Lab5 – Digital Logic and Logisim Tutorial

Introduction

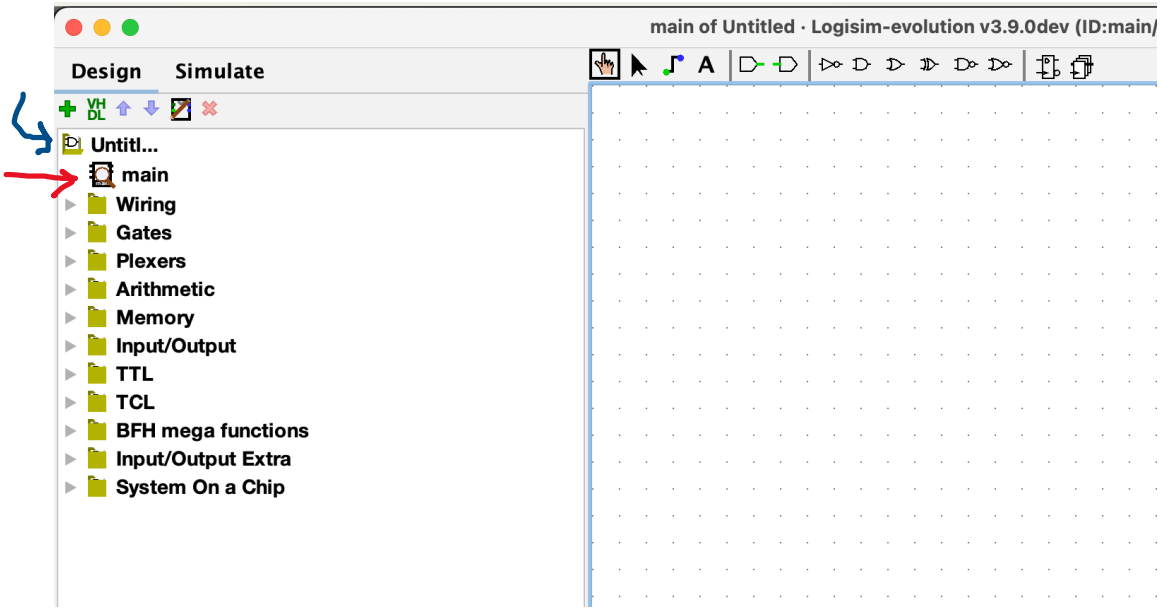
For this lab you will be playing with Logisim Evolution. You will use this knowledge in follow-up labs where we will be building various components on an ALU.

PART 1: Installing Logisim (Evolution)

If you have already not done so, start by installing Logisim evolution. The latest release can be found on their Github page. You can go here: https://github.com/logisim-evolution/logisim-evolution/releases/tag/v3.9.0. Logisim is a java program, but the release page includes installers for Windows, Mac and Linux. Note that the developers of Logisim do not sign their application, so opening it on the Mac for the first time requires one additional step. Its documented on their main readme page. https://github.com/logisim-evolution/logisim-evolution

PART 2: Building a Simple Digital Circuit

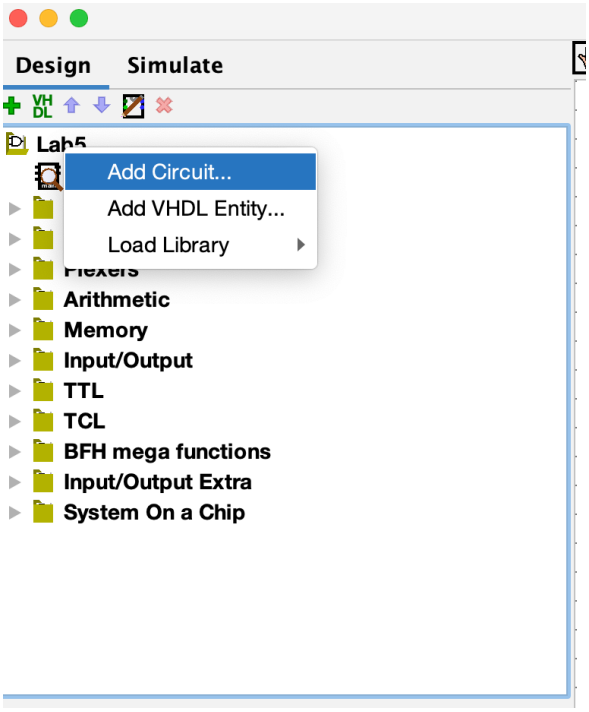
The first part of this lab will require you to build a simple digital circuit using Logisim. After launching Logisim, notice the upper left corner:



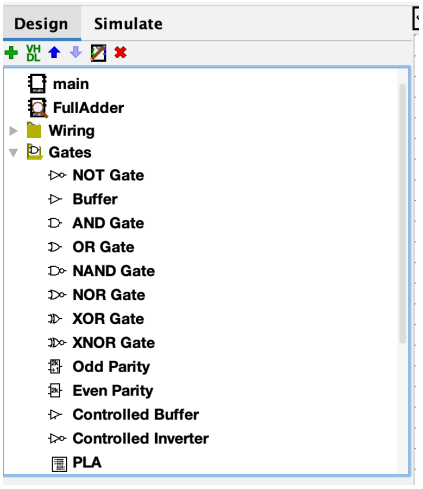
This is where various components can be selected and placed into circuits, as well as controls to manage the simulation. Notice "main" in the menu (highlighted by red arrow). This is for the main circuit, we will not be using this for this lab as our goal will be to learn how we can create foundational components and then build them up to a larger circuit. To begin, lets create a sub-circuit called a "fullAdder".

