David Nguyen

Lab 2

Section A

10/26/2018

Lab 3 Writeup

**Purpose:**

The purpose of this lab was to learn how to build and use multiplexers in combinations to display and adder onto a seven-segment display. I learned how to create a 8bit add /subtractor using mux based on an input. I also used mux to create a 7 segment display based on a 4 bit bus input to a 7 bit bus output.

**Methods:**

Multiplexers:

1. The first part to making a mux module is considering the inputs and outputs provided. The 8\_1 and 4\_1 mux follow the same logic.
2. With input in and sel, the output is in at the number which the selector calls. To use enable in the logic, I & the logic equations in the output.
3. The 2\_1x8 mux is easy as it takes in 2 7 segment buses and a sel, and outputs which ever bus is entered based on the sel.

AddSub8:

1. The first part of AddSub8 is creating a full adder using 2 4x1 mux as half adders. The inputs of the adder will be based on Cin, 0, or 1 with a and b as selectors.
2. The next part is to create a full adder taking 2 7 bit buses as inputs A and B, and outputs a 7bit bus S. To connect the adders together, I connected the cOut of the first bit to Cin of the next bit following the normal schematic of a segmented adder.
3. Finally, AddSub8 also takes in 2 7 bit buses as inputs A and B, and outputs a 7bit bus S. It also takes in sub as an input and outputs ovfl. Sub is used so the output S is A-B or A+B but B is in 2s complement. To convert this to 2sc complement, I inverted B and ran it through a segAdd with cIn = 1.
4. The last part is using the 2\_8x1 mux to choose B or B(2sc) based on the sub as the selector. Ovfl is then assign based of the most significance bits of A, B and S.

7 Segment Display:

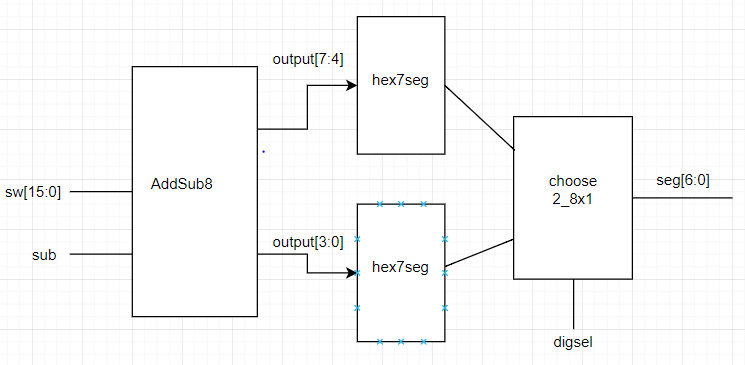
1. The first part of 7 segment display is to make a truth table with inputs being a 4 bit number and the outputs as the 7 columns output based on what the 4 bit number is.
2. The next step is to combine this truth table with an 8x1 mux, by putting boxes around the truth table based on the selector bits on the 4 bit input. I then used 7 8x1 mux, one for each segment on the display and tied the outputs into a wire declared as a bus for testing.
3. I also included the enables on the hex7seg by passing it through input e to the enable for the 8x1 mux.

Top Level Schematic/ Simulation

1. The first part of this is creating a top level module with a 16 bit bus, btnU, btnR, and clkin as inputs, and outputs as a 7bit bus, a 4 bit bus, and dp. This module will have 1 instance of AddSub8 and 2 instances of hex7seg.
2. The next part is to connect the sw15-8 to A, and sw 7-0 to B and run this through AddSub8 then connecting the first 4 lower bits of the output to one hex7seg and the other one to the other hex7seg.
3. I then created a wire dig\_sel which bit what the 7 bit output would be by running choosing which of the hex7seg output we would pick based off of dig\_sel.
4. I then downloaded and configured lab3\_digsel in order to test the design by also selecting values for the 16 switches and sub, to test all 16 hex values on both displays between 100ns changes.

