LAB 8

Group C

Edward Enriquez, eenriquez@cpp.edu, ID 013242619
Lorenzo Antonio Fabian Lopez, lfabianlopez@cpp.edu, ID 014903070
Rafael Gaeta, rgaeta@cpp.edu, ID 014522794

Contribution:
Edward Enriquez
50% Code creation and analysis

Lorenzo Antonio Fabian Lopez
25% Analysis of results and conclusion

Rafael Gaeta
25% Report and graphics

Abstract

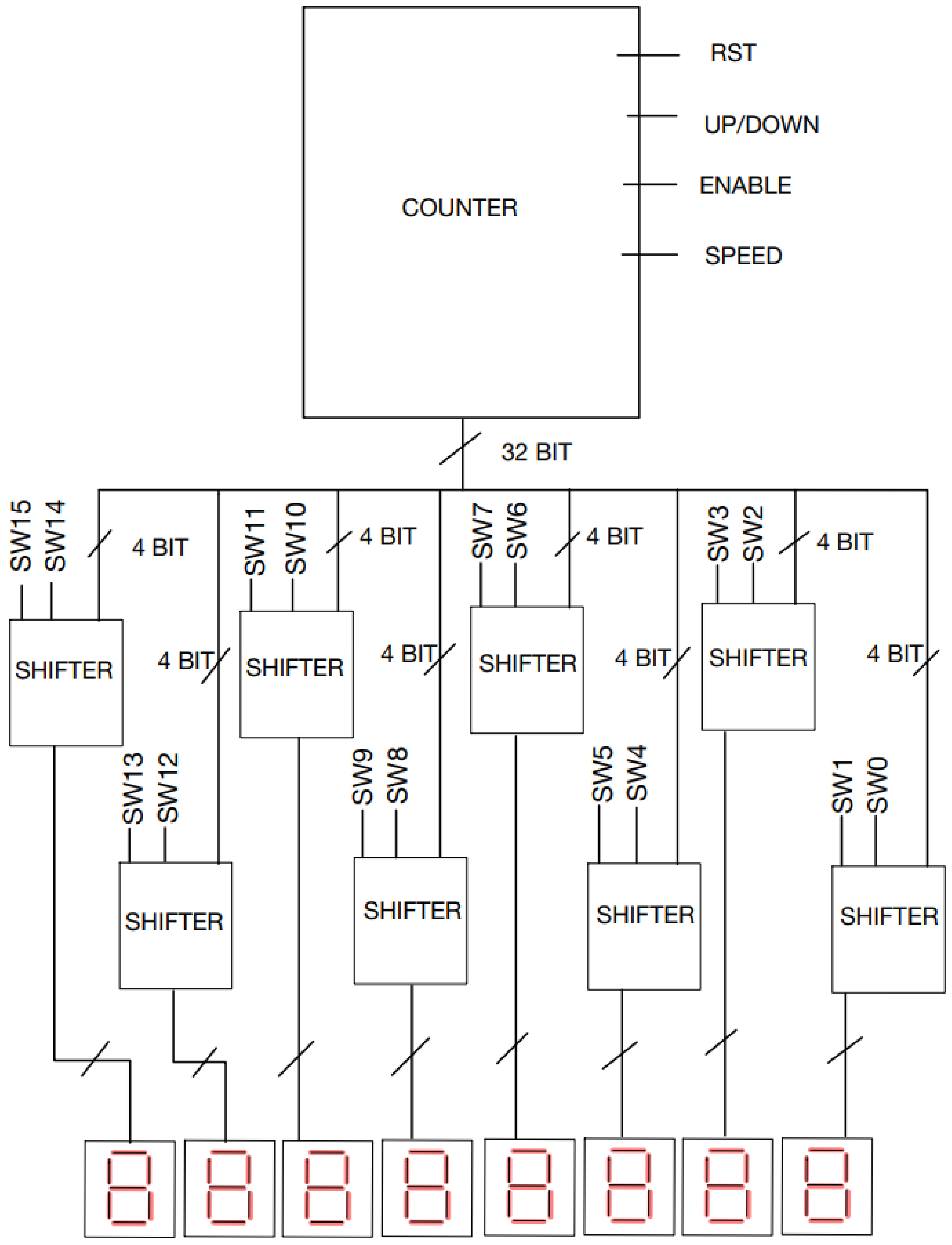
To perform this lab design a 4-bit barrel shifter. The barrel shifter will be located between the output of the 4-bit BCD and the input. Assign the 2-SW per digit to control the barrel shifters. Additionally the circuit design must contain two buttons that when pressed and released, will enable/disable the circuit, and the second button will make the counter go up/down.

