

# CSC258 - Lab 1

## Building Circuits using Logisim Evolution

### 1 Learning Objectives

The purpose of this lab is to illustrate the process of building logic circuits by using Logisim-Evolution. Although circuits are typically built with more powerful tools and libraries in industry, there still needs to be an understanding of how gates are connected at a lower level to implement simple logic functions. In future labs, you will be designing more complicated circuits, therefore this lab is designed for you to gain familiarity with building simple circuits and using the tool Logisim-Evolution.

### 2 Marking Scheme

Each lab is worth 6% of your final grade, but you will be graded out of 8 marks for this lab, as follows.

- Prelab: 2 marks
- Part I (in-lab): 2 marks
- Part II (in-lab): 2 marks
- Part III (in-lab): 2 marks




#### 2.1 The Prelab Exercises

All of your lab exercises involve a set of "pre-lab" exercises that are necessary preparation for the lab itself. Unlike other courses, the lab time is meant for the demonstration of your designs, which means that your designs must be complete before attending the lab.

**BEFORE each lab**, you must read through the sections below and complete all the prelab preparations which are clearly marked in red. These prelab exercises are meant to be completed individually and submitted electronically on Markus on the evening before the lab.

During your lab, use your pre-lab designs to help you complete all the required in-lab actions. The more care you put into your pre-lab designs, the faster you will complete your lab.

### 3 Preparation Before the Lab

The pre-lab exercises for this lab involve the design of digital circuits using AND, OR and NOT gates in order to create a specified behaviour. Your task is to design all the circuits in both Parts I and II using **only** AND , **OR** , and **NOT**  gates.

For example, consider the following function:

$$f = ab + (c + b)$$

This notation is shorthand for the following logical expression:

$$f = a \text{ AND } b \text{ OR } (c \text{ OR } b)$$

The resulting schematic will look like this in Logisim-Evolution:

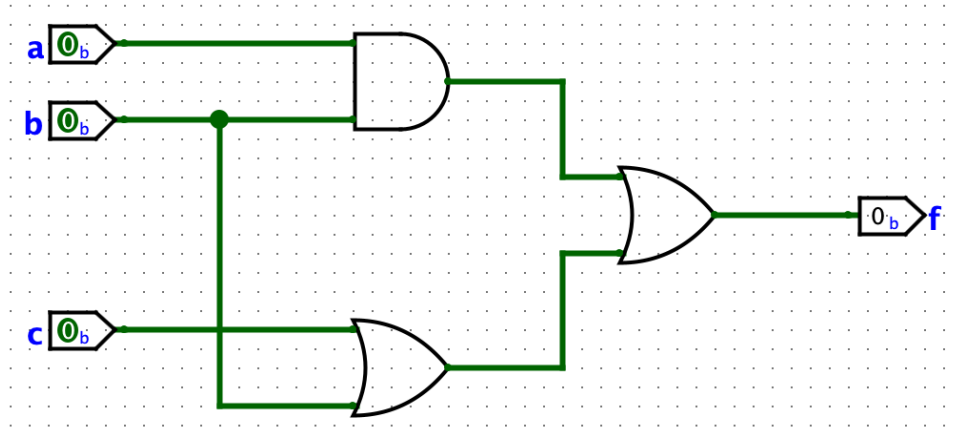


Figure 1: Schematic of the logical expression

In each of the parts below, show all of the steps required to go from the specification provided to the final circuit, including: assigning names to input and output wires, deriving a truth table, the logic function, and then a schematic picture of the final circuit.

**Important:** For this lab, you are allowed to use only the following gates: NOT gates, AND gates and OR gates.

## 4 Part I

The **multiplexer** is a device that selects one of multiple inputs to drive the output value. The following Boolean function is the logical expression for a 2-to-1 multiplexer.

$$f = xs' + ys$$

Note that  $s'$  is the notation for “s inverted” or “NOT s”. As we can see, when the select signal  $s$  is 0, the output has the same value as the input signal  $x$ . When  $s$  is a 1, the  $y$  signal will appear at the output. This is an extremely useful circuit with multiple applications in the datapath of a CPU, which you will be implementing in one of the future labs.

Perform the following steps:

1. Draw the gate diagram that implements this 2-to-1 multiplexer design using the AND, OR and NOT gates as specified above and include this diagram in your prelab report. Have this diagram ready to show your TA as part of your demo to verify the correctness of the design that you are implementing. **(PRELAB)**
2. Write out the truth table for this design. Include this truth table in your prelab report and have it ready to show to the TA as well as part of your demo. **(PRELAB)**

## 5 Part II

Build the gate-level implementation for the following Boolean expression:

$$f = (a + b)' + cb'$$

Perform the following steps:


1. Draw the gate diagram for the function shown above using the AND, OR and NOT gates specified in the Lab Preparation section. Include this diagram in your prelab report and have it ready to show to your TA as part of your demo, to verify that your design was correct before you implemented it. **(PRELAB)**

