ECE3300L

Lab 7

By Justin Wong, Hector Garibay

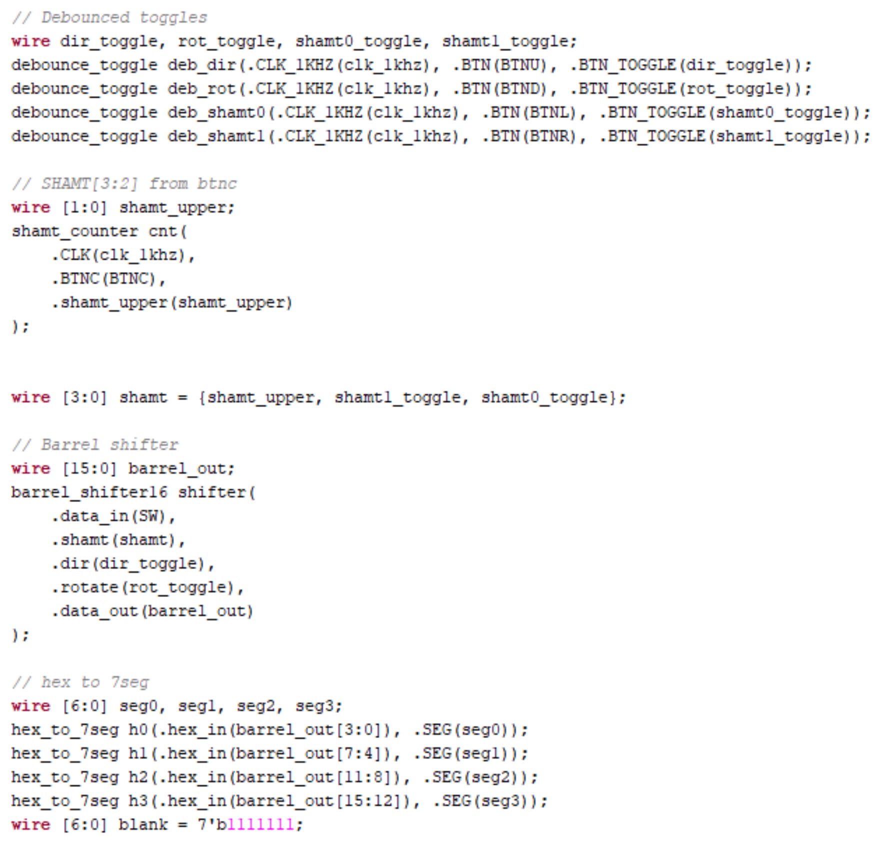
Summer 2025

Report Date: July 6, 2025

Objective

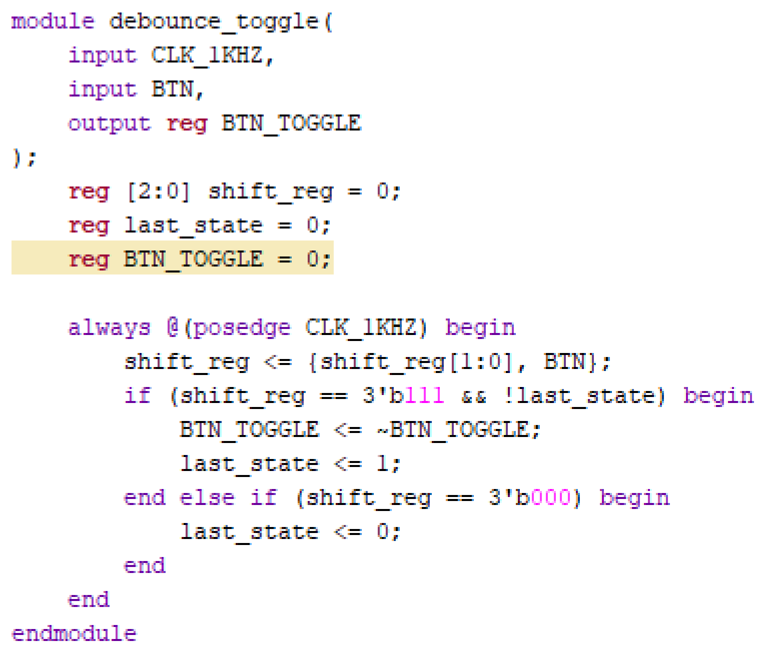
The objective of this lab is to design and implement a 16-bit combinational barrel shifter on an FPGA that supports logical and rotational shifts in both directions, left and right, by a variable amount from 0 to 15. Inputs are provided via slide switches and push buttons, with outputs displayed on a 7-segment display. The system includes a debounced control interface, an asynchronous reset, and visual feedback through LEDs.