Specifications,

- Define Circuit Schematic/Diagram
- 4-bit barrel shifters
- 2 switches to control barrel shifters
- Total power = No Specifications Given
- Number of LUT = No Specifications Given

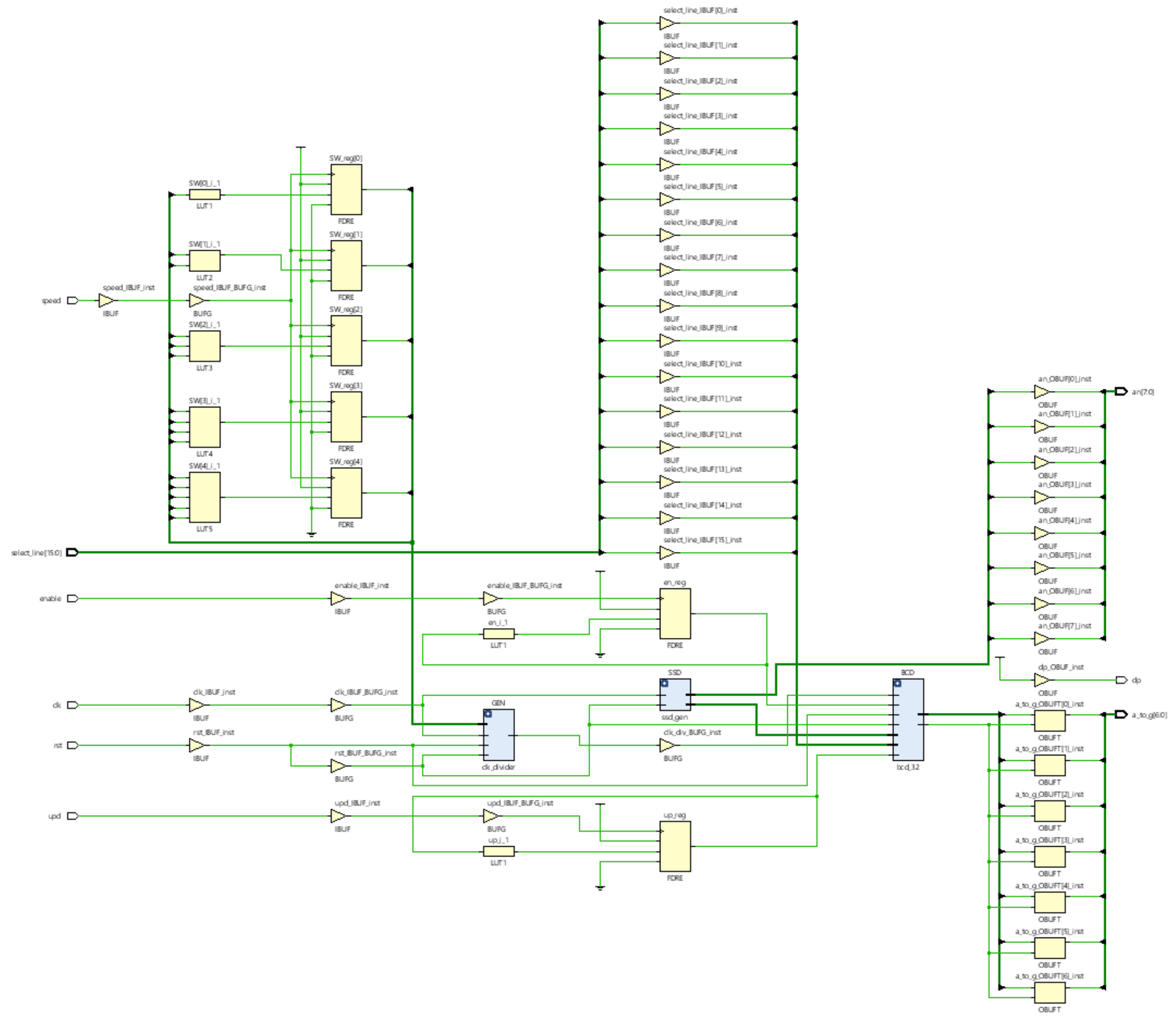
Circuit Design

Diagram



7 segment displays

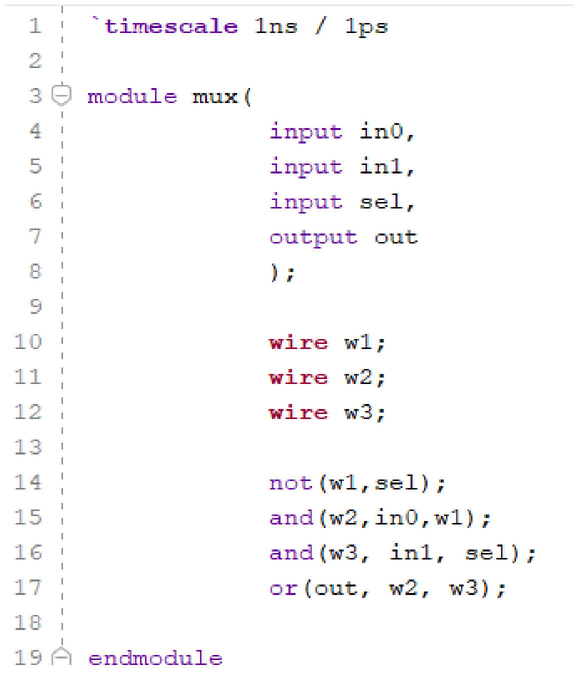
Verilog Generated Schematic



Code Detail

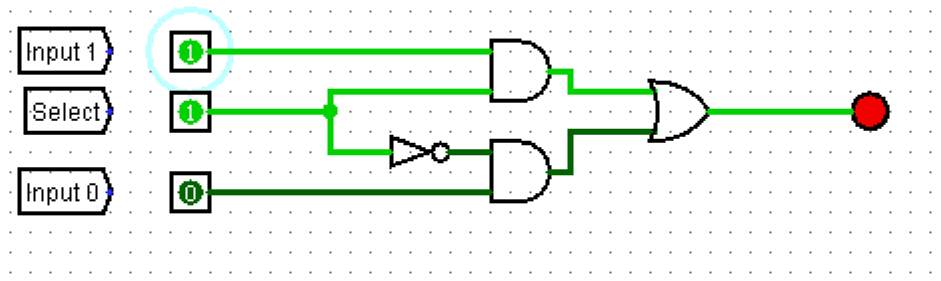
There were 2 files which were added to the code of the previous labs. These were the mux.v module and the shifter.v module. Apart from that, the bdc_32.v, top.v, and nxsys4ddr.xdc files were modified to adjust for the new functions and inputs.

The mux.v module used structural modeling to describe a 2x1 multiplexer. The multiplexer module used the typical schematic which used a NOT gate, two AND, and an OR gate.



```verilog