**Results**

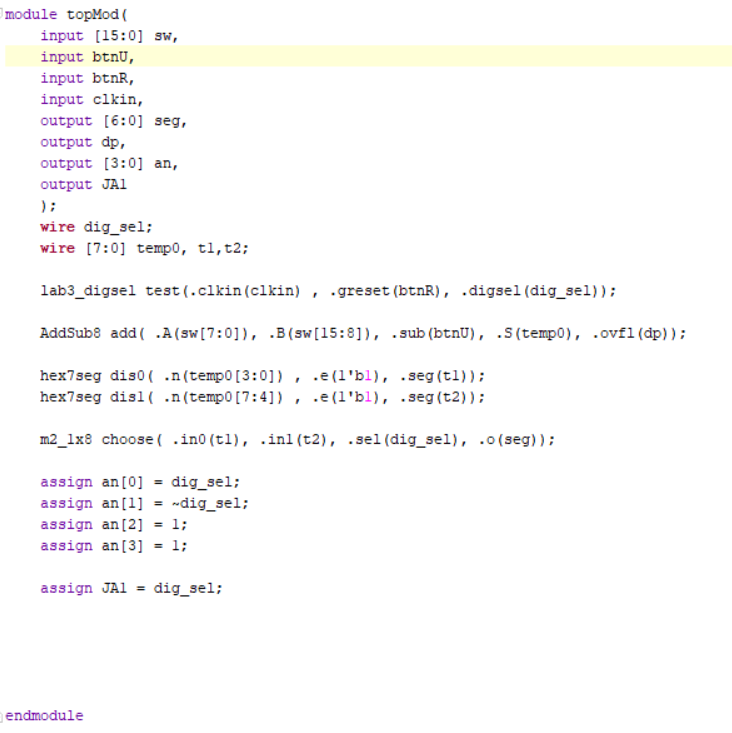
1. In this lab, I used the oscilloscope to determine how fast dig\_sel was changing which was a period of 326 microseconds and a frequency of 3.067 khz. I also noticed no flicking on the board as the clkin is too fast to be distinguishable by human eye.
2. To test all 16 hex values in each display, I set the switch to 16'h0000 - 16'hf00f which will display 00 to FF on each of the displays. I then tested subtracting by setting btnU to 1 and then changed sw and checked if the output matched what was expected. I finally checked overflow by setting sw to 0 then changing it to 16'hff01 and 16'h01ff to see if overflow worked.

