ECE 3300 – Lab 2 Report
16x1 Multiplexer Design

Team Name: Group V

Date: July 2, 2025

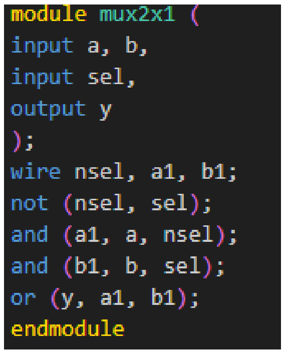
Group Members

Nathan Marlow, Khristian Chan

1. Objective

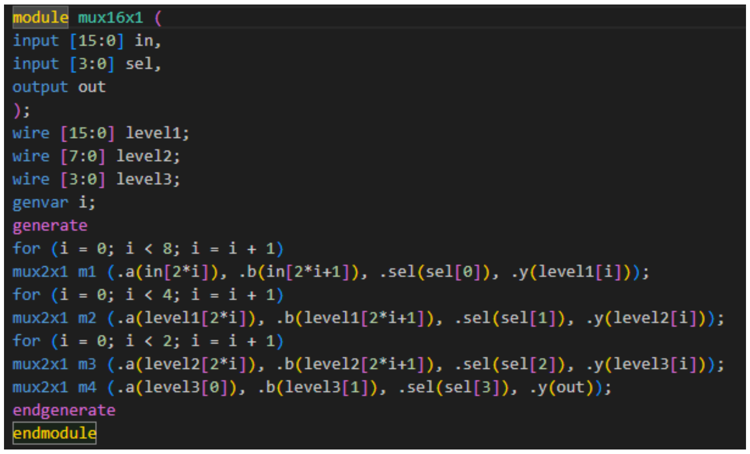
The objective of this lab was to implement a 16x1 multiplexer with a debouncing circuit and a toggle flip flop for each select bit. Additionally, we were required to create a testbench that encapsulates not only the multiplexer, but also the debouncing and toggle flip flop circuits.

2. Verilog Code



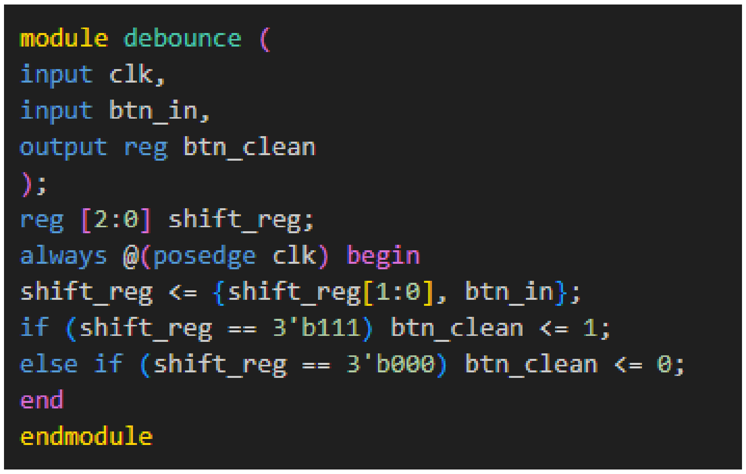
```
module mux2x1 (
input a, b,
input sel,
output y
);
wire nsel, a1, b1;
not (nsel, sel);
and (a1, a, nsel);
and (b1, b, sel);
or (y, a1, b1);
endmodule
```