`timescale 1ns / 1ps

module mux(
            input in0,
            input in1,
            input sel,
            output out
            );

            wire w1;
            wire w2;
            wire w3;

            not(w1,sel);
            and(w2,in0,w1);
            and(w3, in1, sel);
            or(out, w2, w3);

endmodule
```

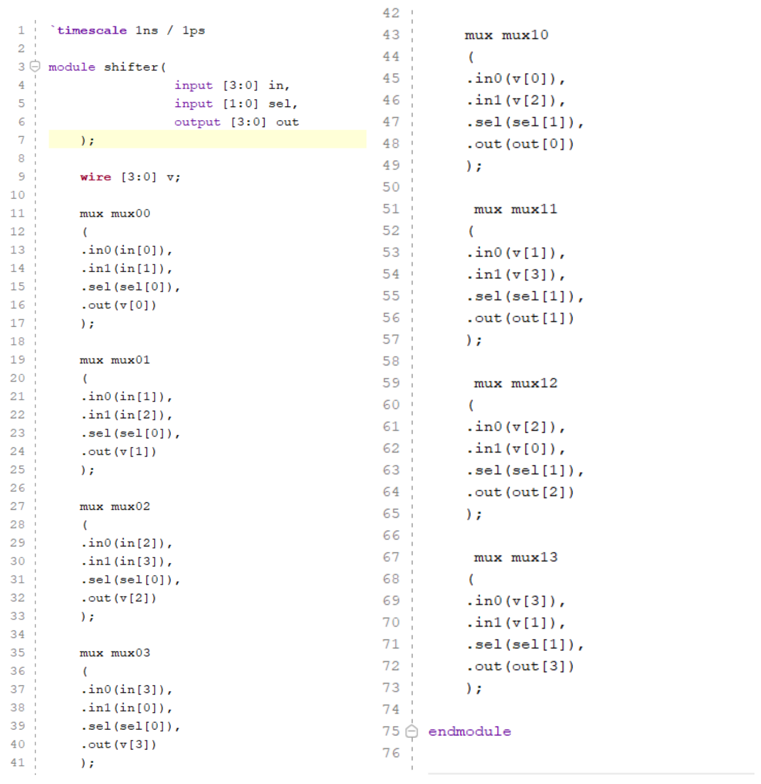
mux.v



The shifter.v module described a 4-bit barrel shifter using 8 instantiations of multiplexers. The module had a 4-bit input, a 4-bit output, and a 2-bit select input. The 8 multiplexer instantiations were split into 2 rows of 4.

Each 2x1 multiplexer in the first row had each module input bit paired with its succeeding bit. The select for the multiplexers in this row was the zero bit of the module select. The outputs of this row were connected to wire v.

