EECS 3201 :: Lab 3
Addition and Subtraction Calculator

Demonstration: Labs must be demonstrated during lab periods on the week of October 19-23.

Note: The TAs will be extremely busy towards the end of the lab. You are **strongly advised** to prepare as much as possible in advance, work efficiently in the lab, and to demonstrate as early as possible. **Leaving the demonstration to the last minute is at your own risk.**

Overview

In this lab you will implement a simple 4-bit calculator for addition and subtraction, using the seven segment displays on the DE10-Lite to display both the arguments and the result.

You may work in groups of **at most two**. Working by yourself is OK if there is enough equipment. Groups of three or larger are **not allowed.** There is no need to stay with the same lab partner as in previous labs, but you can if you want to.

To make the best use of your time in the lab, you should write your Verilog code beforehand. You can then use the lab time to debug your design using the actual board.

Scoring:

- This lab is scored out of 5 marks.
    - *Demonstration:* Once done, you will demonstrate your work to the TA. Correctly demonstrating the result is worth 3 marks. If you can't get the design to work, you can still demonstrate, but you will lose marks. *You will get a grade of zero if you don't demonstrate anything by the end of the lab period.*
    - *Question:* The TA will ask **each** group member a question, for example to modify the circuit. Correctly answering the question is worth 2 marks.
- It is possible for group members to receive different grades, for example, different grades can be received on the question.

Value:

- This lab is worth 2/10 of your total lab score and 2% of your final grade.

You will use your seven segment decoder module from Lab 2. If this is not possible for some reason (e.g. you couldn't get it to work), you can use a module from somewhere else (e.g., a classmate, or an LLM), as long as you give appropriate credit.

Ripple Carry Adder and Subtractor

In the lecture we saw how to design a ripple carry adder, starting with a full adder. We also saw that a ripple-carry adder could implement subtraction, using the two's complement method.

Implement a four-bit adder in Verilog, as follows:

- Create a module that implements a full adder.
- Using four full adders, connected as described in the lecture, implement a four-bit ripple-carry adder.
- Modify your four-bit adder with an additional input bit, called s, so that if s = 0, the adder implements addition, and if s = 1, the adder implements two's complement subtraction.
- Put your adder/subtractor in a module with the following inputs and outputs:
    - Inputs: Two four-bit binary numbers, a0 and a1; and the subtraction control bit s
    - Outputs: One four-bit binary number as the result (a0+a1 for addition, and a0-a1 for subtraction); and the carry out bit "cout" from the final full adder

NOTE: DO NOT USE THE BUILT-IN "+" OPERATOR. You must implement your adder as described above. The TA may ask to inspect your code.

Inputs and Outputs

Inputs a0, a1, and s should be assigned to switches on the DE10-Lite (which ones are up to you).

For outputs, you should use the seven-segment displays to display the digit corresponding to both inputs (a0 and a1), as well as the result. Use HEX5 (the leftmost display) for a0, HEX3 for a1, and HEX1 and HEX0 (the rightmost displays) for the result.

To do this, you will need to use three seven-segment decoder modules: one for a0, one for a1, and one for the result. You can use the one you wrote in Lab 2, or you can use one from a friend or from the internet. (**Important note:** if you didn't write your seven-segment decoder yourself, **you must credit the source; do so in a comment in Verilog.** Failure to credit your sources is an **academic honesty violation**.)

What about the carry out (cout)?

The output cout means different things in addition and subtraction:

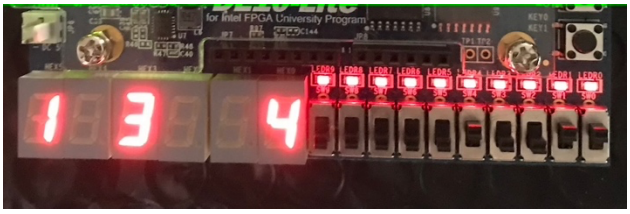
- When doing addition, cout = 1 if the result is greater than hexadecimal F, so that there is a 1 in the position of the next digit. That is, if cout = 1, then the result is in 10, 11, 12, …, 1F. On the other hand, cout = 0 means that the result is completely expressed in the four bits of the output, i.e., the result is in 0, 1, 2, …, F. You can check this with an example, e.g. try adding 8+9 in binary and see where the last carry goes.
- When doing subtraction, cout = 1 if the result is positive and cout = 0 if the result is negative. If the result is negative, then the four-bit output of the adder expresses the result in two's complement form, i.e., in hexadecimal: F = -1, E = -2, and so on.

Displaying the result: On HEX0 you will display a digit corresponding to the four-bit output of the adder. You will control HEX1 using cout, as follows.

- For addition:
  - If cout = 1, a 1 should appear on HEX1.
  - If cout = 0, HEX1 should be **blank**.
- For subtraction:
  - If cout = 1, HEX1 should be **blank**.
  - If cout = 0, HEX1 should contain a minus sign (only the middle LED illuminated), and the digit on HEX0 should undo the two's complement. For example, if the result of the subtraction is E (= -2 in two's complement), HEX0 should display 2, not E; together, HEX1 and HEX0 should read "-2".

Examples

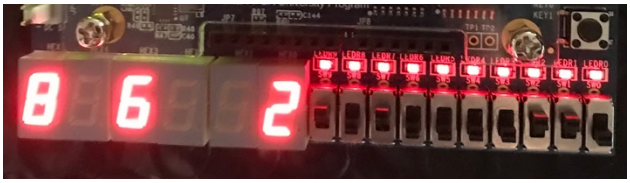
Addition: 1 + 3 = 4



Addition: 9 + 8 = 11 (in hexadecimal)



Subtraction: 8 – 6 = 2



Subtraction: 5 – 7 = -2



In the lab:

- Compile, assign pins, and upload your program to the board. (It is up to you which switches and display to use on the board. Pin labels are given in the DE10-Lite user manual.)
- Check for errors and fix them.
- Once you are happy with your design, demonstrate it to the TA. The TA may ask to see your Verilog module, and will also ask each group member a question about your design.