CMSC389E Project 2: Arithmetic Logic Unit- Adders & Multipliers

Assigned **Monday October 11**

Due **Wednesday October 20**

1 The Joy of Mathematics

Now that we're done building the logical portion of the ALU, it's only right that we attack the arithmetic portion. In this project, we're going to use our newfound knowledge of Adders to design some much needed mathematical architecture for our ALU. Specifically, we're going to build in the ability to **add** and **multiply** 3-bit numbers.

This section will focus a lot on how you go about implementing this architecture in terms of where you place the adders and how you debug them. As such- a word of advice before you begin. The best way to do this project is to build one single adders, properly unit test it with a bunch of test cases, and then when you know that it works, copy paste it over and over to achieve the 3-bit adding functionality that we're looking for. (And afterwards, the multiplier).

This project comes in two parts- the first is the implementation of the 3-bit adder, and the second is the implementation of the 3-bit multiplier. We have split testing into two distinct phases for this reason- first, you will build your 3-bit adder and test it, then you will build your 3-bit multplier.

2 Conceptual Overview: Addition and Multiplication

Adddition and multiplication are processes that we usually can perform as second-nature now, but part of learning how to be a good computer architecture designer is figuring out how one can take these mundane tasks that we as humans perform very easily, and distilling them down to systematic, methodical tasks that any set of logic gates can perform.

These answers come to us in the form of Adders, one of the most basic logical circuits for Arithmetic. You will be building quite a few of them for this project. (Or, preferably, building one of them and then copying them a bunch using your copying method of choice)

3 Part A: Building the Adders

This project builds upon your Project 1 Implementation. In order to correctly load it up, please copy your Project 1 world (you can do this manually, in the `saves` directory of Minecraft) into a Project 2 world. Ensure that you name it Project2. Then, open the world and remove all the project 1 input/output blocks so that you may load the new project.

We will start by loading the appropriate blocks for the first part of this project. Specifically, we will be building out the 3-bit adder. To do this, run the following command.

```
/test load 2
```

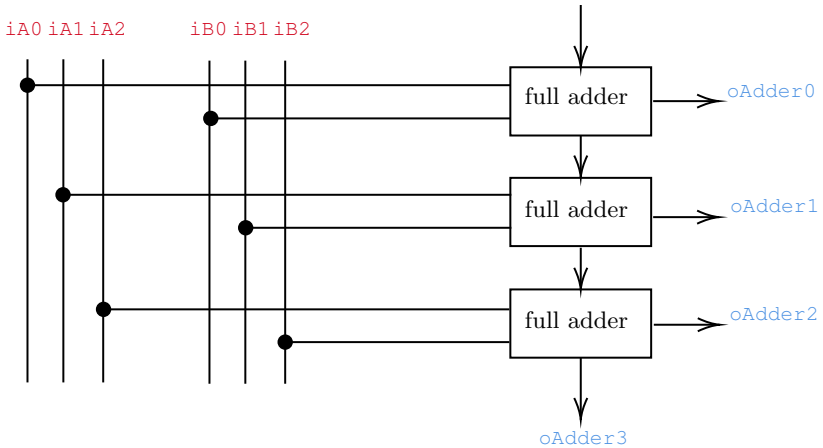
Firstly, you will see that you have the same input blocks at your disposal. You will be building a three bit adder using both input buses, and deliver that output to the outputs labelled `oAdder0`, `oAdder1`, `oAdder2`, `oAdder3`.

4 Part A: Schematics

For reference, the adder you will be building and testing looks like this. Remember, in order to achieve a 3-bit adder, you'll just be chaining full adders together and connecting the carry-out bits of previous adders to the carry-in bits of the next adders.

Note that you will have one more carry-out bit to deal with after the fact. This means that you will take the sum of the first pair of adders and deliver it to `oAdder0`, the sum of the second pair of adders and deliver it to `oAdder1`, the sum of the third pair of adders and deliver it to `oAdder2`, and finally take the carry-out from the third pair of adders and deliver that to `oAdder3`. Make sure, of course, that you are taking the carry-out bits of each pair of adders and delivering them to the carry-in of the appropriate other adder when necessary.

You are expected to extend your output buses from your implementation of Project 1 in order to create this architecture. Refer to the diagram below.



5 Submission Of Part A

You will be submitting this solution **with** the Part B solution for this project. You will only be submitting one map, however, so for this part, simply take a screenshot of your tests passing the adder tests for part A.

Running the following command should invoke the tests.

```
/test start
```

Once you pass all the tests, take a screencap of them passing, and save it. You will include this screenshot in the final submission. When you are ready, move onto Part B of the project below.

6 Part B: Building the Multipliers

Now that we've built the adders, this next bit should be easy. Recall that in class we spoke about multiplication in our particular digital logic setup just being repeated addition. In that same sense, we're going to simply take the addition circuit that we just produced, make a copy of it, and extend it to be a multiplication circuit.

Since we haven't been over this stuff in CMSC250 explicitly, I suggest you understand the relation between long multiplication and the actual multiplier circuit before going ahead with your implementation. You'll find this in the form of a step by step instruction set in the most recent slide deck for the class. It will help with implementation, and will help with testing.

