CALIFORNIA STATE POLYTECHNIC
UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

LAB 3

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

ECE 3300L Summer 2025

Digital Circuit Design using Verilog

Professor Mohamed Aly

By: Clay Kim and Changwe Musonda

Objective

Our objective of this lab is to design and implement a 16-to-1 multiplexer (MUX16x1) using gate-level 2x1 multiplexers in Verilog. The multiplexer's select inputs will be controlled by pushbuttons on the Nexys A7 board, which will be converted into toggle switches through the implementation of a debouncing system and a toggle flip-flop for each select bit. The multiplexer's output will be displayed on LED0, with its 16 inputs driven by the 16 onboard switches (SW [15:0]).

MUX 2x1 Gate-Level 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
```

Code given by instructor that creates a 2x1 multiplexer using not, and, or gates,

MUX 16x1 Using Generate Loops

```
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
```

Code given by instructor that uses multiple 2x1 multiplexer code to create a 16x1 multiplexer using for loops.

Debounce Module

```
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
```

A module created in order to remove any bounces (basically filters out any inconsistent mechanical switch pulses)

Toggle Flip-Flop

```
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
```

This code takes the debounced signal and converts it into a toggling output. It "cleans" the raw button input and then uses a register to detect a rising edge and flips the value.

Top-Level Module

