CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA

COLLEGE OF ENGINEERING

LAB 5

BCD Up/Down Counter on 7-Segment Display

ECE 3300L Summer 2025

Digital Circuit Design using Verilog

Professor Mohamed Aly

By: Clay Kim and Changwe Musonda

Objective

The objective of this lab was to design and implement a two-digit BCD (Binary-Coded Decimal) up/down counter in Verilog. This was controlled by pushbuttons for direction and reset, with speed selection controlled by slide switches.

Hardware Connections

- **CLK:** 100 MHz board clock input.
- **SW[4:0]:** Pins SW0-SW4, providing a 5-bit input for speed-select index (slowest = 0, fastest = 31).
- **BTN0:** Pin PushBtn0, serving as an active-low reset signal.
- **BTN1:** Pin PushBtn1, used to select direction (1 = Up, 0 = Down).
- **AN[1:0]:** Pins AN0-AN1, providing 2-bit output for seven-segment display digit enable lines (for scan multiplexing).
- **SEG[6:0]:** Pins SegPins, providing 7-bit output for controlling the segments of the seven-segment display.
- **LED[4:0]:** Pins LED0-LED4, providing a 5-bit output that mirrors the state of SW[4:0] for debugging purposes.
- **LED[7:0]:** Pins LED5-LED12, providing an 8-bit output representing a BCD output indicator; LED[3:0] shows units digit BCD, and LED[7:4] shows tens digit BCD.

Verilog Code: **clock_divider.v**

```
module clock_divider(
    input CLK,
    output reg [31:0] cnt = 0
  );
  always @(posedge CLK)
    cnt <= cnt + 1;
endmodule
```

mux32x1.v

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

bcd_up_down_counter.v

```
module bcd_up_down_counter(
    input clk_div,
    input rst_n,
    input dir,
    output reg [3:0] digit0 = 0,
    output reg [3:0] digit1 = 0
);
    always @(posedge clk_div or negedge rst_n) begin
        if (!rst_n) begin
            digit0 <= 0;
            digit1 <= 0;
        end else begin
            if (dir) begin  // Up
                if (digit0 == 9) begin
                    digit0 <= 0;
                    digit1 <= (digit1 == 9) ? 0 : digit1 + 1;
                end else
                    digit0 <= digit0 + 1;
            end else begin  // Down
                if (digit0 == 0) begin
                    digit0 <= 9;
                    digit1 <= (digit1 == 0) ? 9 : digit1 - 1;
                end else
                    digit0 <= digit0 - 1;
            end
        end
    end
endmodule
```

seg7_scan.v

```
module seg7_scan(
    input CLK,
    input [3:0] digit0, digit1,
    output reg [1:0] AN,
    output reg [6:0] SEG
    );

    reg [17:0] refresh_counter = 0; // For 100MHz clock, ~1kHz multiplexing
    wire digit_select;

    always @(posedge CLK) begin
        refresh_counter <= refresh_counter + 1;
    end

    assign digit_select = refresh_counter[17];  // Toggle approximately every 1ms

    reg [3:0] current_digit;
```

```
// Multiplex between digit0 and digit1
always @(*) begin
   if (digit_select == 1) begin
      AN = 2'b10; // Activate digit 0 (units)
      current_digit = digit0;
   end else begin
      AN = 2'b01; // Activate digit 1 (tens)
      current_digit = digit1;
   end
end

// Segment decoding logic (assuming common-anode)
always @(*) begin
   case (current_digit)
      4'd0: SEG = 7'b1000000;
      4'd1: SEG = 7'b1111001;
      4'd2: SEG = 7'b0100100;
      4'd3: SEG = 7'b0110000;
      4'd4: SEG = 7'b0011001;
      4'd5: SEG = 7'b0010010;
      4'd6: SEG = 7'b0000010;
      4'd7: SEG = 7'b1111000;
      4'd8: SEG = 7'b0000000;
      4'd9: SEG = 7'b0010000;
      default: SEG = 7'b1111111;  // Blank
   endcase
end
```

```
endmodule
```

