Lab 5: Addition Circuits!

Copied from: https://www.cs.hmc.edu/twiki/bin/view/CS5/AdvancedAdditionGold on 3/20/2017

[30 points; individual or pair] (filename: `hw5.circ`)

Getting started with Logisim and the hw5 starter file

This week you will design several circuits using a digital circuit design tool called Logisim. You'll use Logisim throughout this week's hw (with the same hw file, in fact).

There are two similar, but not fully compatible versions of Logisim.

Try the original Logisim (top link) first. If it doesn't work, try the newer *Logisim Evolution*:

- Here is Logisim, the original version (with the starter hw file, `hw5.circ`)
  - Unzip this folder and try running the Logisim version appropriate for your operating system (Win or Mac)
  - If that does not work, try double-clicking the `jar` file
  - If you need Java, the application may prompt you...
  - Another approach is to run `java -jar logisim-generic-2.7.1.jar` at the command line
- *If you can't get Logisim working*, try Here is Logisim Evolution, the newer, but still-in-development version (with its own starter file, named `hw5evolution.circ`)
- Another alternative is to use the lab machines, which have Logisim (trouble? ask!)

Need Java?

If running Logisim requires you to install Java / Java's runtime environment (JRE), that is totally ok...

- If you encounter any trouble, ask!
- ***Mac OS X*** Here is the link to the spring 2017 Java RE6 download page...
- ***Windows*** Here is the link to spring 2017's Java (JRE) installation page

Use the hw5 starter file!

For this week's homework, all of your circuits will be in a ***single*** file. *Don't start from scratch - please!*

Usually, double-clicking the starter file ***doesn't*** work. Use *File* - *Open* from Logisim's menus.

Be sure to use the starter file for your version of Logisim:

- For the original, "plain" Logisim, use `hw5.circ`
- For Logisim Evolution, use `hw5evolution.circ`

These two starter files are in the `zip` files, along with Logisim, above.

- **Warning**: when you submit your file, you will need to submit one or the other of these two files.
- There are two spots in the submission system, but you should *only* submit one file -- in the appropriately named spot.

Lab 5: *Circuit addition !*

Circuit design!

- To simplify things, we'll use the name `hw5.circ` throughout this lab, even if you're using `hw5evolution.circ`. *Please don't change the filename* - it will need to stay as-is for submission.

- The `hw5.circ` file (again, or `hw5evolution.circ`) is where you will write and save all of your circuits for this assignment.

- **Notice** that in your Logisim explorer pane (the upper left part of the Logisim screen) there are icons
labelled `MyXOR`, `FullAdder`, `4bit_Ripple_Carry_Adder`, `4bit_Multiplier`, `3by2_Divider`, and perhaps some others. These are all of the parts ("subcircuits") of this week's homework. This lab covers only the first three of those subcircuits.

- Many people have found that they complete those subcircuits within the lab time. If so, feel free to work ahead to the others (as explained in the homework problems) if you would like and have time. Or, feel free to submit just these in-lab parts and then come back to the final pieces later.

Part 1: Tutorial and XOR

- Start up your Logisim. From within the program, select File, then Open, and navigate to the `hw5.circ` or `hw5evolution.circ` file you downloaded.

- Go to the "Help" menu and select "Tutorial". Read through the beginner's tutorial up to and including Step 4, *testing your circuit*. As you do so, create the XOR circuit the tutorial is describing. Do this in the "MyXOR" tab of your Logisim file. This is pretty short reading, and don't worry if you don't digest every last "bit"!
    - At least you'll know what's in the tutorial so that you can go back and find the details later if you need them... .

- With `hw5.circ` open, ***double-click*** on the `MyXOR` icon in the explorer pane. For the moment, you'll be working on that subcircuit. Your job here is to build a circuit to compute the XOR function using **only** `AND`, `OR`, and `NOT` gates.

- Be sure to test your XOR circuit by changing the inputs with the "poke tool" (the little hand in the upper left of the menu bar). Check to see if you are getting the right outputs for your inputs. Save the file by going to the "File" menu and clicking on "Save".