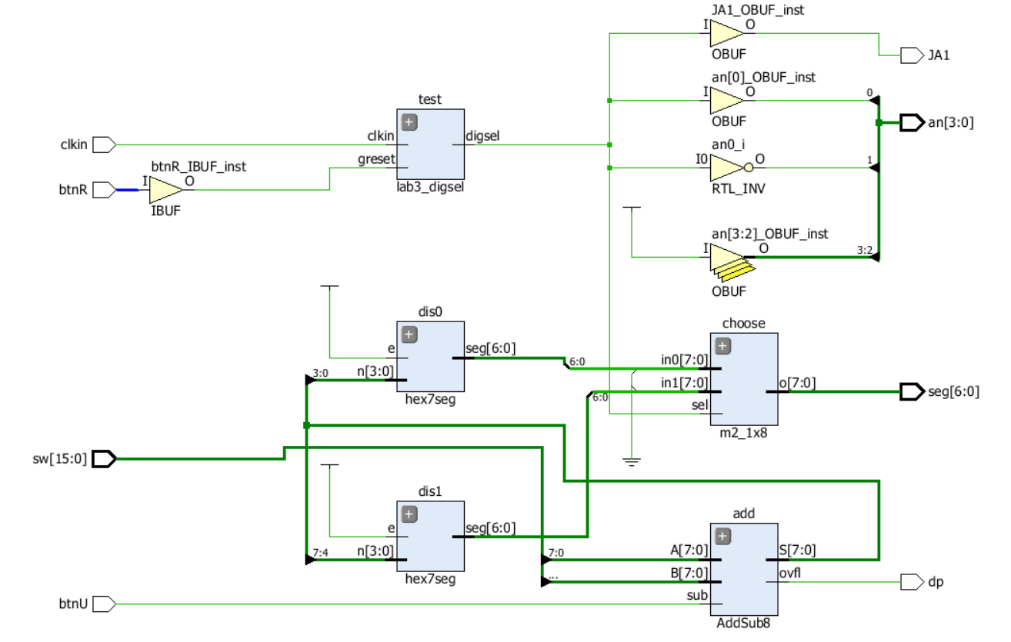


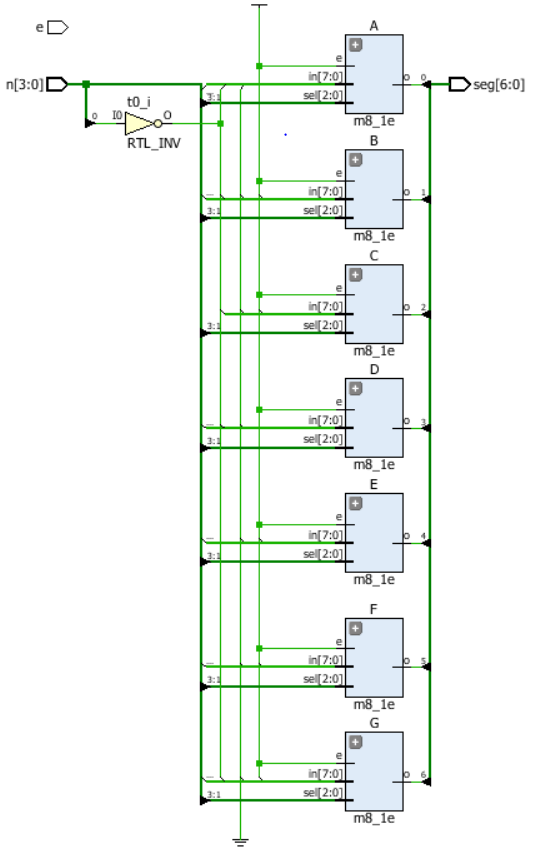
In this design the first input is sw[15:0] and sub. The output of AddSub8 is then split up into with the first 4 bits going to hex7seg and the later 4 bits going to another hex7seg. It is then put into a 2\_8x1 mux with digsel as the selector and outputs seg[6:0] which then displays the correct number on the board.

