3300 Lab 3
16x1 Multiplexer Using Nested 2x1 MUXes with Debounced Toggle Select Control

Group C
Rohan Walia
Parsa Ghasemi
7/2/25

Code:

```
module mux2x1 (
    input a, b,
    input sel,
    output y
);
    wire nsel, a1, b1;
    not (nsel, sel);
    and (a1, a, nsel);
    and (b1, b, sel);
    or (y, a1, b1);
endmodule




module mux16x1 (
    input [15:0] in,
    input [3:0] sel,
    output out
);
    wire [15:0] level1;
    wire [7:0] level2;
    wire [3:0] level3;

    genvar i;
    generate
        for (i = 0; i < 8; i = i + 1)
            mux2x1 m1 (.a(in[2*i]), .b(in[2*i+1]), .sel(sel[0]), .y(level1[i]));
        for (i = 0; i < 4; i = i + 1)
            mux2x1 m2 (.a(level1[2*i]), .b(level1[2*i+1]), .sel(sel[1]), .y(level2[i]));
        for (i = 0; i < 2; i = i + 1)
            mux2x1 m3 (.a(level2[2*i]), .b(level2[2*i+1]), .sel(sel[2]), .y(level3[i]));
        mux2x1 m4 (.a(level3[0]), .b(level3[1]), .sel(sel[3]), .y(out));
    endgenerate
endmodule
```

```
module debounce (
    input clk,
    input btn_in,
    output reg btn_clean
);
    reg [2:0] shift_reg;

    always @(posedge clk) begin
        shift_reg <= {shift_reg[1:0], btn_in};
        if (shift_reg == 3'b111)
            btn_clean <= 1;
        else if (shift_reg == 3'b000)
            btn_clean <= 0;
    end
endmodule
```

