Julio Flores (ID# : 016326856)
Victor Perez (ID# : 016196050)

Group I

Session E02

Lab 3

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

WEDNESDAY

JULY 2, 2025

ECE 3300L

Summer 2025

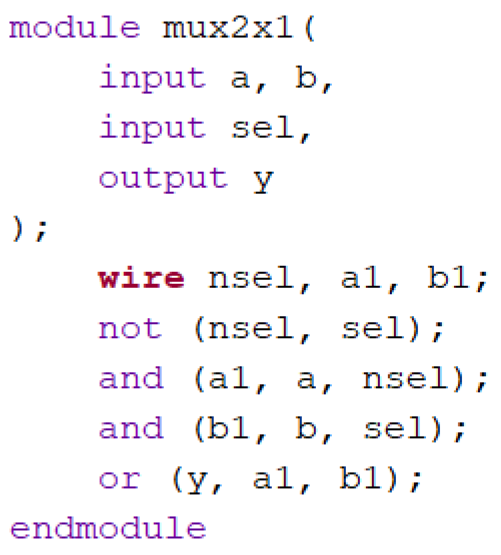
Objective:

The objective of this lab is to design and implement a 16-to-1 multiplexer using gate-level coding in Verilog by building it from 2-to-1 multiplexers. This lab incorporates toggle logic by using the Nexys A7-100T FPGA board's pushbuttons to control the select lines to behave like switches. The 16 input lines are connected to the board's switches through the (SW[15:0]), and the selected output is displayed on LED0. The design will be verified using testbenches and simulation in Vivado.

Design

Gate level code for 2x1 Mux:

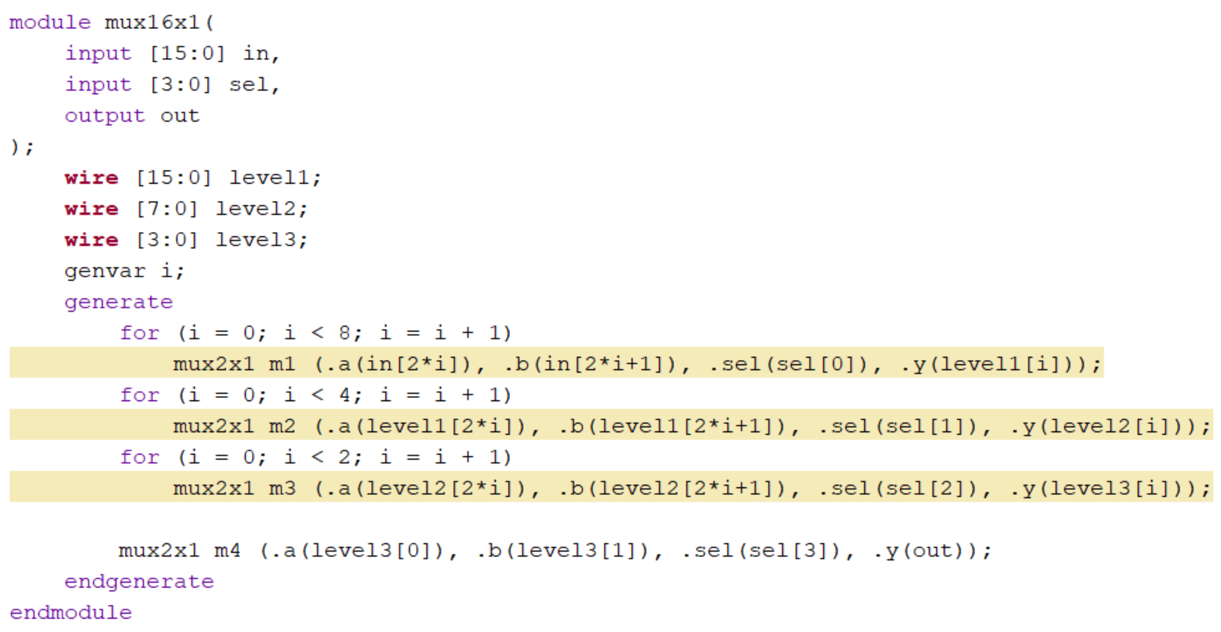
-Addressing the inputs/outputs for the 2x1 mux.



```
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
```

16x1 Mux using loops:

-By using the module done before from the 2x1 mux, we are able to run a for-loop for each level of the mux.

