Lab #9: Introduction to Logisim
COE 0147: Spring 2014

You have 1 week to do this lab, as always. Each of you should submit your own solution via email according to the instructions at your TA's website. Each person must turn in their own copies of the lab. If you choose to work with a neighbor/partner, put your partner's name on your submitted copy of the lab. Submission timestamps will be checked. Late submissions will not be accepted. Follow instructions so that your program produces output in the correct format. You will have to zip your files into a .zip archive and submit it through lab submission website as directed on the lab website.

In this lab, you will experiment with Logisim and build some sample circuits. Logisim is a tool for designing and simulating logic circuits. It's surprisingly powerful, free (as in libre), runs on Java, and is available from http://ozark.hendrix.edu/~burch/logisim/.

There are **three** parts to this lab. You will create two different Logisim circuit files (.circ) and a truth table, and submit those zipped up using the email-based submission policy. Be sure to name your files appropriately as indicated in the instructions.

You must only use two-input gates. For example, each AND gate you create should have exactly two inputs. You will lose points if you do not follow this rule. Two-input gates most closely represent what is available when creating real circuits and let us better understand things like propagation delay and the need for simplification. Logisim by default creates gates that can accept up to five inputs. Be sure to only “use” two of those inputs (i.e., connect only two things up to each gate's inputs).

Part 1: Let's Build an Adder

Let's consider building a 1-bit adder. Recall that a 1-bit adder has 3 inputs: the first 1-bit 'number' you are adding, the second 1-bit 'number' you are adding, and a carry-in bit. There are 2 outputs: the answer (result of the addition) and the carry-out bit. Several 1-bit adders can be joined together to build an 'n'-bit adder (where 'n' is as large as desired).

Start up Logisim and create a new circuit. **Save this file as "lab9part1.circ".** Then, create a 1-bit adder, basing your work off of the circuit design shown on the next page:

- Square boxes are **input pins**. Their values are being used to compute a value.
- Circle boxes are **output pins**, as their values are being computed by the circuit and output to circuitry that wants the result of the 1-bit addition.
- Triangles are **not-gates**, as they invert whatever their input is.
- Green lines are wires. Bright green wires are 'on' (true, 1). Dull green wires are 'off' (false, 0). Blue wires are unconnected. We often have blue lines (e.g., when a 5-input AND gate only has 2 inputs connected, the blue input will be ignored).

All components are available from the main tool bar (below the "File" menu).

The poke tool (the hand icon) lets you change the values of input pins, to test different inputs. The arrow tool lets you add, select, and manipulate wires. You can "undraw" wires to shorten them, or use the delete key to remove the selected wire.

Be sure to add the labels.

You should **never ever** have red wires. These indicate an error, such as two outputs being connected together.

Here are a few useful tips when building circuits:

- Clicking and dragging creates **wires**. The point where you start the click and where you let go of the click, determines where the wire connects to other things. If for example, you click then drag the mouse over another wire, the two wires will not be joined. If you click to draw a wire, then let go of the mouse while it is positioned over top of an existing wire, then the new wire and existing wire will be joined. Green round ovals show where wires connect to other wires.
- You can easily **rotate** circuit components. For example, by default when you add an AND gate to a circuit, it faces east (the input comes from the left side and the output is on the right side). Clicking on the AND gate lets you change some of its properties. Changing the "facing" property rotates the gate. You can also click on a gate and use the up/down/left/right arrow keys to rotate it.
- Gates have a **size** property. Left click an existing gate and change its size (e.g., change the size to narrow). This can free up space in your circuit.
- Gates also have a property to set how many inputs they can use.
- You can quickly add to a component (e.g., a gate) by clicking the label tool (the "A" button on the toolbar) and then clicking a component. Use this to give each component a name (e.g. "C_out" for the carry-out pin).
- If you get an "oscillation apparent" error, it indicates that you are in some way creating an invalid "loop" of wires. For example, the output of a gate may indirectly loop back onto the same gate, causing the gate to keep switching (oscillating) between outputting 1 or 0. You should never get this error if you are doing things properly.

