

# ECE/CS 250 – Prof. Bletsch

## Recitation #4

### Logic Design with Logisim Evolution

---

**Objective:** In this recitation, you will learn how to design digital logic and use Logisim Evolution for the design and simulation of digital circuits.

Complete as much of this as you can during recitation. If you run out of time, please complete the rest at home.

#### 1. Download Logisim Evolution and work through tutorial

In this course, you will design and test logic circuits with Logisim Evolution. Please download Logisim Evolution version **2.15 (not a newer one!)** and work through (at least part of) the tutorial to help you understand how to use this tool (accessible via Help menu, Tutorial).

Download link: <https://github.com/logisim-evolution/logisim-evolution/releases/tag/v2.15.0>

This is a plain .jar file – no installer. Double-clicking it should run it with your installed Java. If not, be sure you have Java runtime 1.8 or similar (e.g. [this one](#)). If on Mac, see *Appendix A: Getting Logisim Evolution v2.15.0 on Mac* at the end of this document.

NOTE: Logisim Evolution is also installed in the Docker environment.

#### 2. Design a small combinational circuit

In Logisim, make and simulate a circuit that implements the following truth table (3 inputs, 1 output). Just do a simple sum-of-products – no need to optimize. Use only basic logic gates (NOT, AND, OR, NAND, NOR, XOR) from the Logisim Evolution library. You must simulate your circuit to test whether it behaves as expected in all cases.

in1	in2	in3	OUTPUT
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

### 3. Design a simple counter

In a new file, make a new subcircuit called “my\_counter”. Using an Adder, a Register, a Clock, and a Constant, design a sequential circuit that counts up by one every clock cycle. Individual pins and wires in Logisim Evolution can be set to contain multiple logical bits. In this case, set the number of data bits in the Adder, Register, and Constant to 8. Set both the simulation and ticks to enabled – your circuit should begin counting on its own.

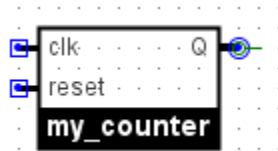
Modify the Constant such that your circuit counts down instead of up. (HINT: Remember two’s complement.)

### 4. Hierarchical design

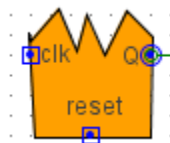
We’re going to make your counter a standalone component we can add to other circuits. Make the following changes:

- Remove the clock and replace it with a 1-bit input pin. Name this pin “clk”.
- Connect a 1-bit input pin to the reset (“R”) port of the register. Name this pin “reset”.
- Connect an 8-bit output pin to the Q port of the register. Name this pin “Q”.

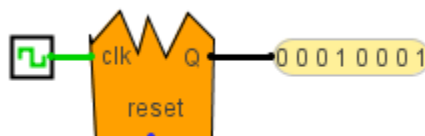
Look at the representation view of your counter circuit. It should look something like this:



This is the symbol that will represent this circuit if it’s placed as a piece in a larger design. Knowing this and being able to modify its appearance will make it easier to keep track of things when you build something large like a CPU. Move the pins around and give it new overall shape and background color. For example, you can make it orange and give it rad spikes:




Lastly, switch to the “main” circuit and place one of your new my\_counter objects. Wire it to a clock and probe and make sure it still counts correctly:



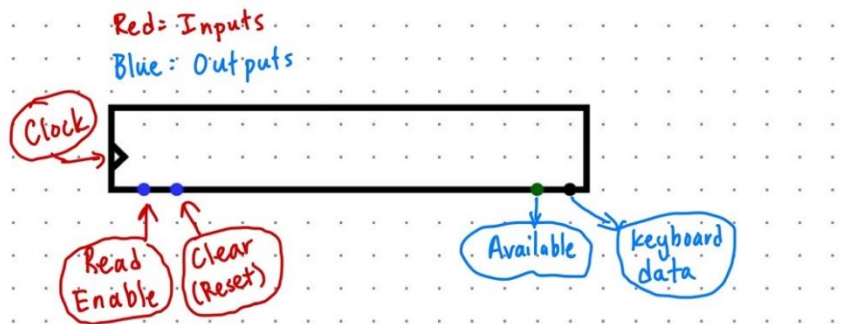
## 5. Working with a Clock and Keyboard and TTY Display

Logisim has components to simulate IO devices; let's practice them. Make a circuit that uses a **clock**, **keyboard**, and **TTY** display component to take user input and display the letters typed on the TTY display. The keyboard should be falling-edge triggered and the TTY display should be rising-edge triggered. The TTY display should have 13 rows and 80 columns. You can change the properties of each component by clicking on them and selecting the properties on the bottom-left side of Logisim. For some of the inputs to the keyboard and/or TTY display, you will need to connect an input pin.