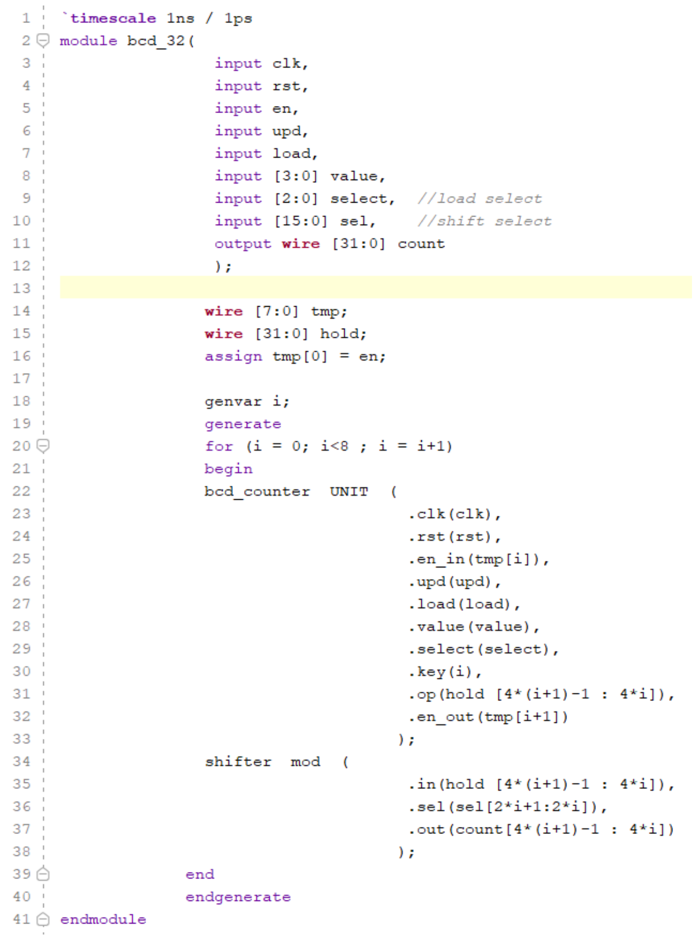
Each 2x1 multiplexer in the second row had each bit of wire v paired with the signal two bits over. The select for the multiplexers in this row was the first bit of the module select. The outputs of this row were connected to the module output.



```verilog
`timescale 1ns / 1ps

module shifter(
                input [3:0] in,
                input [1:0] sel,
                output [3:0] out
    );

    wire [3:0] v;

    mux mux00
    (
    .in0(in[0]),
    .in1(in[1]),
    .sel(sel[0]),
    .out(v[0])
    );

    mux mux01
    (
    .in0(in[1]),
    .in1(in[2]),
    .sel(sel[0]),
    .out(v[1])
    );

    mux mux02
    (
    .in0(in[2]),
    .in1(in[3]),
    .sel(sel[0]),
    .out(v[2])
    );

    mux mux03
    (
    .in0(in[3]),
    .in1(in[0]),
    .sel(sel[0]),
    .out(v[3])
    );

    mux mux10
    (
    .in0(v[0]),
    .in1(v[2]),
    .sel(sel[1]),
    .out(out[0])
    );

     mux mux11
    (
    .in0(v[1]),
    .in1(v[3]),
    .sel(sel[1]),
    .out(out[1])
    );

     mux mux12
    (
    .in0(v[2]),
    .in1(v[0]),
    .sel(sel[1]),
    .out(out[2])
    );

     mux mux13
    (
    .in0(v[3]),
    .in1(v[1]),
    .sel(sel[1]),
    .out(out[3])
    );

endmodule
```

shifter.v

The bcd_32 module was modified in such a way so that the eight 4-bit outputs of the bcd_counter.v instantiations were inputted into eight instantiations of the shifter. The output of the shifter modules was now the output of the bcd_32 module. The select of the shifters was added as a 16-bit input.