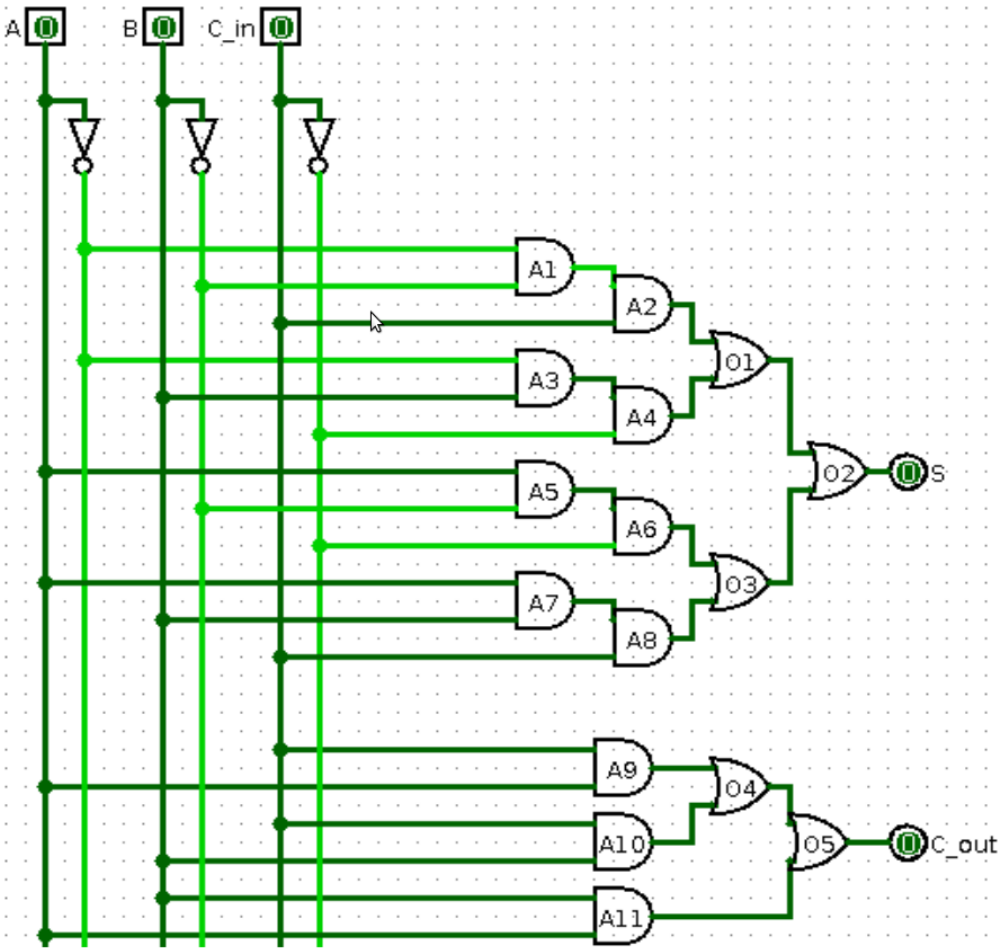

Illustration 1: 1-Bit Adder Circuit. Here 0 + 0 + 0 is being added, to produce 0 (sum) and 0 (carry out).

Part 2: Adder Analysis

The circuit that you built in part 1 implements the following Boolean equation, which is the Boolean equation for a 1-bit adder.

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$$

$$C_{out} = BC_{in} + AC_{in} + AB$$

By changing the values of the input pins, fill out the following truth table. A truth table describes the behavior of the circuit given all possible inputs. Feel free to type up the table in a word processor, spreadsheet editor, etc. Save this file as lab9part2.{ext} where {ext} is an appropriate file type extension (e.g., txt, xlsx, docx, csv). Do not submit a hard copy of this table.