Top Lab Code Snippet:



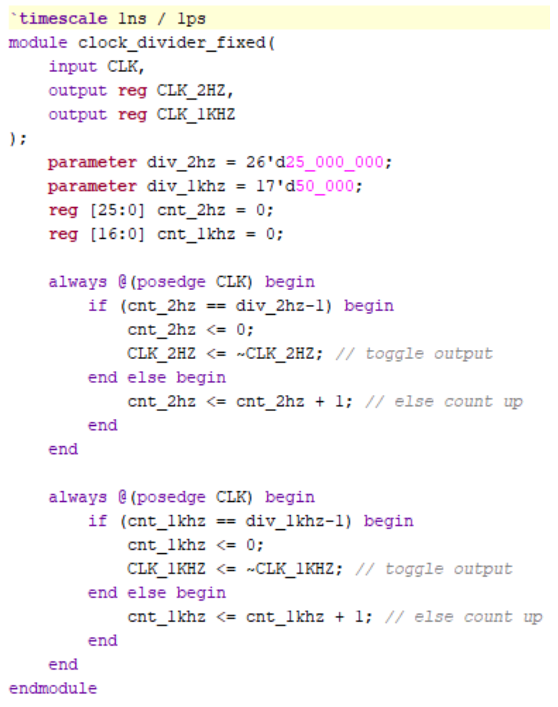
```
// Debounced toggles
wire dir_toggle, rot_toggle, shamt0_toggle, shamt1_toggle;
debounce_toggle deb_dir(.CLK_1KHZ(clk_1khz), .BTN(BTNU), .BTN_TOGGLE(dir_toggle));
debounce_toggle deb_rot(.CLK_1KHZ(clk_1khz), .BTN(BTND), .BTN_TOGGLE(rot_toggle));
debounce_toggle deb_shamt0(.CLK_1KHZ(clk_1khz), .BTN(BTNL), .BTN_TOGGLE(shamt0_toggle));
debounce_toggle deb_shamt1(.CLK_1KHZ(clk_1khz), .BTN(BTNR), .BTN_TOGGLE(shamt1_toggle));

// SHAMT[3:2] from btnc
wire [1:0] shamt_upper;
shamt_counter cnt(
    .CLK(clk_1khz),
    .BTNC(BTNC),
    .shamt_upper(shamt_upper)
);


wire [3:0] shamt = {shamt_upper, shamt1_toggle, shamt0_toggle};

// Barrel shifter
wire [15:0] barrel_out;
barrel_shifter16 shifter(
    .data_in(SW),
    .shamt(shamt),
    .dir(dir_toggle),
    .rotate(rot_toggle),
    .data_out(barrel_out)
);

// hex to 7seg
wire [6:0] seg0, seg1, seg2, seg3;
hex_to_7seg h0(.hex_in(barrel_out[3:0]), .SEG(seg0));
hex_to_7seg h1(.hex_in(barrel_out[7:4]), .SEG(seg1));
hex_to_7seg h2(.hex_in(barrel_out[11:8]), .SEG(seg2));
hex_to_7seg h3(.hex_in(barrel_out[15:12]), .SEG(seg3));
wire [6:0] blank = 7'b1111111;
```