```
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
```

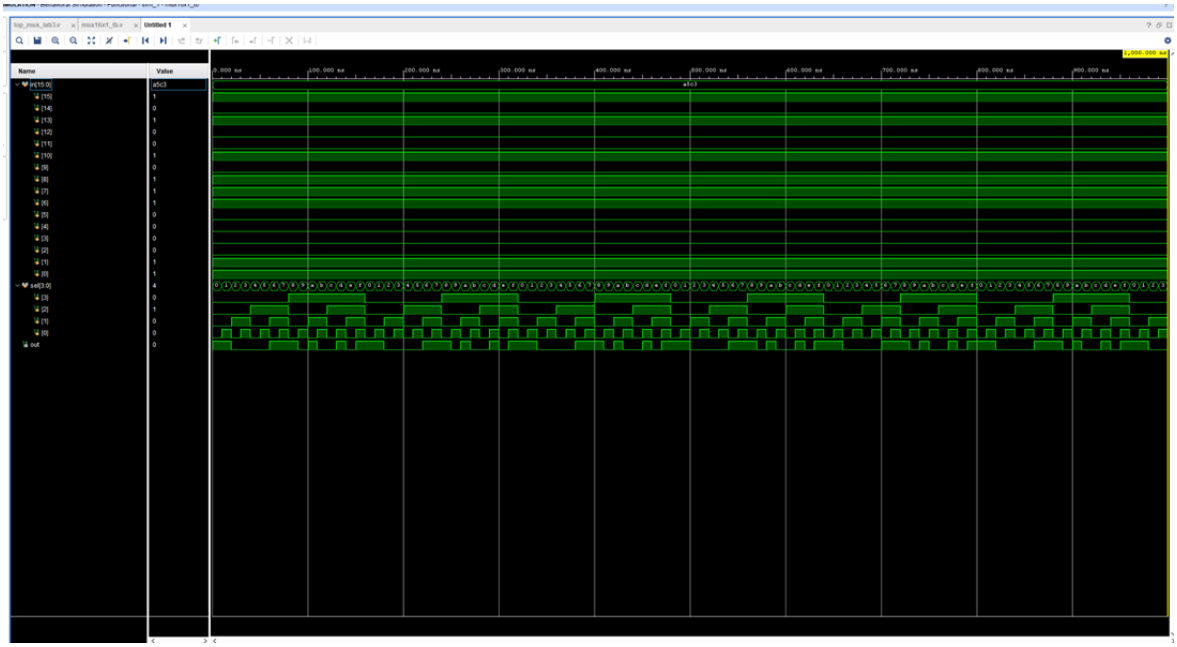
Integrates all the different coded modules to work together.

Testbench code

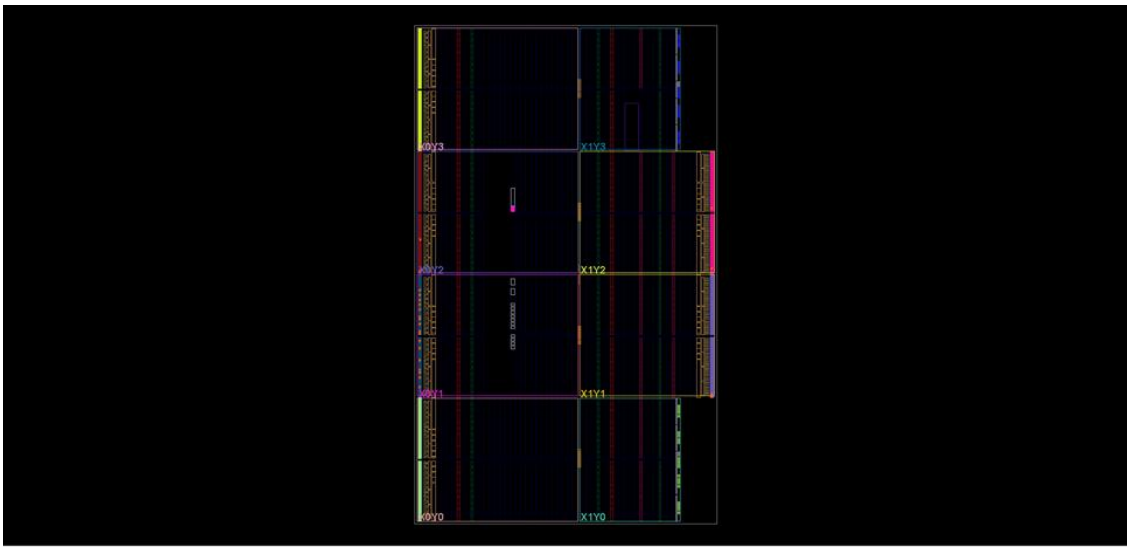
```
module mux16x1_tb;
reg [15:0] in;
reg [3:0] sel;
wire out;
mux16x1 uut (.in(in), .sel(sel), .out(out));
initial begin
in = 16'b1010_0101_1100_0011;
for (sel = 0; sel < 16; sel = sel + 1) begin
#10;
$display("sel=%d, out=%b", sel, out);
end
$finish;
end
endmodule
```

Testbench code to initialize 16 data inputs and cycles through the values from 0 to 15 in the 16x1 MUX and prints the selected value and resulting the output of the MUX to the console.

Simulation Screenshot



Vivado Utilization



Utilization

Hierarchy

Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	F8 Muxes (15850)	Bonded IOB (210)	BUFGCTRL (32)
top_mux_lab3	12	24	2	1	23	1
t0 (toggle_switch)	2	6	0	0	0	0
db (debounce_5)	2	4	0	0	0	0
t1 (toggle_switch_0)	6	6	2	1	0	0
db (debounce_4)	2	4	0	0	0	0
t2 (toggle_switch_1)	2	6	0	0	0	0
db (debounce_3)	2	4	0	0	0	0
t3 (toggle_switch_2)	2	6	0	0	0	0
db (debounce)	2	4	0	0	0	0

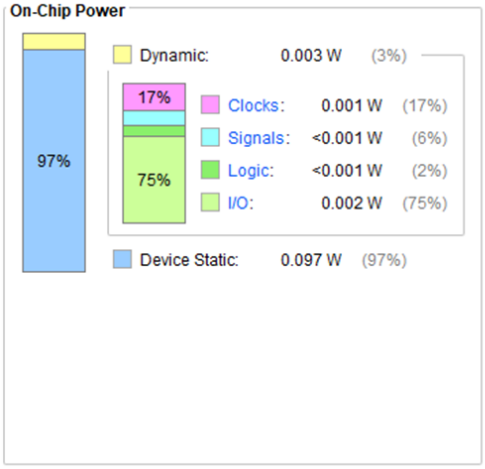
Power | Methodology | DRC | Timing | Utilization

Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.1 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**25.5°C**
Thermal Margin:	59.5°C (12.9 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



Partner Contribution

Clay Kim: Testbench and Simulation

Changwe Musonda: Demonstration video and implementation