A	B	C_{in}	S	C_{out}	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	O1	O2	O3	O4	O5
0	0	0																		
0	0	1																		
0	1	0																		
0	1	1																		
1	0	0																		
1	0	1																		
1	1	0																		
1	1	1																		

Part 3: Grandma Ann's Cookie Watcher

Grandma Ann, owner and founder of Grandma Ann's Crumbling Cookies, recognizes that to stay competitive in today's marketplace, she must integrate technology into her businesses in order to keep quality up and prices down.

To assist her business, she enrolled last year in COE0147. Using her acquired knowledge, she designed a cookie watcher. The cookie watcher “watches” multiple batches of cookies baking simultaneously. It is precisely timed to an individual recipe's cooking time. The cookie watcher alerts an employee to when a batch is ready to be taken out, as well as when a batch is ruined (overcooked). It is easily scalable to watch multiple cookies at once. Thanks to the cookie watcher, fewer cookies are being burnt and the cookies are being consistently baked the proper amount of time. As a result, Grandma Ann's profits soared. She has since retired to a private island in the Antilles.

In this section of the lab, you will recreate her cookie watcher circuit in Logisim. Naturally, Grandma Ann first tested her design in Logisim and then eventually soldered her own physical version. You will recreate her Logisim version. First, look at the final design on the next page.

Now, start the following steps:

1. Start a new Logisim circuit up.
2. Click “Project” then “Add Circuit” to add a subcircuit. We can design a subcircuit once, then instantiate multiple copies of it. Name this subcircuit “CookieWatcher.” We will first design this circuit. It will be capable of watching exactly 1 batch of cookies.
3. Add 2 input pins (square) and give them the proper labels.
4. Add a counter (available from the memory section on the side). Change its “Action on Overflow” property to be “Stay on Value.” When the counter reaches its maximum, it will not wraparound back to 0. Change its “Data Bits” to 10.
5. Connect the input pins to the counter as shown in the CookieWatcher design.
6. Draw a short wire coming out of the “Q” output of the counter. This is the counter's value. The output is, by default, a bundle. A bundle is several (in this case 10) wires all connected together. Using a bundle can save us space when working with multiple bits at once. Bundles are always black in color, and we are unable to see the values on the individual wires within the bundle.
7. Connect a splitter (from the “Base” category) to the end of the bundle. Change the splitter's BitWidthIn to 10 (because the counter's output is 10-bits). Change the splitter's BitWidthOut to be 10. This will split the 10-bit input into 10 pieces. A 10-bit input split 10 ways will result in 10 1-bit outputs.
8. Connect the 10 outputs from the splitter to AND gates as shown. Connect the AND gates as shown.
9. Add 2 output pins and give them the names as shown. The CookieWatcher's output is whether the batch of cookies it is watching is done (1 of the output pins states this) as well as whether that batch has been burnt (the other output pin states this).
10. Before moving on, think about the circuit you've just built. Why are the outputs from counter connected up in the way that they are? What do you think will happen as the counter changes? When will the outputs of the cookie watcher become true? False?

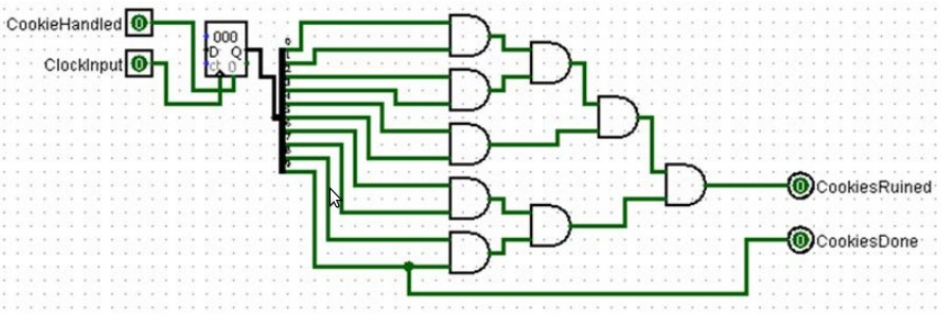

Illustration 2: The CookieWatcher subcircuit. One input is whether that batch of cookies was handled. The other input is the clock signal. The inputs are fed into a counter. The counter's value is used to determine whether the cookies are done and whether they are ruined (overbaked).