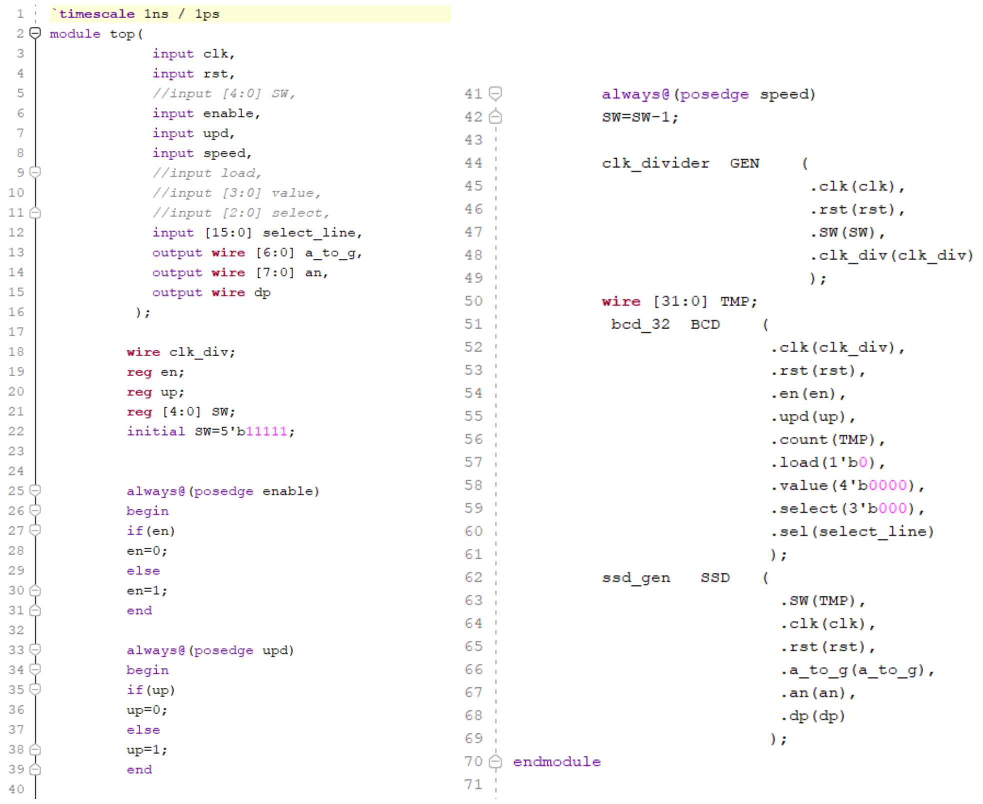
```
`timescale 1ns / 1ps
module bcd_32(
                input clk,
                input rst,
                input en,
                input upd,
                input load,
                input [3:0] value,
                input [2:0] select,  //load select
                input [15:0] sel,    //shift select
                output wire [31:0] count
                );

               wire [7:0] tmp;
               wire [31:0] hold;
               assign tmp[0] = en;

               genvar i;
               generate
               for (i = 0; i<8 ; i = i+1)
               begin
               bcd_counter  UNIT  (
                                    .clk(clk),
                                    .rst(rst),
                                    .en_in(tmp[i]),
                                    .upd(upd),
                                    .load(load),
                                    .value(value),
                                    .select(select),
                                    .key(i),
                                    .op(hold [4*(i+1)-1 : 4*i]),
                                    .en_out(tmp[i+1])
                                   );
               shifter  mod  (
                                    .in(hold [4*(i+1)-1 : 4*i]),
                                    .sel(sel[2*i+1:2*i]),
                                    .out(count[4*(i+1)-1 : 4*i])
                                   );
             end
             endgenerate
endmodule
```

bcd_32.v

For this lab, a 1-bit input named *speed* and a 16-bit input named *select_line* were added to the *top.v* module. Inputs irrelevant to this lab were commented out.

Three *always@* blocks were added to the *top.v* file as well. The first block toggled the enable bit on the positive edge of the enable button. The second block toggled the up/down counter bit with the positive edge of the upd button. The final block decreased the *SW* variable whenever the speed button was pressed which in turn increased the clock speed of the system.

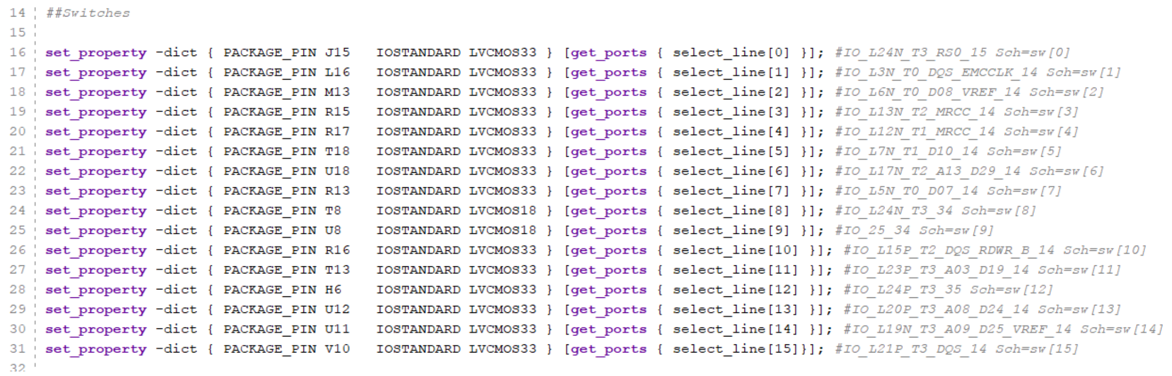
The *select_line* input was used as the select for the shifters in the *bcd_32* module.