Mux2x1: logic gate based mux



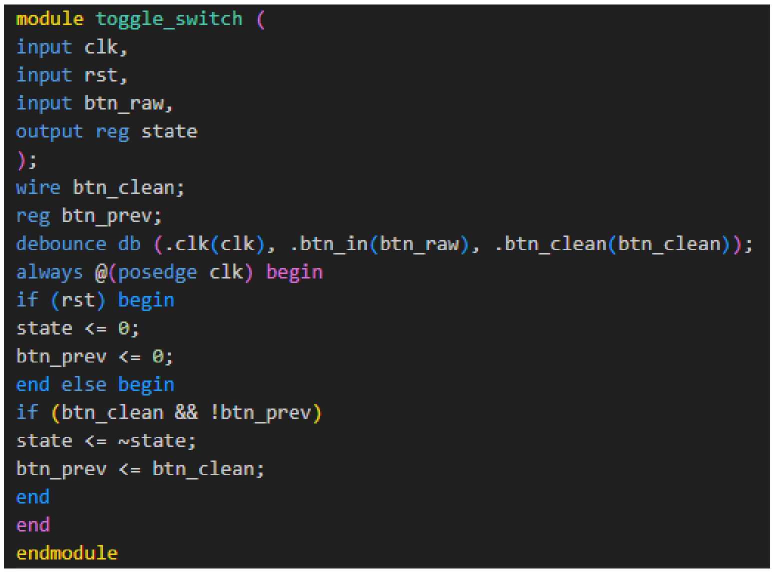
```
module mux16x1 (
input [15:0] in,
input [3:0] sel,
output out
);
wire [15:0] level1;
wire [7:0] level2;
wire [3:0] level3;
genvar i;
generate
for (i = 0; i < 8; i = i + 1)
mux2x1 m1 (.a(in[2*i]), .b(in[2*i+1]), .sel(sel[0]), .y(level1[i]));
for (i = 0; i < 4; i = i + 1)
mux2x1 m2 (.a(level1[2*i]), .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
```

Mux16x1: 16x1 mux using multiple 2x1 muxes with generate loops



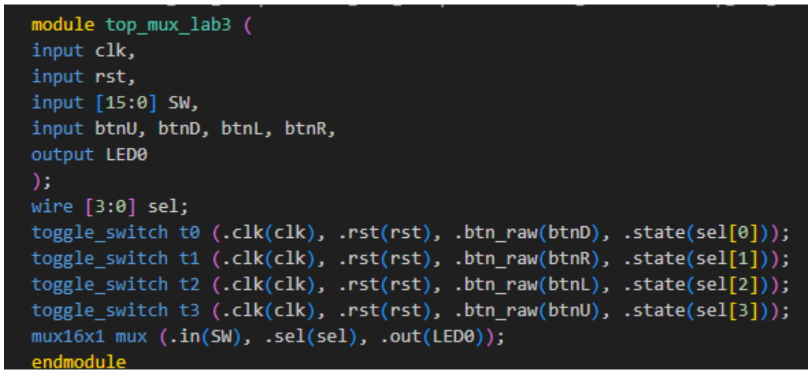
```
module debounce (
input clk,
input btn_in,
output reg btn_clean
);
reg [2:0] shift_reg;
always @(posedge clk) begin
shift_reg <= {shift_reg[1:0], btn_in};
if (shift_reg == 3'b111) btn_clean <= 1;
else if (shift_reg == 3'b000) btn_clean <= 0;
end
endmodule
```

Debounce: button output goes high only if high in the last 3 cycles, else output=0



```
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
```

Toggle_switch: Toggle FF which resets when reset signal goes high, stays the same if reset is not pressed.