debounce_toggle Snippet:



```
module debounce_toggle(
    input CLK_1KHZ,
    input BTN,
    output reg BTN_TOGGLE
);
    reg [2:0] shift_reg = 0;
    reg last_state = 0;
    reg BTN_TOGGLE = 0;

    always @(posedge CLK_1KHZ) begin
        shift_reg <= {shift_reg[1:0], BTN};
        if (shift_reg == 3'b111 && !last_state) begin
            BTN_TOGGLE <= ~BTN_TOGGLE;
            last_state <= 1;
        end else if (shift_reg == 3'b000) begin
            last_state <= 0;
        end
    end
endmodule
```

clock_divider_fixed Snippet:



```
`timescale 1ns / 1ps
module clock_divider_fixed(
    input CLK,
    output reg CLK_2HZ,
    output reg CLK_1KHZ
);
    parameter div_2hz = 26'd25_000_000;
    parameter div_1khz = 17'd50_000;
    reg [25:0] cnt_2hz = 0;
    reg [16:0] cnt_1khz = 0;

    always @(posedge CLK) begin
        if (cnt_2hz == div_2hz-1) begin
            cnt_2hz <= 0;
            CLK_2HZ <= ~CLK_2HZ; // toggle output
        end else begin
            cnt_2hz <= cnt_2hz + 1; // else count up
        end
    end

    always @(posedge CLK) begin
        if (cnt_1khz == div_1khz-1) begin
            cnt_1khz <= 0;
            CLK_1KHZ <= ~CLK_1KHZ; // toggle output
        end else begin
            cnt_1khz <= cnt_1khz + 1; // else count up
        end
    end
endmodule
```

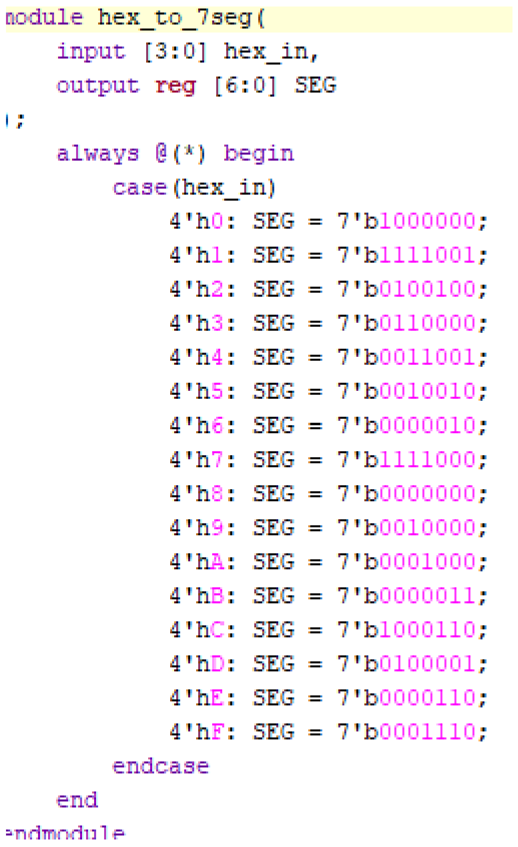
shamt_counter Code:

```verilog
module shamt_counter(
    input CLK,
    input BTNC,
    output reg [1:0] shamt_upper
);
    reg btnc_last = 0;
    always @(posedge CLK) begin
        btnc_last <= BTNC;
        if (BTNC & ~btnc_last)
            shamt_upper <= shamt_upper + 1;
    end
endmodule
```

barrel_shifter16 Code:

```verilog
    output [15:0] data_out
);

    wire [15:0] stage1, stage2, stage3, stage4;

    // shift 1 bit
    assign stage1 = shamt[0] ?
        (dir ?
            (rotate ? {data_in[0], data_in[15:1]} : {1'b0, data_in[15:1]}) :
            (rotate ? {data_in[14:0], data_in[15]} : {data_in[14:0], 1'b0})
        ) : data_in;

   //shift 2 bits
    assign stage2 = shamt[1] ?
        (dir ?
            (rotate ? {stage1[1:0], stage1[15:2]} : {2'b00, stage1[15:2]}) :
            (rotate ? {stage1[13:0], stage1[15:14]} : {stage1[13:0], 2'b00})
        ) : stage1;

    //shift 4 bits
    assign stage3 = shamt[2] ?
        (dir ?
            (rotate ? {stage2[3:0], stage2[15:4]} : {4'b0000, stage2[15:4]}) :
            (rotate ? {stage2[11:0], stage2[15:12]} : {stage2[11:0], 4'b0000})
        ) : stage2;

   //shift 8 bits
    assign stage4 = shamt[3] ?
        (dir ?
            (rotate ? {stage3[7:0], stage3[15:8]} : {8'b00000000, stage3[15:8]}) :
            (rotate ? {stage3[7:0], stage3[15:8]} : {stage3[7:0], 8'b00000000})
        ) : stage3;

    assign data_out = stage4;

endmodule
```