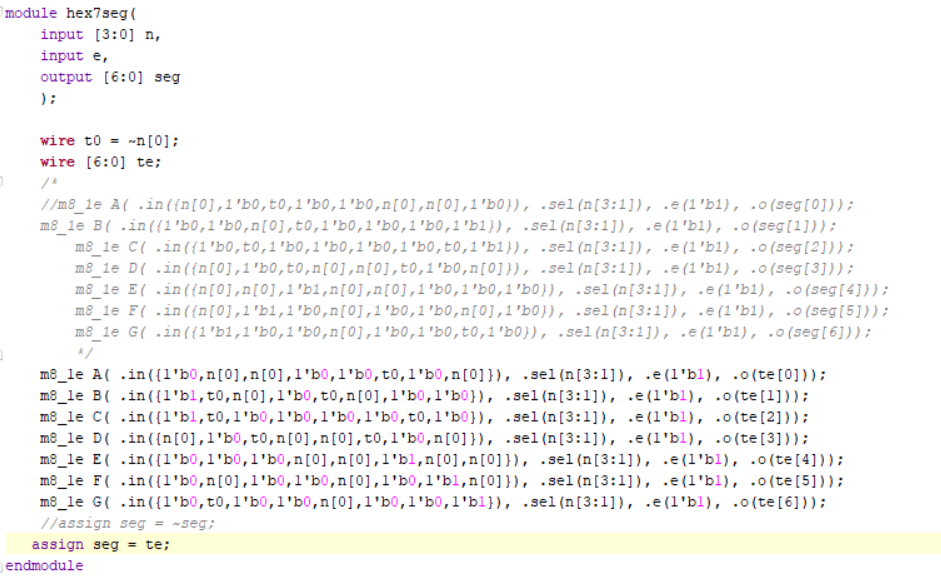
**Conclusion**

In conclusion, this lab was mainly an introduction to multiplexors. In the first part, I created the mux which were used in most of the modules to make the adder/subtractor. I then created AddSub8 which used information we learned in the last lab such as full adder and segmented adder. The next part was to implement a 7-segment display but using mux and the final part. If I were to do this lab again, I would write down more of the truth table for the 8x1 mux and which one to selector to use. It was hard to keep it all in my head while working through it. One thing I learned from this lab was that to input a 0 or 1 it would have to be 1’b0 r 1’b1 because 0 and 1 is not kept as a 1 bit number in Verilog.

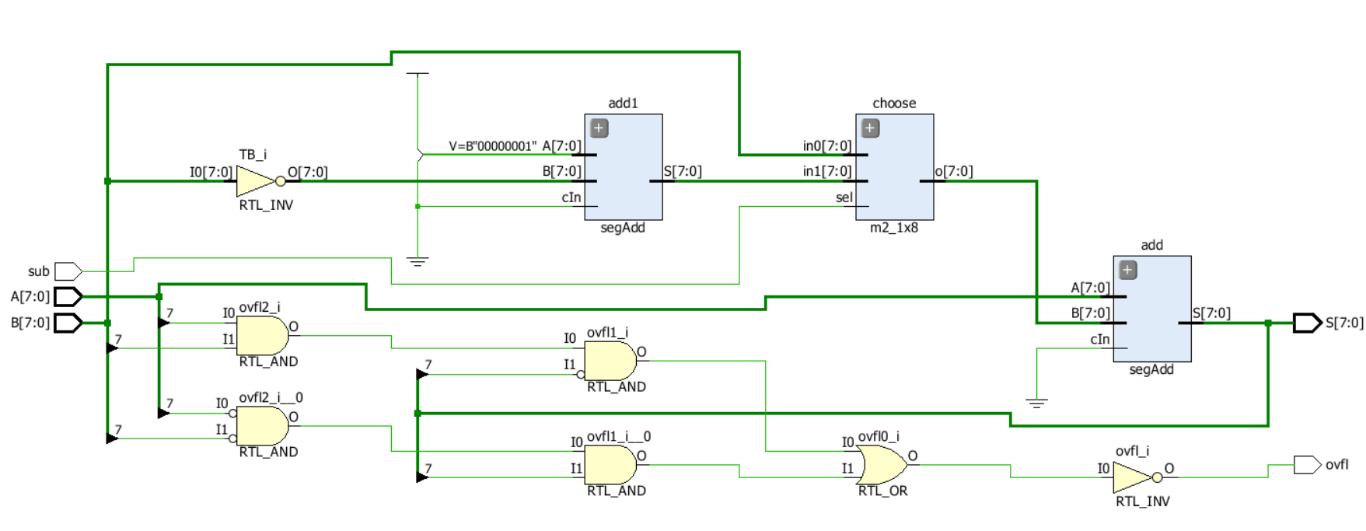
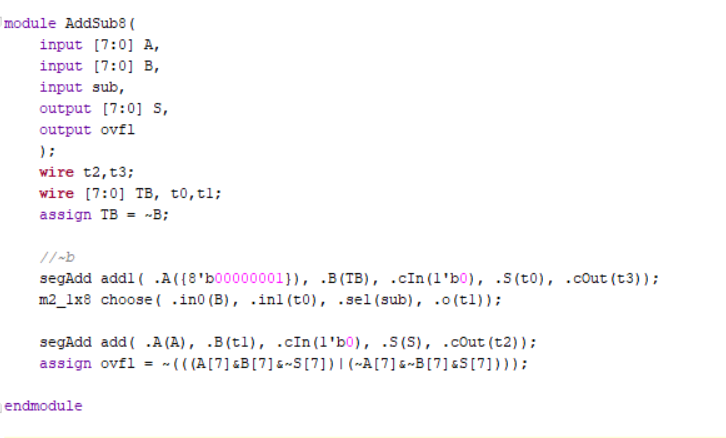
**Supplementary material**