To start lets save this file using the "File"->"Save" option and call it lab5. After saving

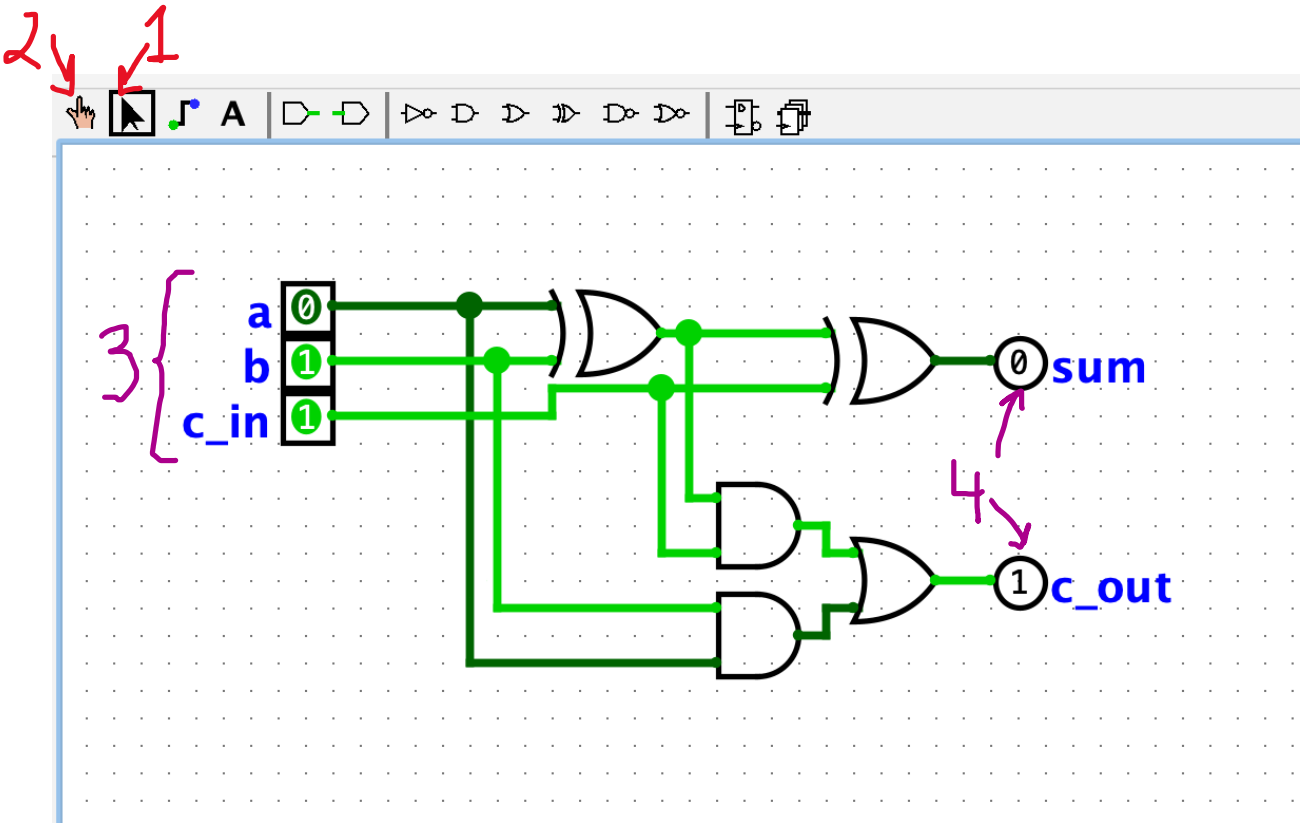
“Untiltled” (shown by blue line above) will change to “Lab5”. Right click on “Lab5” and select “Add Circuit...””



Name this (sub-circuit) “FullAdder”. Now FullAdder will be under “Lab5”, I also expanded the “Gates” submenu



At this point you can select these gates and build a digital circuit in the main window.



Play with the UI and create the circuit shown above, This is a full adder, defined by 3 inputs and 2 outputs. The full adder adds 2 bits and produces a sum output. What makes it a full adder is that the circuit includes a carry_in input and has a carry_out output.

Notice the 2 annotations:

1. The “Arrow” mode is used to build and edit the circuit
2. You can “simulate” the circuit using the hand-pointer icon.

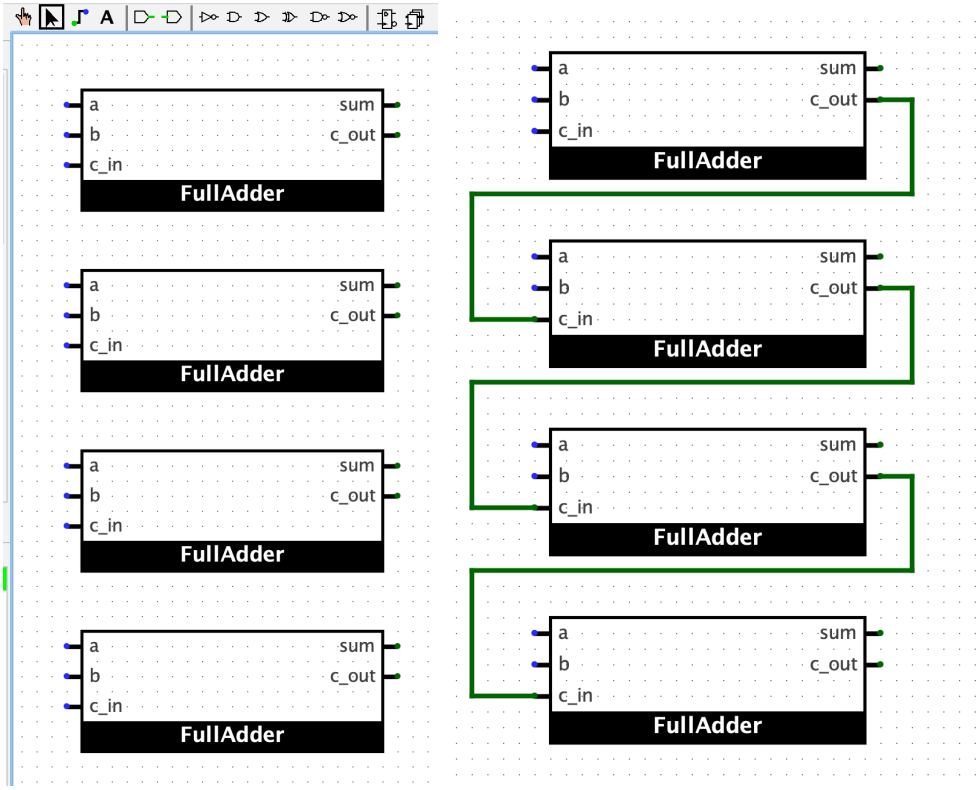
To setup the inputs and outputs (annotated 3 and 4) use the “Pin” control from the “Wiring” menu. Notice that the properties of a “Pin” must be set to “input” or “output”, by default they are set to input.

Continue to play with Logisim until you are comfortable with creating a simple circuit like above and setting up inputs and outputs to demonstrate that the circuit is operating correctly.

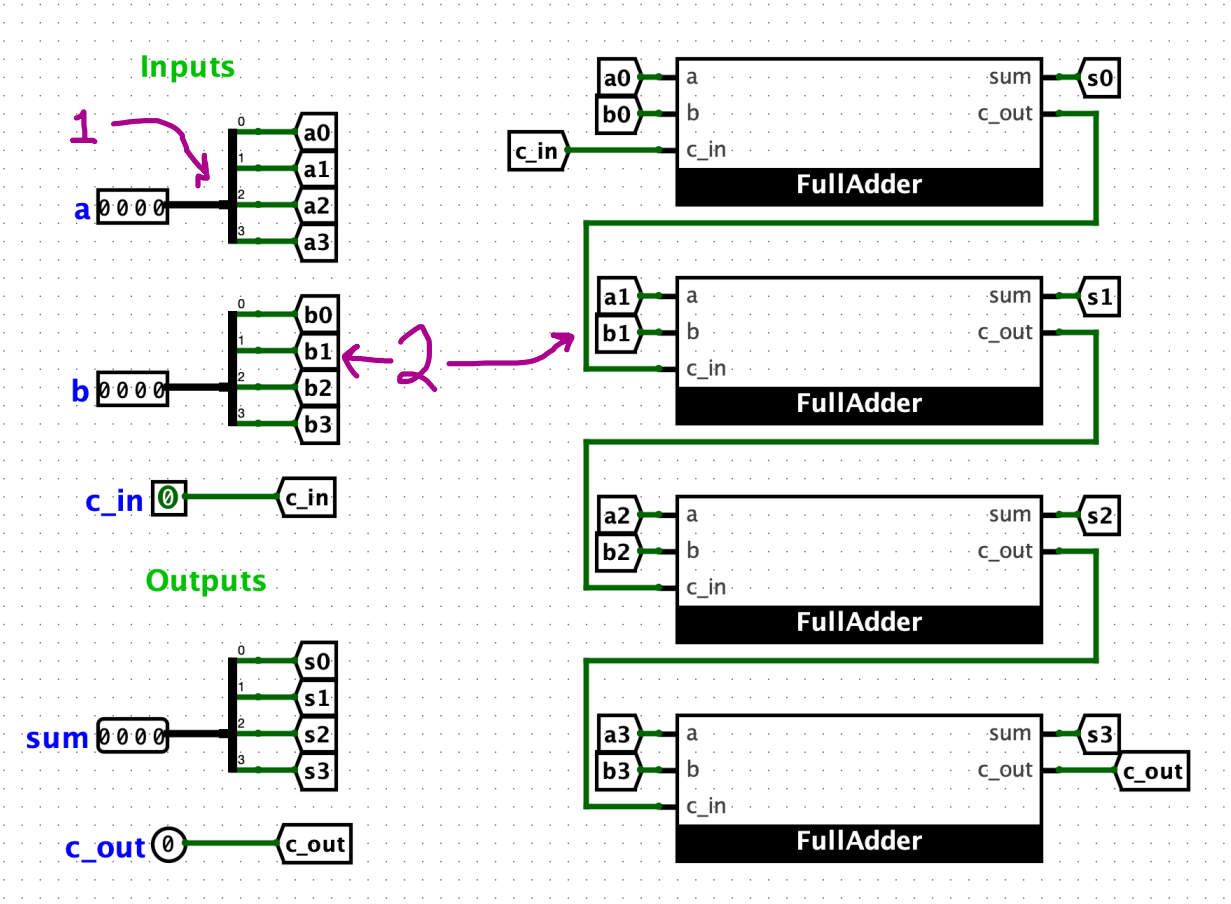
There is nothing to hand in for this part of the lab.