Hex_to_7seg Code:



```
module hex_to_7seg(
    input [3:0] hex_in,
    output reg [6:0] SEG
);
    always @(*) begin
        case(hex_in)
            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;
        endcase
    end
endmodule
```

Test Bench Code Snippet

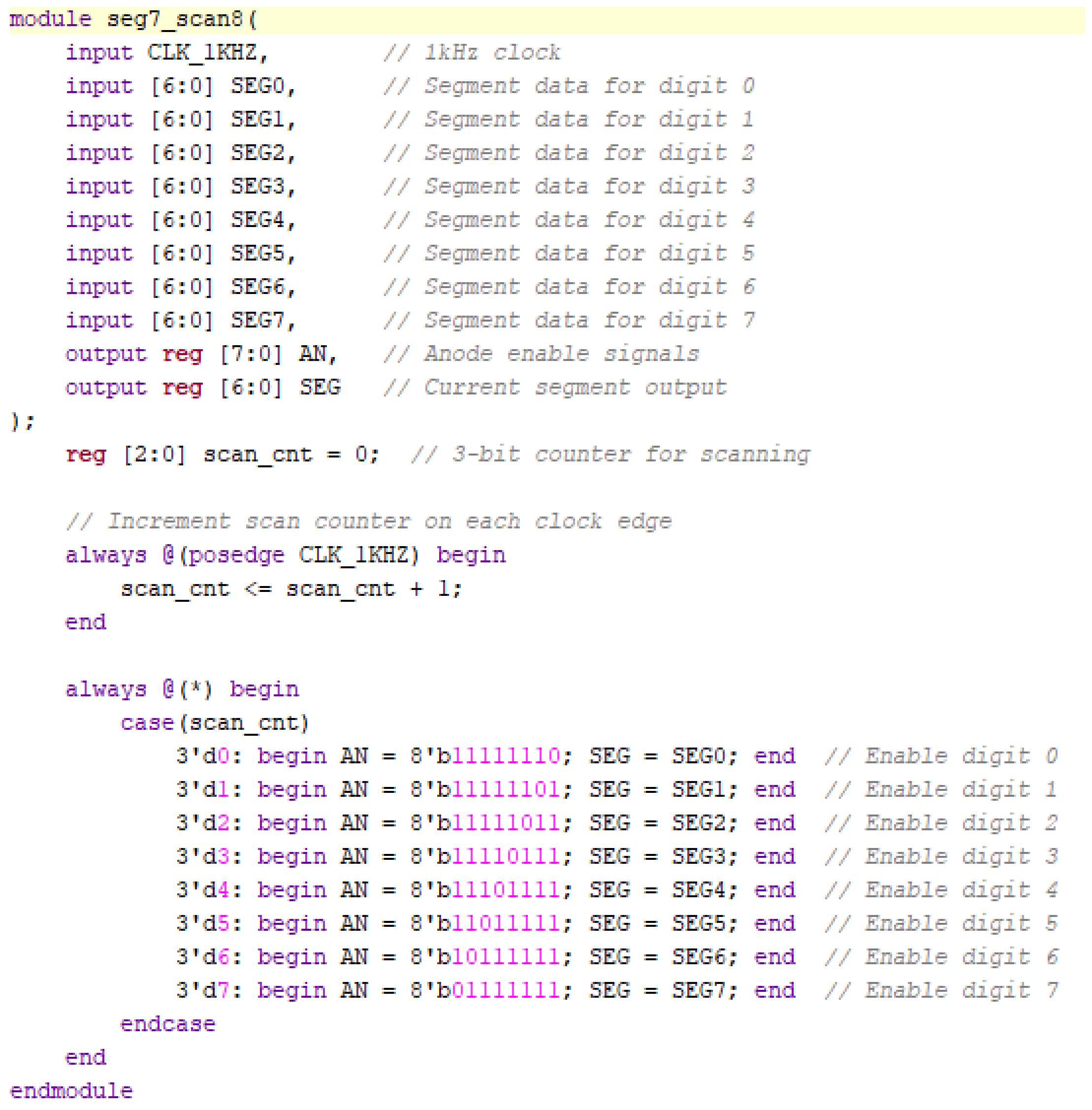
barrel_shifter16_tb

```
initial
    begin
        data_in = 16'b1010010110100101;
        shamt = 4'd0;   //left shift
        dir = 0;
        rotate = 0;
        #100

        shamt = 4'd0;   //right shift
        dir = 1;
        rotate = 0;
        #100

        shamt = 4'd0;   //rotate left
        dir = 0;
        rotate = 1;
        #100

        shamt = 4'd0;   //rotate right
        dir = 1;
        rotate = 1;
        #100
```