Top Module

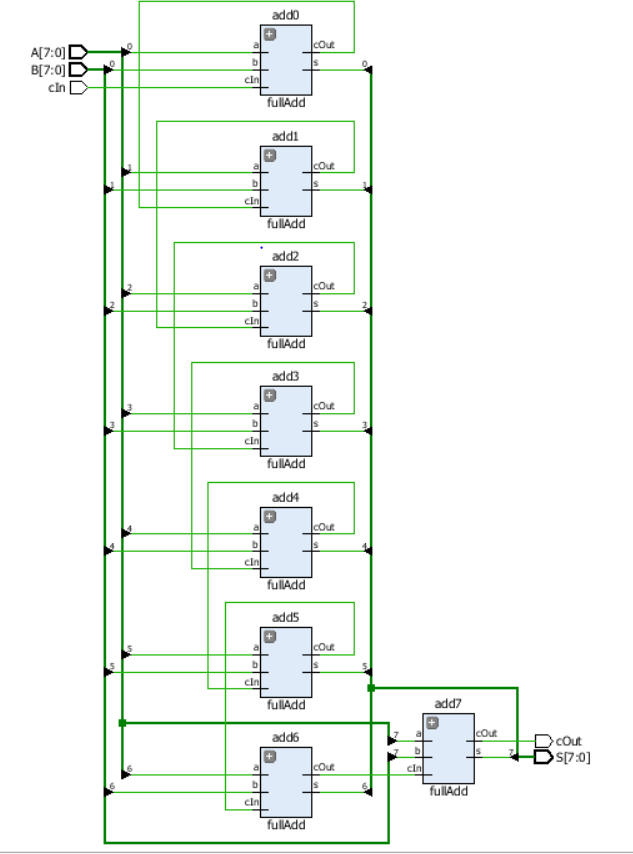


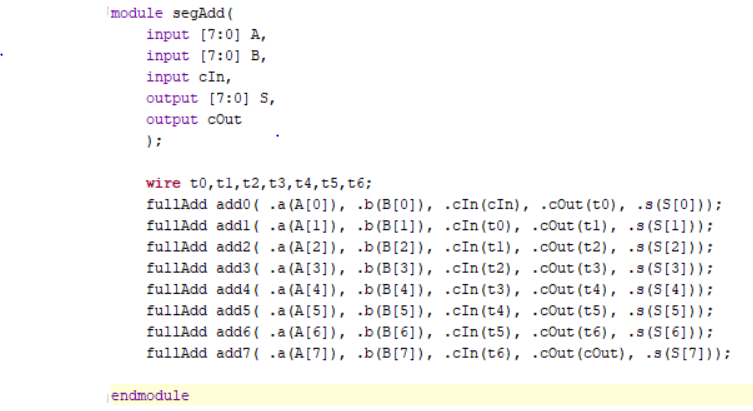


Hex7Seg

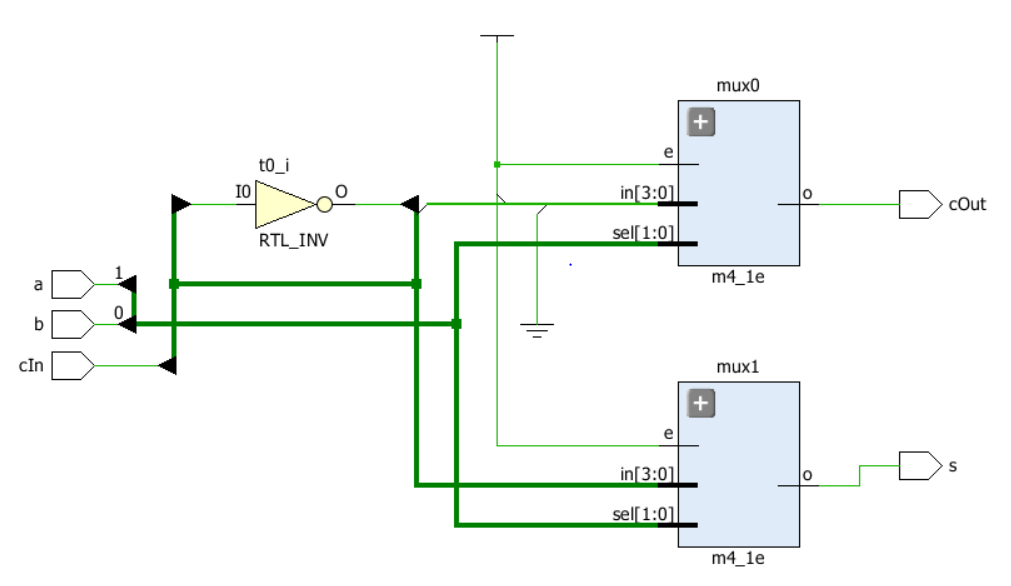


AddSub8

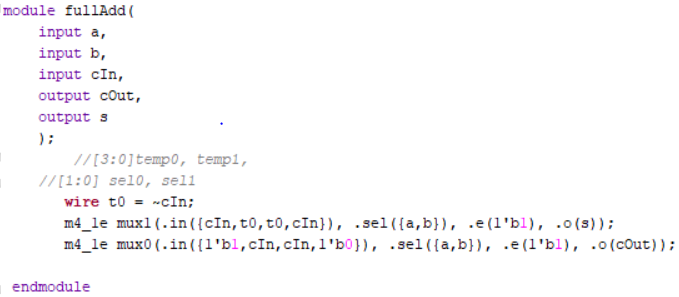