PART 3: Building up a more complete circuit from a sub-circuit

For this part of the lab we will create a 4 bit adder, using the single bit adder we created in part 1. To start, create a new sub-circuit (remember right click on lab5, and select “Add Circuit…”. Call this sub-circuit “FullAdder4Bit”. Now for this circuit, instead of using basic gates, use the circuit you created in Part 1.



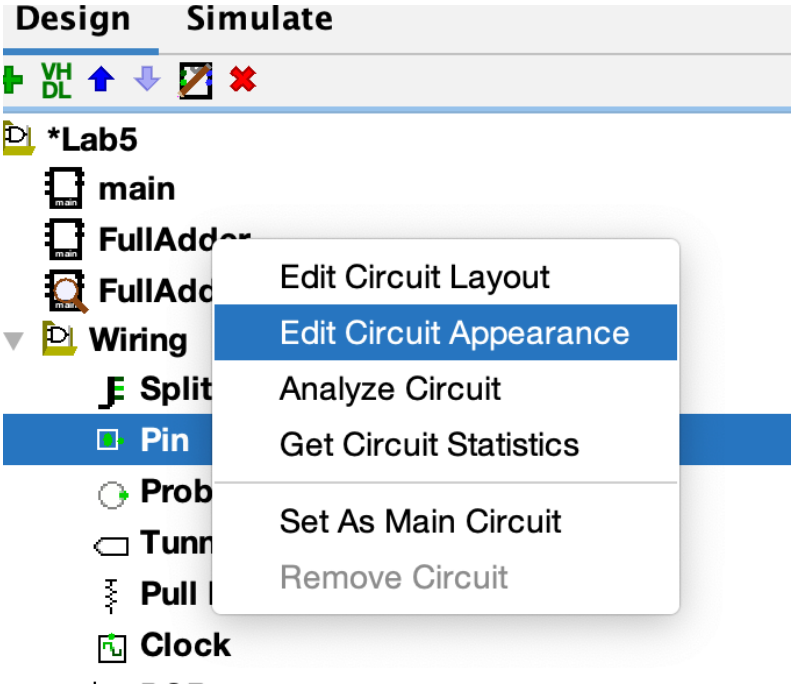
After dragging out 4 full-adders, wire the carry out to the carry_in bits (see above). We are now going to see how we can wire up things and keep things neat. See below:



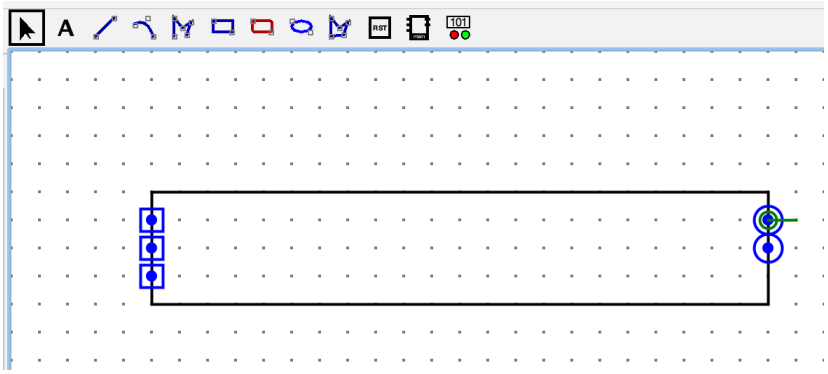
Wire up the circuit as shown above and test it out to ensure its working correctly. Notice that a, b, and c_in are input pins. Sum and c_out are output pins. For pins you can control the number of bits. Notice that a,b and sum are set to 4 bits, carry in and carry out are set to 1 bit.

We are also using some new things to keep the circuit neat. The "splitter" (see "1") takes a 4 bit value as input and splits it out as 4 individual bits on the output side. You need to set the properties to do this properly. Also notice "2". Instead of drawing wires all over the place, this is called a "tunnel" control. A tunnel can be used to keep things neat. Notice I pointed to "b1" thus, when b1 is set on the left, its value will be used as input to the second full adder.

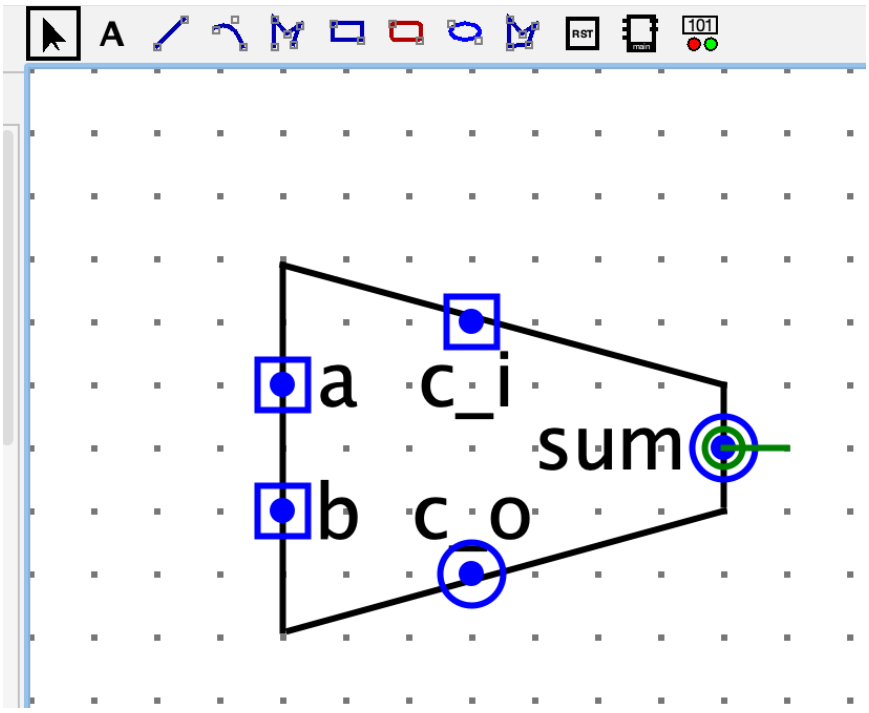
Lets conclude this part by approving the appearance of this circuit a little more. Notice the "Full Adder" is just a rectangle with inputs and outputs layed out for us based on the first circuit we created. Now right click on "Full Adder" (the first circuit we created), and select "Edit Circuit Appearance"



