ECE 3300L Lab 3 - 16x1 Multiplexer Using Nested 2x1 MUX with Debounced Toggle Select Control

By: Raj Gokidi and Priyanka Ravinder

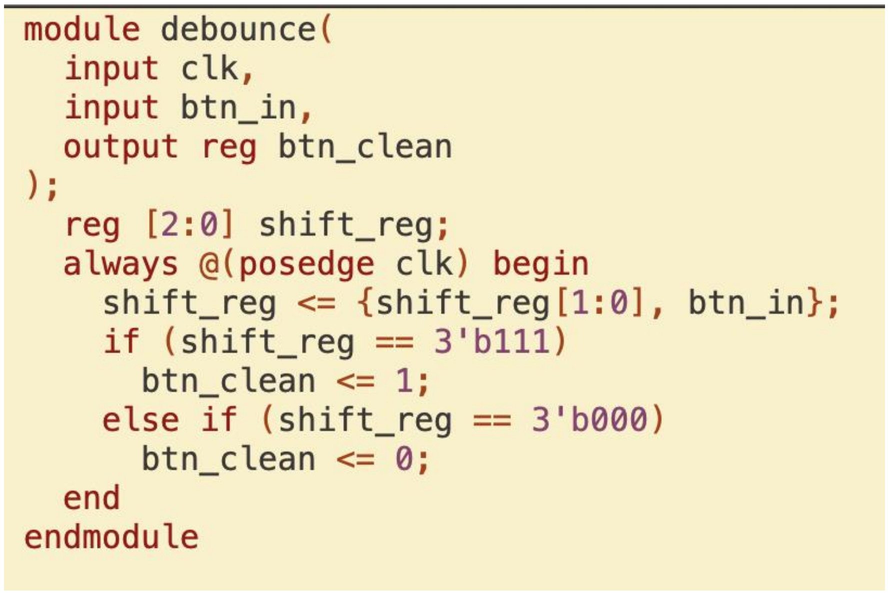
California State Polytechnic University, Pomona

July 2nd, 2025

Introduction

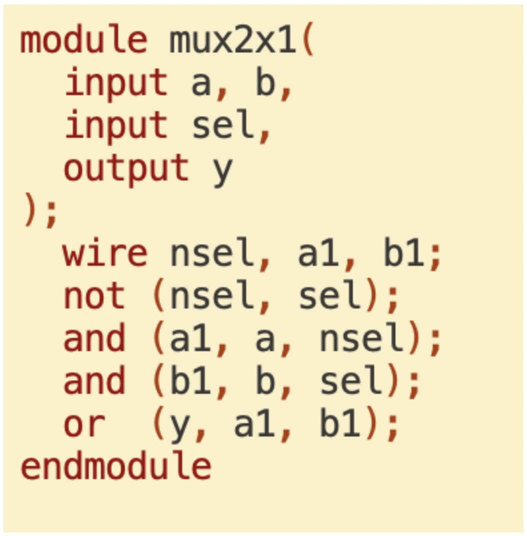
The objective of this lab is to design a 16-to-1 multiplexer using nested 2x1 multiplexers in Verilog and implement it on the Nexys A7 FPGA board. We will create a debouncing circuit and toggle flip-flops to control the multiplexer's select lines using pushbuttons that act like switches. The final design will display the selected input from 16 switches on an LED, with a testbench and lab report documenting the functionality and performance.

