ECE 3300 Lab 3 Group J

Introduction

In this lab, we designed a 16-to-1 multiplexer using gate-level 2x1 multiplexers in Verilog. We will be able to control the MUX using the pushbuttons 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.

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: Implements a basic 2-to-1 multiplexer using NOT, AND, and OR gates.

```
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: Constructs a 16-to-1 multiplexer by using multiple mux2x1 modules across four levels using 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: Filters a noisy pushbutton input by using a 3-bit shift register to confirm stable high or low states.

```
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: Implements a toggle flip-flop that changes state only on a debounced rising edge of a button press, and resets cleanly with rst.

```
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_mux_lab3: Top-level module that connects the 16 inputs (SW) to the 16x1 MUX and uses four pushbuttons as toggle-controlled select lines to display one selected input on LED0.

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { clk }]; #IO_L12P_T1_MRCC_35 Sch=clk100mhz
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {clk}];


##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { SW[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { SW[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { SW[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { SW[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { SW[4] }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { SW[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
set_property -dict { PACKAGE_PIN U18   IOSTANDARD LVCMOS33 } [get_ports { SW[6] }]; #IO_L17N_T2_A13_D29_14 Sch=sw[6]
set_property -dict { PACKAGE_PIN R13   IOSTANDARD LVCMOS33 } [get_ports { SW[7] }]; #IO_L5N_T0_D07_14 Sch=sw[7]
set_property -dict { PACKAGE_PIN T8    IOSTANDARD LVCMOS18 } [get_ports { SW[8] }]; #IO_L24N_T3_34 Sch=sw[8]
set_property -dict { PACKAGE_PIN U8    IOSTANDARD LVCMOS18 } [get_ports { SW[9] }]; #IO_25_34 Sch=sw[9]
set_property -dict { PACKAGE_PIN R16   IOSTANDARD LVCMOS33 } [get_ports { SW[10] }]; #IO_L15P_T2_DQS_RDWR_B_14 Sch=sw[10]
set_property -dict { PACKAGE_PIN T13   IOSTANDARD LVCMOS33 } [get_ports { SW[11] }]; #IO_L23P_T3_A03_D19_14 Sch=sw[11]
set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { SW[12] }]; #IO_L24P_T3_35 Sch=sw[12]
set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33 } [get_ports { SW[13] }]; #IO_L20P_T3_A08_D24_14 Sch=sw[13]
set_property -dict { PACKAGE_PIN U11   IOSTANDARD LVCMOS33 } [get_ports { SW[14] }]; #IO_L19N_T3_A09_D25_VREF_14 Sch=sw[14]
set_property -dict { PACKAGE_PIN V10   IOSTANDARD LVCMOS33 } [get_ports { SW[15] }]; #IO_L21P_T3_DQS_14 Sch=sw[15]


## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { LED0 }]; #IO_L18P_T2_A24_15 Sch=led[0]
#set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
#set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
#set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]

##Buttons
set_property -dict { PACKAGE_PIN N17   IOSTANDARD LVCMOS33 } [get_ports { rst }]; #IO_L9P_T1_DQS_14 Sch=btnc
set_property -dict { PACKAGE_PIN M18   IOSTANDARD LVCMOS33 } [get_ports { btnU }]; #IO_L4N_T0_D05_14 Sch=btnu
set_property -dict { PACKAGE_PIN P17   IOSTANDARD LVCMOS33 } [get_ports { btnL }]; #IO_L12P_T1_MRCC_14 Sch=btnl
set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports { btnR }]; #IO_L10N_T1_D15_14 Sch=btnr
set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports { btnD }]; #IO_L9N_T1_DQS_D13_14 Sch=btnd
```

