

## CSC211 Laboratory 5

### Background Information

Logisim is “a graphical tool for designing and simulating logic circuits,” written for use in educational settings and made freely available under the GPL license. Its home page is here:

<http://ozark.hendrix.edu/~burch/logisim/>

To see the GPL license, look here: <http://ozark.hendrix.edu/~burch/logisim/gpl.html>

In the next set of (2 or 3) labs, we will use Logisim to create circuits that are larger than would be practical on the protoboards.

### Laboratory Exercises

1. To get you up and running with Logisim:
  - Copy the Logisim jar file into your account like so: `~coahranm/share/csc211/logisim-2.3.1.jar`
  - Run Logisim with the following command, issued in the directory where you placed the jar file:  
`java -jar logisim-2.3.1.jar &`
  - In a web browser, visit this web page:  
<http://ozark.hendrix.edu/~burch/logisim/docs/2.3.0/guide/tutorial/index.html>
  - Follow the on-line tutorial through **Step 4: Testing your circuit**.
  - Continue with the *Users Guide* through the sections “Libraries and attributes” and “Subcircuits.” (There is plenty more to read later if you are interested, but this should be sufficient for now.)

2. As you probably noticed when looking at the Users’ Guide, there is a D flip-flop under *Memory* in Logisim’s Explorer Pane. You will also find a Clock tool under *Base*.

Create a small circuit that includes just these two tools, so you can experiment with their use. Note that hovering the mouse over a connection point on the flip-flop will bring up a tooltip describing its purpose. Note that you can “poke” the clock to cause it to change state, just as you can the input signal tool.

When you are ready for the clock to become automatic, select *Simulate* -> *Ticks enabled* from the Logisim menu. I suggest that you also select *Simulate* -> *Tick frequency* and set it to 1 Hz.

Experiment with your circuit to convince yourself that it works correctly. Recall that the flip-flop should only change state on the rising clock edge – regardless of when during the clock cycle the input value is changed.

3. Enhance your circuit from the previous exercise to create a 4-bit right-shift register. For this you will connect four D flip-flops together, such that the output from one feeds the input of the next one to its right. You should also have a single input that you can control manually to feed bits into the leftmost flip-flop. Experiment with this device to be sure it works correctly.

Next, expand your 4-bit shift register to an 8-bit one. You should be able to do this relatively painlessly, using copy and paste (once).

4. Start a new circuit, which will be a 4-bit binary counter. Recall that in class we drew this circuit using T flip-flops triggered on the rising clock edge, while the Stallings text draws the circuit using JK flip-flops triggered on the falling edge. You may use either design you wish. Both flip-flop types are available in Logisim under *Memory*.

5. *For this with extra time:* Design an enhancement to your shift register that allows it to behave either as a shift register, or a normal register (such that all bits are loaded from input sources in parallel). Your circuit should have a control line called  $\overline{shift/load}$  that controls the mode of the register: when the line is low, the device behaves like a shift register; when it is high, the device loads all bits in parallel.