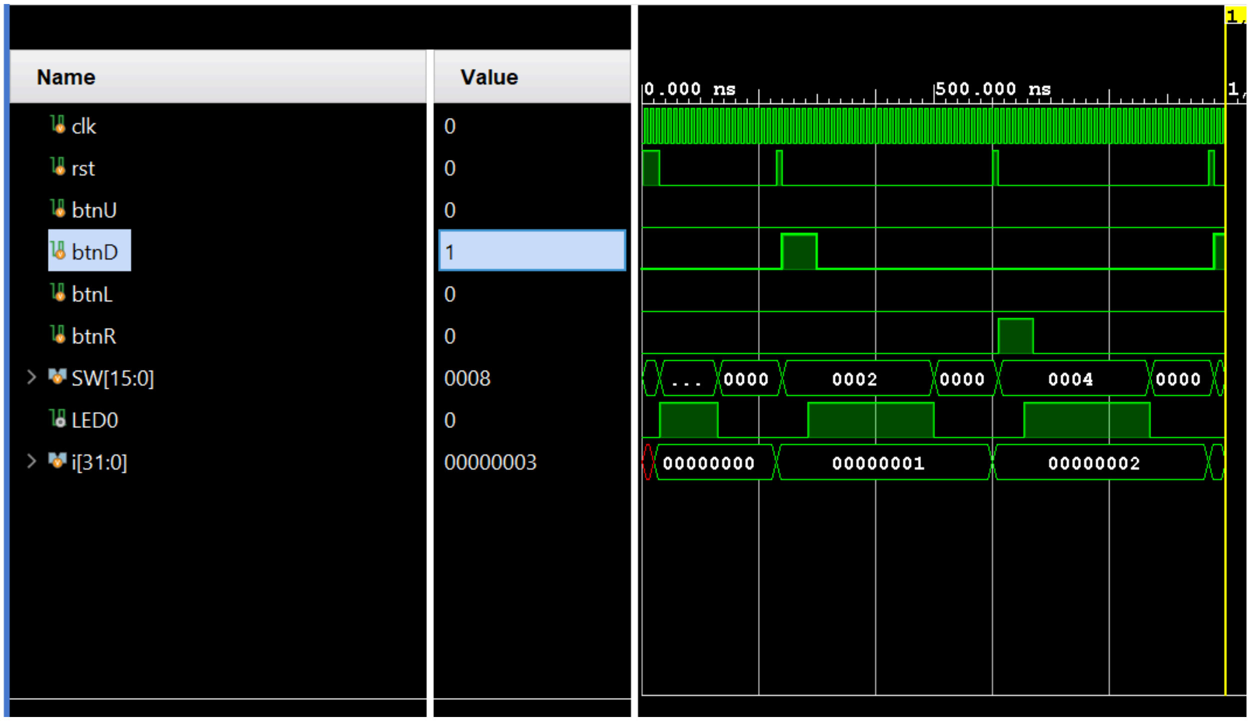
```
module toggle_switch (
    input clk,
    input rst,
    input btn_raw,
    output reg state
);
    wire btn_clean;
    reg btn_prev;

    debounce db (.clk(clk), .btn_in(btn_raw), .btn_clean(btn_clean));

    always @(posedge clk) begin
        if (rst) begin
            state <= 0;
            btn_prev <= 0;
        end else begin
            if (btn_clean && !btn_prev)
                state <= ~state;
            btn_prev <= btn_clean;
        end
    end
endmodule
```

```
module top_mux_lab3 (
    input clk,
    input rst,
    input [15:0] SW,
    input btnU, btnD, btnL, btnR,
    output LED0
);
    wire [3:0] sel;

    toggle_switch t0 (.clk(clk), .rst(rst), .btn_raw(btnD), .state(sel[0]));
    toggle_switch t1 (.clk(clk), .rst(rst), .btn_raw(btnR), .state(sel[1]));
    toggle_switch t2 (.clk(clk), .rst(rst), .btn_raw(btnL), .state(sel[2]));
    toggle_switch t3 (.clk(clk), .rst(rst), .btn_raw(btnU), .state(sel[3]));

    mux16x1 mux (.in(SW), .sel(sel), .out(LED0));
endmodule
```

```
module tb_top_mux_lab3;

    reg clk, rst, btnU, btnD, btnL, btnR;
    reg [15:0] SW;
    wire LED0;

    top_mux_lab3 uut(
        .clk(clk), .rst(rst),
        .btnU(btnU), .btnD(btnD), .btnL(btnL), .btnR(btnR),
        .SW(SW),
        .LED0(LED0)
    );

    // 100 MHz clock
    initial clk = 0;
    always #5 clk = ~clk;

    // Declare integer for loop
    integer i;

    // Tasks for each button
    task pressD; begin btnD = 1; #60; btnD = 0; #100; end endtask
    task pressR; begin btnR = 1; #60; btnR = 0; #100; end endtask
    task pressL; begin btnL = 1; #60; btnL = 0; #100; end endtask
    task pressU; begin btnU = 1; #60; btnU = 0; #100; end endtask

    initial begin
        SW = 0;
        btnU = 0; btnD = 0; btnL = 0; btnR = 0;
        rst = 1; #20; rst = 0;

        for (i = 0; i < 16; i = i + 1) begin
            // Reset selector logic
            rst = 1; #10; rst = 0;

            // Set current switch high
            SW = 16'b1 << i;

            // Press buttons to set sel = i
            if (i[0]) pressD();
            if (i[1]) pressR();
            if (i[2]) pressL();
            if (i[3]) pressU();
            #100;

            $display("Testing sel = %0d | SW[%0d] = %b | LED0 = %b", i, i, SW[i], LED0);

            // Clear switch for next test
            SW = 0;
            #100;
        end

        $finish;
    end

endmodule
```