Part 2: Designing and building a Full Adder

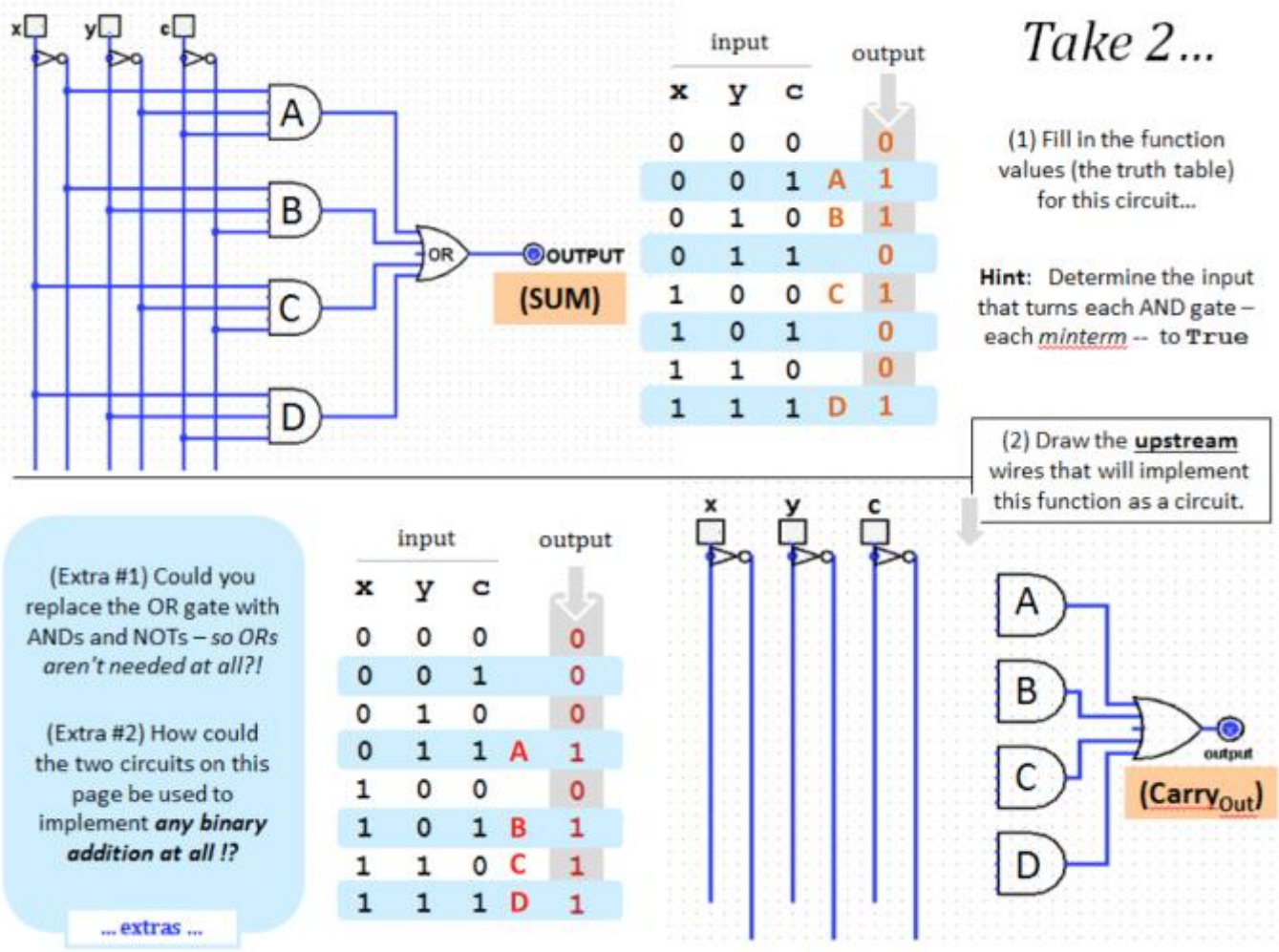
- ***Double-click*** on the **FullAdder** subcircuit tab in your `hw5.circ` file...

