EECS 3201 :: Lab 3
Addition and Subtraction Calculator

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; and
- Create a video to demonstrate your work.

Value: This lab is worth 5/20 of your total lab score.

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 the internet), as long as you give appropriate credit.

Adder and Subtractor

Implement a four-bit adder in Verilog. You might do it as follows (this was explained in the lectures for Section F):

- 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 2's complement subtraction.

You might also do it with procedural Verilog, i.e. performing the calculations within an "always" block. Either way is correct and worth full marks, as is any other method that produces the result specified below.

Inputs and Outputs

However you implement the circuit, the inputs and outputs to your module should work in the following way:

- 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

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 HEX0 (the rightmost display) 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**.)

Example outputs are given below.

Completing the lab perfectly up to this point is worth 5 technical points out of a possible 8 (see "Scoring" below).

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. This is natural if you use the 2's complement method: in this case, 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. If you don't use the 2's complement method, it might not work this way, but cout should still indicate the sign of the subtraction: 1 if positive, and 0 if negative.

Modify your output as follows, using the HEX1 seven-segment display (the display to the left of HEX0, which displays the digit corresponding to the four-bit output of the adder).

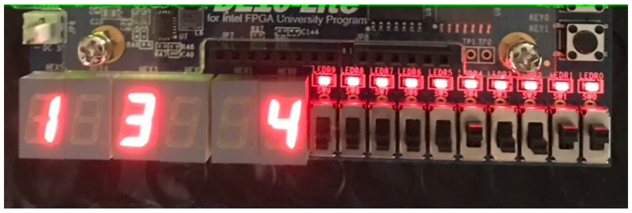
- For addition (*):
    - If cout = 1, a 1 should appear on HEX1.
    - If cout = 0, HEX1 should be blank.
- For subtraction (**):
    - If cout = 1, HEX1 should be blank.
    - If cout = 0, HEX1 should contain a minus sign (only the middle LED illuminated), and the digit on HEX0 should display the magnitude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

When implementing this part, describe your approach in Verilog comments.

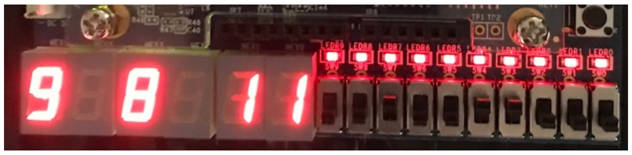
Perfectly implementing (*) is worth an additional 1 technical point, while perfectly implementing (**) is worth an additional 2 technical points.

Examples

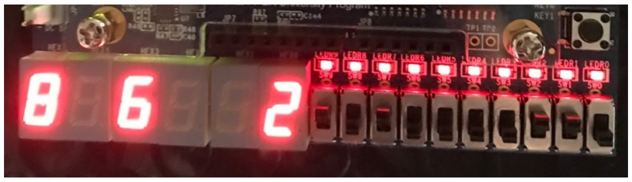
Addition: 1 + 3 = 4



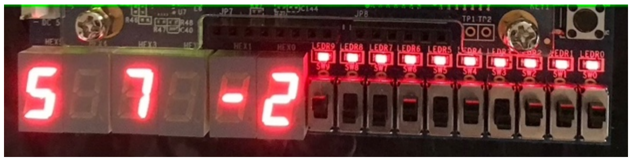
Addition: 9 + 8 = 11 (in hexadecimal) (*)



Subtraction: 8 – 6 = 2



Subtraction: 5 – 7 = -2 (**)



Deliverables

As before, your video should include you stating your name and student number, followed by (without breaks/cuts) a short demonstration of your program.

Your video must include one example each of the following:

- Addition of two numbers with a sum less than or equal to hexadecimal F (i.e., cout = 0); and
- Subtraction of two numbers with a *positive* result.

If you completed part (*), also include:

- Addition of two numbers with a sum greater than F (i.e., cout = 1)

If you completed part (**), also include:

- Subtraction of two numbers with a *negative* result.

Your video submissions must be no more than 90 seconds in length.

Submit your video and all Verilog code files via eClass.

Scoring

The lab is scored out of 10: 2 points for your video, and 8 technical points.

Video: 0/2 if missing, 1/2 if it does not conform to the specification, 2/2 if correct.

Technical points: Scoring is described above. Minor mistakes may lead to point deductions.