xdc file

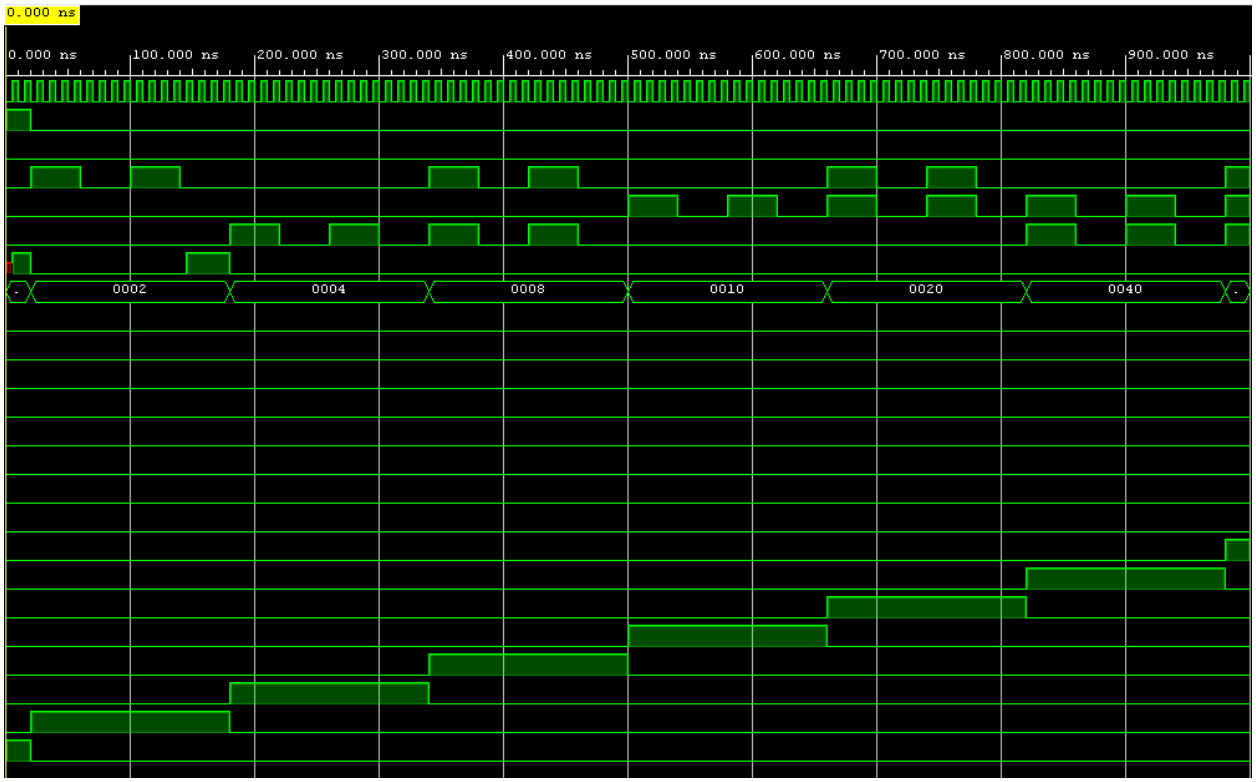
```
module top_mux_lab3_tb;

    reg clk;
    reg rst;
    reg [15:0] SW;
    reg btnU, btnD, btnL, btnR;
    wire LED0;

    top_mux_lab3 uut (
        .clk(clk),
        .rst(rst),
        .SW(SW),
        .btnU(btnU),
        .btnD(btnD),
        .btnL(btnL),
        .btnR(btnR),
        .LED0(LED0)
    );

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

 // Button pulse tasks
 task pulse_btnD;
     begin
         btnD = 1; #20; btnD = 0;
     end
 endtask

 task pulse_btnR;
     begin
         btnR = 1; #20; btnR = 0;
     end
 endtask

 task pulse_btnL;
     begin
         btnL = 1; #20; btnL = 0;
     end
 endtask

 task pulse_btnU;
     begin
         btnU = 1; #20; btnU = 0;
     end
 endtask

initial begin
    // Initialize inputs
    rst = 1;
    SW = 16'b0000_0000_0000_1001;
    btnU = 0;
    btnD = 0;
    btnL = 0;
    btnR = 0;

    #20;
    rst = 0;

    pulse_btnD;  // sel[0] = 1
    #50;

    pulse_btnR;  // sel[1] = 1
    #50;

    pulse_btnL;  // sel[2] = 1
    #50;

    pulse_btnU;  // sel[3] = 1
    #50;

    // Set SW[15] = 1, should be selected by mux
    SW[15] = 1;
    #20;

    $display("Final LED0 value (expecting SW[15] = 1): %b", LED0);
    #50;
```

testbench

Vivado Utilization

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy
✓ synth_1	constrs_1	synth_design Complete!												12	24	0	0	0	6/30/25, 11:26 AM	00:01:06	Vivado Synthesis Defau
✓ impl_1	constrs_1	route_design Complete!	8.285	0.000	0.152	0.000		0.000	0.100	0	6 Warn			12	24	0	0	0	6/30/25, 11:28 AM	00:02:02	Vivado Implementation

Testbench Waveform



Group Video Link
https://www.youtube.com/shorts/S5ORQwUqNxM

Reflection
This lab provided valuable hands-on experience in combining smaller modules into a larger, functional design. Implementing a 16-to-1 multiplexer using gate-level 2x1 multiplexers deepened my understanding of modular design and how basic logic components can be composed to build more complex systems. Overall, this lab reinforced concepts in digital design, Verilog modularization, and hardware interfacing on an FPGA.

Partner Contribution
Sean Go - code, lab report, and video demonstration
Ryan Tran - code, lab report