- Now, you should be in the *Full Adder* circuit's explorer pane. You'll notice that we've provided the labeled inputs and outputs. ***Please use those inputs and outputs!*** - they help us grade the files:
    - Inputs `X` and `Y` are the two bits to be added, and `CarryIn` is the carry from the previous column.
    - Similarly, the two outputs are already placed at the bottom. You can move these if you need to, but please keep their relative positions the same so that we can easily test your circuit!
    - Notice that we have rotated the input and output pins from their normal default orientation. This was done by clicking on the item and selecting the desired rotation in the attribute pane in the lower left of the screen.

- Your task is to build a full adder (FA), which is the circuit you paper-designed in class on the *second* full-page exercise (at least in gold...).

  Recall that a full adder is a device that takes **three** inputs:
    - two bits to be added, and the "carry in" from the previous column of addition.
    - The FA then computes **two** outputs: the "sum" bit, and a "carry out" bit

- Thus, the first step is to look up (e.g., from class notes) the full truth table for a FA (full adder). The image below has most of it:



- Your full adder should use the minterm expansion principle to create four AND gates for the "sum" bit and four AND gates for the "carry out" bit. You designed (on paper) a full adder in class. Now is the chance to implement it in a *verifiable* fashion... !
  - Each of the two outputs demonstrates a separate application of the minterm expansion principle.
  - Please don't try to simplify your circuit and please don't use gates other than `AND`, `OR`, and `NOT`.
  - The purpose of this problem is to practice using the minterm expansion principle - not to optimize!

- Be sure to test your FA before moving on. *Your FA needs to work*, because you'll use it to build the four-bit ripple-carry adder, next!

Part 3: Building a 4-bit ripple-carry Adder

- ***Double-click*** on the ripple-carry adder subcircuit in your `hw5.circ` file...

- Now that you have a full adder (FA), you will build a 4-bit ripple-carry adder. Recall that this device takes two 4-bit numbers and adds them up. Let's call the digits of the first number X3, X2, X1, X0 where X0 is the least significant bit (the rightmost bit) and X3 is the most significant bit. Similarly, let's call the digits of the second number Y3, Y2, Y1, Y0. We wish to compute:
- ```
    X3 X2 X1 X0
  + Y3 Y2 Y1 Y0
  --------------
  ```

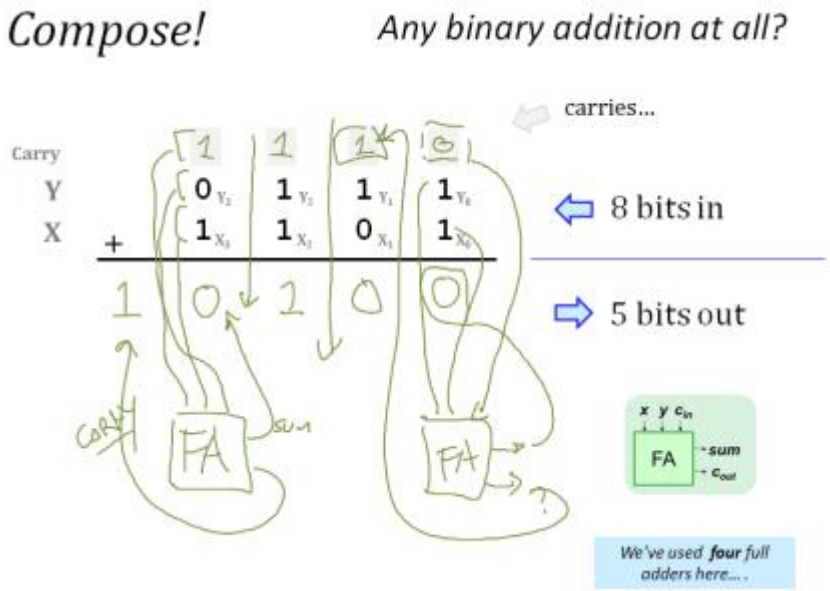
- Notice that although there are only 4 bits in each of the two numbers, the sum can have 5 bits due to a carry!