top_lab5.v

```
module top_lab5(
    input CLK,
    input BTN0,        // BTN0: reset button
    input BTN1,        // BTN1: direction button
    input [4:0] SW,    // SW[4:0]: speed select
    output [1:0] AN,   // 7-seg anodes
    output [6:0] SEG,  // 7-seg segments
    output [4:0] LED,  // Debug LEDS for SW[4:0]
    output [7:0] LED_BCD // BCD display
);

    wire [31:0] cnt;
    wire clk_mux;
    reg clk_div = 0;
    wire rst_n;
    wire [3:0] digit0, digit1;

    assign rst_n = ~BTN0; // Invert BTN0 so reset is active-low

    clock_divider div(.CLK(CLK), .cnt(cnt));
    mux32x1 mux(.cnt(cnt), .sel(SW), .clk_out(clk_mux));

    // Register the mux output to make a stable divided clock
    always @(posedge CLK) begin
        clk_div <= clk_mux;
    end

    bcd_up_down_counter counter(
        .clk_div(clk_div),
        .rst_n(rst_n),
        .dir(BTN1),
        .digit0(digit0),
        .digit1(digit1)
    );

    seg7_scan display(
        .CLK(CLK),
        .digit0(digit0),
        .digit1(digit1),
        .AN(AN),
        .SEG(SEG)
    );

    assign LED = SW;
    assign LED_BCD = {digit1, digit0};
endmodule
```

Testbench code: bcd_up_down_counter_tb.v

```
module bcd_up_down_counter_tb();
```

```
reg clk_div;
reg rst_n;
reg dir;
wire [3:0] digit0, digit1;

// Instantiate the device under test (DUT)
bcd_up_down_counter dut (
    .clk_div(clk_div),
    .rst_n(rst_n),
    .dir(dir),
    .digit0(digit0),
    .digit1(digit1)
);

// Generate a clock with 10ns period (100 MHz)
initial begin
    clk_div = 0;
    forever #5 clk_div = ~clk_div;  // 10ns period
end

initial begin
    // Initialize signals
    rst_n = 0;    // Assert reset (active low)
    dir = 1;      // Start counting up

    // Hold reset for a few clock cycles
    #20;
```

```
        rst_n = 1;    // Release reset, start counting up

        // Let the counter count up for 200 clock cycles
        #(200 * 10);

        // Change direction to count down
        dir = 0;

        // Let the counter count down for 200 clock cycles
        #(200 * 10);

        // Assert reset again to verify reset behavior
        rst_n = 0;
        #20;
        rst_n = 1;

        // Let the counter count up again for 100 clock cycles
        dir = 1;
        #(100 * 10);

        $stop; // End simulation
    end

endmodule
```