For part B of this project, you'll make a new world by copying all of your content from Part A (a.k.a your Project2 world) over into a new world called Project3. (Same process as you went through to create the Project2 world as described earlier)

As you did in Part A, ensure that you are able to load the testing setup for this portion of the project. You may do this using the following command:

```
/test load 3
```

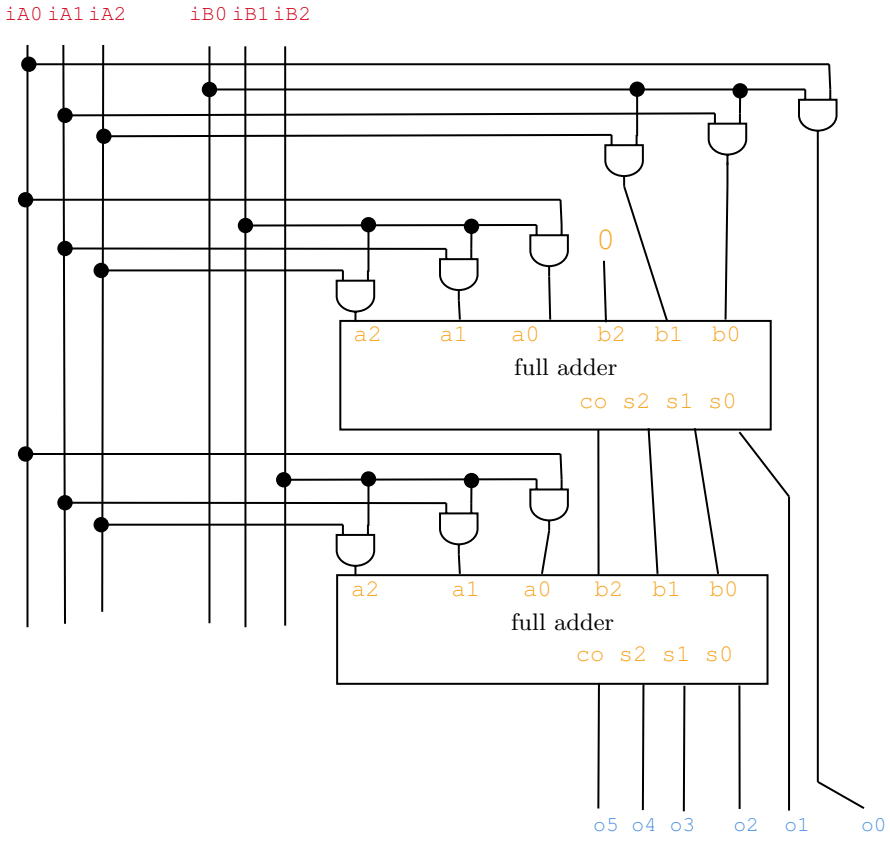
You will see that the in-nodes remain unchanged from the previous part of the project. This is to be expected- instead of **adding** two 3-bit numbers, we will instead be **multiplying** them.

As we mentioned previously, this is quite the trivial task after you have taken care of the creation of the Adders. You will merely take these inputs and feed them into a setup consisting of two 3-bit adders in order to produce a meaningful solution. Keep in mind that

our multiplication solution will be 6 bits of output here, as we will now have to be dealing with larger output numbers than what mere addition can produce.

7 Part B: Schematics

As you may recall from class, **(combinatorial) multipliers are just large groups of adders**. As such, we have defined Part B of this project in terms of the 3-bit adder you have just built! Just a bit more copy pasting and connecting, and you're good to go!



In this diagram, the blocks labelled 'full adder' are simply just copy-pastes (use structure blocks, clone, or worldedit!) of the adder you created in Part A. Just string those together with a few `AND` gates (which you can also copy over from your logical ALU section) and you should be good to go.

You'll note that input `b2` of one of the full adders is an orange zero. That just means that no signal should be going in there, i.e. that input should always be zero.

8 Testing Part B

It's worth noting that at this point, the speed of the actual redstone signals traveling through the circuits catches up with us. In other words, our CPU is showing its first signs of settling time! As mentioned in the conceptual overview section, it's important that we take this settling time into account when we attempt to avoid race conditions.

For this reason, we'll want to run the test command here with a bit of a delay, between 1 and 3 seconds, if we want to ensure correctness in our testing and output.

You can do this by running the test command with the following syntax, where number of seconds represents the delay that you want to enforce between each test.

```
/test start <number of seconds>
```

9 Submission Of Part B

You will be submitting this solution **with** the Part A solution for this project. You will only be submitting one map, however, so for this part, simply take a screenshot of your tests passing the multiplier tests for part B.

Running the following command should invoke the tests.

```
/test start
```

Remember to add a delay this time, especially if you see some tests failing unnecessarily. We will most likely be running your tests with a delay of 2 seconds, so include a small comment on your submission if your implementation needs more settling time than that.

After all is done, submit the following three things to the ELMS assignment.

- Screenshot of Project 2 Part A (Project2) tests passing
- Screenshot of Project 2 Part B (Project3) tests passing
- Your final Project3 world, containing all of your work from P1 up till P2 now. (Logic gates, Adder, Multiplier)

Congratulations, you're done! As always, ping us on Piazza with any questions, or send us an email.