, clk_out);
    end
endmodule
```

control_decoder_tb.v

```
module control_decoder_tb;
    reg [2:0] opcode;
    wire [3:0] ctrl;

    control_decoder uut (
        .opcode(opcode),
        .ctrl(ctrl)
    );

    initial begin
        $monitor("Time=%0t opcode=%b ctrl=%b", $time, opcode, ctrl);

        opcode = 3'b000; #10;
        opcode = 3'b001; #10;
        opcode = 3'b010; #10;
        opcode = 3'b011; #10;
        opcode = 3'b100; #10;
        opcode = 3'b101; #10;
        opcode = 3'b110; #10;
        opcode = 3'b111; #10;
        $finish;
    end
endmodule
```

seg7_scan_tb.v

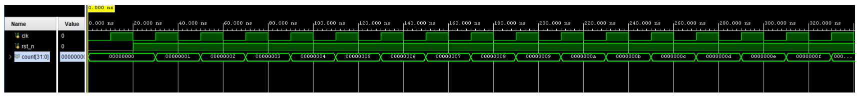
```
module seg7_scan_tb;
    reg clk, reset;
    reg [15:0] data_in;
    wire [3:0] an;
    wire [6:0] seg;

    seg7_scan uut (
        .clk(clk),
        .reset(reset),
        .data_in(data_in),
        .an(an),
        .seg(seg)
    );

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

    initial begin
        reset = 1; data_in = 16'h1234; #10;
        reset = 0; #10;
        data_in = 16'hABCD; #20;
        data_in = 16'h0000; #20;
        data_in = 16'hFFFF; #20;
        $finish;
    end

    initial begin
        $monitor("Time=%0t data_in=%h an=%b seg=%b", $time, data_in, an, seg);
    end
endmodule
```

top_lab6_tb.v

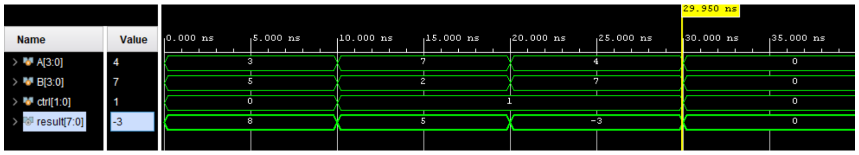
```
module top_lab6_tb;
   reg clk, reset, up, down;
   wire [6:0] seg;
   wire [3:0] an;

   top_lab6 uut (
      .clk(clk),
      .reset(reset),
      .up(up),
      .down(down),
      .seg(seg),
      .an(an)
);

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

   initial begin
      reset = 1; up = 0; down = 0; #20;
      reset = 0; #10;
      // Test counting up
      up = 1; down = 0; #50;
      // Test counting down
      up = 0; down = 1; #50;
      // Both up and down inactive
      up = 0; down = 0; #20;
      $finish;
   end

