# California Polytechnic State University Pomona

Department of Electrical & Computer Engineering

Digital Circuit Design Lab Verilog

ECE 3300L

Lab Report #3

**Experiment #3**

Presented By: Kobe Aquino (StudentID: 015266433)

& Daniel Mondragon Xicotencatl (StudentID: 012803856)

Presented to Mohamed Aly

July 2nd, 2025

## Explanation:

We will design a 16-to-1 multiplexer (MUX16x1) using gate-level 2x1 multiplexers in Verilog. We will control the MUX using the push buttons on the Nexys A7 board, which must behave like switches using toggle logic. This requires implementing a debouncing system and a toggle flip-flop for each select bit. The output of the multiplexer will be shown on LED0, while the inputs will be fed from the 16 switches (SW[15:0])

## Code:

```
module mux2x1(
    input a,b,
    input sel,
    output y
    );
    wire nsel, a1, b1; // notSelect

    not (nsel, sel); //when sel = 1, nsel = 0
    and (a1, a, nsel); // when nsel = 1 AND a, then a1 = a.
    and (b1, b, sel); //when sel = 0 AND b, then b = 0
    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 (
```

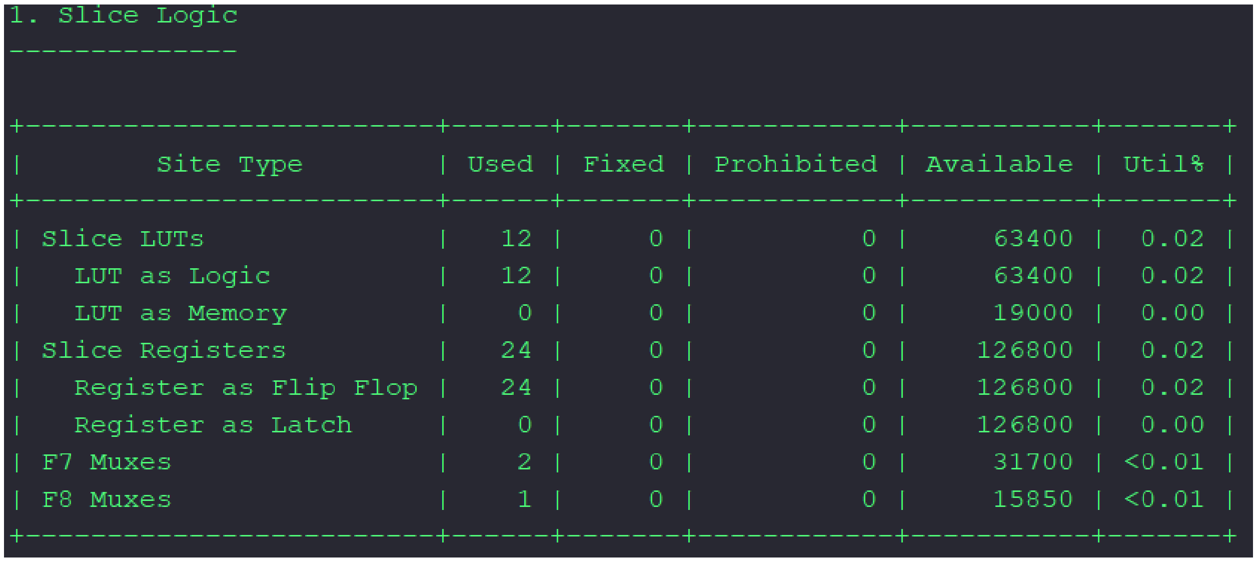
```verilog
    input clk,
    input rst,
    input btn_raw,
    output reg state
);
    wire btn_clean;
    reg btn_prev;
    debounce db (.clk(clk), .btn_in(btn_raw), .btn_clean(btn_clean));
    always @(posedge clk) begin
        if (rst) begin
            state <= 0;
            btn_prev <= 0;
        end else begin
            if (btn_clean && !btn_prev)
                state <= ~state;
            btn_prev <= btn_clean;
        end
    end
endmodule
```

```verilog
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
```