XDC Snippets

```
# Clock signal

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]


## 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_BCD[0] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports {
LED_BCD[1] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]

set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports {
LED_BCD[2] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]

set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {
LED_BCD[3] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]

set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports {
LED_BCD[4] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]

set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports {
LED_BCD[5] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]

set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports {
LED_BCD[6] }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]

set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports {
LED_BCD[7] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]


#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]

##Buttons

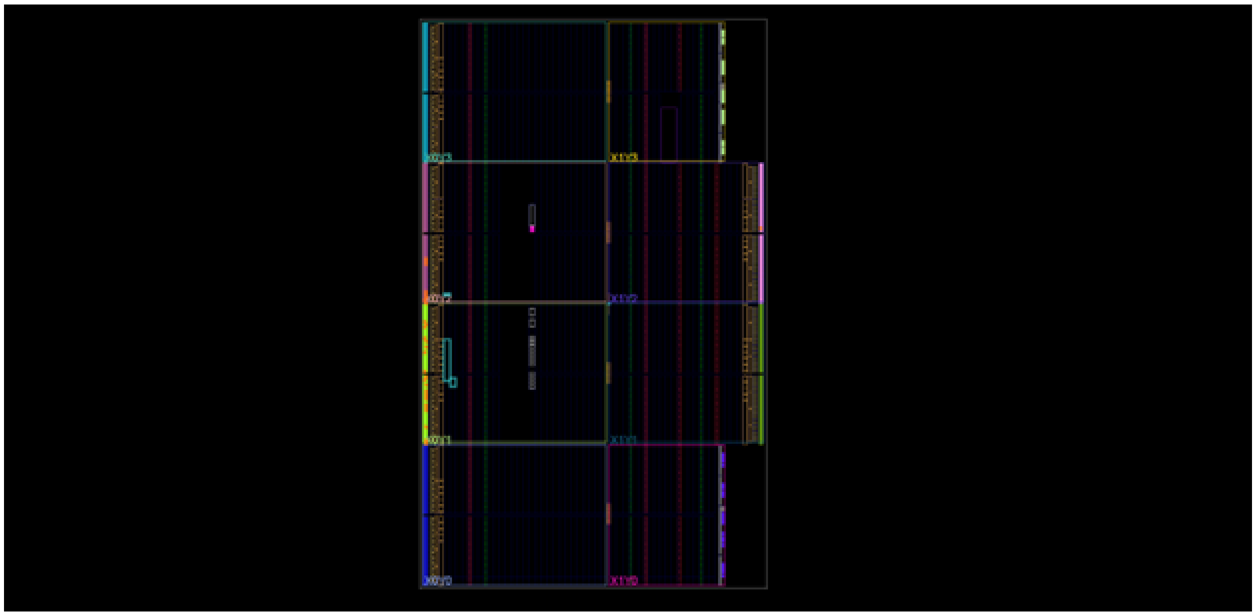
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { BTN0 }]; #IO_L9P_T1_DQS_14 Sch=btnc

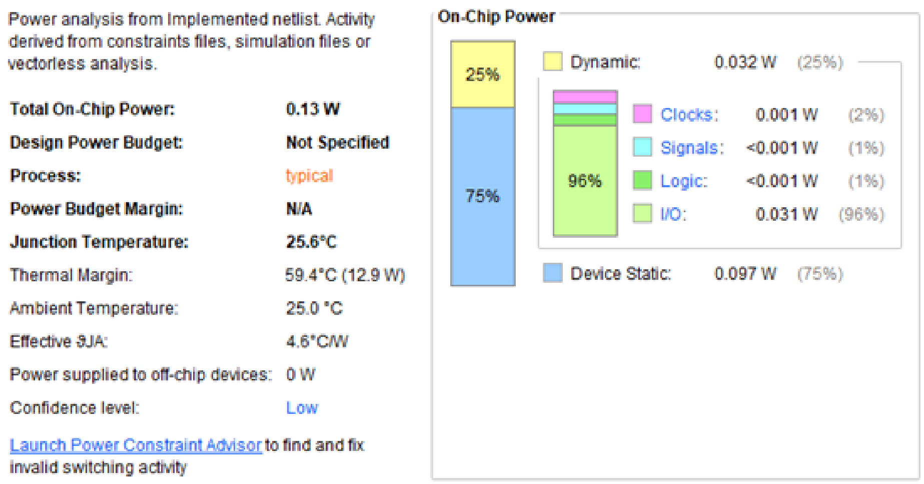
set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports { BTN1 }]; #IO_L4N_T0_D05_14 Sch=btnu

Synthesis and Implementation

Screenshots

Vivado Utilization



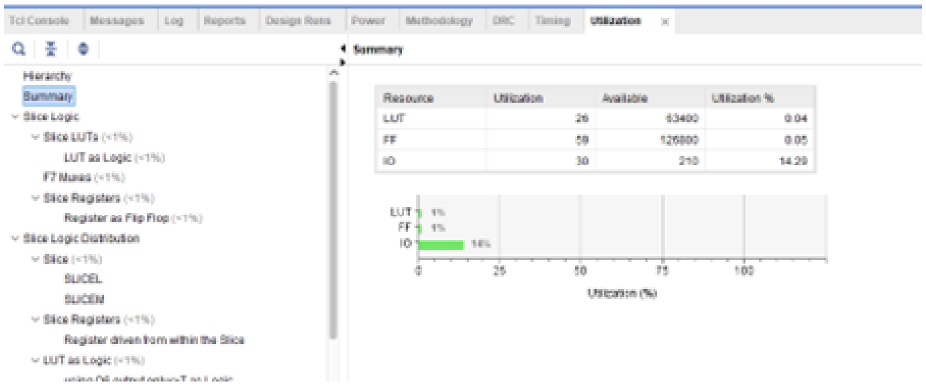