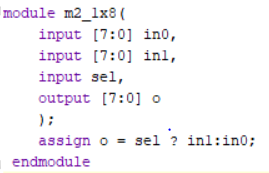


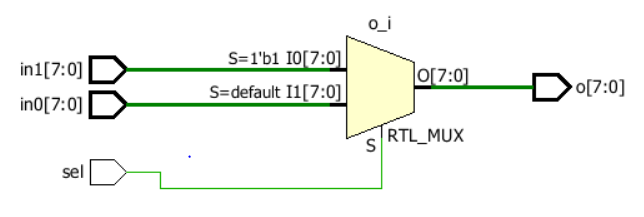
SegAdd



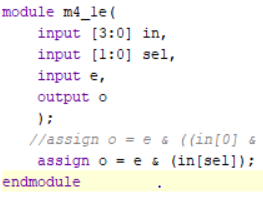
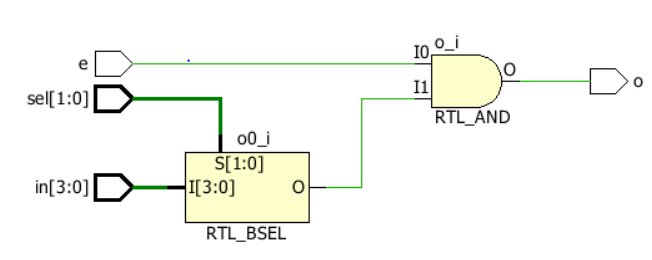
FullAdd



Mux 2\_8x1



Mux 4x1



Mux 8x1