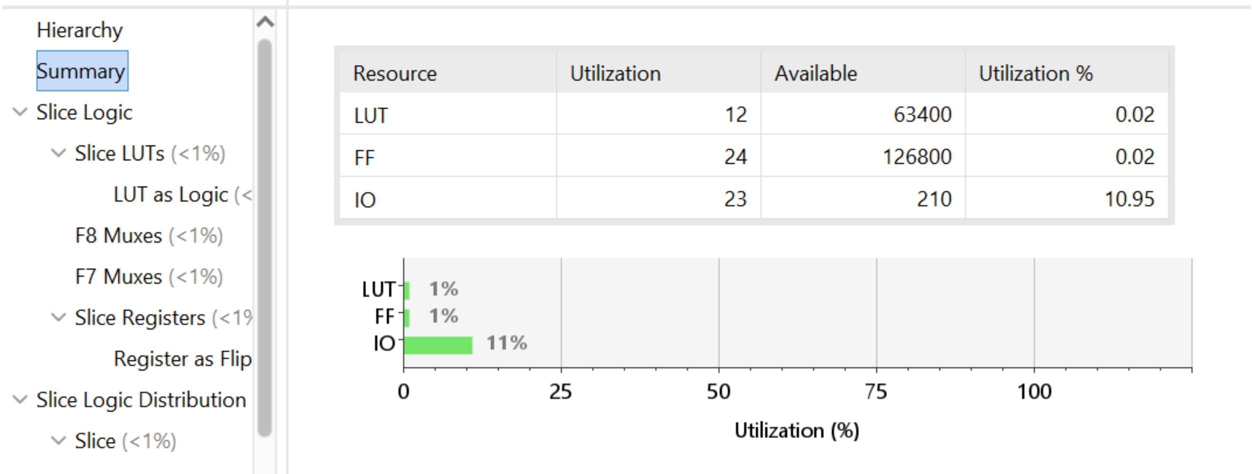
This project implements a 16-to-1 multiplexer on the Nexys A7-100T FPGA using nested 2-to-1 muxes and debounced toggle controls. The design is modular, with a mux2x1 acting as the basic building block. These are nested in the mux16x1 module to select one of 16 input bits (SW[15:0]) based on a 4-bit select signal. The select signal is controlled using four pushbuttons (btnU, btnD, btnL, btnR), each connected to a toggle_switch module that toggles its state on each press, with clean signal edges ensured by the debounce module. The selected input value is displayed on LED0, allowing interactive switching between inputs via buttons. The top_mux_lab3 module integrates all components and connects them to the FPGA's physical interfaces.

Waveform screenshot (testing 0,1,2)



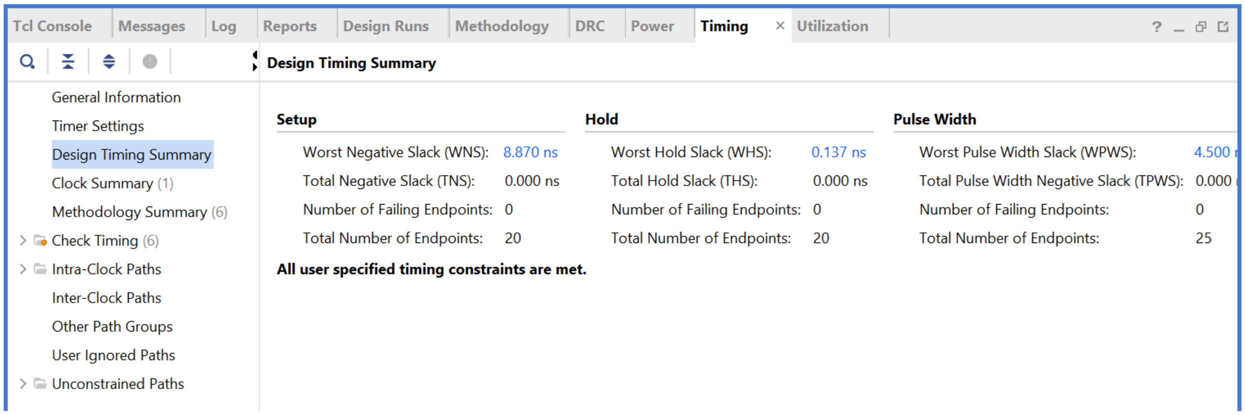
Name	Value
clk	0
rst	0
btnU	0
btnD	1
btnL	0
btnR	0
SW[15:0]	0008
LED0	0
i[31:0]	00000003

LUTs and FF screenshot:



Resource	Utilization	Available	Utilization %
LUT	12	63400	0.02
FF	24	126800	0.02
IO	23	210	10.95