debounce_toggle_tb

```
    reg CLK_1KHZ;
    reg BTN;

    wire BTN_TOGGLE;

    debounce_toggle dut(
        .CLK_1KHZ(CLK_1KHZ),
        .BTN(BTN),
        .BTN_TOGGLE(BTN_TOGGLE)
    );

    always #5 CLK_1KHZ = ~CLK_1KHZ;
    initial begin
        CLK_1KHZ = 0;
        BTN = 0;

        #50
        BTN = 1;    //normal press
        #100
        BTN = 0;
        #500


        BTN = 1;    //bounce
        #10
        BTN = 0;
        #10
        BTN = 1;
        #10
        BTN = 0;
        #10

        $finish;
    end
endmodule
```

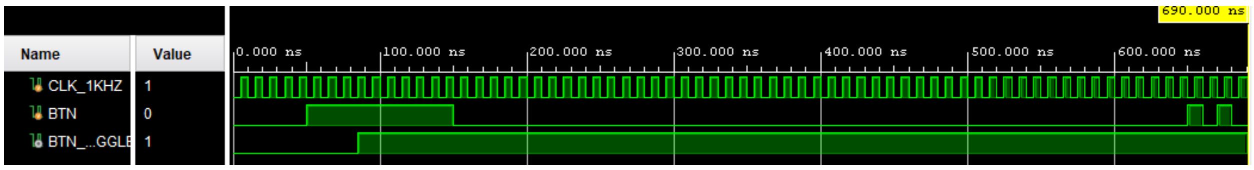
7seg-scan8_tb



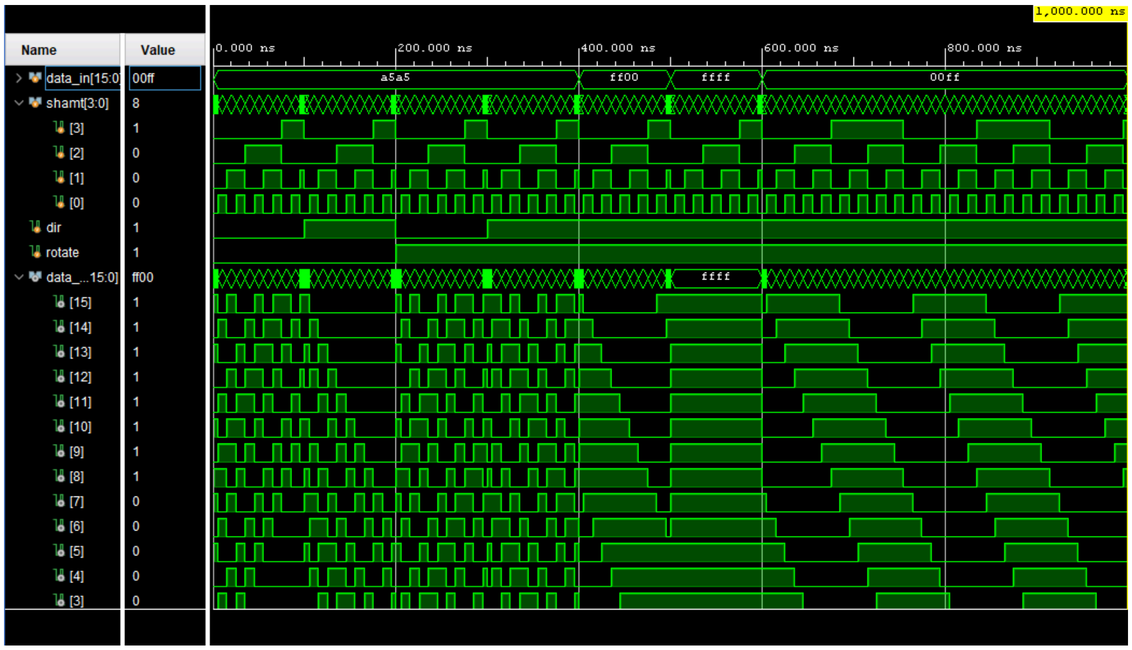
```
module seg7_scan8(
    input CLK_1KHZ,         // 1kHz clock
    input [6:0] SEG0,       // Segment data for digit 0
    input [6:0] SEG1,       // Segment data for digit 1
    input [6:0] SEG2,       // Segment data for digit 2
    input [6:0] SEG3,       // Segment data for digit 3
    input [6:0] SEG4,       // Segment data for digit 4
    input [6:0] SEG5,       // Segment data for digit 5
    input [6:0] SEG6,       // Segment data for digit 6