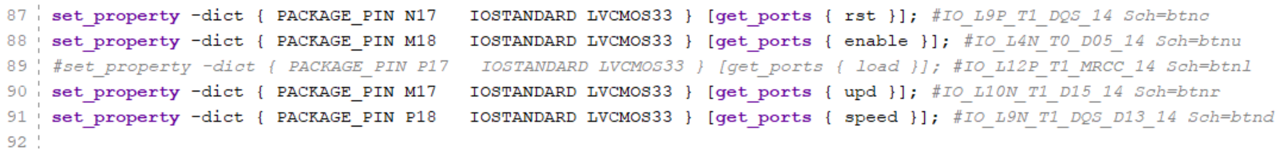
```verilog
`timescale 1ns / 1ps
module top(
            input clk,
            input rst,
            //input [4:0] SW,
            input enable,
            input upd,
            input speed,
            //input load,
            //input [3:0] value,
            //input [2:0] select,
            input [15:0] select_line,
            output wire [6:0] a_to_g,
            output wire [7:0] an,
            output wire dp
          );

        wire clk_div;
        reg en;
        reg up;
        reg [4:0] SW;
        initial SW=5'b11111;


        always@(posedge enable)
        begin
        if(en)
        en=0;
        else
        en=1;
        end

        always@(posedge upd)
        begin
        if(up)
        up=0;
        else
        up=1;
        end

        always@(posedge speed)
        SW=SW-1;

        clk_divider  GEN    (
                             .clk(clk),
                             .rst(rst),
                             .SW(SW),
                             .clk_div(clk_div)
                             );
        wire [31:0] TMP;
         bcd_32  BCD    (
                         .clk(clk_div),
                         .rst(rst),
                         .en(en),
                         .upd(up),
                         .count(TMP),
                         .load(1'b0),
                         .value(4'b0000),
                         .select(3'b000),
                         .sel(select_line)
                         );
        ssd_gen   SSD   (
                          .SW(TMP),
                          .clk(clk),
                          .rst(rst),
                          .a_to_g(a_to_g),
                          .an(an),
                          .dp(dp)
                         );
endmodule
```

top.v

In the nxsys4ddr.xdc file the switches were all set to be a single bit in the 16-bit input *select_line*. The inputs *speed, rst, upd,* and *en* were all set to be buttons.



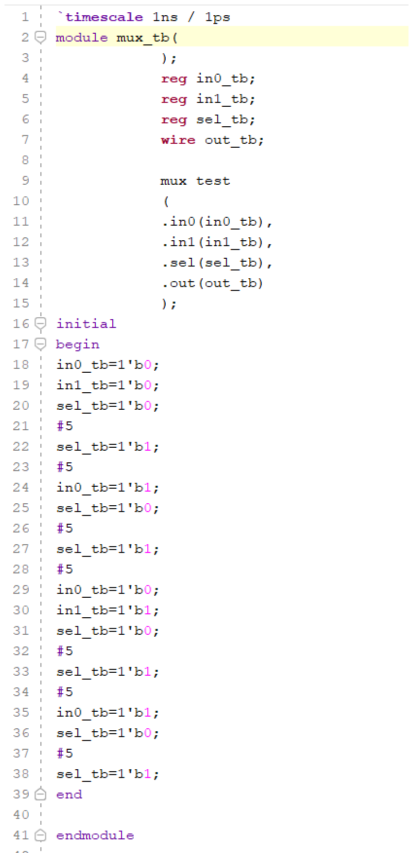
```
##Switches

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

```



```
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 { enable }]; #IO_L4N_T0_D05_14 Sch=btnu
#set_property -dict { PACKAGE_PIN P17   IOSTANDARD LVCMOS33 } [get_ports { load }]; #IO_L12P_T1_MRCC_14 Sch=btnl
set_property -dict { PACKAGE_PIN M17   IOSTANDARD LVCMOS33 } [get_ports { upd }]; #IO_L10N_T1_D15_14 Sch=btnr
set_property -dict { PACKAGE_PIN P18   IOSTANDARD LVCMOS33 } [get_ports { speed }]; #IO_L9N_T1_DQS_D13_14 Sch=btnd

```

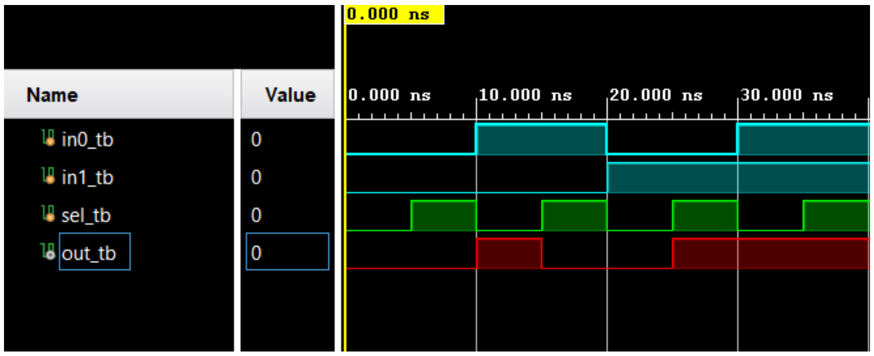
<u>Test Bench</u>

A testbench module was created for each of the new modules in this lab.