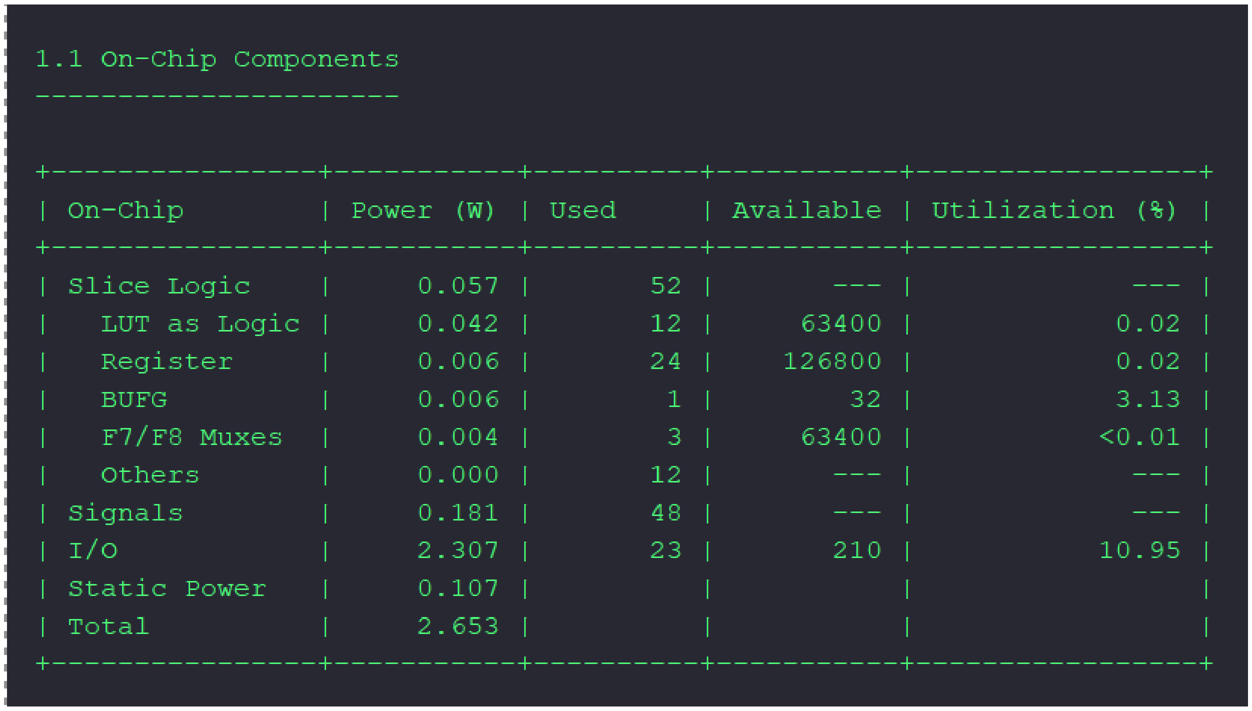
## Vivado Utilization:

### LUTs and FFs:



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

### Power:



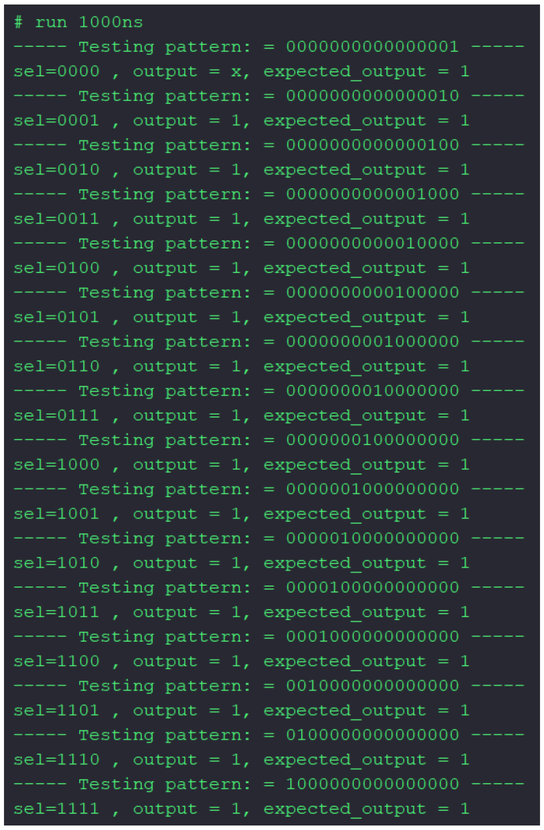
```
1.1 On-Chip Components
----------------------

+----------------+-----------+----------+-----------+-----------------+
| On-Chip        | Power (W) | Used     | Available | Utilization (%) |
+----------------+-----------+----------+-----------+-----------------+
| Slice Logic    |     0.057 |       52 |       --- |             --- |
|   LUT as Logic |     0.042 |       12 |     63400 |            0.02 |
|   Register     |     0.006 |       24 |    126800 |            0.02 |
|   BUFG         |     0.006 |        1 |        32 |            3.13 |
|   F7/F8 Muxes  |     0.004 |        3 |     63400 |           <0.01 |
|   Others       |     0.000 |       12 |       --- |             --- |
| Signals        |     0.181 |       48 |       --- |             --- |
| I/O            |     2.307 |       23 |       210 |           10.95 |
| Static Power   |     0.107 |          |           |                 |
| Total          |     2.653 |          |           |                 |
+----------------+-----------+----------+-----------+-----------------+
```