Timing screenshot:

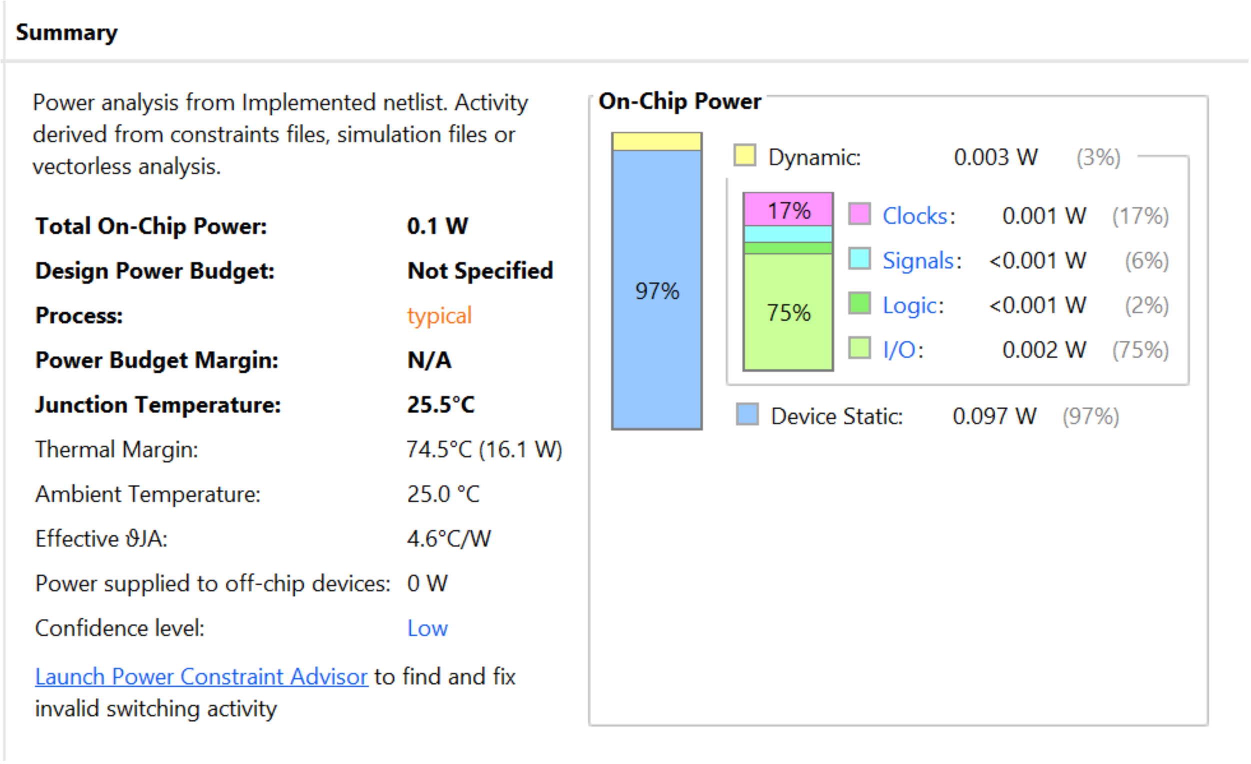


Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	8.870 ns	Worst Hold Slack (WHS):	0.137 ns	Worst Pulse Width Slack (WPWS):	4.500
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	20	Total Number of Endpoints:	20	Total Number of Endpoints:	25

All user specified timing constraints are met.

Power screenshot:



Summary

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

Total On-Chip Power:	**0.1 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.5°C**
Thermal Margin:	74.5°C (16.1 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:	0.003 W	(3%)
Clocks:	0.001 W	(17%)
Signals:	<0.001 W	(6%)
Logic:	<0.001 W	(2%)
I/O:	0.002 W	(75%)
Device Static:	0.097 W	(97%)

Contribution:
Rohan Walia (50%): Implementation, demo, testing, report
Parsa Ghasemi (50%): testbench code, verilog, report

Demo link: https://youtu.be/ATRwbOBZEgo