2. Write out the truth table for your design and show it to the TA as another part of the prelab. **(PRELAB)**
3. Is there a cheaper implementation for your design, assuming you are still limited to using the same three chip types? If yes, explain by rewriting the boolean function. For this question we consider a given implementation cheaper if it uses fewer gates. **(PRELAB)**

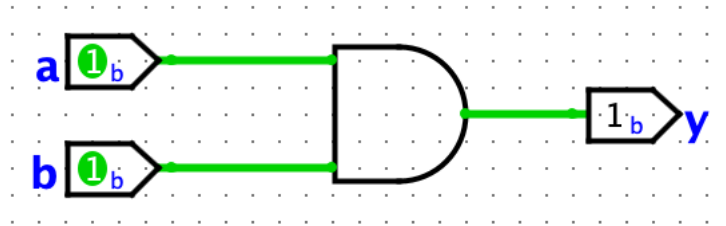
## 6 Part III

In this section, you will start on your first Logisim-Evolution project. You will design Part I and Part II and verify its functionality with the truth tables from your prelab.

Perform the following steps for the logic expression given in Part I and II and test the circuits on the DE1-SoC board.

1. Follow the instructions in our Logisim tutorial to download `logisim-evolution-3.3.0-all.jar` from Quercus / Modules / Logisim-Evolution v3.3.0 JAR File.
2. Make sure to read through Tutorial 1 for general instructions of Logisim-Evolution.
3. Launch Logisim-Evolution (you will need at least Java SE 9 installed; Recommended version: AdoptOpenJDK 11) and complete the Logisim-Evolution *Beginner's Tutorial* step 0-4. The tutorials can be found in *Help > Tutorial* when you launch the tool and *Beginner's Tutorial* can be found on the left of the *Tutorial* window.
4. Change the output file to Verilog in Logisim-Evolution: *Window > Preferences > FPGA Commander Settings > Hardware description language > select Verilog*
5. Now it's time to design your circuit
  - (a) Drag and drop the gates you need from the tool bar onto the canvas. Add labels to each gate by double-clicking it. For future labs, you can also change the number of inputs for each gate. To do that, click on each gate, then in the *Properties* on the left side, you will be able to change the number in *Number Of Inputs*.
  - (b) For this lab, you can use the default inputs and outputs from the tool bar. Make sure to add labels to each input and output as you go. For future labs different types of input/output can be found by double-clicking on *Input/Output* on the left side. *Button* and *Dip switch* are usually used for inputs and *LEDs*, *HEX Display* and *7-Segment Display* are usually used for outputs.
  - (c) Connect the elements together with wires. Be aware of the color of the wires. The color indicates the status of the wires. For this lab, you should only see light or dark green, other colors would mean that your wires are not properly connected.
  - (d) Have the circuits from Parts I and II implemented in Logisim-Evolution and ready to demo to your TA **(PRELAB)**.
6. To test your circuit in Logisim-Evolution
  - (a) One option for testing is to use *Poke*() on the tool bar to change the states of the input and see how the changes affect the output of the circuit on the canvas.
  - (b) To test all the rows in your truth table, you need to create a text test vector file. Here's an example for testing an AND gate.

The AND gate looks like the following:  $a$  and  $b$  are inputs and  $y$  is the output.



The truth table for the AND gate:

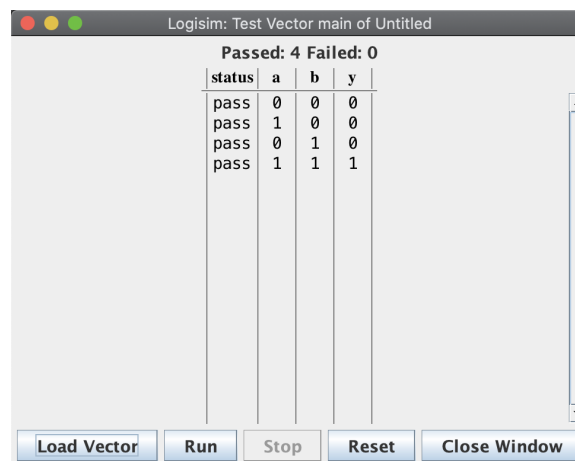
a	b	y
0	0	0
0	1	0
1	0	0
1	1	0

The corresponding test vector file:

```
test.txt
1  # Test Vector for an AND gate
2  a b y
3  0 0 0
4  1 0 0
5  0 1 0
6  1 1 1
```

For the test vector file, it should be stored as a *.txt* file. All the comments start with #. The first row should be a list of the labels for all input/output separated by a space. The following rows are the truth values for the inputs and the expected value for the output, also separated by a space.

Once you have the test file ready, in Logisim-Evolution select the circuit you want to test from the top of the components (it should be *main* if you have not added any new circuits), go to *Simulate > Test Vector...* Then a window would pop up. The very top of the window indicates the name of the circuit you are testing. Then click on *Load Vector*, browse to your test *.txt* file. And you will see the test result for each row of your truth table:



Make sure that the test results match your expectation. You will need to demo this part to your TA (PRELAB).