You have now completed the CookieWatcher circuit. Now, we can connect multiple CookieWatchers to create an automated cookie baking notification system capable of watching any number of baking cookies. Switch to the "main" circuit, which should currently be blank. To do this, double click on the word "main" (which should be above the words "CookieWatcher" on the left hand pane).

1. Add a clock component (available in the "Wiring" category).
2. Add 6 buttons (available from the "Input/Output" category).
3. Add 6 CookieWatcher subcircuits. To insert one subcircuit, single-click on the "CookieWatcher" entry in the side pane. Then, click in the main circuit to place it somewhere. It will appear as a small box. Do this 6 times.
4. With the arrow tool active, hover over the four dots on a CookieWatcher circuit. Notice that the left hand ones are blue, meaning that they are unconnected, expecting an input. The right hand ones are dull green right now, meaning that they are outputting a 0. We call these dots, pins. Hovering over a pin tells you the purpose of the pin, assuming you gave the subcircuit's pin a label. For example, hovering over the top pin on the right hand side of a CookieWatcher should show that the pin is the CookiesRuined pin.
5. Add 8 LEDs (under "Input/Output"). Connect them as shown in the figure.

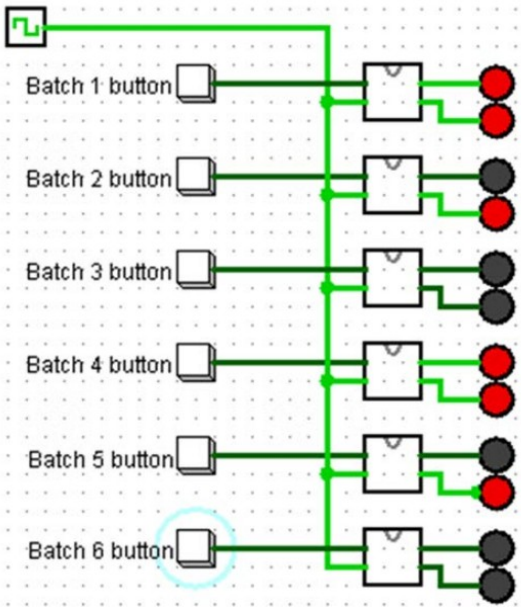

Illustration 3: Six cookie watchers. The second and fifth batches are ready to be taken out of the oven. The first and fourth batches are burnt. The third and sixth batches are not ready to be taken from the oven.

Your circuit should now be done. To test it, we will force the clock to tick, thus updating the circuit state. We can **poke** the clock to have it switch from low (0) to high (1) and vice versa. This lets us step through time, one cycle at a time.

To really test this circuit though, let's force the clock to update automatically at some fixed frequency. Under the “Simulate” menu, go to “Ticks Frequency” and select “128 Hz” (128 Hertz, 128 times a second).

Now, under “Simulate” choose “Ticks Enabled” (keyboard shortcut Ctrl-K). automatically tick away, advancing the circuit state.

The clock is connected to each CookieWatcher, and the CookieWatcher circuit will, eventually, turn on the lights. At 128Hz, the bottom light should come on in about 4 seconds. The bottom light turning on informs the employee that a batch of cookies are done. The employee would then take out the cookies, put in a new batch, and press (i.e., poke) the batch's button to reset the timer.

If the employee waits too long, the top light will come on. This lets the employee know that the cookies have been burnt. The employee should then put in a new batch, throw out the old batch, and press the reset button. At a clockrate of 128Hz, the top light will come on after 8 seconds.

Make sure that your circuit is working as designed. You should understand what each of the components are doing and why they are doing what they do.

Ask yourself the following (you do not have to submit these answers):

- When exactly does the CookiesDone light turn on?
- When exactly does the CookiesRuined light turn on?
- When the user pokes a button, what happens? How does the counter respond?

Save your Logisim circuit file as lab9part3.circ.