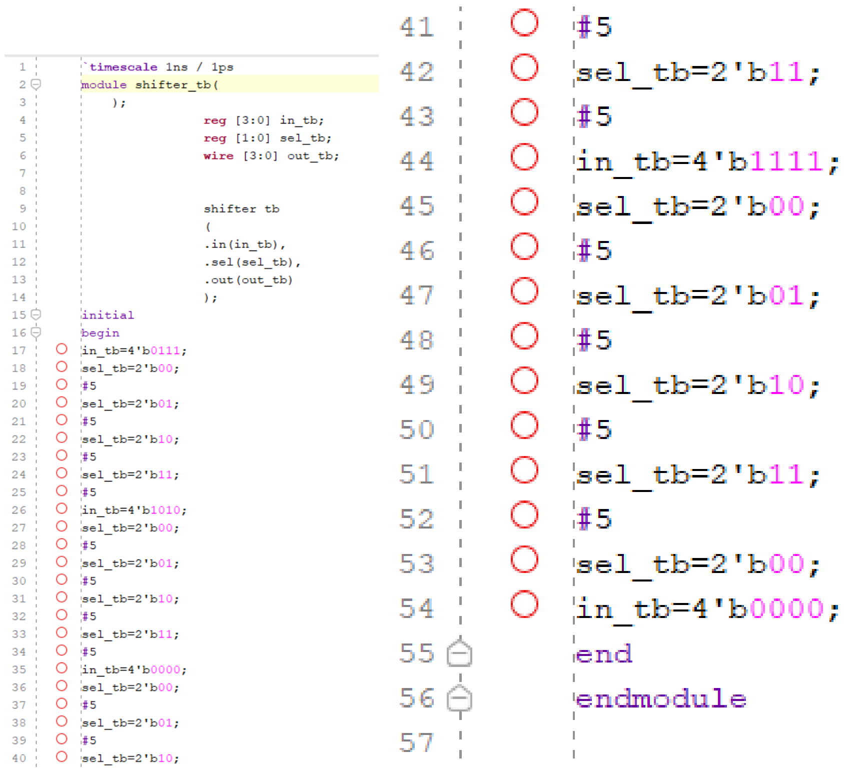


```verilog
`timescale 1ns / 1ps
module mux_tb(
            );
            reg in0_tb;
            reg in1_tb;
            reg sel_tb;
            wire out_tb;

            mux test
            (
            .in0(in0_tb),
            .in1(in1_tb),
            .sel(sel_tb),
            .out(out_tb)
            );
initial
begin
in0_tb=1'b0;
in1_tb=1'b0;
sel_tb=1'b0;
#5
sel_tb=1'b1;
#5
in0_tb=1'b1;
sel_tb=1'b0;
#5
sel_tb=1'b1;
#5
in0_tb=1'b0;
in1_tb=1'b1;
sel_tb=1'b0;
#5
sel_tb=1'b1;
#5
in0_tb=1'b1;
sel_tb=1'b0;
#5
sel_tb=1'b1;
end

endmodule
```