## TestBench:

### TestBench Code:

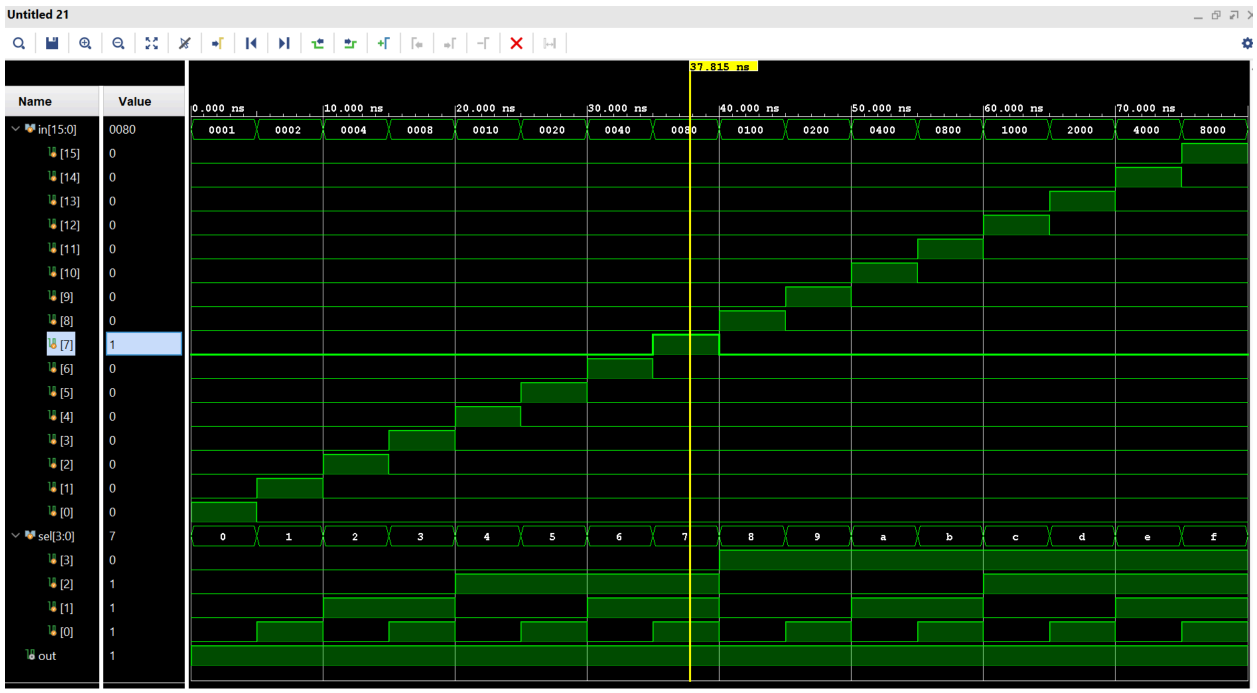
```
module mux16x1_tb;

  reg [15:0] in;
  reg [3:0] sel;


  wire out;

  mux16x1 uut (
    .in(in),
    .sel(sel),
    .out(out)
  );

  initial begin
    //Test pattern
    for(integer j = 0; j < 16; j = j + 1) begin
        in = 2 ** j; // 2 to the power of j
        sel = j;
    $display("----- Testing pattern: = %b -----", in);
    $display("sel=%b , output = %b, expected_output = %b",  sel, out, in[j]);

    #5;
    end

    $finish;

  end

endmodule
```

TestBench Log Description: shows that all cases passed.



```
# run 1000ns
----- Testing pattern: = 0000000000000001 -----
sel=0000 , output = x, expected_output = 1
----- Testing pattern: = 0000000000000010 -----
sel=0001 , output = 1, expected_output = 1
----- Testing pattern: = 0000000000000100 -----
sel=0010 , output = 1, expected_output = 1
----- Testing pattern: = 0000000000001000 -----
sel=0011 , output = 1, expected_output = 1
----- Testing pattern: = 0000000000010000 -----
sel=0100 , output = 1, expected_output = 1
----- Testing pattern: = 0000000000100000 -----
sel=0101 , output = 1, expected_output = 1
----- Testing pattern: = 0000000001000000 -----
sel=0110 , output = 1, expected_output = 1
----- Testing pattern: = 0000000010000000 -----
sel=0111 , output = 1, expected_output = 1
----- Testing pattern: = 0000000100000000 -----
sel=1000 , output = 1, expected_output = 1
----- Testing pattern: = 0000001000000000 -----
sel=1001 , output = 1, expected_output = 1
----- Testing pattern: = 0000010000000000 -----
sel=1010 , output = 1, expected_output = 1
----- Testing pattern: = 0000100000000000 -----
sel=1011 , output = 1, expected_output = 1
----- Testing pattern: = 0001000000000000 -----
sel=1100 , output = 1, expected_output = 1
----- Testing pattern: = 0010000000000000 -----
sel=1101 , output = 1, expected_output = 1
----- Testing pattern: = 0100000000000000 -----
sel=1110 , output = 1, expected_output = 1
----- Testing pattern: = 1000000000000000 -----
sel=1111 , output = 1, expected_output = 1
```

TB Waveform Screenshot:



## Contributions:

Daniel Mondragon + 50% effort: Wrote the test bench, and documented the Vivado utilization. Uploaded to github.

Kobe Aquino + 50% effort: Wrote the test bench, documented the code and waveform onto the pdf, tested the verilog source files with the fpga board, and uploaded to youtube.