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.13 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:	0.032 W	(25%)
🟪	Clocks:	0.001 W	(2%)
🟦	Signals:	<0.001 W	(1%)
🟩	Logic:	<0.001 W	(1%)
🟩	I/O:	0.031 W	(96%)
🟦	Device Static:	0.097 W	(75%)

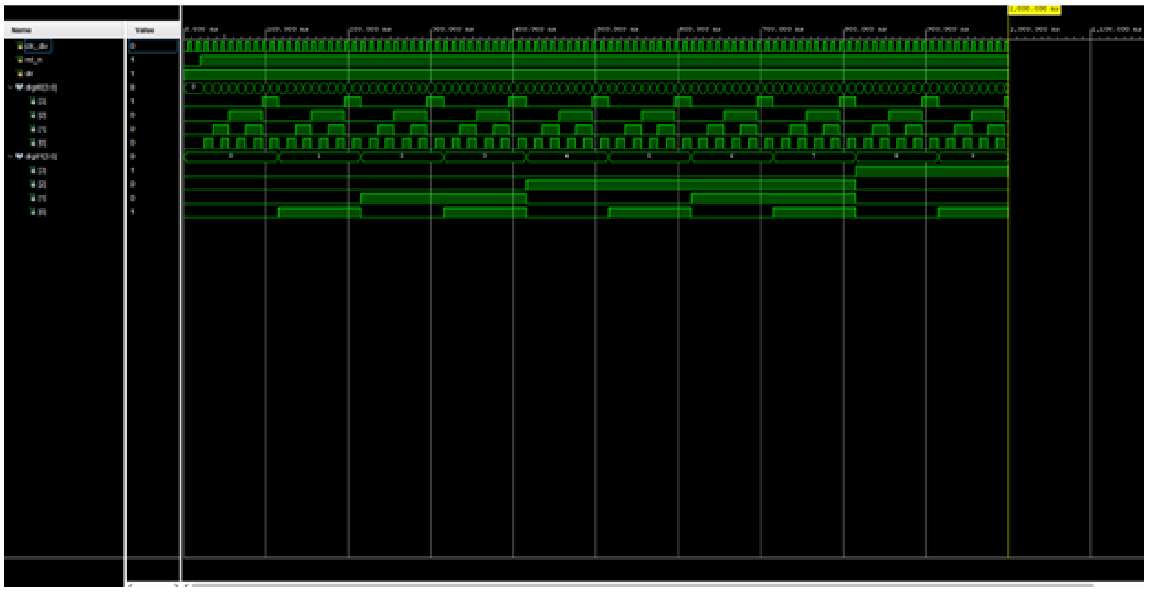


Hierarchy
Summary
Slice Logic
Slice LUTs (<1%)
LUT as Logic (<1%)
F7 Muxes (<1%)
Slice Registers (<1%)
Register as Flip Flop (<1%)
Slice Logic Distribution
Slice (<1%)
SLICEL
SLICEM
Slice Registers (<1%)
Register driven from within the Slice
LUT as Logic (<1%)

Summary

Resource	Utilization	Available	Utilization %
LUT	26	63400	0.04
FF	59	126800	0.05
IO	30	210	14.29

Waveform



Partner Contribution

Clay Kim: Testbench and Simulation

Changwe Musonda: Demonstration video and implementation

Reflections

This lab was a valuable hands-on project that successfully reinforced key digital logic concepts through a modular design approach. By breaking the system into a clock divider, a BCD counter, and a seven-segment display driver, it was easier to manage the design and implement essential techniques like clock division for visible speed control and time-division multiplexing for the dual-digit display. The experience also highlighted important challenges and areas for future improvement, such as the necessity of implementing a pushbutton debouncing circuit to ensure clean inputs, the importance of thorough testing to catch coding errors like the incorrect segment display for the number nine, and the benefits of using synchronous resets in more complex designs.