ECE/CS 250 – Summer 2016 – Prof. Bletsch
Recitation #4
Logic Design with Logisim Evolution

Objective: In this recitation, you will learn how to design digital logic and use Logisim 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 and work through tutorial

In this course, you will design and test logic circuits with Logisim Evolution. Please download Logisim Evolution and work through (at least part of) the tutorial to help you understand how to use this tool (accessible via Help menu, Tutorial).

https://github.com/reds-heig/logisim-evolution

(The download link is under "How to install logisim-evolution"; it's a JAR file that you can run directly.)

NOTE: Logisim Evolution is a fork of the classic tool, Logisim. This is the first time we're using the updated tool – please report to the instructor or TAs if you encounter any problems. Logisim Evolution should fix several annoying bugs found in the original, however, one of the big changes was to add integration with real-world hardware design tools. This means there's a bunch of features you can (and should) ignore for now, including:

- Anything labeled "VHDL", "Verilog", or "FPGA".
- The "HDL-IP", "TCL", and "BFH" tool categories.
- The "Registers" tab of the properties palette
- The Chronogram and Assembly options.

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 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	1
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 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.

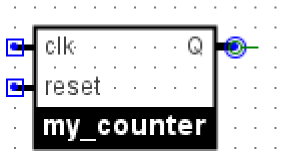
Modify the Constant such that your circuit counts down instead of up. (HINT: Remember two’s complement.)

4. Hierarchical design

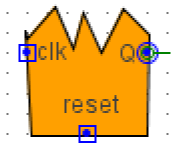
We’re going to make your counter a standalone component we can add to other circuits. Make the following changes:

- 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”.

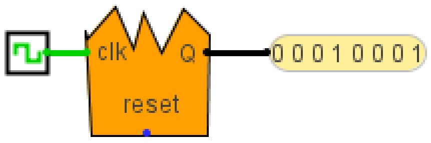
Look at the representation view of your counter circuit. It should look something like this:



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. A little bit of logic optimization

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