Verilog Code



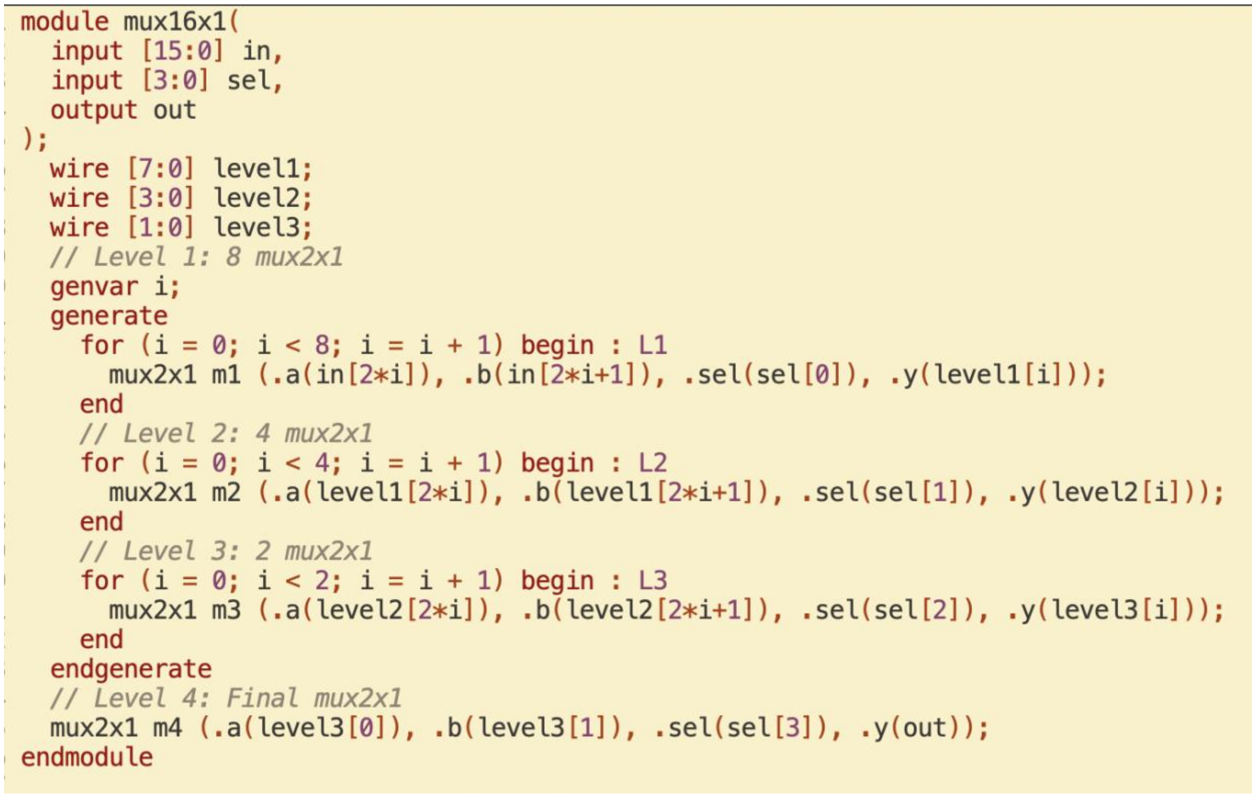
```verilog
module debounce(
  input clk,
  input btn_in,
  output reg btn_clean
);
  reg [2:0] shift_reg;
  always @(posedge clk) begin
    shift_reg <= {shift_reg[1:0], btn_in};
    if (shift_reg == 3'b111)
      btn_clean <= 1;
    else if (shift_reg == 3'b000)
      btn_clean <= 0;
  end
endmodule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

The debounce module filters noisy pushbutton signals to produce a clean, stable output by checking for consistent input over multiple clock cycles.



```verilog
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
```

The mux2x1 module implements a 2-to-1 multiplexer in Verilog using gate-level logic to select one of two inputs based on a select signal.