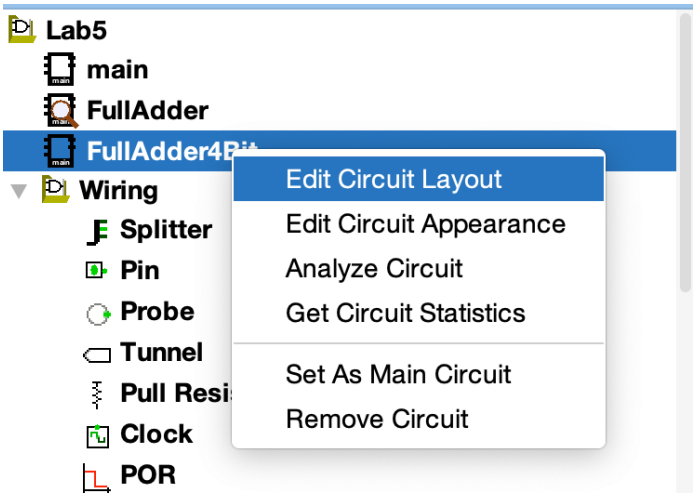
After you do this – you will see the default layout shown.



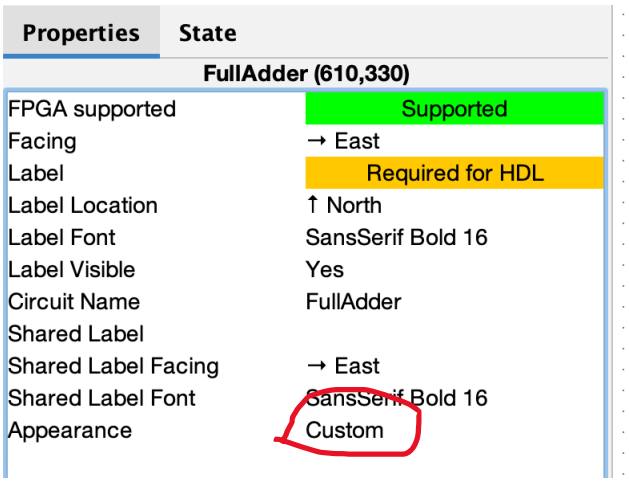
Lets improve the look and feel, play with the UI and create something that looks like the below:



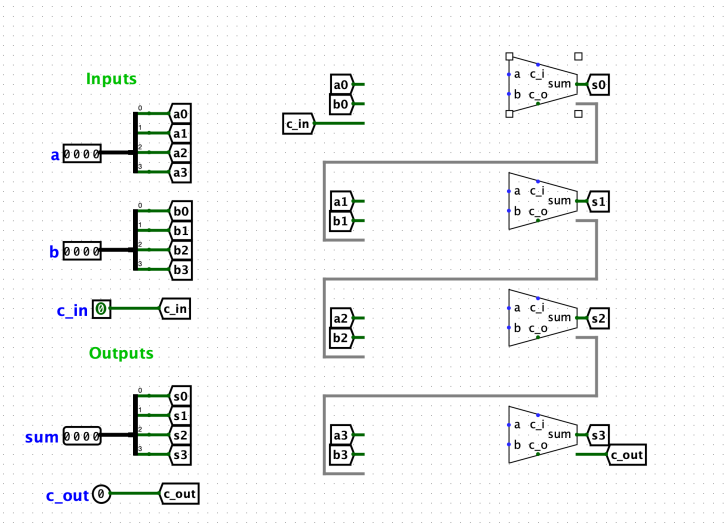
Now go back to the 4 bit adder and hit “Edit Circuit Layout”