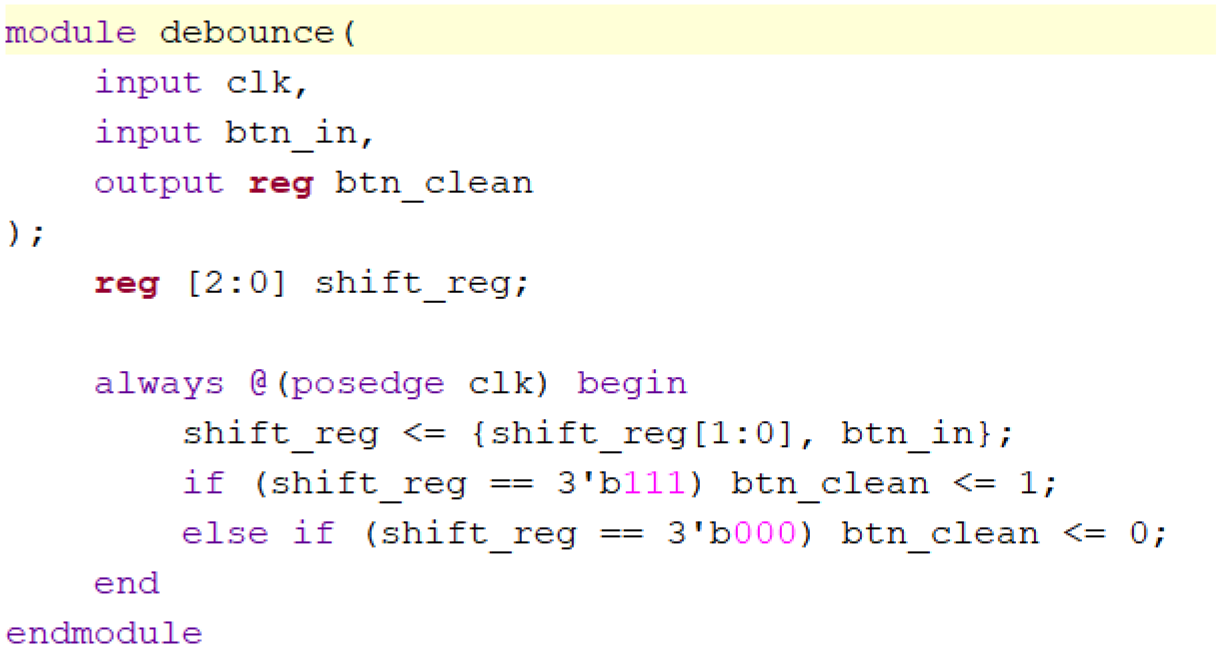


```
module mux16x1(
    input [15:0] in,
    input [3:0] sel,
    output out
);
    wire [15:0] level1;
    wire [7:0] level2;
    wire [3:0] level3;
    genvar i;
    generate
        for (i = 0; i < 8; i = i + 1)
            mux2x1 m1 (.a(in[2*i]), .b(in[2*i+1]), .sel(sel[0]), .y(level1[i]));