```verilog
module mux16x1(
  input [15:0] in,
  input [3:0] sel,
  output out
);
  wire [7:0] level1;
  wire [3:0] level2;
  wire [1:0] level3;
  // Level 1: 8 mux2x1
  genvar i;
  generate
    for (i = 0; i < 8; i = i + 1) begin : L1
      mux2x1 m1 (.a(in[2*i]), .b(in[2*i+1]), .sel(sel[0]), .y(level1[i]));
    end
    // Level 2: 4 mux2x1
    for (i = 0; i < 4; i = i + 1) begin : L2
      mux2x1 m2 (.a(level1[2*i]), .b(level1[2*i+1]), .sel(sel[1]), .y(level2[i]));
    end
    // Level 3: 2 mux2x1
    for (i = 0; i < 2; i = i + 1) begin : L3
      mux2x1 m3 (.a(level2[2*i]), .b(level2[2*i+1]), .sel(sel[2]), .y(level3[i]));
    end
  endgenerate
  // Level 4: Final mux2x1
  mux2x1 m4 (.a(level3[0]), .b(level3[1]), .sel(sel[3]), .y(out));
endmodule
```

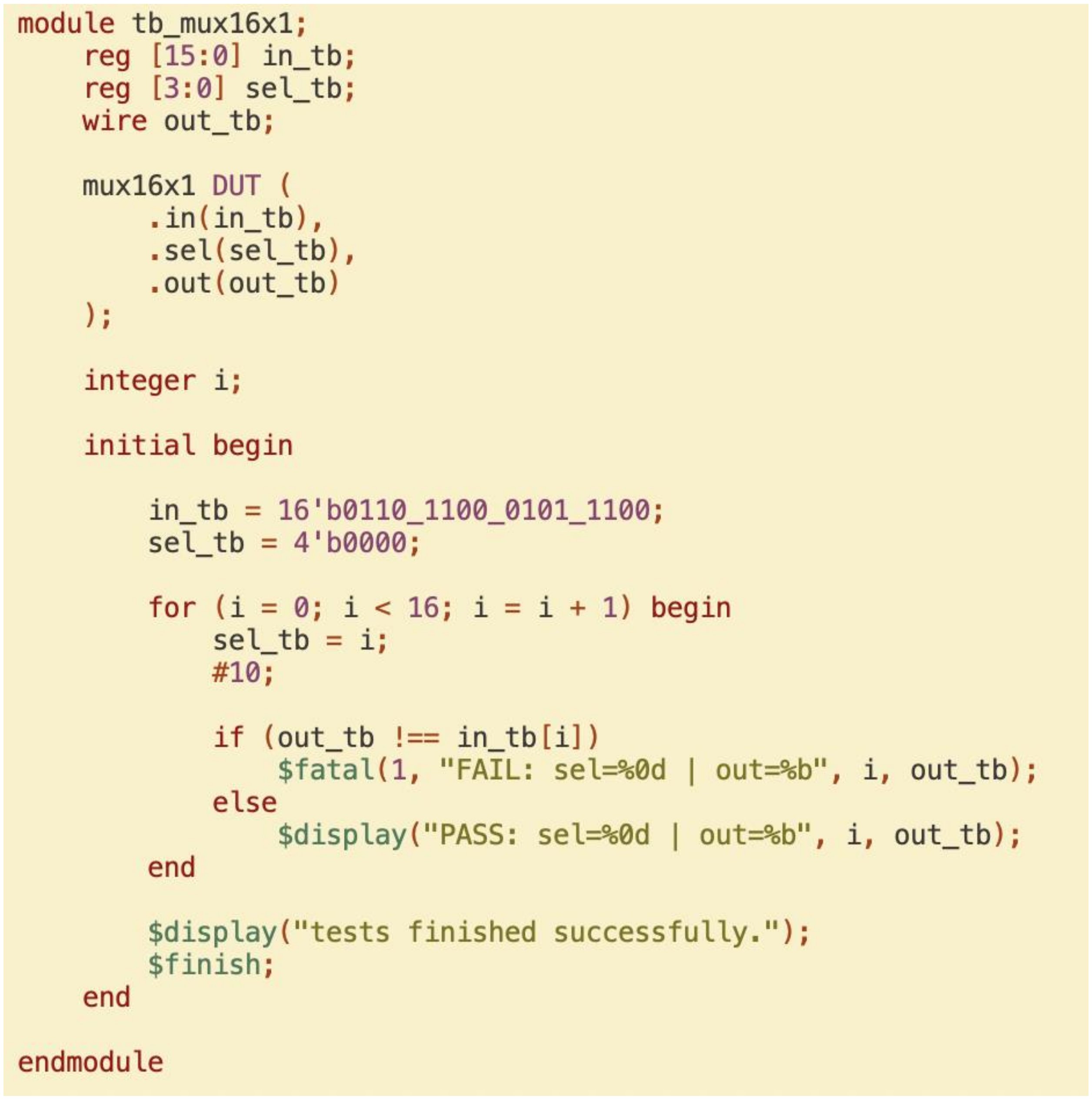
The mux16x1 module constructs a 16-to-1 multiplexer by nesting multiple 2x1 multiplexers to choose one of 16 switch inputs for output on an LED.

```
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
```

The toggle_switch module uses a debounced button input to toggle a flip-flop state, controlling a single select bit for the multiplexer.