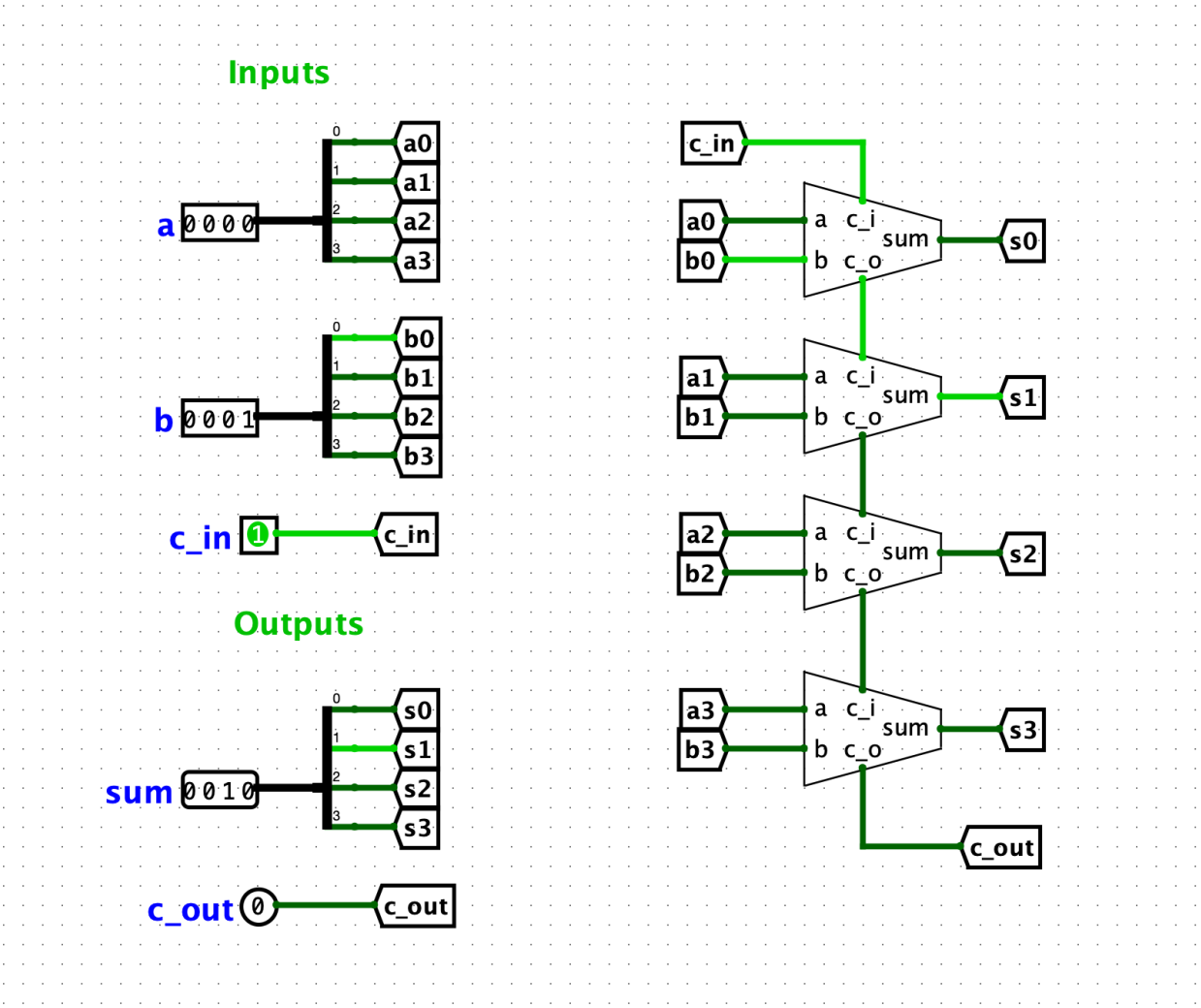
When the ciruit is displayed, click on one of the full adder boxes and change its Appearance property to “Custom”:



Now look at the circuit:



Starting to look a lot better, finally fix up the wiring and ensure the circuit works as before:

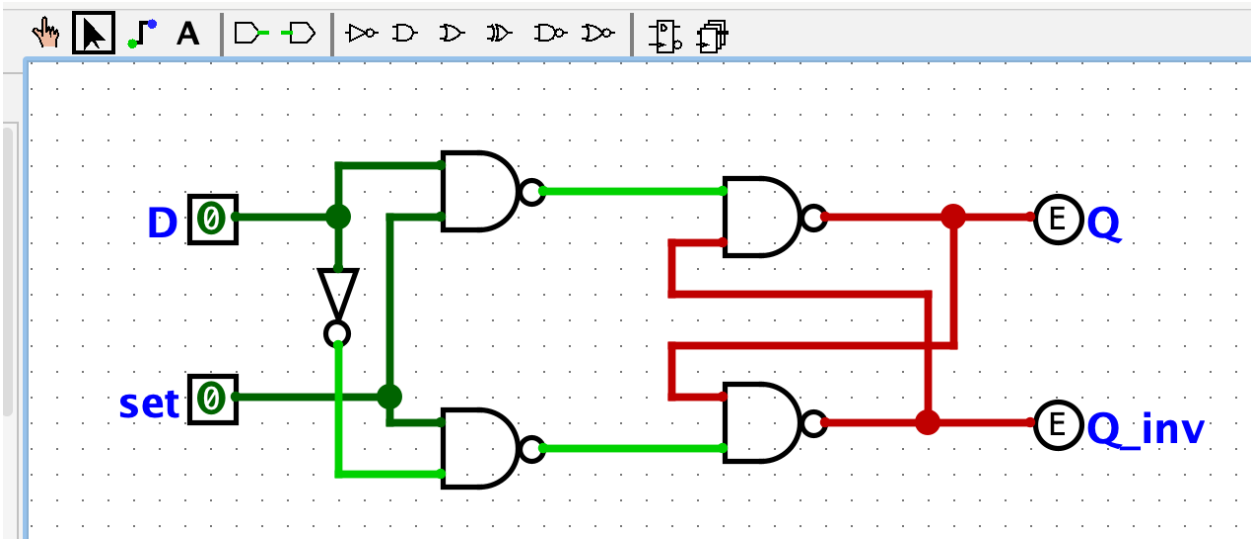


Looks much better ☺.