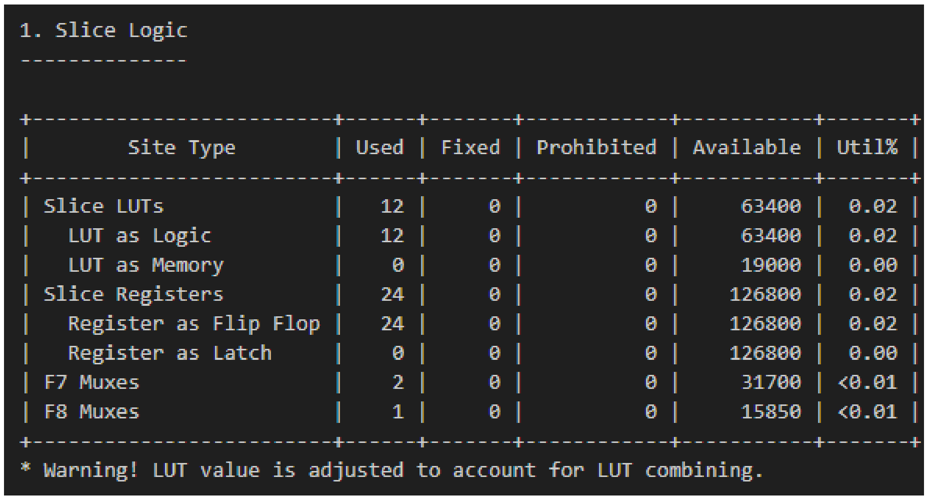


```
module top_mux_lab3 (
input clk,
input rst,
input [15:0] SW,
input btnU, btnD, btnL, btnR,
output LED0
);
wire [3:0] sel;
toggle_switch t0 (.clk(clk), .rst(rst), .btn_raw(btnD), .state(sel[0]));
toggle_switch t1 (.clk(clk), .rst(rst), .btn_raw(btnR), .state(sel[1]));
toggle_switch t2 (.clk(clk), .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
```

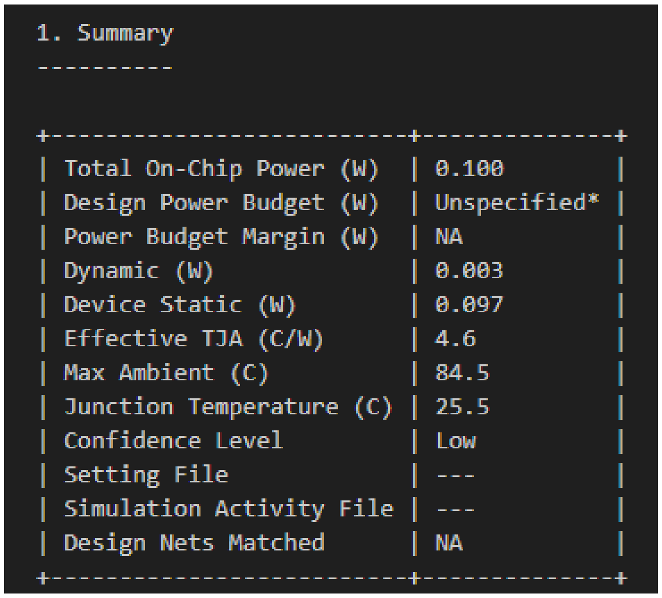
Top_Module_Lab3: integrates toggle FF and debounce modules into the 16x1 mux

3. Implementation Results

After synthesizing and simulating the design in Vivado, the following results were observed:



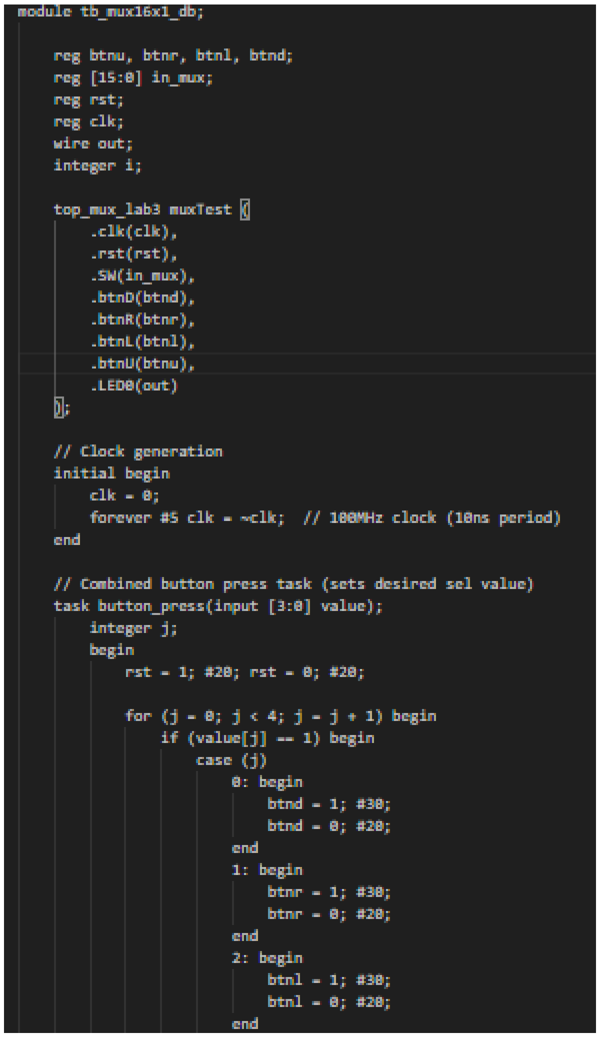
```
1. Slice Logic
--------------

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs              |   12 |     0 |          0 |     63400 |  0.02 |
|   LUT as Logic          |   12 |     0 |          0 |     63400 |  0.02 |
|   LUT as Memory         |    0 |     0 |          0 |     19000 |  0.00 |
| Slice Registers         |   24 |     0 |          0 |    126800 |  0.02 |
|   Register as Flip Flop |   24 |     0 |          0 |    126800 |  0.02 |
|   Register as Latch     |    0 |     0 |          0 |    126800 |  0.00 |
| F7 Muxes                |    2 |     0 |          0 |     31700 | <0.01 |
| F8 Muxes                |    1 |     0 |          0 |     15850 | <0.01 |
+-------------------------+------+-------+------------+-----------+-------+
* Warning! LUT value is adjusted to account for LUT combining.
```



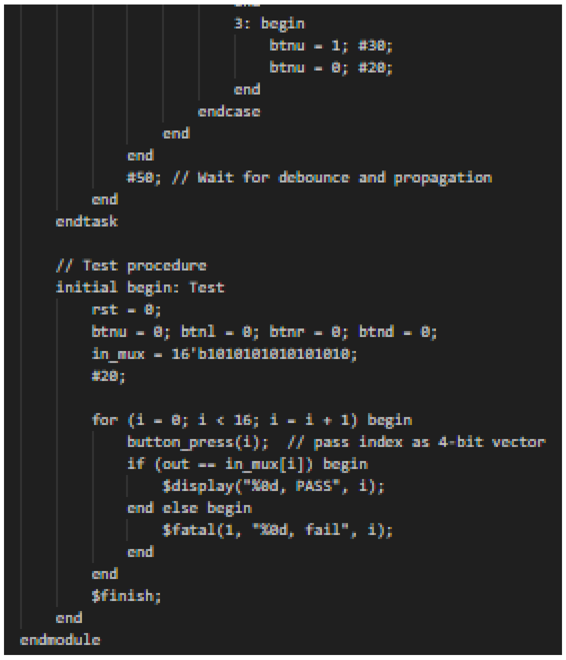
```
1. Summary
----------

+--------------------------+--------------+
| Total On-Chip Power (W)  | 0.100        |
| Design Power Budget (W)  | Unspecified* |
| Power Budget Margin (W)  | NA           |
| Dynamic (W)              | 0.003        |
| Device Static (W)        | 0.097        |
| Effective TJA (C/W)      | 4.6          |
| Max Ambient (C)          | 84.5         |
| Junction Temperature (C) | 25.5         |
| Confidence Level         | Low          |
| Setting File             | ---          |
| Simulation Activity File | ---          |
| Design Nets Matched      | NA           |
+--------------------------+--------------+
```

Simulation Results:

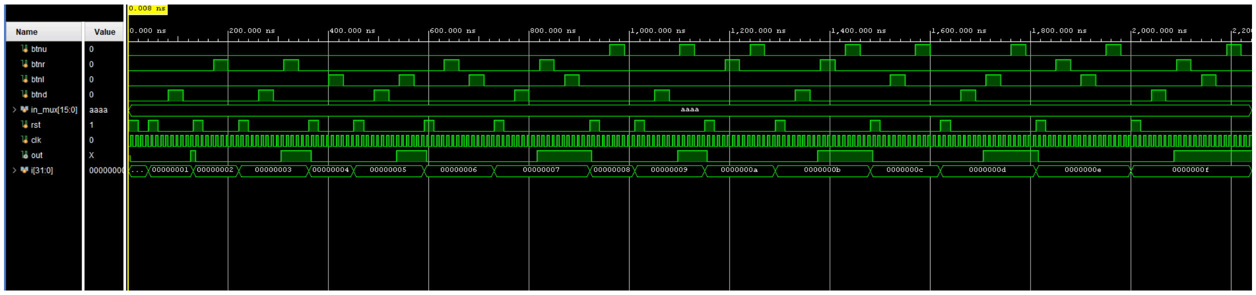
The testbench code, simulation, and log are all shown below. Our testbench consists of our normal variable creation, DUT, and clock creation. Since each button is actually a logic block, we cannot drive the input wire to our top level module by setting our select array manually. Instead, we have to create a task which will take simulated button presses and wait for debounce. We then pass the values of the pressed buttons to our main for loop which will go through all 16 select possibilities. We see that the testbench goes through every single value without a failed test.



```verilog
module tb_mux16x1_db;

    reg btnu, btnr, btnl, btnd;
    reg [15:0] in_mux;
    reg rst;
    reg clk;
    wire out;
    integer i;

    top_mux_lab3 muxTest (
        .clk(clk),
        .rst(rst),
        .SW(in_mux),
        .btnD(btnd),
        .btnR(btnr),
        .btnL(btnl),
        .btnU(btnu),
        .LED0(out)
    );

    // Clock generation
    initial begin
        clk = 0;
        forever #5 clk = ~clk;  // 100MHz clock (10ns period)
    end

    // Combined button press task (sets desired sel value)
    task button_press(input [3:0] value);
        integer j;
        begin
            rst = 1; #20; rst = 0; #20;

            for (j = 0; j < 4; j = j + 1) begin
                if (value[j] == 1) begin
                    case (j)
                        0: begin
                            btnd = 1; #30;
                            btnd = 0; #20;
                        end
                        1: begin
                            btnr = 1; #30;
                            btnr = 0; #20;
                        end
                        2: begin
                            btnl = 1; #30;
                            btnl = 0; #20;
                        end
```



```verilog
                        3: begin
                            btnu = 1; #30;
                            btnu = 0; #20;
                        end
                    endcase
                end
            end
            #50; // Wait for debounce and propagation
        end
    endtask

    // Test procedure
    initial begin: Test
        rst = 0;
        btnu = 0; btnl = 0; btnr = 0; btnd = 0;
        in_mux = 16'b1010101010101010;
        #20;

        for (i = 0; i < 16; i = i + 1) begin
            button_press(i);  // pass index as 4-bit vector
            if (out == in_mux[i]) begin
                $display("%0d, PASS", i);
            end else begin
                $fatal(1, "%0d, fail", i);
            end
        end
        $finish;
    end
endmodule
```



```
Time resolution is 1 ps
0, PASS
1, PASS
2, PASS
3, PASS
4, PASS
5, PASS
6, PASS
7, PASS
8, PASS
9, PASS
10, PASS
11, PASS
12, PASS
13, PASS
14, PASS
15, PASS
$finish called at time : 2240 ns : File "C:/Users/khris/3300L_Lab3_GroupV/3300L_Lab3_GroupV.srcs/sim_1/new/tb_mux16x1_db.v" Line 98
```

6. Demonstration Video

https://youtu.be/JIuWvTNwvos

Contributions:

Khristian Chan- 50%: Testbench, Simulation, Report
Nathan Marlow- 50%: Implementation, XDC, Demo, Report