Once you have built the circuit, you can test to see if input you enter in the keyboard shows up on the TTY Display.

1. Reset the simulation (Simulate menu -> Reset simulation, or CMD+R/Ctrl+R)
2. Use the poke tool  to click on the keyboard component
3. Type letters into the keyboard; they will fill the keyboard's *buffer* (a queue of keys to output as 7-bit signals, one character at a time)
4. Poke the clock and watch the data from the keyboard disappear and reappear on the TTY display

Here are the inputs and outputs of the Keyboard:



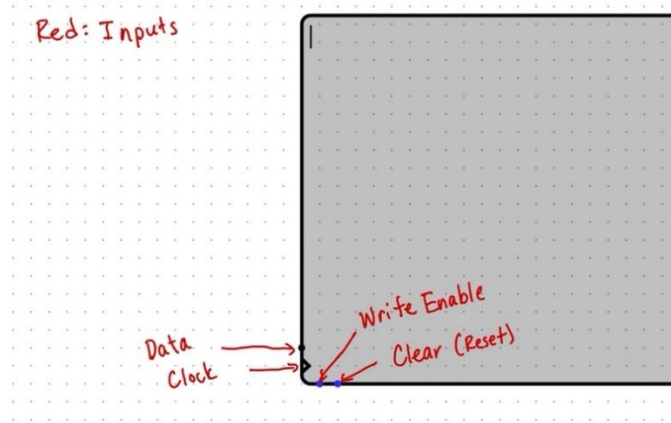
### Inputs:

- **Clock:** 1 bit input – connected to the clock
- **Read Enable:** 1 bit input – tells the keyboard to read a letter from the keyboard
- **Clear:** 1 bit input – clears the keyboard

### Outputs:

- **Available:** 1 bit output – ON when there is data in the keyboard
- **Keyboard Data:** 7 bit output – Contains the ascii value of the letter read from the keyboard

Here are the input and outputs of the TTY Display:



### Inputs:

- **Clock:** 1 bit input – connected to the clock
- **Read Enable:** 1 bit input – tells the keyboard to write a letter to TTY Display
- **Clear:** 1 bit input – clears the TTY Display
- **Data:** 7 bit input – Contains the ascii value of the letter to be written to the TTY Display

An example solution will be shown in Recitation and is linked on the course site.

## 6. A little bit of logic optimization

Time permitting, optimize the logic from Task #2 and implement the optimized circuit.

### ALL DONE?

Nice! Don't head out, though. Work on the current homework and talk to the TAs for help. You can leave if your homework tester shows all passing and you've turned in all the written & code materials.

## Appendix A: Getting Logisim Evolution v2.15.0 on Mac

1. Visit this link to download and install Java (important!!)  
<https://java.com/en/download/>

2. Check if you have Java installed by running this command in Terminal:  
`java -version`

You should see an output similar to below:

```
java version "1.8.0_291"  
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)  
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed  
mode)
```

3. Visit [this link to download logisim-evolution.jar](#). Click “logisim-evolution.jar” to download the file.
4. Using Terminal, cd to the directory where the jar file downloaded to (probably downloads folder)

```
cd Downloads
```

5. Run this command in Terminal to open the jar file with disk access privileges (Important step! If you click the jar file manually, Logisim will not have access to your Mac’s file system)  
`java -jar logisim-evolution.jar &`

NOTE: The ampersand puts this into the *background*, allowing you to continue using the terminal for other tasks.

Note: If you need a new Terminal window, right-click the Terminal icon in the dock and select “New Window” to open another.