For this part, hand in a screen shot of your 4 bit adder.

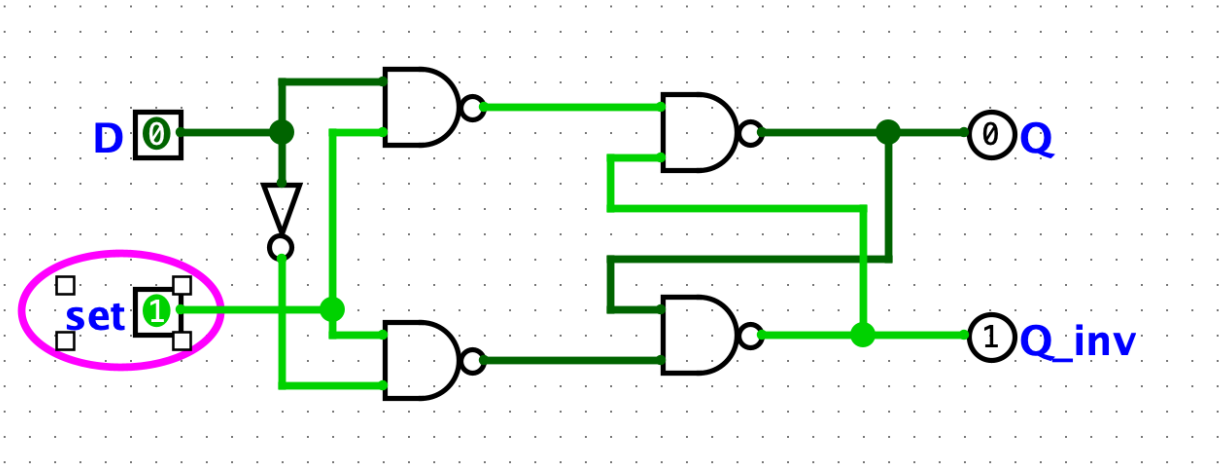
You should now be able to build simple circuits, understand inputs and outputs, and keep things clean with custom icons for sub-circuits and understand the use of tunnels. We will put this knowledge to work for the final parts of this lab.

PART 4: Playing with Combinational Logic
Lets start by creating a D Flip Flop:



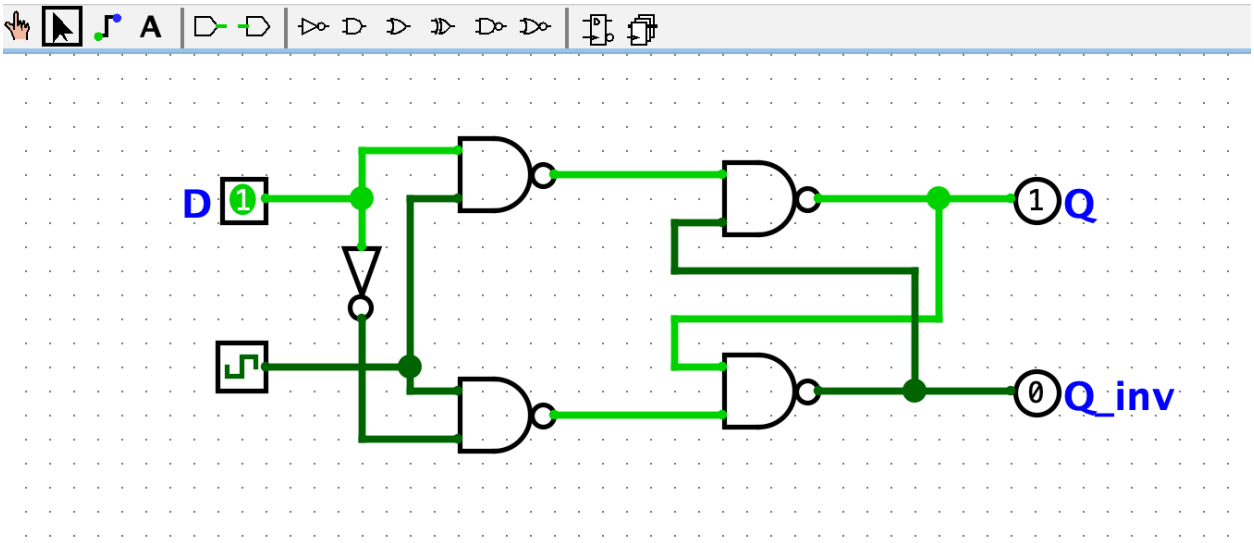
Notice in the above circuit the “red” error lines. This is because right after being drawn the output of the nand gates cannot be determined. Since the nand gates on the left are producing a logic-1 the nand gates on the right can produce an output of either zero or one based on the other input that are cross connected.

To overcome this, flip on the set bit. This will drive the output of the lower nand gate to a one, which will feedback to the upper nand gage to produce an output of a zero.