   initial begin
      $monitor("Time=%0t seg=%b an=%b", $time, seg, an);
   end
endmodule
```

Synthesis and Implementation

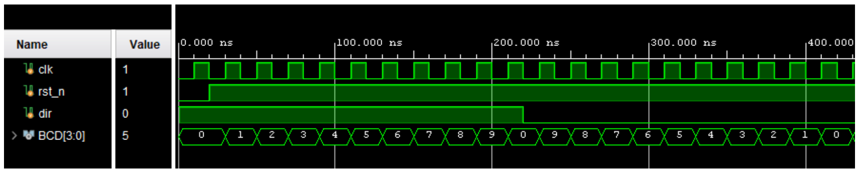
Clock divider



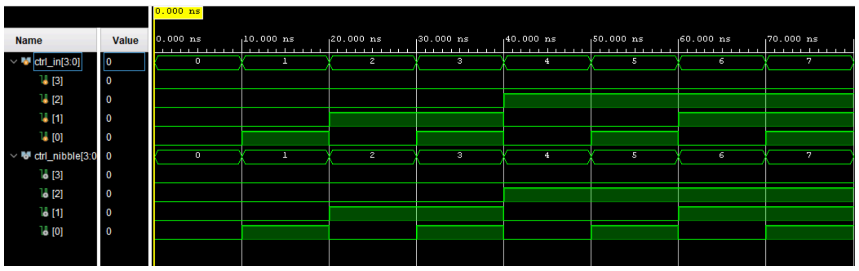
ALU

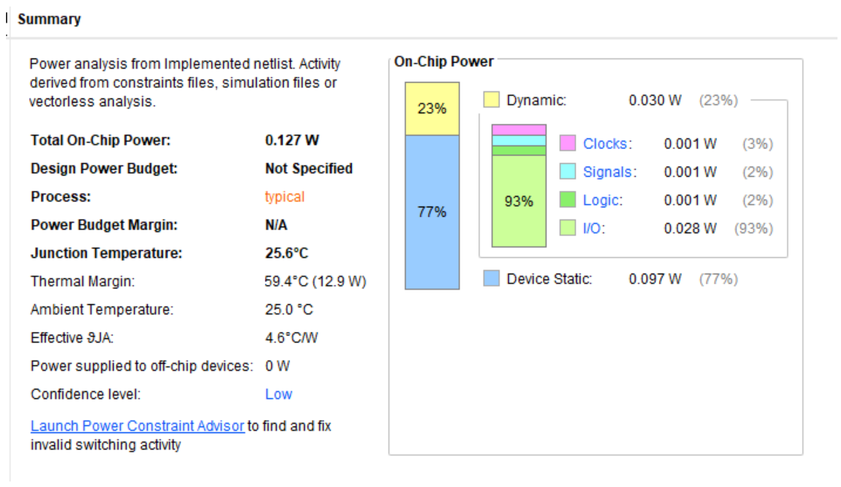


BCD Counter



Cntrl_decoder





Summary

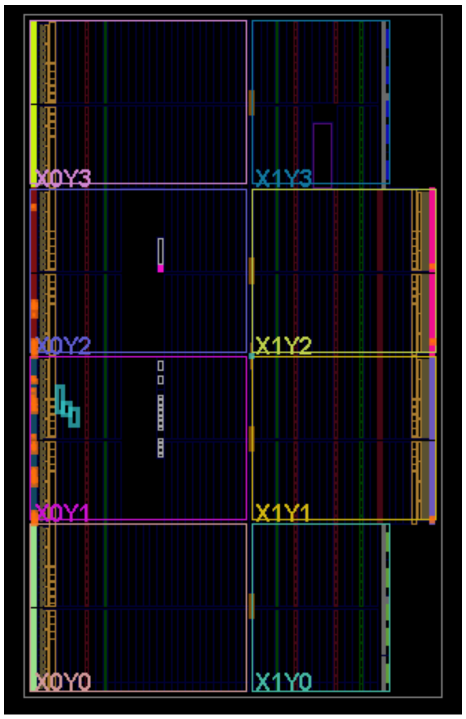
Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.127 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:	0.030 W	(23%)	
Clocks:	0.001 W	(3%)	
Signals:	0.001 W	(2%)	
Logic:	0.001 W	(2%)	
I/O:	0.028 W	(93%)	
Device Static:	0.097 W	(77%)	



<u>Partner Contribution:</u>

Changwe Musonda: Testbench and Simulation
Clay Kim: Implementation and Demonstration Video

Reflections

The successful completion of this lab demonstrated the integration of counters, ALU operations, and display decoding in a cohesive FPGA-based system. By using switches to control counter direction and ALU operations, we were able to observe the effects in real-time through LEDs and the 7-segment display. By implementing testbenches and simulations we were able to verify correctness before programming the hardware. The project emphasized the importance of modular design and signal synchronization in digital systems. Overall, this lab enhanced practical skills in Verilog design and FPGA implementation.