mux_tb.v

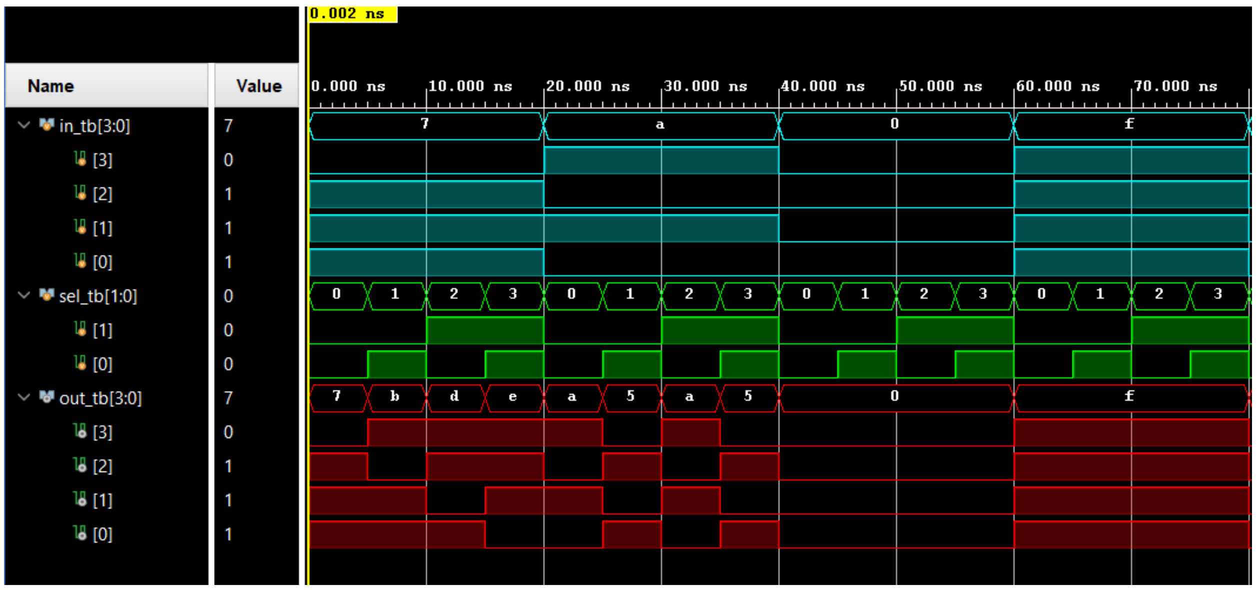


***mux_tb.v* output waveform**



```
`timescale 1ns / 1ps
module shifter_tb(
    );
                reg [3:0] in_tb;
                reg [1:0] sel_tb;
                wire [3:0] out_tb;


                shifter tb
                (
                .in(in_tb),
                .sel(sel_tb),
                .out(out_tb)
                );
initial
begin
in_tb=4'b0111;
sel_tb=2'b00;
#5
sel_tb=2'b01;
#5
sel_tb=2'b10;
#5
sel_tb=2'b11;
#5
in_tb=4'b1010;
sel_tb=2'b00;
#5
sel_tb=2'b01;
#5
sel_tb=2'b10;
#5
sel_tb=2'b11;
#5
in_tb=4'b0000;
sel_tb=2'b00;
#5
sel_tb=2'b01;
#5
sel_tb=2'b10;
#5
sel_tb=2'b11;
#5
in_tb=4'b1111;
sel_tb=2'b00;
#5
sel_tb=2'b01;
#5
sel_tb=2'b10;
#5
sel_tb=2'b11;
#5
sel_tb=2'b00;
in_tb=4'b0000;
end
endmodule
```

shifter_tb.v



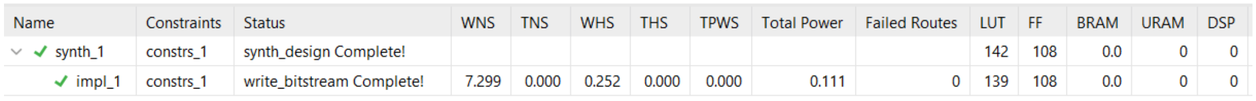
***shifter_tb.v* output waveform**

Corner Cases

Some corner cases for this lab would be that when shifting a digit, the value could be a number which can not be displayed on the 7-segment display. In this case, the *ssg_gen.v* file is set up to not display anything. Another simple corner case is when the digit being displayed is 0, then the shifter does not affect the output.

Technical Report

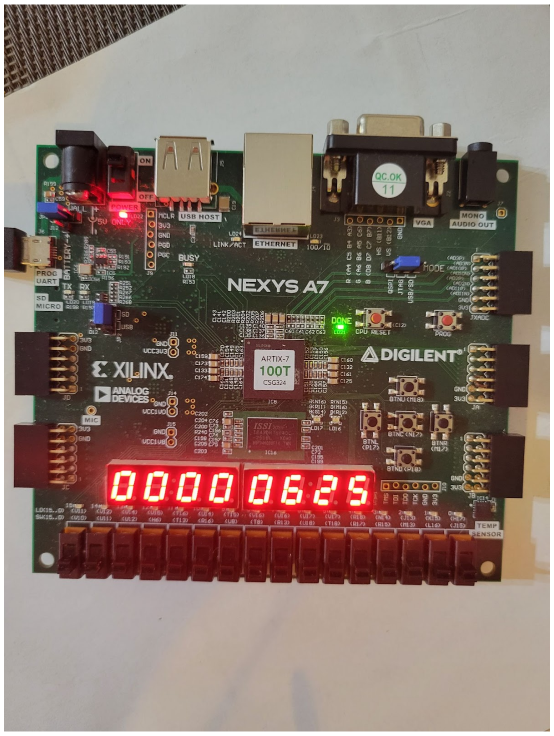
After the simulation was performed. The results obtained from the report are as follows. The total power consumption is 0.111W. The design used less flip flops and look up tables than the Lab 6 design which was essentially the same code without the implementations of shifters and button inputs. However, the power usage increased from the Lab 6's power usage The lower use of look-up tables and flip flops most likely comes from the use of buttons instead of switches as the button signals were coded in the top.v file and were not used as inputs to instantiations as they were in the previous labs.



Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP
✓ synth_1	constrs_1	synth_design Complete!								142	108	0.0	0	0
✓ impl_1	constrs_1	write_bitstream Complete!	7.299	0.000	0.252	0.000	0.000	0.111	0	139	108	0.0	0	0

Synthesis results for design

Once the simulation was complete and the bitstream created the file was transferred to the Nexys 7. The board was tested and the design worked according to the specification requirements provided in the abstract.



Conclusion

After programming the board we were able to insert a barrel shifter between the output of the 4 bit BCD and the input. By assigning 2 switches per seven segment digit to control the barrel shifter, designate a button for the play/pause of the counter & a secondary button for the up/down counter. In doing so we were able to meet the specifications required for the lab while exceeding some of the technical requirements.