  After you've double-clicked on the "4-bit Ripple-Carry Adder" icon in the explorer pane, you can "plop" full-adders into your 4-bit ripple-carry adder by *single-clicking* on the "FA" icon from the explorer pane (since you've already designed the FA). This will create a "box" that represents your FA. After all, now that you've designed the FA, you don't particularly want to see all of its details each time you use it! If you move your mouse ("hover") over this box, you'll see that it says "FA". If you "hover" over the "pins" (the little input and output markers on the box), it will show you what those pins are. This is because we labelled those pins when we designed the FA.

  You can move the inputs and outputs that we've provided, but please keep their relative positions so that we can find them easily.

  One last thing! You'll need to have a carry-in of 0 on the far right of your ripple-carry adder. One way to do this is to add an extra input and set its value to be 0. This is not a great solution, since we really don't want the user of our ripple-carry adder to be able to change that carry-in value. A better choice is to go the explorer pane, click on the "Wiring" folder, and then look down that list to the item labeled "Constant". Plop one of these down in your circuit. Then click on it and you can alter the value of that constant in the attributes pane. This is now a constant value that your circuit can use!
  ```

- In class we sketched (roughly) how four full adders would be used to create a 4-bit ripple-carry adder:



Submitting your file

Test your circuit and then be sure to save it! If you submit it without saving, you will submit the original empty file, rather than your solutions!!

Keep in mind that for this homework, the other problems will be submitted ***in the same*** `hw5.circ` file - you can resubmit it later with more subcircuits filled in... .

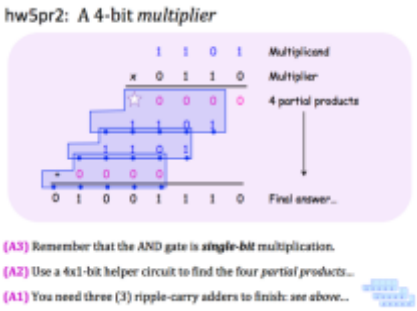
Continuing to the multiplier...

You're done with the lab! If you wish, continue on with the rest of the assignment.

- hw#5 for gold
- hw#5 for black

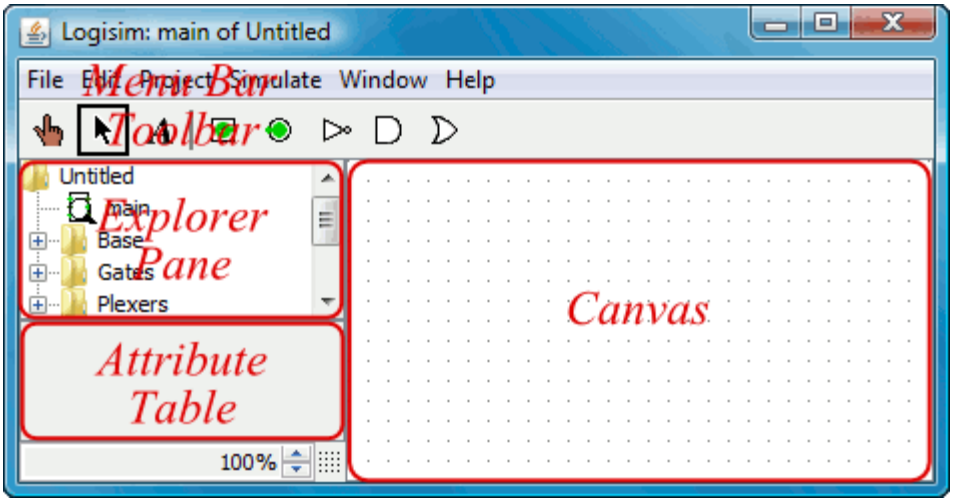
It's possible to get through the multiplier (and, on occasion, even the divider) in lab... if so, wonderful!

- Even if not, however, you'll have the chance to build these more sophisticated circuits later on this week... .
- Either way, good luck with Hw #5: it will certainly *multiply* your circuit-building enjoyment!



Side panel / circuit menus missing in Mac OS?

If you accidentally minimize the Explorer and Attributes Pane from the left side:



It can be frustrating to get it back! One command that we have tried (on the Mac) that has worked is (from the command-line)

```
defaults delete com.cburch.logisim
```

Alternatively, *sometimes* you can restart the program and they'll come back. If your problem persists, ask one of us!