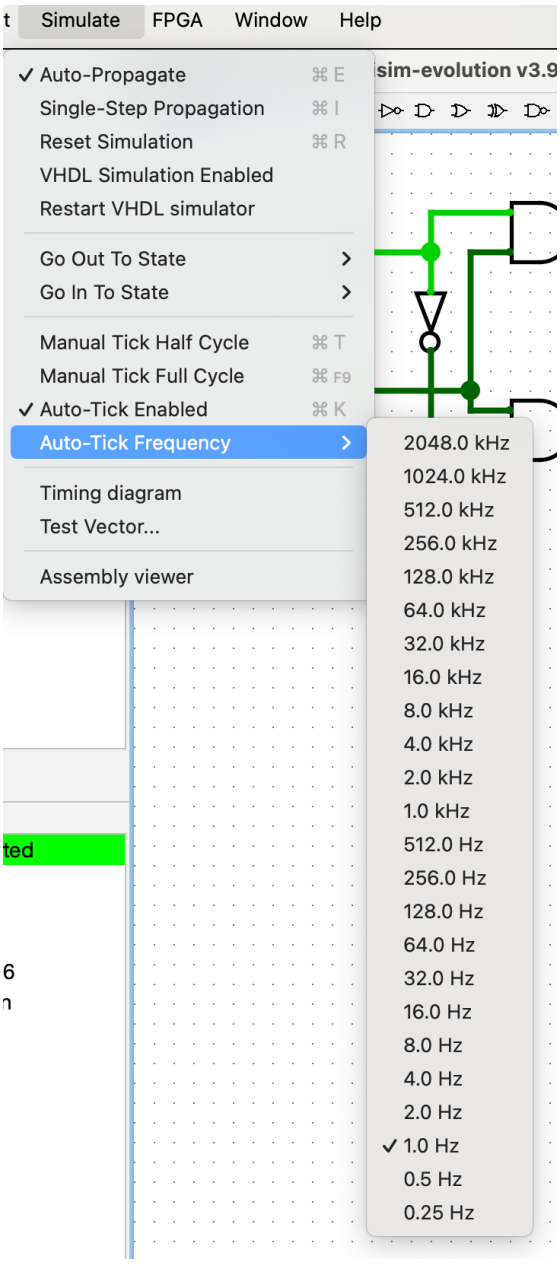


At this point, study the behavior of the circuit. When the “set” bit is 1, the value of Q=D, and when the value of set=0, Q is saved and does not change based on D. In other words: If SET=1, Q(t+1) = D. If SET=0, Q(t+1) = Q(t).

Next, we can replace set with a clock icon and play with the circuit:



Note you can change the clock frequency:



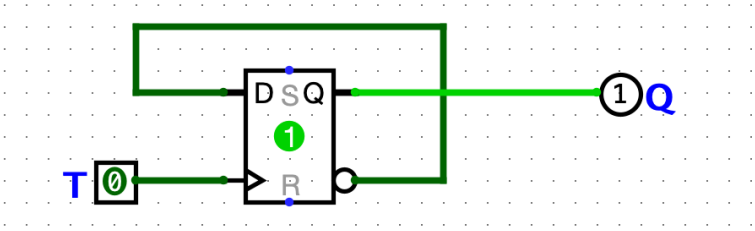
For this part of the lab there is nothing to hand in.

PART 5: The Toggle (T-Flip Flop) and Building a Counter

The truth table for the Toggle, or T Flip Flop is shown below. Its easy to implement a T flip flop by slightly modifying a D flip flop:

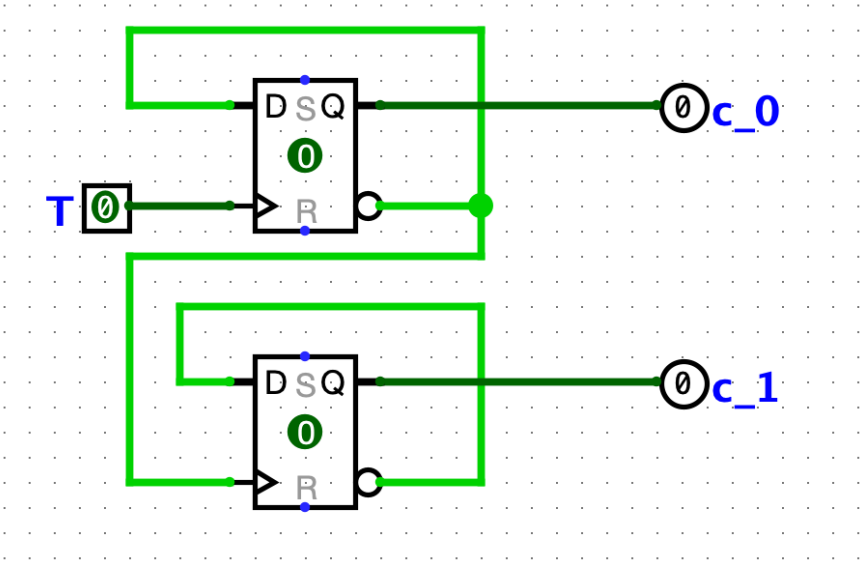
T	Q(t)	Q(t+1)
0	0	0
0	1	1

1	0	1
1	1	0



From above we implement a T flip flop using the Q' output as the input into the D flip flop, and use the clk value to trigger the T flip flop. Build the above circuit and see how it behaves. Every time T shifts from 0->1 it drives the output of Q' as the input into the D flip flop (which is the opposite of Q) resulting in the value shifting. Thus every time T goes from 0->1 the value of Q shifts, and the value of Q' shifts resulting in a "toggle", thus the T flip flop.

Now if we take the above and wire 2 of them together as follows, notice what happens:



This circuit acts as a counter shifting the bits of c_0 and c_1 through the pattern of "00", "01", "10", and "11". Play with this circuit and understand how it is working.

For this final lab part, extend the above and create a 4 bit counter. Also, wire up a 7 segment Hex digit display, and change the T pin input into a clock. Your final circuit should count continuously from 0-7 and then repeat.

Please hand in your final design, showing how the entire circuit is wired after you have it operating properly.