```
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
```

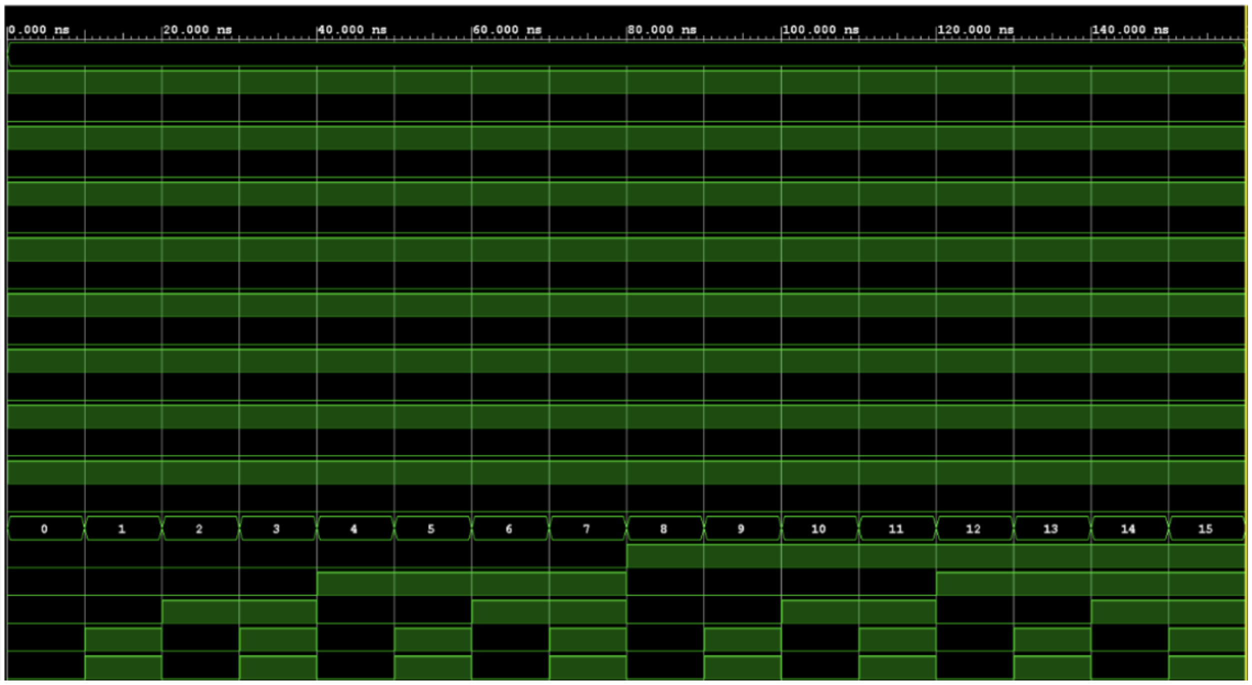
The top_mux_lab3 module integrates the toggle switches, multiplexer, and clock to manage input selection and display the result on the FPGA’s LED.



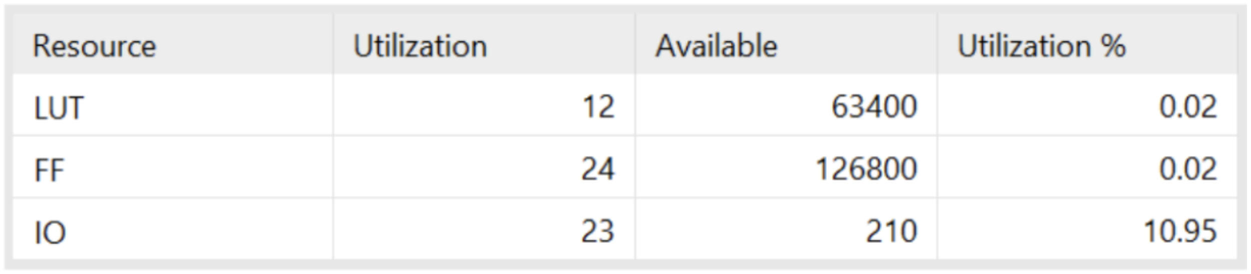
```
module tb_mux16x1;
    reg [15:0] in_tb;
    reg [3:0] sel_tb;
    wire out_tb;

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

    integer i;

    initial begin

        in_tb = 16'b0110_1100_0101_1100;
        sel_tb = 4'b0000;

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

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

        $display("tests finished successfully.");
        $finish;
    end

endmodule
```

The tb_mux16x1 module tests the 16-to-1 multiplexer by applying a fixed 16-bit input pattern, cycling through all select line combinations, and verifying the output matches the expected input bit in a Verilog simulation.

Waveform from Test Bench



LUT and Utilization



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

Partner Contributions

Priyanka Ravinder: Verilog Code, Demo 50%

Raj Gokidi: Testbench, Simulation, Report 50%

Youtube link: https://youtu.be/pP9wt5KdOU4

Demo link: https://youtu.be/cvyDauht9zE (for some reason the original link's demo is very blurry)