Cal Poly Pomona

College of Engineering

California Polytechnic State University, Pomona

ECE3300L

Experiment #3

GROUP K
Hoang, Dalton - 016062800
Siu, Andy - 016205137

June 30th, 2025

Introduction: In this lab, we designed and implemented a 16-to-1 multiplexer (MUX16x1) using gate-level 2-to-1 multiplexers in Verilog. The 16 data inputs were controlled via slide switches (SW[15:0]), and the 4-bit select line was generated using debounced toggle switches implemented with pushbuttons (btnU, btnD, btnL, btnR). A debouncing mechanism was included to ensure reliable toggle behavior of the buttons and eliminate unintended triggers.

Design (Debounce): Removes misinputs from button presses by using a 3-bit shift register to remove unstable transitions.

```
module debounce(
    input clk,
    input btn_in,
    output reg btn_clean
);
    reg [2:0] shift_reg;
    initial begin
        shift_reg = 3'b000;
    end

    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
```

Design(Mux2x1): 2x1 multiplexer using logic gates (NOT, AND, OR)

```
module mux2x1(
    input a,
    input 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
```

Design(Mux16x1): 16x1 multiplexer using multiple 2x1 mux modules in loops for each bit level

```
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
```

Design(toggle_switch): Toggles the output between 0 and 1 each time the debounced button is pressed on rising edge.

```
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
```

Design(Top Mux): Combines all modules, implements inputs and connects outputs to them

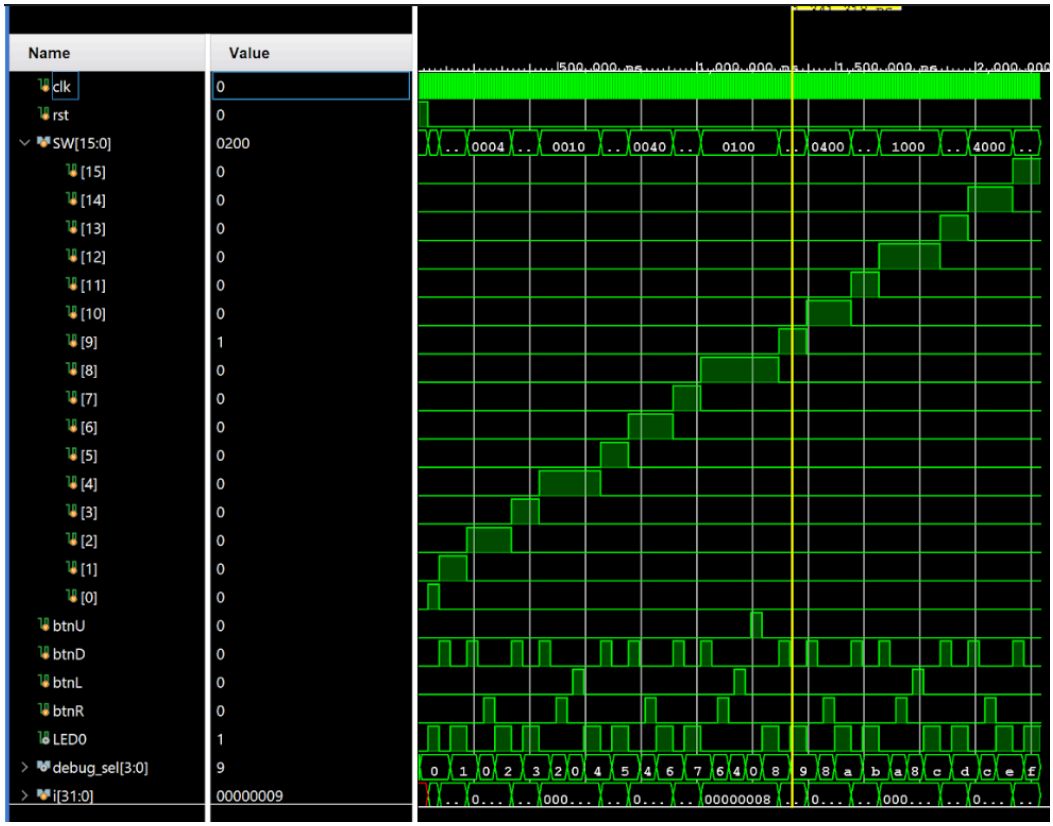
```
module top_mux_lab3 (
    input clk,
    input rst,
    input [15:0] SW,
    input btnU, btnD, btnL, btnR,
    output LED0,
    output [3:0] debug_sel
);
    wire [3:0] sel;
    assign debug_sel = sel;

    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
```