    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

Debounce Module:

- This code help in debounce the signal of the button so it can get a clear number be getting the btn_clean to be 1 when the input signal is all 1111.

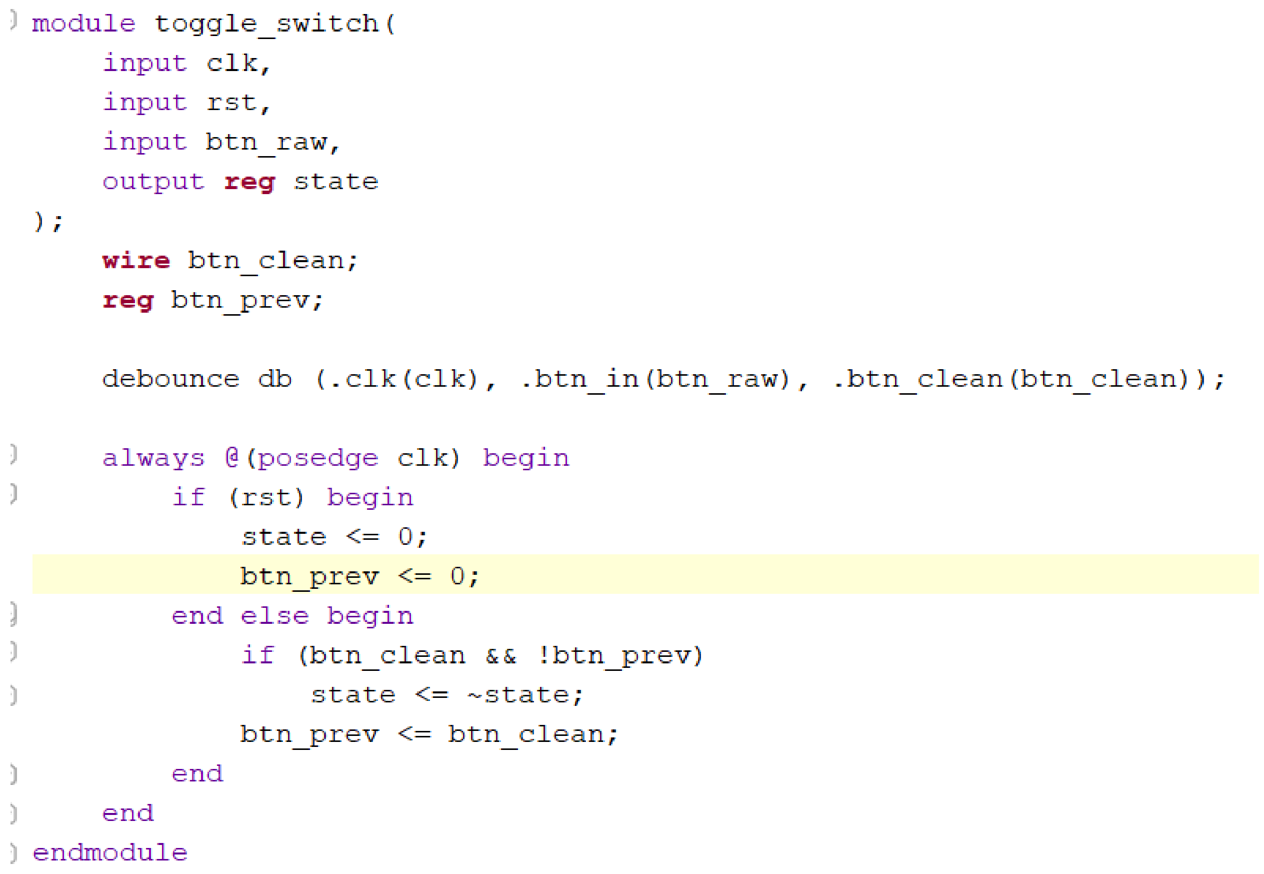