    input [6:0] SEG7,       // Segment data for digit 7
    output reg [7:0] AN,    // Anode enable signals
    output reg [6:0] SEG    // Current segment output
);
    reg [2:0] scan_cnt = 0;  // 3-bit counter for scanning

    // Increment scan counter on each clock edge
    always @(posedge CLK_1KHZ) begin
        scan_cnt <= scan_cnt + 1;
    end

    always @(*) begin
        case(scan_cnt)
            3'd0: begin AN = 8'b11111110; SEG = SEG0; end  // Enable digit 0
            3'd1: begin AN = 8'b11111101; SEG = SEG1; end  // Enable digit 1
            3'd2: begin AN = 8'b11111011; SEG = SEG2; end  // Enable digit 2
            3'd3: begin AN = 8'b11110111; SEG = SEG3; end  // Enable digit 3
            3'd4: begin AN = 8'b11101111; SEG = SEG4; end  // Enable digit 4
            3'd5: begin AN = 8'b11011111; SEG = SEG5; end  // Enable digit 5
            3'd6: begin AN = 8'b10111111; SEG = SEG6; end  // Enable digit 6
            3'd7: begin AN = 8'b01111111; SEG = SEG7; end  // Enable digit 7
        endcase
    end
endmodule
```

Simulation:

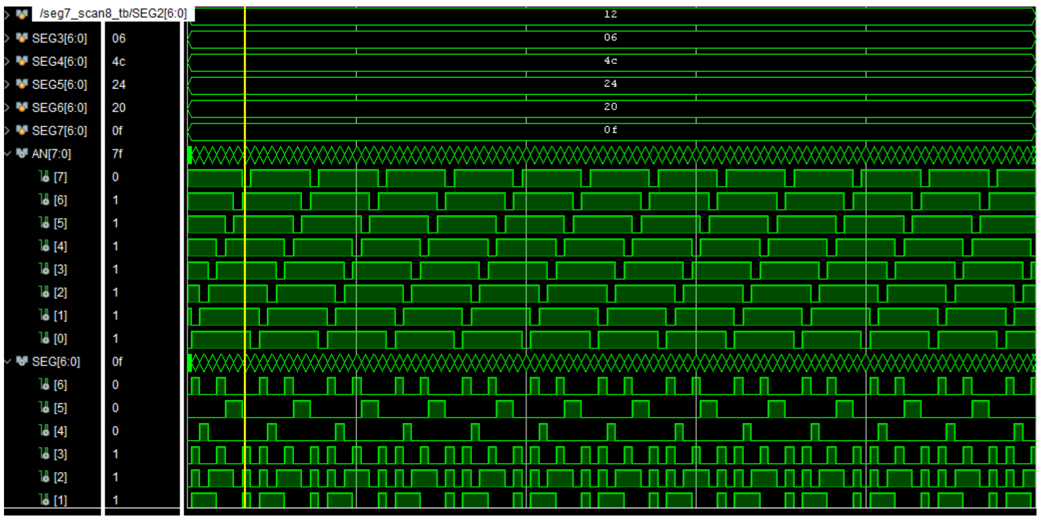
Debounce TB waveform



Barrel Shifter TB waveform

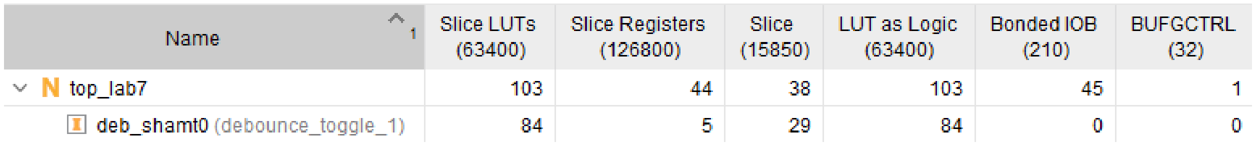


seg7_scan8 testbench

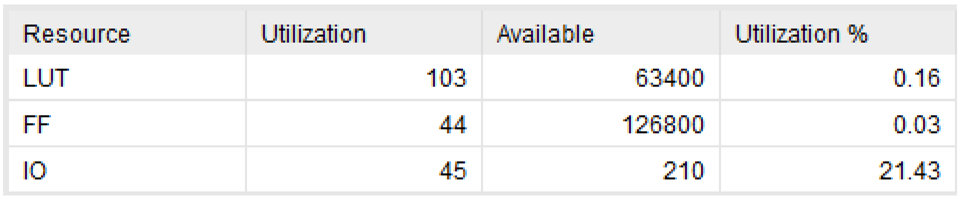


Implementation:

Utilization Table:

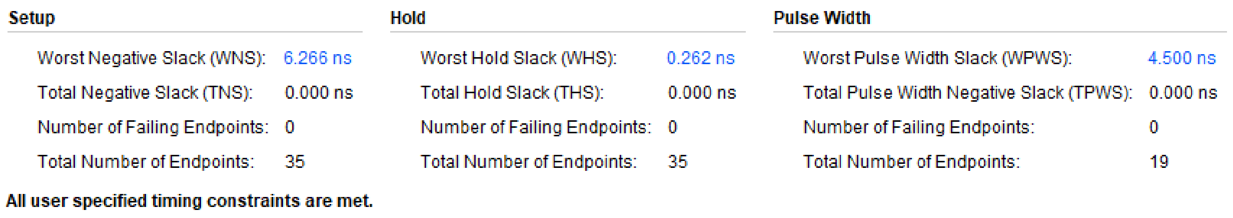


Name	Slice LUTs (63400)	Slice Registers (126800)	Slice (15850)	LUT as Logic (63400)	Bonded IOB (210)	BUFGCTRL (32)
top_lab7	103	44	38	103	45	1
deb_shamt0 (debounce_toggle_1)	84	5	29	84	0	0



Resource	Utilization	Available	Utilization %
LUT	103	63400	0.16
FF	44	126800	0.03
IO	45	210	21.43

Timing Summary



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6.266 ns	Worst Hold Slack (WHS):	0.262 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:	35	Total Number of Endpoints:	35	Total Number of Endpoints:	19

All user specified timing constraints are met.

Video:

https://youtu.be/gB2LnxQ3D1g

Contributions:

Justin Wong: XDC, Driver Code, testbench code, report. 50% for all.
Hector Garibay: XDC, Driver Code, testbench code, report. 50% for all.