Simulation(tb_mux_lab3):

```
`timescale 1ns / 1ps

module tb_top_mux_lab3;

  // DUT inputs
  reg clk = 0;
  reg rst = 1;
  reg [15:0] SW = 16'b0;
  reg btnU = 0, btnD = 0, btnL = 0, btnR = 0;

  // DUT outputs
  wire LED0;
  wire [3:0] debug_sel;

  // Instantiate the DUT
  top_mux_lab3 dut (
    .clk(clk),
    .rst(rst),
    .SW(SW),
    .btnU(btnU),
    .btnD(btnD),
    .btnL(btnL),
    .btnR(btnR),
    .LED0(LED0),
    .debug_sel(debug_sel)
  );

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

  // Task to press a button (held for 4 clock cycles to pass debounce)
  task press(input integer dir);
    begin
      case (dir)
        0: btnD = 1;
        1: btnR = 1;
        2: btnL = 1;
        3: btnU = 1;
      endcase
```

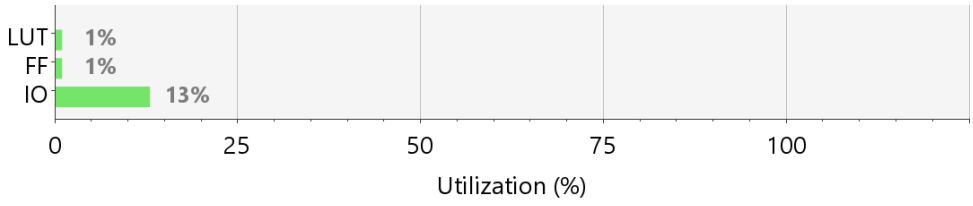
```
      repeat (2) @(posedge clk);  // debounce low time
    end
  endtask

  // Task to set the select bits to a target value using toggle switches
  task set_sel(input [3:0] target);
    begin
      if (debug_sel[0] !== target[0]) press(0); // btnD
      if (debug_sel[1] !== target[1]) press(1); // btnR
      if (debug_sel[2] !== target[2]) press(2); // btnL
      if (debug_sel[3] !== target[3]) press(3); // btnU (MSB)
    end
  endtask

  // Main test
  integer i;
  initial begin
    // Apply reset
    repeat (4) @(posedge clk);
    rst = 0;

    // Test each select value 0-15
    for (i = 0; i < 16; i = i + 1) begin
      // Set only SW[i] = 1, rest = 0
      SW = 16'b0;
      SW[i] = 1'b1;

      // Use buttons to set sel = i
      set_sel(i[3:0]);

      // Wait for MUX to propagate
      repeat (4) @(posedge clk);


      // Check if LED0 matches SW[i]
      if (LED0 !== 1'b1)
        $fatal(1, "SEL=%0d | Expected LED0=1 (SW[%0d]), got %b", i, i, LED0);
      else
        $display("SEL=%0d | LED0 = SW[%0d] = 1", i, i);
    end

    $display("All 16 select values tested successfully.");
    $finish;
  end

endmodule
```

This Verilog testbench verifies the functionality of the 16 to 1 multiplexer module by simulating user input through button presses to the 4-bit selector and checking that the output (LED0) correctly corresponds to the value of the selected input switch (SW[i]). It generates a 100 Mhz clock, applies a reset, and then sets each of the 16 values by toggling the buttons with debounce handling. In the waveform image below, we can see the simulation output testing SW[0] through SW[15]. The debug_sel bits change accordingly through simulated button presses (btnD, btnR, btnL, btnU), incrementing the select value. The output LED0 goes high (1) only when the selected input (SW[i]) is set to 1, confirming correct MUX behavior.



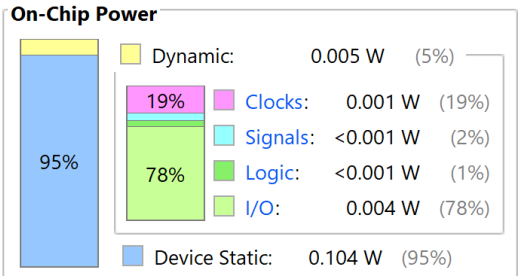
Implementation:

Resource	Utilization	Available	Utilization %
LUT	12	63400	0.02
FF	24	126800	0.02
IO	27	210	12.86



Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.11 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**



Video Link:
https://youtu.be/A48sYE9TMlQ

Contributions:

Andy Siu: 50% (mux16x1.v, mux2x1.v, toggle_switch.v, top_mux.v, .xdc, demo, implementation, report)

Dalton Hoang: 50% (debounce.v, tb_mux_lab3.v, testbench png, report)