```
module debounce(
    input clk,
    input btn_in,
    output reg btn_clean
);
    reg [2:0] shift_reg;

    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
```

Toggle Flip-Flop:

This section of the code implements the switches from the board to work correctly with the debounce button module. It uses if-else statements to toggle on or off the switches.



```
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
```

Top-Level

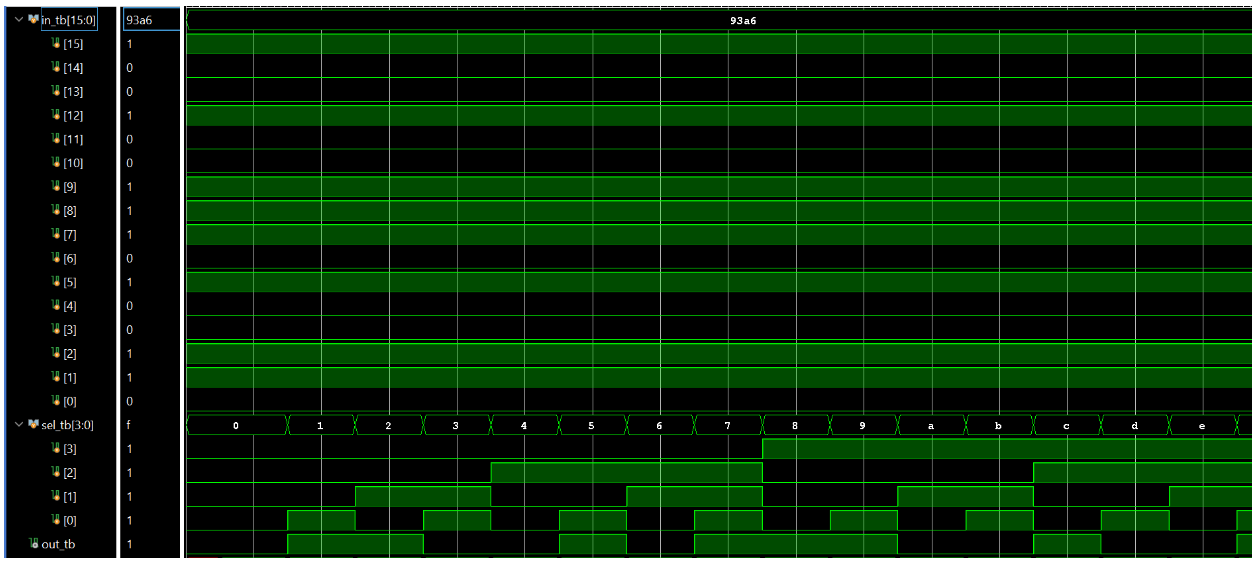
- Combines all of the modules with the by using the toggle switch to verify if the button select is the same as the input lines.

```
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
```

Teshbench and Waveform



```
module tb_mux16x1;

    reg [15:0] in_tb;
    reg [3:0] sel_tb;
    wire out_tb;

    mux16x1 DUT (
        .in(in_tb),
        .sel(sel_tb),
        .out(out_tb)
    );

    integer i;

    initial begin
        in_tb = 16'b1001_0011_1010_0110;
        sel_tb = 4'b0000;

        #5;

        for (i = 0; i < 16; i = i + 1) begin
            sel_tb = i;
            #10;

            if (out_tb !== in_tb[i])
                $fatal(1, "FAIL: sel=%0d | Expected=%b | Got=%b", i, in_tb[i], out_tb);
            else
                $display("PASS: sel=%0d | out=%b", i, out_tb);
        end

        $display("All tests completed successfully.");
        $finish;
    end

endmodule
```

Implementation

```
+-------------------------+------+-------+------------+-----------+-------+
|        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 |
+-------------------------+------+-------+------------+-----------+-------+
```

Contributions:

Julio Flores: 50% - Verilog Code (Constraints Code) and Test demo and report

Victor Perez : 50% - Verilog Code (Testbench Code) and Waveform and report

Reflection:

This lab provided a comprehensive experience in verilog and FPGA implementation by designing a 16 to 1 MUX with a debouncing push button implementation. Seeing the final output accurately reflected on LED0 based on switch inputs confirmed that all components were working together correctly.